P O L I T E C H N I K A R Z E S Z O W S K A
im. Ignacego Łukasiewicza
WYDZIAŁ MATEMATYKI I FIZYKI STOSOWANEJ

Predictive maintenance w wykrywaniu uszkodzeń dysków twardych

Maciej Żak

Rzeszów, 11.05.2022r

Spis treści

1. Dysk twardy

Jest to dysk komputerowy, pamięć masowa wykorzystująca nośnik magnetyczny do przechowywania danych. Nazwa „dysk twardy" wynika z zastosowania twardego materiału jako podłoża dla właściwego nośnika, w odróżnieniu od dysku miękkiego, w którym nośnik magnetyczny nanoszono na podłoże elastyczne. Pojemność dysków twardych w komputerach stacjonarnych wynosiła od 3,75 MB (pierwszy dysk twardy) do 20 TB, natomiast w laptopach od kilkudziesięciu gigabajtów do kilku terabajtów. Opracowano również miniaturowe dyski twarde typu microdrive, o pojemnościach od kilkuset MB do kilku GB, przeznaczone dla cyfrowych aparatów fotograficznych i innych urządzeń przenośnych. Dla dysków twardych najważniejsze są następujące parametry: pojemność, szybkość transmisji danych, czas dostępu do danych, prędkość obrotowa dysków magnetycznych (obr./min) oraz średni czas bezawaryjnej pracy. Kilka dysków twardych można łączyć w macierz dyskową, dzięki czemu można zwiększyć niezawodność przechowywania danych, dostępną przestrzeń na dane, zwiększyć szybkość odczytu/zapisu.

2. Awarie dysku twardego

Występują wtedy, gdy dysk twardy przestaje działać i staje się niezdolny do wykonywania swoich zadań zwykle nie tracąc wszystkich danych na jej temat. Dysk twardy może upaść ze względu na szeroki wachlarz powodów. Można je sklasyfikować w następujący sposób:

Uszkodzenia elektroniki

Najczęściej powstają na skutek: źle działającego zasilacza komputera, przepięcia sieci energetycznej, wyładowań atmosferycznych, zwarcia w komputerze.

W przypadku uszkodzenia elektroniki najczęstszymi objawami (chodź nie zawsze, czasami nie widać uszkodzeń gołym okiem) są widoczne spalenia elementów elektronicznych, brak oznak pracy nośnika po podłączeniu zasilania tzn. dysk nie wydaje jakichkolwiek oznak pracy - dźwięków czy drgań. W niektórych przypadkach uszkodzenie elektroniki zewnętrznej pociąga za sobą również uszkodzenie elektroniki wewnętrznej dysku znajdujących się na ramieniu głowicy wewnątrz dysku.

Uszkodzenia logiczne

Zwykle dysk jest całkowicie sprawny natomiast nie ma dostępu do danych, gdyż uszkodzona została struktura logiczna dysku. Najczęstsze przyczyny powstawania uszkodzeń logicznych to: usunięcie danych, format partycji lub dysku, uszkodzenia partycji dysku, uszkodzenie MBR lub GPT (partycja typu RAW), wirus komputerowy.

W większości wyżej wymienionych przypadkach dane można odzyskać z wyjątkiem sytuacji, kiedy dane zostaną nadpisane tzn. nowe dane zostają zapisane w miejscach, gdzie znajdowały się wcześniejsze dane, które chcemy odzyskać. Niestety przypadku nadpisania danych nie ma możliwości odzyskania danych.

Uszkodzenia mechaniczne

Są to bardzo poważne usterki i częściej prowadzą do częściowej lub całkowitej utraty danych. Dyski twarde z wyjątkiem dysków SSD są urządzeniami bardzo delikatnymi i wrażliwymi na czynniki zewnętrzne. Uszkodzenia mechaniczne mogą powstawać samoistnie na skutek normalnego zużycia materiału czy wady fabrycznej, ale główne czynniki ich powstawania to: wstrząsy, upadki i uderzenia (szczególnie podczas pracy nośnika), wysoka wilgotność lub zalania czy temperatura. Do głównych uszkodzeń mechanicznych należą: uszkodzenie głowic dysku twardego, uszkodzenie silnika dysku twardego, uszkodzenie powierzchni talerzy, uszkodzenia obudowy dysku. W przypadku uszkodzeń mechanicznych dysków twardych bardzo często objawy są słyszalne i z dysku wydobywają się dziwne odgłosy takie jak: stukanie, pikanie, odgłosy tarcia czy głośne kręcenie silnika.

3. Przewidywanie awarii dysku twardego.

Konserwacja to wszystko, co robimy, aby nasze maszyny i urządzenia działały tak, jak powinny. Kiedy wykonujemy konserwację, są dwie główne rzeczy, których staramy się uniknąć. Pierwszą z nich jest zużywająca się maszyna i działająca nieefektywnie lub niespodziewanie się psująca. Jest to prawdopodobnie bardziej oczywiste – i często bardziej kosztowne – wyzwanie. Nieoczekiwana przerwa może stanowić poważne zagrożenie dla pracowników i środowiska, zepsute maszyny muszą zostać wymienione lub naprawione, a nieefektywny lub zepsuty sprzęt może oznaczać utratę przychodów. Ale sama konserwacja może być kosztowna i niebezpieczna, a więc drugą rzeczą, której chcielibyśmy uniknąć, jest robienie tego niepotrzebnie i niewykorzystywanie pełnego okresu użytkowania naszych maszyn. Powszechną strategią konserwacji jest run to failure (R2F), czasami nazywane

