POLITECHNIKA GDAŃSKA

WYDZIAŁ FIZYKI TECHNICZNEJ I MATEMATYKI STOSOWANEJ

Imię i nazwisko studenta: Paula Dutkowska
Nr albumu: 144292
Studia drugiego stopnia
Forma studiów: stacjonarne
Kierunek studiów: Matematyka
Specjalność: Geometria i grafika komputerowa

PRACA DYPLOMOWA MAGISTERSKA

Tytuł pracy w języku polskim: Analiza wybranego zbioru danych przy użyciu SAS Enterprise Miner i ekosystemu Hadoop.

Tytuł pracy w języku angielskim: Analysis of the selected data set using SAS Enterprise Miner and ecosystem of Hadoop.

Potwierdzenie przyjęcia pracy	
Opiekun pracy podpis	Kierownik Katedry/Zakładu (pozostawić właściwe) podpis
dr inż. Patryk Jasik	

Data oddania pracy do dziekanatu:

POLITECHNIKA GDAŃSKA
WYDZIAŁ FIZYKI TECHNICZNEJ I MATEMATYKI STOSOWANEJ

OŚWIADCZENIE

Imię i nazwisko: Paula Dutkowska
Data i miejsce urodzenia: 27.02.1993, Gdańsk
Nr albumu: 144292
Wydział: Wydział Fizyki Technicznej i Matematyki Stosowanej
Kierunek: matematyka
Poziom studiów: pierwszy
Forma studiów: stacjonarne

Ja, niżej podpisany(a), wyrażam zgodę/nie wyrażam zgody* na korzystanie z mojej pracy dyplomowej zatytułowanej: Analiza wybranego zbioru danych przy użyciu SAS Enterprise Miner i ekosystemu Hadoop.
do celów naukowych lub dydaktycznych.[1]

Gdańsk, dnia ....
podpis studenta

Świadomy(a) odpowiedzialności karnej z tytułu naruszenia przepisów ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz. U. z 2016 r., poz. 666 z późn. zm.) i konsekwencji dyscyplinarnych określonych w ustawie Prawo o szkolnictwie wyższym (Dz. U. z 2012 r., poz. 572 z późn. zm.),[2] a także odpowiedzialności cywilno-prawnej oświadczam, że przedkładana praca dyplomowa została opracowana przeze mnie samodzielnie.

Niniejsza(y) praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadaniem tytułu zawodowego.

Wszystkie informacje umieszczone w ww. pracy dyplomowej, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami zgodnie z art. 34 ustawy o prawie autorskim i prawach pokrewnych.

Potwierdzam zgodność niniejszej wersji pracy dyplomowej z załączoną wersją elektroniczną.

Gdańsk, dnia ....
podpis studenta

Upoważniam Politechnikę Gdańską do umieszczenia ww. pracy dyplomowej w wersji elektronicznej w otwartym, cyfrowym repozytorium instytucjonalnym Politechniki Gdańskiej oraz poddawania jej procesom weryfikacji i ochrony przed przywłaszczaniem jej autorstwa.

Gdańsk, dnia ....
podpis studenta

*) niepotrzebne skreślić

---

[1] Zarządzenie Rektora Politechniki Gdańskiej nr 34/2009 z 9 listopada 2009 r., załącznik nr 8 do instrukcji archiwalnej PG.

[2] Ustawa z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym:
Art. 214 ustęp 4. W razie podejrzenia popełnienia przez studenta czynu podlegającego na przypisaniu sobie autorstwa istotnego fragmentu lub innych elementów cudzego utworu rektor niezwłocznie poleca przeprowadzenie postępowania wyjaśniającego.
Art. 214 ustęp 6. Jeżeli w wyniku postępowania wyjaśniającego zebrany materiał potwierdza popełnienie czynu, o którym mowa w ust. 4, rektor wstrzymuje postępowanie o nadanie tytułu zawodowego do czasu wydania orzeczenia przez komisję dyscyplinarną oraz składa zawiadomienie o popełnieniu przestępstwa.

Streszczenie

Niniejsza praca magisterska stanowi krótkie omówienie metod przetwarzania danych w big data. W szczególności zwrócimy uwagę na Apache Hadoop, czyli projekt działający na licencji wolnego oprogramowania. Jednocześnie przyjrzymy się rozwiązaniom firmy SAS. Oznacza to, że połączymy moc SAS z rozproszonymi technologiami obliczeniowymi, aby przeobrazić potencjał big data w dużą wiedzę.

Tekst został utworzony przy pomocy książek oraz innych źródeł podanych w Bibliografii. Pojawiające się kody źródłowe powstały jako samodzielna praca. Właściwe wyniki uzyskane dzięki nim potwierdzają prawdziwość oraz użyteczność praktyczną omawianych zagadnień.

Słowa kluczowe: Big Data, Apache Hadoop, SAS, zbiory danych, python
Dziedzina nauki i techniki, zgodnie z wymogami OECD: 1.1 Matematyka (matematyka stosowana oraz statystyka); 1.2 Nauki o komputerach i informatyka.

Abstract

This Master's thesis is a brief overview of the big data methods for data processing. In particular, we will draw attention to the open-source project named Hadoop as well as SAS solutions. It means that we will combine the analytics power of SAS with distributed computing technologies from Apache Hadoop to transform big data into knowledge.

Text was created by means of source books and websites listed in the Bibliography. Source codes emerged, in turn, as a completely independent work. Obtaining the correct results confirms the reality and the genuineness and the practical utility of discussed issues.

Keywords: Big Data, Apache Hadoop, SAS, datasets, python

Branch of science: 1.1 Mathematics (Applied mathematics and statistics); 1.2 Computer science.

Spis treści

Wykaz ważniejszych oznaczeń i skrótów

Praca składa się z sześciu rozdziałów i odpowiedniej liczby podrozdziałów - ich numeracja została jasno podana w spisie treści. Występujące w tekście odsyłacze [x] nawiązują do odpowiedniej pozycji z bibliografii. Za symbolem a.b kryje się zaś informacja o konkretnym rysunku, zamieszczonym w rozdziale a. Nazwy plików, ścieżki dostępu oraz krótkie kody zostały zapisane czcionką pochyłą. Do pozostałych kodów programistycznych zastosowano maszynowy krój pisma.

W tekście zostały użyte następujące skróty:

ang. - po angielsku, skrót dodawany w sytuacji potrzeby podania równoważnego terminu, występującego w języku angielskim

CRISPDM - Cross-Industry Standard Process for Data Mining

DBMS - System zarządzania bazą danych, ang. Database Management System

GFS - Google File System

HDP - The Hortonworks Data Platform

JDBC - łącze do baz danych w Java, ang. Java DataBase Connectivity

OSS - otwarte oprogramowanie, ang. open-source software.

SAS EG - SAS Enterprise Guide

SDLfH - SAS Data Loader for Hadoop

ROZDZIAŁ 1

Wprowadzenie

Już od najmłodszych lat jesteśmy uczeni, aby dbać o środowisko. Szczególną uwagę przywiązujemy zwłaszcza do segregacji śmieci. Szkło, plastiki, baterie, odpady organiczne... Przynajmniej parę razy dziennie spotkamy pojemnik z podobnymi oznaczeniami. Oczywiście jest to wynikiem walki o jak najczystszą Ziemię. Zdajemy sobie sprawę z realnego „problemu śmietnikowego" i mamy pomysły jak go zmniejszać.

Jak to wygląda z „wirtualnym wysypiskiem"?

Zacznijmy od początku. Wraz z postępem ewolucji człowiek wytwarzał coraz więcej zasobów danych. Prymitywne środki zapisu, związane z malowidłami naskalnymi czy papirusami, z czasem doprowadziły do tak zwanego boomu informacyjnego, który obserwujemy od jakiegoś czasu. Był on już prognozowany w czasach II wojny światowej, o czym pisał m.in. Fremont Rider z Wesleyan University Librarian (źródło: [**8**]). W jego publikacji z 1944 roku pojawia się oszacowanie, mówiące o podwajaniu się zbiorów bibliotek amerykańskich uniwersytetów średnio co 16 lat. Według jego wyliczeń sama biblioteka w Yale powinna do 2040 roku składać się z około 200 milionów tomów. Półka, która mogłaby zmieścić taką kolekcję, musiałaby mierzyć około 6 tysięcy mil.

Trzeba przy tym pamiętać, że przypuszczenia F. Ridera nie brały pod uwagę pojawienia się urządzeń komputerowych, a tym bardziej sieci internetowej. Oznacza to, że powyższa predykcja daje jedynie niewielką cząstkę problemu, który obserwujemy dzisiaj.

Skupmy się na chwilę na danych, określających część ruchu, generowanego w globalnej sieci. Weźmy jako przykład portal *facebook.com*, który według rankingu ze strony *alexa.com* (referencja: [**1**]) stanowi trzecią najczęściej odwiedzaną witrynę na świecie (zaraz po *Google.com* oraz *youtube.com*). Oficjalnie przyciąga on dziennie średnio 1,28 miliarda użytkowników (stan na marzec 2017). Aż 1,15 miliarda osób korzystało w grudniu 2016 roku z mobilnego dostępu, a liczba ta teoretycznie wzrasta o 23 % w ciągu roku (źródło: [**4**]). Tak duży ruch na portalu daje wielki potencjał dotarcia do fanów i klientów osobom publicznym, firmom czy politykom. Przykładowo, Robert Lewandowski ma ponad 9 239 tysięcy fanów, a jedna z sieci telefonii komórkowych - Play - prawie 2 289 tysięcy polubień (referencje: [**2**] i [**3**], stan na czerwiec 2017).

Powyższe liczby pokazują skalę, a jednocześnie potencjał boomu informacyjnego. Wraz z pierwszymi symptomami jego pojawienia się, człon „big" z tytułowego „big data" (polskie: „duże dane") odnosił się do pewnych kategorii fizycznych - wagi książek, miejsca jakie zajmują. Dzisiaj zaś spotykamy się z pewnym złudnym zjawiskiem. Wytwarzane przez nas dane już nie potrzebują dla siebie tyle przestrzeni co niegdyś, po prostu „zawieszamy je wirtualnie". Łatwość i możliwa szybkość ich generowania doprowadza do zalania niepotrzebnymi „śmieciami danowymi", wśród których - jak na prawdziwym wysypisku - skrywają się perełki. Trzeba tylko nauczyć się skutecznie je wydobywać i właściwie wykorzystywać.

Celem pracy jest analiza dużego zbioru danych, odnoszącego się do kursów nowojorskich taksówek z 2016 roku. Tym samym przedstawimy, w jaki sposób okiełznać żywioł danych i jak wycisnąć ich potencjał przy wykorzystaniu otwartej platformy programistycznej Apache Hadoop. Jednocześnie zaprezentujemy moc drzemiącą w środowisku SAS, w szczególności w SAS Miner oraz SAS Enterprise Guide.

Dodatkowym celem jest omówienie zagadnień związanych z dziedziną big data. Teoretyczną część tekstu dopełni opis jednego ze sposobów połączenia ekosystemu Hadoop z oprogramowaniem z rodziny SAS.

ROZDZIAŁ 2

Wejście w świat Big Data

INFORMACJA 1. *W następującej części pracy zajmiemy się terminem Big Data. Oznacza to, że zdefiniujemy co za nim się kryje, a jednocześnie pokrótce stawimy czoła stereotypom krążącym wokół niego. Ponadto wspomnimy parę podstawowych reguł pracy przy big data.*

W drugiej połowie rozdziału przejdziemy do zbiorów danych. Wytłumaczymy problemy z nimi, jak z nimi pracować oraz opiszemy wybrany na potrzeby pracy zbiór.

Wokół pojęcia Big Data krąży dość wiele, często kompletnie nieprawdziwych mitów. Wiążą się one głównie ze strachem przed permanentną inwigilacją. Szukając materiałów do tej pracy, natknęłam się na parę porównań Big Data do „Big Brother". Jest to pewnego rodzaju odwołanie do rzekomego końca prywatności szarego użytkownika internetu. Według przeciwników, za big data kryją się wielkie firmy, które (często nielegalnie) miałyby pozyskiwać nasze dane dla niecnych celów. Oczywiście wszystko to dla zysków finansowych.

Jaka jest prawda? Jak zawsze, w każdej legendzie tkwi źdźbło prawdy. W tym przypadku związek pomiędzy faktami, a mitami tkwi w zbiorach informacji oraz w pieniądzach, jakie można uzyskać przy ich pomocy. Za głównym tematem tej pracy kryje się bowiem ogromny potencjał, który warto poznać.

Pierwszy raz termin Big Data pojawił się w 1999 roku. Korzystając ze słownika języka angielskiego, dwa słowa wchodzące w jego skład przetłumaczymy na język polski jako „duże dane". To pokrywa się z tym, że omawiane pojęcie odnosi się do ogromnych, wciąż zmieniających się zbiorów informacji, często liczonych w jednostkach większych niż tera- czy petabajty. Ze względu na wielkość, a niejednokrotnie i chaotyczność takich zbieranin danych, ich przetwarzanie i analiza jest utrudniona. Nieraz wręcz niemożliwa do wykonania przy wykorzystaniu powszechnych sposobów. W połączeniu z dużą ich wartościowością, prowadzi to do poszukiwania nowych, niestandardowych metod. W wielu publikacjach pojawia się dodatkowy termin, częściowo równoważny z ideą Big Data - High Performance Analytics.

Mówiąc prosto, Big Data to wielkie zbiory wciąż ruchomych danych i niestandardowe technologie ich przetwarzania. Powinny być one realizowane przy pomocy niedrogich systemów. Tym samym dąży się do wykorzystywania lokalnych dysków

twardych i niewyszukanych systemów, a jednocześnie wolnych, darmowych oprogramowań. Przykładem takiego oprogramowania jest omawiany w kolejnym rozdziale Apache Hadoop.

W zgodzie z zasadami big data często wykorzystuje się chmury. Z punktu widzenia geografa, stanowią one skupiska drobnych kropelek wody, zawieszone w atmosferze. Dla użytkownika internetu będzie to także pewne skupisko, ale plików, "zawieszone" w świecie wirtualnym. Przetwarzanie w chmurze pozwala nam wyjść poza obręb naszego komputera i pracować grupowo lub indywidualnie na zewnętrznych serwerowniach. Dzięki temu unikamy trudów utrzymania własnej infrastruktury, zachowujemy moc obliczeniową i zyskujemy możliwość tworzenia kopii zapasowych ważnych plików. Ponadto wiele chmur oferuje rozbudowane funkcje.

Taki typ dostępu do plików jest szczególnie pomocny przy klastrach komputerowych. Sprowadzają się one do połączonych ze sobą jednostek komputerowych (ang. nodów), współpracujących ze sobą jako zintegrowane środowisko pracy. To zaś często związane jest z obliczeniami rozproszonymi (ang. distributed computing). Są to obliczenia, w których pewne zadania zostają poddane dekompozycji - podzielone na procesy, podprogramy, podczęści. Dzięki temu skomplikowany problem nie musi być rozwiązywany przez pojedynczą osobę/komputer, a na przykład przez wspomniany klaster komputerowy. Podobne zasady występują w internetowych projektach, w których zwykli obywatele udostępniają moc obliczeniową swoich komputerów różnym instytucjom (na przykład NASA).

Na potrzeby pracy korzystaliśmy z klastra, będącego własnością wydziału Fizyki Technicznej i Matematyki Stosowanej Politechniki Gdańskiej.

Praca przy Big Data powinna być związana z metodologią Agile. Jest to sposób pracy nad wytwarzaniem oprogramowania, który opiera się o tzw. manifest Agile. Zgodnie z nim, bardziej cenimy

- ludzi i interakcje niż procesy i narzędzia;
- działające oprogramowanie od obszernej dokumentacji;
- współpracę z klientem niż formalne ustalenia;
- reagowanie na zmiany niż podążanie za planem.

Agile zaczyna się od zespołów, w skład których wchodzą takie osoby jak klient, inżynier, analityk big data i tak dalej. Im mniejsze grupy, a tym samym im więcej omnibusów, tym lepiej. Istnieje przy tym silna potrzeba dzielenia się wynikami w ramach zasady, że nie ma danych niegotowych. Wyniki pośrednie mogą być udostępniane jako kolaż danych, m.in. przy pomocy omówionych wyżej chmur.

Metodologia Agile wymusza zwinne wytwarzanie oprogramowania, przekreślając model kaskadowy. Projekt może ewoluować w drodze jego tworzenia, podczas

którego nie musimy trzymać się odrębnych, występujących w porządku jeden po drugim, faz pracy.

To zaś prowadzi do programowania imperatywnego, w którym występuje opis czynności potrzebnych do manipulowania danymi przetwarzanymi potokowo - krok po kroku. Jest to swego rodzaju przeciwstawienie się programowaniu relacyjnemu. W języku SQL określamy bowiem co chcemy zrobić, nie jak.

1. Zbiory danych

Mając do czynienia z tak dużymi zbiorami danych, warto mieć wypracowany jakiś sposób pracy z nimi. W 1996 roku pojawiła się metodologia Cross-Industry Standard Process for Data Mining, w skrócie CRISPDM. Dostarcza ona schemat cyklu życia danego projektu eksploracji danych. Dzielony jest on na sześć etapów, których kolejność możemy dopasować. Przy tym następna faza często mocno zależy od poprzedniej. Istnieje też możliwość, że pracując przy jednym etapie, będziemy musieli powrócić do wcześniejszego - w celu dopracowania go, czy wyeliminowania problemów. Metodologię tę możemy przedstawić przy pomocy następujących punktów:

- zrozumienie uwarunkowań - stwierdzenie celów, opracowanie definicji problemu, początkowy plan działań;
- zrozumienie danych - zebranie danych, zaznajomienie się z nimi przy pomocy wstępnej analizy, znalezienie prostych wzorców i zależności;
- przygotowanie danych - pracochłonne wyczyszczenie i przeobrażenie wstępnego, surowego zbioru danych w ostateczną formę, wybranie przypadków i zmiennych odpowiednich do analizy, przekształcenia na zmiennych;
- modelowanie - wybranie technik modelujących i zastosowanie ich, optymalizacja wyników;
- ewaluacja - ocenienie efektów etapu modelowania (sprawdzenie jakości, efektywności, założeń, czy wszystkie cele osiągnięte);
- wdrożenie.

W publikacji [**6**] pojawia się podobna metodologia, nawiązująca do hierarchii potrzeb Maslowa. Tak zwana piramida wartości danych prezentuje, że punkty-potrzeby z wyższej warstwy są uzależnione od zaspokojenia punktów-potrzeb z niższej. Tym samym nie można ominąć żadnego z poziomów. Prezentują się one następująco, zaczynając od dołu piramidy:

- rekordy - zbieranie i wyświetlanie pojedynczych rekordów;
- wykresy - oczyszczenie, wyodrębnienie własności oraz wizualizacja;
- raporty - dalsze szukanie powiązań i trendów, opisywanie, eksploracja;
- predykcje - wnioskowanie, przewidywanie;

- działania - wytwarzanie większych wartości.

Niestety, jak już zostało wspomniane, Big Data to często jeden wielki śmietnik informacyjny. Większość danych to dane nieoczyszczone i nieustrukturyzowane (lub na wpół ustrukturyzowane). Oznacza to, że nie są one wtłoczone w ściśle zdefiniowane schematy i przed analizą zostajemy zmuszeni „uprzątnąć" ten bałagan.

Przykład 1. *Surowa wiadomość e-mail, do odczytania na koncie Gmail przy pomocy opcji „Pokaż oryginał", to przykład danych na wpół ustrukturyzowanych. Taka forma nie spełnia modeli i tabel typowych dla relacyjnych baz danych. Jednocześnie pojawiają się etykiety lub inne znaczniki, narzucające kolejność rekordów.*

Często zbiory informacji obarczone są błędami lub niepotrzebnymi informacjami, które w końcowym etapie mogą dać nam niewłaściwe wyniki - zgodnie z zasadą GIGO. Odnosi się ona do angielskiego wyrażenia garbage in - garbage out. Oznacza to tyle, że jak na początku wprowadzamy śmieci (błędy), to i na końcu je uzyskamy. W wielu przypadkach jest to spowodowane tym, że dane zbierane są z różnych źródeł, często stosujących odmienne jednostki lub sposoby zapisu. Sprowadza się to do potrzeby sprawdzenia zbioru pod kątem:

- pola zbędne lub przestarzałe;
- punkty oddalone;
- brakujące wartości;
- różne lub niewłaściwe formaty;
- wartości absurdalne, mogące być wynikiem błędu.

Poza powyższym, trzeba także zwrócić uwagę na atrybuty skorelowane. Oznacza to tyle, że powinniśmy wyeliminować te kolumny ze zbioru danych, które są pomiędzy sobą powiązane. Jeżeli tego nie zrobimy, musimy liczyć się z potencjalnym wyolbrzymieniem jakiejś części danych, bądź doprowadzeniem do niestabilności modelu i niepewnych wyników.

1.1. Wybór zbioru danych. Aby przejść do głównego celu pracy, musimy wpierw uzyskać dostęp do znacznego zbioru informacji. W Internecie istnieje wiele źródeł danych, opatrzonych angielskim określeniem „datasets". Omawiany Hadoop poniekąd sam proponuje nam chmury, między innymi te tworzone przez Amazon Web Services.

