



**WYDZIAŁ
ELEKTROTECHNIKI
I INFORMATYKI**
POLITECHNIKI RZESZOWSKIEJ

inż. Krzysztof Lang

Implementacja wybranych algorytmów wypełniania
brakujących wartości, dla strumieni dużych zbiorów danych

Praca dyplomowa magisterska

Opiekun pracy:
dr Michał Piętał

Rzeszów, 2023

Spis treści

1. Wstęp	6
2. Wprowadzenie	7
2.1. Rys historyczny	7
2.2. Idea wypełniania brakujących wartości	7
2.3. Korzyści i zagrożenia	7
2.4. Perspektywy na przyszłość	7
3. Omówienie narzędzi i danych	8
3.1. Python	8
3.2. Visual Studio Code	8
3.3. GitHub	9
3.4. Biblioteki	9
3.4.1. Pandas	9
3.4.2. NumPy	10
3.4.3. SciKit-learn	10
3.4.4. EasyGUI	10
3.5. Źródła danych	11
3.5.1. Użyte repozytoria danych	11
3.5.2. Adult Data Set	11
3.5.3. Stock Exchange Data	12
4. Implementacja i testy	14
4.1. Opis przygotowanego programu	14
4.1.1. Założenia i realizacja	14
4.1.2. Działanie programu	15
4.2. Opis implementacji algorytmów	22
4.2.1. Algorytm pierwszy	23
4.2.2. Algorytm drugi	25
4.2.3. Algorytm trzeci	30
4.3. Napotkane problemy	34
4.4. Testy algorytmów na wybranych źródłach danych	34
5. Podsumowanie i wnioski końcowe	35
Załączniki	36

Literatura	37
-----------------------------	-----------

1. Wstęp

W pracy opisano znaczenie wypełniania brakujących wartości w zbiorach danych. Przedstawiono 3 algorytmy służące wypełnianiu brakujących danych, które następnie zostały zaimplementowane z użyciem stworzonej na te potrzeby aplikacji. Dokonano porównania dokładności wypełniania danych dla poszczególnych algorytmów i przeanalizowano uzyskane wyniki. Do przygotowania aplikacji wykorzystano język programowania Python [1], z wykorzystaniem bibliotek służących do analiz i manipulacji danymi: Pandas [2], NumPy [3] i SciKit-learn oraz biblioteki udostępniającej narzędzia do przygotowania interfejsu graficznego EasyGUI [5]. Dane wykorzystane do testów pochodzą z repozytorium danych do uczenia maszynowego Uniwersytetu Kalifornijskiego w Irvine [6] oraz zbioru bazy danych udostępnionych w serwisie Kaggle.com [7].

W pierwszym rozdziale przedstawiono zarys treści dalszej pracy. W drugim rozdziale przybliżono ideę wypełniania brakujących wartości w dużych zbiorach danych, na co składa się rys historyczny zagadnienia, przedstawienie istniejących rozwiązań i ich znaczenie, oraz zastanowiono się nad przyszłością zagadnienia. Trzeci rozdział przybliża narzędzia użyte do przygotowania części praktycznej niniejszej pracy, to jest programu realizującego testy algorytmów wypełniających puste miejsca oraz dane wykorzystane do testów. Na rozdział czwarty składa się opis praktycznej części badań. Znajduje się w nim opis przygotowanego programu wraz z przykładem działania, opisy zaimplementowanych algorytmów, oraz przedstawione zostały wyniki przeprowadzonych testów. Rozdział piąty składa się z podsumowania przeprowadzonych analiz i przedstawienia wyciągniętych wniosków.

2. Wprowadzenie

2.1. Rys historyczny

2.2. Idea wypełniania brakujących wartości

2.3. Korzyści i zagrożenia

2.4. Perspektywy na przyszłość

3. Omówienie narzędzi i danych

3.1. Python

Python to interpretowany język wysokiego poziomu o ogólnym zastosowaniu. Stworzony został w roku 1991 przez Guido van Rossuma. Został zaprojektowany jako następca języka ABC [8], posiadający obsługę wyjątków i działający na systemie operacyjnym Amoeba. [9] Python jest wciąż wciąż rozwijany, najnowsza stabilna wersja to 3.11.2.

