Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

Instytut Informatyki

Studia Podyplomowe
Big Data - przetwarzanie i analiza dużych zbiorów danych

PRACA KOŃCOWA

Aleksandra Maria Biernat

Analiza Sentymentu oraz Ranking Ocen z użyciem narzędzi Big Data: Hadoop, Hive, Spark oraz Deep Learning

Opiekun pracy
mgr inż. Patryk Pilarski

Warszawa, 2021

STRESZCZENIE

Głównym celem tej pracy było stworzenie modelu analizy sentymentu w kontekście Big Data, w którym, bazując na opinii danego produktu dostępnego na Kindle, przewidywana jest ogólna ocena nadana przez konsumenta na podstawie opinii w formie pisemnej / tekstowej. Dla wyżej wspomnianego założenia, zbudowanych i porównanych zostało kilka modeli uczenia maszynowego. Przeanalizowane modele to: wieloklasowa regresja logistyczna i tzw. algorytm gradient boosted tree w Apache Spark Scala w ekosystemie Apache Hadoop oraz głębokie sieci neuronowe korzystające z bibliotek tensorflow i keras, wytrenowane z wykorzystaniem przyspieszenia obliczeń używając GPU dostępnego na Google Colab. Inną cechą, która została rozważona była 'category' zawierająca kategorię, do której przynależy dany tytuł. Dodatkowo zbudowany został system hierarchizujący książki pod względem najwyższej oceny, również w Apache Hadoop – z wykorzystaniem Apache Hive. To ostatnie jest szczególnie adekwatne w kontekście dziedziny e-commerce, ponieważ potencjalny klient często chce się dowiedzieć jakie są najlepsze produkty na podstawie doświadczeń innych użytkowników, lecz bazując nie tylko wyłącznie albo na zwykłej średniej ocenie albo na popularności / liczbie ocen, które często okazują się niewystarczające i prowadzą wyłącznie do niepełnych wniosków, ale także na mierze, która zawiera obydwa te składowe na raz. Takie podejście jest lepszym odzwierciedleniem rzeczywistej atrakcyjności i de facto „średnią oceną" danego produktu.

Dla poparcia wniosków, zostały policzone i zaprezentowane adekwatne metryki oceny modeli, lub też przykłady wyników obrazujące przydatność zaproponowanych metod. Na ich podstawie, zostały wybrane „najlepsze" modele i metody oraz zaproponowane jako ostateczne rozwiązanie. Zostały również omówione możliwe zastosowania i korzyści dla środowiska biznesowego płynące z użycia zaprezentowanych modeli i metod.

Słowa kluczowe: Uczenie maszynowe, Apache Hadoop, Apache Hive, Ambari, Apache Hive, Apache Spark, Głębokie ieci neuronowe, Przetwarzanie języka naturalnego, Systemy hierarchizujące

---

Sentiment Analysis and Ranking of Reviews using Big Data tools and concepts: Apache Hadoop, Apache Hive, Apache Spark and Deep Learning

The main goal of this project was a task of sentiment analysis in the context of big data, where based on a text review of a certain Kindle product, an overall mark given by a consumer is being predicted. For the purpose of this analysis, several models have been built and compared in order to ultimately choose two final ones. In this project, we explored Multinomial Logistic Regression and Gradient Boosted Tree - both done in Spark Scala in the Apache Hadoop ecosystem as well as Deep Learning models which were trained making use of the keras and tensorflow libraries and GPU acceleration functionality available on Google Colab. Another explanatory feature investigated in the models was 'category' feature with a category / genre that a certain Kindle book belongs to. Moreover, a ranking system has been built also using the Apache Hadoop ecosystem and its component – Apache Hive. The latter is particularly useful in the e-commerce world, where potential clients would want to learn what the best products are, but not only taking into account either the mean review or popularity measure of a certain book, which have proved to be insufficient and often lead to only incomplete conclusions, but

instead use a mixed measure of the two. This provides a better reflection as well as a fuller picture of an overall attractiveness and review of a certain book.

For all of the aforementioned tasks, relevant evaluation metrics have been obtained and compared or snapshots of results depicting the usefulness of a proposed approach were presented. Based on those, the best models and methods have been chosen and proposed as the final product. The possible applications and benefits of the presented models and methods have also been discussed and stated.

Keywords: Machine Learning, Apache Hadoop, Apache Hive, Ambari, Apache Hive, Apache Spark, Deep Learning, Natural Language Processing, Ranking Systems, Recommendation Systems, E-commerce

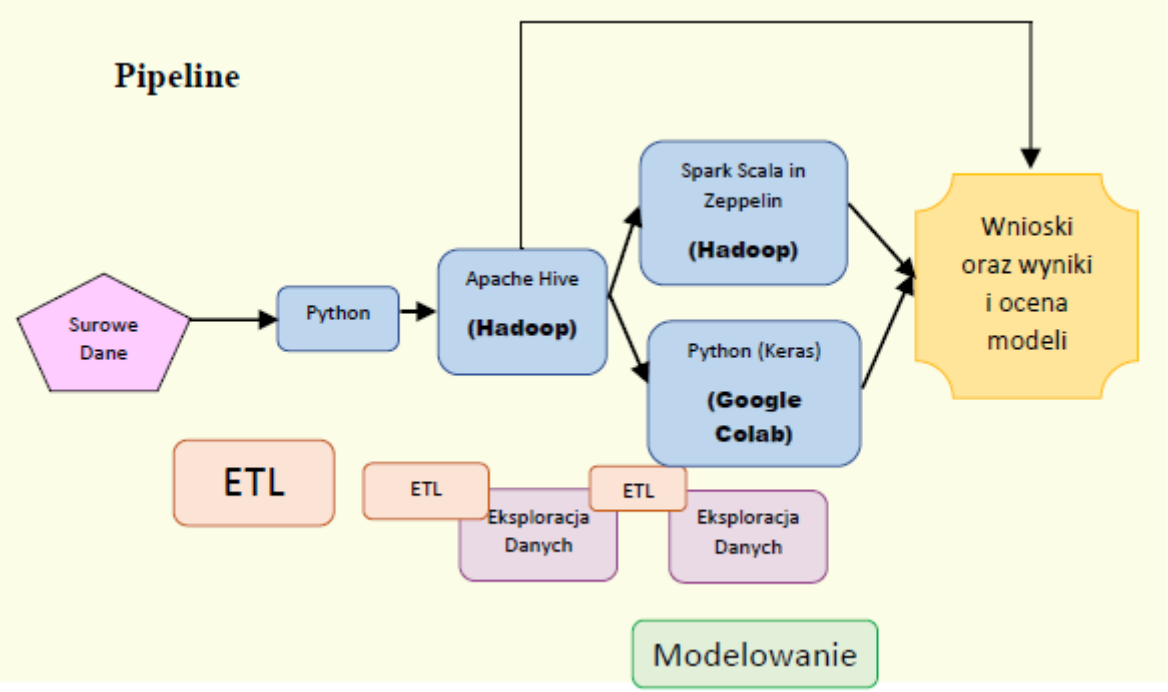
SPIS TRESCI

WSTĘP

ROZDZIAŁ I: Pipeline

Poglądowy potok projektu został przedstawiony na grafie Rysunek 1 Poglądowy potok (pipeline) projektu. Jest to graficzna reprezentacja głównych składowych procesu w tym projekcie, przy czym mniejsze zadania takie jak np. czyszczenie danych pod kątem konkretnego modelu były wykonywane na odpowiadających im etapach. Ponadto należy zauważyć, że proces eksploracji, czyszczenia jak i modelowania danych często nie odbywa się w sposób jednorazowy, po czym przenosimy się do kolejnego kroku. Często są to procesy iteracyjne zależące od wniosków jakie nasuną się i zależności, które uwidaczniają się na różnych etapach budowania procesu.

Rysunek 1 Poglądowy potok (pipeline) projektu



ROZDZIAŁ II: Wybór Technologii

Python – Google Colab

Python został wybrany, ponieważ jest obecnie jednym z najbardziej uniwersalnych, powszechnych i elastycznych języków. Ma on również szerokie zastosowanie w uczeniu maszynowym, dzięki licznym pakietom i bibliotekom zawierającym po części gotowe modele. W kontekście tego projektu, dzięki integracji z pakietami takimi jak keras i tensorflow, wydaje się on być najlepszym wyborem do zbudowania modeli głębokich sieci neuronowych. Python jest również jednym z języków, których można używać na Google Colab – otwartym środowisku z dostępem poprzez przeglądarkę internetową stworzonym i zarządzanym przez Google. To z kolei czyni Google Colab bardzo wygodnym narzędziem, zwłaszcza, że moce obliczeniowe – a w szczególności możliwość obliczeń przy użyciu GPU i TPU – są nierzadko większe niż moce domowych komputerów (w tym mojego). Ze względu na ograniczoną pojemność RAM, prezentowane w tym projekcie modele głębokich sieci neuronowych nie mogły być wytrenowane na moim lokalnym komputerze domowym.

Apache Hadoop

Ekosystem Apache Hadoop jest z kolei jednym z częściej wykorzystywanych narzędzi do przetwarzania dużych zbiorów danych. Jest on o tyle przydatny, że jest zintegrowany z wieloma różnymi technologiami, takimi jak np. Hive (Tez), Spark SQL - do przetwarzania danych; Kafka, Spark Streaming - do przetwarzania danych strumieniowo; Spark, Zeppelin – przydatne w procesach uczenia maszynowego, automatyzacji i przetwarzania danych oraz wiele innych. Poprzez HDFS oferuje on również system przechowywania plików, jak również rozwiązania bazodanowe takie jak HBase. Poprzez to daje on ogromne możliwości - znaczna część tego projektu – poprzez wykorzystanie Hive (interfejsu do Hadoop), Spark i Zeppelin została wykonana właśnie w ekosystemie Apache Hadoop.

Ambari – Hortonworks Data Platform

Dystrybucja Apache Hadoop stworzona przez Hortonworks i utrzymywana przez Cloudera. Jest to otwarto-źródłowe środowisko (ang. open source), co było głównym powodem wyboru właśnie tej dystrybucji Apache Hadoop. Proces instalacji i wersja użyta są takie same jak wyjaśnione w MOOC Udemy „The Ultimate Hands-On Hadoop: Tame your Big Data!" (Kane, brak daty)

Apache Hive

Postrzegany jako interfejs do Apache Hadoop. Składnią przypomina SQL. Podobnie jak środowiska służące do pisania w SQL, jest bardzo wygodny do zwykłego przetwarzania danych takiego jak tworzenie zapytań, agregacje, filtrowanie, czyszczenie danych, co było główną motywacją do użycia go w projekcie.

Apache Spark - Scala - Zeppelin

Apache Spark jest bardzo szybkim silnikiem stworzonym do przetwarzania wielkoskalowych danych z użyciem przetwarzania rozproszonego na różne klastry[1]. Jest obecnie bardzo popularny i dzięki temu dostęp do wiedzy w zakresie Sparka jest dosyć łatwy. Spark jest zintegrowany z innymi językami takimi jak Java, Scala, czy Python (tzw. Pyspark) a także, podobnie do Python'a jest bardzo elastyczny - daje szeroki wachlarz możliwości i celów do jakich może zostać użyty takich jak uczenie maszynowe (Mllib – bazujący na RDD, podstawowej strukturze / obiekcie Sparka oraz Spark ML bazujący na DataFrame, strukturze podobnej do idei DataFrame w bibliotece pandas w Python'ie), analiza grafów (GraphX), przetwarzanie strumieniowe (Spark Streaming), czy bardziej „standardowe" przetwarzanie danych podobne do takiego jakiego dokonuje się zazwyczaj za pomocą zapytań w SQL (Spark SQL i bardziej podstawowa, standardowa składnia Sparka). Jest również szybszy niż MapReduce, czyli pierwotna koncepcja, na jakiej opierał się Hadoop. Scala jest jednak najbardziej natywnym środowiskiem dla Sparka i dlatego część uczenia maszynowego w Sparku została wykonana w Sparku – Scala. Operacje napisane w Scali są z reguły szybsze niż w Pyspark'u, zużywa on wtedy mniej zasobów. Scala jako język / środowisko do programowania w Sparku jest również bardziej dostosowana do przetwarzania rozproszonego i lepsza pod kątem produkcyjnym (Kane, 2017). Zeppelin to z kolei środowisko, w którym można pisać kod w Sparku Scala umożliwiające zapisywanie wyników i wizualizacji. Powyższe argumenty są motywacją dla której użyty został Spark Scala w Zeppelinie w ekosystemie Apache Hadoop.

CZĘŚĆ GŁÓWNA

ROZDZIAŁ I: Proces ETL

PODROZDZIAŁ 1: Zbieranie surowych danych (EXTRACT)

W przypadku tego projektu, proces pozyskiwania danych jest mało złożony dlatego też odpowiadające czynności konieczne do ich pozyskania są mało wymagające. Dane pochodzą z jednego źródła (strony internetowej) i wszystkie dane zawarte są jedynie w dwóch plikach.

[1] ang. cluster – czyli kilka komputerów połączonych razem

Analizowane dane przedstawiają oceny e-booków i produktów dostępnych na Kindle ze sklepu Amazon (Jianmo Ni, 2019). Dane są w formacie JSON i zawarte są w dwóch plikach: Kindle_Store.json (Ni, 2018) i meta_Kindle_Store.json (Ni, 2018) o rozmiarach odpowiednio 3.52 GB i 462 MB. Wg definicji, aby dane zostały uznane za Big Data (Szmit, brak daty), muszą one spełnić warunek „4V": **Volume** (duża liczba danych), **Velocity** (dane szybko przyrastają), **Variety** (duża różnorodność danych) oraz **Value** (duża wartości danych). Jeśli chodzi o obecne standardy Big Data, gdzie dane wykorzystywane przez różne firmy nierzadko mają rozmiar liczony w tera- bądź petabajtach, pliki z danymi analizowanymi w tym projekcie nie są ogromne, jednak aby stworzyć efektywne narzędzie do przetwarzania takich danych, techniki Big Data są co najmniej bardzo przydatne. Spełniają one również warunek „Velocity", ponieważ niemalże w sposób ciągły dodawane są nowe opinie produktów dostępnych na Kindle przez użytkowników. Dane te są również zróżnicowane, gdyż składają się z wielu schematów, które mogą być wykorzystywane w zależności od analizowanych hipotez i wniosków do jakich chcemy dojść przy dokonywaniu konkretnej analizy. Ponadto, poszczególne pola zawierają podschematy, często niejednorodne na przestrzeni różnych wierszy. Dane te są również wartościowe pod względem potencjalnych korzyści biznesowych wynikających z wniosków wyciągniętych z różnych analiz wykonywanych na tychże danych.

Warto zauważyć również, że dane są w formacie JSON, a konkretnie JSON Lines (zwany również newline-delimited JSON), w którym poszczególne wiersze są w formacie zagnieżdżonego słownika. Klucze odpowiadają polom tabeli, a wartości – wartościom. Taki format jest bardzo wygodny dla ustrukturyzowanych jak również semi-ustrukturyzowanych danych (jak w tym przypadku), które mogą być przetwarzane linijka po linijce (JSON Lines Org., brak daty), a jest to najwygodniejszy sposób dla tych danych na etapie ETL[2]. Można zauważyć podobieństwa w strukturze tych danych do modelu występującego w bazach dokumentowych np. Mongo DB jak i samym HBase, bazie danych w ekosystemie Hadoop. Tutaj poszczególne „rowkeys" sa oddzielone poprzez nowe linie, kolumny column family odpowiadają najbardziej zewnętrznym kluczom, a column qualifiers kluczom w zagnieżdżonych słownikach. Z powodu tej struktury, przetwarzanie tych danych w bazie SQL, gdzie jest zachowana struktura schematów nie byłoby możliwe bez wcześniejszej modyfikacji i pozbycia się struktury zagnieżdżenia. Format ten można z powodzeniem również wgrać do Hive, zachowując znaczenie struktury, która jest uzyskiwana poprzez efekt zagnieżdżenia słowników. W takiej formie (dokumentów JSON), każdy dokument stanowi niezależny byt.

Dane zawierają m.in. ocenę, ASIN (numer identyfikacyjny nadawany i wykorzystywany przez Amazon), treść oceny / opinii o produkcie, liczbę ocen, numer identyfikacyjny klienta, numer identyfikacyjny samej opinii oraz inne mniej ważne pola. Dane zostaną omówione w bardziej szczegółowy sposób w części *ROZDZIAŁ II: Eksploracja Danych; PODROZDZIAŁ 1: Dane Kindle* i *ROZDZIAŁ II: Eksploracja Danych*; *PODROZDZIAŁ 1: Dane Kindle i PODROZDZIAŁ 2:* .

PODROZDZIAŁ 2: Wstępne przetwarzanie danych (TRANSFORM)

Proces transformacji danych polegał głównie na doprowadzeniu ich do takiej formy, aby można było je załadować do HDFS w dystrybucji Hadoop Hortonworks Data Platform – Ambari, a nastepnie wgrac je do Hive. Motywy wyboru tych technologii zostaly opisane w sekcji *WSTĘP*; *ROZDZIAŁ II: Wybór Technologii*.

[2] ETL – Extract, Transform, Load, czyli proces obróbki surowych danych na wstępnym etapie ich przetwarzania.