W pracy będziemy wykorzystywać zestaw danych związanych z nowojorskimi taksówkami, możliwy do pobrania z [**22**], jednakże dostępny także i z poziomu chmury Amazon. Na nasze potrzeby wybraliśmy informacje o zielonych taksówkach z 2016 roku. Pliki w formacie CSV obejmują takie dane jak:

- *VendorID* - kod wskazujący dostawcę rekordu;

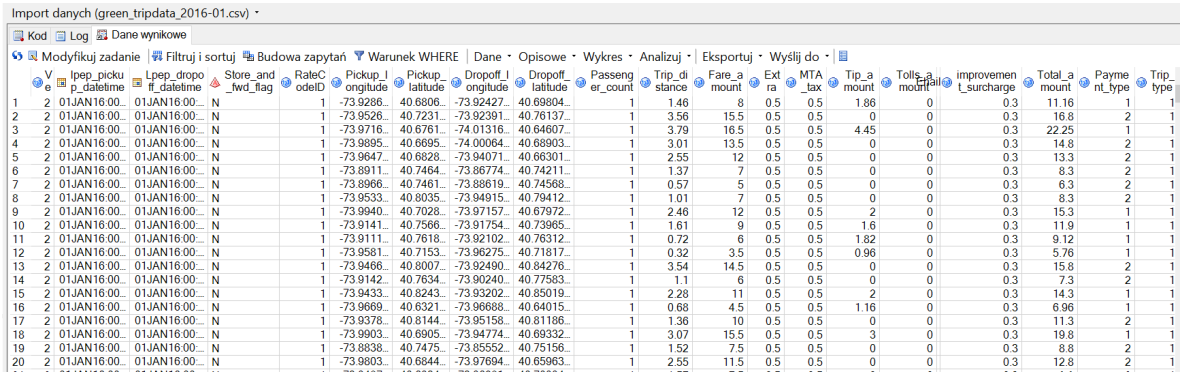
- *lpep_pickup_datetime* - data i godzina początku kursu;
- *lpep_dropoff_datetime* - czas wyłączenia taksometru;
- *Passenger_count* - liczba pasażerów, wprowadzana przez kierującego;
- *Trip_distance* - dystans podróży w milach, liczony przez taksometr;
- *Pickup_longitude* oraz *Pickup_latitude* - lokalizacja początku kursu, długość i szerokość geograficzna;
- *Dropoff_longitude* oraz *Dropoff_latitude* - miejsce zakończenia podróży, długość i szerokość geograficzna;
- *RateCodeID* - kod stawki, obowiązujący pod koniec podróży;
- *Store_and_fwd_flag* - informacja czy dane były przechowywane w pamięci i dopiero wtedy wysłane do serweru (oznaczane poprzez Y), czy też nie.
- *Payment_tipe* - kod liczbowy określający, w jaki sposób pasażer zapłacił za podróż;
- *Fare_amount* - taryfa jakościowa i dystansowa, wyliczona przez licznik;
- *Extra* - różne dodatki i opłaty. Obecnie obejmuje to tylko 0,50$ i 1$ za godziny szczytu oraz noce;
- *MTA_tax* - 0,50$ podatku, który jest automatycznie doliczany w zależności od użytej stawki;
- *Improvement_surcharge* - dodatkowa opłata w wysokości 0,30 $;
- *Tip_amount* - pole określające napiwek, automatycznie wypełniane przy płatności kartą. Brak wpisu dla napiwków pieniężnych;
- *Tolls_amount* - łączna kwota wszystkich opłat za przejazd;
- *Total_amount* - łączna kwota pobierana od pasażerów, nie obejmuje napiwków gotówkowych;
- *Trip_type* - kod określający czy podróż była pośpieszna, czy powolna. Dane dodawane automatycznie na podstawie mierzonej szybkości, ale możliwe do zmienienia przez kierowcę.

Przykładowy sposób wypełnienia zbioru danych, zobrazowany na dwudziestu pierwszych rekordach ze styczniowego zbioru, przedstawiono na rysunku 2.1. Jest to efekt importu danych w SAS Enterprise Guide (w skrócie: SAS EG), w wyniku którego przy ustalaniu atrybutów pól zbadano ponad 1.4 mln (1 445 285) rekordów. Informacje o tychże atrybutach znajdują się na zrzucie ekranu na rysunku 2.2. Po szczegóły na temat środowiska SAS odsyłam do kolejnego rozdziału.

Wskażmy kody odpowiadające poszczególnym przypadkom. W polach oznaczonych *VendorID* istnieją takie możliwości dostawcy:

- 1 - Creative Mobile Technologies, LLC;
- 2 - VeriFone Inc.;

Przy *RateCodeID* mamy do wyboru kody stawki:



	Ve	lpep_pickup_datetime	Lpep_dropoff_datetime	Store_and_fwd_flag	RateCodeID	Pickup_longitude	Pickup_latitude	Dropoff_longitude	Dropoff_latitude	Passenger_count	Trip_distance	Fare_amount	Extra	MTA_tax	Tip_amount	Tolls_amount	Ehail_fee	improvement_surcharge	Total_amount	Payment_type	Trip_type
1	2	01JAN16:00:...	01JAN16:00:...	N	1	-73.9286...	40.6806...	-73.92427...	40.69804...	1	1.46	8	0.5	0.5	1.86	0		0.3	11.16	1	1
2	2	01JAN16:00:...	01JAN16:00:...	N	1	-73.9526...	40.7231...	-73.92391...	40.76137...	1	3.56	15.5	0.5	0.5	0	0		0.3	16.8	2	1
3	2	01JAN16:00:...	01JAN16:00:...	N	1	-73.9716...	40.6761...	-74.01316...	40.64607...	1	3.79	16.5	0.5	0.5	4.45	0		0.3	22.25	1	1
4	2	01JAN16:00:...	01JAN16:00:...	N	1	-73.9895...	40.6695...	-74.00064...	40.68903...	1	3.01	13.5	0.5	0.5	0	0		0.3	14.8	2	1
5	2	01JAN16:00:...	01JAN16:00:...	N	1	-73.9647...	40.6828...	-73.94071...	40.66301...	1	2.55	12	0.5	0.5	0	0		0.3	13.3	2	1
6	2	01JAN16:00:...	01JAN16:00:...	N	1	-73.8911...	40.7464...	-73.86774...	40.74211...	1	1.37	7	0.5	0.5	0	0		0.3	8.3	2	1
7	2	01JAN16:00:...	01JAN16:00:...	N	1	-73.8966...	40.7461...	-73.88619...	40.74568...	1	0.57	5	0.5	0.5	0	0		0.3	6.3	2	1
8	2	01JAN16:00:...	01JAN16:00:...	N	1	-73.9533...	40.8035...	-73.94915...	40.79412...	1	1.01	7	0.5	0.5	0	0		0.3	8.3	2	1
9	2	01JAN16:00:...	01JAN16:00:...	N	1	-73.9940...	40.7028...	-73.97157...	40.67972...	1	2.46	12	0.5	0.5	2	0		0.3	15.3	1	1
10	2	01JAN16:00:...	01JAN16:00:...	N	1	-73.9141...	40.7566...	-73.91754...	40.73965...	1	1.61	9	0.5	0.5	1.6	0		0.3	11.9	1	1
11	2	01JAN16:00:...	01JAN16:00:...	N	1	-73.9111...	40.7618...	-73.92102...	40.76312...	1	0.72	6	0.5	0.5	1.82	0		0.3	9.12	1	1
12	2	01JAN16:00:...	01JAN16:00:...	N	1	-73.9581...	40.7153...	-73.96275...	40.71817...	1	0.32	3.5	0.5	0.5	0.96	0		0.3	5.76	1	1
13	2	01JAN16:00:...	01JAN16:00:...	N	1	-73.9466...	40.8007...	-73.92490...	40.84276...	1	3.54	14.5	0.5	0.5	0	0		0.3	15.8	2	1
14	2	01JAN16:00:...	01JAN16:00:...	N	1	-73.9142...	40.7634...	-73.90240...	40.77583...	1	1.1	6	0.5	0.5	0	0		0.3	7.3	2	1
15	2	01JAN16:00:...	01JAN16:00:...	N	1	-73.9433...	40.8243...	-73.93202...	40.85019...	1	2.28	11	0.5	0.5	2	0		0.3	14.3	1	1
16	2	01JAN16:00:...	01JAN16:00:...	N	1	-73.9669...	40.6321...	-73.96688...	40.64015...	1	0.68	4.5	0.5	0.5	1.16	0		0.3	6.96	1	1
17	2	01JAN16:00:...	01JAN16:00:...	N	1	-73.9378...	40.8144...	-73.95158...	40.81186...	1	1.36	10	0.5	0.5	0	0		0.3	11.3	2	1
18	2	01JAN16:00:...	01JAN16:00:...	N	1	-73.9903...	40.6905...	-73.94774...	40.69332...	1	3.07	15.5	0.5	0.5	3	0		0.3	19.8	1	1
19	2	01JAN16:00:...	01JAN16:00:...	N	1	-73.8838...	40.7475...	-73.85552...	40.75156...	1	1.52	7.5	0.5	0.5	0	0		0.3	8.8	2	1
20	2	01JAN16:00:...	01JAN16:00:...	N	1	-73.9803...	40.6844...	-73.97694...	40.65963...	1	2.55	11.5	0.5	0.5	0	0		0.3	12.8	2	1

RYSUNEK 2.1. Zrzut ekranu z SAS EG - początkowe rekordy danych wynikowych z importu danych styczniowych.

Etykieta	Typ	Informat źródła	Dł.	Format wyniku	Informat wyniku
VendorID	Liczba	BEST1.	8	BEST1.	BEST1.
lpep_pickup_datetime	Data/Czas	ANYDTDTM19.	8	DATETIME18.	DATETIME18.
Lpep_dropoff_datetime	Data/Czas	ANYDTDTM19.	8	DATETIME18.	DATETIME18.
Store_and_fwd_flag	Łańcuch	$CHAR1.	1	$CHAR1.	$CHAR1.
RateCodeID	Liczba	BEST2.	8	BEST2.	BEST2.
Pickup_longitude	Liczba	COMMA19.	8	BEST19.	BEST19.
Pickup_latitude	Liczba	COMMA18.	8	BEST18.	BEST18.
Dropoff_longitude	Liczba	COMMA19.	8	BEST19.	BEST19.
Dropoff_latitude	Liczba	COMMA18.	8	BEST18.	BEST18.
Passenger_count	Liczba	BEST1.	8	BEST1.	BEST1.
Trip_distance	Liczba	COMMA6.	8	BEST6.	BEST6.
Fare_amount	Liczba	COMMA6.	8	BEST6.	BEST6.
Extra	Liczba	COMMA4.	8	BEST4.	BEST4.
MTA_tax	Liczba	COMMA4.	8	BEST4.	BEST4.
Tip_amount	Liczba	COMMA6.	8	BEST6.	BEST6.
Tolls_amount	Liczba	COMMA6.	8	BEST6.	BEST6.
Ehail_fee	Łańcuch	$CHAR1.	1	$CHAR1.	$CHAR1.
improvement_surcharge	Liczba	COMMA4.	8	BEST4.	BEST4.
Total_amount	Liczba	COMMA6.	8	BEST6.	BEST6.
Payment_type	Liczba	BEST1.	8	BEST1.	BEST1.
Trip_type	Liczba	BEST1.	8	BEST1.	BEST1.

RYSUNEK 2.2. Zrzut ekranu z SAS EG - informacje o atrybutach pól.

- 1 - standardowa stawka (Standard rate);
- 2 - związana z portem lotniczym JFK;
- 3 - odnosząca się do portu lotniczego Newark;
- 4 - Nassau lub Westchester;
- 5 - wynegocjonowana stawka;
- 6 - stawka grupowa.

Dla *Store_and_fwd_flag* pojawiają się tylko dwie wartości:

- Y - store and forward trip;
- N - not a store and forward trip.

Informacje opisane w *Payment_type* mogą być opatrzone następującymi oznaczeniami:

- 1 - karta kredytowa;
- 2 - gotówka;
- 3 - bez opłat;
- 4 - spór;
- 5 - nieznane;
- 6 - unieważniony kurs,

Ostatnia kolumna, czyli *Trip_type*, związana jest z:

- 1 - powolna podróż (street-hail);
- 2 - większa prędkość kursu (dispatch).

Dane wykorzystane we wspomnianych zestawach danych zostały zebrane i dostarczone do NYC Taxi and Limousine Commission (TLC) przez dostawców technologii - Taxicab & Livery Passenger Enhancement Programs.

Mamy za sobą zebranie danych. Wiedząc już, nad czym będziemy pracować, możemy cofnąć się do etapu zrozumienia uwarunkowań. Jakie pytania warto zadać, mając wspomniany zbiór informacji?

- W jakiej porze dnia/tygodnia jest największe zapotrzebowanie na taksówki?
- W jakich rejonach jest najwięcej zleceń, a tym samym trzeba postawić więcej pojazdów?
- Czy warto przerzucić większą flotę pod lotniska JFK oraz Newark? Które z nich daje lepsze przychody?
- Czy istnieje zależność pomiędzy lokalizacją początkową (tudzież taryfą) a czasem podróży?
- Czy powyższy czas podróży jest związany z tempem (korkami) czy z odległością?
- W jakich godzinach są największe korki (na podstawie tempa podróży)?
- Jak wygląda zadowolenie klientów (pobieżne spojrzenie na podstawie napiwków)?
- Jak dużo kierowców trzeba zatrudniać na nocne zmiany, aby stanąć naprzeciw zapotrzebowaniu?
- Ile mamy przypadków unieważnienia kursu bądź sporu? Z jakich rejonów/w jakim czasie dnia najczęściej? Czy jest ich na tyle dużo, że powinniśmy porozmawiać o problemie z kierowcami?

Jak widzimy, istnieje wiele możliwych pytań, a to tylko część z nich. Zdarza się, że dopiero w czasie późniejszej pracy wpadamy na pewne pomysły, niewidoczne przy pierwszych próbach analizy zbioru.

ROZDZIAŁ 3

Oprogramowanie

INFORMACJA 2. *Niniejszy rozdział stanowi omówienie wykorzystanego w pracy oprogramowania. Znajdują się tu informacje o wybranej wersji, wymaganiach sprzętowych oraz sposobie instalacji. Ponadto pojawia się krótkie zarysowanie sposobu działania danej platformy programistycznej czy środowiska.*

Na kolejnych stronach znajdują się następujące podrozdziały:

- *Apache Hadoop;*
- *Apache Ambari;*
- *Apache Zeppelin;*
- *środowisko SAS.*

Każdy z nich odnosi się do konkretnego programu, podanego w tytule.

Zacznijmy od krótkiego zdefiniowania otwartego oprogramowania, zwanego także open-source software bądź OSS. Jest to pewnego rodzaju odłam wolnego oprogramowania (ang. free software). Aplikacje działające na takiej licencji dają dostęp do kodu źródłowego, wraz z możliwością jego badania, zmiany czy późniejszego rozpowszechniania każdemu i dla dowolnych celów. Za popularnością tego typu aplikacji stoją niskie koszty oraz liczna społeczność, pracująca nad ulepszeniami i lukami. Do tworów działających na takiej licencji zaliczamy OpenOffice.org oraz Linux.

W 1999 roku powstała organizacja Apache Software Foundation, mająca na celu wspomaganie projektów działających na otwartej licencji. Działa ona charytatywnie, nadzorując ponad 350 projektów OSS.

Wśród nich odnajdziemy platformy programistyczne, zwane także framework. Są to szkielety do budowy aplikacji. Programista uzyskuje dzięki niemu strukturę programu i ogólny mechanizm jego działania. Przy pomocy dostarczanych komponentów i bibliotek może on stworzyć pełną aplikację oraz dostosować ją do swoich celów.

1. Apache Hadoop

Przejdźmy do głównego programu, na którym będziemy pracować przy tej pracy. Trzeba przy tym zauważyć, że Apache Hadoop nie jest zwykłym programem, a właśnie platformą programistyczną, działającą na zasadzie otwartego oprogramowania. Wskazuje na to słowo "Apache", odnoszące się do Apache Software Foundation. Stanowi on jeden z wielu projektów tejże fundacji, napisany w języku Java.

Rysunek 3.1. Logo Apache Hadoop

Tym samym jego instalacja musi być poprzedzona uzyskaniem oprogramowania Java (ostatnie rekomendacje wskazują na wersję Java 8 - stan na 05.2017).

Początki Hadoopa sięgają roku 2003, a konkretniej - Google File System (GFS). Jest to rozproszony system plików, przeznaczony dla aplikacji przetwarzających większe ilości danych. Powstał on na potrzeby systemu indeksowania firmy Google. Wraz z GFS pojawiło się kolejne rozwiązanie Google - MapReduce: uproszczony proces przetwarzania danych w dużych klastrach. Rozwój tychże rozwiązań rozpoczął się w ramach projektu Apache Nutch, ale w styczniu 2006 roku został przeniesiony do nowego podprojektu. Swoją nazwę - Hadoop - zawdzięcza pracującemu w tym czasie w Yahoo! Doug Cuttingowi. Zaczerpnął on ją od maskotki-słonika swego syna, którą możemy teraz zobaczyć w logo omawianej platformy programistycznej (patrz: rysunek nr 3.1).

Pierwsze pełne wydanie pojawiło się pod koniec 2011 roku, jednakże już 5 lat wcześniej istniał Hadoop 0.1.0. Z tejże odmiany platformy korzystała firma Yahoo!, która jeszcze w początkowych miesiącach podłączyła 600 komputerów do klastra. W 2008 roku Hadoop pomógł ustanowić światowy rekord przesortowania terabajta danych. Przy wykorzystaniu 910 nodów zajęło to 209 sekund, co rok później spadło do 68 sekund.

Hadoop to narzędzie typu NoSQL, jednocześnie dające możliwość rozproszonego składowania oraz przetwarzania wielkich zbiorów danych. Wykorzystywane są do tego klastry komputerowe oraz proste modele programowania. Występuje przy tym skalowanie od pojedynczych serwerów do tysięcy maszyn, z zastosowaniem ich lokalnej mocy obliczeniowej i zdolności przechowywania informacji.

Skalowanie związane jest z łatwością rozbudowy lub zmniejszenia niektórych operacji, w zależności od zapotrzebowania. Oznacza to, że tych samych narzędzi możemy używać do danych małych jak i dużych. Przyjmujemy przy tym metodologię budowania narzędzi jednokrotnie, z ewentualnym dopracowywaniem ich w drodze zapotrzebowania.

Projekt Apache Hadoop obejmuje następujące moduły:

- Hadoop Common - biblioteki i narzędzia dla pozostałych modułów;
- Hadoop Distributed File System (HDFS) - rozproszony system plików;
- Hadoop YARN - platforma służąca do zarządzania zasobami klastra;
- Hadoop MapReduce.

System plików rozproszonych Hadoop związany jest na ogół z bardzo dużymi zestawami danych (liczonych w terabajtach) oraz wieloma węzłami w jednym klastrze. Oznacza to, że musi on zeskalować dane nawet do setek maszyn, a jednocześnie umożliwiać ich dobrą przepustowość. Ponadto HDFS wykazuje szybką wykrywalność błędów oraz pomaga zmniejszyć przeciążenie sieci. Związane jest to z założeniem, że często lepiej jest migrować obliczenia bliżej miejsca, w którym znajdują się dane.

HDFS ma architekturę master/slave. Klaster zawiera w sobie węzeł nadzorczy NameNode. Zarządza on obszarem systemu plików i reguluje dostęp do plików przez klientów. Ponadto NameNode wykonuje takie operacje jak otwieranie, zamykanie i zmiana nazwy plików oraz katalogów. System został przy tym tak zaprojektowany, aby dane nigdy nie przepływały przez węzeł nadzorczy. NameNode stanowi repozytorium dla wszystkich metadanych HDFS i określa jak wygląda podział plików na bloki. Człony te są tworzone zgodnie z instrukcją z NameNode, a także przechowywane, w węzłach danych zwanych DataNodes. Co więcej, DataNodes są odpowiedzialne za obsługę żądań klienta HDFS odnośnie odczytu i zapisu.

MapReduce to platforma do przetwarzania równoległego dużych zbiorów danych w klastrach. Jej nazwa wzięła się od dwóch funkcji programowania funkcyjnego - map i reduce. Tak też nazywają się dwa kroki, w których realizowane są operacje:

- krok MAP;
- krok REDUCE.

Pierwszy z nich odnosi się do podziału zestawu danych wejściowych (problemów) na niezależne kawałki. W kroku reduce podproblemy są przetwarzane równolegle.

Zwykle węzły obliczeniowe i węzły magazynowania są takie same, to znaczy, że MapReduce i system plików rozproszonych Hadoop działają na tym samym zestawie węzłów. Konfiguracja ta pozwala platformie skutecznie zaplanować zadania na węzłach, w których dane są już obecne, co powoduje bardzo dużą przepustowość całego klastra.

Poza wymienionymi modułami Hadoopa, istnieje wiele innych projektów, podłączanych pod niego. Należy do nich między innymi Apache Hive. Jest to oprogramowanie hurtowni danych, ułatwiające czytanie, pisanie i zarządzanie dużymi zbiorami danych, przechowywanych w sieciowym systemie plików z SQL.

Hadoop jest wykorzystywany zarówno do celów badawczych, jak i komercyjnych przez wiele firm. Są one przy tym zachęcane do wpisywania się na stronę PoweredBy [**12**]. Na liście tam zamieszczonej odnajdziemy m.in. takie giganty jak Google, Facebook i Twitter. Być może takich firm byłoby więcej, gdyby nie pewne

bariery - takie jak na przykład problemy ze znalezieniem wykwalifikowanego eksperta od Hadoopa.

1.1. Instalacja. Początki z oprogramowaniem platformy Hadoop związane są ze stosunkowo niestandardową instalacją. Spowodowane jest to tym, że oficjalna wersja Hadoop nie posiada odpowiednich plików binarnych dla systemu Windows. Mamy zatem dwie możliwości:

- Wykorzystanie systemu Linux;
- Samodzielne zbudowanie szkieletu startowego dla systemu Windows.

Niestety drugi punkt prowadzi do nieco skomplikowanego procesu. By go przeskoczyć, możemy skorzystać z gotowych plików instalacych, które będą już odpowiednio przez kogoś skonfigurowane pod system Windows. Trzeba przy tym pamiętać, że bazowanie na czyjejś pracy może skutkować nietypowymi błędami i problemami z pewnymi funkcjami. Jeśli nic nie stoi nam na przeszkodzie, najlepiej stworzyć maszynę wirtualną z systemem Linux, tudzież zainstalować go jako drugi system na komputerze. Będzie to skutkowało nie tylko pracą w oryginalnym systemie, ale także utrzyma nas w głównych zasadach Big Data - tanie, ogólnodostępne oprogramowanie.

Wymagania:

- możliwość wirtualizacji systemu BIOS;
- maszyna wirtualna wraz z wgranym systemem Linux;
- zainstalowana najnowsza wersja Java JDK lub gotowość do jej instalacji.