Python znany jest ze swojej prostej do zrozumienia składni, czytelnego kodu, wsparcia różnych paradygmatów programowania (obiekтового, imperatywnego, funkcyjnego), dużej biblioteki standardowej i bardzo szerokiego wyboru dodatkowych bibliotek. [10]

Do słabszych stron języka zaliczyć można słabą wydajność pod względem prędkości działania i wykorzystania pamięci bądź problemy z kompatybilnością kodu napisanego w różnych wersjach lub implementacjach Pythona. [10]

3.2. Visual Studio Code

Zintegrowane środowisko programistyczne, w skrócie IDE (od ang. integrated development environment) to aplikacja dostarczająca komplet narzędzi potrzebnych programistom w celu tworzenia aplikacji. Zazwyczaj IDE składa się z minimum: edytora kodu, kompilatora i debuggera. Niektóre środowiska oferują dodatkowe funkcje takie jak automatyczne uzupełnianie kodu, podświetlanie składni, automatyczne formatowanie kodu, narzędzia do analizy kodu czy narzędzia kontroli wersji. Dobór odpowiedniego środowiska programowania jest istotny, ponieważ może znacznie zwiększyć produktywność programisty. [11]

Jako środowisko programowania wykorzystane do przygotowania programu do testowania algorytmów wybrano Microsoft Visual Studio Code. [12] Jest to darmowy IDE o otwartym kodzie źródłowym. Stworzony został przez firmę Microsoft w roku 2015. Zaprojektowany został z myślą tworzenia aplikacji webowych i chmurowych z wykorzystaniem wielu języków. Ze względu na otwartość kodu, dostępne jest wiele rozszerzeń do programu, które znacznie ułatwiają tworzenie także projektów o dużym stopniu złożoności. [13]

3.3. GitHub

W celu umożliwienia pracy nad programem z wielu urządzeń oraz dla zachowania pełnej historii tworzenia programu wykorzystano integrację Visual Studio Code z repozytorium GitHub, tworząc prywatne repozytorium kodu. [14]

GitHub to platforma umożliwiająca programistom współpracę nad projektami, udostępnianie kodu i udzielanie się w projektach o otwartym kodzie źródłowym. Jest to największa tego typu platforma na świecie. [15]

Jednym z najważniejszych elementów GitHub jest Git. Jest to rozproszony system kontrol wersji, przygotowany przez Linusa Torvaldsa i wydany w roku 2005. Oryginalnie służył do pomocy przy rozwoju jądra Linux. Pozwala na śledzenie zmian w kodzie oraz pracę nad wieloma gałęziami projektu w tym samym czasie. [16]

3.4. Biblioteki

3.4.1. Pandas

Pandas to popularna biblioteka służąca manipulacji i analizie danych w Pythonie. Stworzona została w roku 2008 przez Wesa McKinney'a i jest stale rozwijana przez grupę deweloperów. Nazwa Pandas pochodzi od "Python data analysis". [2] Najważniejsze zalety Pandas to:

- wsparcie dla wielu formatów danych, w tym CSV, Excel, SQL i inne,
- duża szybkość przeprowadzania operacji na danych,
- obsługa brakujących wartości w danych,
- łatwa integracja z innymi popularnymi bibliotekami,
- bardzo rozbudowana dokumentacja.

Pandas wykorzystuje dwie główne struktury danych: Series i DataFrame. Ta pierwsza to jednowymiarowa tablica wartości posiadająca indeksy każdego elementu, z kolei druga to dwuwymiarowa tablica wartości z indeksami zarówno rzędów jak i kolumn. Obydwie struktury mogą przechowywać dowolne typy danych. [17]

3.4.2. NumPy

NumPy to biblioteka dla Pythona wykorzystywana do pracy na obiektach będących wielowymiarowymi tablicami oraz udostępniająca szereg narzędzi do obliczeń naukowych. Została stworzona w roku 2005 przez Trávisa Oliphanta jako następcę bibliotek Numeric [18] i Numarray [19]. Biblioteka jest wciąż rozwijana przez grupę programistów. NumPy jest szeroko wykorzystywana do przeprowadzania obliczeń naukowych ze względu na szereg zalet: [20][21]

- wspiera wielowymiarowe tabele, zarówno o homogenicznych jak i heterogenicznych typach danych,
- oferuje dużą ilość funkcji matematycznych, generatorów liczb pseudolosowych, transformat i innych narzędzi przydatnych w zastosowaniach naukowych,
- łatwo integruje się z innymi bibliotekami,
- posiada bardzo rozbudowaną dokumentację.