konserwacją korekcyjną. Jest to w zasadzie najprostsza możliwa strategia: jeśli maszyna się zepsuje, naprawiamy ją. Tego rodzaju czysto reaktywne podejście może prowadzić do nieprzewidywalnych wyników, a co za tym idzie do wielu nieoczekiwanych awarii maszyny. Alternatywnie mamy konserwację zapobiegawczą, w której naprawy są przeprowadzane zgodnie z regularnym, zorientowanym na czas harmonogramem. Dzięki takiemu podejściu możemy być w stanie zmniejszyć liczbę nieoczekiwanych przerw, ale jesteśmy naiwnie proaktywni. Niektóre z naszych czynności konserwacyjnych mogą być całkowicie niepotrzebne i nadal możemy mieć nieoczekiwane przerwy, ponieważ tak naprawdę nie bierzemy pod uwagę stanu maszyny.

Predictive Maintenance

Strategia przeprowadzania konserwacji w oparciu o szacowany stan urządzenia. Konserwacja predykcyjna, zwana również konserwacją opartą na stanie, może pozwolić nam zmniejszyć niepewność działań konserwacyjnych poprzez inteligentną pro aktywność i wykonywanie konserwacji we właściwym czasie. Umożliwia to gromadzenie dużych ilości danych oraz możliwość przetwarzania i analizowania tych danych. Używamy tych danych – często z wbudowanych czujników lub dzienników maszyn – aby znaleźć dowody na degradację maszyny. Po postawieniu diagnozy stanu maszyny na podstawie zebranych danych, możemy wykorzystać tę wiedzę do wykonania odpowiednich czynności konserwacyjnych. Prowadzi nas to do następującego podstawowego przepływu pracy: zbieranie danych, szacowanie stanu maszyny i wykonywanie odpowiednich działań.

Uczenie maszynowe pojawia się na drugim etapie przepływu pracy, wykorzystując zebrane dane do przewidywania stanu maszyny. Oczywiście konserwacja predykcyjna wydaje się być idealną strategią konserwacji. Prawidłowo wykonane możemy poprawić stan sprzętu, zmniejszyć wskaźniki awarii sprzętu, zminimalizować koszty konserwacji i zmaksymalizować żywotność sprzętu. Są też mniej oczywiste korzyści. W zależności od naszego podejścia możemy być w stanie powiązać nasz szacowany stan zdrowia maszyny z konkretnym komponentem, pomagając w procedurach diagnostycznych w celu zidentyfikowania źródła naszych problemów.

Musi istnieć odpowiedni i wykonalny przypadek użycia. Największy problem związany z można konserwacją predykcyjna przypisać danym, z którymi musimy pracować. Wiele razy pracujemy w reżimie "big data". Big data zwykle charakteryzuje się różną: objętością, różnorodnością, prędkością, prawdziwością i wartością. W jaki sposób mogą one być powiązane z danymi, które prawdopodobnie napotkamy w przypadku użycia konserwacji predykcyjnej:

Objętość

Zwykle mamy do czynienia z dużą ilością danych. Każdy element wyposażenia może mieć dziesiątki czujników, z których każdy stale monitoruje maszynę przez cały okres jej użytkowania.

Różnorodność

Czujniki mogą rejestrować dane w różnych formatach lub w różnych skalach czasowych. To tylko dla jednego elementu wyposażenia. Możemy mieć do czynienia z różnymi modelami lub różnymi typami maszyn w większym zakładzie.

Prędkość

Dane mogą być generowane szybko, co prowadzi do problemów z tym, jak wyodrębniamy sygnały lub agregujemy te odczyty w czasie. Być może będziemy musieli również analizować te dane w czasie rzeczywistym.

Prawdziwość

Dane często mogą być wątpliwej jakości, a niektórych danych może brakować. Dane z czujników są często hałaśliwe, a same czujniki mogą łatwo ulec uszkodzeniu.

Wartość

W danych mogą znajdować się przydatne informacje, które można wyodrębnić, aby pomóc w podejmowaniu decyzji.

Poza tymi wyzwaniami związanymi z dużymi zbiorami danych mogą pojawić się dodatkowe wyzwania. Być może mamy do czynienia z danymi o wysokiej wymiarowości ze złożonymi korelacjami między zmiennymi. Rozumowanie w tych wysokowymiarowych przestrzeniach jest trudne, a odkrycie tych korelacji może być prawie niemożliwe dla człowieka. W tym właśnie może pomóc uczenie maszynowe.

4. Dane do analizy

Dane pobieraliśmy ze strony www.backblaze.com Backblaze to duży dostawca usług przechowywania danych z ponad 130 000 dysków twardych w swoich centrach danych. Dyski twarde mają wiele czujników, które stale monitorują i raportują stan dysku, znane jako SMART (Self-Monitoring, Analysis and Reporting Technology). Umiejętność zrozumienia i przewidywania, kiedy dysk twardy ulegnie awarii, jest niezwykle cenna dla działania takiego centrum danych, ponieważ oferuje możliwość uniknięcia przestojów dzięki lepszemu terminowi wymiany i naprawy sprzętu. Każdy ze 130 000 dysków twardych Backblaze jest monitorowany codziennie, a jego aktywność jest rejestrowana jako pojedynczy wiersz w codziennym pliku danych. Każdego dnia, dla każdego dysku twardego, Backblaze rejestruje kilka dziennych zagregowanych wskaźników SMART, a także czy nie zawiódł w tym dniu. Każdy atrybut SMART ma wartość surową, której znaczenie jest określane w całości przez producenta napędu (ale często odpowiada mierzonej surowej jednostce fizycznej), a także znormalizowaną wartość od 1 do 253, gdzie 1 jest najgorsza, a 100 jest zwykle wartością początkową. Chociaż większość metryk w zestawie danych pokazuje podobny wzorzec do 197 z wyraźnym odwzorowaniem między wartościami, istnieje wiele metryk, które, podobnie jak metryka 7, wykazują niewielką korelację między wartościami. W rezultacie w naszej analizie zdecydujemy się zachować zarówno surowe, jak i znormalizowane wartości w przypadku, gdy posiadanie tych różnych sygnałów spowoduje większą moc przewidywania .