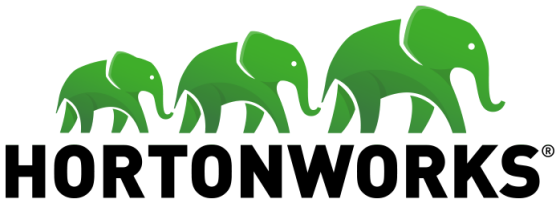
Więcej szczegółów, włącznie z dokładnym omówieniem krok po kroku procesu instalacji, znajduje się w referencji [**10**]. Warto także zajrzeć do oficjalnej dokumentacji - strona [**13**].

Dodatkowym sposobem dojścia do omawianej platformy programistycznej jest skorzystanie z HortonWorks Sandbox. Stanowi on przenośną wersję Apache Hadoop, którą możemy zainstalować w dowolnym systemie - Windows, Linux czy Mac OS (z zachowaniem wymagań). Po więcej informacji odsyłam do kolejnej sekcji.

2. HortonWorks Sandbox

Ciekawe rozwiązanie udostępnia firma Hortonworks, której nazwa jak i logo (patrz: rysunek nr 3.2) nawiązują do Apache Hadoop. Została ona założona w 2011 roku przez osoby pracujące niegdyś w Yahoo! nad omawianą platformą programistyczną. Ich doświadczenie zaowocowało stworzeniem jedynej w pełni otwartej i przenośnej dystrybucji Apache Hadoop. HortonWorks Sandbox, bo o nim mowa, zgodnie z oficjalnymi zaleceniami wymaga jedynie 15 minut pracy nad instalacją.

Sandbox to proste, wstępnie skonfigurowane środowisko, zawierające najnowsze osiągnięcia Apache Hadoop, w szczególności dystrybucję Hortonworks Data



Rysunek 3.2. Logo Hortonworks

Platform (HDP). Poza tym mamy do dyspozycji inne technologie dostępne w omawianej platformie - takie jak system plików rozproszonych, MapReduce, Pig, Hive, HBase i tak dalej. Oznacza to, że będziemy mogli przechowywać, przetwarzać i analizować duże ilości danych, pochodzących z wielu źródeł.

To wszystko jest dostępne w postaci wirtualnego środowiska, które możemy uruchomić w chmurze lub na naszym komputerze.

Podsumowanie wymagań w przypadku wersji HDP_2.6_virtualbox:

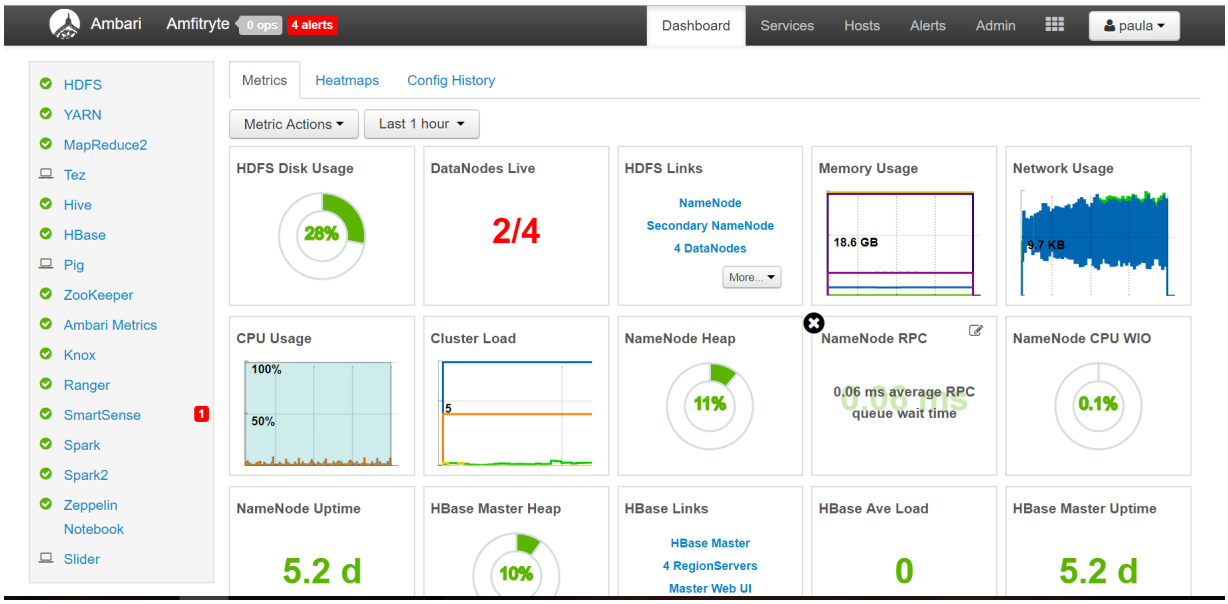
- 64-bitowy system operacyjny;
- przynajmniej 8GB pamięci RAM;
- przynajmniej 10,4 GB miejsca na dysku;
- możliwość wirtualizacji systemu BIOS;
- program Putty;
- przeglądarka internetowa;
- HortonWorks Sandbox.

Po informacje jak zaczynać w HortonWorks Sandbox odsyłam do tutorialu [**14**].

3. Apache Ambari

Jak wspomnieliśmy niemal na początku tekstu, na jego potrzeby uzyskaliśmy dostęp do klastra komputerowego wydziału FTiMS Politechniki Gdańskiej. Jego zarządzanie i monitorowanie odbywa się przy pomocy Apache Ambari (wersja 2.5.0.3). Jest to kolejny projekt Apache, mający na celu uproszczenie zarządzania Hadoopem. Mówiąc bardziej szczegółowo, Ambari daje możliwość:

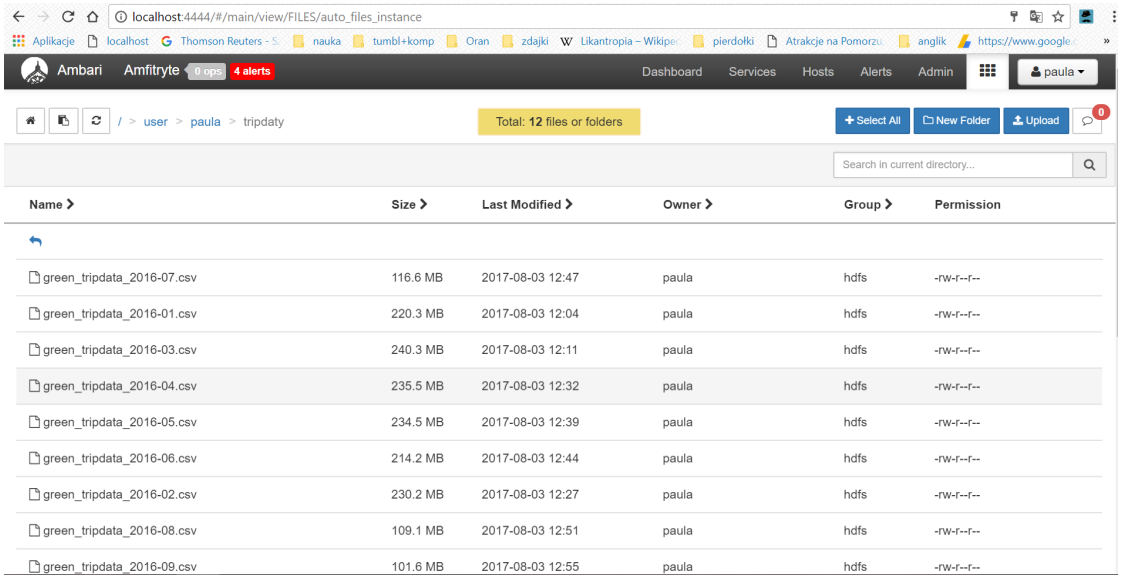
- skorzystania z kreatora instalacji usług Hadoopa na dowolnej liczbie hostów;
- zarządzania klastrem - uruchamanie, zatrzymywanie i konfiguracja usług w całym klastrze;
- monitorowania klastra, za pomocą następujących systemów:
  - tablica kontrolna stanu i statusu klastra;
  - system pomiarów Ambari Metrics System do zbierania danych;
  - system ostrzegania Ambari Alert Framework, alarmujący o błędach i innych wartych uwagi sprawach (na przykład zbyt małej ilości wolnego miejsca na dysku).



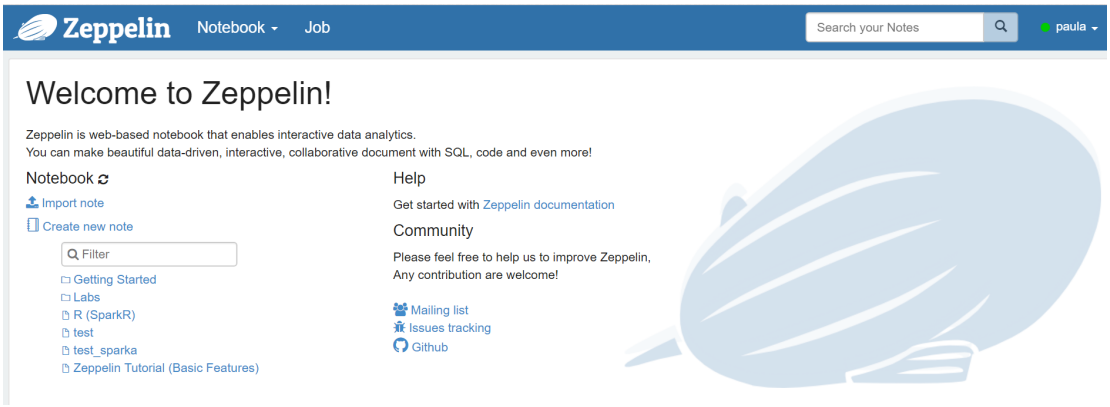
RYSUNEK 3.3. Strona główna Ambari

Strona główna Ambari została zaprezentowana na rysunku 3.3. Widzimy na nim trzy główne segmenty. Pierwszy z nich, szary prostokąt z lewej strony, prezentuje listę usług i ich stan włączenia. Widnieje wśród nich między innymi notatnik Zeppelin, który omówimy w dalszej części tekstu. Główną część okna zajmuje wspomniana tablica stanu, z której odczytamy najważniejsze parametry - na przykład użycie internetu czy zajęcie dysku przez pliki z rozproszonego systemu plików HDFS (w momencie robienia zrzutu ekranu 28%). Do tychże plików dostajemy dostęp poprzez kliknięcie siatki dziewięciu kwadratów, zaraz na końcu czarnego pasku na górze strony. Po wybraniu Files View przechodzimy do zbioru folderów. Jeden z nich, stworzony specjalnie dla potrzeb pracy, można zobaczyć na rysunku 3.4. Umieściliśmy w nim zbiory danych, na których będziemy pracować, a których zawartość opisaliśmy powyżej. Interface Ambari daje wgląd w ich nazwę, rozmiar, datę ostatniej modyfikacji, właściciela oraz uprawnienia (permission). Ostatnia kolumna oznacza tylko tyle, że odpowiednie grupy osób mogą czytać (oznaczane jako *r* od angielskiego *read*) i edytować pliki (*w*, ang. *write*). Możliwe do edycji na górze okna.

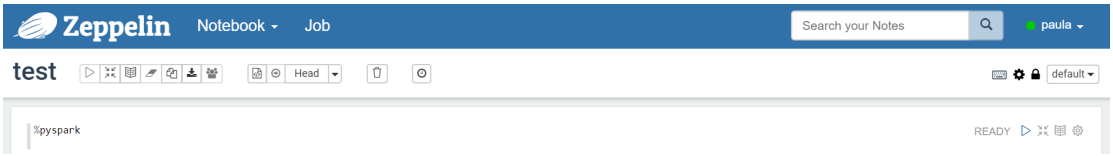
Do pisania kodów wykorzystaliśmy notatnik Zeppelin (patrz: rysunek 3.5). Udostępnia on możliwość pisania w wielu językach, takich jak R czy SQL. Wybraliśmy odmianę Pythona, która w środowisku Sparka określana jest PySparkiem. Wiąże się to z tym, że po rozpoczęciu pisania, każda z komórek powinna zostać rozpoczęta komendą *%pyspark* (patrz: rysunek 3.6) bądź *%spark2.pyspark*. Podobnie wygląda w przypadku użycia języka SQL (*%sql*) oraz innych języków programistycznych. Do powłoki systemowej dostajemy się przy pomocy *%sh*.



RYSUNEK 3.4. Widok na jeden z folderów plików w Ambari



RYSUNEK 3.5. Widok na stronę główna notatnika Zeppelin.



RYSUNEK 3.6. Podstawowy wygląd notatnika Zeppelin

4. SAS

Flagowy produkt firmy SAS stanowi wszechstronne rozwiązanie statystyczne i programistyczne. Najważniejsze (z naszego punktu widzenia) możliwości tego środowiska możemy zamknąć w następujących punktach:

- analiza statystyczna;
- ogólna praca przy danych - import, przetwarzanie, zarządzanie, eksploracja;
- tworzenie raportów, grafiki.

Uzyskamy to przy pomocy ponad 200 składowych oprogramowania.

Pakiet SAS był rozwijany na Uniwersytecie Stanowym Karoliny Północnej od 1966 roku. Początkowo projekt miał na celu analizę danych dotyczących rolnictwa i poprawę plonów, a za jego sterem stali Anthony Barr oraz student James Goodnight. Pierwsze wydania pojawiały się w latach 1971 oraz 1972, jednakże pełna komercyjna wersja została wydana dopiero w 1976 roku. W 2004 roku dodano znany interface point-and-click dla mniej zaawansowanych użytkowników.

Aby korzystać z zalet analizy statystycznej w oprogramowaniu SAS, dane muszą być wprowadzane w formacie tabeli arkusza kalkulacyjnego lub w formacie SAS. Występują przy tym tylko dwa typy danych - numeryczne oraz znakowe, a w razie ich braku, zostają zastępowane - odpowiednio - przez kropkę '.' lub puste pole.

Typowy program SAS składa się z dwóch bloków instrukcji (tzw. steps), występujących w tej kolejności:

- DATA step - związany z odczytem i modyfikacją danych;
- PROC step - służący do analizy danych oraz tworzenia raportów.

Dane SAS możemy publikować w formacie HTML, PDF, Excel, jak i wielu innych.

SAS Institute udostępnia pełną dokumentację w języku polskim na swojej stronie internetowej. Po szczegóły instalacyjne dla najnowszej wersji oprogramowania statystycznego (stan na 05.2017) odsyłam na stronę [**27**]. Szczegółowe wymagania sprzętowe znajduję się w pozycji [**28**].

Na potrzeby tejże pracy wybrano srodowisko SAS w wersji 9.4 TS Level 1M4.

ROZDZIAŁ 4

SAS & Hadoop

INFORMACJA 3. *Następujący rozdział przedstawia sposób połączenia zalet drzemiących w dwóch narzędziach - środowisku SAS oraz platformie programistycznej Apache Hadoop. Tym samym kolejne strony będą odnosić się do SAS Data Loader for Hadoop. Zaprezentujemy na nich zalety tego rozwiązania oraz sposób instalacji.*

Wspomniany SAS Data Loader for Hadoop (SDLfH) to rozwiązanie, stworzone przez SAS Institute. Zgodnie z oficjalnymi informacjami, przy pomocy tego „złącza" możemy przechowywać dane tam, gdzie są one nam potrzebne, ładując je z lub do Hadoopa. Daje nam to ich gotowość do użycia na potrzeby raportów, wizualizacji oraz zaawansowanej analizy statystycznej. Robimy to samodzielnie, bez potrzeby pisania skomplikowanego kodu MapReduce czy szukania zewnętrznej pomocy - co potencjalnie sprowadzałoby się do zatrudnienia kolejnej osoby. SDLfH jest proste w obsłudze, szybkie, a co za tym idzie, wysokie umiejętności specjalistów mogą zostać wykorzystane także i w innych zadaniach.

SDLfH wykorzystuje moc Hadoopa, a funkcje działają w pamięci Spark, co zapewnia lepszą wydajność. Zminimalizowanie ruchu danych zwiększa ich bezpieczeństwo. Jednocześnie szeroka funkcjonalność środowiska SAS pozwala uzyskać jeszcze więcej z naszych zbiorów informacji.

Podsumowując powyższe, omawiane narzędzie pociąga za sobą cztery główne korzyści:

- zarządzanie danymi bez specjalistycznej wiedzy;
- szybkość, wszechstronność pracy;
- zwiększenie efektywności;
- uzyskanie więcej wartości z zasobów Big Data (zaawansowana analityka).

Spójrzmy na kluczowe cechy SDLfH. Zacznijmy od możliwości przekształcania i przenoszenia danych z Hadoopa. Dzięki omawianemu rozwiązaniu kopiujemy relacyjne bazy i zestawy danych SAS do i z Hadoopa. Importujemy dane, między innymi w formacie CSV (jak pamiętamy, jest to format wybranego zbioru danych). Usuwamy wiersze w tabelach, przekształcamy dane przy pomocy filtrowania, zarządzamy kolumnami - w tym przenosimy i grupujemy wybrane, podsumowujemy wiersze. Ponadto przy pomocy SAS/ACCESS libraries (biblioteki) uzyskujemy dostęp do dziesiątków chmur oraz źródeł danych (zarówno relacyjnych jak i big data).

Korzystając z SDLfH oczyszczamy dane w Hadoopie. Ujednolicamy, eliminujemy powtórzenia w zbiorze i analizujemy dane, wyodrębniamy wzorce i cechy. Dostajemy dostęp do inteligentnego filtrowania, stawiamy zapytania oraz sortujemy dane w istniejących tabelach Hadoop. Kolejne przyspieszenie pracy uzyskujemy poprzez określenie typu danych w kolumnie na podstawie wartości w niej podanych.

Co ważne, nie potrzebujemy znajomości SQL, intuicyjnie zmieniamy tabelę czy dołączamy ją do innych. Agregacja następuje przy tym na wybranych kolumnach, a dane źródłowe w każdej chwili filtrujemy. Użytkownicy zaawansowani mogą generować i edytować kwerendę HiveQL lub wkleić istniejące zapytanie.

Proces zarządzania danymi dodatkowo przyspieszamy dzięki Spark. Wydajność jest zwiększana poprzez to, że funkcje jakości danych działają w pamięci tejże platformy obliczeniowej. W razie potrzeby odczytamy zbiory danych przy użyciu Sparka.

SDLfH to zbiór kilku produktów od firmy SAS:

- SAS Data Loader;
- SAS/ACCESS Interface to Hadoop;
- SAS In-Database Code Accelerator for Hadoop;
- SAS Data Quality Accelerator for Hadoop.

SDLfH działa wewnątrz maszyny wirtualnej lub vApp. Drugie rozwiązanie jest uruchamiane i zarządzane przez hipernadzorcę o nazwie VMware Player Pro. Udostępnia on adres internetowy, który otwieramy przy pomocy przeglądarki internetowej. Zostanie wówczas załadowane okno SAS Data Loader: Information Center. W nim skonfigurujemy połączenie oraz sprawdzimy dostępne aktualizacje. Co najważniejsze, centrum informacji uruchomi w nowej karcie aplikację internetową SAS Data Loader.

1. Informacje instalacyjne

SDLfH dostarczany jest zarówno dla systemu Windows, jak i dla Linuxa. W pierwszej wersji będziemy potrzebowali 1160 MB miejsca na dysku, zaś w drugim - 688 MB. Przy wykorzystaniu dystrybucji Linuxa musimy się liczyć z posiadaniem na komputerze następujących narzędzi:

- bazowy SAS 9.4;
- DATA Step to DS2 Translator 3.1;
- SAS DS2 Extensions for Data Quality 9.43;
- SAS Data Management Data Server 2.1;
- SAS Data Management Infrastructure Threaded Kernel Extensions 3.1;
- SAS Data Profile Engine 3.1;
- SAS Data Quality Accelerator for Hadoop 9.44;

- SAS In-Database Code Accelerator for Hadoop 9.43;
- SAS In-Database Technologies for Hadoop 3.1;
- SAS Threaded Kernel for Data Quality 9.42;
- SAS Web Infrastructure Platform Data Server 9.4;
- SAS/ACCESS Interface to Hadoop 9.43.

W wersji windowsowej sprawa wygląda analogicznie, niestety część funkcji jest wówczas niedostępna.

Instalacja SDLfH wiąże się z przymusem współpracy administratora klastra hadoopowego oraz osoby odpowiedzialnej za środowisko SAS. Jest to spowodowane potrzebą wymiany pomiędzy sobą informacji, a także i plików. Dotyczy to między innymi udostępnienia odpowiednich plików rar z folderu SAS Software Depot (o czym później). Jednocześnie zarządca klastra hadoopowego dostarcza Hadoop JAR oraz pliki konfiguracyjne Hadoopa (ang. Hadoop client files) przed stworzeniem aplikacji sieciowej SDLfH. Sprawę tę można ułatwić i przyspieszyć poprzez skorzystanie z SAS Deployment Wizard. Takie ułatwienie wymaga jednakże uprawnień administratorskich dla Ambari, bądź innego używanego manadżera klastra hadoopowego.

Na początku współpracy trzeba rozpatrzeć potrzebę wdrożenia opcjonalnych, ale często zalecanych spraw. Należy do nich między innymi zdecydowanie się na używanie sterowników JDBC (łącze do baz danych w języku Java, ang. Java DataBase Connectivity) dla dostępu do systemu zarządzania bazą danych (ang. Database Management System, DBMS). SDLfH wykorzystuje bowiem wybrane łącza do przenoszenia danych do i z bazy danych. Może jednakże korzystać z połączeń JDBC do bezpośredniego dostępu do DBMS oraz do wybierania tabel źródłowych, tabel docelowych i schematów docelowych. Szczegóły znajdują się w referencji [**31**].

Jak wspomnieliśmy, osoba odpowiedzialna za SAS powinna dostarczyć pliki rar administratorowi klastra. Mieszczą się w folderach SAS Software Depot (oczywiście o ile wcześniej zostało zainstalowane to oprogramowanie):

- SAS-Software-Depot\standalone_installs\
  SAS_Core_Embedded_Process_Package_for_Hadoop\12_0\
  Hadoop_on_Linux_x64\en_sasexe.zip;
- SAS-Software-Depot\standalone_installs\
  SAS_Data_Loader_for_Hadoop_Spark_Engine\3_1\
  Hadoop_on_Linux_x64\en_sasexe.zip.

