

Zespół: Szymon Gut, Maciej Orłowski

Cel projektu

Celem projektu jest przeprowadzenie kompleksowej analizy danych z portalu DataScience StackExchange oraz danych uzyskiwanych poprzez API z platformy Stack Overflow w celu zagłębienia wiedzy na temat trendów, preferencji użytkowników oraz najważniejszych tematów w obszarze informatyki, w tym data science, uczenia maszynowego oraz przetwarzania danych. Analiza wyszczególnionych źródeł danych ma charakter eksploracyjny w celu wyznaczenia ciekawych charakterystyk oraz wniosków, w dynamicznie zmieniającym się w sektorze IT.

Zagadnienia, które chcielibyśmy poruszyć w naszej analizie:

- Identyfikacja i analiza najbardziej popularnych tematów związanych z informatyką, w tym obszarami takimi jak data science, uczenie maszynowe, przetwarzanie danych, sztuczna inteligencja i inne;
- Analiza zmian w popularności tematów oraz preferencji użytkowników, takich jak preferowane języki programowania, narzędzia, technologie i frameworki w obszarze data science i uczenia maszynowego, na przestrzeni czasu, pozwalająca zidentyfikować ewolucję trendów w obszarze informatyki;
- Aktywności użytkowników taka jak liczba komentarzy, liczba głosów za oraz przeciw, w zależności od zakresu tematycznego posta;
- Ruch na portalu tj. Zbadanie jaki rozkład ma zakładanie nowych kont, jak wygląda rozkład liczby pytań oraz jak wygląda rozkład wyświetleń i odpowiedzi postów na przestrzeni lat.

Opis zbiorów danych wykorzystywanych w projekcie.

1. API StackExchange - dane pytań ze StackOverflow

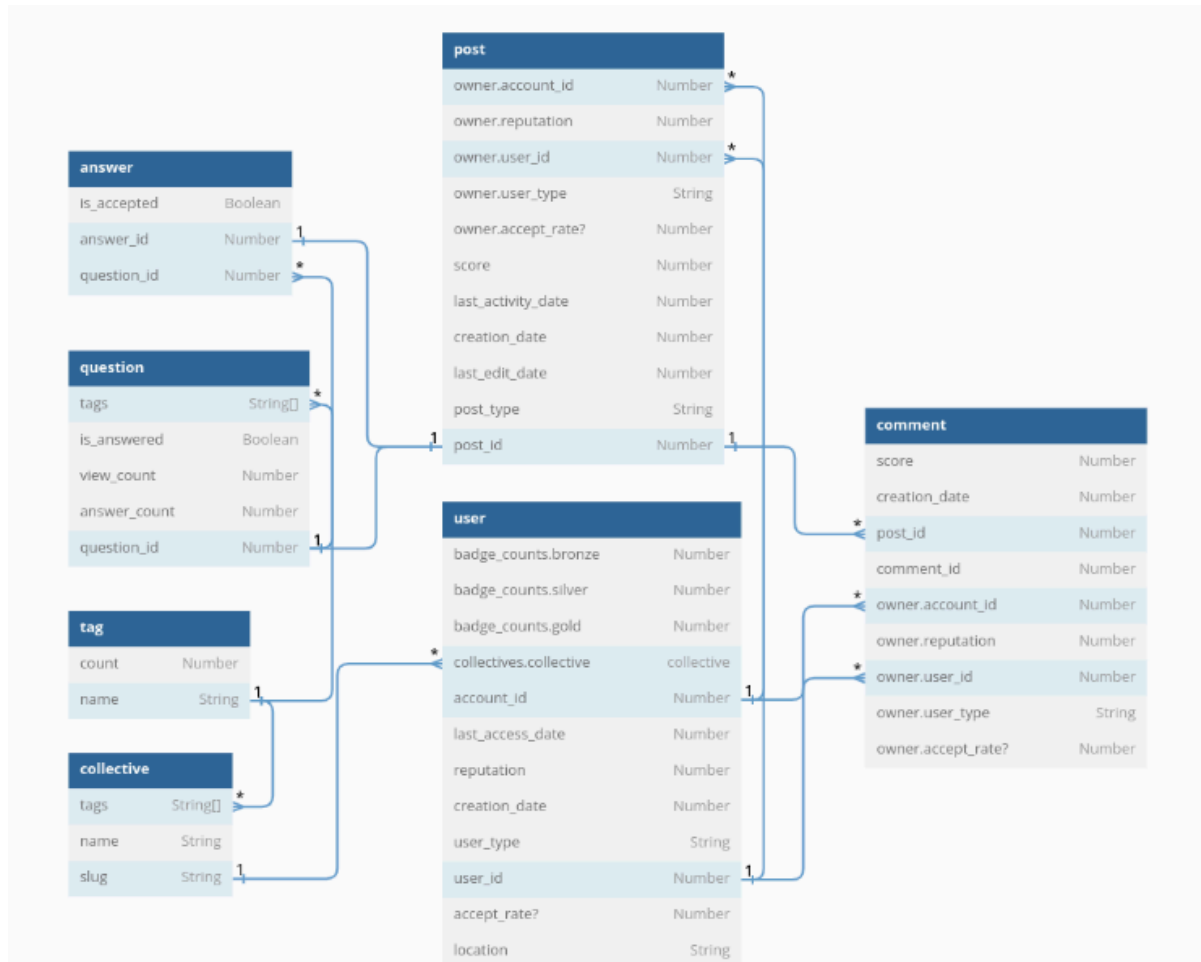
API StackExchange zawiera wiele endpointów umożliwiających pobranie informacji ([dokumentacja](#)). Najważniejsze z nich to:

- /answers - dane o odpowiedziach pod pytaniami;
- /questions - dane na temat pytań;
- /posts - kolektywne dane zawierające za równo pytania jak i odpowiedzi. Dane te są jednak mniej szczegółowe od danych pytań/odpowiedzi, gdyż zwracane obiekty zawierają jedynie wspólne atrybuty obiektów pytań i odpowiedzi;
- /badges - dane na temat odznak użytkowników;
- /collectives - dane na temat grup tematycznych dostępnych w serwisie;
- /comments - dane o komentarzach;
- /tags - informacje o tagach przypisywanych do pytań;
- /users - informacje na temat użytkowników.

Nasz projekt przede wszystkim będzie wykorzystywał informacje o pytaniach z serwisu StackOverflow poświęconego tematom z obszaru IT. Dane zwracane przez

API są w formacie JSON. Dane dotyczące pytań zawierają informacje takie jak wynik pytania, liczba wyświetleń, liczba odpowiedzi czy informacje o autorze. Każde z nich ma również przypisane tagi, które umożliwiają grupowanie podobnych pytań.

Najważniejsze dane wraz z połączeniami:



2. Dane statyczne z postami z Data Science StackExchange

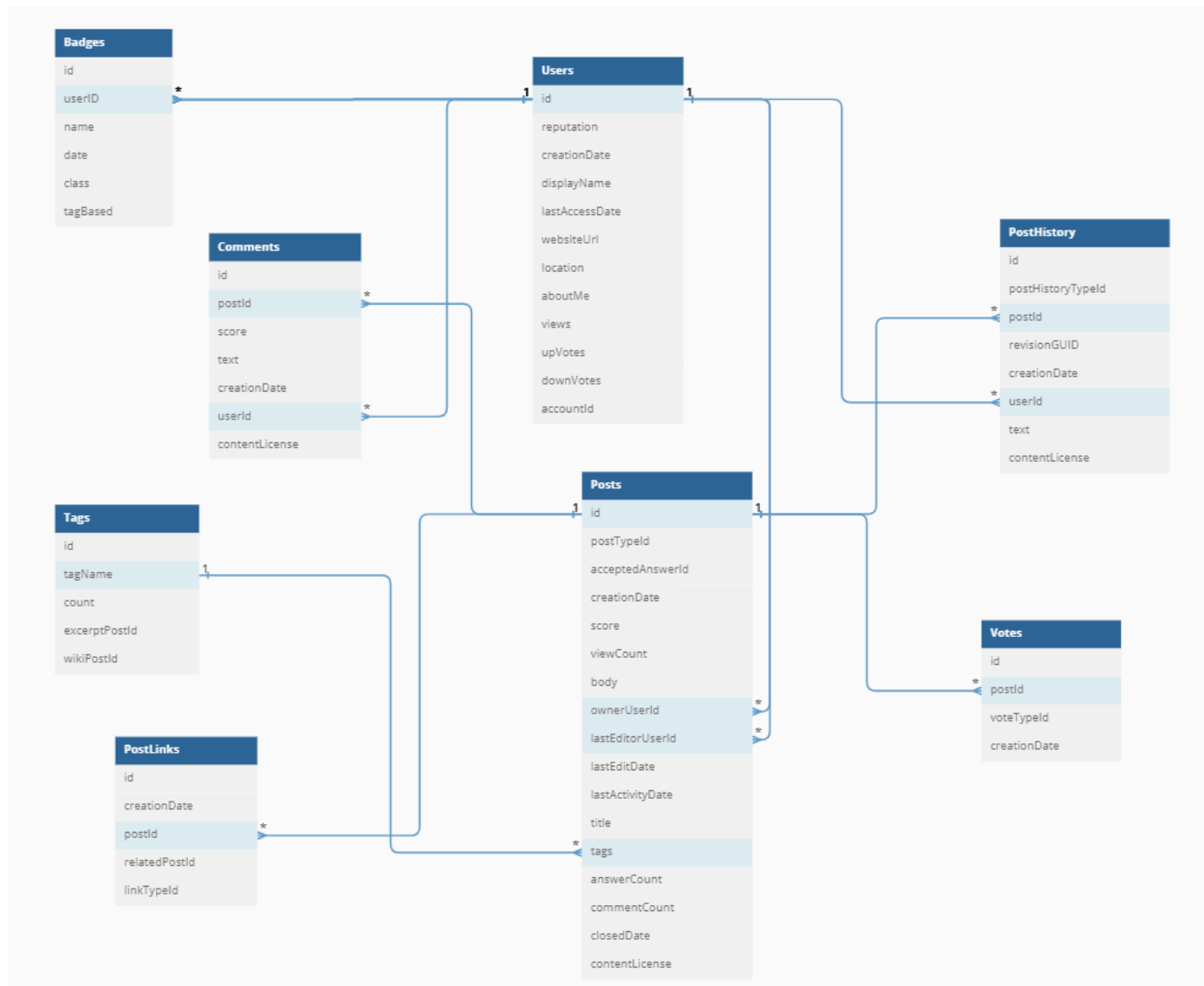
Dane zostały pobrane ze strony: <https://archive.org/details/stackexchange>. Jest to strona umożliwiająca pobranie wielu różnych danych z różnych serwisów StackExchange. Dane zawarte na tej stronie dostępne są od 2014-01-21 do 2023-12-08. Dane udostępniane są w archiwum .7z i zawierają 7 oddzielnych plików XML:

- **Badges** - zawiera informacje o odznakach (badges) przyznawanych użytkownikom za osiągnięcia i aktywność na platformie. Odznaki są specjalnymi wyróżnieniami dla użytkowników, które otrzymują za konkretne działania, takie jak udzielenie odpowiedzi otrzymanej przez dużą liczbę użytkowników, czy za zdobycie dużej liczby głosów pozytywnych na swoje pytania lub odpowiedzi.
- **Comments** - zawiera dane dotyczące komentarzy, które użytkownicy dodają do pytań, odpowiedzi i innych treści na platformie. Komentarze pozwalają

użytkownikom na wyrażanie swoich opinii, pytań lub dodawanie dodatkowych informacji do istniejących treści.

- PostHistory - zawiera historię zmian w postach, czyli pytaniach lub odpowiedziach. Zawiera informacje o tym, kiedy i jakie zmiany zostały wprowadzone do posta, w tym edycje, zamknięcia, otwarcia, usuwania, itp.
- PostLinks - zawiera informacje o linkach między różnymi postami na platformie. Może to obejmować linki do pytań, odpowiedzi, komentarzy lub innych treści.
- Posts - zawiera główne treści platformy, czyli pytania i odpowiedzi użytkowników. Zawiera treść pytania lub odpowiedzi, informacje o autorze, daty publikacji, ilości głosów, komentarzy itp.
- Tags - zawiera listę tagów (słów kluczowych) przypisanych do pytań na platformie. Tagi pomagają w kategoryzacji pytań według tematów, co ułatwia użytkownikom znalezienie odpowiednich treści.
- Users - zawiera informacje o użytkownikach platformy, takie jak identyfikator użytkownika, reputacja, data dołączenia do społeczności, liczba zdobytych głosów pozytywnych i negatywnych, itp.
- Votes - zawiera informacje o głosach, które użytkownicy przyznają pytaniom i odpowiedziom na platformie. Może to obejmować głosy pozytywne (upvotes), negatywne (downvotes) oraz inne akcje związane z głosowaniem.

Najważniejsze dane wraz z połączeniami:



Stos architektoniczny

Do zrealizowania projektu wykorzystane zostały następujące technologie:

- Apache NiFi - do pobierania danych online z API StackExchange oraz ładowanie danych statycznych z Data Science Stack Exchange
- Apache Hadoop - do składowania **master data setów**, tutaj składowane są dane bez żadnego wstępnego przetworzenia, dane są bezpośrednio czytane ze źródła i składowane tutaj w swojej pierwotnej postaci, aby zapobiec ewentualnej utracie danych podczas awarii
- Apache HBase - do składowania danych potrzebnych do analizy pochodzących z API StackExchange (w formacie JSON) oraz danych statycznych z Data Science Stack Exchange. Dane na tym etapie są już wyselekcjonowane oraz przetworzone (są również odpowiednio załadowane do odpowiednich rodzin kolumn). Znajdują się tu tylko dane potrzebne do przeprowadzanej w następnym kroku analizy. Dane oczywiście pochodzą z odpowiedniego master data setu (w zależności czy ładujemy

dane z Stackoverflow czy z Data Science Stack Exchange) znajdującego się w Hadoop.

- Apache Spark do wsadowej analizy zebranych danych, która następnie będzie wizualizowana przy pomocy pakietu matplotlib dostępnym w Python.

Całość kodu przechowywana jest na platformie GitHub. Link do repozytorium znajduje się [tutaj](https://github.com/maciejors/big_data) (https://github.com/maciejors/big_data).

Przechowywanie danych w Hadoop

Hadoop służy nam do przechowywania danych bezpośrednio na klastrze umożliwiającym szybki dostęp danych nieprzetworzonych. To tutaj znajdują się dane w swojej pierwotnej postaci pełniące rolę naszych master data setów. Zapobiega to niechcianej i tragicznej utracie danych podczas wystąpienia krytycznego błędu podczas ładowania danych przy Nifi. Przepływ danych w Nifi ładujący dane do Hadoop zawiera obsługę błędów, dzięki czemu przy wystąpieniu pewnej awarii, dane które nie zostały poprawnie załadowane, zostają przechwycone w buforze błędów i zapisywane do pliku z rozszerzeniem .txt.

Zalety używania HBase

Nasze dane zawierają dużo wartości opcjonalnych. Do takich wartości należy przykładowo pola z danych z StackOverflow: LastEditDate, AcceptRate. Do przechowywania takich danych idealnym rozwiązaniem wydaje się Apache Hbase. Poprzez mechanizm rodziny kolumn oraz zmiennej ilości kolumn dla każdego wiersza, nie mamy obowiązku przechowywania braków danych, na skutek czego w bardzo dużym stopniu optymalizowana jest pamięć zużywana na przechowywanie danych, potrzebnych do analizy.

Podejście do aktualizacji w tabeli Hbase:

Przy przechwytywaniu postów o tym samym id, możliwe były dwa najpopularniejsze podejścia. Pierwszym z nich byłoby zapisywanie nowych zmian z innym timestampem, aby śledzić zmienność naszych danych i zachowywać je w tabeli, lub aktualizować rekordy odnoszące się do tego samego pytania przechwyconego z API. Zdecydowaliśmy się na podejście drugie ustawiając klucz w tabeli HBase jako *question_id*, gdyż w naszej analizie nie są ważne zmiany rzędu: jak zmieniła się liczba wyświetleń posta pomiędzy aktualizacjami spływającymi poprzez API, lub czy powstały jakieś nowe odpowiedzi do tego posta, w czasie pomiędzy kolejnymi zapytaniami. Ważny u nas jest fakt, ile komentarzy oraz innych statystyk znajduje się w aktualnym czasie wywoływania analiz w Jupyterze, gdyż zawsze chcemy operować na najbardziej aktualnych danych i nie potrzebny jest nam dostęp do ich wcześniejszych wersji.