W szczególności, głównymi celami bylo:

1. Podzielenie plików na mniejsze części (na wzór koncepcji shardingu)
   Konieczność tego zabiegu wynikała z limitu wielkości pliku jaki można jednorazowo wgrać do HDFS w Ambari.

2. Wstępne wyczyszczenie danych, aby dane i występujące w nich znaki specjalne były zgodne z wymaganiami Hive. - Niezbędne było usuniecie niektórych dwukropków, ponieważ są to znaki specjalne używane do identyfikowania komórek określających klucz-wartość przy definiowaniu tabeli w Hive. Plik meta_Kindle_Store.json wymagał dodatkowych zmian znaków specjalnych.

Z powodów opisanych wyżej (str. 6) zależało mi również na zachowaniu formatu JSON Lines. Powyższe punkty zostały osiągnięte za pomocą kodu napisanego w paradygmacie obiektowym (Marek Gągolewski, 2016) w Pythonie w środowisku Jupyter. Kod źródłowy dostępny jest pod adresem zawartym w rozdziale *DODATEK 1: Kod źródłowy generujący mniejsze pliki w języku Python*. W efekcie, Kindle_Store.json został podzielony na 29 plików, a meta_Kindle_Store.json na 5 plików.

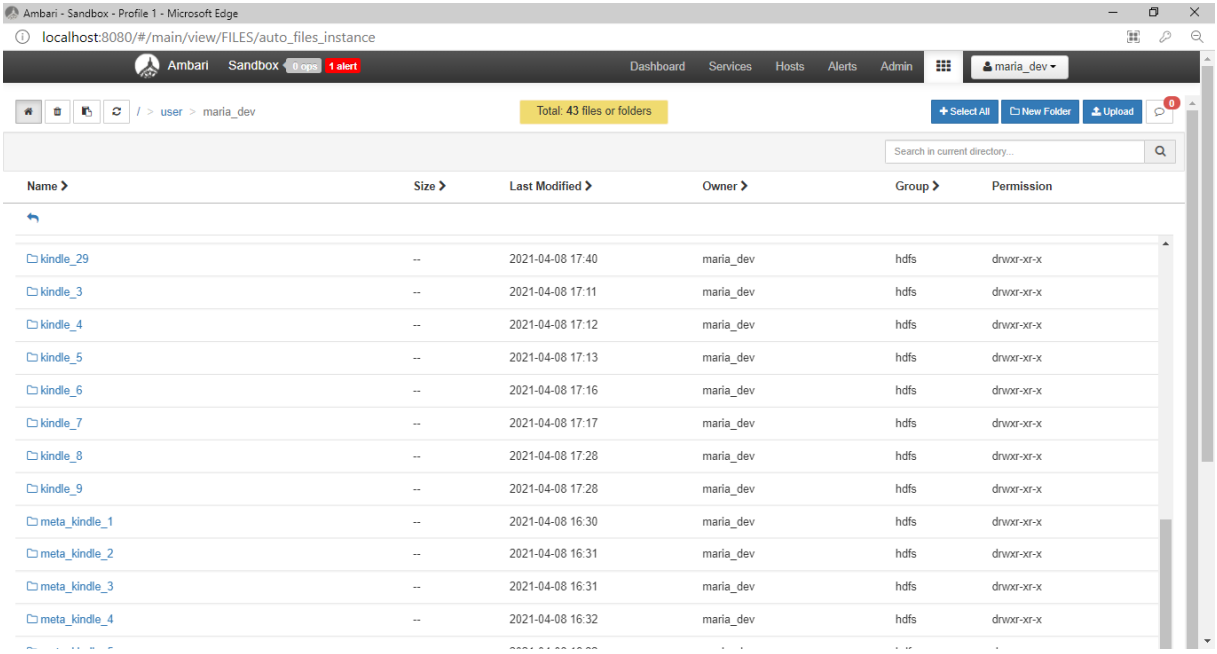
PODROZDZIAŁ 3: Załadowanie danych (LOAD)

Pierwszym krokiem koniecznym do wykonania, aby móc analizować dane w ekosystemie Apache Hadoop jest załadowanie danych do HDFS[3]. HDFS jest rozproszonym systemem plików, będącym częścią architektury Apache Hadoop (Szmit, brak daty) oraz pełniącym funkcje przechowywania plików. Pliki zostały wgrane w taki sposób, żeby każdy z wygenerowanych wyżej plików znajdował się w oddzielnym pliku jak zostało do pokazane na zrzucie ekranu Rysunek 2 Widok HDFS w Ambari z poziomu http. Jest to konieczne, aby móc stworzyć tabele w Hive na podstawie tych plików. Warto wspomnieć, że w tym przypadku pliki zostały wgrane z poziomu http:// manualnie. Możliwym usprawnieniem tej części procesu, np. gdyby potok miał być wprowadzony do potocznego środowiska produkcyjnego, byłoby nieznaczne zautomatyzowanie zapisywania plików (np. na github lub prywatnej stronie internetowej) i załadowywanie plików do HDFS z poziomu CLI[4].

[3] Hadoop Data File System

[4] Command Line Interface – poprzez terminal / konsolę; w przypadku Ambari połączenie do maszyny wirtualnej z dystrybucja Hadoop dokonuje się przez PuTTy

Rysunek 2 Widok HDFS w Ambari z poziomu http z pojedynczymi folderami z danymi kindle oraz meta kindle.



Następnie, na podstawie tak wgranych plików, zostały utworzone tabele w Apache Hive. Apache Hive jest oprogramowaniem, umożliwiającym czytanie, pisanie i ogólne zarzadzanie, za pomocą składni języka SQL, dużymi zbiorami danych przechowywanymi w rozproszonych systemach plików (np. HDFS) i bazach danych, które są zintegrowane z Hadoop (The Apache Software Foundation, brak daty), (Wikipedia, brak daty). Z tego powodu często rozumiany jest jako interfejs w formie SQL służący do tworzenia zapytań na tychże zbiorach danych (Szmit, brak daty). Jest on zatem bardzo dobrym wyborem do procesu czyszczenia i ujednolicania danych w celu łatwiejszej analizy[5] jak również do badania zależności miedzy danymi, eksploracji danych i formułowania wstępnych wniosków i analiz poprzez np. agregacje.

Wgranie danych do Apache Hive może wymagać wgrania dodatkowej biblioteki JSON SerDe jeśli dane w plikach źrodłowych mają niestandardową strukturę. Podstawowa biblioteka JSON SerDe jest już zazwyczaj dostępna. Dzięki tej bibliotece możliwe jest tzw. zdeserializowanie danych w formacie JSON. Jest to przekonwertowanie danych JSON tak, że jest możliwe późniejsze zserializowanie ich (zapisanie) w innym formacie np. Parquet. Hive JSON SerDe jest często używany do przeprocesowywania danych w formie „bloków" danych, w których każdy blok stanowi dane zakodowane w formacie JSON oddzielone między sobą znakiem nowej linii (Amazon AWS, brak daty), (JSON Lines Org., brak daty), czyli wspomniany w sekcji *PODROZDZIAŁ 2: Wstępne przetwarzanie danych (TRANSFORM)*, JSON Lines. Przykładowy kod używający JSON SerDe i tworzący tabelę bazując na danych zapisanych w HDFS został przedstawiony na rysunku Rysunek 3 poniżej. Cały kod, zapisujący wszystkie tabele oraz tworzący oddzielną bazę danych może zostać znaleziony w części *DODATEK 2: Wczytywanie danych z HDFSa do Apache Hive*. Kod ten został puszczony z poziomu CLI po uprzednim zalogowaniu się do Ambari poprzez PuTTy i port SSH oraz podłączeniu Hive przez CLI.

[5] ang. data wrangling

Rysunek 3 Kod DDL, który używa JSON SerDe i tworzy tabelę bazując na danych zapisanych w HDFS

```
Create external table projectdb.meta_kindle_1(
  category ARRAY <string>,
  tech1 string,
  description ARRAY <string>,
  fit string,
  title string,
  also_buy ARRAY <string>,
  image ARRAY <string>,
  tech2 string,
  brand string,
  feature ARRAY <string>,
  rank ARRAY <string>,
  also_view ARRAY <string>,
  details
struct<release_date:string,file_size:string,print_length:string,Page_Numbers_Source_ISBN:strin
g,Simultaneous_Device_Usage:string,publisher:string,publication_date:string,language:string,is
bn_10:string, isbn_13:string,asin:string,word_wise:string,lending:string>,
  main_cat string,
  similar_item string,
  date string,
  price string,
  asin string
)
  row format serde 'org.apache.hive.hcatalog.data.JsonSerDe'
  LOCATION '/user/maria_dev/meta_kindle_1/';
```

W najogólniejszym rozróżnieniu w Hive występują dwie kategorie danych: prymitywne (ang. primitive) i zespolone (ang. complex). Typy te definiują typ danych jaki znajduje się w kolumnie / polu w tabeli Hive. Jeśli chodzi o podstawowe typy danych, to logika pod ich spodem jest podobna do logiki typów danych w SQL. Zespolone kategorie dzielą się natomiast na cztery podkategorie: **Array**, **Map**, **Struct** i **Union**. **Array** jest uporządkowaną sekwencją elementów podobnego typu które mogą być poindeksowane używając liczb naturalnych (podobnie jak w innych językach np. Python). **Map** jest kolekcją par klucz-wartość, czyli też jest to struktura często używana w innych językach (słownik – Python (Marek Gągolewski, 2016), JSON – wiele języków, RDD – Spark (Kane, brak daty) itp.). **Struct** w Hive jest podobny do typu Struct w języku C. Jest to typ, który może zawierać w sobie zbiór innych pól zdefiniowanych nazwą, które to pola mogą być każdego podstawowego typu, jak np. string, integer, date, itp. **Union** znowu jest podobny do union w języku C. Jako że union nie jest jeszcze w pełni zaimplementowany w Hive jak również ten typ danych nie był użyty w tym projekcie, nie będzie on głębiej wyjaśniany (Data-Flair Trainings Blog, brak daty).

Następnym krokiem jest połączenie tak załadowanych danych już w Apache Hive. Kod użyty do wykonania tego zadania znajduje się w sekcji *DODATEK 3: Ponowne połączenie danych Kindle i meta_Kindle w pojedyncze pliki w Apache Hive*.

ROZDZIAŁ II: Eksploracja Danych

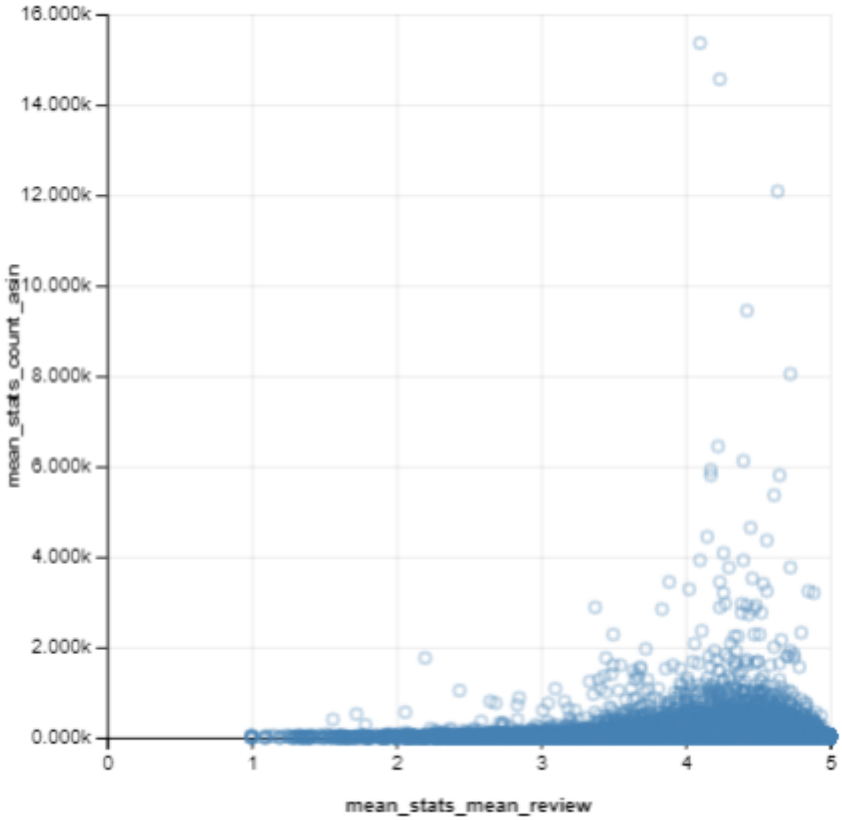
PODROZDZIAŁ 1: Dane Kindle

Wstępna analiza eksploracyjna danych została wykonana na próbce pierwszych 1000 wierszy w zbiorze – jest to jedynie pomocnicza analiza pozwalająca ustalić, co zawierają poszczególne kolumny oraz podjęcie decyzji, które dane chcemy zatrzymać do dalszej analizy, a które można już usunąć. W związku, że mamy do czynienia z tematyką big data, takich wyfiltrowywań należy dokonywać jak najwcześniej, żeby możliwie jak najbardziej ograniczać wymiarowość

danych, w szczególności zanim zaczniemy wykonywać na nich bardziej intensywne obliczeniowo i „drogie” operacje jak np. łączenie danych (joins).

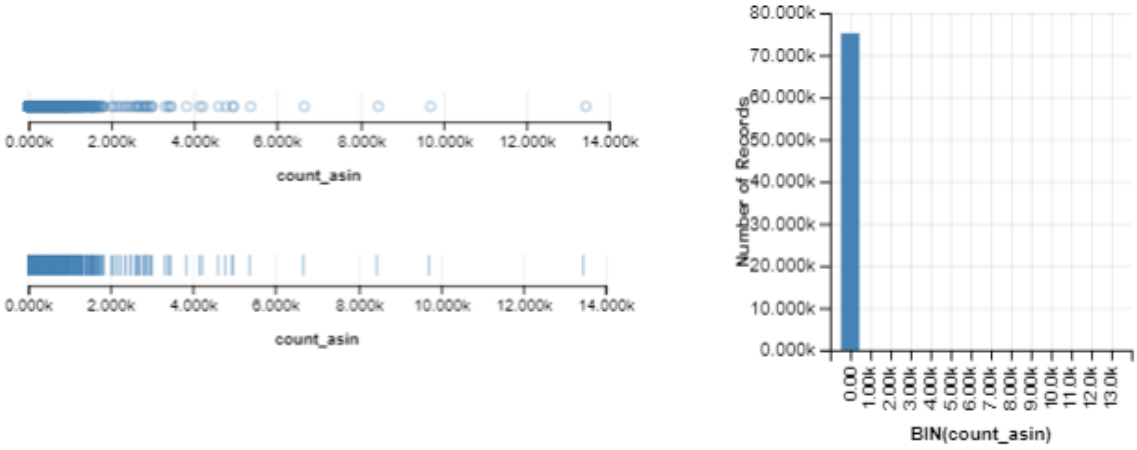
- overall – ocena od 1 do 5 nadana przez użytkownika
- vote – ta kolumna wydaje się zawierać dużo nieuporządkowanych danych; zawiera zarówno dane numeryczne, które nie do końca wiadomo co oznaczają, jak również zdania, które wydaje się, że powinny być w kolumnie reviewtext (kiedykolwiek w kolumnie vote znajduje się tekst, to odpowiadające pole w kolumnie reviewtext jest wartością null). Można zatem przenieść tekst z kolumny vote do kolumny reviewtext, jednak zazwyczaj dla tych wierszy nie ma odpowiadającej im oceny w kolumnie ‘overall’, więc do dalszej analizy semantycznej opinii nie będą wykorzystane. Mogą być natomiast użyte jako ślepy zbiór testowy – zbiór na którym dokonujemy predykcji po wyborze najlepszego modelu. Dlatego też, przeniosę pola tekstowe z kolumny ‘vote’ do kolumny ‘reviewtext’ jeżeli odpowiadające pole w ‘reviewtext’ ma wartość null.
- verified – kolumna zawierająca wyłącznie 3 różne wartości: True, False i null. Najprawdopodobniej oznacza ona, że zakup danego e-booka został zweryfikowany przez Amazon, czyli, że klient rzeczywiście nabył daną pozycję poprzez Amazon.
- reviewtime – data, kiedy dana opinia została wystawiona; ogólnie można byłoby zbadać, czy istnieje jakaś zależność między czasem i oceną, ale w przypadku tego projektu, nie to dalej badane
- reviewerid – numer identyfikacyjny osoby wystawiającej recenzję.
- asin – numer identyfikacyjny produktu wystawiany przez Amazon; jest to tak naprawdę numer oznaczający konkretny ebook w przypadku tej analizy.
- style – kolumna pokazująca wersję książki: czy jest to wydanie Kindle czy wersja drukowana. Większość danych jednak jest pusta. Kolumna ta niewiele wnosi pod kątem celu analizy wykonywanej w tym projekcie.
- reviewername – imię i nazwisko osoby wystawiającej opinię; ze względu na konieczność ochrony danych osobowych, takie dane powinny być stokenizowane, czyli zamienione na tokeny. Jako że w tych danych istnieje już kolumna ‘reviewerid’, ta kolumna może zostać bezpiecznie usunięta. Ponadto, w przypadku tej analizy te dane nie będą przydatne, zatem nie będą one używane.
- reviewtext – opis recenzji wystawionej przez użytkownika; niestety, jak w przypadku rzeczywistych danych, niektóre wpisy zawierają literówki bądź inne charakterystyki wyrażające emocje np. ‘bok’ zamiast ‘book’ lub ‘gooooooood’ zamiast ‘good’. Jedną z technik jaką potencjalnie można byłoby wykorzystać jest wyczyszczenie takich danych zwane „normalizacją” jak zostało to opisane w sekcji *ROZDZIAŁ IV: Modelowanie; PODROZDZIAŁ 3: Głębokie Sieci Neuronowe - PYTHON – GOOGLE COLAB.*
- summary – zawiera krótkie podsumowanie opinii. Najprawdopodobniej jest to tekst zazwyczaj wpisywany w tytule opinii. Z reguły można byłoby dołączyć te dane do reviewtext i analizować jako całość, jednak można przyjąć założenie, że ‘summary’ będzie spójne z opinią wystawioną w ‘reviewtext’, tylko skromniejsze, a zatem zawierające mniej informacji. Dlatego, w dalszej części założono, że summary nie wnosi znaczącej wartości do analizy i badania danych, a przez wgląd na zmniejszenie wymiarowości, może zostać usunięte w dalszej analizie.
- unixreviewtime – data i czas w formie unixowej; podobne do ‘reviewtime’. Ta kolumna nie będzie używana w dalszej analizie.