Pojawiające się w powyższych lokalizacjach liczby (12_0 oraz 3_1) mogą się różnić w zależności od używanej wersji oprogramowania.

Poza uzyskaniem podanych plików, administrator klastra hadoopowego powinien:

- sprawdzić czy dana wersja Hadoop znajduje się na liście wspieranych przez SDLfH (na przykład pod [**30**]);
- posiadać praktyczną wiedzę na temat systemów plików rozproszonych HDFS, MapReduce, YARN, Hive i HiveServer2;
- upewnić się, że w klastrze są uruchomione usługi HCatalog, HDFS, Hive, MapReduce, Oozie i YARN;
- zainstalować Oozie 4.0 lub nowszy;
- skonfigurować klaster pod skrypty Oozie;
- poznać lokalizację domową MapReduce;
- znać nazwę hosta serwera Hive i Namenode;
- określić gdzie działa HDFS i serwery Hive. Jeśli serwer Hive nie działa na tej samej maszynie co NameNode, zarejestrować serwer i numer portu serwera Hive dla przyszłej konfiguracji;
- uzyskać pozwolenie na zrestartowanie usługi Yarn;
- sprawdzić czy można korzystać z powodzeniem z MapReduce;
- skontrolować czy istnieje połączenie z klastrem hadoopowym z jego komputera, ale poza środowiskiem SAS, za pomocą zdefiniowanego protokołu zabezpieczeń;
- dowiedzieć się czy klaster hadoopowy jest zabezpieczony przez Kerberos (w takiej sytuacji zastosować się do [**31**]).

W tym miejscu możemy wyjawić, iż wykorzystany klaster nie stosuje zabezpieczenia Kerberos.

Kolejna faza pracy zależy od tego, czy korzystamy z Ambari, Clouder Manager, czy Shell Script. W tekście tym opiszemy sytuację przy pierwszej opcji.

Administrator klastra hadoopowego rozpakowuje plik *en_ sasexe.zip*, którego ścieżka została podana wyżej w pierwszej kolejności. Następnie stwarza podkatalog katalogu tmp (*/tmp/sasep*) i kopiuje STACK DIRECTORY (z pliku zip) do tegoż miejsca (*cp -r sasep\sasexe\stack \tmp\sasep*). Idąc dalej, uruchamia skrypt *install-sasepstack.sh* z utworzonego katalogu:

- dodaje uprawnienia wykonywania (ang. execute) do skryptu: *chmod +x install-sasepstack.sh*;
- uruchamia skrypt z dostępem sudo lub root: *.\install_ sasepstack.sh ambariAdminUsername*;
- restartuje Ambari.

Po ponownym zalogowaniu do Ambari administrator klika Actions i wybiera +Add Service. Dzięki temu pojawi się okno kreatora dodawania usług i panel wyboru usług. W tymże panelu klika SASEP i Next. W panelu Assign Slaves and Clients zaznacza węzły, w których ma być rozmieszczony stos. Nie może przy tym

pominąć NameNode. Bez zmieniania opcji w Customize Services przechodzi dalej i sprawdza informacje w oknie Review. Jeśli wszystko jest w porządku, klika Deploy. Po pomyślnym zainstalowaniu wszystkiego pojawi się okno Summary oraz przycisk Complete.

Na skutek powyższych kroków usługa SAS Embedded Process skończy się instalować na wszystkich nodach klastra i pojawi się na panelu Ambari. Należy jeszcze tylko sprawdzić czy w systemie plików Hadoop znajduje się plik *\sas\ep\config\ep-config.xml*.

Przejdźmy do drugiego pliku. Tak jak z poprzednim, trzeba go rozpakować (np. *unzip en_ sasexe.zip -d sasspark*) i skopiować go do nowego podkatalogu /tmp (*cp -r ep/sasexe/stack /tmp/sasspark*). Administrator uruchamia skrypt *install-sasdmspark.sh* i dodaje uprawnienia. Następnie uruchamia go z dostępem sudo lub root i restartuje Ambari.

Będąc z powrotem w Ambari zaznacza w panelu wyboru usług SASDM-SPARK. Kolejne kroki pokrywają się z powyżej wymienionymi. Na końcu trzeba jeszcze zweryfikować czy istnieje plik *\sas\ep\config\dmp-config.xml*.

Po powyższych czynnościach administrator Hadoopa powinien wykonać konfigurację poinstalacyjną dla SAS Embedded Process. W pierwszej kolejności potrzebna jest możliwość otwierania usługi HCatalog.

HCatalog to tabelowa i magazynowa warstwa zarządzająca dla Hadoopa. Umożliwia użytkownikom z różnymi narzędziami przetwarzania danych (Pig, MapReduce) łatwiejsze odczytywanie i zapisywanie danych w sieci. HCatalog jest częścią Hive. Dzięki niemu struktury danych, które są zarejestrowane w Hive (w tym dane w formacie SAS), można uzyskać za pośrednictwem kodu Hadoop. SAS Embedded Process for Hadoop wykorzystuje HCatalog do przetwarzania złożonych formatów plików.

Jeśli używana dystrybucja korzysta z MapReduce 2 i YARN, SAS Embedded Process automatycznie ustawia HCatalog CLASSPATH w pliku *ep-config.xml*. W przeciwnym razie trzeba dodać ręcznie pliki jar HCatalog do biblioteki MapReduce2 lub Hadoop CLASSPATH.

Kolejny krok to dodanie CLASSPATH aplikacji YARN do pliku konfiguracyjnego. We wszystkich węzłach klastra hadoopowego, które mają plik *yarn-site.xml*, dwie główne właściwości konfiguracyjne określają CLASSPATH aplikacji: *yarn.application.classpath* oraz *mapreduce.application.classpath*. Jeśli nie określi się CLASSPATH YARN, MapReduce przyjmuje domyślną CLASSPATH. Jeśli jednak zostanie określona CLASSPATH MapReduce, CLASSPATH aplkacji YARN zostanie zignorowana. SAS Embedded Process for Hadoop wymaga CLASSPATH obu aplikacji. Aby zapewnić istnienie CLASSPATH aplikacji YARN, musimy ręcznie dodać to do pliku *yarn-site.xml*.

Domyślne CLASSPATH aplikacji YARN dla Windowsa:

- *%HADOOP_CONF_DIR%;*
- *%HADOOP_COMMON_HOME%/share/hadoop/common/*;*
- *%HADOOP_COMMON_HOME%/share/hadoop/common/lib/*;*
- *%HADOOP_HDFS_HOME%/share/hadoop/hdfs/*;*
- *%HADOOP_HDFS_HOME%/share/hadoop/hdfs/lib/*;*
- *%HADOOP_YARN_HOME%/share/hadoop/yarn/*;*
- *%HADOOP_YARN_HOME%/share/hadoop/yarn/lib/*.*

Podobnie dla Linuxa, ale w miejscu pierwszych procentów pojawia się znak dolara ($), zaś środkowe procenty zanikają.

Mając za sobą to wszystko, można przejść do ostatecznej fazy instalacji SDLfH. Zaczyna ją administrator klastra, który musi zebrać hadoopowe pliki JAR oraz instalacyjne. Może ku temu wykorzystać Hadoop tracer script, który tworzy pliki JAR oraz konfiguracyjne w katalogach */tmp/jars* oraz */tmp/sitexmls*.

Wymagania wstępne Hadoop tracer script prezentują się następująco:

- użytkownik uruchamiający skrypt musi posiadać konto UNIX z SSH, hasło albo klucz prywatny do węzła Hive lub NameNode;
- zainstalowany strace oraz Python 2.6 lub nowszy. Należy skontaktować się z administratorem systemu, jeśli te pakiety nie są zainstalowane;
- użytkownik uruchamiający skrypt powinien posiadać uprawnienia do wydawania poleceń HDFS i Hive (sprawdzamy to na przykład poleceniem *time hive –e 'set –v'*);
- jeśli Hadoop jest zabezpieczany przy pomocy Kerberos, trzeba uzyskać identyfikator dla użytkownika przed uruchomieniem skryptu;
- uruchomiony Hadoop oraz Hive w klastrze. Jeśli to się nie uda, skrypt nie uruchomi się.

Aby uruchomić skrypt Hadoop tracer script postępujemy następująco:

- na serwerze hadoopowym tworzymy katalog temp:
  /opt/sas/hadoopfiles/temp;
- wklejamy do przeglądarki internetowej adres pozwalający pobrać *hadooptracer.zip* do powyższego katalogu:
  ftp://ftp.sas.com/techsup/download/blind/access/hadooptracer.zip;
- używając jednej z metod przesyłania (PSFTP, SFTP, SCP lub FTP) przenosimy plik zip do węzła Hive w klastrze;
- rozpakowujemy plik, a następnie zmieniamy uprawnienia zawartego w nim pliku *hadooptracer.py* na EXECUTE: *chmod 755 ./hadooptracer.py*;
- wpisujemy komendę: *python hadooptracer.py –filterby=latest –postprocess -f/tmp/sitexmls/hadooptracer.json;*

- usuwamy następujące pliki (o ile istnieją): *derby*.jar, spark-examples*.jar, ranger-plugins-audit*.jar, avatica*.jar, hadoop-0.20.2-dev-core*.jar.*

Po uzyskaniu katalogów */tmp/jars* oraz */tmp/sitexmls* administrator SAS może zacząć swoją pracę. Powinien on zweryfikować zmienne środowiskowe związane z programem Hadoop. Oznacza to, że trzeba wywołać SAS. W zależności od wybranego systemu, prowadzi to do:

- UNIX: uruchomienia w command shell na serwerze z SAS Data Loader Workspace Server:
  /usr/local/sas/config/Lev1/SASApp/WorkspaceServer/WorkspaceServer.sh;
- Windows: uruchomienia jako administrator:
  C:\SAS \Lev1\SASApp\ WorkspaceServer\ WorkspaceServer.bat

Będąc już w środowisku SAS, wpisuje *PROC options; run;*. Następnie sprawdza, czy ścieżki *SAS_ HADOOP_ JAR_ PATH* i *SAS_ HADOOP_ CONFIG_ PATH* są wymienione tylko raz w dzienniku SAS oraz usuwa możliwe duplikaty.

Na końcu, osoba odpowiedzialna za środowisko SAS z pomocą administratora klastra sprawdza ustawienia Hadoop w plikach *inventory.json* i *ep-config.xml*. Pierwszy z nich zawiera ustawienia Hadoop dla aplikacji sieciowej SAS Data Loader. Powinien mieścić się w lokalizacji:
SAS-installation-directory\Lev1\ Web\Applications\DataLoader\Hadoop\ Configs\inventory.json

Upewniamy się czy występuje format JSON oraz weryfikujemy zawartość pliku. W przypadku Sparka:

- jeśli wartość przy Sparku to false, zapewne klaster został źle skonfigurowany w tym kierunku;
- sprawdzamy czy wartość dla classpath hcatalog to pojedyncza linia, nie zawierająca nowej linii bądź jej przerwania;
- upewniamy się, że sas_dmarchitech jest dostępny, a ścieżka instalacji jest poprawna. W przeciwnim wypadku instalacja SDLfH Spark Engine będzie problematyczna.

Dla MapReduce ważne jest z kolei, aby wartości hadoop_distro i ustawienia hadoop_internal_name były prawidłowe.

Przejdźmy do drugiego pliku, to jest *ep-config.xml*. Jako plik zawierający ustawienia Hadoopa dla SAS Embedded Process tworzy się on podczas instalacji SAS Embedded Process. Dodawany jest do lokalizacji HDFS (*SAS-installation-directory\sas\ep\config\ep-config.xml*). Korzystając z niej otwieramy plik i sprawdzamy czy przy sas.ep.classpath znajduje się właściwa wersja Hadoopa. Jeśli nie, uruchamiamy skrypt *sasep-admin.sh* z opcją the-genconfig.

Nie można zapomnieć o poinstalacyjnych zadaniach, związanych z końcową konfiguracją środowiska SAS. Niektóre z tych czynności nie są obowiązkowe, lecz wskazane. Do takich należy między innymi sprawdzenie zmiennych środowiskowych związanych z Hadoopem. Sprowadza się to do ponownego usunięcia potencjalnych duplikatów SAS_HADOOP_JAR_PATH i SAS_HADOOP_CONFIG_PATH. Powinniśmy także powtórzyć weryfikację ustawień Hadoopa - patrz: czynności z plikami *inventory.json* oraz *ep-config.xml*.

W przypadku gdy nie zostało to wcześniej wykonane, określamy przynajmniej jedno konto użytkownika SAS jako administrator Data Loader. W sytuacji zaś gdy widzimy taką potrzebę, możemy wykorzystać aktualizację SAS Workspace Server dla międzynarodowych danych. Dzięki temu wyeliminujemy błędy wynikające z zastosowania innych języków czy znaków specjalnych w tabelach lub nazwach kolumn. Aby to uzyskać, ustawiamy w pliku *sasv9_usermods.cfg* (lokalizacja: *SAS-configuration-directory\1\server-context\9_usermods.cfg*) następujące opcje dla SAS Workspace Server:

- ENCODING UTF8
- VALIDVARNAME ANY
- VALIDMEMNAME EXTEND

Pierwszy punkt jest odpowiedzialny za redukcję błędów, które wynikają z pracy nad plikiem w jednym języku, na komputerze operującym innym językiem. Opcja VALIDVARNAME związana jest ze spacjami i znakami specjalnymi. Ostatnia z nich zwiększa ilość dozwolonych znaków w tabelach.

Pozostałe kroki poinstalacyjne znajdują się pod [**31**].

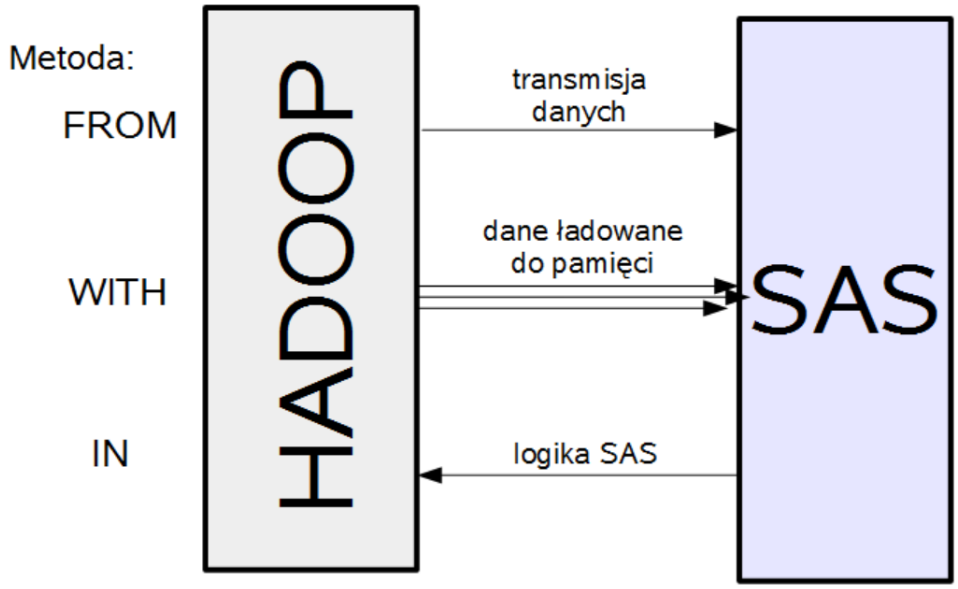
2. Sposoby złączenia SAS i Hadoop

Istnieją trzy metody łączenia pracy w środowisku SAS oraz platformie programistycznej Hadoop (patrz: rysunek 4.1):

(1) wyciąganie danych z Hadoopa do SAS (from);
(2) współpraca (with);
(3) SAS pracuje bezpośrednio w Hadoopie (in).

Omówmy po krótce te punkty oraz ich związek z tym, co już się nauczyliśmy.

Pierwsza z możliwości to traktowanie Hadoopa jako kolejne źródło danych. Są one pobierane do SAS, przetwarzane, a następnie zapisuje się wyniki z powrotem. Niestety skutkuje to słabą wydajnością, ale zalecane jest gdy proces nie uruchamia się w Hadoopie, bądź nie wszystkie dane się w nim znajdują.



RYSUNEK 4.1. Schemat prezentujący trzy metody współpracy SAS oraz Hadoop.

Współpracujące podejście pozwala stworzyć współdzielone środowisko. Jednocześnie umożliwa niezależność w skalowaniu obu środowisk - przechowywania danych oraz środowiska analitycznego. Wielkości obu klastrów nie muszą być takie same.

Z tym punktem związany jest LASR Analytic Server. Stanowi on serwer obliczeniowy in-memory, stworzony na potrzeby zaawansowanej analityki. Pojawiające się tutaj angielskie wyrażenie in-memory odnosi się do szybkości dostępu do danych. W tym przypadku występują obliczenia równoległe oraz rozproszone przetwarzanie na klastrze. Powinien on działać w zgodzie z multi_user - to znaczy pozwalać na dużą ilość użytkowników.

Gdy wszystkie dane znajdują się w Hadoopie i jest to nasze docelowe miejsce przetwarzania, warto skorzystać z trzeciej ewentualności. Tu pojawia się SAS Embedded Process, który daje możliwość skalowalnych obliczeń SAS w Hadoopie.

Związmy to wszystko z zamkniętym cyklem analitycznym obróbki danych:

- zarządzanie danymi;
- eksploracja danych;
- badanie i rozwijanie modeli;
- wdrażanie.

Trzymając się powyższej kolejności, zaczynamy od analitycznego przygotowania danych przy pomocy metody in. Po przeniesieniu danych z Hadoopa do środowiska SAS (from) przechodzimy do eksploracji danych. Zaznajamiamy się z

nimi korzystając ze środowiska in-memory, a następnie budujemy modele (with). Proste wdrożenie tych modeli dokonujemy jak najszybciej w Hadoopie (in).

SAS Data Loader pomaga w pracy głównie na początku tego cyklu. Jest to narzędzie dobre dla tych, którzy chcą przetworzyć dane lub je oczyścić.

ROZDZIAŁ 5

Badanie zbioru danych

INFORMACJA 4. *Doszliśmy do clou tekstu. W dalszej części zajmiemy się omawianą bazą danych przy pomocy technik Big Data, zaczynając od podstawowego oczyszczania danych. Oznacza to, że spojrzymy na zdobyte informacje przy pomocy narzędzi dostępnych na klastrze (notatnik Zeppelin) oraz w środowisku SAS.*

Wykorzystajmy notatnik Zeppelin do sprawdzenia wersji Sparka. Posłuży do tego komenda

```
%spark2
spark.version
```

Uzyskujemy wówczas:

```
res1: String = 2.1.0.2.6.0.3-8
```

Przejdźmy do kilku poleceń w powłodze systemowej. Musimy usunąć stare zbiory danych, posiadające takie same nazwy co te, na których będziemy pracować. Uzyskamy to przy pomocy:

```
%sh
if [ -e /tmp/green_tripdata_2016-01.csv ]
then
    rm -f /tmp/green_tripdata_2016-01.csv
fi
```

Podobnież robimy dla pozostałych plików. Po oczyszczeniu folderu */tmp* możemy umieścić w nim uprzednio pobrane zbiory danych:

```
%sh
wget https://s3.amazonaws.com/nyc-tlc/trip+data/green_tripdata_2016-01.csv
hadoop fs -put green_tripdata_2016-01.csv /tmp
```

Wpisując następnie

```
%spark2.spark
val path = "/tmp/green_tripdata_2016-01.csv"
val zbior = spark.read.csv(path)
zbior.printSchema()
zbior.show(10)
```

zobaczymy jakie pola wchodzą w skład tabeli (*zbior.printSchema*) oraz 10 pierwszych jej wierszy (*zbior.show(10)*).

Na potrzeby pracy zdecydowaliśmy, że język programowania Python doskonale wychodzi naprzeciw naszym zapotrzebowaniom. Jedna z jego bibliotek - Pandas - udostępnia szereg funkcji do analizy danych (szczegóły i tutoriale w referencji **[16]**). Aby z niej korzystać, należy poprzedzać kody wyrażeniem *import pandas as pd.*

Kolejną wartą uwagi biblioteką Pythona jest Seaborn, oparta o Matplotlib (więcej szczegółów w **[17]**). Aby wyrysowywać przy pomocy tych bibliotek wykresy w Zeppelinie, musimy dodać do kodu *plt.switch_ backend('agg').*

Uwaga 1. *Dla lepszej czytelności kodów przyjmijmy, że w dalszej części tekstu rezygnujemy z zapisu wspomnianych początkowych instrukcji, to jest:*

```
%pyspark
import matplotlib.pyplot as plt; plt.rcdefaults()
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
import seaborn as sns
import pandas as pd
plt.switch_backend('agg')
```

Dojście do danego pliku (na przykład styczniowego) uzyskujemy przy pomocy *styczen = pd.read_ csv("green_ tripdata_ 2016-01.csv").* Zsumowanie wszystkich miesięcy w jeden zbiór powstaje na skutek komend typu *styluty = styczen.append(luty, ignore_ index=True)*, gdzie luty odnosi się do pliku *green_ tripdata_ 2016-02.csv*, zaś *ignore_ index=True* daje nam możliwość zastąpienia dwóch oddzielnych indeksów jednym wspólnym.

Uwaga 2. *Ze względu na wiele nieprawidłowości w zbiorach odnoszących się do drugiego półrocza (na przykład kompletnie puste różne pola) będziemy w dalszej części tekstu pracować nad plikiem taxi, odnoszącym się do miesięcy styczeń - czerwiec.*

Pierwsze wiersze zbioru danych (rys. 5.1) pokaże polecenie *taxi.head()* - w nawiasie możemy przy tym określić ile ich nas interesuje. Występujące pola wypisze komenda *taxi.columns*, zaś ich podstawową statystykę (patrz rys. 5.2), zobaczymy po wpisaniu *taxi.describe().* Do liczby wierszy (7 481 370) dostaniemy się komendą *taxi.index.*

1. Czyszczenie danych

Przejdźmy do oczyszczania zbioru danych. Zacznijmy od odszukania skorelowań. Przykładowo - bierzemy pod uwagę pola opisujące pewną sumę, możliwą do

	VendorID	lpep_pickup_datetime	Lpep_dropoff_datetime	Store_and_fwd_flag	\
0	2	2016-01-01 00:29:24	2016-01-01 00:39:36	N	
1	2	2016-01-01 00:19:39	2016-01-01 00:39:18	N	
2	2	2016-01-01 00:19:33	2016-01-01 00:39:48	N	
3	2	2016-01-01 00:22:12	2016-01-01 00:38:32	N	
4	2	2016-01-01 00:24:01	2016-01-01 00:39:22	N	

	RateCodeID	Pickup_longitude	Pickup_latitude	Dropoff_longitude	\
0	1	-73.928642	40.680611	-73.924278	
1	1	-73.952675	40.723175	-73.923920	
2	1	-73.971611	40.676105	-74.013161	
3	1	-73.989502	40.669579	-74.000648	
4	1	-73.964729	40.682854	-73.940720	

RYSUNEK 5.1. Fragment wyniku taxi.head(5) - 8 pierwszych pól.