Okresowe przechwytywanie danych online z Stackoverflow

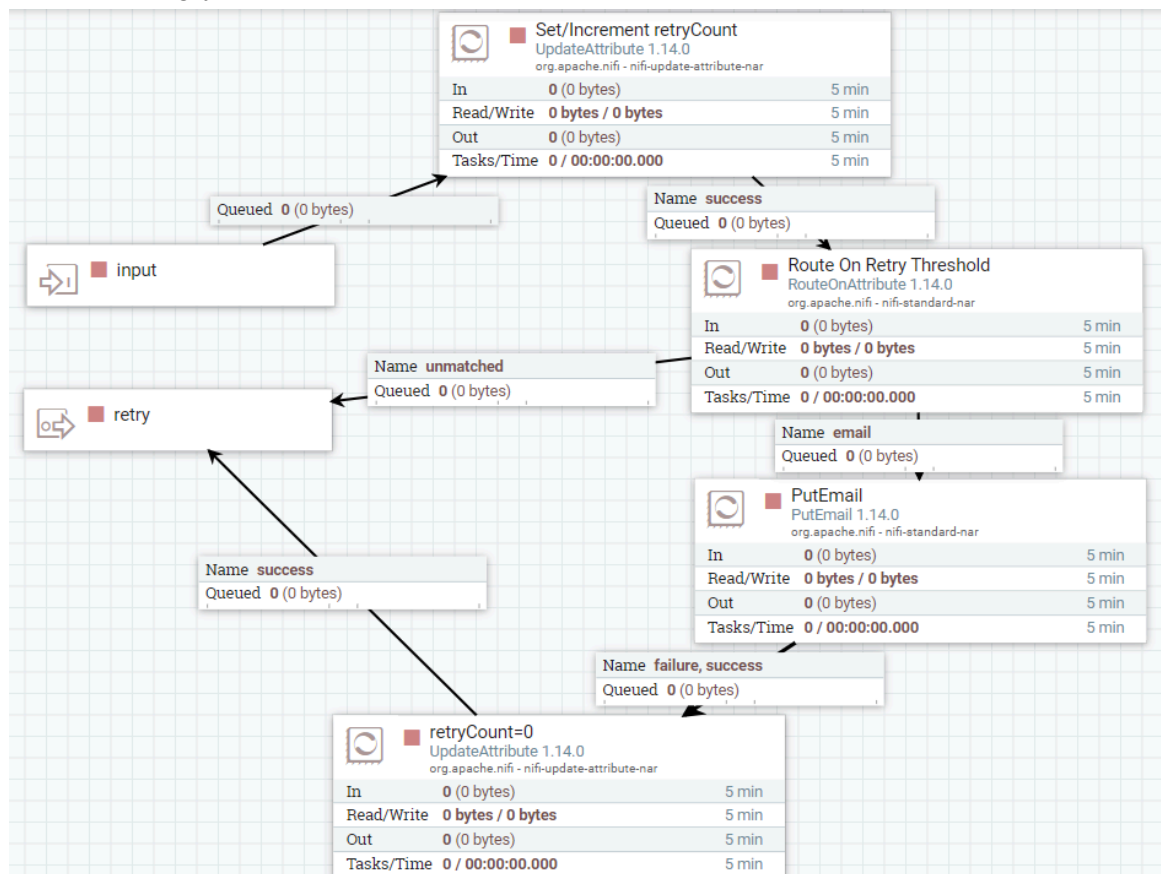
Ważnym aspektem jest przechwytywanie danych z serwisu Stackoverflow w sposób ujednolicony oraz okresowy, tak aby przechwytywać cały czas najnowsze dane z tego serwisu. W tym celu został napisany skrypt bashowy, komunikujący się z API tego serwisu i pobierający aktualne dane. W ramach jednego zapytania, poprzez ograniczenia wynikające z API otrzymujemy 30 ostatnich postów, a takie zapytanie jest wykonywane przez nas 4 razy dziennie. Scheduling takiego zapytania ustawiony jest w procesorze *Execute Processor* dostępnego w Nifi, gdzie wykonywany jest napisany przez nas skrypt komunikujący się z API.

Załadowanie dużych wolumenów danych statycznych

Z uwagi na duży rozmiar używanych przez nas danych statycznych niezbędne było ich podzielenie. W tym celu napisaliśmy skrypt w pythonie, który będzie dzielił je na mniejsze porcje oraz naprawiał formatowanie, było to niezbędne gdyż w pierwotnej postaci są one udostępniane w formacie .xml, więc aby nie było problemu z nieznymi tagami, musieliśmy również obsłużyć tę sytuację. Dane podzielone na mniejsze porcje są umieszczane w master data secie na klastrze w Apache Hadoop (oprócz dzielenia ich na mniejsze porcje żadne inne modyfikacje nie są przeprowadzane). Następnie odwołując się do danych przechowywanych na master data secie, pobieramy je z Hadoop i dokonujemy konwersji typu na json oraz dokonujemy niezbędnych transformacji. Dane już gotowe do analizy umieszczane są w dwóch tabelach HBase *datascience_posts* oraz *datascience_users*, odpowiadającym danym użytkownika oraz postom zamieszczanym na portalu.

Obsługa błędów przy ładowaniu danych

Główny nacisk został położony na obsługę błędów podczas ładowania danych do master data setu, dla danych spływających przez API online. Zdając sobie sprawę z możliwości potencjalnej utraty danych podczas awarii Hadoop, została dokonana szczególnie dokładna obsługa wystąpienia ewentualnych błędów. W ramach jakiegokolwiek awarii, błąd zostaje obsłużony przez specjalną grupę procesów, która w razie niepowodzenia ponawia próbę ściągnięcia i zapisania danych w Hadoop do **3 razy**, przy czym 4 niepowodzenie kończy się poinformowaniem użytkownika o wystąpieniu awarii, wraz z załączeniem niezapisanych rekordów drogą email.



Mniej zaawansowana obsługa błędów występowała przy zagadnieniach mniej priorytetowych z założeń funkcjonowania systemu. Podczas przetwarzania danych do widoków wsadowych umieszczanych w tabelach w Apache HBase oraz jednorazowego ładowania danych statycznych do master data setu do katalogu w Hadoop, obsługa błędów była skupiona na przechwyceniu rekordów, których nie udało się obsłużyć i zapisywanie ich do specjalnie wyznaczonych na tą ewentualność plików płaskich z rozszerzeniem .txt.

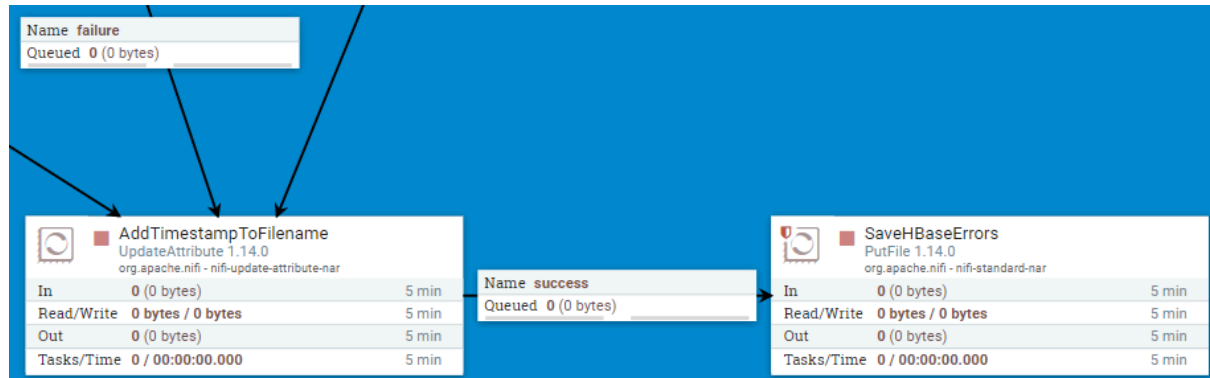
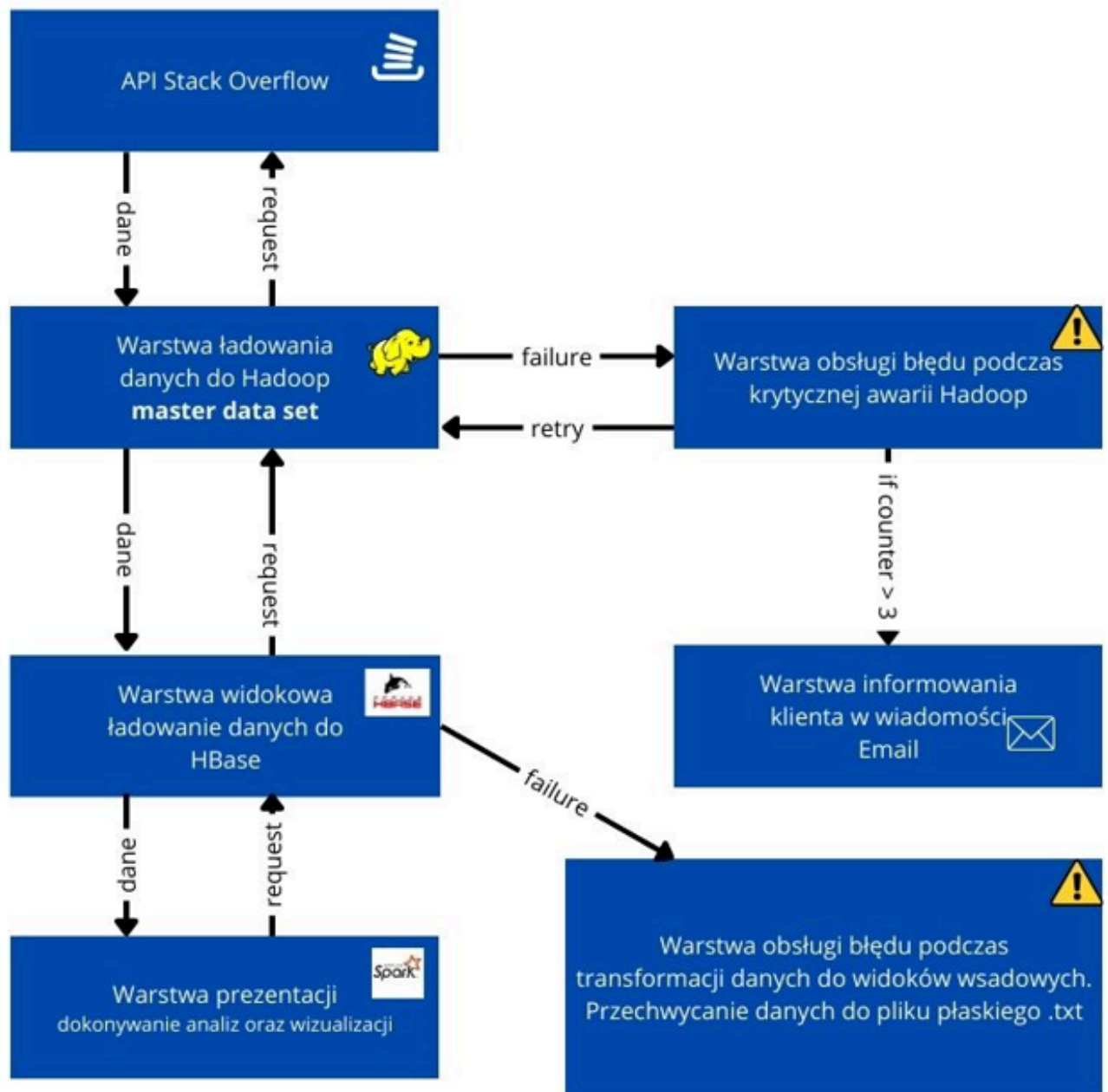
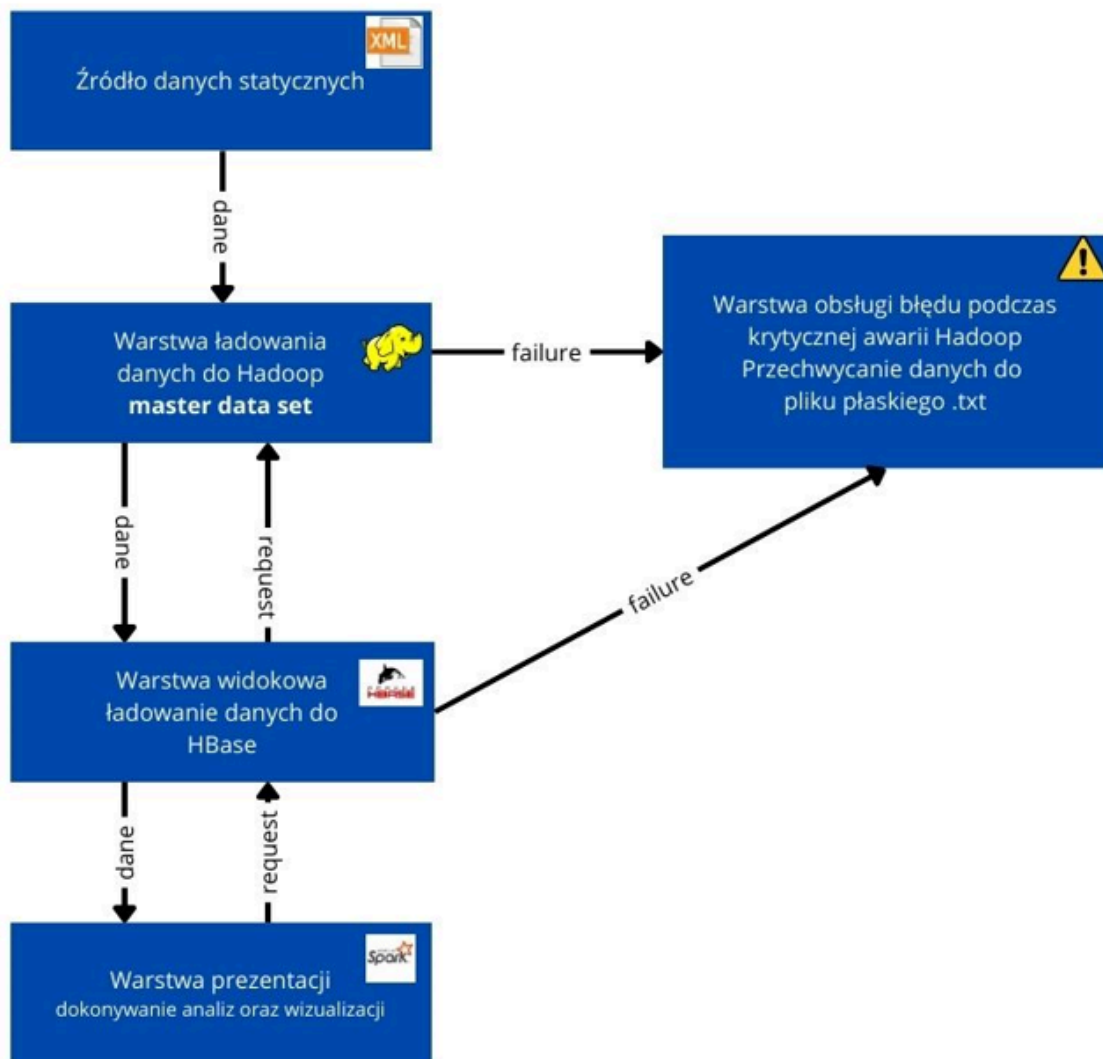


Diagram naszej architektury

Obsługa danych online



Obsługa danych statycznych



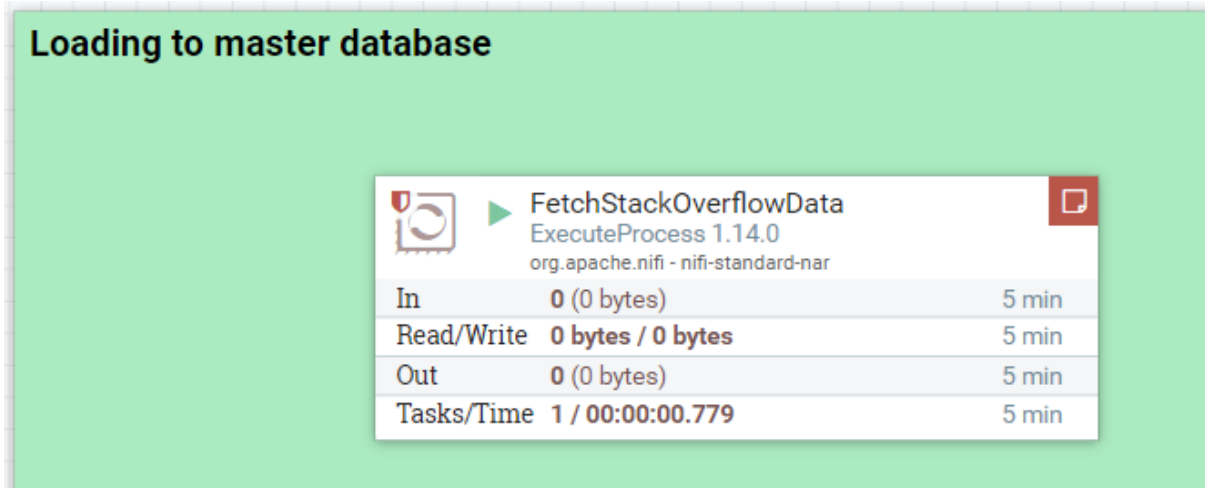
Testy funkcjonalności

Aby potwierdzić poprawność działania procesu ładowania danych przeprowadzony został szereg testów operacji pomiędzy wszystkimi warstwami.