Do wad zaliczają się stosunkowo niska wydajność i problemy z obsługą niektórych typów danych i brakujących wartości. [21]

3.4.3. SciKit-learn

SciKit-learn to biblioteka zawierająca narzędzia związane z uczeniem maszynowym dla Pythona. Jest częścią większego zestawu narzędzi SciKit, zawierającego biblioteki oferujące szeroki wachlarz narzędzi do analizy danych. Początkowo stworzony przez Davida Cournapeau jako projekt w ramach "Google Summer of Code" w roku 2007, został później przepisany i rozwinięty głównie przez Francuski Instytut Badań w Dziedzinie Informatyki i Automatyki (INRIA) i udostępniony publicznie w roku 2010. SciKit-learn przestrzega zasad projektowania API dla oprogramowania uczenia maszynowego, które obejmują między innymi spójność, rozsądne domyślne ustawienia i dokumentację. [4]

3.4.4. EasyGUI

EasyGUI to biblioteka Pythona służąca do implementacji prostego interfejsu graficznego. Został stworzony przez Stephena Ferga i udostępniony w roku 2004. Oparty jest na bibliotece Tkinter [22]. Główną zaletą EasyGUI jest jego prosta składnia, pozwalająca na tworzenie nieskomplikowanych aplikacji bez konieczności uczenia się stosowania bardziej zaawansowanych bibliotek. Osiągnięte to zostało przez zastosowanie

predefiniowanych okien spełniających najczęściej wykorzystywane role, takie jak okna wyboru, wyświetlanie wiadomości, wpisywanie tekstu i inne. Niestety zaprzestano dalszego rozwoju biblioteki, przez co nie można spodziewać się dodania nowych funkcji czy poprawy działania istniejących. [5]

3.5. Źródła danych

3.5.1. Użyte repozytoria danych

Aby wyniki badań niosły ze sobą odpowiednią wartość merytoryczną, potrzebne są odpowiednie zbiory danych na których zostaną przeprowadzone testy. W celu znalezienia odpowiednich zbiorów danych, przyjęto następujące założenia:

- zbiór danych musi być wystarczająco duży,
- zbiór danych musi zawierać odpowiednią ilość atrybutów aby modele decyzyjne miały do dyspozycji wystarczającą ilość danych uczących,
- atrybuty powinny zawierać różnorodne typy danych w celu przetestowania wypełniania zarówno danych liczbowych (całkowitych i zmiennoprzecinkowych) jak i kategoriycznych,
- zbiór danych nie może mieć pustych wartości.

Do wyszukania odpowiednich zbiorów danych wykorzystano narzędzie Google Dataset Search. [23][24] Z jego pomocą wybrano 2 zbiory danych z różnych dziedzin. Po uprzedniej ich obróbce zostały wykorzystane do przeprowadzenia testów algorytmów wypełniania.

3.5.2. Adult Data Set

Pierwszy zbiór danych "Adult Data Set" zawiera dane ze spisu ludności przeprowadzonego w roku 1994 w Stanach Zjednoczonych. Jest szeroko wykorzystywany do testowania uczenia maszynowego. Zawiera ponad 30000 rekordów i 15 atrybutów. [25] Opis atrybutów:

- age: wiek spisanej osoby, liczba całkowita,
- workclass: rodzaj zatrudnienia, dane kategoriyczne, 8 możliwych wartości,

- fnlwgt: jaka proporcja populacji ma identyczny zestaw pozostałych wartości, liczba całkowita,
- education: osiągnięty poziom edukacji, dane kategoryczne, 16 możliwych wartości,
- education-num: osiągnięty poziom edukacji zakodowany jako liczba całkowita,
- martial-status: status matrymonialny, dane kategoryczne, 7 możliwych wartości,
- occupation: zawód, dane kategoryczne, 14 możliwych wartości,
- relationship: rola w związku, dane kategoryczne, 6 możliwych wartości,
- race: klasyfikacja rasowa, dane kategoryczne, 5 możliwych wartości,
- sex: płeć, dane kategoryczne, 2 możliwe wartości,
- capital-gain: zysk kapitału w związku z inwestycjami, liczba całkowita,
- capital-loss: strata kapitału w związku z inwestycjami, liczba całkowita,
- hours-per-week: ilość godzin pracujących w tygodniu, liczba całkowita,
- native-country: kraj pochodzenia, dane kategoryczne, 41 możliwych wartości,
- attribute: czy osoba zarabia powyżej czy poniżej 50000\$ rocznie.