Rysunek 4 Relacja między średnią liczbą ocen danego ebook’a a jego średnią oceną (graf został sporządzony na podstawie wszystkich danych)

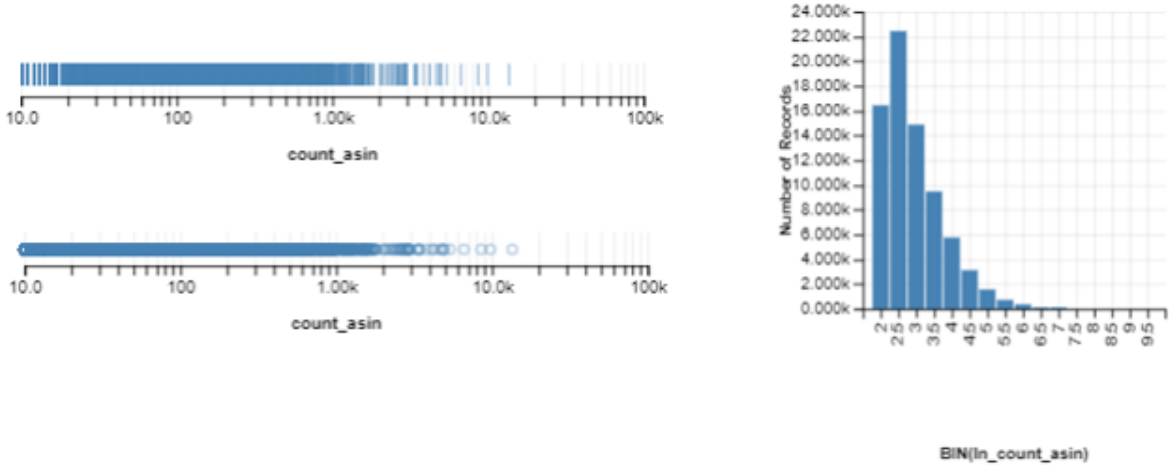


Stworzenie rankingu wyłącznie na podstawie ocen jest niemiarodajne. Wyobraźmy sobie np. sytuację, że mamy 2 książki, które mają taką samą średnią ocenę. Wg tego kryterium są one zatem na tym samym miejscu w rankingu. Załóżmy, że jedna ma natomiast 10 ocen a druga 5000. Druga książka jest zatem bardziej popularna. Pierwsza może być bardziej niszowa i trafiać w gusta bardzo ograniczonemu gronu czytelników o mocno wysublimowanych upodobaniach. Druga książka powinna zatem zająć wyższe miejsce w rankingu. Należy więc w jakiś sposób zawrzeć informację o liczbie ocenw końcowej mierze używanej do shierarchizowania produktów. Jak zatem to zrobić? Jedną z możliwości jest wyliczenie średniej ważonej oceny, gdzie wagą jest liczba ocen. Takie podejście niesie za sobą natomiast innego rodzaju komplikacje. Rysunek 5 ilustruje rozkład liczby ocen dla różnych pozycji książkowych w zbiorze. Można zauważyć, że rozkład ten przypomina rozkład lognormalny, w którym większość obserwacji jest skupiona wokół lewego końca rozkadu, a niewiele obserwacji przybiera nieproporcjonalnie duże wartości (innym przykładem takiej dystrybucji w populacji jest też rozkład dochodów czy wartości netto w społeczeństwie). Używanie wartości z takiego rozkładu jako wag sprawi, że ostateczne wyniki będą bardzo mocno z nim skorelowane, a kilka najwyższych pozycji w rankingu będzie właściwie tożsame z najwyższymi pozycjami z tego rozkładu. Odpowiednim rozwiązaniem jest nieznaczne zmienienie kształtu takiego rozkładu, aby był on bardziej wyrównany. Jedną z możliwości jest przetransformowanie go używając logarytmu naturalnego, który w teorii doprowadza rozkład lognormalny do rozkładu normalnego. Efekt tego zabiegu został zobrazowany przez Rysunek 6.

Rysunek 5 Rozkład liczby recenzji. Na rysunku widać, że „środek ciężkości” rozkładu jest znacznie przesunięty w stronę pierwszego kwartyla. Pozycje z bardzo wysoką liczbą ocen mogą być uważane za ‘outliery’ – wartości odstające. Stanowią one jednak ważną część.

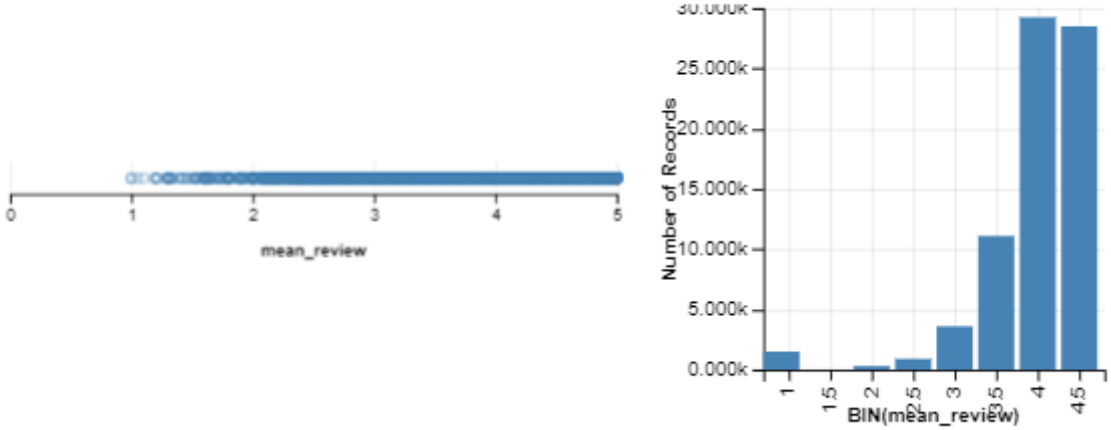


Rysunek 6 Rozkład liczby recenzji po zastosowaniu transformacji logarytmicznej. Rozkład jest o wiele bardziej podobny do rozkładu Gaussa, jednak wciąż nie jest on symetryczny i posiada ogon po prawej stronie. Efekt częściowego „znormalizowania” został jednak osiągnięty.

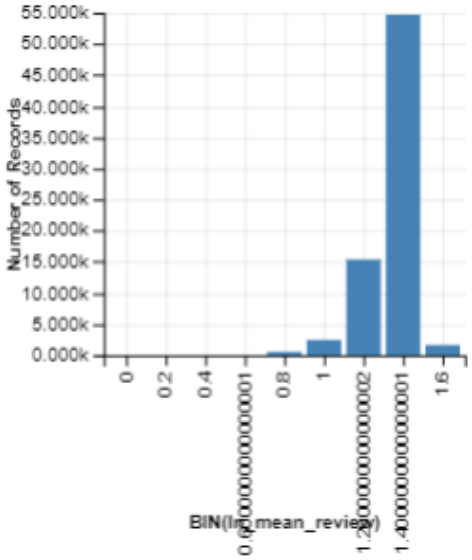


Z kolei średnia ocena dla książki, czyli ‘mean_review’, jest o wiele bardziej regularna jak zostało to pokazane w części Rysunek 7. Rozkład ten, jest wciąż dość nieregularny i większość średnich ocen mieści się w przedziale 3.5-5, jednak nie wymaga on szczególnej ingerencji. Dodatkowym argumentem jest to, że jesteśmy głównie zainteresowani wyodrebnieniem kilku najwyżej plasujących się pozycji, więc tak naprawdę będziemy zainteresowani właśnie częścią w przedziale 3.5-5, gdzie mieści się znaczna większość danych. Gdyby transformację logarytmu naturalnego zastosować na zmiennej średnia ocena, to rozkład wypadkowy byłby o wiele bardziej podobny do rozkładu normalnego. Dla zobrazowania, zostało to pokazane w części Rysunek 8. Jako że pierwotny rozkład nie jest bardzo mocno niezbilansowany, a żależy nam na możliwie bliskim trzymaniu się skali 0-5, transformacja ta nie została zastosowana w dalszej analizie.

Rysunek 7 Rozkład średniej oceny dla każdej książki / produktu.



Rysunek 8 Rozkład średniej oceny dla każdej książki / produktu po zastosowaniu transformacji logarytmu naturalnego.



PODROZDZIAŁ 2: Dane Meta Kindle

Podobnie jak w przypadku danych Kindle, wstępna analiza eksploracyjna danych została wykonana na próbce pierwszych 1000 danych w zbiorze – jest to jedynie pomocnicza analiza pozwalająca ustalić, co zawierają poszczególne kolumny oraz podjęcie decyzji, które dane chcemy zatrzymać do dalszej analizy, a które można już usunąć.

- category – kolumna zawierająca wszystkie kategorie, do których należy dany produkt w formacie listy stringów (np. ["Kindle Store","Kindle eBooks","Arts & Photography"]). Ogólnie, kategoria do jakiej należy dana książka jest przydatna i będzie jedną ze zmiennych w modelu predykcji oceny danej książki. Powinna bowiem istnieć korelacja między oceną a kategorią jeśli spojrzymy na ludzi jako populację – będą istniały bowiem nurty i kategorie książek preferowane przez większość ludzi np. fantasy, książki popularnonaukowe itp. I dla tych książek spodziewamy się zobaczyć więcej ocen jak również wyższą ocenę. W obecnej formie natomiast dokonywanie takiego wnioskowania jest niemożliwe – należy wyczyścić te dane i wyciągnąć wartościowe z perspektywy tej analizy kategorie.

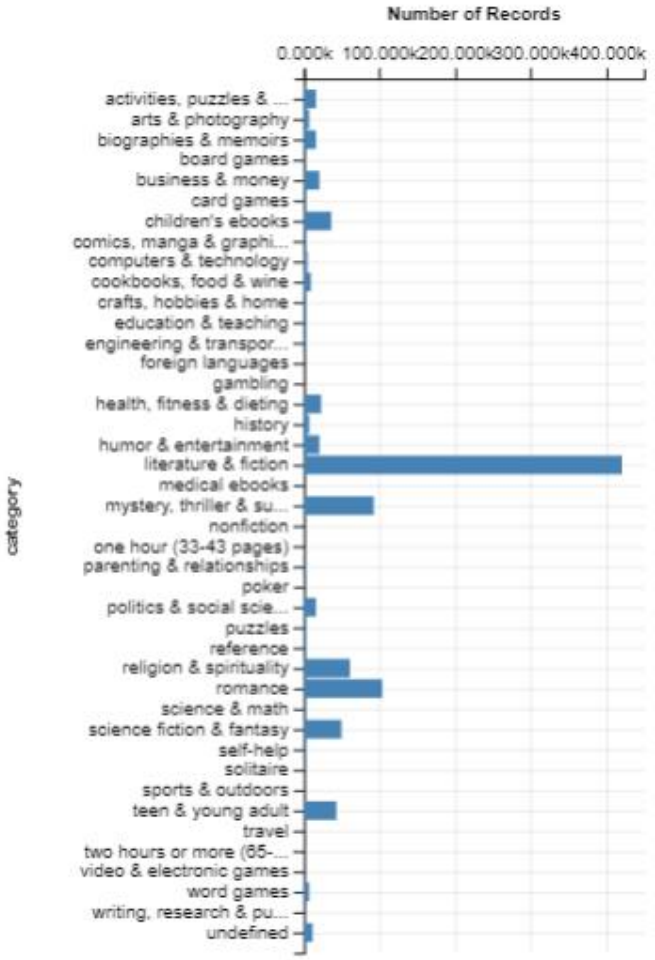
- tech1 – w pierwszym 1000 wierszy nie było w tej kolumnie żadnych danych; nie brzmi ona również jak kolumna, która może być wartościowa z perspektywy tego projektu. Nie będzie ona zatem używana.
- description – wnioski podobne do tech1
- fit – wnioski podobne do tech1
- title – tytuł książki / produktu
- also_buy – lista numerów ASIN, które są często kupowane z daną pozycją. W tej analizie te dane nie będą używane, ale są to bardzo interesujące dane, które mogłyby być wykorzystane do analizy zależności między poszczególnymi e-book'ami za pomocą np. jakiejś bazy danej grafowej takiej jak Neo4j albo używając sparkowego pakietu GraphX.
Grafowe bazy danych pozwalają na zaadresowanie makroskopowych trendów biznesowych – wykorzystanie skomplikowanych i dynamicznych zależności w danych, żeby lepiej zrozumieć połączenia i stworzyć przewagę konkurencyjną. (Ian Robinson, 2015) Jeśli chodzi o te dane, dodatkowe zamodelowanie zmiennych also_buy i also_view mogłoby pozwolić lepiej zrozumieć nie tylko zależności pomiędzy konkretnymi książkami, ale także pomiędzy gustami klientów i np. pomóc budować lepsze systemy rekomendacyjne produktów.
- image – wnioski podobne do tech1
- tech2 – wnioski podobne do tech1
- brand – przedstawia w większości nazwiska, prawdopodobnie autorów danej pozycji; w dalszej analizie ta kolumna nie będzie wykorzystywana
- feature – wnioski podobne do tech1
- rank – przedstawia ranking danej pozycji w Kindle Store; jest to pośrednio zmienna, która jest przedmiotem inferencji w tym projekcie, jednak Amazon używa zapewne swojego własnego algorytmu i analiza, którą wykonujemy pozwala na potencjalne osiągnięcie trochę innego wglądu w dane, nawet nieznacznie różnego oraz umożliwia bieżące dopasowywanie kryteriów w zależności od hipotezy i pytania jakie sobie zadamy
- also_view – lista pozycji, które są również często oglądane razem z daną książką; wnioski podobne do 'also_buy'
- details – dodatkowe informacje w formie słownika np. {"release_date":null,"file_size":null,"print_length":null,"page_numbers_source_isbn":null,"simultaneous_device_usage":null,"publisher":"Pinteresting (February 16, 2012)","publication_date":null,"language":"English","isbn_10":null,"isbn_13":null,"asin":"B007A6YWZ6","word_wise":null,"lending":null}; niektóre z nich mogą być przydatne do różnych analiz, do tego projektu dane te nie będą jednak wykorzystywane
- main_cat – pierwsze 1000 wierszy składa się wyłącznie z tekstu „Buy a Kindle". Zmienna ta nie będzie zatem dalej wykorzystywana.
- similar_item - wnioski podobne do tech1
- date – data, zmienna ta nie będzie dalej wykorzystywana. W trakcie tej analizy zauważyłam, że 'date' jest słowem kluczem wykorzystywanym w Hive. Gdyby kolumna ta miała być używa, niezbędne byłoby zmienienie nazwy tej kolumny na taką, żeby była jednoznaczna
- price – pierwsze 1000 wierszy jest pustych, gdyby jednak ta zmienna była dostępna, to byłoby interesujące zobaczyć, czy istnieje zależność między ceną i oceną wystawioną przez użytkownika, a zatem zbadanie czy cena ma wartość predykcyjną w odniesieniu do tego projektu

Kod pomocniczy użyty do wykonania powyższej eksploracji został dołączony w sekcji *DODATEK 4: Exploracja Danych w Hive*.