	VendorID	RateCodeID	Pickup_longitude	Pickup_latitude	\
count	7.481371e+06	7.481371e+06	7.481371e+06	7.481371e+06	
mean	1.787652e+00	1.099367e+00	-7.381668e+01	4.067981e+01	
std	4.089697e-01	9.610320e-01	3.016695e+00	1.656081e+00	
min	1.000000e+00	1.000000e+00	-1.152825e+02	0.000000e+00	
25%	2.000000e+00	1.000000e+00	-7.396084e+01	4.069439e+01	
50%	2.000000e+00	1.000000e+00	-7.394646e+01	4.074619e+01	
75%	2.000000e+00	1.000000e+00	-7.391884e+01	4.080203e+01	
max	2.000000e+00	9.900000e+01	0.000000e+00	4.316801e+01	

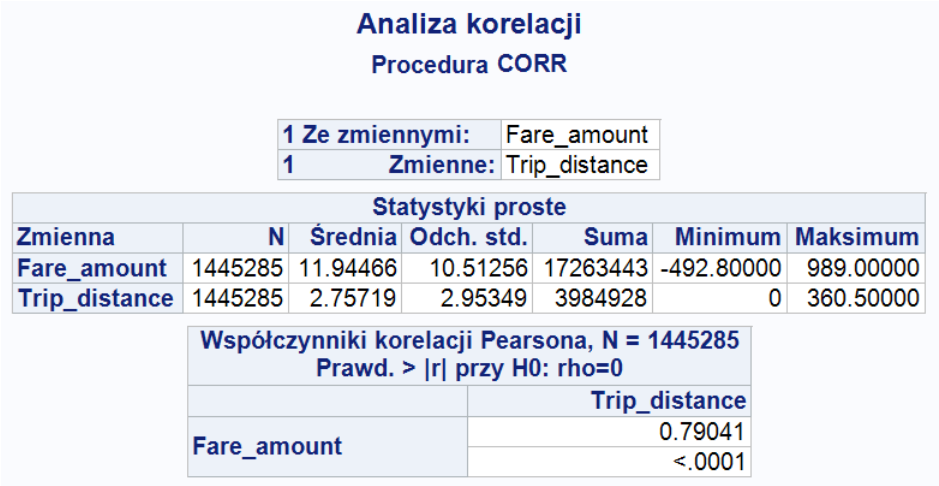
	Dropoff_longitude	Dropoff_latitude	Passenger_count	Trip_distance	\
count	7.481371e+06	7.481371e+06	7.481371e+06	7.481371e+06	
mean	-7.382858e+01	4.068531e+01	1.356821e+00	2.799212e+00	
std	2.849633e+00	1.563652e+00	1.023290e+00	2.931521e+00	
min	-1.153322e+02	0.000000e+00	0.000000e+00	0.000000e+00	
25%	-7.396806e+01	4.069573e+01	1.000000e+00	1.030000e+00	
50%	-7.394543e+01	4.074653e+01	1.000000e+00	1.840000e+00	
75%	-7.391219e+01	4.079083e+01	1.000000e+00	3.510000e+00	
max	0.000000e+00	4.293814e+01	9.000000e+00	8.322000e+02	

RYSUNEK 5.2. Fragment wyniku taxi.describe() - 8 pierwszych pól.

wyliczenia w razie potrzeby. W naszym zbiorze występuje kilka takich przypadków. Opisaliśmy je poniżej, wraz z wytłumaczeniem ich wyboru:

- *Fare_ amount* - wyliczenie z taryfy i dystansu, podane przez licznik;
- *Tolls_ amount* - jest to łączna kwota opłat, które możemy sami (poprzez komputer) zsumować;
- *Total_ amount* - kolejna suma, dająca kwotę końcowej płatności.

Na dodatkowe potwierdzenie powyższego, można sprawdzić jak silnie *Fare_ amount* jest związane z *Trip_ distance* (pozostałe punkty analogicznie). Wykorzystać do tego można na przykład styczniowy zbiór danych (jako obrazowy, w pozostałych stwierdzamy podobne zależności) i kreator korelacji w SAS EG. W nim umieszczamy pierwsze ze wspomnianych pól w "koreluj z", zaś drugi - "zmienne analizowane". Następnie zaznaczamy korelację Pearson i uruchamiamy. Uzyskane wyniki zostały zaprezentowane na rysunku 5.3. Widzimy na nim, że współczynnik korelacji



Analiza korelacji

Procedura CORR

1 Ze zmiennymi:	Fare_amount
1 Zmienne:	Trip_distance

Statystyki proste

Zmienna	N	Średnia	Odch. std.	Suma	Minimum	Maksimum
Fare_amount	1445285	11.94466	10.51256	17263443	-492.80000	989.00000
Trip_distance	1445285	2.75719	2.95349	3984928	0	360.50000

Współczynniki korelacji Pearsona, N = 1445285
Prawd. > |r| przy H0: rho=0

	Trip_distance
Fare_amount	0.79041
	<.0001

RYSUNEK 5.3. Zrzut ekranu z SAS EG - analiza korelacji pomiędzy *Trip_ distance* oraz *Fare_ amount*

Pearsona jest na poziomie 0.79041. Oznacza to, że stwierdzono wysoką zależność między badanymi polami.

Współczynnik korelacji Pearsona jest wyliczany ze wzoru

$$r(X,Y) = \frac{cov(X,Y)}{\sigma_X * \sigma_Y}. \tag{1}$$

Tym samym musimy wpierw uzyskać kowariancję pomiędzy badanymi zmiennymi:

$$cov(X,Y) = E(X \cdot Y) - EX \cdot EY = \frac{1}{n}\sum_{i=1}^{n} X_i Y_i - sr(X)sr(Y),$$

przy czym E to wartość oczekiwana, a $sr(X)$ to średnia zbioru X. Następnie wynik podzielić przez iloczyn odchyleń standardowych dla obu zmiennych. Odchylenie standardowe (ang. standard deviation) opisuje równość:

$$\sigma(X) = \sqrt{\frac{\sum_{i=1}^{n}(x_i - sr(X))^2}{n}}, \tag{2}$$

gdzie n to liczba kursów, x_i - dany kurs, $sr(X)$ - średnia rozpatrywanego zbioru. Tym samym wzór (1) możemy zastąpić następującym:

$$r(X,Y) = \frac{\sum X_i Y_i - sr(X)sr(Y)}{\sqrt{\sum_{i=1}^{n}(x_i - sr(X))^2}\sqrt{\sum_{i=1}^{n}(y_i - sr(Y))^2}}.$$

UWAGA 3. *Wzór (2) na odchylenie standardowe jest właściwy dla populacji. Przybliżenie odchyleniem standardowym z próby jest związane z zamieniem w mianowniku n na $n-1$.*

W celu sprawdzenia czy wyliczony współczynnik korelacji jest istotny, przeprowadza się test o hipotezach:

- H_0: $r = 0$ - zależność nieistotna;
- H_1: $r \neq 0$ - zależność istotna.

Weryfikacja hipotezy odbywa się przy pomocy testu t:

$$t = \frac{r(X,Y)}{\sqrt{1 - r(X,Y)^2}} \cdot \sqrt{n-2}$$

Jeżeli wartość istotności p, uzyskana z testu t, jest większa od poziomu istotności p^* ($p > p^*$, $p^* = 0.05$) to przyjmujemy H_0. W sytuacji przeciwnej (na przykład tej z rysunku 5.3) wybieramy H_1 i stwierdzamy skorelowanie.

Na ogół przyjmuje się, że wartości współczynnika korelacji Pearsona powyżej 0.5 ukazują już pewien stopień korelacji. Im jest on większy, tym zależność także bardziej zauważalna. Trzeba jednakże uważać - to że w jednej sytuacji współczynnik ten jest wystarczająco duży nie oznacza, że podobnie będzie w drugiej (na przykład w problemach fizycznych).

W miejscu tym chcielibyśmy podkreślić, że wartości współczynnika korelacji Pearsona są zaniżane przez obserwacje odstające. Po wyeliminowaniu ich (o czym w dalszej części pracy) stwierdzana jest większa zależność pomiędzy badanymi polami.

Innym sposobem sprawdzenia współzależności dwóch pól jest wyrysowanie wykresu rozrzutu. Wówczas jedna oś odpowiada wynikom jednej zmiennej, a druga - drugiej. Im bardziej uzyskane tak punkty zbiegają do jednej prostej, tym współczynnik korelacji jest bliższy 1, a tym samym wskazuje skorelowanie. Prosta ta to linia regresji, o równaniu $y = b_0 + b_1 x$, gdzie b_0 oraz b_1 to współczynniki regresji.

Niestety SAS EG nie pozwolił wytworzyć wykresu rozrzutu dla tak dużych zbiorów danych jak badane. Ukazuje to potrzebę szukania narzędzia, który potrafiłby to zrobić nie tylko na grupie informacji z jednego miesiąca, ale i z całego roku.

Warto wyeliminować te elementy zbioru, które na pewno będą zbyteczne w głównej części pracy, a jedynie wpływają na wielkość pliku. W naszej bazie należą do takich te pola, które dają informacje o podatkach i dodatkowych opłatach:

- *Extra*;
- *MTA_tax*;
- *Improvement_surcharge*.

Są to kwoty, które w każdej chwili możemy ponownie dopisać do tabeli (przy znajomości amerykańskiego prawa podatkowego odnośnie do taksówek, badź sposobie dopisu opłat).

Przejdźmy do języka Python. Przy jego pomocy możemy jeszcze raz, tym razem dokładniej, przyjrzeć się kwestii współzależności pól. Korzystając z biblioteki Seaborn i kodu

```
plt.figure(figsize=(40, 40))
sns.heatmap(taxi.corr(), annot = True)
show(plt)
```

uzyskaliśmy tabelę korelacji, która pozwala w szybki sposób zauważyć zależność pomiędzy polami. Wynika to z faktu zastosowania kolorystycznej skali na macierzy - im kolor jest bliższy czerni/bieli tym większa występuje korelacja (ujemna/dodatnia). Znak pokazuje przy tym jak układają się na wykresie punkty. Jeżeli funkcja liniowa, wokół której się skupiają, jest malejąca, wówczas pojawia się minus. W przeciwnym wypadku mamy korelację dodatnią.

Niestety ze względu na dużą wielkość i szczegółowość uzyskanej macierzy korelacji, nie jesteśmy w stanie zamieścić tutaj jej obrazu bez utraty czytelności. Zamiast tego wypiszemy, że wysoki bądź znaczny stopień korelacji pojawia się przy polach:

(1) Pickup_longitude oraz Pickup_latitude: -0.99;
(2) Dropoff_longitude oraz Dropoff_latitude: -0.99;
(3) MTA_tax oraz Trip_tipe: -0.84;
(4) improvement_surcharge oraz Trip_Type: -0.84;
(5) MTA_tax oraz RateCodeID: -0.55;
(6) improvement_surcharge oraz RateCodeID: -0.55;
(7) Tip_amount oraz Total_amount: 0.55;
(8) Fare_amount oraz Trip_distance: 0.82;
(9) Total_amount oraz Trip_distance: 0.82;
(10) Trip_Type oraz RateCodeID: 0.96;
(11) MTA_tax oraz improvement_surcharge: 0.96;
(12) Total_amount oraz Fare_amount: 0.97;

Tym samym potwierdziliśmy zauważoną w środowisku SAS zależność pomiędzy polami *Fare_amount* oraz *Trip_distance*. Różnica w wartości bierze się z analizy innych zbiorów (wcześniej sprawdzaliśmy sam styczeń). Pozostałe punkty odnoszą się zaś do przynajmniej jednej z kolumn przygotowanych do eliminacji. Dokonamy jej kodem *taxi.drop('nazwa_pola', axis=1, inplace=True)*.

Nasze spostrzeżenia potwierdziłoby wykonanie macierzy wszystkich wykresów, pomiędzy każdymi polami. Możemy dokonać tego poleceniem *sns.pairplot(taxi)*. Niestety wstawienie tutaj wyniku wiązałoby się ponownie z małą czytelnością rysunku.

Macierz korelacji uzyskamy także w środowisku SAS za pomocą kodu:

```
proc corr data=nazwa_zbioru
outp=corr (where=(_type_='CORR')) print;
run;
proc print data=corr (drop=_type_) noobs;
```

The SAS System

NAME	VendorID	lpep_pickup_datetime	Lpep_dropoff_datetime	RateCodeID	Pickup_longitude	Pickup_latitude	Dropoff_longitude	Dropoff_latitude	Passenger_count
VendorID	1.00000	-0.00091	-0.00056	-0.00586	-0.02036	0.02251	-0.03490	0.03689	0.08422
lpep_pickup_datetime	-0.00091	1.00000	0.99997	-0.00990	-0.00203	0.00071	-0.00111	-0.00028	-0.00468
Lpep_dropoff_datetime	-0.00056	0.99997	1.00000	-0.00987	-0.00207	0.00074	-0.00114	-0.00026	-0.00462
RateCodeID	-0.00586	-0.00990	-0.00987	1.00000	0.07864	-0.07414	0.01749	-0.01268	-0.00524
Pickup_longitude	-0.02036	-0.00203	-0.00207	0.07864	1.00000	-0.99321	0.33252	-0.32266	-0.00514
Pickup_latitude	0.02251	0.00071	0.00074	-0.07414	-0.99321	1.00000	-0.32307	0.32705	0.00442
Dropoff_longitude	-0.03490	-0.00111	-0.00114	0.01749	0.33252	-0.32307	1.00000	-0.99267	-0.00526
Dropoff_latitude	0.03689	-0.00028	-0.00026	-0.01268	-0.32266	0.32705	-0.99267	1.00000	0.00465
Passenger_count	0.08422	-0.00468	-0.00462	-0.00524	-0.00514	0.00442	-0.00526	0.00465	1.00000
Trip_distance	-0.00465	-0.01593	-0.01530	0.04071	-0.02167	0.01975	-0.00929	0.00708	0.00801
Fare_amount	0.00437	0.00354	0.00413	0.15036	-0.00064	-0.00172	-0.00062	-0.00208	0.01172
Extra	-0.00090	0.02906	0.02907	-0.14288	-0.01751	0.01545	-0.00984	0.00805	0.02106
MTA_tax	-0.00703	0.00972	0.00972	-0.82339	-0.09495	0.08760	-0.03483	0.02715	0.00354
Tip_amount	-0.01649	0.01437	0.01449	0.00807	0.00156	-0.00614	-0.00833	0.00343	0.00424
Tolls_amount	-0.00089	-0.00136	-0.00127	0.02567	-0.00269	0.00356	0.00086	-0.00063	0.00285
improvement_surcharge	0.00002	0.00907	0.00908	-0.80640	-0.09372	0.08708	-0.03736	0.03027	0.00460
Total_amount	-0.00016	0.00730	0.00786	0.12196	-0.00212	-0.00107	-0.00311	-0.00051	0.01222

RYSUNEK 5.4. Fragment tabeli korelacji z środowiska SAS

Uruchomienie go na pliku styczniowym da nam tabelę, której część widzimy na rysunku 5.4. Ponowne pojawienie się wysokich wartości współczynników korelacji przy konkretnych polach potwierdza wcześniejsze spostrzeżenia.

UWAGA 4. *Usunięte zostały pola: MTA_tax, Extra, improvement_surcharge, Tolls_amount oraz Fare_amount. Pozostałe Total_amount pozostawiliśmy dla wygody.*

W innej sytuacji kolumna Total_amount powinna zostać zastąpiona wyliczeniem:

$$Total\_amount = Fare\_amount + Extra + MTA\_tax + \\ + Tip\_amount + improvement\_surcharge + Tolls\_amount.$$

Ze względu jednakże na usunięcie opłat startowych, można stworzyć pomocniczą sumę:

$$Start\_amount = Extra + MTA\_tax + improvement\_surcharge + Tolls\_amount.$$

Pozycję Fare_amount moglibyśmy zaś uzyskać poprzez pomnożenie liczby przebytych mil przez kwotę zastosowanej stawki. Jest to jednakże problematyczne - stawka wynegocjonowana jest trudna do przewidzenia, każdy kurs z nią związany odbył się na osobnych zasadach. Pokazuje to współczynnik korelacji Pearsona pomiędzy polem Fare_amount, a Trip_distance, który uzyskaliśmy przy pomocy SAS Miner (o którym w dalszej części tekstu). Dla RateCodeID=5 ma on wartość poniżej 0.2, czyli nie wychwytujemy korelacji. Przypadek ten zaniża ogólną współzależność pomiędzy wspomnianymi kolumnami - współczynnik Pearsona dla wszystkich przypadków to 0.82. Po oczyszczeniu danych wzrasta do ponad 0.93, mamy tu powód czemu nie więcej. Tym samym nie stworzymy prawidłowego wzoru na Fare_amount nie tylko dla stawki wynegocjonowanej, ale także i dla całego zbioru. Mamy jedynie do dyspozycji jego przybliżenie, czyli linię regresji o współczynnikach regresji 3.836 oraz 2.887 (uzyskane przy pomocy węzła Regresja, SAS Miner).

Oznacza to, że w przypadku, gdybyśmy zrezygnowali z Total_ amount, musielibyśmy pozostawić Fare_ amount oraz jeszcze dodać sumę pomiędzy kwotami początkowymi.

Tabela korelacji pozwala przypuszczać, że istnieje pewna nieprawidłowość przy polu *Ehail_fee*. Zarówno w macierzy ze środowiska SAS, jak i w tej utworzonej z wykorzystaniem Pysparka, zauważyliśmy puste okna przy tej kolumnie. Sprawdźmy które pola mają wartości puste. Na skutek *taxi.apply(lambda x: sum(x. isnull()),axis=0)* dowiemy się iż wspomniana kolumna pozostała w całości niewypełniona (liczba wartości pustych pokrywa się z liczbą wierszy). Potwierdza to polecenie *taxi['Ehail_fee'].max()*, które daje wartość maksymalną *nan*. Oznacza to, że pole te powinno zostać wyeliminowane z analizy.

Podany wcześniej kod na wartości puste informuje dodatkowo o pojedynczych niewpisanych wartościach *Tolls_ amount*, *Total_ amount* oraz *Payment_ type*. Powinniśmy je wyeliminować, przykładowo wpisując w ich miejsce najczęściej występującą wartość. Jeżeli jednakże sprawdzimy wpierw problematyczny wiersz na przykład dla *Tolls_ amount* (*taxi[taxi.Tolls_ amount.isnull()]*) okaże się, że wszystkie trzy błędy są przy tej samej pozycji. Jeśli wiersz ma aż trzy wartości zerowe lepiej go usunąć, aby wyeliminować możliwość błędów przy pozostałych polach. Poza tym wykazano 443 braków przy *Trip_ Type*, z którymi także trzeba się rozprawić.

Należy sprawdzić czy zbiór został wypełniony zgodnie z założeniami, opisanymi na stronach 13-14. Stosując komendę *taxi['RateCodeID'].unique()* zobaczymy, że pole RateCodeID zawiera wartości: $1, 5, 2, 4, 3, 6, 99$. Jak pamiętamy, są to kody stawki (standardowa, porty lotnicze, grupowa, wynegocjonowana, Nassau/Westchester), wśród których nie wymieniono liczby 99. Najczęściej używaną (z dużą przewagą) jest kod 1. Możemy więc zastąpić problematyczną liczbę odnośnikiem do standardowej stawki. Warto jednak wpierw sprawdzić jak wyglądają oba wspomniane wiersze, na przykład komendą *taxi[taxi.RateCodeID>7]*. Zobaczymy wówczas ciekawą zależność - źle wypełnione pole *RateCodeID* pokrywa się ze wspomnianymi w poprzednim akapicie brakami w *Trip_ Type* (patrz: rysunek 5.5, same kłopotliwe kolumny pojawią się dzięki *taxi[taxi.RateCodeID>7].loc[:,['RateCode ID', 'Trip_ type']]*). Do tego wpisując *taxi['RateCodeID'].value_ counts()* dowiemy się, że wartość 99 wystąpiła tyle samo razy co braki przy *Trip_ Type*. Oznacza to, że ponownie należy usunąć wiersze.

Zaskakującą obserwację dokonamy na skutek *taxi['Payment_ type'].unique()*. Okazuje się bowiem, że brakuje wartości 6, która odpowiada za unieważnione kursy. Istnieją dwie potencjalne powody takiego zjawiska:

- informacja o polu *Payment_ type* została zmieniona po czerwcu 2016 roku i wyglądała inaczej w tamtym czasie;

```
        VendorID lpep_pickup_datetime Lpep_dropoff_datetime  \
230181         1  2016-01-05 18:39:10   2016-01-05 20:39:10
794212         1  2016-01-17 11:30:46   2016-01-17 15:30:46
       Store_and_fwd_flag  RateCodeID  Pickup_longitude  Pickup_latitude  \
230181                  Y          99        -73.952454        40.685215
794212                  Y          99        -73.973030        40.683105
        Dropoff_longitude  Dropoff_latitude  Passenger_count     ...      \
230181                0.0               0.0                0     ...
794212                0.0               0.0                0     ...
        Fare_amount  Extra  MTA_tax  Tip_amount  Tolls_amount  Ehail_fee  \
230181         0.00    0.0      0.0         0.0           0.0        NaN
794212        20.15    0.0      0.0         0.0           0.0        NaN
        improvement_surcharge  Total_amount  Payment_type  Trip_type
230181                    0.0          0.00             1         NaN
794212                    0.0         20.15             1         NaN
[2 rows x 21 columns]
```

RYSUNEK 5.5. Przykład błędnie wypełnionych wierszy, wraz z zaznaczeniem pól, które na to nakierowały.