Ten zbiór danych został wybrany ze względu na występowanie zarówno atrybutów liczbowych jak i kategorycznych, zadowalającą ilość rekordów oraz atrybutów. Został wybrany do celu przetestowania skuteczności działania algorytmów do wypełniania brakujących miejsc w zbiorach danych z brakami w danych o różnych typach. Nie wymaga dodatkowej obróbki przed rozpoczęciem testów.

3.5.3. Stock Exchange Data

Drugi zbiór danych "Stock Exchange Data" zawiera informacje o cenach akcji na giełdach w różnych krajach w latach 1965-2021. Dane zostały zebrane z Yahoo Finance, posiadającego dane o giełdzie z wielu lat w wielu krajach. Posiada ponad 100000 rekordów i 9 atrybutów. [26] Opis atrybutów:

- Index: symbol wskazujący z jakiej giełdy pochodzą dane, dane kategoryczne, 5 możliwych wartości,

- Date: data obserwacji, dane katagoryczne,
- Open: cena akcji podczas otwarcia, liczba wymierna,
- High: najwyższa cena w ciągu dnia, liczba wymierna,
- Low: najniższa cena w ciągu dnia, liczba wymierna,
- Close: cena akcji w momencie zamknięcia, liczba wymierna,
- Adj Close: cena akcji w momencie zamknięcia skorygowana o podziały jak i dywidendy, liczba wymierna,
- Volume: liczba akcji będących przedmiotem obrotu w ciągu dnia sesyjnego, liczba całkowita,
- CloseUSD: cana akcji w momencie zamknięcia wyrażona w dolarach amerykańskich

Ten zbiór danych został wybrany ze względu na bardzo popularną kategorię danych, to jest dane finansowe. Ma na celu przetestowanie skuteczności działania algorytmów w przypadku danych numerycznych, w szczególności liczb wymiernych (typu float). W celu lepszego przygotowania do testów zakodowano kolumnę Data z wykorzystaniem label encoding, to jest zamiany danych na postać numeryczną. Usunięto też rekordy posiadające wartość "0" w kolumnie "Volume". Ich duża liczba (ponad 30%) mogłaby negatywnie wpłynąć na uczenie modeli decyzyjnych. W wyniku tego zmniejszono liczbę rekordów do ponad 62000.

4. Implementacja i testy

4.1. Opis przygotowanego programu

4.1.1. Założenia i realizacja

Założono, że program ma realizować 3 zadania:

- 1) Przygotować dane do wypełniania poprzez sztuczne utworzenie brakujących wartości.
- 2) Wypełnić brakujące wartości z wykorzystaniem wybranych algorytmów.
- 3) Ocenić skuteczność wypełniania w celu porównania algorytmów.

Poszczególne zadania zrealizowano jako osobne moduły.

Przyjęto też następujące założenia:

- 1) Wykorzystanym językiem ma być Python.
- 2) Program ma być napisany zgodnie z paradygmatem programowania obiektowego.
- 3) Poszczególne klasy mają być zawarte w osobnych plikach.
- 4) Interakcja z programem ma opierać się o prosty interfejs graficzny.
- 5) Dostęp do wszystkich modułów programu ma być zapewniony z jednego miejsca.
- 6) Pliki wygenerowane przez jeden moduł mają być przygotowane w sposób umożliwiający wykorzystanie ich przez kolejny. Oprócz odpowiedniego formatowania wewnątrz pliku, oznacza to przyjęcie konwencji nazewnictwa plików opartej o prefiksy i sufiksy.