Oto opisy odpowiednich wskaźników SMART:

5 - Liczba ponownie przydzielonych sektorów - Liczba uszkodzonych sektorów, które zostały znalezione i ponownie przydzielone. Dysk twardy, który został ponownie przydzielony, najprawdopodobniej ulegnie awarii w ciągu najbliższych miesięcy.

187- Zgłoszone błędy nienaprawialne - Liczba błędów, których nie można było odzyskać przy użyciu sprzętowego ecc (Error-Correcting Code), typu pamięci używanej do korygowania błędów uszkodzenia danych.

188- Limit czasu poleceń - Liczba przerwanych operacji z powodu przekroczenia limitu czasu dysku twardego.

197- Liczba bieżących oczekujących sektorów - Liczba uszkodzonych sektorów, które zostały znalezione i czekają na ponowną alokację z powodu nieodwracalnych błędów odczytu.

198- Liczba nienaprawialnych sektorów w trybie offline - Całkowita liczba nienaprawialnych błędów podczas odczytu/zapisu sektora, wskazujących na defekty powierzchni dysku lub problemy w podsystemie mechanicznym.

5. Realizacja

Projekt realizujemy w środowisku R w wersji: RStudio Desktop 2022.02.2+485, korzystamy także z Excela w wersji Microsoft Excel 2019.

Pierwszym krokiem jest pobranie wszystkich danych. Pobraliśmy je ze strony www.backblaze.com. Zliczyliśmy ilość wszystkich modeli oraz wybraliśmy ten który występuje najczęściej by liczba danych była największa. Model, który występuje najczęściej to ST12000NM0007, występował on 37002 razy w pliku „2020-01-03” dlatego dalsza analiza będzie dotyczyła właśnie tego modelu dysku.

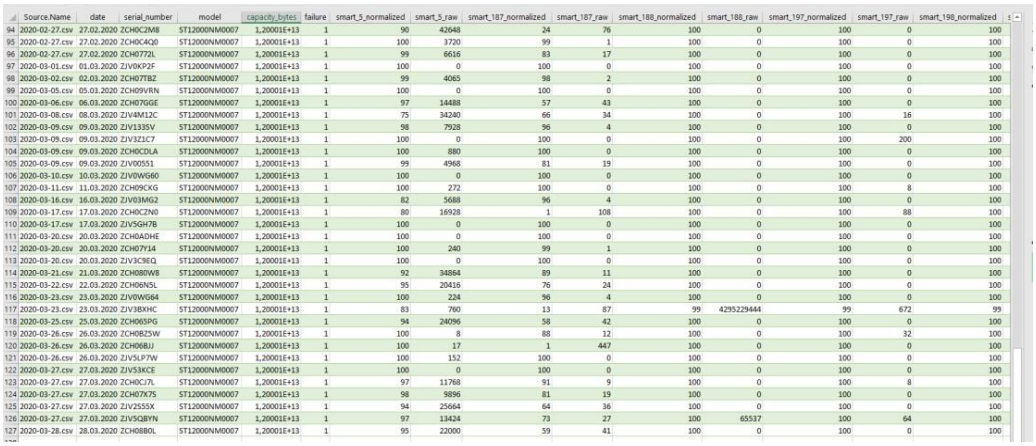
Używając narzędzia power query w exelu posortowaliśmy wybrane dane wybierając tylko interesujące, te kolumny, które przydadzą się do dalszej analizy (kolumny zostały wybrane na podstawie artykułu ze strony blackbaze: https://www.backblaze.com/blog/what-smart-statsindicate-hard-drive-failures/), tylko wadliwe dyski.