- rozpiskę kodów niewłaściwie dostarczono taksówkarzom;
- kierowcy "oszukiwali" i z jakiegoś powodu (na przykład premii) nie zaznaczali unieważnienia.

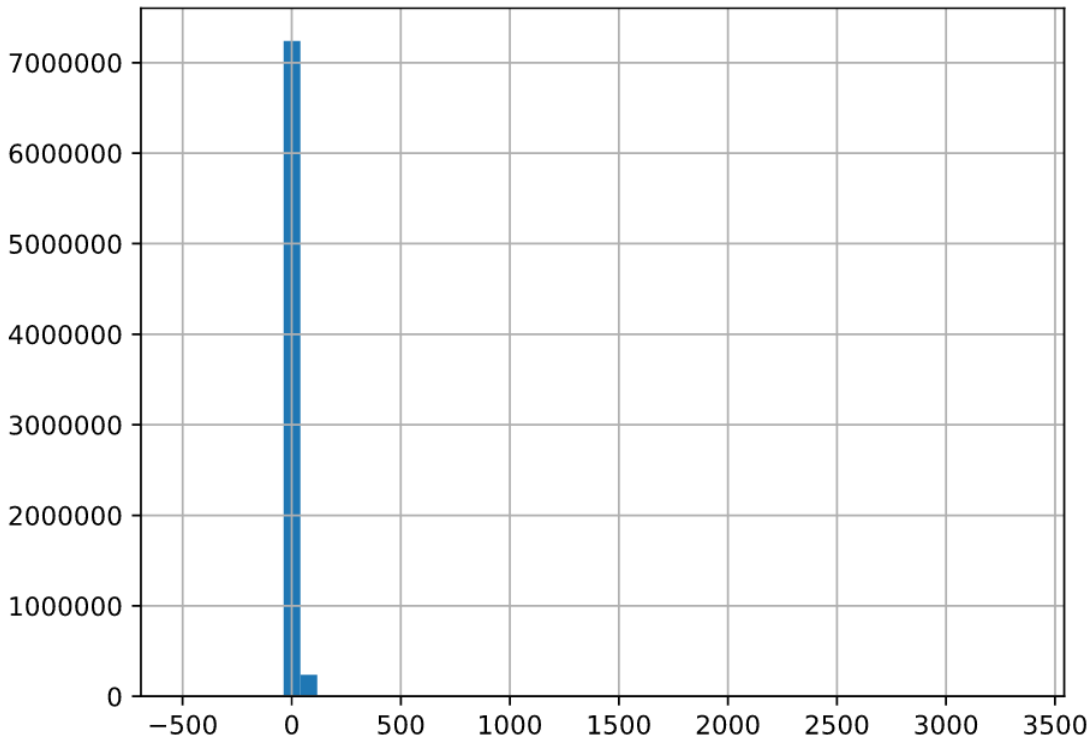
Korporacja powinna sprawdzić tę sprawę.

Bazując na podobnym kodzie odkryjemy liczbę 0 przy *Passenger_count*. Jak pamiętamy, pole te jest wypełniane przez kierowcę, więc zapewne wynika to z niewpisania odpowiedniej cyfry. Okazuje się, że takich przypadków jest dość dużo. Warto je pozostawić, aby sprawdzić w jakich sytuacjach do nich dochodzi. Przy innych sytuacjach wykorzystania tego pola pamiętajmy o zerowej kwestii.

Dobrym sposobem zauważenia punktów oddalonych jest skorzystanie z graficznego przedstawienia rozkładu empirycznego cechy. Rozkład ten to opis wartości przyjmowanych przez badaną cechę statystyczną, ukazujący częstotliwość jej występowania w danej próbce zbioru danych. Uzyskamy go między innymi przy pomocy:

- histogramu;
- wykresu pudełkowego.

Sprawdźmy pierwszy wykres na polu *Total_amount*. Dzięki *taxi['Total_amount'].hist(bins=50)* (patrz: rys 5.6) zauważymy, że większość osób zapłaciło mniej niż 100$ za kurs. Niestety szeroki zakres na osi odciętych tworzy przypuszczenie, że trafiały się także kwoty powyżej 300$. Ze względu na skalę osi rzędnych nie jesteśmy w stanie wyczytać z wykresu ile osób odbyło tak drogie podróże. Dowiemy się tego przy pomocy *taxi[taxi.Total_amount>300].count()*. Jak się okazuje, są to jedynie 422 pozycje, co stanowi bardzo małą część prawie 7.5 mln kursów. Potraktujmy więc wiersze opisujące te przypadki za punkty oddalone i usuńmy je ze zbioru.



RYSUNEK 5.6. Histogram uzyskany na polu Total_amount

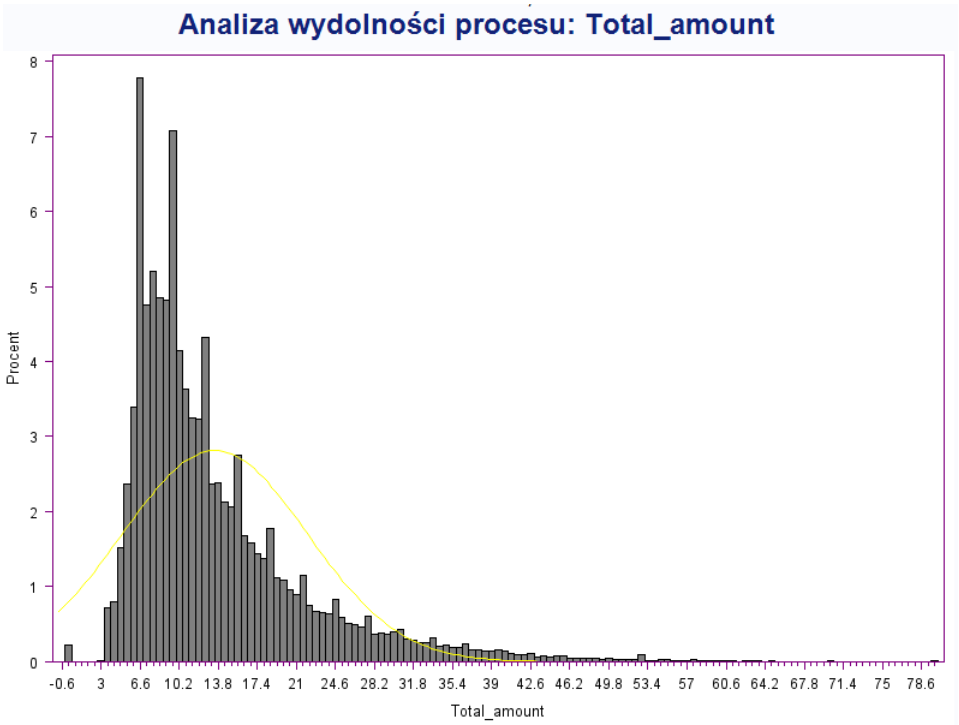
Dokładniej powyższy przypadek prezentuje histogram, uzyskany w środowisku SAS (rysunek 5.7). Przy jego pomocy widzimy, że założenie o usunięciu wszystkich punktów powyżej 300$ jest za małe. Jednocześnie występuje dużo przypadków z zerowym dystansem, po których na wykresie pojawia się ciekawy spadek. Możemy więc ponownie zastosować eliminację wierszy, tym razem z uwzględnieniem wszystkich, których wartość pola *Total_ amount* nie mieści się w przedziale (3, 53).

Mała uwaga - porty lotnicze (obie stawki) są związane z dłuższymi trasami. Aby nie zatracić tych przypadków, zwiększyliśmy możliwy zakres wartości pola *Total_ amount* do $(3, 120)$.

Kolejnym potencjalnym błędem jest źle wypełniona lokalizacja początkowa lub końcowa. Najprawdopodobniej wychodzi ona wówczas znacznie poza zakres miasta (czyli znowu sprawa punktów oddalonych). Przyjmijmy, że będą one mieściły się w przedziałach:

- dla szerokości: 40.65 i 40.85;
- dla długości: -74.00 i 73.85.

Założenie to może być przy tym wynikiem wyboru - na przykład poprzez spojrzenie na mapę i wybranie interesujących nas rejonów, jak było w naszym przypadku. Często tego typu czynności związane są z pewną analizą interesujących nas pól. Sposobem eliminującym całkowicie punkty oddalone zdaje się być określenie rozstępu



RYSUNEK 5.7. Histogram uzyskany na polu Total_amount w środowisku SAS

ćwiartkowego (ang. interquartile range, IRQ), związanego z kwartylami. Wyróżnia się trzy kwartyle:

- Q_1, kwartyl pierwszy (dolny) - dzieli zbiór na dwie części w taki sposób, że 25% obserwacji ma wartości mniejsze lub równe Q_1, a 75% większe lub równe;
- Q_2, kwartyl drugi (mediana, wartość środkowa) - połowa wyników znajduje się poniżej tejże liczby, a druga - powyżej. Stanowi on przykład miary środka, podobnie jak średnia oraz moda (najczęściej występująca wartość);
- Q_3, kwartyl trzeci (górny) - granica określająca, że 25% obserwacji jest powyżej niej, albo jest jej równa.

Sposób wyliczenia kwartyli zależy od tego, czy mamy do zbadania szereg szczegółowy, czy szereg rozdzielczy. Pierwszy przypadek jest związany z wypisaniem wszystkich obserwacji. Gdy są one w pewien sposób pogrupowane, to mówimy o szeregu rozdzielczym (punktowym albo przedziałowym). Zaprezentujmy to na przykładzie podania ocen w pewnej klasie:

- szereg szczegółowy: 1, 1, 1, 2, 2, 3, 3, 3, 3, 4, 4, 5;
- szereg rozdzielczy (punktowy):

ocena	liczba wystąpień
1	3
2	2
3	4
4	2
5	1

Dla szeregów szczegółowych o n obserwacjach kwartyl drugi wyznaczany jest ze wzoru:

$$Q_2 = \begin{cases} x_{\frac{n+1}{2}} & \text{gdy } n \text{ jest nieparzyste;} \\ \frac{1}{2}(x_{\frac{n}{2}} + x_{\frac{n}{2}+1}) & \text{gdy } n \text{ jest parzyste.} \end{cases}$$

Kwartyl pierwszy i trzeci uzyskuje się analogicznie. Korzystając z wartości środkowej, dzielimy zbiór na dwie części, a następnie ponownie wyliczamy medianę dla wybranej połowy.

W szeregach rozdzielczych kwartyle wyznacza się zgodnie z równaniem

$$Q_n = x_{0m} + \frac{N_{Q_n} - \sum_{i=1}^{m-1} n_i}{n_m} h_m,$$

gdzie:

- Q_n - konkretny kwartyl;
- N_{Q_n} - jego pozycja;
- m - numer przedziału, w którym jest wybrany kwartyl;
- x_{0m} - dolna granica przedziału występowania danego kwartyla;
- $\sum_{i=1}^{m-1} n_i$ - suma liczebności przedziałów poprzedzających przedział kwartyla (liczebność skumulowana);
- n_m - liczebność przedziału z kwartylem;
- h_m - rozpiętość przedziału, w którym jest kwartyl.

Pozycje kwartyli podajemy zgodnie z poniższym ustaleniem:

- $N_{Q_1} = \frac{n}{4}$;
- $N_{Q_2} = \frac{n}{2}$;
- $N_{Q_3} = \frac{3n}{4}$.

Powróćmy do pojęcia rozstępu ćwiartkowego. Jest to różnica pomiędzy trzecim, a pierwszym kwartylem, czyli

$$IRQ = Q_3 - Q_1.$$

Tym samym rozstęp ćwiartkowy dla naszego zbioru to 50% kursów. Zakładamy, że dobrze określone punkty powinny mieścić się w przedziale $(Q_1 - 1.5IQR, Q_3 + 1.5IQR)$.

Do innych sposobów odnalezienia problematycznych wartości należą wspomniane histogramy i wykresy pudełkowe (*taxi.boxplot(column='nazwa_pola')*), czy

wykresy rozrzutu, określające położenie obserwacji jako punktów względem układu współrzędnych. Poza tym przypuszczamy, że punkty oddalone znajdują się w odległości przynajmniej trzech odchyleń standardowych od średniej. Wyliczając to w Zeppelinie trzeba pamiętać, że niektóre wersje nie potrafią tego zrobić przy pojawiających się wartościach pustych. Kolejną ważną kwestią jest to, że metoda ta nie jest polecana z powodu wpływania (często znacznie) poszukiwanych punktów na średnią. Z tego powodu mówi się, że średnia jest wrażliwa na punkty oddalone.

Sprawdźmy tym sposobem, czy nasze ustalenia graniczne są poprawne. Na oczyszczonym już zbiorze uruchomiliśmy węzeł Eksploracja Statystyk w SAS Miner (więcej o tym w dalszej części tekstu). Dla pola *Pickup_ latitude* uzyskaliśmy następujące wartości:

- średnia: 40.7529;
- odchylenie standardowe: 0.05325.

Tym samym nasze wartości powinny mieścić się w przedziale

$$(40.7529 - 3 \cdot 0.05325, 40.7529 + 3 \cdot 0.05325),$$

czyli w $(40.59315, 40.91265)$. Ustalenie granic na wartościach 40.65 oraz 40.85 jest więc poprawne dla tego pola.

Ponownie przetestujmy podane limity, tym razem z wykorzystaniem rozstępu ćwiartkowego oraz pola *Dropoff_ latitude*. Za pomocą środowiska SAS otrzymaliśmy następujące wartości:

- $Q_1 = 40.7003$;
- $Q_3 = 40.7901$.

Oznacza to, że rozstęp ćwiartkowy jest równy 0.0898, a obserwacje powinny zawierać się w przedziale

$$(Q_1 - 1.5IQR, Q_3 + 1.5IQR) = (40.7003 - 1.5 \cdot 0.0898, 40.7901 + 1.5 \cdot 0.0898),$$

co daje $(40.5656, 50.9248)$. Ponownie uzyskujemy potwierdzenie właściwie określonych granic wartości.

Wracając do kwestii oddalonych punktów na mapie, poniższy kod usunie nam wszystkie wartości lokalizacji je przekraczające:

```
longitude_limity = [-74.00, -73.85]
latitude_limity = [40.65, 40.85]
taxi = taxi[(taxi.Pickup_longitude.between(longitude_limity[0],
  longitude_limity[1], inclusive=False))]
taxi = taxi[(taxi.Dropoff_longitude.between(longitude_limity[0],
  longitude_limity[1], inclusive=False))]
taxi = taxi[(taxi.Pickup_latitude.between(latitude_limity[0],
```

```
  latitude_limity[1], inclusive=False))]
taxi = taxi[(taxi.Dropoff_latitude.between(latitude_limity[0],
  latitude_limity[1], inclusive=False))]
```

Podobny kod został zastosowany do wartości oddalonych pozostałych pól. Należy do nich między innymi kolumna *Total_ amount*, którą związaliśmy z przedziałem $[0, 20]$.

Należy upewnić się, że pola odnoszące się do dat są zapisane właściwym typem danych. Posłuży do tego polecenie konwertujące na datetime:

```
%pyspark
taxi["lpep_pickup_datetime"] = pd.to_datetime(taxi
  ["lpep_pickup_datetime"])
taxi["Lpep_dropoff_datetime"] = pd.to_datetime(taxi
  ["Lpep_dropoff_datetime"])
```

Często zdarza się, że wartości danego pola są od siebie mocno odległe, chociaż nie zauważamy punktów oddalonych. Taka sytuacja zaistnieje, jeśli zbierzemy prędkości wszystkich uczestników ruchu, a nie tylko taksówek. Różnica pomiędzy szybkością rowerzysty, a samochodu jest przecież znacząca. Aby wyeliminować takie sytuacje, warto dokonać normalizacji. Jedną z najczęściej stosowanych jest standaryzacja, którą przeprowadzamy zgodnie ze wzorem:

$$X^* = \frac{X - sr(X)}{OS(X)},$$

gdzie sr(X) to średnia danego zbioru, zaś OS(X) to odchylenie standardowe.

UWAGA 5. *Zaprezentowany proces oczyszczania danych nie przedstawia wszystkich możliwych dróg dojścia do błędów. Zapisaliśmy jedynie przykładowe sposoby, prezentujące często występujące problemy ze zbiorami danych.*

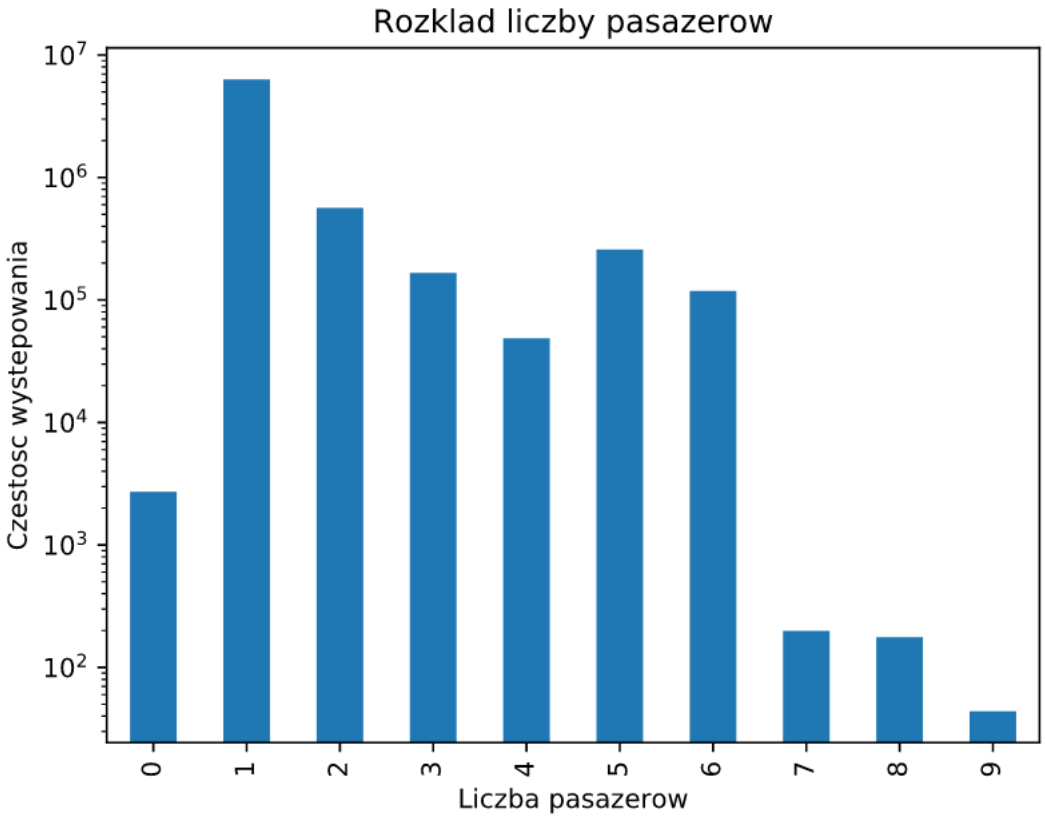
Liczba wierszy przed czyszczeniem: 7 481 372.

Liczba wierszy po oczyszczeniu: 2 575 334.

2. Analiza danych

Po oczyszczeniu zbioru danych można przejść do właściwej analizy. W naszym tekście zaprezentujemy ją wpierw przy pomocy narzędzi dostępnych na klastrze Hadoop, a następnie przejdziemy do środowiska SAS.

2.1. Klaster Hadoop. Na potrzeby pracy postawiliśmy kilka przykładowych pytań odnośnie badanego zbioru. Odpowiedzi zostały uzyskane przy pomocy Zeppelina oraz języka Python:



RYSUNEK 5.8. Wykres rozkładu pasażerów.

(1) Posiadanie konkretnej kategorii prawa jazdy oraz wielkość samochodu nakładają maksymalną liczbę pasażerów do przewożenia. Czy warto zainwestować w nowego kierowcę oraz busa?

Komenda *taxi['Passenger_ count'].max()* da nam informację o maksymalnej ilości pasażerów równej 9 osobom. Wpisanie jednakże *taxi['Passenger _ count'].value_ counts()* pokazuje, że zapotrzebowanie na większe przewozy jest za małe na takową inwestycję (w zbiorze początkowym 44 przypadki największej grupy). Jeszcze lepiej widzimy to na wykresie z rysunku 5.8, który powstał za pomocą następującego kodu:

```
pasazerowie = taxi['Passenger_count'].value_counts()
  .sort_index()
pasazerowie.plot(kind = 'bar',logy = True)
plt.xlabel('Liczba pasazerow')
plt.ylabel('Czestosc wystepowania')
plt.title('Rozklad liczby pasazerow')
show(plt)
```

Wykres ten został stworzony na początkowym zbiorze danych. Po oczyszczeniu go (patrz: wcześniejszy podrozdział) różnice są jeszcze większe:

- 1 osoba - 2172478;
- 2 osoby - 193315;
- 5 osób - 92210;
- 3 osoby - 58628;
- 6 osób - 44129;
- 4 osoby - 14559;
- 0 osób - 54;
- 7 osób - 7;
- 8 osób - 4;
- 9 osób - 3.

Lepiej w sytuacji większej liczby pasażerów po prostu podstawić dwa mniejsze samochody.

Często zdarza się, że w wyniku badania jednego pytania nasuwa się kolejne. W tym przypadku na wykresie 5.8 zauważamy znaczący procent kursów z zerową liczbą pasażerów. Oczyszczenie zbioru nie wyeliminowało całkowicie tego problemu, zostało bowiem 54 takich obserwacji. Należy upewnić się jakich sytuacji się one tyczą - czy taksówkarze zapisują w ten sposób powroty do swoich stref, czy są to błędy. Wówczas warto przeprowadzić analizę zbioru dla $Passenger\_count$=0. Dla pierwszego przypadku będziemy zastanawiali się jak długo kierowcy jadą bez pasażera, ile tracą na tym paliwa, które miejsca są obarczone dłuższą jazdą do kolejnego klienta. W sytuacji gdy są to błędy, sprawdzamy czy i kiedy nie było awarii systemu (częstsze przypadki wystąpień tej sytuacji).