PODROZDZIAŁ 3: Połączone dane po wstępnym czyszczeniu

Po wykonaniu czyszczenia opisanego w sekcjach *ROZDZIAŁ III: Czyszczenie danych; PODROZDZIAŁ 1: Dane Kindle* i *PODROZDZIAŁ 2: Dane Meta Kindle*, został wykonany test, sprawdzający, czy w tabeli reduced_asin_hive_analysis, nie ma duplikatów w numerach ASIN. Jeśli liczba takich duplikatów byłaby znacząca, mogłoby to zaburzyć wyniki analizy – wpłynęłyby one na miary średnich ważonych poprzez liczenie takich pozycji książkowych kilka razy. Z dalszej analizy wypłynął wniosek, że nieunikalność wierszy jest spowodowana przez przynależność niektórych numerów ASIN do kilku kategorii. Zachowanie ich mogłoby zatem również wpłynąć na wyniki na poziomie granularności per kategoria. Z bardziej dogłębnej analizy widać, że większość takich przypadków dotyczy gier, a nie książek. Liczba unikalnych numerów asin, dla których występują takie duplikaty również jest niewielka – jest to tylko 188 unikalnych numerów (tabela z wynikami w formacie Excel została załączona w *DODATEK 8: Duplikaty – ASIN i category*). Dokonując dokładniejszego czyszczenia, można byłoby zachować jedynie po jednym przykładzie z danej kategorii oraz pogrupować kategorie w pozostałej części zbioru danych, zwłaszcza, że liczba kategorii jest stosunkowo duża (Rysunek 9 Kategorie w „surowym" zbiorze – przed usunięciem duplikatów). Ze względu jednak na niewielką liczbę takich duplikatów i to, że w większości dotyczą one gier, wydaje się to być zbyt czaso- i praco-chłonne jak na potencjalne korzyści, które można by uzyskać. Wszystkie takie przypadki, dla których występują duplikaty zostaną zatem usunięte ze zbioru.

Rysunek 9 Kategorie w „surowym" zbiorze – przed usunięciem duplikatów



Kolejnym ważnym wnioskiem jaki zazwyczaj wyciąga się na etapie eksploracji danych jest zależność między zmienną docelową, przewidywaną a zmiennymi objaśniającymi. O ile w przypadku danych tekstowych i analizy stricte sentymentu jest to niemożliwe, o tyle w przypadku zmiennej 'category' byłoby to standardowym krokiem. Możliwe rozwiązania to

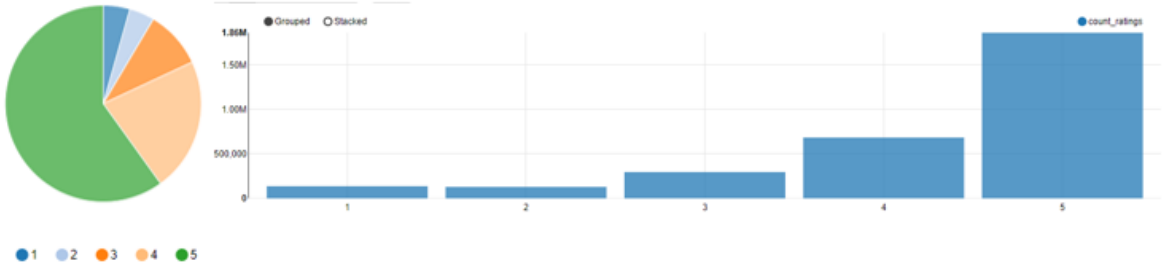
skonstruowanie tabeli częstotliwości (ang. frequency table) pomiędzy zmienną ‘category’ a zmienną objaśnianą, pozwalające ocenić, czy istnieją zależności między kategorią a oceną. Dodatkowym testem statystycznym używanym w tym celu jest też np. ANOVA[6]. Byłby to zabieg o tyle przydatny, że ponieważ liczba kategorii jest stosunkowo duża, używanie standardowego tzw. „one-hot” zakodowania wartości kategorycznych zmiennej kategorycznej na wartości numerycznych prowadzi do powstania rzadkich macierzy jak również znacząco zwiększa wymiarowość danych, co w przypadku problemów typu „big data” jest szczególnie niepożądane. Rozwiązaniem mogłoby być tzw. „target encoding”, gdzie wartości kategoryczne zamieniane są na wartości w przedziale [0, 1] w zależności od proporcji klasy „dodatniej” w każdej kategorii w przypadku klasyfikacji binarnej. Jako, że w tym projekcie ogólnie celem jest traktowanie zmiennej objaśnianej jako wieloklasowej, taka tabela częstotliwości mogłaby raczej pomóc w wyborze odpowiednich wartości, które chcielibyśmy użyć do stworzenia takiego mapowania zmiennej kategorycznej na numeryczną w przypadku później omówionego modelu GBT, w którym zmienna docelowa jest traktowana jako zmienna binarna. W tym projekcie, użyjemy jednak prostego indeksowania stringów od 0 do n, gdzie n jest liczbą poziomów w danej zmiennej i pozwolimy modelom podjąć decyzję co do istotności zmiennej ‘category’.

PODROZDZIAŁ 4: SPARK - SCALA – ZEPPELIN (HADOOP)

Tabelą z danymi używaną w tej części jest for_nlp_models.csv – tabela otrzymana w kroku *ROZDZIAŁ III: Czyszczenie danych; PODROZDZIAŁ 3: Kroki końcowe*.

W tej części podejmujemy próbę przewidzenia oceny tytułu na podstawie przede wszystkim zmiennej reviewtext, więc jest to rodzaj analizy NLP[7], czyli przetwarzania języka naturalnego. Konkretniej, jest to przewidywanie oceny na podstawie tekstu napisanego przez użytkownika / konsumenta, więc jest to analiza sentymentu. Dodatkową zmienną jaka zostanie użyta do przewidywania jest ‘category’, czyli zmienna kategoryczna zawierająca kategorię do jakiej należy dana pozycja książkowa. Na podstawie grafów na rysunku Rysunek 10 możemy zauważyć, że zbiór jest dość niezbilansowany. Znaczną większość zbioru stanowią oceny ‘5’, dalsze ok. 25% zbioru oceny ‘4’ a ‘3’, ‘2’ i ‘1’ – resztę zbioru.

Rysunek 10 Rozkład ocen zbiorze danych for_nlp_models.csv



ROZDZIAŁ III: Czyszczenie danych

[6] ANOVA – akronim dla ‘Analysis of Variance’, czyli metoda statystyczna do oceny czy istnieją istotne różnice w średnich pomiędzy poszczególnymi grupami

[7] ang. Natural Language Processing

PODROZDZIAŁ 1: Dane Kindle

Z powyższej analizy eksploracyjnej w sekcji *ROZDZIAŁ II: Eksploracja Danych; PODROZDZIAŁ 1: Dane Kindle* płyną wstępne wnioski na temat użyteczności, wartości i miarodajności danych. W związku z nimi zostały podjęte następujące kroki:

1. Zostały zachowane tylko takie dane, dla których jest obecny numer ASIN, ponieważ, jeśli nie znamy numeru ASIN, nie jesteśmy w stanie określić jaka to jest książka. Chociaż dane, które nie mają numeru ASIN mogą być wartościowe z punktu widzenia analizy semantycznej, to w przypadku tego projektu chcemy również użyć kategorii, do której należy książka jako zmiennej wsadowej (jeśli np. pewne kategorie książek są z reguły preferowane przez użytkowników), a nie posiadając numeru ASIN, nie będziemy w stanie dołączyć odpowiadającej kategorii ze zbioru z meta danymi[8]. Ponadto, ASIN jest używany do filtrowania danych i uzyskiwania różnych statystyk per książka. Jeśli tego numeru nie ma, nie jesteśmy tego w stanie zrobić.
2. Do dalszej analizy wybieramy jedynie książki, które zostały ocenione co najmniej 10 razy. O ile do analizy semantycznej mogłyby być wzięte pod uwagę, o tyle do wyciągania wniosków, które filmy mają najwyższe oceny zakładamy, że mają zbyt mało danych, żeby ich oceny mogły być uznane za wartościowe (mogą to być np. książki bardzo specjalistyczne / niszowe, które mogą przypaść do gustu tylko ograniczonej grupie czytelników, lub po prostu książki mało popularne. Na uwagę zasługuje Rysunek 4, na którym widać zależność pomiędzy liczbą ocen a średnią oceną dla danego e-book'a. Na grafie widoczny jest interesujący wzorzec – ogólnie korelacja pomiędzy 'popularnością' a średnią oceną jest widoczna, jednak nie jest ona bardzo silna. Brak tej zależności szczególnie widać dla produktów, które zostały ocenione mniej niż ok. 1000 razy. Powyżej 2000 ocen widać natomiast trend wyraźnie wyższej oceny – przeciętnej lub ponadprzeciętnej. Jest to dość spodziewana zależność w przypadku dobrze działających systemów rekomendacyjnych oraz swobodnego dostępu do informacji. W obecnych czasach, dzięki rozpowszechnieniu Internetu oraz użycia metod uczenia maszynowego na wysokim poziomie rzeczywiście możemy stwierdzić, że obydwa te zjawiska występują. I tak, można się spodziewać, że książki które są bardziej popularne będą miały ponadprzeciętną ocenę stąd obserwowana korelacja. Jest to zatem argument, aby w dalszej analizie wziąć pod uwagę produkty, które otrzymały liczbę recenzji powyżej jakiegoś minimalnego progu.
3. Wyfiltrowanie tylko takich ocen, dla których zakup został zweryfikowany. Ma to na celu uniknięcie dokonywania analiz na ocenach, które mogą nie być do końca miarodajne – np. zostały dodane przez przypadek lub przez bezpośrednią konkurencję.
4. Kalkulacja średniej liczby recenzji i średniej oceny per książka (czyli per ASIN).
5. Wyczyszczenie kolumny reviewtext – jeśli kolumna reviewtext jest pusta, a jakiś tekst znajduje się w kolumnie 'vote', możemy stwierdzić na podstawie wcześniejszej analizy powyżej, że jest to recenzja zawarta w nieodpowiednim miejscu.

Kod wykonujący powyższe kroki może zostać znaleziony w sekcji *DODATEK 5: Wstępne przygotowanie danych* .

PODROZDZIAŁ 2: Dane Meta Kindle

[8] Metadane – dane o danych

Z powyższej analizy eksploracyjnej w sekcji *ROZDZIAŁ II: Eksploracja Danych; PODROZDZIAŁ 2: Dane Meta Kindle* płyną wstępne wnioski na temat użyteczności, wartości i miarodajności danych. W związku z nimi zostały podjęte następujące kroki:

1. Do dalszej analizy zostały zachowane tylko pola: ASIN, title, category.
2. Kolumna ‘category’ wymagała dość dużego czyszczenia i przeprocesowania. Najpierw kolumna ta została przetransformowana za pomocą metody **explode** w taki sposób, że pojedyncze elementy listy w kolumnie zostały zapisane w oddzielnych wierszach, tworząc duplikaty w rekordach na przestrzeni pola ‘asin’. Następnie, kolumna ta zamieniana jest na małe litery. Ten zabieg ujednolici format danych i ułatwi dalszą manipulację nimi. Kolejnym krokiem jest wyfiltrowanie tylko tych wierszy, które nie zawierają pewnych słów kluczowych – substringów. Tym sposobem zachowujemy tylko element z listy zawierający właściwą kategorię, do której należy dany e-book. Na końcu, usuwane są niektóre znaki specjalne. Efekt powyższego czyszczenia obrazują poniższe zrzuty ekranu tabel z danymi z widoku w Apache Hive. Rysunek 11 pokazuje kolumnę ‘category’ w jej pierwotnej formie, natomiast Rysunek 12 po zastosowaniu powyższego czyszczenia.

Rysunek 11 Wycinek danych – asin i category – z danych meta_kindle z ‘nieobrobioną’ kolumną category

asin	category
B000FA5KKA	["Kindle Store","Kindle eBooks","Science Fiction & Fantasy"]
B000FA5M3K	["Kindle Store","Kindle eBooks","Engineering & Transportation",""]
B000FA5KJQ	["Kindle Store","Kindle eBooks","Biographies & Memoirs"]
B000FA5NSO	["Kindle Store","Kindle eBooks","Science Fiction & Fantasy"]
B000FA5KX2	["Kindle Store","Kindle eBooks","Business & Money"]

Rysunek 12 Wycinek danych – asin, title i category – z danych meta_kindle po przeprocesowaniu ‘category’; dla łatwiejszej możliwości porównania zostały zaprezentowane te same numery ‘asin’

meta_kindle_all_cat_v2.asin	meta_kindle_all_cat_v2.title	meta_kindle_all_cat_v2.category
B000FA5KKA	""	science fiction & fantasy
B000FA5M3K	""	engineering & transportation
B000FA5KJQ	""	biographies & memoirs
B000FA5NSO	""	science fiction & fantasy
B000FA5KX2	""	business & money

Kod wykonujący powyższe kroki może zostać znaleziony w sekcji *DODATEK 6: Wstępne przygotowanie danych meta_Kindle.*

PODROZDZIAŁ 3: Kroki końcowe

Kolejnym krokiem jest połączenie obydwu zbiorów Kindle i meta_Kindle oraz wstępna ekstrakcja danych, które zostaną użyte w dalszej analizie.

Na tym etapie tworzone są 2 tabele:
- for_nlp_models zawierająca kolumny asin, overall, reviewtext, reviewerid i category; ta tabela zostanie następnie wyeksportowana w formacie .csv i będzie używana do zbudowania modelu przewidującego ocenę daną danemu e-book'owi
- reduced_asin_hive_analysis zawiera wyniki pogrupowane po numerze ASIN; w jej skład wchodzą kolumny asin, category, title, count_asin z liczbą ocen oraz mean_review, czyli średnią oceną dla książki; ta tabela będzie dalej wykorzystana do wysunięcia wniosków na temat najpopularniejszych książek, a analiza ta zostanie zrobiona w Hive.

Kod wykonujący powyższy krok końcowy został załączony w sekcji *DODATEK 7: Połączenie dwóch zbiorów oraz przygotowanie danych do dalszej analizy w Hive, Spark Scala i Python*.

Powyższe zbiory zostały zapisane w formacie .csv na HDFS w Hadoop.

Jako że procesy eksploracji i czyszczenia danych są poniekąd procesami iteracyjnymi – często nie jest możliwe jednorazowe zbadanie danych, niektóre zależności są widoczne po wstępnym czyszczeniu i agregacjach danych itp., to dalsze przeczyszczenie / przygotowanie do modelowania zostanie jednak wykonane w odpowiednim języku w zależności od użytego modelu – Apache Hive, Spark Scala lub Python.

ROZDZIAŁ IV: Modelowanie

PODROZDZIAŁ 1: HIVE

Często interesującym pytaniem ze strony konsumenta, a więc również z perspektywy środowiska biznesowego jest: Jakie produkty są najwyżej oceniane? Na większości stron internetowych, można poszeregować wyniki według np. popularności (te które mają najwięcej ocen) lub najwyższej oceny (często w formie od 1 do 5). Tak naprawdę jednak sama ocena produktu, nie biorąc pod uwagę popularności daje jedynie szczątkowy obraz i wgląd na najlepiej oceniane produkty, gdyż może się zdarzyć, że wyżej będą oceniane produkty z mniejszą liczbą ocen lub w ogóle tylko z jedną oceną.
Spróbujmy przeanalizować 2 różne produkty z następującymi ocenami i liczbą recenzji:

1. Produkt A – średnia ocena: 5 – liczba ocen: 1
2. Produkt B – średnia ocena: 4.5 – liczba ocen: 800

Ciężko zatem stwierdzić, który produkt jest lepszy używając jedynie jednej z miar popularności lub maksymalnej średniej oceny. Ten problem został już po części omówiony na stronie 13. Z punktu widzenia biznesu, przydatne byłoby, gdyby potencjalni klienci mogli uszeregować produkty według najwyższych ocen, ale tak, aby popularność produktu była również zawarta w mierze, używanej do oceny. Jakie istnieją zatem możliwości? Jednym rozwiązaniem byłoby wprowadzić możliwość filtru wyłącznie tych produktów, które zostały zrecenzowane przynajmniej określoną liczbę razy. Rzeczywiście – niektóre sklepy e-commerce dają taką

możliwość. Jednakże, prowadzi to do całkowitego wykluczenia pozycji, które nie otrzymały wystarczająco dużej liczby recenzji. Może również być tak, że jakaś pozycja jest często wybierana i kupowana, ale nie otrzymała dużo recenzji (chociaż z reguły będzie istniała silna korelacja między liczbą ocen, a tym, jak często dany produkt jest kupowany). Takie podejście nie rozwiązuje również do końca zagadnienia badania popularności książek z liczbą recenzji powyżej ustalonego progu – wciąż wybór dla tych przypadków będzie ograniczał się do kompromisu pomiędzy liczbą recenzji a najwyższą oceną. Zaproponowanym rozwiązaniem byłoby np. stworzenie mieszanej miary (takiej jak średnia ważona), która będzie składała się z oceny jak również liczby recenzji. Jak wspomniano wyżej na stronie 13, użycie np. średniej ważonej oceny, gdzie liczba recenzji jest wagą prowadzi do komplikacji z powodu nieregularnego rozkładu liczby ocen i w rezultacie otrzymane najwyższe wyniki są tak mocno skorelowane z liczbą recenzji, że właściwie odzwierciedlają głównie tę cechę. Zastosowanie przekształceń typu normalizacja bądź standaryzacja również nie rozwiązuje w pełni tego problemu, gdyż skala liczby recenzji jest bardzo duża (od praktycznie 0 do 13000+), a rozkład mocno nieregularny z tzw. „ogonami" i po takiej transformacji wciąż pozostaje znaczący wpływ liczby recenzji na średnią ważoną. Jednym z zabiegów jest zatem przetransformowanie rozkładu poprzez transformację logarytmiczną - po tym rozkład jest nieco bardziej symetryczny i regularny jak widać na rysunkach: Rysunek 5 i Rysunek 6.