Zapytania

	Source.Name	date	serial_number	model	capacity_bytes	failure	smart_5_normalized	smart_5_raw	smart_187_normalized	smart_187_raw	smart_1
1	2020-01-03.csv	03.01.2020	ZCH07WFE	ST12000NM0007	1,20001E+13	1	97	14360	4	96	
2	2020-01-03.csv	03.01.2020	ZCH083CK	ST12000NM0007	1,20001E+13	1	100	400	100	0	
3	2020-01-03.csv	03.01.2020	ZCH0663M	ST12000NM0007	1,20001E+13	1	100	33	100	0	
4	2020-01-03.csv	03.01.2020	ZCH07A7E5	ST12000NM0007	1,20001E+13	1	98	9448	91	9	
5	2020-01-04.csv	04.01.2020	ZCH0BMRV	ST12000NM0007	1,20001E+13	1	95	20024	71	29	
6	2020-01-04.csv	04.01.2020	ZCH0AAZ1	ST12000NM0007	1,20001E+13	1	100	3888	86	14	
7	2020-01-05.csv	05.01.2020	ZJV4YWS0	ST12000NM0007	1,20001E+13	1	100	304	99	1	
8	2020-01-06.csv	06.01.2020	ZCH096PL	ST12000NM0007	1,20001E+13	1	97	14584	86	14	
9	2020-01-06.csv	06.01.2020	ZJV2DZ4X	ST12000NM0007	1,20001E+13	1	100	272	95	5	
10	2020-01-06.csv	06.01.2020	ZCH071PX	ST12000NM0007	1,20001E+13	1	98	10512	89	11	
11	2020-01-07.csv	07.01.2020	ZCH0849V	ST12000NM0007	1,20001E+13	1	100	3512	95	5	
12	2020-01-07.csv	07.01.2020	ZCH07WAB	ST12000NM0007	1,20001E+13	1	100	1	100	0	
13	2020-01-08.csv	08.01.2020	ZCH0A88Z	ST12000NM0007	1,20001E+13	1	100	1424	100	0	
14	2020-01-08.csv	08.01.2020	ZJV00BYW	ST12000NM0007	1,20001E+13	1	100	0	100	0	
15	2020-01-09.csv	09.01.2020	ZCH07QC8	ST12000NM0007	1,20001E+13	1	96	17656	82	18	
16	2020-01-10.csv	10.01.2020	ZJV02SNQ	ST12000NM0007	1,20001E+13	1	100	0	99	1	
17	2020-01-10.csv	10.01.2020	ZJV1K4H0	ST12000NM0007	1,20001E+13	1	100	112	100	0	
18	2020-01-10.csv	10.01.2020	ZJV0WF1D	ST12000NM0007	1,20001E+13	1	100	2520	96	4	
19	2020-01-11.csv	11.01.2020	ZCH0C1G4	ST12000NM0007	1,20001E+13	1	100	728	99	1	
20	2020-01-12.csv	12.01.2020	ZJV2E7XA	ST12000NM0007	1,20001E+13	1	100	0	100	0	
21	2020-01-12.csv	12.01.2020	ZJV2EH03	ST12000NM0007	1,20001E+13	1	100	48	100	0	
22	2020-01-13.csv	13.01.2020	ZCH08YK6	ST12000NM0007	1,20001E+13	1	100	0	100	0	
23	2020-01-13.csv	13.01.2020	ZJV03CVD	ST12000NM0007	1,20001E+13	1	100	32	100	0	
24	2020-01-13.csv	13.01.2020	ZJV03KMC	ST12000NM0007	1,20001E+13	1	100	1	100	0	
25	2020-01-13.csv	13.01.2020	ZCH02YZ5	ST12000NM0007	1,20001E+13	1	100	616	96	4	
26	2020-01-14.csv	14.01.2020	ZCH0962K	ST12000NM0007	1,20001E+13	1	100	3848	97	3	
27	2020-01-15.csv	15.01.2020	ZCH0D3J3	ST12000NM0007	1,20001E+13	1	92	34632	72	28	
28	2020-01-16.csv	16.01.2020	ZJV03N8Y	ST12000NM0007	1,20001E+13	1	97	15312	89	11	
29	2020-01-16.csv	16.01.2020	ZCH08XTV	ST12000NM0007	1,20001E+13	1	95	21152	83	17	
30	2020-01-17.csv	17.01.2020	ZCH076JB	ST12000NM0007	1,20001E+13	1	99	4240	100	0	

LICZBA KOLUMN: 16, LICZBA WIERSZY: 126

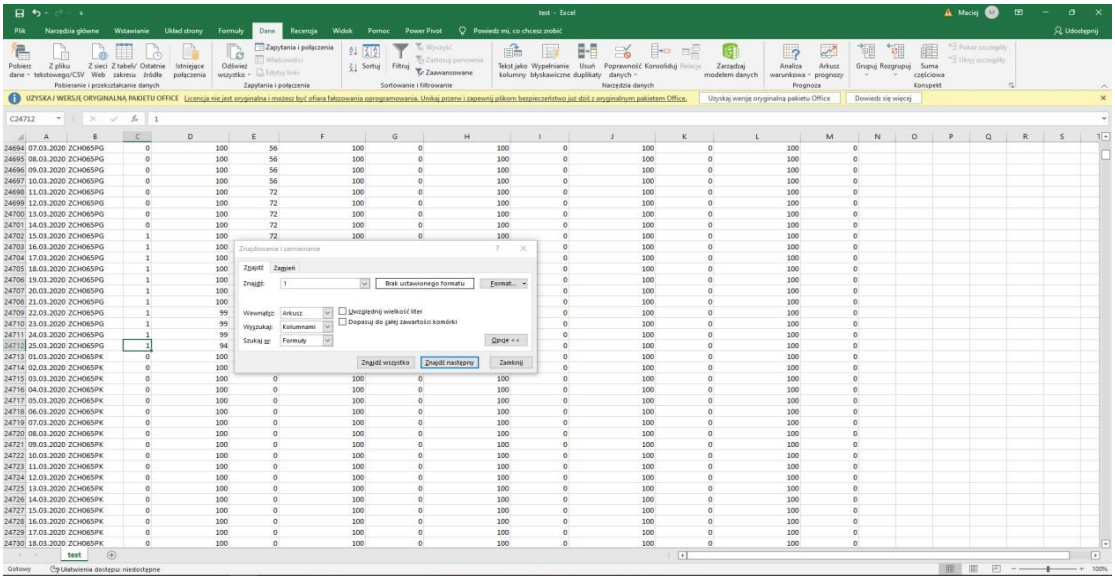
Po posortowaniu pierwszego kwartału roku 2020 uzyskaliśmy 127 dysków które uległy awarii. Przez ograniczenia sprzętowe dalsza analiza nie dotyczy wszystkich rejestrów o dysku ST12000NM0007(było by ich ponad 3mln wierszy; limit wierszy w pliku Excela to lekko ponad milion, dlatego trzeba było by tworzyć 3 pliki, gdzie i tak 99.9% danych to dane o dyskach działających). Aby rozwiązać ten problem wzięliśmy wadliwe dyski z całego kwartału i dorzuciliśmy je do pliku z pojedynczego dnia co dało nam i tak 37tys. wierszy które posłużą do nauki modelu. (Wadliwe dyski to wciąż tylko 0,35%, ten problem rozwiążemy dalej w kodzie.)