Program składa się z następujących plików:

- mgr_main.py: główny plik nie zawierający żadnej klasy, odpowiadający za wybór modułu do uruchomienia, i uruchomienie odpowiedniego modułu po wybraniu,

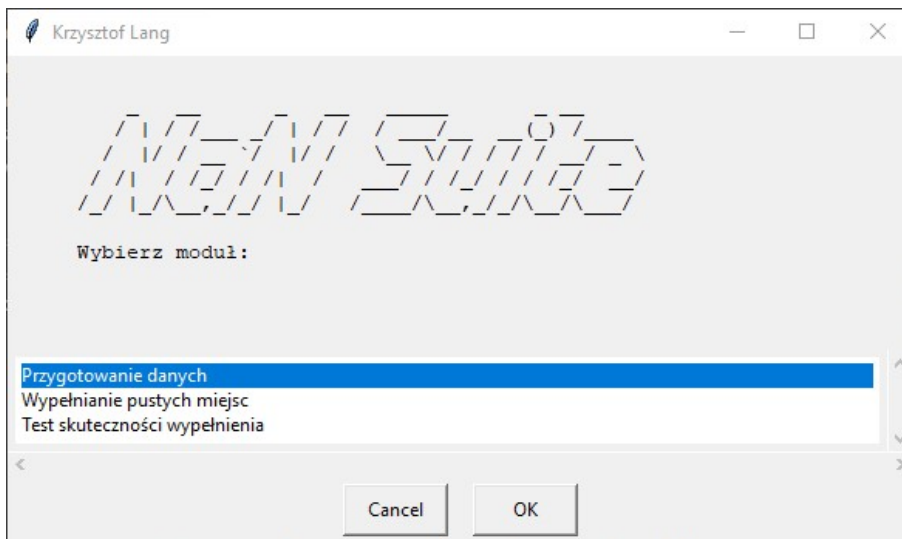
- mgr_nan_gen.py: plik zawierający klasę NanGen, odpowiadającą za realizację modułu przygotowującego plik,
- mgr_fill.py: plik zawierający klasę Fill, odpowiadającą za realizację modułu wypełniającego brakujące dane,
- mgr_data.py: plik zawierający klasę Data, odpowiadającą za wybranie pliku do wypełnienia i przygotowanie do dalszej obróbki, oraz klasę PrepareData, odpowiadającą za przygotowanie danych do przekazania silnikowi uczenia maszynowego celem wypełnienia oraz późniejszemu przywróceniu danym ich pierwotnego wyglądu
- mgr_di.py: plik zawierający klasę DownImpu, odpowiadającą za przygotowanie danych dla algorytmu trzeciego.
- mgr_temp_fill: plik zawierający klasę TempFill, odpowiadającą za tymczasowe wypełnianie brakujących miejsc, potrzebne podczas przygotowywania danych dla algorytmu Prostego
- mgr_acc: plik zawierający klasę AccuracyTester, odpowiadającą za obliczanie skuteczności wypełniania danych

Wykorzystując narzędzie auto-py-to-exe, utworzono plik mgr_suite.exe, pozwalający uruchomić program bez konieczności instalowania interpretera Python i potrzebnych bibliotek. [27]

4.1.2. Działanie programu

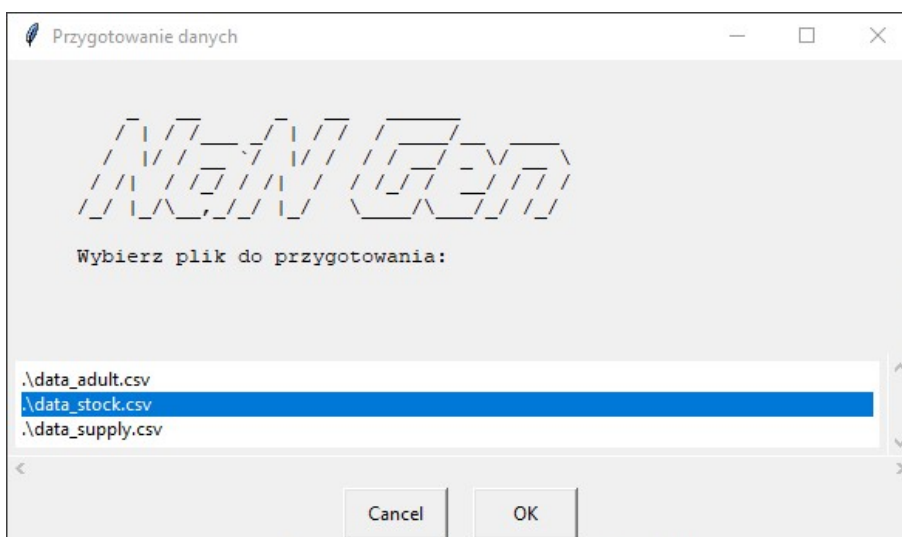
Poniżej zaprezentowano działanie programu na przykładzie pliku data__stock.csv. Zostanie on najpierw przygotowany do testów, następnie brakujące dane zostaną wypełnione z wykorzystaniem jednego z algorytmów, po czym zostanie obliczona dokładność tego wypełnienia.