Test 1: Pobieranie nowej porcji danych online

Celem testu jest weryfikacja czy skrypt pobierający nową porcję danych faktycznie pobiera i zapisuje niepuste nowe dane. Aby to zweryfikować, skrypt pobierający dane zostanie raz uruchomiony. Początkowo folder do którego zapisywane są nowe dane na maszynie lokalnej jest pusty. Oczekiwane jest, że po wykonaniu skryptu znajdować się w nim będzie jeden plik z danymi.

Potwierdzenie wykonania skryptu (błąd wyświetlony na komponencie jest nieistotny i nie oznacza że skrypt nie zadziałał - NiFi traktuje standardowy output komendy curl jako błąd, stąd to podświetlenie):



Loading to master database

| FetchStackOverflowData ExecuteProcess 1.14.0 org.apache.nifi - nifi-standard-nar | | |
|--|-------------------|-------|
| In | 0 (0 bytes) | 5 min |
| Read/Write | 0 bytes / 0 bytes | 5 min |
| Out | 0 (0 bytes) | 5 min |
| Tasks/Time | 1 / 00:00:00.779 | 5 min |

Zawartość katalogu docelowego po wykonaniu skryptu:

```
vagrant@node1:~/PROJECT$ ls -lsh data/stackoverflow/  
total 24K  
24K -rw-r--r-- 1 root root 22K Dec 18 09:17 questions-1702891025.json
```

Nowy niepusty plik rzeczywiście pojawił się w katalogu co potwierdza poprawność działania skryptu.

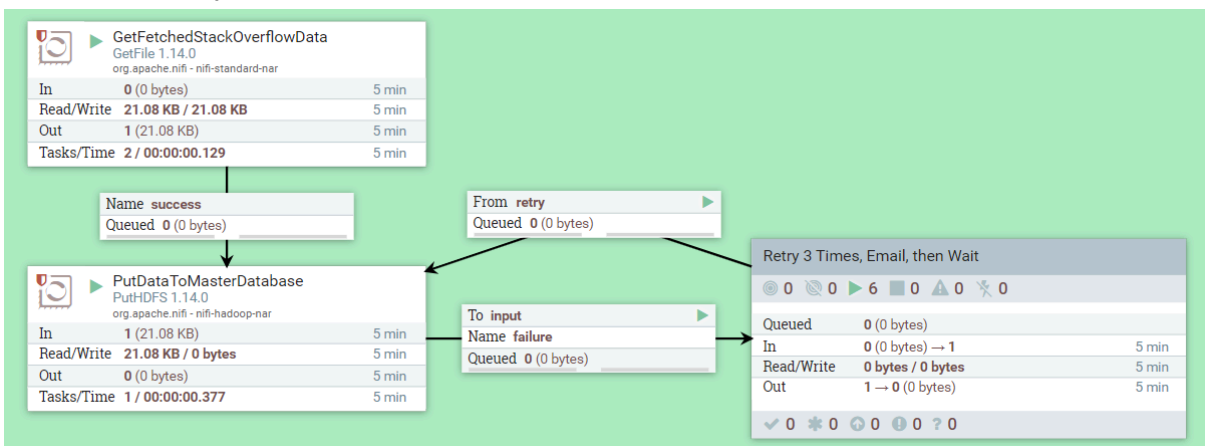
Test 2: Ładowanie nowych danych online do master database

Celem testu jest sprawdzenie czy nowo pobrane dane są poprawnie wczytywane do master database. W celu przeprowadzenia testu wykorzystane zostaną dane pobrane w poprzednim teście (plik *questions-1702891025.json*). Przed uruchomieniem komponentów NiFi odpowiedzialnych za wczytywanie nowych danych do HDFS, w master dataset znajduje się 1213 plików, wśród których nie ma jeszcze nowo pobranego pliku. Stan ten potwierdza zrzut ekranu poniżej:

```
vagrant@node1:~/PROJECT$ hdfs dfs -count /user/vagrant/PROJECT/online_master  
1      1213      26547105 /user/vagrant/PROJECT/online_master  
vagrant@node1:~/PROJECT$ hdfs dfs -find /user/vagrant/PROJECT/online_master -name questions-1702891025.json  
vagrant@node1:~/PROJECT$ |
```

Oczekiwane jest, że po uruchomieniu odpowiedniego flow, w master database pojawi się 1 nowy plik i będzie to właśnie plik pobrany w poprzednim teście.

Potwierdzenie wykonania flow:



Sprawdzenie zawartości master database po wykonaniu operacji:

```
vagrant@node1:~/PROJECT$ hdfs dfs -count /user/vagrant/PROJECT/online_master
1                1214                26568692 /user/vagrant/PROJECT/online_master
vagrant@node1:~/PROJECT$ hdfs dfs -find /user/vagrant/PROJECT/online_master -name questions-1702891025.json
/user/vagrant/PROJECT/online_master/questions-1702891025.json
vagrant@node1:~/PROJECT$
```

Jak widać, w HDFS jest teraz 1214 plików, czyli o 1 więcej, a także oczekiwany plik faktycznie jest teraz obecny w master database. Potwierdza to poprawność ładowania nowych danych.

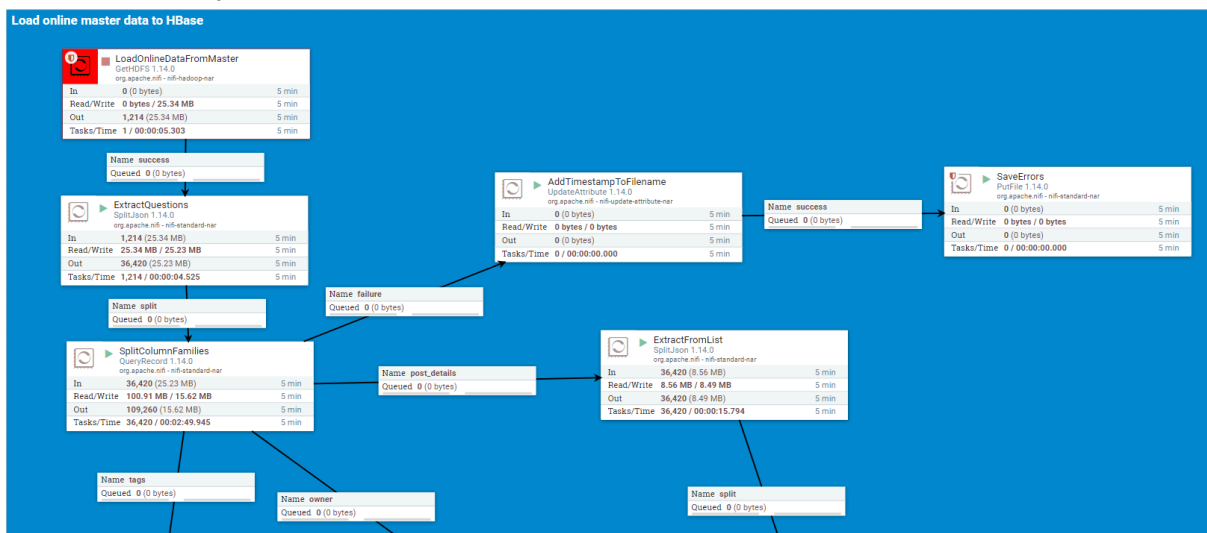
Test 3: Wczytywanie danych online z master database do warstwy widokowej

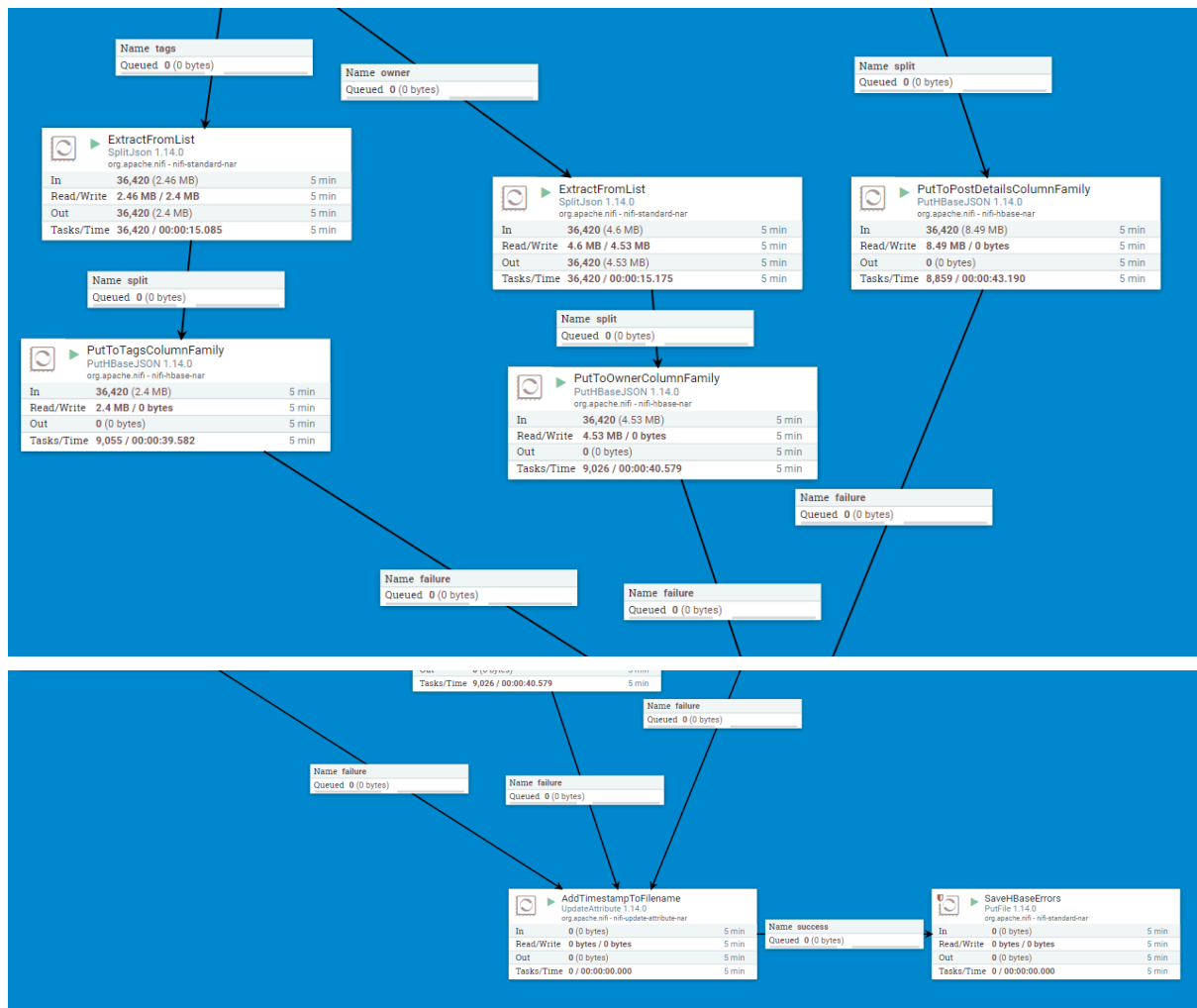
Celem testu jest sprawdzenie czy działa ładowanie danych online z master database do HBase. W tym celu uruchomiony zostanie odpowiedni flow w NiFi. Przed jego wykonaniem tabela docelowa w HBase jest pusta, a master database wypełniony jest danymi tak jak na koniec poprzedniego testu. Po wykonaniu flow oczekuje się, że dane w master database pozostaną nienaruszone, zaś tabela w HBase zostanie wypełniona rekordami.

Stan tabeli przed uruchomieniem flow w NiFi:

```
hbase(main):012:0> count 'stackoverflow'
0 row(s)
Took 0.0349 seconds
=> 0
```

Potwierdzenie wykonania flow:





Stan tabeli po wykonaniu flow (komenda `count 'stackoverflow'` - z uwagi na duży rozmiar informacji zwrotnej komendy umieszczona poniżej została tylko końcówka wydruku komendy informująca o liczbie wierszy):

```
Took 3.7190 seconds
=> 36420
```

Stan master database po wykonaniu flow:

```
vagrant@node1:~/PROJECT$ hdfs dfs -count /user/vagrant/PROJECT/online_master
1 1214 26568692 /user/vagrant/PROJECT/online_master
```

Zgodnie z oczekiwaniami, dane zostały wczytane do tabeli w HBase bez naruszania ich w master database co potwierdza poprawność flow.

Test 4: Wczytywanie danych online z warstwy widokowej do warstwy prezentacji

Celem testu jest potwierdzenie poprawności wczytywania danych z HBase do PySpark'owej ramki danych. W tym celu uruchomiony zostanie fragment kodu odpowiedzialny za wczytywanie tych danych w głównym notebooku z wynikami. Oczekuje się, że dane zostaną wczytane poprawnie oraz w pełni - tzn. liczba wierszy taka sama jak w HBase oraz brak znaczących problemów z danymi po wczytaniu (takich jak niepoprawnie wczytana kolumna bez danych itp.)

Cały kod odpowiedzialny za wczytywanie danych zawarty jest w notebooku z analizą danych. Można go rozbić na poszczególne kroki:

1. Zczytanie wierszy z HBase i zapisanie ich do listy. Jednocześnie do osobnej listy zapisywane są informacje o tagach dla każdego pytania;
2. Stworzenie ramki danych Pandas z listy wierszy z HBase stworzonej w poprzednim kroku. Osobno tworzona jest też ramka danych dla listy z informacjami o tagach;
3. Konwersja ramek danych stworzonych w poprzednim kroku na ramkę danych PySpark.

Schema ramki danych PySpark po wczytaniu danych:

```
df_stackoverflow.printSchema()

root
|-- owner_account_id: double (nullable = true)
|-- owner_reputation: double (nullable = true)
|-- owner_user_id: double (nullable = true)
|-- owner_user_type: string (nullable = true)
|-- post_answer_count: long (nullable = true)
|-- post_creation_date: date (nullable = true)
|-- post_is_answered: boolean (nullable = true)
|-- post_last_activity_date: date (nullable = true)
|-- post_score: long (nullable = true)
|-- post_title: string (nullable = true)
|-- post_view_count: long (nullable = true)
|-- question_id: long (nullable = true)
```

Pierwsze 5 wierszy tabeli:

```
df_stackoverflow.show(n=5)

23/12/18 10:51:29 WARN scheduler.TaskSetManager: Stage 0 contains a task of very large size (2414 KiB). The maximum recommended task size is 1000 KiB.
[Stage 0:> (0 + 1) / 1]
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|owner_account_id|owner_reputation|owner_user_id|owner_user_type|post_answer_count|post_creation_date|post_is_answered|post_last_activity_date|post_score|
|post_title|post_view_count|question_id|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|117126.0|545.0|306857.0|registered|1|2023-02-12|true|2023-02-13|0|
|Echo input into o...|106|75430619|registered|0|2023-02-12|true|2023-02-12|0|
|2.4174919E7|13.0|1.8136532E7|registered|0|2023-02-12|true|2023-02-12|0|
|How to compute im...|60|75430621|registered|0|2023-02-12|true|2023-02-12|1|
|1.8517358E7|100.0|1.3491206E7|registered|0|2023-02-12|true|2023-02-12|0|
|Flutter QuillEdit...|199|75430622|registered|0|2023-02-12|true|2023-07-11|0|
|2.7769408E7|1.0|2.1199996E7|registered|0|2023-02-12|true|2023-02-12|0|
|Unable to render ...|121|75430626|registered|0|2023-02-12|true|2023-02-12|0|
|2.7741092E7|1.0|2.1177532E7|registered|0|2023-02-12|true|2023-02-12|0|
|Maya Python GUI -...|161|75430628|
```

Liczba wierszy:

```
df_stackoverflow.count()

23/12/18 10:51:31 WARN sc

36420
```

Tabela z tagami:

```
df_stackoverflow_tags.printSchema()

root
 |-- question_id: long (nullable = true)
 |-- tag: string (nullable = true)
```

```
df_stackoverflow_tags.show(n=10)
```

```
+-----+-----+
|question_id|          tag|
+-----+-----+
|  75430619|           c|
|  75430619|         input|
|  75430619|           io|
|  75430619|       user-input|
|  75430619|   io-redirection|
|  75430621|           r|
|  75430621|       bayesian|
|  75430621|vector-auto-regre...|
|  75430622|         flutter|
|  75430622|         quill|
```

Wstępny przegląd danych nie wykazał żadnych problemów z nimi, zaś liczba wierszy we wczytanej tabeli jest równa liczbie wierszy w tabeli w HBase. Tagi zostały wczytane do osobnej tabeli i również w tym przypadku podgląd tabeli nie wykazuje żadnych znaczących problemów. Można zatem stwierdzić, że otrzymano oczekiwany rezultat a zatem dane są poprawnie ładowane.