	Source.Name	date	serial_number	model	capacity_bytes	failure	smart_5_normalized	smart_5_raw	smart_187_normalized	smart_187_raw	smart_188_normalized	smart_188_raw	smart_197_normalized	smart_197_raw	smart_198_normalized
94	2020-02-27.csv	27.02.2020	ZCH0C2M8	ST12000NM0007	1,20001E+13	1	90	42648	24	76	100	0	100	0	100
95	2020-02-27.csv	27.02.2020	ZCH0C4Q0	ST12000NM0007	1,20001E+13	1	100	3720	99	1	100	0	100	0	100
96	2020-02-27.csv	27.02.2020	ZCH0772L	ST12000NM0007	1,20001E+13	1	99	6616	83	17	100	0	100	0	100
97	2020-03-01.csv	01.03.2020	ZJV0KP2F	ST12000NM0007	1,20001E+13	1	100	0	100	0	100	0	100	0	100
98	2020-03-02.csv	02.03.2020	ZCH07TBZ	ST12000NM0007	1,20001E+13	1	99	4065	98	2	100	0	100	0	100
99	2020-03-05.csv	05.03.2020	ZCH09VRN	ST12000NM0007	1,20001E+13	1	100	0	100	0	100	0	100	0	100
100	2020-03-06.csv	06.03.2020	ZCH07GGE	ST12000NM0007	1,20001E+13	1	97	14488	57	43	100	0	100	0	100
101	2020-03-08.csv	08.03.2020	ZJV4M12C	ST12000NM0007	1,20001E+13	1	75	34240	66	34	100	0	100	16	100
102	2020-03-09.csv	09.03.2020	ZJV133SV	ST12000NM0007	1,20001E+13	1	98	7928	96	4	100	0	100	0	100
103	2020-03-09.csv	09.03.2020	ZJV321C7	ST12000NM0007	1,20001E+13	1	100	0	100	0	100	0	100	200	100
104	2020-03-09.csv	09.03.2020	ZCH0CDLA	ST12000NM0007	1,20001E+13	1	100	880	100	0	100	0	100	0	100
105	2020-03-09.csv	09.03.2020	ZJV00551	ST12000NM0007	1,20001E+13	1	99	4968	81	19	100	0	100	0	100
106	2020-03-10.csv	10.03.2020	ZJV0WG60	ST12000NM0007	1,20001E+13	1	100	0	100	0	100	0	100	0	100
107	2020-03-11.csv	11.03.2020	ZCH09CKG	ST12000NM0007	1,20001E+13	1	100	272	100	0	100	0	100	8	100
108	2020-03-16.csv	16.03.2020	ZJV03MG2	ST12000NM0007	1,20001E+13	1	82	5688	96	4	100	0	100	0	100
109	2020-03-17.csv	17.03.2020	ZCH0CZN0	ST12000NM0007	1,20001E+13	1	80	16928	1	108	100	0	100	88	100
110	2020-03-17.csv	17.03.2020	ZJV5GH7B	ST12000NM0007	1,20001E+13	1	100	0	100	0	100	0	100	0	100
111	2020-03-20.csv	20.03.2020	ZCH0ADHE	ST12000NM0007	1,20001E+13	1	100	0	100	0	100	0	100	0	100
112	2020-03-20.csv	20.03.2020	ZCH07Y14	ST12000NM0007	1,20001E+13	1	100	240	99	1	100	0	100	0	100
113	2020-03-20.csv	20.03.2020	ZJV3C9EQ	ST12000NM0007	1,20001E+13	1	100	0	100	0	100	0	100	0	100
114	2020-03-21.csv	21.03.2020	ZCH080W8	ST12000NM0007	1,20001E+13	1	92	34864	89	11	100	0	100	0	100
115	2020-03-22.csv	22.03.2020	ZCH06N5L	ST12000NM0007	1,20001E+13	1	95	20416	76	24	100	0	100	0	100
116	2020-03-23.csv	23.03.2020	ZJV0WG64	ST12000NM0007	1,20001E+13	1	100	224	96	4	100	0	100	0	100
117	2020-03-23.csv	23.03.2020	ZJV3BXHC	ST12000NM0007	1,20001E+13	1	83	760	13	87	99	4295229444	99	672	99
118	2020-03-25.csv	25.03.2020	ZCH065PG	ST12000NM0007	1,20001E+13	1	94	24096	58	42	100	0	100	0	100
119	2020-03-26.csv	26.03.2020	ZCH0BZSW	ST12000NM0007	1,20001E+13	1	100	8	88	12	100	0	100	32	100
120	2020-03-26.csv	26.03.2020	ZCH06BJJ	ST12000NM0007	1,20001E+13	1	100	17	1	447	100	0	100	0	100
121	2020-03-26.csv	26.03.2020	ZJV5LP7W	ST12000NM0007	1,20001E+13	1	100	152	100	0	100	0	100	0	100
122	2020-03-27.csv	27.03.2020	ZJV53KCE	ST12000NM0007	1,20001E+13	1	100	0	100	0	100	0	100	0	100
123	2020-03-27.csv	27.03.2020	ZCH0CJ7L	ST12000NM0007	1,20001E+13	1	97	11768	91	9	100	0	100	8	100
124	2020-03-27.csv	27.03.2020	ZCH07X7S	ST12000NM0007	1,20001E+13	1	98	9896	81	19	100	0	100	0	100
125	2020-03-27.csv	27.03.2020	ZJV2555X	ST12000NM0007	1,20001E+13	1	94	25664	64	36	100	0	100	0	100
126	2020-03-27.csv	27.03.2020	ZJV5QBYN	ST12000NM0007	1,20001E+13	1	97	13424	73	27	100	65537	100	64	100
127	2020-03-28.csv	28.03.2020	ZCH08B0L	ST12000NM0007	1,20001E+13	1	95	22000	59	41	100	0	100	0	100