Przed dalszą częścią pracy usunęliśmy wspomniane przypadki ze zbioru danych, aby źle wypełnione wiersze nie wpływały na wyniki końcowe.

(2) Zakładając, że firma taksówkarska chciałaby zmienić cennik usług, od ilu osób powinna być stawka grupowa?

Najczęściej występujące samochody osobowe umożliwiają zabranie 4 pasażerów (licząc środkowe siedzenie z tyłu). Więcej osób wymaga większego samochodu (dobre wytłumaczenie wyższej opłaty!), który potencjalnie spalałby więcej paliwa. Poza tym wydatki na stacjach benzynowych zwiększyłaby waga dodatkowych klientów. Z punktu widzenia korporacji dobrym pomysłem byłoby więc postawienie stawki grupowej od pięciu osób. Co na to mówią dane?

Na wykresie 5.8 zauważamy, że liczba pasażerów systematycznie spada, aż do prawdopodobnego zdobywcy środkowego miejsca. Widoczny wzrost częstości występowania pojawia się przy pięciu osobach, zaś szóstka nie jest aż tak odległa od trzech klientów. Sytuacja ta nie zmieniła się także i po

oczyszczeniu zbioru. Oznacza to, że istnieje dużo kursów, które wymagałyby dodatkowej opłaty.

Oczywiście na wynik końcowy ma także wpływ ilość pokonanych mil, które zostaną pomnożone przez stawkę grupową/podstawową. Sprawdzimy to na przykład poleceniem *taxi.groupby(taxi.Passenger_ count).Trip_ distance. mean()*, który ukaże średnią przebytą odległość dla danej liczby pasażerów:

- 1 - 2.788372;
- 2 - 2.877653;
- 3 - 2.991665;
- 4 - 2.923603;
- 5 - 2.766745;
- 6 - 2.843020;
- 7 - 1.158571;
- 8 - 0.817500;
- 9 - 1.203333.

Jak widzimy, pomiędzy jednym pasażerem, a sześcioma, występują stosunkowo podobne średnie odległości - mieszczące się w przedziale (2.7667, 2.9917). Najczęstsze kursy potencjalnie związane ze stawką grupową (pięć i sześć osób) nie wyróżniają się specjalnie na tle stawki podstawowej. Znacznie krótsze kursy przy większej liczbie pasażerów (7, 8, 9) nie wpływają na ostateczną decyzję - i tak będą musiały być w stawce grupowej. Poza tym stanowią niewielki procent wszystkich przypadków (odpowiednio: 7, 4 i 3 kursy).

Największy średni dystans zauważamy przy trzech pasażerach. Niestety trudno wytłumaczyć stawkę grupową dla rodziców z jednym dzieckiem, więc pozostańmy przy wcześniejszych postanowieniach.

(3) Na który port lotniczy - Newark czy JFK - powinniśmy wysyłać więcej taksówek?

Wypisanie *taxi['RateCodeID'].value_ counts()* pokaże, że ponad półtora raza więcej kursów (dokładniej: 1.5714) jest realizowanych z portu JFK:

- 2 (JFK) - 143;
- 3 (Newark) - 91.

Mała uwaga - w zbiorze, w którym

- pole *Tip_ amount* mieści się w zbiorze większym (do 300$, zamiast do 120$);
- kolumna *Trip_ distance* nie została ograniczona do 35 mil;

występuje jeszcze więcej przypadków opatrzonych stawką, odnoszącą się do portów lotniczych:

- 2 - 6483;

- 3 - 1901.

Wówczas proporcjonalność pomiędzy nimi stoi na poziomie 3,527. Ze względu na z reguły dłuższe podróże do lotniska, istnieje możliwość, że zatraciliśmy część cennych informacji, odnośnie tych przypadków. Oba porty są w stosunkowo tej samej odległości od centrum Nowego Yorku (zgodnie z Google Maps: 24,8km oraz 21,5km). Tak duże trasy skłoniły nas do badania rozszerzonego zbioru.

Powróćmy do analizy. Trzeba pamiętać, że nie tylko sama liczba zleceń generuje zyski. Co ciekawe, różnica w ostatecznych kwotach ukazuje ciekawe zjawisko. Okazuje się, że kwestia ta jest odmienna w stosunku do liczby przejazdów - to na porcie Newark zarobiono łącznie znacznie więcej (*taxi['Total_ amount']. groupby(taxi.RateCodeID).sum()*). Potwierdza to wypisanie median - dla JFK jest to 58,34$, zaś dla Newark 69,32$. Podobnie wygląda ta zależność dla średniej:

- 2 - 56.964378;
- 3 - 63.359890,

oraz dla kwartyla trzeciego:

- 2 - 68.8400;
- 3 - 86.3574.

Tym samym więcej kursów odbywa się z/do portu JFK, ale na ogół są one związane z małymi kwotami. Co prawda na medianie tego aż tak nie widać (różnica 11 $ zdaje się być niewielka), ale przy całkowitej liczbie zleceń jest to znacząca kwestia. Na tyle, że znacznie mniej kursów na stawce Newark generuje więcej przychodu niż na JFK. Oznacza to, że korporacja powinna wysyłać mniej samochodów w stronę lotniska Newark, ale bardziej skupić się aby zawsze tam ktoś czekał na potencjalnych pasażerów.

(4) Jakie rejony wykazują największe zapotrzebowanie na taksówki?

Dobrym sposobem na ukazanie tego jest umieszczenie lokalizacji z pól *Pickup_ longitude* i *Pickup_ latitude* oraz/lub *Dropoff_ longitude* i *Dropoff_ latitude* na mapie Nowego Jorku i okolic. Warto to zrobić przy pomocy ArcGIS API dla Pythona. Więcej o tej bibliotece do pracy z mapami oraz danymi geoprzestrzennymi (w tym jak ją wykorzystać do naszych celów) na stronie [**19**].

Na potrzeby pracy stworzyliśmy innym sposobem prowizoryczne zrzutowanie współrzędnych geograficznych za pomocą poniższego kodu.

```
pk = taxi[['Pickup_latitude','Pickup_longitude']]
fig = plt.figure(figsize=(20,20))
plt.scatter(pk['Pickup_longitude'].head(15000),
```

```
    pk['Pickup_latitude'].head(15000), color='green', s=0.5,
    label='Pozycje')

plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.xlim(-74.00,-73.85)
plt.ylim(40.65, 40.85)
plt.gca().set_facecolor('black')
plt.legend()
show(plt)
```

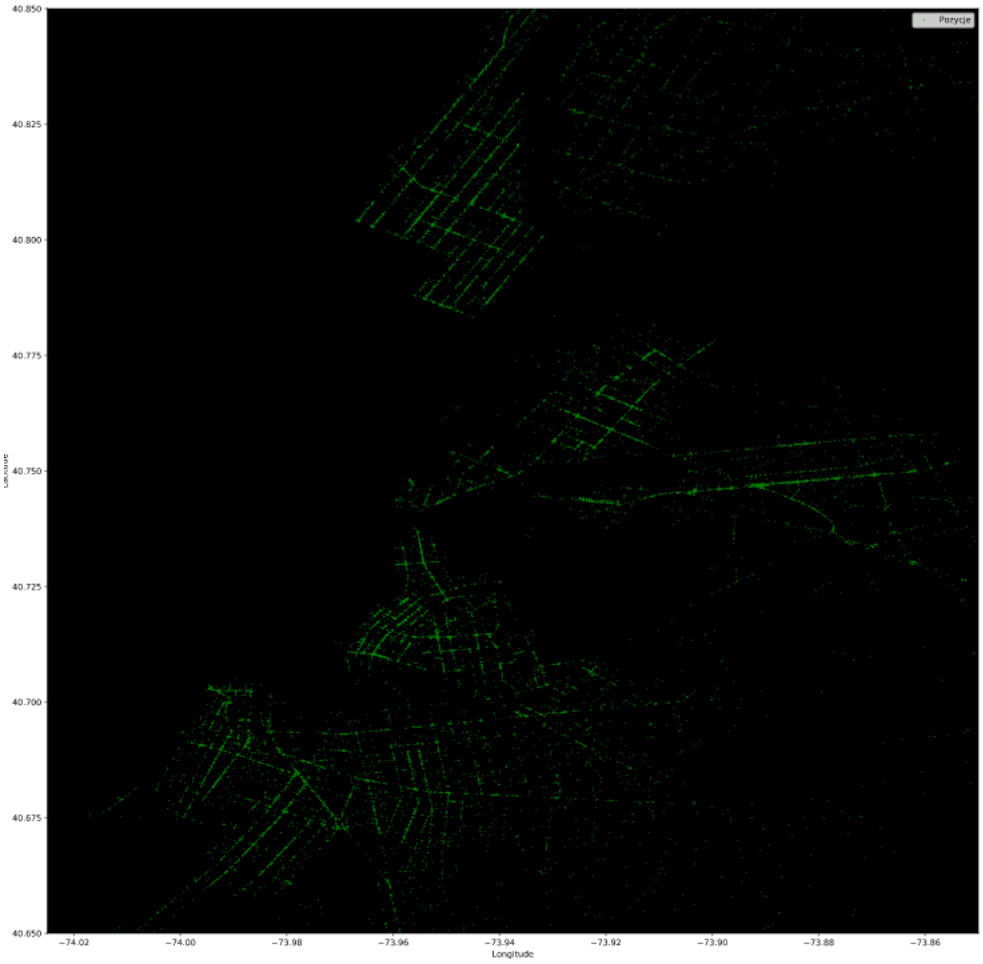
Wynik uzyskany na próbce z 15 tysięcy pierwszych kursów zaprezentowaliśmy na rysunku 5.9. Wybranie większego zbioru skutkowałoby zatraceniem informacji o częstszych miejscach. Jednocześnie warto spróbować uczynić to samo dla losowo wybranych kursów (aby wyeliminować sytuacje, że na przykład jakaś dzielnica była związana z dużymi wydarzeniami kulturalnymi). Jeśli istnieje taka potrzeba, ustawienie czarnego tła (*set_facecolor('black')*) zamieniamy na zrzut ekranu odpowiednio przyciętej mapy Nowego Jorku (rysunek 5.10, źródło mapy: [**18**]).

Na bazie zaprezentowanej mapy możemy stwierdzić, które miejsca Nowego Jorku są bardziej zatłoczone (północnozachodni Bushwick), a które mniej. Poza tym przypuszczamy gdzie rozmieszczono centra handlowe/wielkie zakłady i tereny prywatne/inne miejsca raczej niedostępne dla taksówek (stocznia powyżej dolnego oznaczenia drogi 278). Jest to więc dobry sposób odnajdywania pierwszych potencjalnych miejsc na założenie restauracji przez osobę nieznającą miasta.

Poza powyższym, od razu zauważamy brak taksówek w rejonie Manhattanu. Nie jest to jednakże błąd, a efekt regulaminu, który odnosi się do zielonych taksówek. Ich kierowcy mają bowiem zakaz odbierania pasażerów z tejże dzielnicy (poza północną częścią powyżej West 110th street i East 96th street). Tworzenie tego typu map może dodatkowo pomóc wyłapać tych taksówkarzy, którzy łamią ogólne postanowienia. Tym samym możemy zadać kolejne pytanie.

(5) Jak często kierowcy zielonych taksówek zaczynają kursy w rejonie zakazanym (Manhattan)?

Pierwszym sposobem na uzyskanie odpowiedzi jest rozbudowanie mapy z poprzedniego punktu. Drugi z nich to podzielenie dzielnicy na kwadraty, zdefiniowanie ich granicznych szerokości i długości geograficznych, a następnie wyszukanie w zbiorze lokalizacji, jakie mieszczą się pomiędzy nimi (kod



RYSUNEK 5.9. Prowizoryczne rozmieszczenie początków kursów - na czarnym tle.

podobny do tego, który pojawił się przy oczyszczaniu problematycznych wartości geoprzestrzennych). Po tym wystarczy zsumować liczbę zakazanych kursów ze wszystkich czworokątów. Niestety metoda ta jest obarczona dużym ryzykiem uzyskania błędnych wyników - powinniśmy dążyć do zamienienia kwadratów w nieregularną figurę, wyrysowującą właściwie kształt Manhattanu.

(6) Jak różnią się pomiędzy sobą poszczególne dni tygodnia w kwestii kursów?

Zacznijmy od podliczenia liczby kursów w zależności od dnia, co dokonamy z pomocą kodu *pd.DatetimeIndex(taxi['lpep_pickup_datetime']).dayofweek.value_counts()*. Uzyskamy sumy, pogrupowane w zgodzie z zasadą: 0 - poniedziałek, 6 - niedziela:

- 5 - 482610 (sobota);
- 4 - 418289 (piątek);
- 6 - 391723 (niedziela);
- 3 - 361675 (czwartek);

RYSUNEK 5.10. Prowizoryczne rozmieszczenie początków kursów - w połączeniu z mapą Nowego Jorku.

- 2 - 334524 (środa);
- 1 - 300744 (wtorek);
- 0 - 285768 (poniedziałek).

Jak przypuszczaliśmy, największy ruch jest w piątek i w weekend (w dokładnej kolejności: 5, 4, 6). Co ciekawe, będąca na trzecim miejscu niedziela to o około 20 % mniej kursów niż w sobotę. Zaczynając od poniedziałku z najniższym zainteresowaniem, reszta dni wzrasta wartościami w kalendarzowej kolejności.

Średnia cena kursu nie różni się zbytnio w ciągu tygodnia. Przy pomocy komendy *taxi.groupby(pd.DatetimeIndex(taxi['lpep_pickup_datetime']).dayofweek).Total_amount.mean()* dowiemy się, że mieści się ona bowiem w przedziale (15.74, 16.22). Podobnie małe skoki wartości pojawiają się przy medianie:

- 0 - 12.88;
- 1 - 12.96;

- 2 - 12.96;
- 3 - 13.00;
- 4 - 13.50;
- 5 - 13.55;
- 6 - 13.50.

Kwartyle dolne są niemal takie same przez cały tydzień:

- 0 - 9.35;
- 1 - 9.35;
- 2 - 9.36;
- 3 - 9.36;
- 4 - 9.49;
- 5 - 9.36;
- 6 - 9.35.

Zauważamy przy tym największy skok ponownie w piątek.

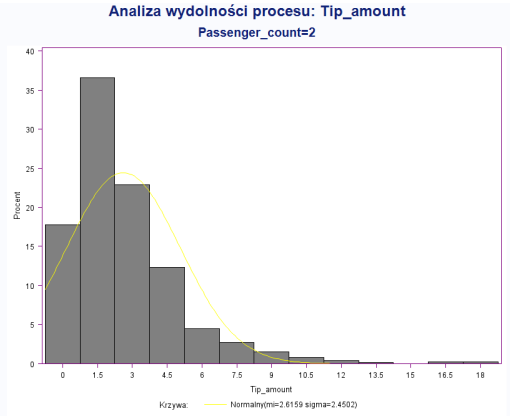
Podsumowanie: Dawanie pracownikom korporacji wolnego weekendu lub piątku nie jest dobrym pomysłem, ze względu na duże oblężenie w tych dniach (miejscowi jeżdżą na niemal takie same kwoty, ale częściej). Najlepiej odłożyć je na początek tygodnia, albo zatrudnić dodatkowe osoby, które będą niwelowały braki pomiędzy piątkiem, a niedzielą.

2.2. Środowisko SAS. Środowisko SAS udostępnia cały szereg programów. Jednym z nich jest SAS Enterprise Guide. Zaprezentujmy przykładowe sposoby jego wykorzystania.

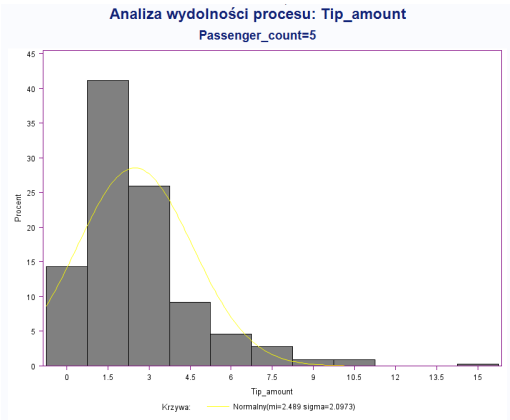
(1) Więcej dostaniemy napiwków z kursu z mniejszą ilością osób (na przykład 2) czy z większą (przyjmijmy 5 pasażerów)?

Większe zbiory danych w środowisku SAS często skutkują problemami z komputerami o niższych parametrach. Chcielibyśmy tutaj pokazać, że i w takiej sytuacji warto próbować analizować zbiory wielkości big data. Dokonujemy tego poprzez próbę losową. Na nasze potrzeby ustaliliśmy jej liczebność na poziomie 10 000 wierszy. SAS daje sposobność określenia które pola pozostaną (w razie usunięcia czegoś, jeszcze bardziej zwiększamy prędkość późniejszych programów), więc wybraliśmy jedynie *Passenger_ count* oraz *Tip_ amount*. Dodatkowo ze względu na istnienie informacji o napiwkach tylko w sytuacji płatności kartą, wstawiliśmy filtr zapytań na *Payment_ type*=1.

Po wygenerowaniu próbki uruchomiliśmy histogram z paroma dodatkowymi miarami statystycznymi. Zmienną analizowaną zostało przy tym pole *Tip_ amount*, grupowane według liczby pasażerów. Uzyskamy wówczas między innymi wykresy, które przedstawiono na rysunku 5.11.



(A) histogram dla 2 osób



(B) histogram dla 5 osób

RYSUNEK 5.11. Histogramy dla kwot napiwków

Bazowe miary statystyczne			
Położenie		Zmienność	
Średnia	2.615913	Odchylenie std.	2.45015
Mediana	2.040000	Wariancja	6.00325
Moda	0.000000	Rozstęp	18.71000
		Rozstęp międzyćwiartkowy	2.39000

(A) dla 2 pasażerów

Bazowe miary statystyczne			
Położenie		Zmienność	
Średnia	2.488994	Odchylenie std.	2.09733
Mediana	2.000000	Wariancja	4.39880
Moda	0.000000	Rozstęp	15.00000
		Rozstęp międzyćwiartkowy	1.96500

(B) przy 5 pasażerach

RYSUNEK 5.12. Bazowe informacje dla Tip_amount.

Histogramy wydają się być podobne. Jednakże już po chwili zauważamy różnice - na przykład w tym, że pięć osób jest bardziej skłonne nie dawać w ogóle napiwku albo dać go niewiele (około 18%, gdzie na 5.11a jest około 14%). Bazowe miary statystyczne poznamy na 5.12, zaś kwantyle zostały zaprezentowane na rysunkach 5.13. Na bazie zebranych tam informacji możemy przypuszczać, że przy większej liczbie pasażerów nie istnieje nacisk grupy na większe napiwki. Wyższe pojawiają się bowiem przy dwóch osobach (niemal wszystkie miary to pokazują!). Być może chęć zaimponowania jednej osobie (na przykład na randce) jest silniejsza niż w przypadku paczki znajomych.

Mała uwaga - na obu histogramach widzimy punkty oddalone, które powstały po zmniejszeniu badanego zbioru. Jak pamiętamy, obserwacje powinny mieścić się w przedziale $(Q_1 - 1.5IQR, Q_3 + 1.5IQR)$. Dla pięciu osób daje to $(-1.3475, 6.0725)$, a dla dwóch - $(-2.435, 7.125)$. Jest to błąd, nie mamy przecież minusowych napiwków. Liczba ujemna pojawia się tutaj z powodu dużej liczby zerowych napiwków, które także chcieliśmy ująć w naszych badaniach. Tym samym lepszym sposobem wyeliminowania punktów oddalonych jest spojrzenie na histogramy. Powinniśmy ustalić tutaj

Kwantyle (definicja 5)	
Poziom	Kwantyl
100% Maks.	18.71
99%	11.65
95%	7.26
90%	5.36
75% Q3	3.54
50% Mediana	2.04
25% Q1	1.15
10%	0.00
5%	0.00
1%	0.00
0% Min.	0.00

(A) dla 2 osób

Kwantyle (definicja 5)	
Poziom	Kwantyl
100% Maks.	15.000
99%	10.000
95%	6.660
90%	5.080
75% Q3	3.125
50% Mediana	2.000
25% Q1	1.160
10%	0.000
5%	0.000
1%	0.000
0% Min.	0.000

(B) dla 5 osób

RYSUNEK 5.13. Kwantyle Tip_amount

$Tip\_amount < 13.5$. W takiej sytuacji konkluzja jest jednakże wciąż taka sama.

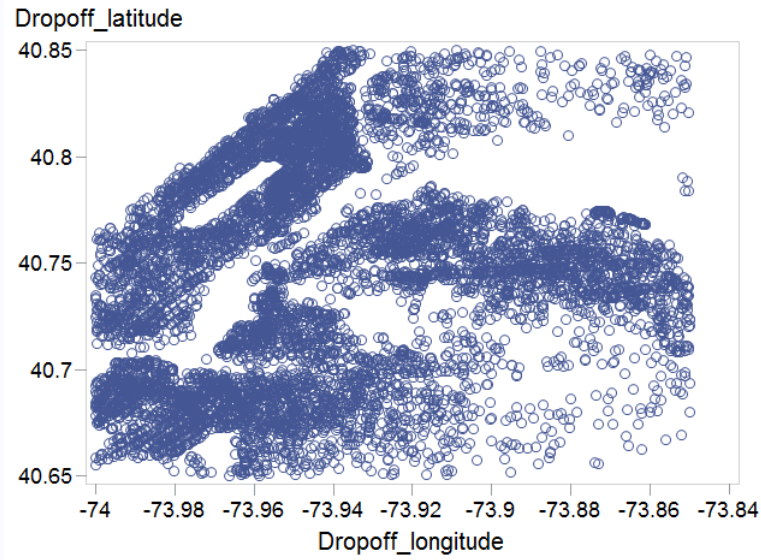
Porównajmy jak bardzo próbka danych odbiega od całego zbioru. Dla dwóch osób będą to następujące wartości miar środka (po lewej wynik z 10000 wierszy, po prawej z wszystkich):

- średnia: 2.6159 i 2.6591;
- mediana: 2.04 i 2.00.