Test 5: Rozpakowanie i wstępna obróbka danych statycznych

Celem testu jest weryfikacja czy dane statyczne są poprawnie rozpakowywane i partycjonowane. Wstępne partycjonowanie jest konieczne, gdyż w innym wypadku pliki te są za duże do przetworzenia przez NiFi i zwracane są błędy związane z brakiem pamięci. Na początku w folderze z danymi znajduje się jedynie archiwum ze spakowanymi danymi statycznymi. Oczekuje się, że po wykonaniu skryptu w folderze będą znajdować się dodatkowo:

- Folder users_split wypełniony partycjami danych o użytkownikach,
- Folder posts_split wypełniony partycjami danych o postach,
- Folder posthistory_split wypełniony partycjami danych o historii zmian w postach,
- Pozostałe niepodzielone pliki XML.

Potwierdzenie że skrypt został wykonany:

Loading static master data

 **decompress_static.sh**
ExecuteProcess 1.14.0
org.apache.nifi - nifi-standard-nar

| | | |
|------------|---------------------|-------|
| In | 0 (0 bytes) | 5 min |
| Read/Write | 0 bytes / 566 bytes | 5 min |
| Out | 0 (0 bytes) | 5 min |
| Tasks/Time | 1 / 00:01:20.566 | 5 min |

Zawartość folderu z danymi statycznymi po wykonaniu skryptu:

```
vagrant@node1:~/PROJECT$ ls data/datascience/
Badges.xml      PostLinks.xml  Votes.xml      posthistory_split  users_split
Comments.xml    Tags.xml       datascience.stackexchange.com.7z  posts_split
```

Zawartości poszczególnych folderów z partycjami dużych plików:

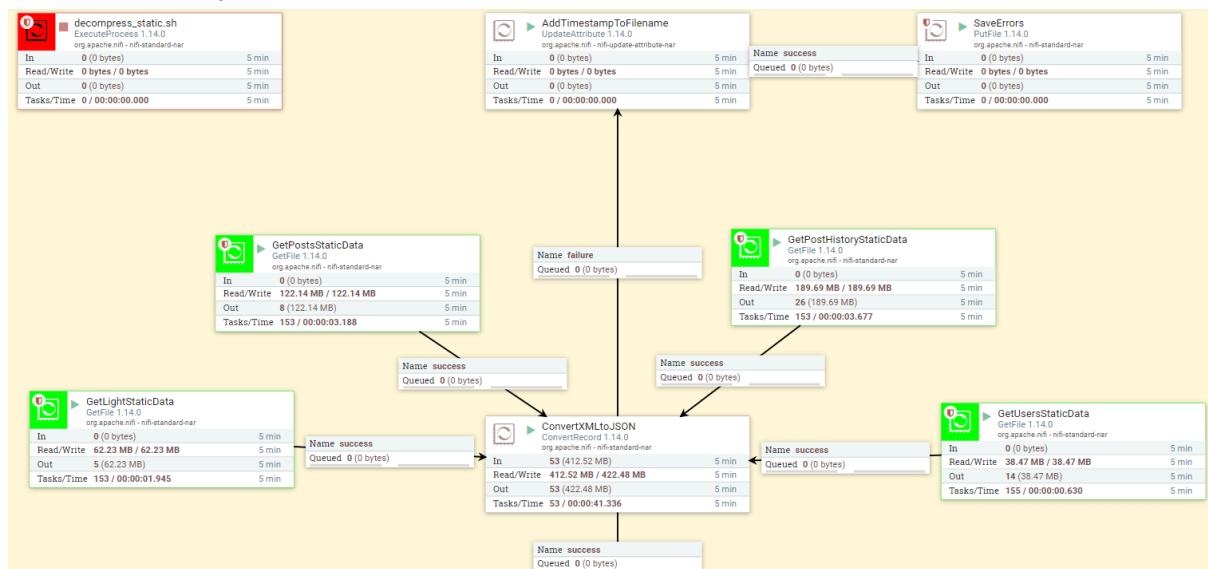
```
vagrant@node1:~/PROJECT$ ls data/datascience/posts_split/
posts-113744.xml  posts-31392.xml  posts-5.xml      posts-77106.xml
posts-16619.xml  posts-46411.xml  posts-62363.xml  posts-94765.xml
vagrant@node1:~/PROJECT$ ls data/datascience/posthistory_split/
posthistory-112261.xml  posthistory-192442.xml  posthistory-289752.xml  posthistory-387364.xml  posthistory-81559.xml
posthistory-127799.xml  posthistory-207101.xml  posthistory-306312.xml  posthistory-404884.xml  posthistory-96053.xml
posthistory-144390.xml  posthistory-222885.xml  posthistory-326823.xml  posthistory-421288.xml
posthistory-160826.xml  posthistory-239308.xml  posthistory-32801.xml   posthistory-48212.xml
posthistory-175961.xml  posthistory-255363.xml  posthistory-346449.xml  posthistory-66864.xml
posthistory-17960.xml   posthistory-274200.xml  posthistory-367360.xml  posthistory-7.xml
vagrant@node1:~/PROJECT$ ls data/datascience/users_split/
users--1.xml      users-118183.xml  users-141394.xml  users-15594.xml  users-40294.xml  users-62643.xml  users-84874.xml
users-107069.xml  users-131301.xml  users-152433.xml  users-28160.xml  users-51491.xml  users-73773.xml  users-95962.xml
vagrant@node1:~/PROJECT$
```

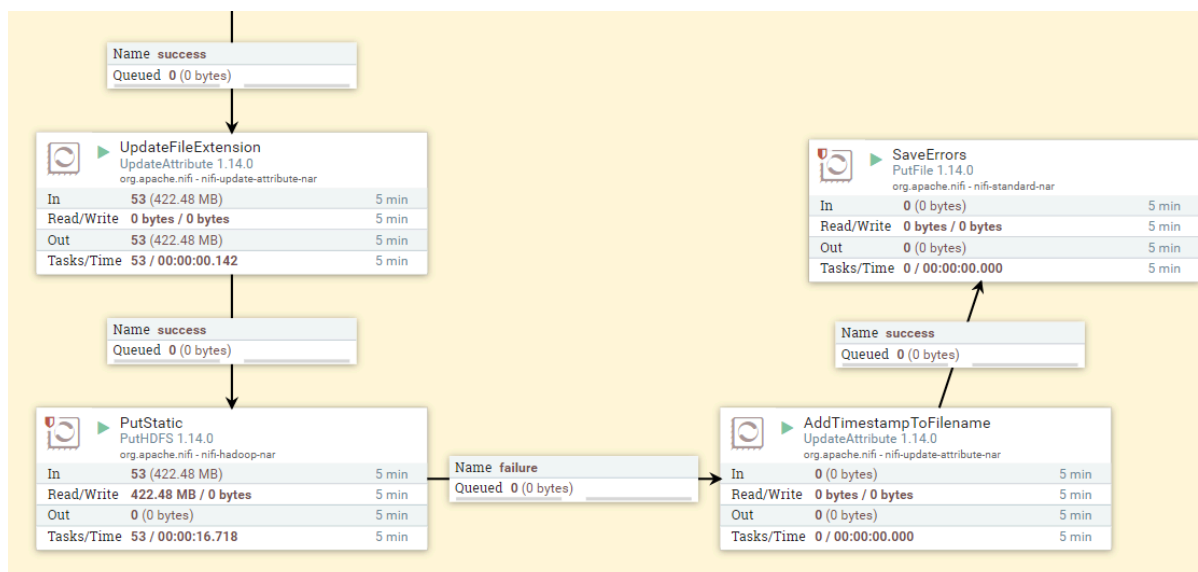
Zawartość folderów pokrywa się z oczekiwaniami co potwierdza poprawność działania skryptu.

Test 6: Konwersja danych statycznych do JSON i składowanie ich w master database

Celem testu jest sprawdzenie czy rozpakowane dane są poprawnie przenoszone do HDFS. Odpowiedzialny za to flow konwertuje je również na JSON, dlatego też w takim formacie spodziewane są dane w master database. Oczekiwany rezultat jest obecność danych rozpakowanych w poprzemianku w odpowiednim folderze na HDFS. Przed rozpoczęciem testu część master database odpowiedzialna za dane statyczne jest pusta - dane statyczne ładowane są jednorazowo i w całości.

Potwierdzenie wykonania flow w NiFi:





Sprawdzenie zawartości katalogu docelowego w HDFS (z wylistowania plików zamieszczony niżej jest jedynie fragment z uwagi na duży rozmiar wyniku komendy):

```

vagrant@node1:~/PROJECT$ hdfs dfs -ls /user/vagrant/PROJECT/static_master
Found 53 items
-rw-r--r-- 1 root supergroup 16394739 2023-12-18 12:08 /user/vagrant/PROJECT/static_master/Badges.json
-rw-r--r-- 1 root supergroup 29019837 2023-12-18 12:08 /user/vagrant/PROJECT/static_master/Comments.json
-rw-r--r-- 1 root supergroup 396939 2023-12-18 12:08 /user/vagrant/PROJECT/static_master/PostLinks.json
-rw-r--r-- 1 root supergroup 60234 2023-12-18 12:08 /user/vagrant/PROJECT/static_master/Tags.json
-rw-r--r-- 1 root supergroup 25997845 2023-12-18 12:09 /user/vagrant/PROJECT/static_master/Votes.json
-rw-r--r-- 1 root supergroup 7600643 2023-12-18 12:09 /user/vagrant/PROJECT/static_master/posthistory-112261.json
-rw-r--r-- 1 root supergroup 7655872 2023-12-18 12:09 /user/vagrant/PROJECT/static_master/posthistory-127799.json
-rw-r--r-- 1 root supergroup 8808710 2023-12-18 12:09 /user/vagrant/PROJECT/static_master/posthistory-144390.json
-rw-r--r-- 1 root supergroup 8194008 2023-12-18 12:09 /user/vagrant/PROJECT/static_master/posthistory-160826.json
-rw-r--r-- 1 root supergroup 7800582 2023-12-18 12:09 /user/vagrant/PROJECT/static_master/posthistory-175961.json
  
```

Zawartość katalogu pokrywa się z oczekiwaniami co świadczy o poprawnym załadowaniu danych.

Test 7: Wczytywanie danych statycznych z master database do warstwy widokowej

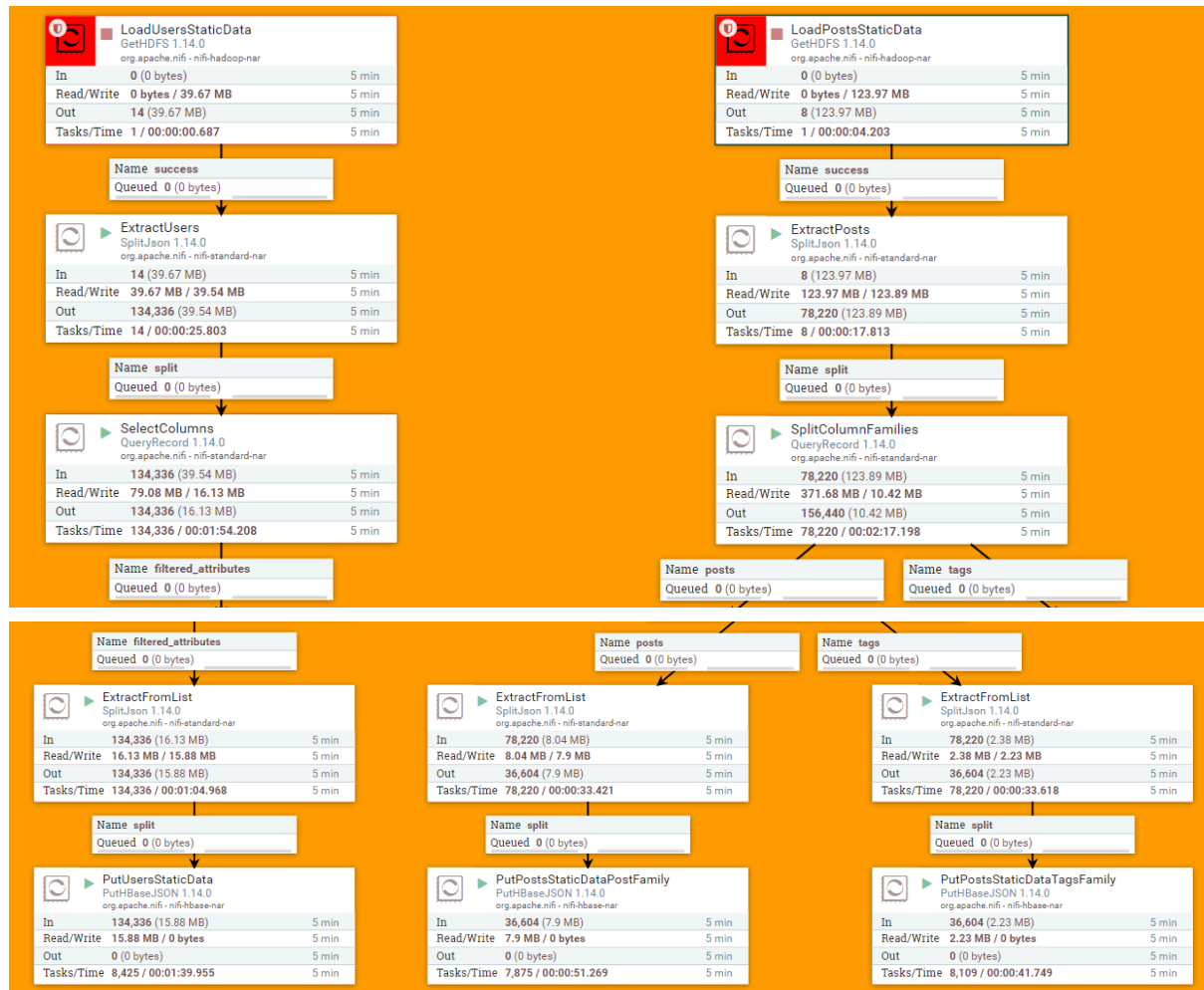
Celem testu jest sprawdzenie czy działa ładowanie danych statycznych z master database do HBase. W tym celu uruchomiony zostanie odpowiedni flow w NiFi. Przed jego wykonaniem tabele docelowe w HBase są puste, a master database znajduje się komplet danych statycznych (stan taki sam jak na koniec poprzedniego testu). Po wykonaniu flow oczekuje się, że dane w master database pozostaną nienaruszone, zaś tabele w HBase zostaną wypełnione rekordami.

Stan tabeli przed uruchomieniem flow w NiFi:

```

hbase(main):025:0> count 'datascience_posts'
0 row(s)
Took 0.0353 seconds
=> 0
hbase(main):026:0> count 'datascience_users'
0 row(s)
Took 0.0254 seconds
=> 0
hbase(main):027:0>
  
```


Potwierdzenie wykonania flow:



Sprawdzenie liczby wierszy tabel z HBase po wykonaniu flow:

Posts:

```
Took 1.8375 seconds
=> 36604
```

Users:

```
Took 4.4557 seconds
=> 134336
```

Stan master database po wykonaniu flow:

```
vagrant@node1:~/PROJECT$ hdfs dfs -count /user/vagrant/PROJECT/static_master
1 53 443007622 /user/vagrant/PROJECT/static_master
```

Tabele zostały zapełnione danymi, zaś master database nie został naruszony co świadczy o poprawności działania flow.

Test 8: Wczytywanie danych statycznych z warstwy widokowej do warstwy prezentacji

Celem testu jest potwierdzenie poprawności wczytywania danych statycznych z HBase do PySpark'owej ramki danych. W tym celu uruchomiony zostanie fragment kodu odpowiedzialny za wczytywanie tych danych w głównym notebooku z wynikami. Oczekuje się, że dane zostaną wczytane poprawnie oraz w pełni - tzn. liczba wierszy taka sama jak w

HBase oraz brak znaczących problemów z danymi po wczytaniu (takich jak niepoprawnie wczytana kolumna bez danych itp.)