Po uruchomieniu mgr_suite.exe wyświetlone zostaje okno służące do wyboru modułu, pokazane na rysunku 4.1.



Rysunek 4.1: Główne okno programu, pozwalające na wybór modułu do uruchomienia

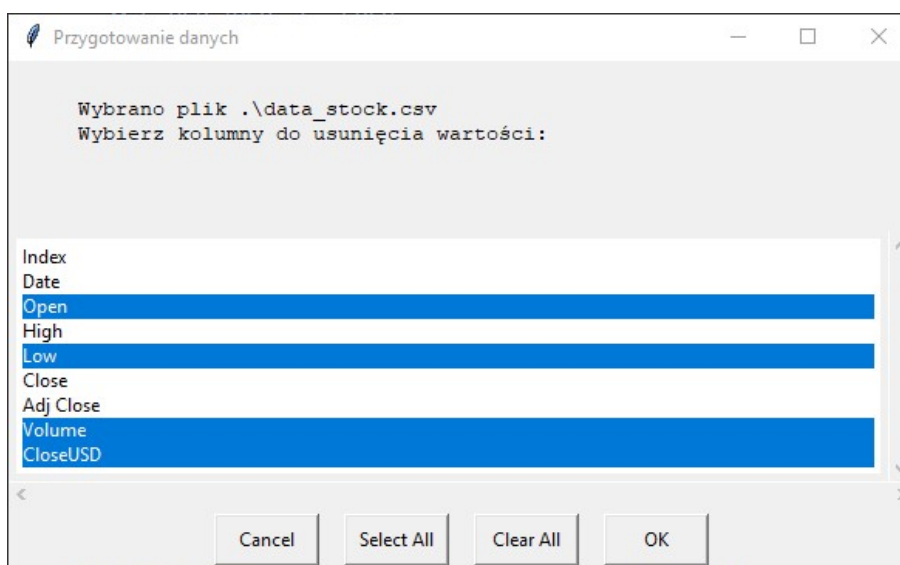
Pierwszy moduł odpowiada za przygotowanie danych do wypełniania poprzez usunięcie losowych wartości ze zbioru danych. Pierwszym krokiem jest wybranie pliku który ma zostać przygotowany z użyciem okna pokazanego na rysunku 4.2. Lista plików generowana jest na podstawie plików znajdujących się w tym samym folderze co uruchamiany program spełniających założony format nazwy. Założono, że pliki które nadają się do wypełnienia mają być zapisane w formacie CSV, natomiast nazwa zaczynać się ma od prefiksu “data_” i nie posiadać żadnych sufiksów.