Rysunek 1

Jako że nie mogłem znaleźć żadnej funkcji, która automatycznie wpisywała by w miejsce zera jedynkę dziesięć dni wstecz po znalezieniu jedynki, zrobiłem to ręcznie sortując dyski alfabetycznie a później wyszukując tych zepsutych i wpisując im w poprzedzające dni jedynkę. Dzięki temu algorytm będzie w stanie wykrywać zmiany, w parametrach które powodują awarię.



Rysunek 2

Kolejnym krokiem będzie nauczenie algorytmu by rozpoznawał, kiedy dysk może ulec uszkodzeniu.

Najpierw czytujemy niezbędne biblioteki:

install.packages('xgboost')
Tworzy modele o wysokiej dokładności predykcyjnej.

install.packages('caret')
Caret Package to kompleksowa platforma do budowania modeli uczenia maszynowego w języku.

install.packages('mltools')
Przyspiesza przygotowywanie danych do zasilania modeli uczenia maszynowego, identyfikuje struktury i wzorce w zestawie danych, ocenia wyniki modelu uczenia maszynowego.

install.packages('data.table') Pakiet służący do pracy z danymi tabelarycznymi.

Kolejnym krokiem będzie ustawienie typu danych. Wartości smart muszą być zapisane jako wartości numeryczne, natomiast kolumna failure musi być typu factor ponieważ, algorytm XGBoost wymaga tego, aby móc w prawidłowy sposób trenować model.

Jako że różnica między ilościom danych o dyskach wadliwych i działających nie jest równomierna (rekordów o dyskach działających mamy 36994 a wadliwych jedynie 302) użyjemy funkcji upSample która wyrównuje ilość danych, poprzez dopisanie rekordów o podobnych wartościach. Robimy to, aby algorytm XGBoost posiadał równomierną ilość danych do nauki.

```
> table(data_train$failure)

    0     1
36994   302
```

Rysunek 3 Ilość dysków wadliwych i działających przed przekształceniem

```
> train_up_sample<-upSample(x=data_train[,-1],y=data_train$failure)
```

Rysunek 4 Funkcja dodająca sztuczne rekordy

```
> table(train_up_sample2$class)

    0     1
36994 36994
```

Rysunek 5 Ilość dysków wadliwych i działających po przekształceniu

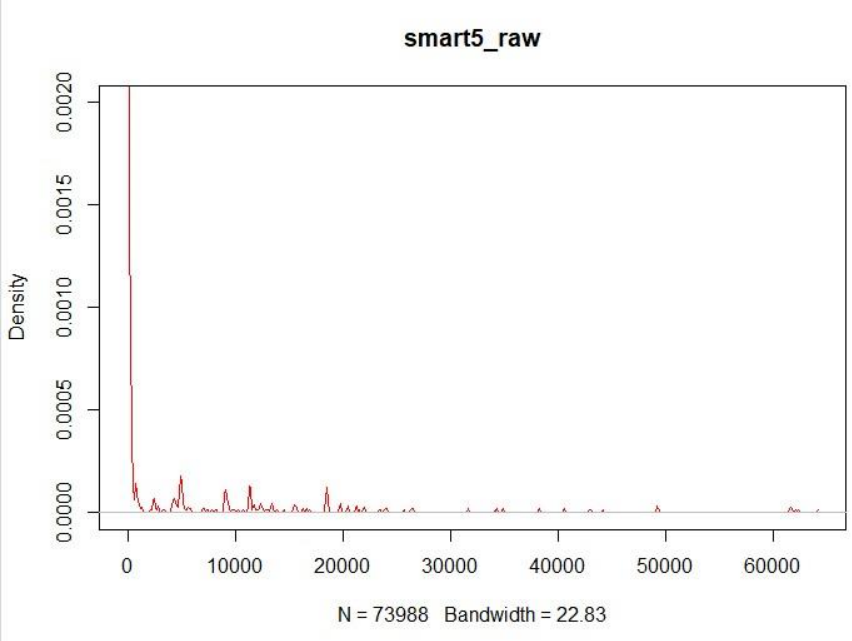
Użycie komendy upSample zmienia również nazwę kolumny „failure „na „Class" i przerzuca ją na koniec. Przez to, gdy chcemy sprawdzić działanie algorytmu dla danych testowych musimy pamiętać, o przerzuceniu kolumny „failure" na koniec, by zapewnić odpowiedni odczyt.

W następnym kroku rysujemy wykresy obrazujące zależności:

```
#wykresy
plot(density(train_up_sample2$smart_5_raw), main = "smart5_raw", type="l", col="red",ylim=c(0,0.002))
plot(density(train_up_sample2$smart_197_raw), main = "smart197_raw", type="l", col="red",ylim=c(0,0.1))
```

Funkcja density - gęstości prawdopodobieństwa opisuje prawdopodobieństwo przyjęcia przez zmienną określonej wartości.