Wciąż natomiast pozostaje kwestia skali. Jeśli nie zastosujemy dodatkowych dopasowani zmiennej ln(count_asin), to nadal finalna najwyższa ocena otrzymana poprzez przeważenie średniej oceny przez tą zmienną, będzie mocno odzwierciedlał „popularność". Wpływ ten można zmniejszyć poprzez przeskalowanie danych i zrzutowanie ich do przedziału [0,1]. W data science i uczeniu maszynowym ta technika jest często nazywana MinMaxScaler() wg poniższego wzoru (Równanie 1 Skalowanie min-max):

Równanie 1 Skalowanie min-max

$$u = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Zabieg ten mapuje wartości naszego rozkładu w przestrzeń [0,1]. Nie rozwiązuje to jednak do końca problemu, ponieważ rozkład ln(count_asin) wciąż jest nieregularny – wysoko ocenione pozycje, które mieszczą się w np. 3cim kwartylu, wciąż mogą cieszyć się stosunkowo dużą popularnością np. 4000 ocen, jednak ze względu na to, że najczęściej oceniana pozycja ma aż 14000 ocen i dystans pomiędzy tymi dwoma przykładami jest duży na przestrzeni zmiennej ln(count_asin), to taka książka (z 4000 ocenami) będzie niedoszacowana. W związku z tym, aby „skurczyć" nieco rozkład, zastosowano kolejne mapowanie wartości na przedział [0.7, 1]. Pozwoliło to na prowizoryczne „zmniejszenie" dystansów i zminimalizowanie wpływu popularności na rzecz wpływu oceny. Zostało to osiągnięte poprzez poniższą transformację:

Równanie 2 Skalowanie standardowego rozkładu jednostajnego

$$s = 0.7 + 0.3u$$

Jest to często używana transformacja, gdy mamy do czynienia ze zmienną ze standardowego rozkładu jednostajnego i chcemy przetransformować ją aby była z rozkładu U(0.7, 1), czyli w przedziale [0.7, 1]. Oczywiście nie mamy tutaj do czynienia ze zmienną stricte z rozkładu

jednostajnego. Jednak jest to wyłącznie pewne przybliżenie umożliwiające stworzenie mieszanej miary do oceny najwyżej ocenianej książki biorąc pod uwagę jej popularność.

Równanie 3 Ostatecznie użyty wzór na mieszaną średnią ocenę

$$rating(m,n) = m * \left(0.7 + 0.3\frac{\ln n - \min(\ln n)}{\max(\ln n) - \min(\ln n)}\right)$$
$$= m * \left(0.7 + 0.3\frac{x - \min(x)}{\max(x) - \min(x)}\right) = m * (0.7 + 0.3u) = m * s$$

Gdzie m oznacza mean_review a n count_asin.

Inną metodą najczęściej obecnie stosowaną w e-commerce jest skonstruowanie mieszanej miary posiłkując się oszacowywaniem Bayes'a[9]. Szacowanie takie, bazujące na metodyce wywodzącej się ze statystyki Bayes'a bierze pod uwagę fakt, że nie mamy wystarczająco dużo danych, aby bazować szacowanie na średniej jak również inkorporuje resztę informacji jaką mamy o innych danych w zbiorze (w tym przypadku nie tylko dla poszczególnego numeru ASIN). Idea tego podejścia polega na tym, że estymujemy, biorąc obecne oceny (rozkład prior) rozkład posteriori proporcji osób, które wystawiłyby daną ocenę, jeśli moglibyśmy zapytać nieskończoną liczbę osób. W ten sposób zawieramy informację o liczbie recenzji w naszej finalnej ocenie. Jest to o wiele bardziej usystematyzowany i stabilny sposób na zawarcie liczby ocen w ostatecznym rankingu, jednak w przypadku tego projektu, to rozwiązanie nie będzie już bardziej eksplorowane. Jest to dość spore zadanie jak na część szeregującą pozycje książkowe, a głównym celem tego projektu jest analiza sentymentu. Jest to natomiast temat, o który praca ta mogłaby być rozszerzona w przyszłości ((Masurel, 2013), (Benjamin Bengfort, 2017)).

Ostateczną tabelą, na której będziemy dokonywać zapytać aby znaleźć najpopularniejsze i najwyżej oceniane filmy jest:

hive_analysis_final – oprócz kolumn już wcześniej omówionych (asin, category, count_asin, title, mean_review), zawiera również poniższe istotne pola:

- `ln_count_asin` – count_asin przetransformowany przez logarytm naturalny
- `ln_count_asin_suniform` – ln_count_asin przeskalowany wg wzoru w Równanie 1 Skalowanie min-max
- `ln_count_asin_uniform` – ln_count_asin_suniform przeskalowany wg wzoru w Równanie 2 Skalowanie standardowego rozkładu jednostajnego

Ostatecznie tak przetransformowana zmienna count_asin posłużyła za wagę dla średniej oceny (Równanie 3). Kod źródłowy został zawarty w sekcji *DODATEK 9: Ostateczne czyszczenie danych do analizy w Hive.*

PODROZDZIAŁ 2: SPARK - SCALA – ZEPPELIN (HADOOP)

Predykcja oceny na podstawie recenzji i kategorii wymaga odpowiedniego przygotowania zmiennych objaśniających. Zależy to od ogólnych wymagań takich jak np. zamienienie kategorii na wartości numeryczne, czy podobny zabieg lecz odnoszący się do tekstu. Bardziej

[9] ang. Bayes estimation

szczegółowe metody i podejście zależą jednak od kilku czynników takich jak: same dane, cel modelowania – jaki efekt chcemy osiągnąć, czy od modeli jakich używamy. W tej sekcji porównam dwa modele: Multinomial Logistic Regression (MLR), czyli regresję logistyczną, gdy zmienna docelowa posiada więcej niż dwie klasy (tak jak w przypadku surowej zmiennej objaśnianej w tych danych) i Gradient Boosted Tree (GBT), który w Sparku, w wersji, której używam, może być użyty tylko dla predykcji zmiennej binarnej. Będzie on zatem wymagał stworzenia odpowiedniej konwersji.

W przypadku modelu MLR użyję tzw. trzyskładowej cross-validation, czyli metody randomizowania wyników w celu jak największej generalizacji modelu, a zarazem jak najmniejszego dopasowania do konkretnych danych. Zbiór treningowy jest losowo dzielony na trzy części. Następnie jedna część jest traktowana jako ślepa, testowa, a na pozostałych dwóch model uczy się zależności w danych. Proces ten jest powtarzany trzy razy, a na końcu każdego wybierany jest taki model, który pozwolił na uzyskanie jak najlepszych metryk. Następnie, miary ocen są uśredniane, żeby odzwierciedlić bardziej rzeczywistą siłę predykcyjną modelu. Niestety, choć przynoszący dobre rezultaty, jest to proces bardzo złożony obliczeniowo. Ponadto zastosowałam automatyczną optymalizację niektórych hiperparametrów w formie tzw. grid.

Uczenie modelu GBT niestety było dużo bardziej czasochłonne niż uczenie modelu MLR. Nie zastosowałam zatem ani metody cross-validation ani optymalizowania hiperparametrów. Większość ustawień dotyczących hiperparametrów była domyślna. Jako potencjalne rozszerzenie tego projektu, można byłoby zoptymalizować hiperparametry bardziej manualnie metodą tzn. ‘sensitivity analysis’ lub dodać inne ulepszenia jak również optymalizacje modelu.

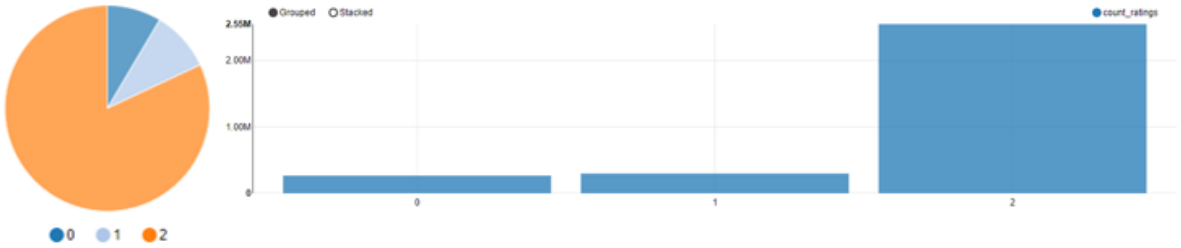
Poniżej wyjaśniam w bardziej szczegółowy sposób podejście do modelowania w przypadku obydwu modeli:

- MLR:
  1. Zagregowanie i zmapowanie zmiennej docelowej z ocenami ‘overall’ w następujący sposób:

Ocena Przed	Ocena Po
1	0
2	0
3	1
4	2
5	2

     W rzeczywistości ‘4’ i ‘5’ mogą obydwie mieć bardzo podobne opisy recenzji. Obydwa zaklasyfikowalibyśmy bowiem jako „dobre” oceny. To samo dotyczy ocen ‘1’ i ‘2’ – obydwie są ‘złe’, a ‘3’ przeciętne. Poprzez zastosowanie takiego grupowania w zmiennej docelowej, model może być w stanie lepiej wykonać zadanie klasyfikacji. Otrzymany rozkład zmiennej ‘overall’ jest natomiast jeszcze bardziej nieregularny i niezbilansowany niż poprzednio (Rysunek 13). Może to spowodować zbytnie „skupienie się” modelu na klasie większościowej i niedoszacowanie pozostałych klas jeśli to zjawisko nie zostanie odpowiednio zaadresowane poprzez techniki niwelujące jak np. przypisanie różnych wag klasowych.

Rysunek 13 Rozkład zmiennej ‘overall’ z oceną po konwersji do modelu MLR



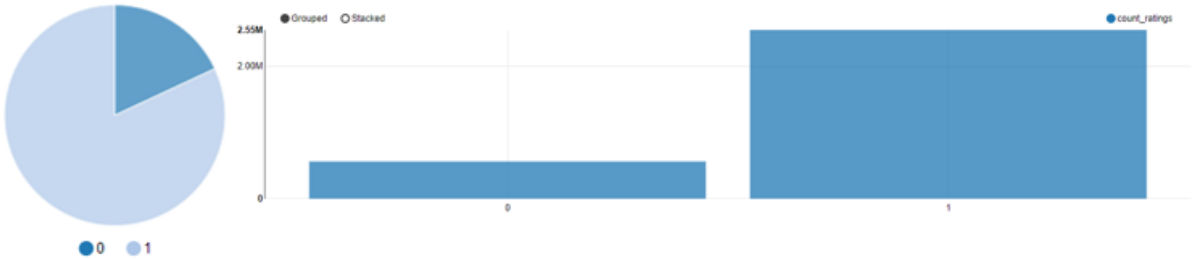
2. Zamiana zmiennej ‘reviewtext’ zawierającej recenzje w formie zdań na tokeny, czyli listę pojedynczych słów.
3. Usunięcie stopwords, czyli słów, które występują często, są przerywnikami i gramatycznymi konstrukcjami, ale nie wnoszą istotnych informacji pod względem analizy sentymentu.
4. Wektoryzowanie słów, czyli zamienianie poszczególnych słów na wartości liczbowe metodą TFIDF. Jest to metoda, która, w skrócie, mapuje poszczególne wyrazy na wartości liczbowe w zależności od częstotliwości ich występowania w zbiorze zdań / dokumentach. Jeśli natomiast jakiś wyraz występuje w wielu dokumentach, to zakłada się, że jego istotność predykcyjna jest niższa i obniża się taką wartość. Dodatkowo został użyty tutaj HashingTF, czyli sposób wektoryzowania używający tzw. „Hashing trick” i jego idea jest podobna do tzw. „Hash” tablic. Polega on na tym, że mapowania nie są do końca do unikalnych wartości. Jest to natomiast o tyle przydatne, że pozwala zachować najważniejsze informacje, zmniejszając wymiarowość danych, co jest szczególnie przydatne w przypadku zagadnień „big data”.
5. Przekonwertowanie zmiennej kategorycznej ‘category’ na watości numeryczne używając metody StringIndexer. Jest to ponumerowanie kategorii w kolejności. Nie jest to idealne rozwiązanie – można byłoby jakoś zakodować w liczbach informację o proporcjach występowania różnych ocen tzw.”target encoding”. Sama metoda StringIndexer jest natomiast stosunkowo szybkim, nieskomplikowaym sposobem bez zbędnego zwiększania wymiarowości danych.
6. Obydwie zmienne są potem łączone w jeden wektor za pomocą metody VectorAssembler(). Jest to krok typowy dla Sparka i sposóbu przetworzenia danych w nim, żeby można było skonstruować pipeline w Sparku.
7. Tak stworzone dane. są następnie dzielone na zbiory uczący i testowy. W dalszej kolejności model jest uczony oraz oceniany w standardowy sposób.

- GBT:
  1. Zagregowanie i zmapowanie zmiennej docelowej z ocenami ‘overall’ w następujący sposób:

Ocena Przed	Ocena Po
1	0
2	0
3	0
4	1
5	1

Model GBT w wersji Sparka z jakiej korzystałam wymagał, aby zmienna objaśniana była binarna, stąd konieczność powyższej zamiany. Przyjmujemy, że oceny ‘4’ i ‘5’ są uważane jako ‘dobre’, a reszta jako ‘nie-dobre’. Podział ten wynika po części z subiektywnego spojrzenia, a po części z faktu, że ‘4’ i ‘5’ stanowią znaczną większość zbioru jak widać na załączniku Rysunek 10. Można byłoby też wyodrębnić jedynie ‘5’, jednak zapewne tekstowe recenzje w przypadku oceny ‘4’ i ‘5’ są dosyć podobne. Otrzymany rozkład zmiennej ‘overall’ jest, jak w przypadku modelu MLR, dosyć nieregularny i niezbilansowany (Rysunek 14). Może to spowodować zbytnie „skupienie się” modelu na klasie większościowej i niedoszacowanie pozostałej. Niezbilansowanie to nie jest jednak bardzo duże jak na tego typu problemu, które spotyka się w data science.

Rysunek 14 Rozkład zmiennej ‘overall’ z oceną po konwersji do modelu GBT



2. Tak jak punkt 2. na stronie 24.
3. Tak jak punkt 3. na stronie 24.
4. Tak jak punkt 4. na stronie 24.
5. Tak jak punkt 5. na stronie 24.
6. Tak jak punkt 6. na stronie 24.
7. Następnie na tak przekonwertowanych danych budowany jest model GBT z w większości domyślnymi hiperparametrami oraz bez użycia metody cross-validation.

Kod źródłowy został załączony w sekcji *DODATEK 12: Modele MLR i GBT na próbce* danych w Sparku. W celu pokazania kodu i niektórych wyników, wspomniany kod został wgrane na github w formie notebook’ów Jupyter’a, czyli .ipynb. W przypadku pierwotnej analizy, kod został napisany i wykonany w Zepplin i notebook’i mają formę .json.

Wybór modelu

Najpierw zostanie omówiony wpływ skali i ilości danych użytej do uczenia modelu (w tym przypadku MLR). Rysunek 16 przedstawia wyniki wówczas, gdy zostały użyte wszystkie dane z tabeli for_nlp_models.csv, natomiast Rysunek 17 wyniki dla podzbioru ok. 500,000 danych. Porównując miarę mieszaną F1 widzimy, że przy użyciu wszystkich danych jest ona wyższa (80.7%) niż w przypadku użycia 500,000 danych (80.1%). Różnica ta nie jest znacząca, co może świadczyć o tym, że znajdujemy się blisko punktu granicznego, w którym użycie większej ilości danych nie powoduje znaczących korzyści w dokładności metryk tak jak to zostało zauważone na przykładzie slajdu w Rysunek 19 Slajd z konferencji Andrew Ng pokazujący uzysk ze zwiększenia ilości danych treningowych w przypadku algorytmów Deep Learning i starszych algorytmów uczących. Jako że teoretycznie w Sparku w Ambari można użyć większej ilości danych, nie są narzucane żadne zewnętrzne ograniczenia i model można zostawić na noc, żeby się liczył (w przeciwieństwie np. do Google Colab, który wymaga, aby

aktywności użytkownika minimum w określonych odstępach czasowych czy dość wąskie ograniczenia dotyczące zużycia pamięci RAM), do dalszej analizy zostanie wzięty pod uwagę model korzystający z pełnego zbioru danych.

Następnie porównajmy modele z wynikami przy użyciu jedynie zmiennej ‘reviewtext’ (Rysunek 15) i taki, gdzie zostały użyte obydwie zmienne ‘reviewtext’ i ‘category’ (Rysunek 16). Porównują miarę F1 widzimy, że model bazującu wyłącznie na zmiennej ‘reviewtext’ osiągnął 80.4%, podczas gdy model, który wykorzystał informacje z obydwu zmiennych osiągnął 80.7%. Widać zatem nieznaczną przewagę modelu, który miał do dyspozycji również informację o kategorii, do jakiej należy dana pozycja książkowa, jednak wpływ ten był znikomy, co może świadczyć o małej mocy predykcyjnej kategorii w odniesieniu do oceny.