Cały kod odpowiedzialny za wczytywanie danych zawarty jest w notebooku z analizą danych. Proces ten jest identyczny jak w przypadku danych online z tą różnicą, że dodatkowo do osobnej tabeli wczytywane są dane użytkowników. Dla przypomnienia, proces ten wygląda następująco dla tabeli z postami (dla tabeli z użytkownikami jest identyczny z wyłączeniem fragmentu o tagach):

1. Zczytanie wierszy z tabeli HBase o postach i zapisanie ich do listy. Jednocześnie do osobnej listy zapisywane są informacje o tagach dla każdego pytania;
2. Stworzenie ramki danych Pandas z listy wierszy z HBase stworzonej w poprzednim kroku. Osobno tworzona jest też ramka danych dla listy z informacjami o tagach;
3. Konwersja ramek danych stworzonych w poprzednim kroku na ramkę danych PySpark.

Schema ramki danych postów:

```
df_datascience_posts.printSchema()

root
 |-- answer_count: long (nullable = true)
 |-- comment_count: long (nullable = true)
 |-- creation_date: date (nullable = true)
 |-- last_activity_date: date (nullable = true)
 |-- last_edit_date: date (nullable = true)
 |-- closed_date: date (nullable = true)
 |-- owner_user_id: double (nullable = true)
 |-- score: long (nullable = true)
 |-- view_count: long (nullable = true)
 |-- question_id: string (nullable = true)
```

Podgląd pierwszych 5 wierszy ramki postów oraz liczba wierszy:

```
df_datascience_posts.show(n=5)
```

| answer_count | comment_count | creation_date | last_activity_date | last_edit_date | closed_date | owner_user_id | score | view_count | question_id |
|--------------|---------------|---------------|--------------------|----------------|-------------|---------------|-------|------------|-------------|
| 1 | 2 | 2016-01-27 | 2017-12-27 | 2016-01-27 | null | 8820.0 | 15 | 7402 | 10000 |
| 2 | 0 | 2021-08-12 | 2021-08-18 | null | null | 121019.0 | 1 | 215 | 100001 |
| 0 | 3 | 2021-08-12 | 2021-08-12 | null | null | 122641.0 | 0 | 114 | 100003 |
| 0 | 2 | 2021-08-12 | 2021-08-12 | null | null | 120849.0 | 0 | 58 | 100005 |
| 1 | 0 | 2021-08-12 | 2021-08-14 | null | null | 112224.0 | 0 | 117 | 100010 |

only showing top 5 rows

```
23/12/18 13:29:37 WARN scheduler.TaskSetManager: Stage 4 contains a task of very large size (1720 KiB). The maximum recommended task
```

```
df_datascience_posts.count()
```

```
23/12/18 13:29:37 WARN scheduler.TaskSetManager: Stage 5 contains a task of very large size (1720 KiB). The maximum recommended task
```

```
36604
```

Tabela z tagami:

```
df_datascience_tags.printSchema()
```

```
root
|-- question_id: string (nullable = true)
|-- tag: string (nullable = true)
```

```
df_datascience_tags.show(n=10)
```

```
+-----+-----+
|question_id|          tag|
+-----+-----+
|      10000| bayesian-networks|
|      10000|          pgm|
|     100001| machine-learning|
|     100001|          nlp|
|     100001|      transformer|
|     100001| machine-translation|
|     100003| class-imbalance|
|     100003| multilabel-classi...|
|     100005| machine-learning|
|     100005|      deep-learning|
```

Schema ramki danych użytkowników:

```
df_datascience_users.printSchema()
```

```
root
|-- location: long (nullable = true)
|-- creation_date: date (nullable = true)
|-- down_votes: long (nullable = true)
|-- up_votes: long (nullable = true)
|-- reputation: long (nullable = true)
|-- views: long (nullable = true)
|-- user_id: long (nullable = true)
```

Podgląd pierwszych 5 wierszy ramki użytkowników oraz liczba wierszy:

```
df_datascience_users.show(n=5)
```

```
+-----+-----+-----+-----+-----+-----+-----+
|location|creation_date|down_votes|up_votes|reputation|views|user_id|
+-----+-----+-----+-----+-----+-----+-----+
|      1|  2014-05-13|      2148|      841|         1|  495|    -1|
|      1|  2014-05-13|         0|         0|        101|  723|     1|
|      1|  2014-05-13|         0|         1|        101|    5|    10|
|      1|  2014-05-14|         0|        38|        101|    2|   100|
|      1|  2014-06-19|         0|         0|         1|    3|   1000|
+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