Wykres nr 1:



Rysunek 6 Wykres parametru smart 5_raw

Z wykresu możemy odczytać, że większość dysków przyjmuje wartości smart 5 równe zeru. Jako, że większość dysków jest działająca, (co wiemy z Rysunek 3) możemy spekulować, że właśnie one przyjmują wartość 0.

Po upewnieniu się, że liczba rekordów i typ danych się zgadza, możemy przejść do ustawiania parametrów algorytmu.

```
grid_tune <- expand.grid(
  nrounds = c(500,1000,1500), #liczba drzew
  max_depth = c(2,4,6), #zwiekszanie tego wplywa na zwiekszenie dokladnosci i zuzycia pamieci
  eta = 0.3, #c(0.025,0.05,0.1,0.3), #tempo nauki
  gamma = 0, # wplywa na to jak duzy wplyw ma pojedyncza probka treningowa
  colsample_bytree = 1, # c(0.4, 0.6, 0.8, 1.0) próbka kolumn dla drzewa
  min_child_weight = 1, # c(1,2,3) # odpawiada za minimalna liczbe wystapien w kazdym wezle
  subsample = 1 # c(0.5, 0.75, 1.0) # uzywa sie by nie przeszlo do przesadnego dopasowania modelu
)
```

Rysunek 7 Parametry użyte do nauki modelu

```
train_control <- trainControl(method = "cv",
                              number=3,
                              verboseIter = TRUE,
                              allowParallel = TRUE)
```

Rysunek 8

Dla funkcji sprawdzającej proces nauki ustawiamy metodę sprawdzianu krzyżowego z podziałem na 3 zbiory z możliwością równoległego sprawdzania. Oznacza to mniej więcej tyle, że dzielimy funkcję na 3 zbiory. Na części z nich przeprowadzamy analizę a resztę pozostawiamy do sprawdzenia wyników tych analiz.

Po Ustawieniu parametrów możemy zająć się trenowaniem modelu. Do nauki użyjemy metody drzewa. XGBoost to algorytm uczenia maszynowego, który służy do implementacji drzew decyzyjnych zwiększających gradient. XGBoost został zasadniczo zaprojektowany w celu znacznej poprawy szybkości i wydajności modeli uczenia maszynowego. Posiada liniowe uczenie się modelu, dzięki czemu jest w stanie wykonywać równoległe obliczenia na jednej maszynie. Używa metod drzewa przy użyciu architektury opadania gradientu. Algorytm XGBoost wykorzystuje podejście oparte na głębokości, używa także funkcji/ parametru maksymalnej głębokości, a zatem przycina drzewo w kierunku wstecznym. Proces sekwencyjnego budowania drzewa odbywa się za pomocą równoległej implementacji w algorytmie XGBoost. Jest to możliwe dzięki zewnętrznym i wewnętrznym pętlom, które są wymienne. Zewnętrzna pętla zawiera listę węzłów liści drzewa, podczas gdy pętla wewnętrzna obliczy cechy. Ponadto, aby pętla zewnętrzna mogła się rozpocząć, pętla wewnętrzna musi zostać ukończona. Ten proces przełączania poprawia wydajność algorytmu.

XGBoost jest znany z tego, że bardzo skutecznie radzi sobie z różnymi rodzajami wzorców rzadkości. Algorytm ten uczy się brakującej wartości gniazda, widząc stratę treningu. Ma wbudowane funkcje sprawdzania poprawności krzyżowej, które są implementowane w każdej iteracji podczas tworzenia modelu.

Zapobiega to konieczności obliczania liczby potrzebnych iteracji zwiększających. Wykorzystuje rozproszony ważony szkic kwantylowy, aby uzyskać optymalną liczbę punktów podziału między ważonymi zestawami danych. Główne obsługiwane metody zwiększania gradientu to - szybkość wykonywania (kiedy porównamy XGBoost z innymi algorytmami zwiększającymi gradient, okazuje się naprawdę szybki, około 10 razy szybszy niż inne implementacje).Wykorzystuje algorytm drzewa decyzyjnego zwiększającego gradient, metoda zwiększania gradientu tworzy nowe modele, które wykonują zadanie przewidywania błędów i pozostałości wszystkich poprzednich modeli, które następnie są sumowane, a następnie dokonywane jest ostateczne przewidywanie.

```
xgb_tune <- train(x = train_up_sample[,-11],
                  y = train_up_sample[,11],
                  trControl = train_control,
                  tuneGrid = grid_tune,
                  method= "xgbTree",
                  verbose = TRUE)
```

Rysunek 9 Trenujemy model

Dla funkcji x przypisujemy wszystkie wartości smart, funkcja y to wartości „failure”. Funkcja kontrolująca przebieg trenowania opisana jest na Rysunek 8 natomiast funkcja ustalająca parametry na Rysunek 8Rysunek 6.