Pobieranie próbki informacji stanowi pewien sposób radzenia sobie ze zbyt dużymi zbiorami. Niestety - jak widzimy powyżej - musimy się wówczas liczyć z wynikającymi z tego błędami. W tym przypadku były one niewielkie (błąd względny: 0.01625 dla średniej oraz 0.02 dla mediany), ale nie zawsze tak jest. Ponadto w większości przypadków nie jesteśmy w stanie określić, jak bardzo daleko jesteśmy od prawdy - nie dysponujemy wartościami dla całej populacji. Należy w takim razie dążyć do zebrania jak najwięcej informacji do analizy.

(2) Dokąd jeżdżą taksówki? Tym samym gdzie potencjalnie będą mogły zacząć kolejny kurs?

Pytanie to powstało aby udowodnić, że przy pomocy środowiska SAS także możemy przedstawić prowizoryczną mapę. Wykorzystaliśmy do tego



RYSUNEK 5.14. Uproszczony obraz lokalizacji 15 tysięcy losowych kursów.

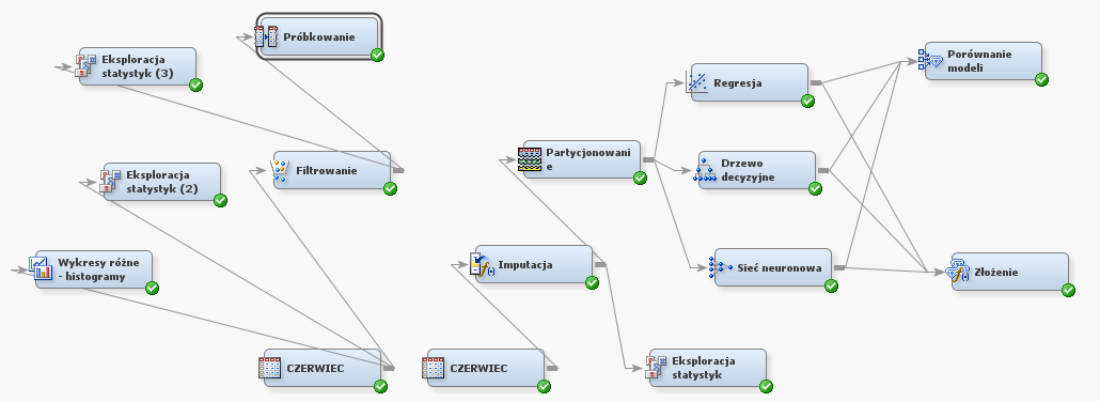
wykres punktowy (patrz: 5.14) i próbkę 15 000 wierszy, tym razem uzyskanych losowo.

Rzucający się w oczy Central Park (pochylony prostokąt przecinający współrzędne 40.8N, 73.96W) pomaga zauważyć, że wiele kursów realizowanych jest w stronę Manhattanu. Tym samym dużo kierowców musi tracić czas oraz paliwo na wydostanie się z tejże dzielnicy.

Jeszcze lepiej pracuje się przy pomocy SAS Miner. Bazując na projekcie, który przygotowaliśmy powyżej, eksportujemy jeden z plików miesięcznych (na przykład czerwcowy) jako etap projektu. Dzięki temu baza danych w formacie csv staje się plikiem danych SAS-owych (*.sas7bdat). Następnie w SAS Miner dodajemy bibliotekę (Plik -> Nowy -> Biblioteka) związaną z folderem, do którego zamieściliśmy wcześniej plik. Po tym włączamy kreator źródeł danych (Plik -> Nowy -> Źródło danych).

Po stworzeniu nowego diagramu możemy zacząć pracować nad informacjami o taksówkach. Nasz przykładowy diagram został zaprezentowany na rysunku 5.15. Widzimy na nim dwa odnośniki do danych wejściowych (okienka ”CZERWIEC”), a wokół nich kilka związanych z nimi węzłów.

Z sekcji Eksploracja pochodzą ”Wykresy różne” oraz ”Eksploracja statystyk”. Pierwszy z nich da nam dojście do wykresów dla wszystkich pól. Nie musimy pisać żadnych kodów, wystarczy uruchomić ten węzeł, a następnie zaprezentować rezultaty. Drugi zaś to zebranie podstawowych statystyk opisowowych analizowanych kolumn - średnia, odchylenie standardowe, liczba niebrakujących wartości oraz braków, minimum, mediana, maksimum, skośność (miara asymetrii rozkładu), kurtoza (określenie rozmieszczenia wartości wokół średniej). Tabela ta pozwala szybko sprawdzić



RYSUNEK 5.15. Wygląd przykładowego diagramu w SAS Miner.

czy dobrze oczyściliśmy zbiór. W razie zauważenia takowej potrzeby dokonujemy filtracji lub imputację (sztuczne wstawienie danych).

W przypadku gdy potrzebujemy skorzystać z technik modelowania danych takich jak regresja, sieć neuronowa, czy drzewa decyzyjne, zmieniamy w zbiorze danych interesujące nas pole na zmienną celu. Mając ją wybraną, warto sprawdzić "Eksploracje statystyk" - chociażby dla wykresu korelacji pomiędzy tą kolumną, a wszystkimi pozostałymi.

Na diagramie z 5.15 zauważamy, że wyliczone węzły modelowania zostały poprzedzone partycjonowaniem. Jest to podział zbioru na trzy części:

- treningowa - na niej buduje się model, najlepiej ustalić ją powyżej 40%;
- walidacyjna - do porównywania różnych modeli;
- testowa - służy do końcowej oceny modelu wybranego dzięki danym walidacyjnym.

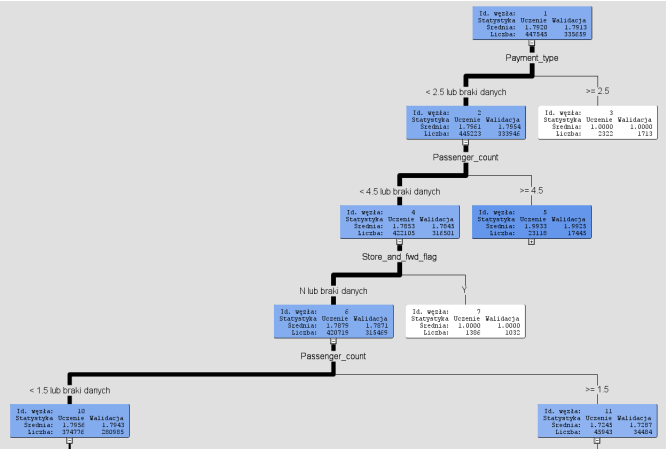
Ich wartości procentowe ustawiamy we właściwościach tego węzła. Na potrzeby pracy nie zmienialiśmy proporcji domyślnej - 40:30:30.

Drzewo decyzyjne to zbiór węzłów decyzyjnych, połączonych gałęziami. Rozchodzą się one od korzenia do liści, ułożonych na dole okna. Węzły te numerujemy od góry, w rzędach, od lewej do prawej. Podstawowe drzewo decyzyjne, posiadające wszystkie możliwe gałęzie, związane jest z

$$w = 1 + 2 + 2^2 + 2^3 + 2^4 + \cdots + 2^{n-1} = \sum_{i=0}^{n-1} 2^i$$

węzłami, gdzie n to liczba rzędów. Często jednakże stosuje się rezygnację z pewnych gałęzi, które nic nie wnoszą do dalszej analizy. Najmniejsza możliwa liczba węzłów w takiej sytuacji może być wyliczona ze wzoru:

$$w = 1 + 2 + 2 + \cdots + 2 = \sum_{i=0}^{n-1} 2.$$



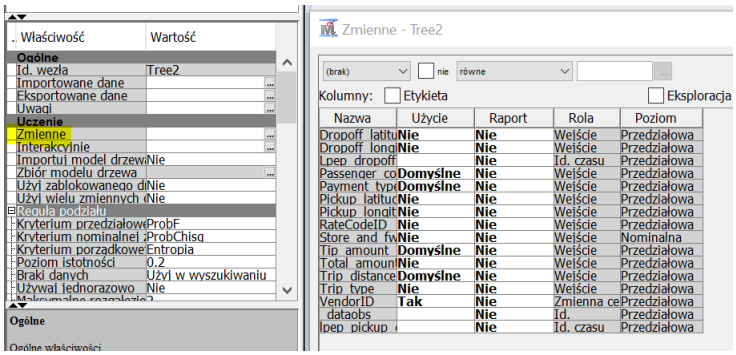
RYSUNEK 5.16. Widok na przykładowe drzewo decyzyjne, uzyskane przy pomocy SAS Miner.

Drzewo decyzyjne pokaże nam między innymi jak dana wartość przy jednym polu, wpływa na wybór konkretnej wartości przy drugiej kolumnie. Na przykładzie drzewa z rysunku 5.16 - pogrubiona gałąź oraz wyliczone informacje ukazują, że prawie wszystkie kursy były związane z *Payment_time* o wartości 1 lub 2. Czyli: płatność gotówką lub kartą. Wśród nich pojawiło się więcej przypadków z czterema pasażerami lub mniej.

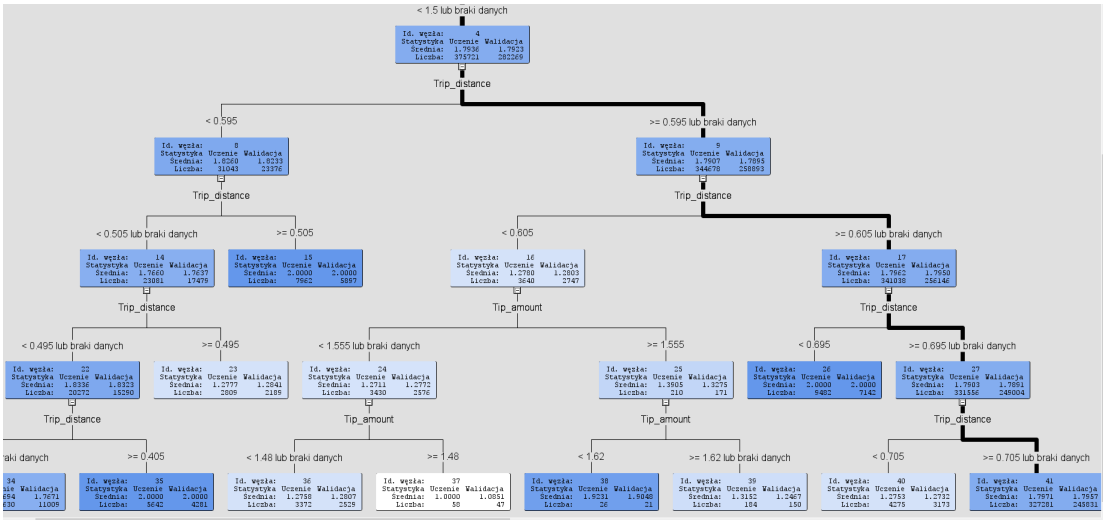
Oczywiście warto rozważyć sprawdzenie drzewa decyzyjnego po usunięciu na przykład wartości *Payment_time* powyżej 2. Jednakże na potrzeby pracy pozostawiliśmy przed uruchomieniem drzewa decyzyjnego część rzadziej występujących wartości, aby efekt był bardziej czytelny i zrozumiały.

Przedstawmy jeszcze za pomocą drzewa decyzyjnego jak wyglądają jednoosobowe kursy w odniesieniu do napiwków oraz przebytej trasy. Jak pamiętamy z poprzednich analiz, jeden pasażer zdarza się najczęściej. Po odpowiednim sfiltrowaniu pola *Passenger_count* (węzeł "Filtrowanie" -> właściwości -> zmienne przedziałowe) ustawmy drzewo decyzyjne. Z właściwości wybierzmy punkt odnoszący się do zmiennych, a następnie określmy użycie jedynie interesujących nas kolumn (rysunek 5.17). Po uruchomieniu uzyskamy drzewo, którego część zaprezentowano na rysunku 5.18. Pomocą w analizie jest sprawdzenie w rezultatach wykresu kafelkowego (patrz: 5.19). Każdy kafelek to jeden węzeł, ustawiony w takiej samej kolejności jak w drzewie. Jego wielkość ukazuje jaka liczebność obserwacji wchodzi w jego skład, gdzie okno to 100% kursów.

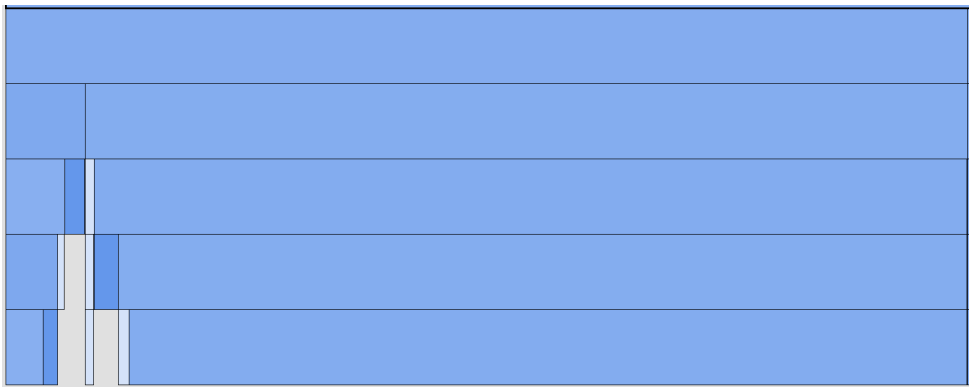
W przypadku tylko jednego pasażera na ogół są to podróże dłuższe niż 0.605 mili. Stanowią one ponad 90% wszystkich, co wyliczamy porównując węzeł 17 z sumą węzłów 8 oraz 16 (ze względu na usunięcie części wykresu, ich numerowanie zaczynamy w tym przypadku od cyfry 4). Jednocześnie więcej niż 80% przypadków to podróże o dystancie powyżej 0.695 mili (węzeł 27 kontra suma węzłow: 8, 16



RYSUNEK 5.17. Sposób wyboru konkretnych pól do drzewa decyzyjnego.



RYSUNEK 5.18. Drzewo decyzyjne uzyskane przy zmiennych wybranych jak w poprzednim rysunku.



RYSUNEK 5.19. Wykres kafelkowy dla powyższych ustaleń.

oraz 26). Najkrótsze kursy (poniżej 0.505 mili) zdarzają się dwa razy częściej niż te pomiędzy 0.505 mili, a 0.605 mili. Trafia się więc dość dużo małych przewozów, na których cenę znacząco wpływają kwoty początkowe. Może korporacja powinna pomyśleć nad ich zwiększeniem?

Najmniejsze prawdopodobieństwo wystąpienia danego przypadku zauważane jest przy dystansie z przedziału (0.595, 0.605), za który dano napiwek pomiędzy 1.48$, a 1.555$. Pomaga to zauważyć biały kolor węzła 37.

Większość narzędzi dostępnych w SAS Miner można zaprogramować w Pysparku. Przykłady kodów można znaleźć w referencji [**20**].

ROZDZIAŁ 6

Zakończenie

Rozdział ten nie będzie stanowił podsumowania każdego kroku analizy. Tylko z tych niewielu stron możnaby spisać długie konkluzje (które już częściowo podaliśmy), a przecież nie opisaliśmy wszystkich możliwych sposobów statystycznego ujęcia zbioru, a jedynie podstawowe metody. Istnieje wiele sposobów analizy, o których tutaj nawet nie wspomnieliśmy, a w prawdziwej pracy analityka dość często się przewijają. Chcemy jednakże zauważyć, że zależało nam na dokładnym zaprezentowaniu tego co nas interesowało - małej "walki" pomiędzy środowiskiem SAS, a platformą programistyczną Hadoop. Z jednej strony pokazaliśmy, że odpowiednio skonfigurowany klaster z Hadoopem doskonale radzi sobie z zadaniami analizy statystycznej. Z drugiej - mamy do dyspozycji narzędzia dostarczane przez środowisko SAS. Pomiędzy tymi dwoma gigantami istnieje pewna "unia", połączenie, które kryje się za SDLfH. Z naszej strony zalecamy, aby w przypadku takiej możliwości dokonywać obliczeń z wykorzystaniem obu programów. Jeżeli jednakże nie dysponujemy taką sposobnością, programowanie w Pysparku powinno w znaczącym stopniu wyjść naprzeciw naszym potrzebom. Niestety takie rozwiązanie wymaga więcej pracy, a tym samym i czasu. Wystarczy chociażby spojrzeć na kody niezbędne do zaprogramowania drzewa decyzyjnego, a następnie porównać je z wysiłkiem potrzebnym do uzyskania tego samego w SAS Miner.

Przy pomocy rozległej dokumentacji dostarczanej przez firmę SAS, opisaliśmy jeden ze sposobów połączenia ekosystemu Hadoop z oprogramowaniem z rodziny SAS - SDLfH. Podstawowe cechy tego oprogramowania zaprezentowaliśmy w rozdziale trzecim, zaś w rozdziale czwartym wytłumaczyliśmy dogłębnie jego proces instalacji. Teoretyczną część tekstu dopełniliśmy podstawowymi zagadnieniami z dziedziny big data.

Pojawiające się w poprzednim rozdziale wnioski były inwencją twórczą autora, spisaniem potencjalnej możliwości. Przy prawdziwym zleceniu mielibyśmy kontakt z korporacją, która rozwiałaby wszelkie wątpliwości oraz narzuciła kilka swoich pytań.

Spis rysunków

Bibliografia

[1] Spis najczęściej odwiedzanych stron internetowych, http://www.alexa.com/topsites, stan na: 05.2017

[2] strona Roberta Lewandowskiego na portalu Facebook.com, https://www.facebook.com/rl9official/, stan na: 05.2017

[3] strona sieci Play na portalu Facebook.com, https://www.facebook.com/Play/, stan na: 05.2017

[4] Statystyki firmy Facebook, https://newsroom.fb.com/company-info/#statistics, stan na: 05.2017

[5] Czy Big Data to Big Brother, https://archive.wilgucki.pl/2011/10/czy-big-data-to-big-brother.html, stan na 10.2016

[6] Russell Jurney, Zwinna analiza danych. Apache Hadoop dla każdego, wydawnictwo Helion, 2015

[7] Daniel T. Larose, Odkrywanie wiedzy z danych - Wsprowadzenie do eksploracji danych, wydawnictwo PWN, Warszawa 2006

[8] Mathias Rosenthal, Historia w pigułce: Big Data, http://www.brief.pl/artykul,2824,historia_w_pigulce_big_data.html, stan na: 15.10.2016

[9] Marian Płaszczyca, Kowariancja i korelacja Pearsona, http://www.statystyka.az.pl/kowariancja-i-korelacja.php, stan na: 09.2017

[10] Yahoo! Hadoop Tutorial, https://developer.yahoo.com/hadoop/tutorial/, stan na: 10.2016

[11] Michał Wąsowski, Co to jest Big Data, http://natemat.pl/52911,co-to-jest-big-data-nie-bojcie-sie-na-pewno-nie-inwigilacja, stan na: 02.2017

[12] PowerBy Hadoop, https://wiki.apache.org/hadoop/PoweredBy, stan na: 09.2017

[13] dokumentacja Hadoop, http://hadoop.apache.org/docs/current/, stan na: 05.2017

[14] Tutorial Hortonworks, https://hortonworks.com/tutorial/getting-started-with-hdf-sandbox/, stan na: 05.2017

[15] Przewodnik po HDFS, https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html, stan na: 08.2017

[16] Tutorial biblioteki Pandas, http://pandas.pydata.org/pandas-docs/stable/tutorials.html, stan na: 08.2017

[17] Dokumentacja biblioteki Seaborn, https://seaborn.pydata.org/, stan na: 09.2017

[18] Google Maps, https://www.google.pl/maps/, stan na: 09.2017

[19] Wykorzystanie narzędzi ArcGIS do omawianego zbioru danych, https://developers.arcgis.com/python/sample-notebooks/analyze-new-york-city-taxi-data/, stan na: 09.2017

[20] Przykłady wykorzystania Pysparka do statystyki znanej ze środowiska SAS, https://spark.apache.org/docs/latest/ml-classification-regression.html, stan na: 09.2017

[21] Oficjalna strona Fundacji Apache, http://www.apache.org/, stan na: 01.2017

[22] Badany zbiór, http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml, stan na: 03.2017

Pomoc od SAS:

[23] Patryk Choroś, Czym jest "Big Data", http://www.sas.com/pl_pl/news/artykuly/2014/czym-jest-big-data.html, stan na: 09.2016

[24] SAS 9.4 Support for Hadoop, http://support.sas.com/resources/thirdpartysupport/v94/hadoop/, stan na: 09.2016

[25] Oficjalna strona SDLfH, https://www.sas.com/en_us/software/data-loader-for-hadoop.html, stan na: 05.2017

[26] SDLfH - wymagania sprzętowe, http://support.sas.com/documentation/installcenter/en/ikdmddhadpvofrsr/69853/HTML/default/index.html, stan na: 05.2017

[27] Pobieranie i instalacja oprogramowania SAS 9.4, https://www.sas.com/content/dam/SAS/pl_pl/doc/support/instalacja-oprogramowania-sas-9-4/sas94_pobieranie_instalacja.pdf, stan na: 05.2017

[28] Wymagania sprzętowe dla SAS 9.4, http://support.sas.com/documentation/installcenter/en/ikfdtnwinsr/67228/PDF/default/sreq.pdf, stan na: 05.2017

[29] Oficjalne informacje o SDLfH, https://www.sas.com/content/dam/SAS/en_us/doc/factsheet/sas-data-loader-hadoop-107474.pdf, stan na: 05.2016

[30] Spis wspieranych wersji Hadoopa, http://support.sas.com/resources/thirdpartysupport/v94/hadoop/hadoop-distributions.html, stan na: 05.2017

[31] Tutorial instalacji SDLfH, http://documentation.sas.com/?docsetId=dmddicg&docsetTarget=p0w33lf3g7kne2n189yynzugpedb.htm&docsetVersion=3.1&locale=pl

[32] Webinarium: Analiza i przetwarzanie dużych wolumenów danych na platformie SAS, https://www.youtube.com/watch?v=v1Cr1-Iymww&feature=youtu.be, stan na: 09.2017