Rysunek 15 Wyniki dla modelu MLR w Sparku przy użyciu wyłącznie zmiennej ‘reviewtext’ (wszystkie dane – ok. 3mln)

```
Confusion matrix:
14995.0  2721.0  34848.0
3775.0   6757.0  48122.0
2900.0   3391.0  498820.0
accuracy: Double = 0.8446333046149054
Summary Statistics
Accuracy = 0.8446333046149054
Weighted precision: 0.8116540554906322
Weighted recall: 0.8446333046149055
Weighted F1 score: 0.8046790477510173
Weighted false positive rate: 0.6134454688876396
```

Rysunek 16 Wyniki dla modelu MLR w Sparku przy użyciu zmiennych ‘reviewtext’ i ‘category’ (wszystkie dane – ok. 3mln)

```
Confusion matrix:
14125.0  2593.0  33628.0
3363.0   6536.0  46984.0
2692.0   3281.0  491108.0
accuracy: Double = 0.846865019609141
Summary Statistics
Accuracy = 0.846865019609141
Weighted precision: 0.8144684381243038
Weighted recall: 0.8468650196091411
Weighted F1 score: 0.8070515918639329
Weighted false positive rate: 0.6202997707641787
```

Rysunek 17 Wyniki dla modelu MLR w Sparku przy użyciu zmiennych ‘reviewtext’ i ‘category’ (podzbiór danych – ok. 500,000)

```
Confusion matrix:
2372.0  461.0   5674.0
619.0   1071.0  7886.0
463.0   603.0   80501.0
accuracy: Double = 0.8423883592574009
Summary Statistics
Accuracy = 0.8423883592574009
Weighted precision: 0.8073653061777808
Weighted recall: 0.842388359257401
Weighted F1 score: 0.801804353087814
Weighted false positive rate: 0.6159478974618667
```

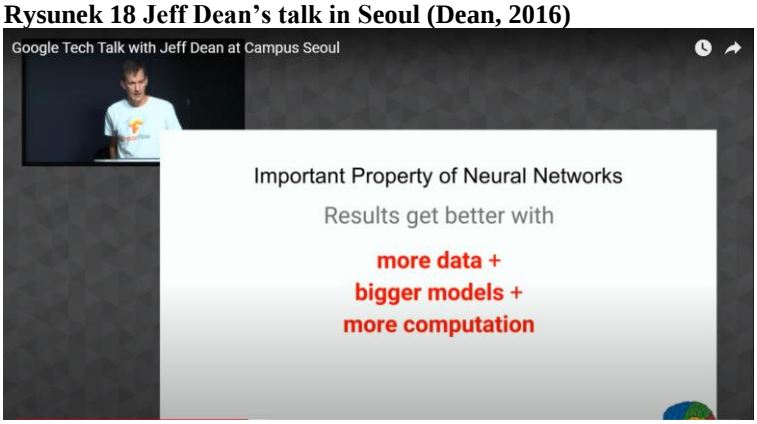
Ostatecznie przyjrzyjmy się jeszcze wynikom modelu GBT. Ze względu na wyniki powyższej analizy, został sprawdzony jedynie model bazujący na dwóch zmiennych – ‘reviewtext’ i ‘category’. Miara auROC, czyli pole pod krzywą ROC dla modelu GBT było jedynie 54%. W takiej konfiguracji, model GBT osiągną zatem bardzo słaby wynik. Ciekawe byłoby zoptymalizowanie tego modelu tak, aby móc uzyskać lepsze wyniki, jednak jako, że model MLR wydaje się być wystarczająco porządny, jak też prostszy obliczeniowo i zużywający mniej zasobów, w dalszej analizie będziemy brać pod uwagę jedynie model MLR>

PODROZDZIAŁ 3: Głębokie Sieci Neuronowe - PYTHON – GOOGLE COLAB

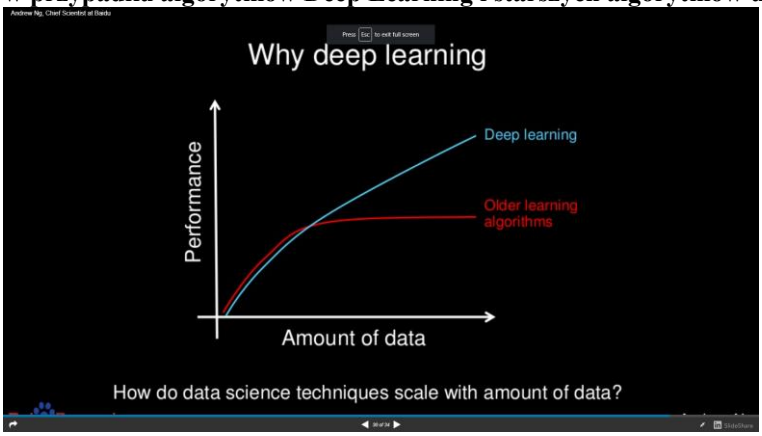
Motywacja wyboru głębokich sieci neuronowych do modelowania

Chociaż myśląc o „big data”, głębokie sieci neuronowe nie są zazwyczaj pierwszym pojęciem, które przychodzi nam do głowy, to istnieje wiele punktów stycznych między tymi dwoma zastosowaniami. Przede wszystkim, głębokie sieci neuronowe są skalowalne i im więcej danych się użyje, tym lepsze wyniki można uzyskać w przeciwieństwie do „standardowych” algorytmów uczenia maszynowego, które zazwyczaj osiągają jakąś wartość graniczną (Rysunek 18, Rysunek 19). Im większa liczba danych i im głębsze modele, tym większe również wymagania odnośnie możliwości obliczeniowych. Na szczęście postęp w tym zakresie w ciągu ostatnich kilku lat jest znaczny, co zresztą doprowadziło również do wzrostu popularności i zastosowania sieci neuronowych na większą skalę (Rysunek 20). Dostępność komputerów i innych rozwiązań (klastrów komputerów) z większymi zasobami obliczeniowymi wzrosła także w przypadku otwartych środowisk służących do modelowania takich jak np. Google Colab, gdzie teraz również jest możliwe wykonywanie obliczeń na GPU, które w sposób znaczący przyspiesza uczenie sieci. Odpowiednio zamodelowane, sieci neuronowe są też stosunkowo szybkie, zwłaszcza biorąc pod uwagę poziom skomplikowania wzorów i funkcji, które stanowią ich podstawę co dodatkowo zwiększa ich atrakcyjność w kontekście big data. Ponadto, zadaniem, do którego są używane w tym projekcie jest analiza sentymentu i przetwarzanie języka naturalnego, a sieci neuronowe są najbardziej przydatne właśnie w dostrzeganiu skomplikowanych struktur i wzorców w danych jak również sekwencji. Dlatego najczęściej mają zastosowanie w dziedzinach takich jak przetwarzanie / rozpoznawanie obrazów, rozpoznawanie języka wraz z tłumaczeniem, czy właśnie przetwarzanie języka naturalnego.

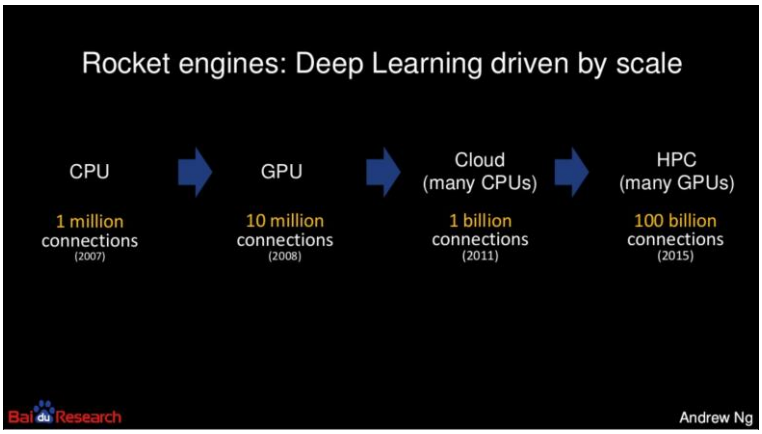
Rysunek 18 Jeff Dean’s talk in Seoul (Dean, 2016)



Rysunek 19 Slajd z konferencji Andrew Ng pokazujący uzysk ze zwiększenia ilości danych treningowych w przypadku algorytmów Deep Learning i starszych algorytmów uczących (Ng, 2015)



Rysunek 20 Rysunek pokazujący zwiększające się możliwości obliczeniowe na przestrzeni ostatnich lat (Ng, 2015)



Sprawdzone podejścia do modelowania

W przypadku podejścia do modelowania zostały przetestowane podejścia różnego typu:

1. Użycie jako zmiennej objaśnianej jedynie ‘reviewtext’ oraz obydwu ‘reviewtext’ i ‘category’.
2. Dla zmiennej ‘reviewtext’ - sprawdzenie wielu różnych konstrukcji sieci neuronowych. Szczegóły wypróbowanych typów architektury sieci zostały zaprezentowane w obrazie Rysunek 21 Typu architektury głębokich sieci neuronowych przetestowanych w modelu głębokich sieci neuronowych dla zmiennej ‘reviewtext’.

Rysunek 21 Typu architektury głębokich sieci neuronowych przetestowanych w modelu głębokich sieci neuronowych dla zmiennej 'reviewtext'

```
models = {
           "clf_LSTM_500n":
                                [LSTM(500),
                                 Dropout(0.5),
                                 Dense(nb_classes, activation="softmax")],

           "clf_BiLSTM_500n":
                                [Bidirectional(LSTM(500)),
                                 Dropout(0.5),
                                 Dense(nb_classes, activation="softmax")],

           "clf_LSTM_500n_MultipleDense":
                                [LSTM(500),
                                 Dropout(0.5),
                                 Dense(30, activation="relu"),
                                 Dropout(0.5),
                                 Dense(nb_classes, activation="softmax")],

           "clf_BiLSTM_500n_MultipleDense":
                                [Bidirectional(LSTM(500)),
                                 Dropout(0.5),
                                 Dense(30, activation="relu"),
                                 Dropout(0.5),
                                 Dense(nb_classes, activation="softmax")],

           "clf_CNN1D_500n_BiLSTM_MultipleDense":
                                [Conv1D(32, kernel_size=2, padding='valid', input_
shape=(1000, 1)),
                                 MaxPooling1D(pool_size=2),
                                 Bidirectional(LSTM(500)),
                                 Dense(30, activation="relu"),
                                 Dropout(0.5),
                                 Dense(nb_classes, activation="softmax")],

          "clf_BiLSTM_500n_CNN1D_MultipleDense":
                               [Bidirectional(LSTM(500, return_sequences=True)),
                                Conv1D(32, kernel_size=3, padding='valid'),
                                MaxPooling1D(pool_size=2),
                                Dense(30, activation="relu"),
                                Dropout(0.5),
                                Flatten(),
                                Dense(nb_classes, activation="softmax")]
}
```

Motywacja wybrania tych konkretnych typów sieci:

LSTM jest typem sieci rekurencyjnej, czyli takiej która najlepiej sprawdza się w rozpoznawaniu danych sekwencyjnych (rekurencji). Wydaje się więc być dobrym wyborem do celów modelowania danych tekstowych, które często mają pewną gramatyczną jak i znaczeniową strukturę. W przeciwieństwie do zwykłych sieci rekurencyjnych, LSTM jest natomiast w stanie nauczyć się dłuższych sekwencji – takich, gdzie często występujące w danej sekwencji słowa niekoniecznie są blisko siebie, czyli „nie zapomina" słów występujący w jakimś logicznym ciągu tak szybko. Jednocześnie, dzięki swej architekturze, czas obliczeń nie jest nadmiernie wydłużony w stosunku do „podstawowych" sieci rekurencyjnych.. Bidirectional LSTM uczy się sekwencji zarówno z lewej do prawej jak i z prawej do lewej, czyli dwustronnie. Dropout to warstwa mająca za zadanie działać podobnie do regularyzacji w przypadku bardziej standardowych modeli uczenia maszynowego. Podczas procesu uczenia, pewna losowa część danych jest chowana i niewykorzystywana. Jest to robione po to, żeby zapobiegać przeuczeniu się modelu i dopasowaniu do konkretnych danych zbyt mocno, zamiast nauczyć się bardziej ogólnych strukturalnych zależności w danych. Sieci konwolucyjne / splotowe (Conv1D) najczęściej są wykorzystywane w przetwarzaniu obrazów i mechanika ich działania polega na tym, że na dane nakładane są filtry, które są przesuwane po przestrzeni cech / zmiennych generalizując i wychwytując wzorce w danych. Można wyspecyfikować sposób w jaki sieć będzie dokonywała takiego filtrowania (może to być np. maksimum lub średnia). Z tego powodu, że sieci splotowe są efektywne w wychwytywaniu bardziej skomplikowanych struktur w danych, wydają się być dobrym wyborem do zastosowania w analizie sentymentu. Ostatecznie, użyte zostały również warstwy Dense, które są najprostsze w swej strukturze i są jedynie transformacją w postaci jakiejś funkcji zastosowaną do danych. (Geron, 2019)

Przygotowanie danych do modelu głębokich sieci neuronowych

Warto wspomnieć tutaj, że częstymi podejściami w przetwarzaniu języka naturalnego (zależnie też od ostatecznego celu analizy i tego, co chcemy przewidzieć) są następujące główne kroki (Ganesan, 2019):
- stemming – proces wyciągnięcia „trzonu" z każdego słowa pod względem językowym (a nie znaczeniowym), czyli np. „trouble" będzie zamienione na „troubl"
- lematyzacja - proces wyciągnięcia „trzonu" z każdego słowa pod względem znaczeniowym, czyli np. 'goose' i 'geese' będą zamienione na 'goose'
- usunięcie przerywników[10] - czyli słów takich jak 'the', 'an' itp.
- normalizacja – zmiana do formy kanonicznej wyrazu, czyli ujednolicenie pod względem językowym i znaczeniowym. Np. '2moro', 'tmr' itp. na 'tomorrow'.

W tym przypadku, zostały użyte gotowe embeddingi (GloVe), czyli gotowe rzutowania / mapowania słów na przestrzeń wektorów liczb, zawierające „odległości" między poszczególnymi słowami. Wektory te reprezentują „bliskość" poszczególnych słów w w kontekście danego modelu (często więc znaczeniową) taki sposób, że jeżeli jakieś słowa są podobne, to będzie to odzwierciedlone poprzez mniejszą odległość pomiędzy odpowiadającymi wektorami w przestrzeni wektorowej. W związku z tym, że zostały użyte gotowe embeddingi zawierające znaczną większość słów, to większość z powyższych metod przeprocesowywania nie musiała być zastosowana.

Poniżej wyjaśniono w bardziej szczegółowy sposób podejście do modelowania w przypadku modeli:

[10] ang. stopwords

- Głębokie sieci neuronowe z wykorzystaniem wyłącznie zmiennej tekstowej ‘reviewtext’:

    1. Podobnie jak w przypadku modelu MLR w Sparku (*ROZDZIAŁ IV: Modelowanie PODROZDZIAŁ 2: SPARK - SCALA – ZEPPELIN (HADOOP)*), zmienna docelowej z ocenami ‘overall’ została zagregowana i zmapowana w następujący sposób:

Ocena przed mapowaniem	Ocena po mapowaniu
1	0
2	0
3	1
4	2
5	2

    2. Zamienienie wszystkich liter na małe.
    3. Tokenizacja zmiennej ‘reviewtext’, czyli zamienienie zdań na pojedyncze słowa (podobnie jak w przypadku podejścia do modelowania w Sparku). Jest to standardowy krok w przypadku analizy NLP. Jest to dokonywane wbudowaną metodą w pakiecie keras.
    4. Poindeksowanie stokenizowanej zmiennej ‘reviewtext’, czyli zamiana na wartości numeryczne.
    5. Sprowadzenie wszystkich przypadków przetransformowanej zmiennej ‘reviewtext’ do jednego wymiaru – w przypadku gdy jakiś wyraz nie występuje w danym zdaniu, ma on przypisywaną wartość 0.
    6. Stworzenie macierzy embeddingów – czyli reprezentacji poszczególnych słów jako wektory w macierzy, gdzie bliskość znaczeniowa pomiędzy słowami została zakodowana jako „odległości” liczbowe pomiędzy poszczególnymi wektorami-słowami. W tym przypadku została użyta gotowe reprezentacje wektorowe wytrenowane na Uniwersytecie Stanforda (Jeffrey Pennington, 2014) – konkretnie model glove.42B.300d zawierający 42 biliony tokenów.
    7. Zastosowanie w modelu adekwatnych metryk dla modelu klasyfikacji wieloklasowej, takich jak categorical_crossentropy, softmax i categorical_accuracy
    8. Najlepszy model został zdefiniowany jako taki, który charakteryzuje się najniższą stratą dla zbioru walidacyjnego, przy wcześniej zdefiniowanej metryce jako ‘categorical_accuracy’ (celem jest osiągnięcie jak największej ‘accuracy’, czyli poprawnych przyporządkowań danych testowych do odpowiednich klas, jak również minimalizacja funkcji straty, czyli zmniejszenie błędu / odległości pomiędzy konkretnymi wartościami prawdziwymi i przewidzianymi przez model.)
    9. Zostały zastosowane również dwa rodzaje tzw. callbacks:
    - EarlyStopping z ‘cierpliwością’ 5 – oznacza to, że model zatrzyma uczenie, jeśli pięć następujących po sobie epoch nie przyniesie poprawy w wartości metryki – straty dla zbioru walidacyjnego
    - ReduceLROnPlateau z ‘cierpliwością’ 2 – oznacza, że jeżeli dwie następujące epochi nie przyniosą poprawy, to parametr ‘learning rate’ zostanie zmniejszony o zdefiniowany czynnik. Oznacza to, ze potencjalnie stopa dążenia / zbiegania do minimum będzie odbywała się w wolniejszym tempie, jednak zapobiegnie to sytuacji, gdy, jeśli będziemy wystarczająco blisko minimum, nie będziemy

„skakali" od jednego brzegu funkcji do drugiego, możliwe oddalając się od tego minimum.