Po przeanalizowaniu danych algorytm zwraca nam jaką dokładność odniósł w zależności od zadanych parametrów.

max_depth	nrounds	Accuracy	Kappa
2	500	0.9063271	0.8125400
2	1000	0.9071275	0.8141407
2	1500	0.9075209	0.8149275
4	500	0.9080364	0.8159584
4	1000	0.9081450	0.8161755
4	1500	0.9081857	0.8162569
6	500	0.9082399	0.8163654
6	1000	0.9082671	0.8164197
6	1500	0.9082671	0.8164197

Rysunek 10 Dokładność w zależności od parametrów

Z tych wartości wybieramy najlepsze parametry, a następnie korzystamy tylko z nich**.**

```
final_grid <- expand.grid(nrounds = xgb_tune$bestTune$nrounds,
                          eta = xgb_tune$bestTune$eta,
                          max_depth = xgb_tune$bestTune$max_depth,
                          gamma = xgb_tune$bestTune$gamma,
                          colsample_bytree = xgb_tune$bestTune$colsample_bytree,
                          min_child_weight = xgb_tune$bestTune$min_child_weight,
                          subsample = xgb_tune$bestTune$subsample)
```

Rysunek 11 Ustawiamy optymalne parametry

Następnie sprawdzamy, czy algorytm wykryje awarię z wyprzedzeniem dla danych testowych (w plikach, gdzie znajdują się dane

1 oznacza awarię dysku, w celu przewidywania 10 poprzednich wyników, które zwrócił dysk zapisujemy jako 1) oraz generujemy macierz:

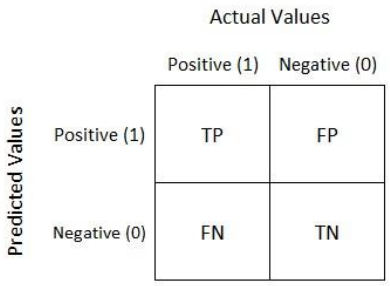
```
> confusionMatrix(as.factor(as.numeric(xgb.pred1)),
+                 as.factor(as.numeric(data_test$failure)))
Confusion Matrix and Statistics

          Reference
Prediction     1     2
         1 36003   110
         2   324    32

               Accuracy : 0.9881
                 95% CI : (0.9869, 0.9892)
    No Information Rate : 0.9961
    P-Value [Acc > NIR] : 1
```

Rysunek 12 Tworzymy tablice pomyłek dla danych testowych

Z tablicy odczytujemy, że algorytm 36008 razy odczytał działający poprawnie (było 0 przeczytał 0), 35 razy odczytał niedziałający poprawnie (było 1, przeczytał 1), 107 odczytał, dla zepsutego dysku, że działa (było 1 przeczytał 0), oraz 319 razy błędnie odczytał niedziałający (było 0 przeczytał 1). Ostatnie odczytanie wiąże się z tym, że algorytm przewidywał, że dla zadanych wartości dysk może ulec awarii i właśnie o to nam chodziło.
Poprawne przeanalizowanie tabeli umożliwiła nam poniższa macierz:



Rysunek 13 sposób odczytywania tabeli pomyłek

A teraz sprawdzamy, czy algorytm działa na przykładzie konkretnego dysku (ZCH08034), który wiemy, że uległ uszkodzeniu.

```
> predict(xgb_model2, data_test2)
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1
[68] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Levels: 0 1
>
>
> confusionMatrix(as.factor(as.numeric(xgb.pred2)),
+                 as.factor(as.numeric(data_test2$failure2)))
Confusion Matrix and Statistics

          Reference
Prediction  1  2
         1 58  0
         2 26  1
```

Rysunek 14 Wyniki dla dysku ZCH08034

Jak widzimy algorytm już dużo wcześniej pokazuje nam dysk jako wadliwy, mimo że w rzeczywistości jeszcze działa. W tym przypadku jest to widzimy około 40 dni przed awarią algorytm zwrócił nam 2 jedynki (już wtedy stwierdził możliwość awarii) następne przez następne 10 dni wskazywał 0 po czym przez dwadzieścia dwa dni znów wskazywał, że dysk może ulec awarii co faktycznie się zdarzyło.

Dla kolejnego dysku (ZCH0CJ7L) algorytm również wykrył z dużym wyprzedzeniem zachodzące anomalie i że może ulec uszkodzeniu. Jak widzimy z tablicy pomyłek algorytm 33 razy przewidywał uszkodzenie dysku, gdy ten jeszcze funkcjonował.

```
> predict(xgb_model2, data_test2)
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[59] 1
Levels: 0 1
> confusionMatrix(as.factor(as.numeric(xgb.pred2)),
+                 as.factor(as.numeric(data_test2$failure2)))
Confusion Matrix and Statistics

          Reference
Prediction  1  2
         1 25  0
         2 33  1
```

Rysunek 15 Wyniki dla dysku ZCH0CJ7L

Wnioski:

Jak widzimy na przykładzie dwóch dysków algorytm jest zbyt czuły w przewidywaniu i robi to zdecydowanie za często i za wcześnie co w rzeczywistości przełożyło by się na wiele fałszywych alarmów co wiążę się z kosztami i stratami dla firmy, która potencjalnie chciałaby korzystać z takiego rozwiązania. Aby rozwiązać ten problem konieczna była by dalsza praca nad danymi lub samym algorytmem, co przełożyłoby się na lepszą skuteczność i ostatecznie wpłynęło by na to, że takie rozwiązanie stało by się opłacalne.

Źródła:

1. Rodzaje uszkodzeń dysków twardych - ZRecovery Odzyskiwanie danych
2. Jak przewidzieć awarię dysku twardego ? - BackUp Academy
3. Backblaze Hard Drive Stats
4. SMART Drive and Failure Rates (backblaze.com)
5. Machine Learning for Predictive Maintenance - Part 1 | Datatonic : Datatonic
6. Machine Learning for Predictive Maintenance - Part 2: Predicting Hard Drive Failure | Datatonic : Datatonic
7. Understanding and Applying XGBoost Classification Trees in R - YouTube
8. XGBoost Parameters