```
23/12/18 13:30:13 WARN scheduler.TaskSetManager: Stage 8 contains a task
```

```
df_datascience_users.count()
```

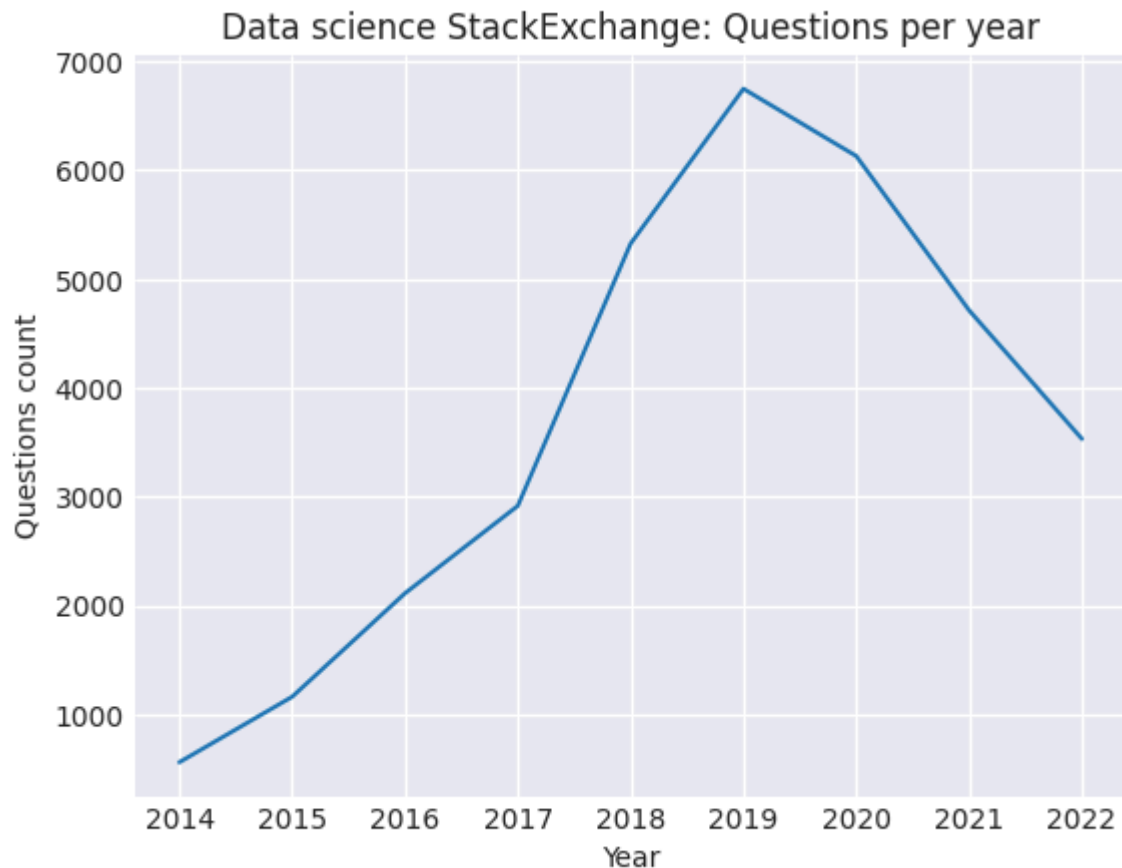
```
23/12/18 13:30:14 WARN scheduler.TaskSetManager: Stage 9 contains a task
134336
```

Liczby wierszy poszczególnych tabel pokrywają się z liczbami wierszy odpowiadających im tabel w HBase. Ponadto nie widać żadnych znaczących problemów z załadowanymi danymi, co oznacza, że zostały one pomyślnie wczytane.

Wykonane przez nas analizy

Data Science StackExchange

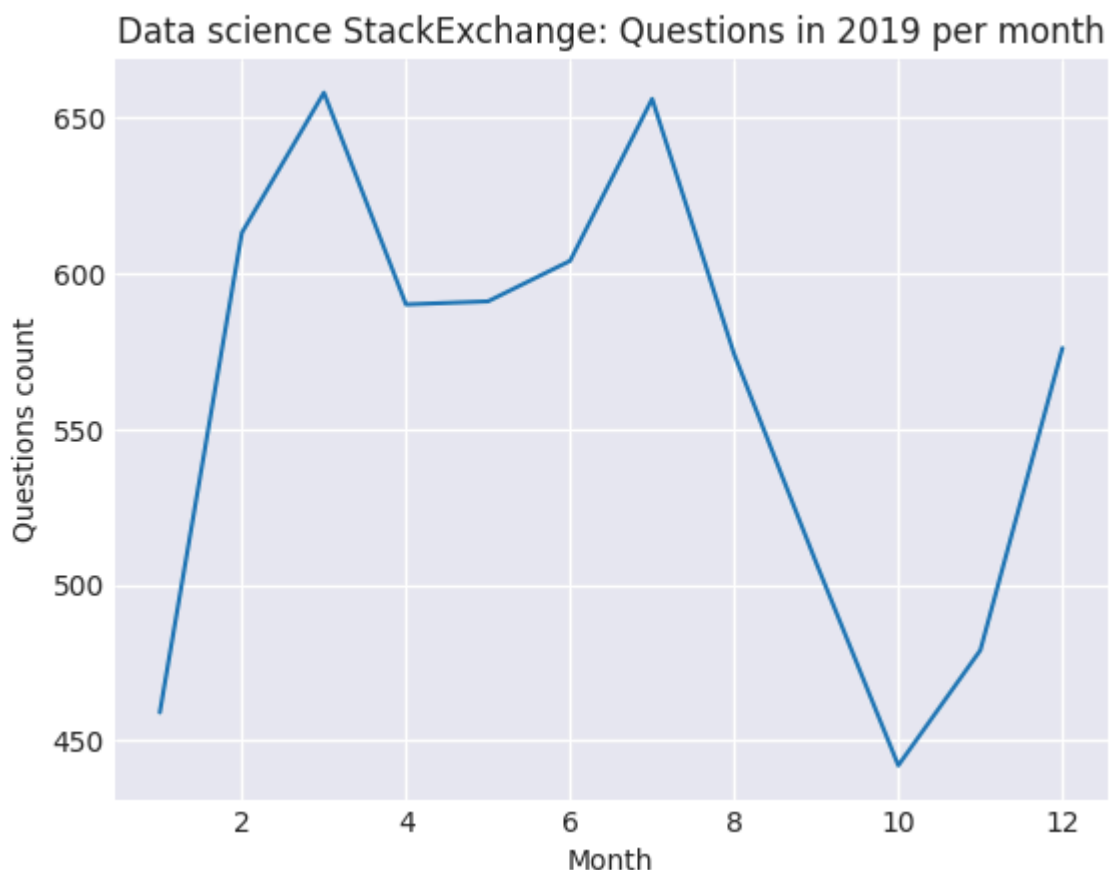
W pierwszej kolejności zdecydowaliśmy się zbadać poziom aktywności w serwisie. W tym celu zliczyliśmy liczbę pytań w każdym roku i umieściliśmy te dane na wykresie 1.



Wykres 1: Liczba pytań na Data Science StackExchange w podziale na rok

Można zauważyć, że popularność serwisu rosła aż do roku 2019 kiedy to liczba pytań osiągnęła szczytową wartość. Trend odwrócił się w kolejnych latach gdzie widać, że z roku na rok liczba pytań zadawanych w serwisie zaczęła spadać w podobnym tempie w jakim wcześniej rosła.

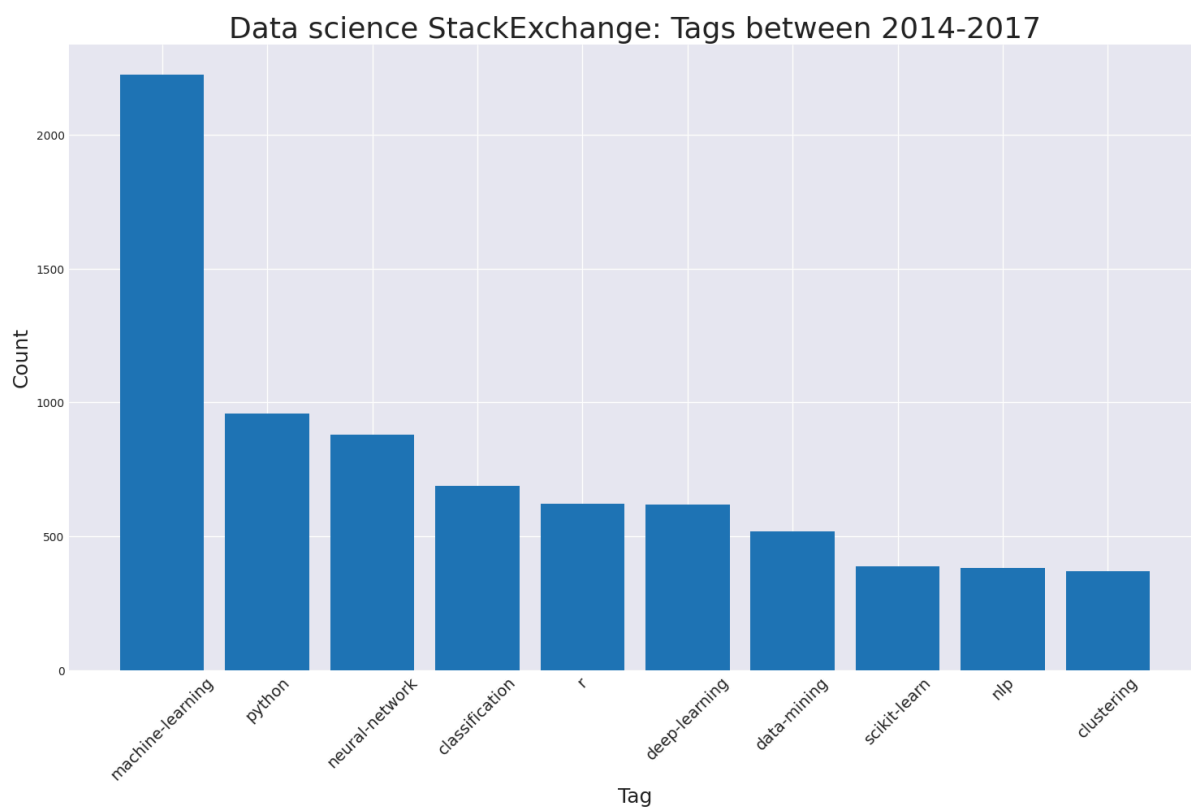
W kolejnym kroku chcieliśmy dokładniej przyjrzeć się tej samej statystyce w roku 2019, kiedy to serwis cieszył się największą popularnością. Wykres 2 przedstawia liczbę zadawanych pytań w każdym z miesięcy w roku 2019.



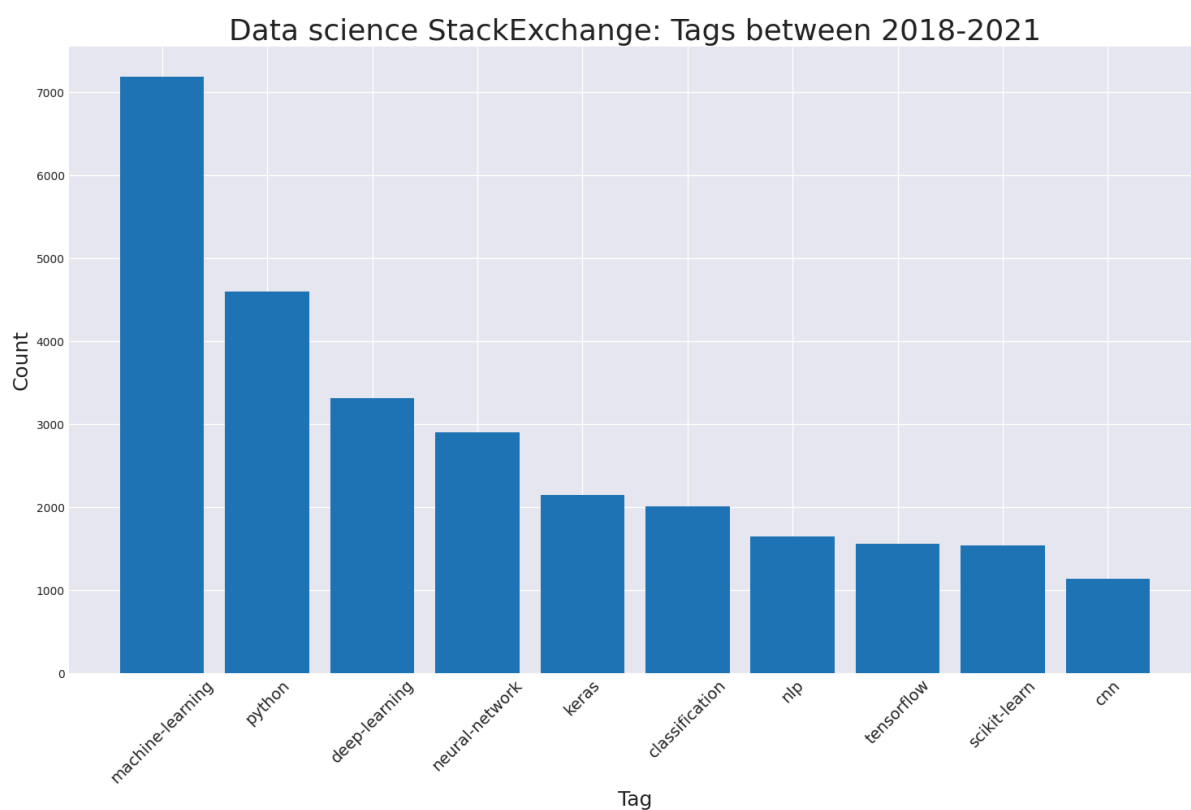
Wykres 2: Liczba pytań w roku 2019 na Data Science StackExchange w podziale na miesiąc

W oczy rzucają się dwa szczytowe miesiące (marzec oraz lipiec), oraz wyraźny spadek w październiku. Poświęciliśmy chwilę aby wykonać research wydarzeń ze świata data science które mogły potencjalnie wpłynąć na taki rozkład aktywności w tym roku. Przykładowo, w połowie lutego OpenAI wydało model językowy GPT-2, co mogło wpłynąć na wzmożone zainteresowanie tematami powiązanymi z uczeniem maszynowym.

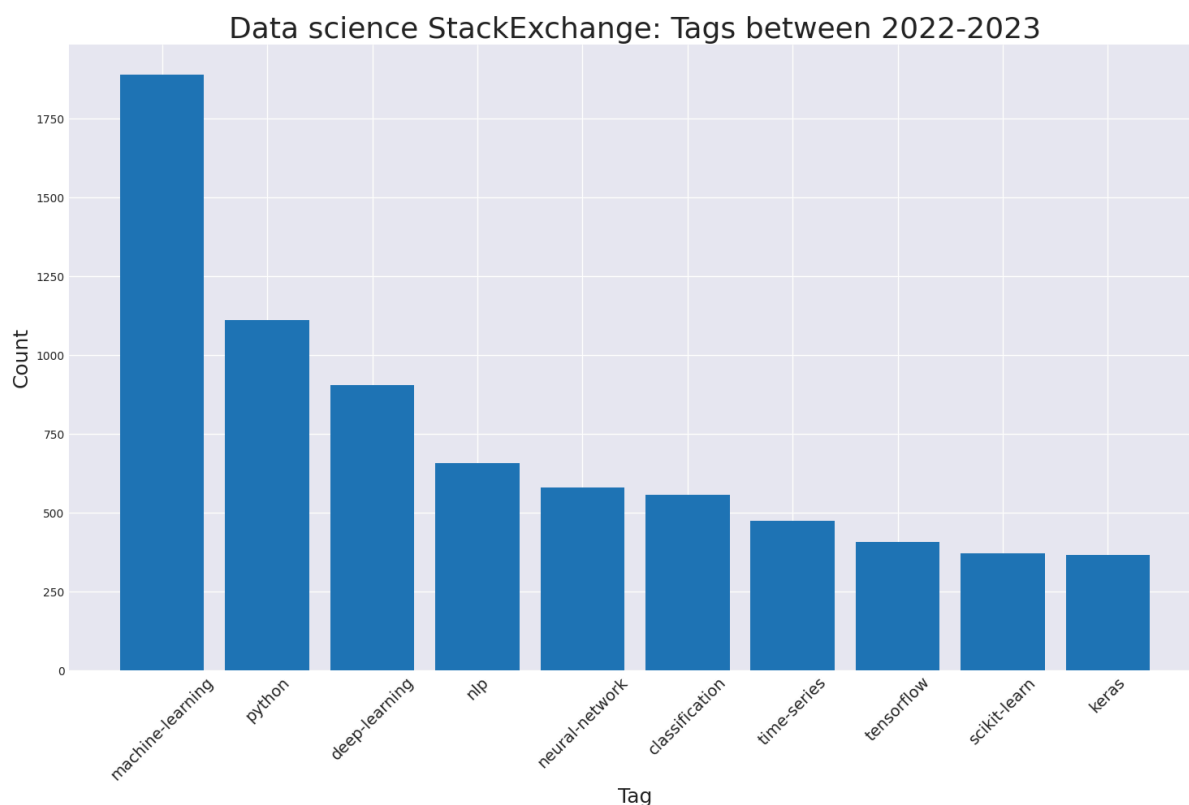
Kolejną rzeczą jaką chcieliśmy zbadać w kontekście okresu największej popularności serwisu są tematy zadawanych pytań zawarte w tagach. Chcieliśmy zbadać, czy popularność serwisu w tym okresie jest zawdzięczona jakimś obszarom tematycznym będącym wówczas na czasie. W tym celu porównaliśmy popularne tagi w latach 2018-2021 z tagami popularnymi w mniej aktywnych latach, w podziale na przedziały przed 2018 oraz po 2021. Poniższe wykresy przedstawiają zatem liczby wystąpień dziesięciu najpopularniejszych tagów kolejno: w latach 2014-2017 (wykres 3), w latach 2018-2021 (wykres 4) oraz w latach 2022-2023 (wykres 5).



Wykres 3: Dziesięć najpopularniejszych tagów na Data Science StackExchange w latach 2014-2017



Wykres 4: Dziesięć najpopularniejszych tagów na Data Science StackExchange w latach 2018-2021



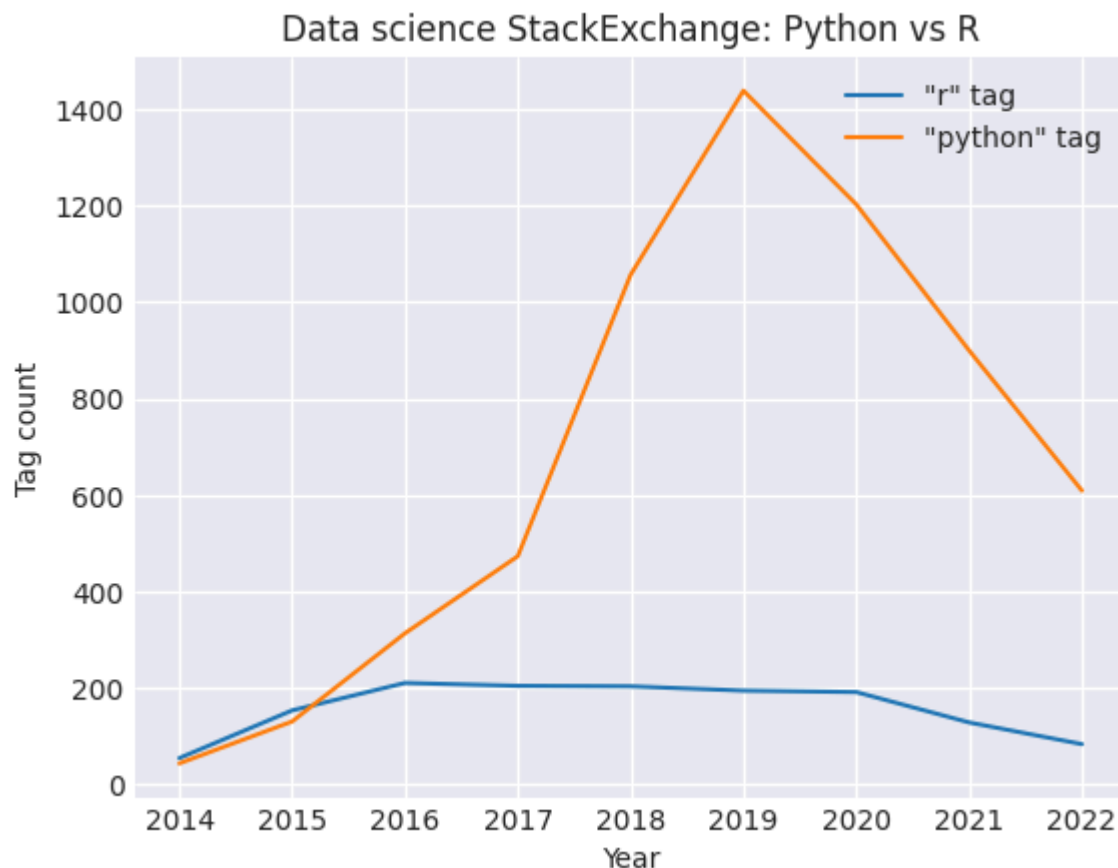
Wykres 5: Dziesięć najpopularniejszych tagów na Data Science StackExchange w latach 2022-2023

Pomiędzy pierwszym (2014-2017, wykres 3) a drugim (2018-2021, wykres 4) okresem widać wyraźną zmianę trendów spowodowaną wzrostem zainteresowania tematem sieci neuronowych. Większość tagów powiązana z tym tematem (jak “deep-learning” czy “neural-network”) zajmowała wysokie miejsca w rankingu popularności podczas gdy zainteresowanie klasycznym data miningiem spadło, przynajmniej w porównaniu do bardziej popularnych tematów - w okresie 2018-2021 tag “data-mining” zajął dopiero 22 miejsce, zaś “clustering” 16. Wyraźnie widać także wzrost zainteresowania frameworkiem Tensorflow o czym świadczy obecność tagów “tensorflow” oraz “keras” wśród dziesięciu najpopularniejszych tagów w najaktualniejszym okresie.

Porównując z kolei najpopularniejszy okres z okresem najnowszym (2022-2023, wykres 5) widać, że trendy nie zmieniły się aż tak bardzo jak w poprzednim przypadku. Jest jednak kilka różnic takich jak stosunkowo wysoka pozycja tagu “time-series”. Nie jest to wprawdzie żaden ogromny skok popularności, gdyż w poprzednich okresach był odpowiednio 14 i 11, warto jednak odnotować stopniowy wzrost zainteresowania tym tematem względem innych tematów. Odnotować można także wzrost popularności tagu “nlp” względem innych tagów. Ten ostatni może być spowodowany wydaniem 30 listopada 2022 roku przez OpenAI narzędzia ChatGPT, który szybko zyskał popularność w szerokiej grupie odbiorców.

Naszą uwagę przykuła jeszcze jedna zmiana wśród trendów, mianowicie język R, który w okresie do 2017 był szóstym najpopularniejszym tagiem na platformie, zaś po tym okresie nie łapał się nawet do najlepszej dziesiątki (miejsce 17). Chcieliśmy zatem zbadać popularność tego języka na platformie Data Science StackExchange, dodatkowo

porównując go z jego największą konkurencją - językiem Python. Wyniki tego porównania zamieszczone są na wykresie 6.

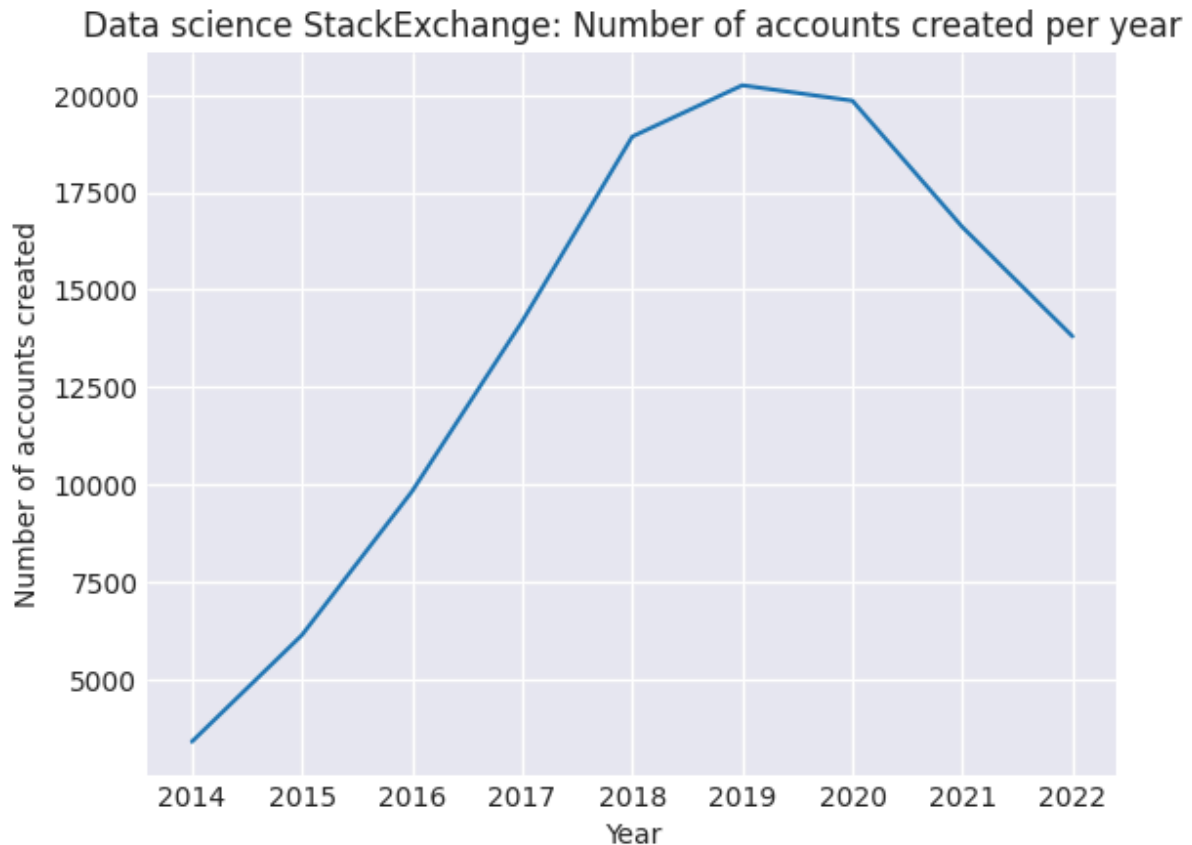


Wykres 6: Porównanie liczebności występowania tagów R i Python na Data Science StackExchange w kolejnych latach

W początkowej fazie istnienia serwisu język R był na nim nieco bardziej popularny niż język Python. Zależność ta jednak odwróciła się w roku 2016 gdy Python zaczął gwałtownie zyskiwać na popularności podczas gdy język R od roku 2017 doświadczał tendencji spadkowej. W oczy rzuca się przede wszystkim fakt że ten język nie skorzystał zbytnio na zwiększeniu się aktywności w serwisie w okolicach roku 2019. Język Python z kolei prawdopodobnie napędzał wzrost popularności na serwisie będąc jednym z najpopularniejszych tagów w okresie wzrostu, przez co krzywa jego popularności jest bardzo podobna do krzywej liczby pytań w serwisie przedstawionej na wykresie 1.