Kod źródłowy wraz z wynikami został zapisany w sekcji *DODATEK 13: Model głębokich sieci neuronowych z użyciem zmiennej 'reviewtext' dla 5000* danych.

- Głębokie sieci neuronowe z wykorzystaniem wyłącznie zmiennej tekstowej 'reviewtext' i model dla zmiennej kategorycznej 'category':

  1. Wszystkie punkty są takie same jak w punktach 1 – 9 powyżej (str.31 – str. 31)
  2. Dodatkowo została zastosowana tzw. metoda bag of words dla zmiennej 'reviewtext'
  3. Zmienna 'category' została zamieniona na zero-jedynkowe wektory tzw. one-hot
  4. Model łączący obydwa te przeprocesowania to model składający się jedynie z warstwy Dense.
  5. Ostatecznie obydwa modele zostały zagregowane w jeden. Model ten został wytrenowany tylko dla tego modelu, który osiągnął najlepsze wyniki w przypadku zawarcia jedynie zmiennej 'reviewtext'. Zostało to zrobione, żeby ograniczyć czas obliczeń. Ponadto na podstawie tego, co zostało już wspomniane wyżej (Rysunek 18 Jeff Dean's talk in Seoul), istnieje dodatnia zależność pomiędzy złożonością i głębokością modelu a wynikiem.

Kod źródłowy wraz z wynikami został zapisany w sekcji *DODATEK 14: Model sieci neuronowych z użyciem zmiennych 'reviewtext' i 'category' dla 5000 danych.*

Wybór modelu

Niestety obowiązują ograniczenia użycia GPU na Google Colab. Z tego względu, zanim model zostanie puszczony na dużych danych, została dokonana wstępna selekcja modelu na mniejszych danych przy użyciu wyłącznie CPU.

Wyniki dla modelu opartego wyłącznie na zmiennej 'reviewtext' zostały przedstawione w tabeli Rysunek 22 Wyniki modeli głębokich sieci neuronowych przy użyciu jedynie zmiennej 'reviewtext' poniżej. Z wyników widać, że rzeczywiście dwa najgłębsze modele osiągnęły najlepsze wyniki. Jednak, jako, że model **clf_BiLSTM_5oon_CNN1D_MultipleDense** ma nieznacznie lepsze wyniki patrząc na metrykę F1 (mieszana metryka pomiędzy recall i precision, czyli dwiema miarami, na których maksymalizacji nam zależy i w przypadku których często trzeba „zgodzić się" na pewien kompromis w przypadku ulepszania modelu), to będzie to model analizowany w dalszych częściach tego projektu.

Rysunek 22 Wyniki modeli głębokich sieci neuronowych przy użyciu jedynie zmiennej 'reviewtext'

	accuracy	precision	recall	f1-score	model
0	0.805333	0.776924	0.805333	0.782558	clf_BiLSTM_500n_CNN1D_MultipleDense
1	0.772000	0.802483	0.772000	0.779085	clf_CNN1D_500n_BiLSTM_MultipleDense
2	0.758667	0.770607	0.758667	0.761451	clf_BiLSTM_500n_MultipleDense
3	0.744000	0.789070	0.744000	0.759532	clf_LSTM_500n
4	0.710667	0.811766	0.710667	0.749627	clf_LSTM_500n_MultipleDense
5	0.713333	0.793764	0.713333	0.746284	clf_BiLSTM_500n

Przyjrzyjmy się teraz wynikom dla tego samego modelu, ale używając dwóch zmiennych – 'reviewtext' i 'category'; jednocześnie (Rysunek 23). Można na nich zauważyć, że otrzymane metryki są dość niskie i niezadowalające. Patrząc na confusion matrix widzimy, że nieproporcjonalnie duża liczba obserwacji została przypisana do pierwszej grupy. Może to być spowodowane albo źle dobranymi wagami dla klas które były częścią modelu (chociaż wagi zostały ustawione na podstawie proporcji występowania klas w zmiennej docelowej), lub znalezienie przez model nieoptymalnego lokalnego minimum (zamiast globalnego). Na tym etapie dobrze byłoby zobaczyć jak zachowują się wyniki przy zmianie zmiennej 'learning rate', jednak ze względu na to, że model bazujący jedynie na zmiennej 'reviewtext' osiągnął wystarczająco zadowalające wyniki, zwłaszcza na jedynie 5000 danych, a wiemy z powyższej analizy, że im więcej danych, tym lepiej działają modele uczenia maszynowego, a w szczególności głębokie sieci neuronowe, to do ostatecznego uczenia zostanie wykorzystany tylko model bazujący jedynie na zmiennej 'reviewtext'.

Rysunek 23 Wyniki modeli głębokich sieci neuronowych przy użyciu zmiennych 'reviewtext' i 'category

	accuracy	precision	recall	f1-score	model
0	0.137333	0.825892	0.137333	0.108997	clf_BiLSTM_500n_CNN1D_MultipleDense

```
[[ 56   4   0]
 [ 53  19   1]
 [537  52  28]]
```

```
[[0.93333333, 0.06666667, 0.        ],
 [0.7260274 , 0.26027397, 0.01369863],
 [0.87034036, 0.08427877, 0.04538088]]
```

ROZDZIAŁ V: Produkt oparty na danych

PODROZDZIAŁ 1: Analiza w HIVE

Ostatecznie prezentujemy rankingi na dwóch poziomach granularności: ogólny (zawierający 5 najwyżej plasujących się pozycji) oraz w rozróżnieniu na poszczególne kategorie (po jednym produkcie dla każdej kategorii). Kryteria oceny, również dla celów porównawczych to:

1. najbardziej popularne książki (liczone na podstawie liczby recenzji) 2.
2. najwyższa średnia ocena.
3. najwyższa średnia ocena ważona liczbą recenzji obliczona w taki sposów jak zostało to opisane powyżej (ROZDZIAŁ IV: ; PODROZDZIAŁ 1: HIVE).

Kod źródłowy użyty do uzyskania poniższych wyników został zapisany w sekcji *DODATEK 10: Zapytania wynikowe w Hive*. Wyniki zostały zaprezentowane w tabelach poniżej. W przypadku najwyżej plasujących się tytułów dla poszczególnych kategorii (Tabela 2, Tabela 4, Tabela 6) zostały załączone i porównane tylko wyniki dla najczęściej występujących w zbiorze kategorii wg wniosków na podstawie rysunku Rysunek 9 Kategorie w „surowym" zbiorze – przed usunięciem duplikatów. Pełne wyniki są dostępne w sekcji *DODATEK 11: Hive – wszystkie wyniki*. Pomimo tego, że Tabela 4 Najwyżej oceniane książki wg zwykłej średniej oceny dla najczęstszych kategorii została otrzymana poprzez najpierw znalezienie tytułów z najwyższymi średnimi dla każdej kategorii, a następnie wybraniu tych, które miały najwyższą liczbę ocen widać, że liczby ocen są wciąż stosunkowo małe. Tabela 5 Najwyżej oceniane książki wg średniej ważoneji Tabela 6 przedstawiają najwyższe wyniki przy użyciu miary mieszanej i można zauważyć, że są one skorelowane z liczbą ocen – np. B00DMCV7K0, *The Invention of Wings: A Novel* jest najwyżej ocenianą pozycją w kategorii literature & fiction (ok. 9.7k recenzji i średnia ocena 4.65) jak również drugą w kolejności pod względem liczby recenzji. Chociaż B00YN6XHMU *Grey: Fifty Shades of Grey as told* jest bardziej popularna (ok. 13.5k recenzji i 4.28 średnia ocena), to rzeczywiście patrząc na obydwa wyniki, można byłoby stwierdzić, że B00DMCV7K0 powinno być wyżej w ogólnym rozrachunku. Zatem wpływ liczby ocen jest umiarkowany i w rozsądnym stopniu.

book	count_asin	mean_review
B00YN6XHMU, Grey: Fifty Shades of Grey as told	13442	4.288424342
B00DMCV7K0, The Invention of Wings: A Novel	9691	4.656588587

Podobnie w przypadku reszty pozycji w tabelach Tabela 5 Najwyżej oceniane książki wg średniej ważoneji Tabela 6 – zarówno liczba ocen jak i średnia ocena wyglądają na przyzwoicie wysokie tak, że można mieć zaufanie co do miarodajności otrzymanego rankingu.

1. Najbardziej popularne książki

Tabela 1 Najpopularniejsze książki ogółem (5 najczęściej ocenianych)

asin	category	count_asin	title	mean_review
B00YN6XHMU	literature & fiction	13442	Grey: Fifty Shades of Grey as told	4.288424342
B00DMCV7K0	literature & fiction	9691	The Invention of Wings: A Novel (Original Publisher's Edition-No Annotations) - Kindle edition	4.656588587
B00C2WDD5I	literature & fiction	8437	The Atlantis Gene: A Thriller (The Origin Mystery, Book 1) - Kindle edition	4.085101339
B00XSSYR50	politics & social sciences	6660	When Breath Becomes Air - Kindle edition	4.742642643
B00571F26Y	health, fitness & dieting	5341	Wheat Belly: Lose the Wheat, Lose the Weight, and Find Your Path Back to Health - Kindle edition	4.441115896

Tabela 2 Najpopularniejsze tytuły dla najczęstszych kategorii

asin	category	count_asin	title	mean_review
B00YN6XHMU	literature & fiction	13442	Grey: Fifty Shades of Grey as told	4.2884243
B00ABLJ5X6	mystery, thriller & suspense	4932	Takedown Twenty: A laugh-out-loud crime adventure full of high-stakes suspense (Stephanie Plum Book 20) - Kindle edition	4.2538524

B006P5CH1O	religion & spirituality	2857	When God Whispers Loudly (Terreldor Press Shorts Book 1) - Kindle edition	4.3916696
B015BIHKH6	romance	4214	Sense And Sensibility (Annotated	4.2019459
B00GIUG3ES	science fiction & fantasy	3438	Red Rising	4.6023851

2. Najwyżej oceniane książki (zwykła średnia)

Tabela 3 Najwyżej oceniane książki wg zwykłej średniej oceny

asin	category	count_asin	title	mean_review
B00X40JLYO	teen & young adult	15	I Love Your Crazy (The Songbooks of Jessie V Book 1) eBook	5
B01HD16PKW	teen & young adult	10	Letters to Beulah eBook	5
B00X92ANV2	religion & spirituality	12	Knowing Your Rights: A 30 Day Journey to Powerful, Purposeful, Practical Praying - Kindle edition	5
B00X5BJGZ6	science fiction & fantasy	21	The Quick and the Blue eBook	5
B00X566V86		12	Cumming With Amy Kindle Edition	5

Tabela 4 Najwyżej oceniane książki wg zwykłej średniej oceny dla najczęstszych kategorii

asin	category	count_asin	title	mean_review
B00MH6I3PE	literature & fiction	45	Finding Forever (Living Again #4) (Living Again Series) eBook	5
B00FP5CUV2	literature & fiction	45	The Corner 10 - Kindle edition	5
B01CJHZYY8	mystery, thriller & suspense	36	The Devil's Jukebox: a Virgil Roy Proctor novel - Kindle edition	5
B00VQR2NQK	religion & spirituality	52	The Chronicles of the Kings Collection: Five Novels in One - Kindle edition	5
B00OYBAI3A	romance	29	Slow and Steady Rush: A Sweet Home Alabama Novel - Kindle edition	5
B00WC90QEW	science fiction & fantasy	37	Dawn of Man (Thanos Book 1) eBook	5

3. Najwyżej oceniane książki (miara mieszana)

Tabela 5 Najwyżej oceniane książki wg średniej ważonej

asin	category	count_asin	title	mean_review
B00EFWU9QY	children's ebooks	2970	Rush Revere and the First Patriots: Time-Travel Adventures With Exceptional Americans - Kindle edition	4.9154882
B00XSSYR50	politics & social sciences	6660	When Breath Becomes Air - Kindle edition	4.7426426
B00DMCV7K0	literature & fiction	9691	The Invention of Wings: A Novel (Original Publisher's Edition-No Annotations) - Kindle edition	4.6565886
B00ESJ3S94	literature & fiction	2702	Fueled (The Driven Series Book 2) - Kindle edition	4.8619541
B001BPYMCU	business & money	4752	The Total Money Makeover	4.694234

Tabela 6 Najwyżej oceniane książki wg średniej ważonej dla najczęstszych kategorii

asin	category	count_asin	title	mean_review

B00DMCV7K0	literature & fiction	9691	The Invention of Wings: A Novel (Original Publisher's Edition-No Annotations) - Kindle edition	4.65659
B00GXV16H6	mystery, thriller & suspense	1222	The Joe Dillard Series Box Set (Books 1 through 5) - Kindle edition	4.7144
B00ET7Q4LE	religion & spirituality	1759	The 7-Day Prayer Warrior Experience (Free One-Week Devotional) - Kindle edition	4.70438
B01CZCW26K	romance	1598	Island of Glass (The Guardians Trilogy Book 3) - Kindle edition	4.77597
B00GIUG3ES	science fiction & fantasy	3438	Red Rising	4.60239

PODROZDZIAŁ 2: Porównanie i wybór ostatecznego modelu do analizy sentymentu

Ostatecznie zostały wybrane dwa modele, które osiągnęły najlepsze wyniki – model wieloklasowej regresji logistycznej w Sparku i głebokich sieci neuronowych nauczonych z wykorzystaniem mocy obliczeniowych GPU w Google Colab. Przy obydwu modelach zostało zastosowane trochę inne podejście, w zależności od wcześniej przeanalizowanych metryk oceny jak i ograniczeń narzuconych przez wymagania systemowe. Poniżej znajdują się głowne różnice (Tabela 7):

Tabela 7 Różnice w finalnie użytych modelach

Szczegóły modelu	**Bidirectional LSTM with 500 neurons + Convolutional Neural Network 1D**	**Multinomial Logistic Regression**
Dane	Podzbiór ok. 500,000 (ze względu na ograniczenia zużycia RAM w Google Colab)	Całe dane – ok. 3mln
Zmienne	'reviewtext'	'reviewtext' i 'category'
Technologia Big Data	Obliczenia z wykorzystaniem przyśpieszenia GPU	Spark – rozproszone przetwarzane danych

Zrzuty ekranu wyników zostały przedstawione w części Rysunek 24 Ostateczne wyniki dla modelu głębokich sieci neuronowych i Rysunek 25 Ostateczne wyniki dla modelu MLR w Sparku. Zostały one również podsumowane w Tabela 8 Porównanie głównych metryk oceny dla obydwu modeli. Podkreślone zostały wyższe miary. Wyniki pokazuje również Tabela 9 Confusion Matrix i znormalizowana Confusion Matrix pokazująca proporcje dla obydu modeli. W tabeli Tabela 8 widać, że model sieci neuronowych uzyskał wyższą miarę F1, jednak obydwa modele są dobre, a model MLR charakteryzuje się wyższą miarą recall, więc w zależności od wymogów danego zadania / klienta, może być uznany za lepszy. W tabeli Tabela 9 widać jednak, że model MLR zbyt dużą wagę przypisuje kategorii większościowej niż rzeczywistemu odzwierciedleniu struktury danych i tekstowi w recenzji. Jest to zachowanie dość spodziewane w przypadku niezbilansowanych zbiorów danych jak zostało to zobrazowane na rysunku Rysunek 13 Rozkład zmiennej 'overall' z oceną po konwersji do modelu MLR. Model sieci neuronowych jest o wiele bardziej poprawny – większość obserwacji z każdej klasy została

zaklasyfikowana poprawnie. W modelu sieci neuronowych zostało zastosowane natomiast wagi dla klas (kary skalibrowane wg proporcji klas w zbiorze), a wmodelu MLR nie. Koniecznym ulepszeniem zatem byłoby zastosowanie podobnych wag, w efekcie model MLR mógłby okazać się o wiele lepszy niż obecnie.

Tabela 8 Porównanie głównych metryk oceny dla obydwu modeli. Podkreślone zostały wyższe miary.

Model	Bidirectional LSTM with 500 neurons + Convolutional Neural Network 1D	Multinomial Logistic Regression
accuracy	80.27%	**84.69%**
precision	**86.01%**	81.45%
recall	80.27%	**84.69%**
F1-score	**82.48%**	80.71%

Tabela 9 Confusion Matrix i znormalizowana Confusion Matrix pokazująca proporcje dla obydu modeli

Confusion Matrix						
Bidirectional LSTM with 500 neurons + Convolutional Neural Network 1D				Multinomial Logistic Regression		
4526	1447	520		14125	2593	33628
1360	3913	1929		3363	6536	46984
1817	7722	51766		2692	3281	491108

Normalised Confusion Matrix						
Bidirectional LSTM with 500 neurons + Convolutional Neural Network 1D				Multinomial Logistic Regression		
69.71%	22.29%	8.01%		28.06%	5.15%	66.79%
18.88%	54.33%	26.78%		5.91%	11.49%	82.60%
2.96%	12.60%	84.44%		0.54%	0.66%	98.80%

Rysunek 24 Ostateczne wyniki dla modelu głębokich sieci neuronowych

	accuracy	precision	recall	f1-score	model
0	0.802733	0.860067	0.802733	0.824827	clf_BiLSTM_500n_CNN1D_MultipleDense

```
# confusion matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[ 4526  1447   520]
 [ 1360  3913  1929]
 [ 1817  7722 51766]]
```

Rysunek 25 Ostateczne wyniki dla modelu MLR w Sparku

```
Confusion matrix:
14125.0  2593.0  33628.0
3363.0   6536.0  46984.0
2692.0   3281.0  491108.0
accuracy: Double = 0.846865019609141
Summary Statistics
Accuracy = 0.846865019609141
Weighted precision: 0.8144684381243038
Weighted recall: 0.8468650196091411
Weighted F1 score: 0.8070515918639329
Weighted false positive rate: 0.6202997707641787
```

Kod źródłowy dla powyższgeo modelu głębokich sieci neuronowych został załączony w sekcji *DODATEK 15: Model głębokich sieci neuronowych z użyciem zmiennej 'reviewtext'*

dla 500,000 danych., natomiast kod dla powyższego modelu MLR w Sparku został załączony w sekcji *DODATEK 16: Model MLR w Sparku nauczony na całych danych*.

ROZDZIAŁ VI: Wnioski końcowe

W projekcie została przedstawiona wielowymiarowa analiza danych zbioru danych Kindle ze strony e-commerce Amazon. W pierwszej części, w Hive, został stworzony ranking pozycji książkowych wg mieszanej miary biorącej pod uwagę zarówno ocenę jak i liczbę ocen (popularność) danego tytułu. Z punktu widzenia biznesu e-commerce jest to bardzo przydatna analiza, ponieważ pozwoli ona użytkownikom i potencjalnym klientom wybrać pozycje, które rzeczywiście są odzwierciedleniem zarówno popularności jak i tego jak dana książka przypadła do gustu tym osobom, które już ją przeczytały. W efekcie, klient będzie w stanie szybciej odnaleźć interesujące go produkty, co z kolei może przyczynić się do wzrostu sprzedaży i budowania pozytywnej relacji z klientem oraz kultywowanie poczucia lojalności względem marki. Idea tworzenia tej miary mogłaby być również zastosowana przy budowaniu systemów rekomendacji. Następnie, za pomocą metod uczenia maszynowego w konteksie big data, została przeprowadzona analiza sentymentu i ogólna próba predykcji oceny na podstawie tekstu opinii i kategorii do jakiej należy dana książka. Na podstawie otrzymanych wyników zostało stwierdzone, że modele w dobry sposób są w stanie przewidzieć ocenę. Motywacją dla takiej analizy jest to, że opis zawiera więcej informacji niż sama ocena, a niektórzy klienci przypadkowo mogą kliknąć niezamierzoną ocenę. W recenzji tekstowej ryzyko takiego błędu jest diametralnie mniejsze. Taka analiza może być zatem następnie wykorzystana np. do wystawienia oceny danemu produktowi i później przysłużyć się w analizie hierarchizującej tytuły pod względem od najlepszego do najgorszego.

POTENCJALNE DALSZE KROKI

Jest jeszcze wiele ciekawych możliwości jakie mogłyby być potencjalnie zastosowane w tym projekcie. Poniżej opisano kilka z nich, które nasunęły się jako wnioski podczas pisania tej pracy:

1. W fazie pozyskiwania danych, bardzo ciekawym rozszerzeniem mogłoby być zbudowanie procesu pozyskiwania danych strumieniowych, tak aby można byłoby je analizować w czasie rzeczywistym. Technologiami, które mogłyby być użyte w tym celu są np. Spark Streaming lub Kafka.

2. Zautomatyzowanie części obecnego procesu, które były wykonywane teraz manualnie – np. zapisywania danych w HDFS

3. Opisane w sekcji *ROZDZIAŁ IV: Modelowanie*; *PODROZDZIAŁ 1: HIVE* użycie estymacji Bayes'a do stworzenia bardziej stabilnego i usystematyzowanego narzędzia do tworzenia rankingów pozycji książkowych tak, żeby brane były pod uwagę zarówno sama recenzja, jak i liczba recenzji.

4. Wypróbowanie nowszych metod istniejących do zapisywania danych tekstowych w postaci wektorowej jak np. ELMO i BERT. Byłoby to jednak bardziej rozszerzenie projektu pod kątem dziedziny przetwarzania języka naturalnego, a zatem bardziej wpadające w kategorię data science i uczenia maszynowego aniżeli Big Data.

5. Rozszerzenie projektu o zamodelowanie relacji w danych za pomocą grafowych baz danych np. Neo4j. Mogłoby to pomóc wyciągnąć nowe wnioski, które niekoniecznie są widoczne w momencie, gdy dane analizowane są w sposób tabelaryczny.

BIBLIOGRAFIA

Amazon AWS, brak daty *JSON SerDe Libraries.* [Online]
Dostęp pod adresem: https://docs.aws.amazon.com/athena/latest/ug/json-serde.html
[Data uzyskania dostępu: 05 2021].

Benjamin Bengfort, D. D. L., 2017. *Computing a Bayesian Estimate of Star Rating Means.* [Online]
Dostęp pod adresem: https://medium.com/district-data-labs/computing-a-bayesian-estimate-of-star-rating-means-651496a890ab
[Data uzyskania dostępu: 3 06 2021].

Data-Flair Trainings Blog, brak daty *Hive Data Types – Primitive and Complex Data Types in Hive.* [Online]
Dostęp pod adresem: https://data-flair.training/blogs/hive-data-types/
[Data uzyskania dostępu: 23 05 2021].

Dean, J., 2016. *"Large-Scale Deep Learning for Intelligent Computer Systems".* Seul: Google.

Ganesan, K., 2019. *All you need to know about text preprocessing for NLP and Machine Learning.* [Online]
Dostęp pod adresem: https://towardsdatascience.com/all-you-need-to-know-about-text-preprocessing-for-nlp-and-machine-learning-bc1c5765ff67
[Data uzyskania dostępu: 4 06 2021].

Geron, A., 2019. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* 2 ed. s.l.:O'Reilly.

Ian Robinson, J. W. E. E., 2015. *Graph Databases - New Opportunities For Connected Data.* 2 red. Sebastopol: O'Reilly Media, Inc..

Jeffrey Pennington, R. S. C. D. M., 2014. *GloVe: Global Vectors for Word Representation.* [Online]
Dostęp pod adresem: https://nlp.stanford.edu/projects/glove/
[Data uzyskania dostępu: 4 06 2021].

Jianmo Ni, J. L. J. M., 2019. Justifying recommendations using distantly-labeled reviews and fined-grained aspects; Empirical Methods in Natural Language Processing (EMNLP).

JSON Lines Org., brak daty *JSON Lines; Documentation for the JSON Lines text file format.* [Online]
Dostęp pod adresem: https://jsonlines.org/
[Data uzyskania dostępu: 05 2021].

Kane, F., 2017. *Taming Big Data with Apache Spark and Python: Real-world examples to help you analyze large datasets with Apache Spark.* s.l.:Packt.

Kane, F., brak daty *The Ultimate Hands-On Hadoop: Tame your Big Data!,* brak miejsca: brak nazwiska

Marek Gągolewski, M. B. A. C., 2016. *Przetwarzanie i analiza danych w języku Python.* 1 red. Warszawa: Wydawnictwo Naukowe PWN SA.

Masurel, P., 2013. *Of bayesian average and star ratings.* [Online]
Dostęp pod adresem: https://fulmicoton.com/posts/bayesian_rating/
[Data uzyskania dostępu: 03 06 2021].

Ng, A., 2015. *What data scientists should know about deep learning.* brak miejsca, brak nazwiska

Ni, J., 2018. [Online]
Dostęp pod adresem:
http://deepyeti.ucsd.edu/jianmo/amazon/categoryFiles/Kindle_Store.json.gz
[Data uzyskania dostępu: 05 2021].
Ni, J., 2018. [Online]
Dostęp pod adresem:
http://deepyeti.ucsd.edu/jianmo/amazon/metaFiles2/meta_Kindle_Store.json.gz
[Data uzyskania dostępu: 05 2021].
Szmit, R., brak daty *Apache Hadoop - prezentacja na studiach podypomowych "Big Data"*, brak miejsca: brak nazwiska
Szmit, R., brak daty *Wprowadzenie do Big Data*. brak miejsca, brak nazwiska, p. 4.
The Apache Software Foundation, brak daty *https://hive.apache.org/*. [Online]
[Data uzyskania dostępu: 06 2021].
Wikipedia, brak daty *https://en.wikipedia.org/wiki/Apache_Hive*. [Online]
[Data uzyskania dostępu: 05 2021].

DODATKI

DODATEK 1: Kod źródłowy generujący mniejsze pliki w języku Python

https://github.com/ambiernat/Big_Data/blob/main/File_Generator-OOP.ipynb

DODATEK 2: Wczytywanie danych z HDFSa do Apache Hive

https://github.com/ambiernat/Big_Data/blob/main/read_data_from_HDFS_to_Hive.sql

DODATEK 3: Ponowne połączenie danych Kindle i meta_Kindle w pojedyncze pliki w Apache Hive

https://github.com/ambiernat/Big_Data/blob/main/read_data_from_HDFS_to_Hive.sql

DODATEK 4: Exploracja Danych w Hive

https://github.com/ambiernat/Big_Data/blob/main/HIVE_4_exploratory_hive.sql

DODATEK 5: Wstępne przygotowanie danych Kindle

https://github.com/ambiernat/Big_Data/blob/main/HIVE_3_kindle_preproc.sql

DODATEK 6: Wstępne przygotowanie danych meta_Kindle

https://github.com/ambiernat/Big_Data/blob/main/HIVE_5_meta_kindle_preproc.sql

DODATEK 7: Połączenie dwóch zbiorów oraz przygotowanie danych do dalszej analizy w Hive, Spark Scala i Python

https://github.com/ambiernat/Big_Data/blob/main/HIVE_6_merge_datasets.sql

DODATEK 8: Duplikaty – ASIN i category

q2.asin	q2.catego	q2.counted
B003P2PH	solitaire	4
B003P2PH	activities,	4
B003P2PH	humor & e	4
B003P2PH	card game	4
B003P2QC	humor & e	3
B003P2QC	word gam	3
B003P2QC	activities,	3
B003P2T6	activities,	3
B003P2T6	humor & e	3

DODATEK 9: Ostateczne czyszczenie danych do analizy w Hive

https://github.com/ambiernat/Big_Data/blob/main/HIVE_7_preprocess4hive.sql

DODATEK 10: Zapytania wynikowe w Hive

https://github.com/ambiernat/Big_Data/blob/main/HIVE_8_hive_rankings.sql

DODATEK 11: Hive – wszystkie wyniki

B0102NDRKY	one hour (33-43 pages)	14	7 Steps to Self-Confidence	4.85714	1.580450376	2.63905733	9.506139413	2.302585093	0.336472237	7.203554320	0.046709197	0.714012759	0.226873242	3.468061973	3.468061973
B00N5UEGZE	parenting & relationship	207	Big Love: the practice of living beyond your limits - Kindle edition	4.93237	1.595819025	5.332718793	9.506139413	2.302585093	3.030133700	7.203554320	0.420644249	0.826193275	2.074771878	4.075088568	4.075088568
B00KSSYR5O	politics & social sciences	6660	When Breath Becomes Air - Kindle edition	4.74264	1.5565945	8.803874764	9.506139413	2.302585093	6.501289671	7.203554320	0.902511369	0.970753411	4.280288904	4.603936521	4.603936521
B00AGFU5V5	reference	560	APC: Author, Publisher, Entrepreneur- How to Publish a Book - Kindle edition	4.70536	1.548701677	6.327936784	9.506139413	2.302585093	4.025351691	7.203554320	0.558800769	0.867640231	2.629357188	4.082557156	4.082557156
B00ET7Q4LE	religion & spirituality	1759	The 7-Day Prayer Warrior Experience (One One-Week Devotional) - Kindle edition	4.70438	1.548493456	7.472500745	9.506139413	2.302585093	5.169915652	7.203554320	0.717689549	0.915306865	3.376282558	4.305949008	4.305949008
B01CZCW26K	romance	1598	Island of Glass (The Guardians Trilogy Book 1) - Kindle edition	4.77597	1.563597087	7.376508126	9.506139413	2.302585093	5.073923033	7.203554320	0.704363819	0.911309146	3.364020444	4.352385107	4.352385107
B00G3L19ZU	science & math	572	Numbers: How to Excel at Math and Science (Even If You Flunked Algebra),TorchenPerigee,Careers - General,General,APPLIED MATHEMATICS,Advice on careers & achieving success,Applied,BUSINESS & ECONOMICS / Careers / General,Business & Economics,CAREER & PERSONAL DEVELOPMENT,Career,Career Development,Career/Job,Careers - General,Careers-Jobs,Educational psychology,Educational psychology ,GENERAL,General (Adult),MATHEMATICS/ Applied,Math anxiety,Math anxiety,,Mathematics,Mathematics - Study and teaching - Psychological aspects,Mathematics / Probability & Statistics / General,Mathematics,Study and teaching,Psychological aspects,,Non-Fiction,Personal Growth - Success,Philosophy & Social Aspects,Psychological aspects,SCIENCE (GENERAL),SCIENCE/ General,SELF-HELP / Personal Growth / Success,STUDY AIDS / Study & Test-Taking Skills,Science,Philosophy & Social Aspects,Self-Help/Personal Growth - Success,Study and teaching,United States,,learning how to learn; business; learning; psychology; neuroscience; math; science; education; math books;	4.6486	1.5365664	6.349138991	9.506139413	2.302585093	4.046553898	7.203554320	0.561744067	0.86852322	2.611324254	4.037418255	4.037418255
B00GIUG3E5	science fiction & fantasy	3438	Red Rising	4.60239	1.526574671	8.142645186	9.506139413	2.302585093	5.840060093	7.203554320	0.810719241	0.943215772	3.731242163	4.341042224	4.341042224
B00A73F7PI	self-help	252	The Complete Works of Florence Scovel Shinn (Dover Empower Your Life) - Kindle edition	4.87698	1.584527022	5.529429088	9.506139413	2.302585093	3.226843995	7.203554320	0.447951643	0.834385493	2.184653054	4.069284805	4.069284805
B01F9FAJBA	sports & outdoors	198	The Law of Self Defense, 3rd Edition eBook	4.86364	1.581786381	5.288267031	9.506139413	2.302585093	2.985681938	7.203554320	0.41447344	0.824342032	2.015848094	4.009299883	4.009299883
B00EFWUB1E	teen & young adult	3307	City of Heavenly Fire (The Mortal Instruments Book 6) eBook	4.76142	1.560544931	8.103796713	9.506139413	2.302585093	5.80121162	7.203554320	0.805326282	0.941597885	3.834492786	4.483338462	4.483338462
B00F0B993Q	travel	95	The Photographer's Guide to Chicago	4.84211	1.577349598	4.553876892	9.506139413	2.302585093	2.251291799	7.203554320	0.312525137	0.793757541	1.513279609	3.843457567	3.843457567
B003JTHMD6	two hours or more (65-1	554	Shattering Hearts (Virgin River Book 11) eBook	4.32671	1.464808548	6.317164687	9.506139413	2.302585093	4.014579594	7.203554320	0.557305382	0.867191615	2.411301446	3.752090795	3.752090795

DODATEK 12: Modele MLR i GBT na próbce danych w Sparku

https://github.com/ambiernat/Big_Data/blob/main/SparkML_Scala_BD_MLR_GBT_small.ipynb

DODATEK 13: Model głębokich sieci neuronowych z użyciem zmiennej 'reviewtext' dla 5000 danych.

https://github.com/ambiernat/Big_Data/blob/main/NLP_small_data_tidy1.ipynb

DODATEK 14: Model sieci neuronowych z użyciem zmiennych 'reviewtext' i 'category' dla 5000 danych.

https://github.com/ambiernat/Big_Data/blob/main/NLP_small_data_reviewtext_category.ipynb

DODATEK 15: Model głębokich sieci neuronowych z użyciem zmiennej 'reviewtext' dla 500,000 danych.

https://github.com/ambiernat/Big_Data/blob/main/NLP_big_data_500k_final_GloVe.ipynb

DODATEK 16: Model MLR w Sparku nauczony na całych danych

https://github.com/ambiernat/Big_Data/blob/main/SparkML_Scala_BD_MLR_allData.ipynb