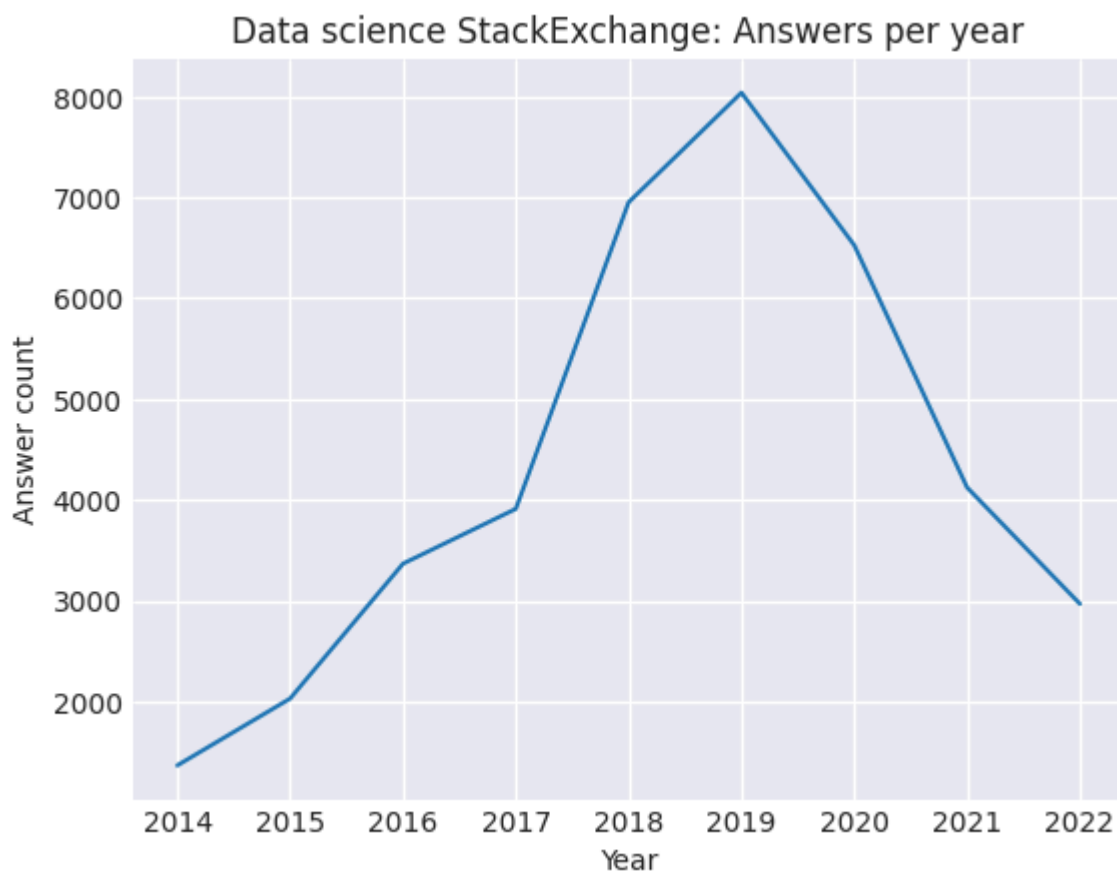
W kolejnej analizie chcieliśmy się przyjrzeć innym aspektom cechującym badane forum. Chcieliśmy zbadać czy pozostałe zmienne powiązane z popularnością serwisu zmieniają się podobnie jak liczba publikowanych postów. W tym celu przyjrzelśmy się:

- liczbie zakładanych kont w podziale na rok (wykres 7),
- łącznej liczbie tworzonych odpowiedzi pod pytaniami z kolejnych lat (wykres 8),
- łącznej liczbie wyświetleń pytań tworzonych w kolejnych latach (wykres 9).



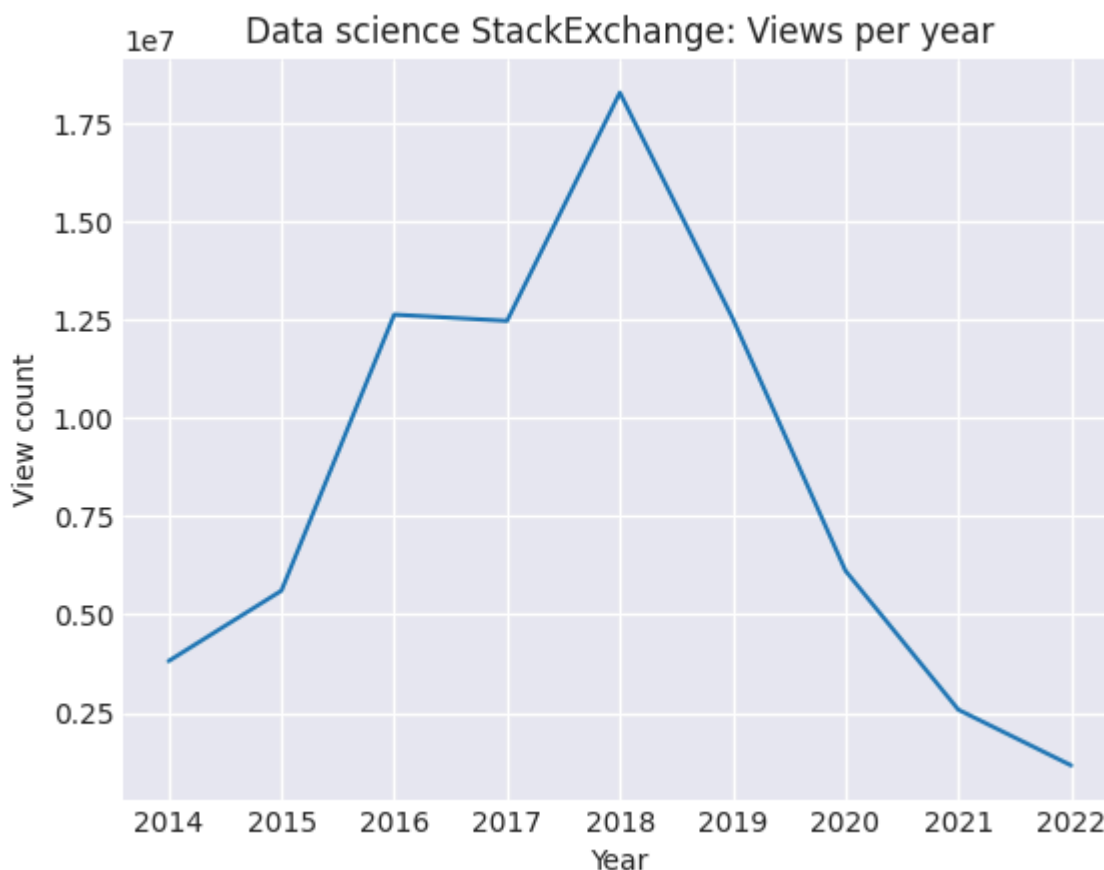
Wykres 7: Liczba zakładanych kont na Data Science StackExchange w podziale na rok

Krzywa na wykresie 7 jest podobna do tej na wykresie 1 co oznacza, że liczba tworzonych postów idzie najczęściej w parze z liczbą zakładanych kont. Co ciekawe, wartości na tym wykresie są mniej więcej trzy razy większe niż na wykresie 1 - oznacza to, że większość użytkowników tworzy konta w serwisie jednak nie decyduje się na zadanie żadnego pytania. Może to zaskakiwać gdyż do przeglądania serwisu nie jest potrzebne na nim konto, zatem naturalne mogłoby się wydawać, że użytkownicy zakładają w serwisie konta głównie po to by zaraz potem zadać jakieś pytanie - tak jednak nie jest.



Wykres 8: Łączna liczbie tworzonych odpowiedzi pod pytaniami z kolejnych lat na Data Science StackExchange

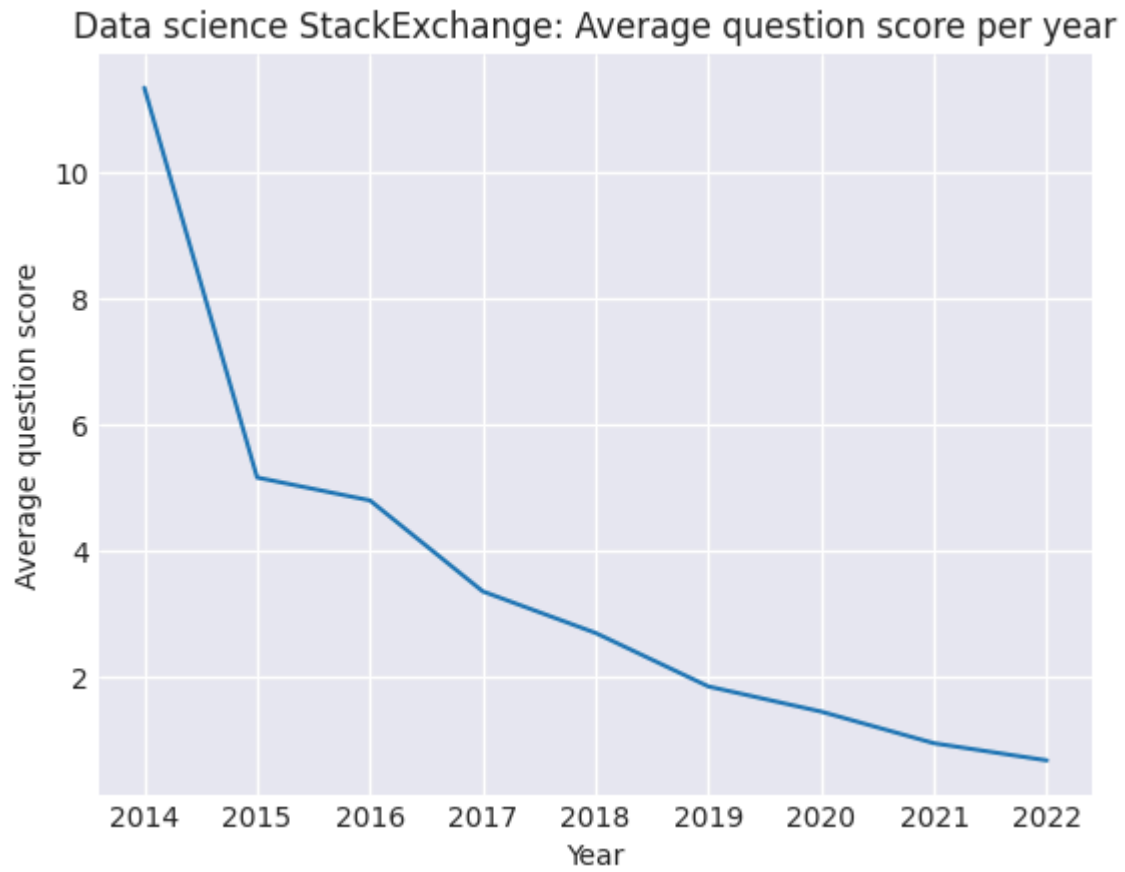
Krzywa na wykresie 8 jest jeszcze bardziej podobna do tej na wykresie 1. Wartości na nim widoczne są tylko niewiele większe od wartości na wykresie 1, co sugeruje, że średnio pod jednym pytaniem rzadko znajduje się więcej niż jedna odpowiedź.



Wykres 9: Łączna liczba wyświetleń pytań tworzonych w kolejnych latach na Data Science StackExchange

Z trzech ostatnich wykresów wykres 9 wydaje się najbardziej zaskakujący, gdyż w przeciwieństwie do pozostałych wykresów, tutaj zdecydowane maksimum osiągnięte zostało dla postów stworzonych w roku 2018. Oznacza to że mimo, że w roku 2018 zadane zostało mniej pytań, to jednak te pytania były bardziej popularne niż te zadane w 2019 roku.

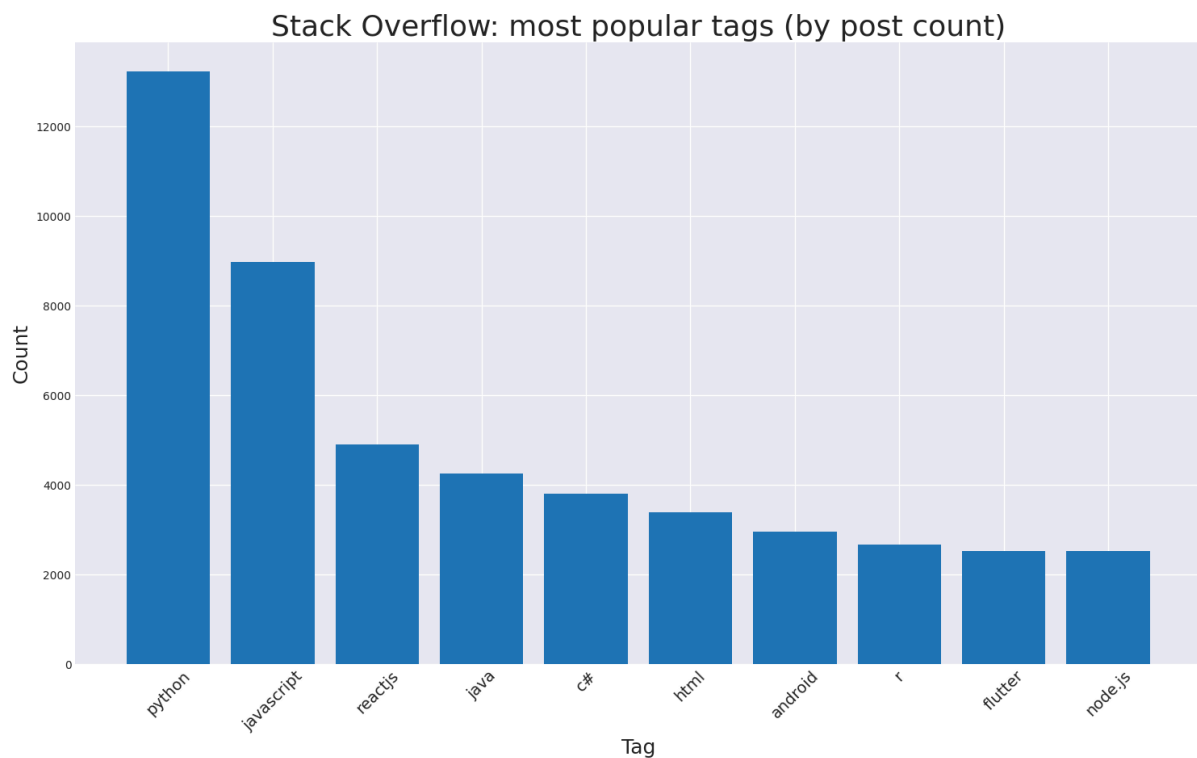
W ostatniej kolejności chcieliśmy się przyjrzeć średniej ocenie pytań tworzonych w kolejnych latach. Ciekawiło nas, czy tutaj również będą widoczne podobne tendencje co w przypadku pozostałych statystyk. Dane te zostały przedstawione na wykresie 10. Okazuje się, że niezależnie od liczby zadawanych pytań w danym roku czy liczbie wyświetleń pod tymi pytaniami średnia ocena pytań stale maleje. Możliwym wyjaśnieniem takiej sytuacji może być fakt, że im więcej pytań jest na platformie, tym większa szansa że kolejne zadane pytanie jest duplikatem wcześniej zadanego pytania, przez co zbiera dużo negatywnych ocen zaniżając tym samym średnią ocenę dla pytań tworzonych w danym roku.



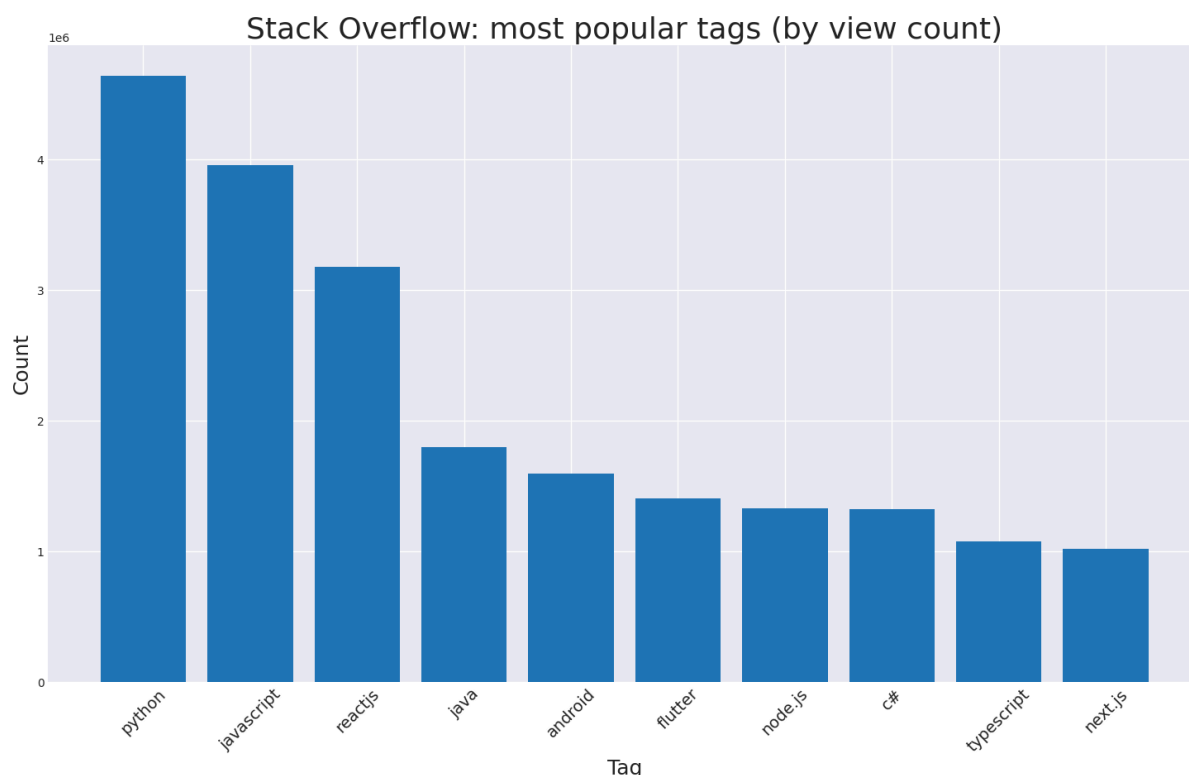
Wykres 10: Średnia ocena pytań tworzonych w kolejnych latach na Data Science StackExchange

StackOverflow

W przypadku serwisu StackOverflow, z uwagi na dużą popularność serwisu i ograniczenia API posiadamy jedynie dane dotyczące podzbioru pytań, stąd nie miałoby sensu analiza tych danych pod kątem liczby pytań. Chcieliśmy się zatem skupić jedynie na analizie trendów w tym serwisie. W pierwszej kolejności zdecydowaliśmy się więc spojrzeć na najpopularniejsze tagi zarówno pod kątem liczby ich wystąpień (wykres 11) jak i łącznej liczby wyświetleń pytań zawierających dane tagi (wykres 12).



Wykres 11: Dziesięć najpopularniejszych tagów na Stack Overflow od 2022 roku pod względem liczebności ich występowania



Wykres 12: Dziesięć najpopularniejszych tagów na Stack Overflow od 2022 roku pod względem łącznej liczby wyświetleń postów zawierających dane tagi

Z obu wykresów można wyciągnąć podobne wnioski. Python oraz JavaScript są zdecydowanie najbardziej popularnymi językami o które użytkownicy zadają pytania na platformie. Wśród JavaScript'owych frameworków zdecydowanie najpopularniejszy jest React, który jest trzecim najbardziej popularnym tagiem.

Wśród najbardziej popularnych języków znajdują się także Java oraz C#, które wykorzystywane są często do tworzenia backendów aplikacji. Popularne również jest tworzenie aplikacji mobilnych o czym świadczy wysoka popularność tagów Android oraz Flutter.

W dalszej kolejności chcieliśmy się przyjrzeć samym pytaniom. Postanowiliśmy sporządzić ranking dziesięciu najpopularniejszych pytań chcąc między innymi sprawdzić, czy zawarte w nich tagi będą pokrywać się z tymi widocznymi na wykresach 11 i 12. Tytuły tych pytań, wraz z przypisanymi do nich tagami zamieszczone są w tabeli poniżej:

Tabela 1: Dziesięć najpopularniejszych pytań spośród zebranych danych

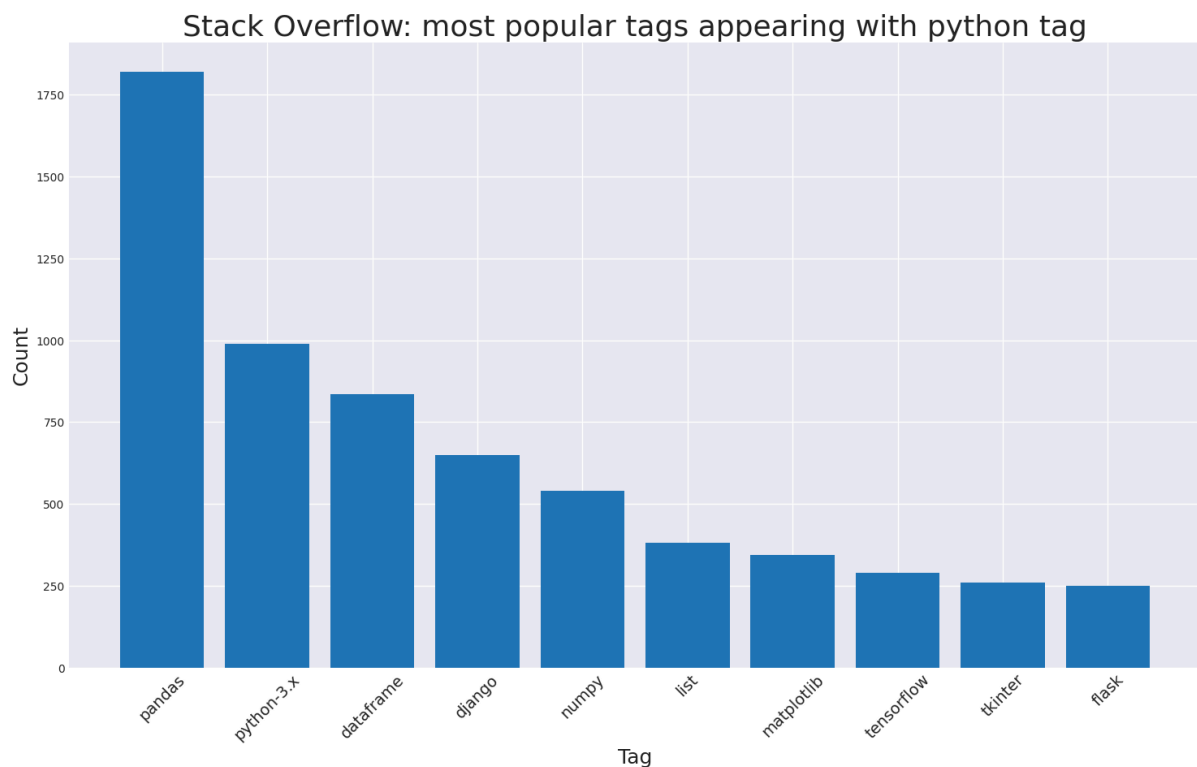
| Nr | Wyświetlenia | Tytuł | Tagi |
|----|--------------|---|--|
| 1 | 228969 | Ionic - how to prevent error "TS2339: Property XXX does not exist on type" | angular ionic-framework |
| 2 | 175222 | ViewPropTypes will be removed from React Native. Migrate to ViewPropTypes exported from 'deprecated-react-native-prop-types | react-native |
| 3 | 170102 | Fatal error "unsafe repository ('/home/repon' is owned by someone else)" | git cve-2022-24765 |
| 4 | 122929 | Redux createStore() is deprecated - Cannot get state from getState() in Redux action | javascript redux redux-toolkit |
| 5 | 87148 | How to use Font Awesome 6 icons? | font-awesome font-awesome-6 |
| 6 | 86760 | Uncaught (in promise) TypeError: Cannot read properties of undefined (reading 'data') | javascript reactjs validation |
| 7 | 82312 | Kubectrl error upon applying agones fleet: ensure CRDs are installed first | kubernetes minikube agones |
| 8 | 78524 | Error: There was an error while hydrating. Because the error happened outside of a Suspense boundary, the entire root will switch to client rendering | javascript reactjs next.js tailwind-css |
| 9 | 75992 | Sign_and_send_pubkey: no mutual signature supported | ssh openssh |
| 10 | 70751 | AttributeError: module 'PIL.Image' has no attribute 'ANTIALIAS' | python python-imaging-library |

Co ciekawe, mimo że tag “python” jest zdecydowanie najpopularniejszym tagiem zarówno pod względem sumarycznej liczby wyświetleń jak i liczby pytań, to w rankingu najpopularniejszych pytań post powiązany z tym językiem programowania jest dopiero dziesiąty. Może to świadczyć o tym, że w przypadku Python’a pojawia się duża liczba pytań, które niekoniecznie potem stają się popularne, podczas gdy społeczności innych technologii skupiają się wokół mniejszej liczby pytań.

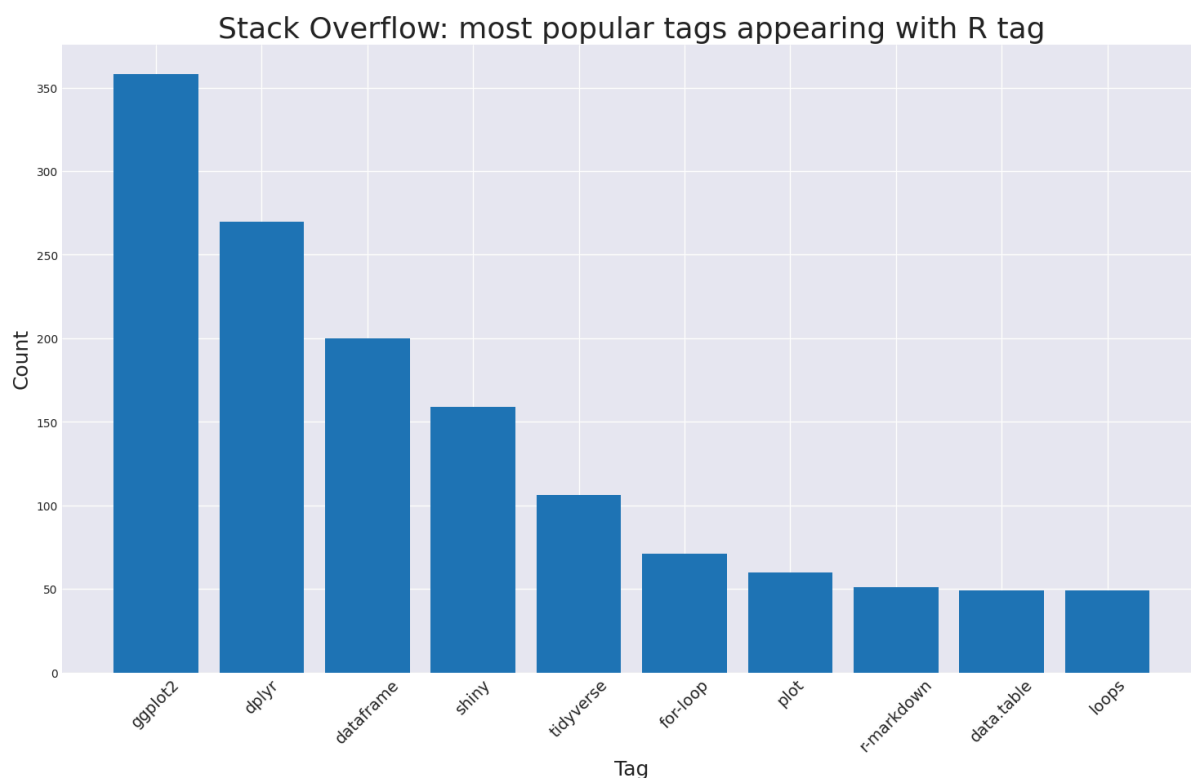
Warto też jednak zwrócić uwagę na pewne niedoszacowanie popularności technologii webowych na wykresach 11 i 12, które najprawdopodobniej wynika z tego, że posty powiązane z konkretnymi framework’ami webowymi niekoniecznie posiadają tag “javascript” (ani “typescript”), a co za tym idzie ich popularność jest rozbita na wiele różnych tagów. Widać to doskonale na przykładach dwóch najpopularniejszych spośród zebranych pytań. Pierwsze z nich odnosi się do Angulara, czyli jednego z najpopularniejszych framework’ów do tworzenia aplikacji webowych, do którego wykorzystuje się język TypeScript. Drugie z nich dotyczy React Native, które umożliwia tworzenie aplikacji mobilnych w językach

JavaScript oraz TypeScript. Żadne z tych pytań jednak nie posiada jednak tagu odpowiadającego konkretnemu językowi programowania.

Przechodząc dalej, chcieliśmy przyjrzeć się trendom w językach najczęściej pojawiających się na zajęciach na naszym kierunku studiów, czyli Python i R. Ciekawiło nas które tagi występują najczęściej razem z tagami odpowiadającymi tym językom. Wyniki tej analizy przedstawione są na wykresach 13 (Python) i 14 (R).



Wykres 13: Dziesięć najpopularniejszych tagów występujących razem z tagiem Python na Stack Overflow od 2022 roku pod względem liczebności ich występowania

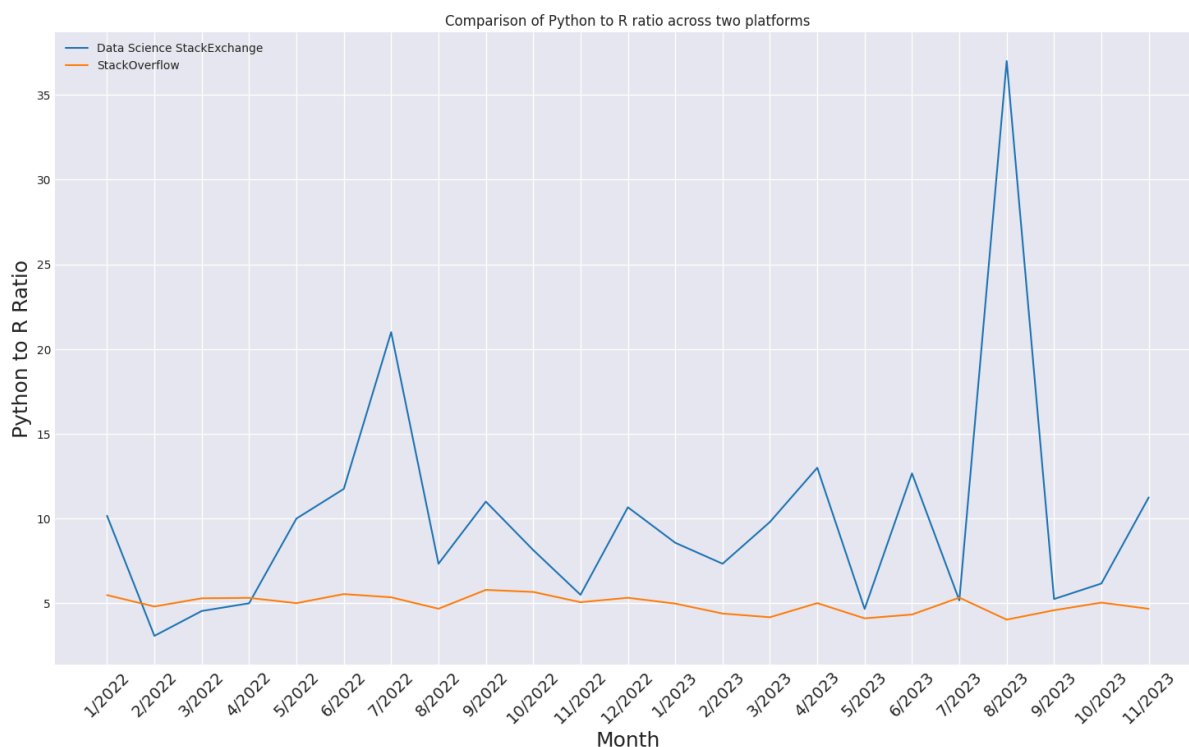


Wykres 14: Dziesięć najpopularniejszych tagów występujących razem z tagiem R na Stack Overflow od 2022 roku pod względem liczebności ich występowania

Zdecydowanie najpopularniejszym pakietem pythonowym jest Pandas (wykres 13), służący do wykonywania operacji na ramkach danych, o czym świadczy pierwsze miejsce tagu “pandas” i trzecie tagu “dataframe”. Popularny jest także pakiet NumPy, co nie dziwi ze względu na dużą uniwersalność wykorzystania tego pakietu - oferuje on m.in. zoptymalizowane implementacje tablic. Taki zestaw pierwszych czterech najpopularniejszych tagów sugeruje, że język Python w tym okresie był najczęściej wykorzystywany do analizy danych. Jednakże język Python ma również wiele innych popularnych zastosowań, m.in. tworzenie aplikacji webowych lub backendów, o czym świadczy obecność wśród dziesięciu najpopularniejszych tagów Django oraz Flaska.

W przypadku języka R (wykres 14) niemal wszystkie najpopularniejsze tagi powiązane są z pakietami poświęconymi różnym aspektom związanych z analizą danych, co potwierdza główne zastosowanie tego języka. Najpopularniejszym pakietem do tworzenia wizualizacji w R jest ggplot2, do operacji na ramkach danych dplyr, zaś do tworzenia dashboardów - shiny.

Na sam koniec chcieliśmy zobaczyć, czy proporcja liczby postów powiązanych z Pythonem a R'em różni się pomiędzy Stack Overflow a Data Science StackExchange. Zestawienie to zaprezentowane jest na wykresie 15.



Wykres 15: Porównanie proporcji liczby postów z tagiem Python do liczby postów z tagiem R w kolejnych miesiącach pomiędzy Data Science StackExchange a Stack Overflow

Ze względu na mniejszą popularność serwisu Data Science StackExchange liczba zadawanych tam pytań jest mniejsza niż na Stack Overflow stąd większe wahania krzywej odpowiadającej temu serwisowi. Widoczne jest jednak że mimo to ta krzywa znajduje się prawie w każdym miesiącu powyżej krzywej odpowiadającej Stack Overflow.

Podział pracy w zespole

- Skonfigurowanie flow w Apache NiFi - Maciej, Szymon
- Stworzenie niezbędnych tabel w Apache Hbase - Szymon, Maciej
- Skrypty bashowe oraz pythonowe - Szymon, Maciej
- Analiza danych, tworzenie wizualizacji, wyciąganie wniosków - Maciej, Szymon
- Prezentacja końcowa - Szymon, Maciej