Rysunek 4.2: Okno wyboru pliku do przygotowania

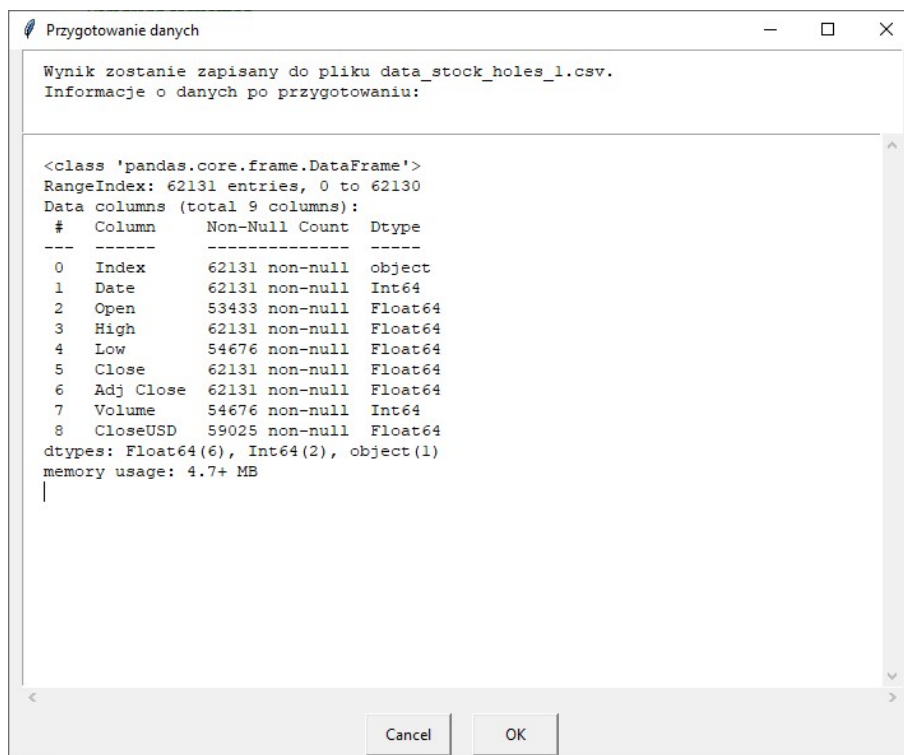
Następnie wybrane z listy zostają kolumny w których mają zostać usunięte

dane. Wyboru dokonuje się z użyciem okna pokazanego na rysunku 4.3. Kolumny do wyboru wyekstrahowane są bezpośrednio z załadowanego wcześniej pliku. Wybrać można dowolną kombinację, lecz zalecane jest poniżej 50%.



Rysunek 4.3: Okno wyboru kolumn

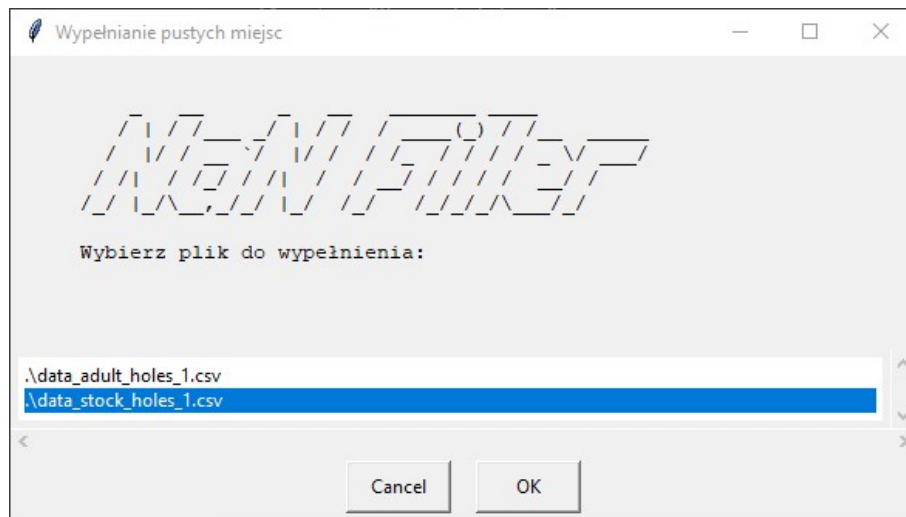
W wybranych kolumnach zostaje usunięte od 5% do 15% wartości - dla każdej kolumny faktyczna wartość jest losowana. Dodatkowo stworzony zostaje plik przechowujący informacje które wartości zostały usunięte. Ta informacja zostaje później wykorzystana do oceny skuteczności wypełnienia zbioru danych. Po zakończeniu usuwania wartości wyświetlone zostaje okno z podsumowaniem jak na rysunku 4.4. Nazwa utworzonego pliku z gotowymi danymi tworzona jest poprzez dodanie sufiksu “_holes_X” do nazwy oryginalnego pliku, gdzie X to kolejna liczba naturalna. Umożliwia to tworzenie plików z danymi usuniętymi z różnych zestawów kolumn bez konieczności ręcznej zmiany ich nazw. Nazwa pliku z informacją które dane zostały usunięte tworzona jest przez dodanie sufiksu “_journal” do nazwy oryginalnego pliku.



Rysunek 4.4: Okno z podsumowaniem

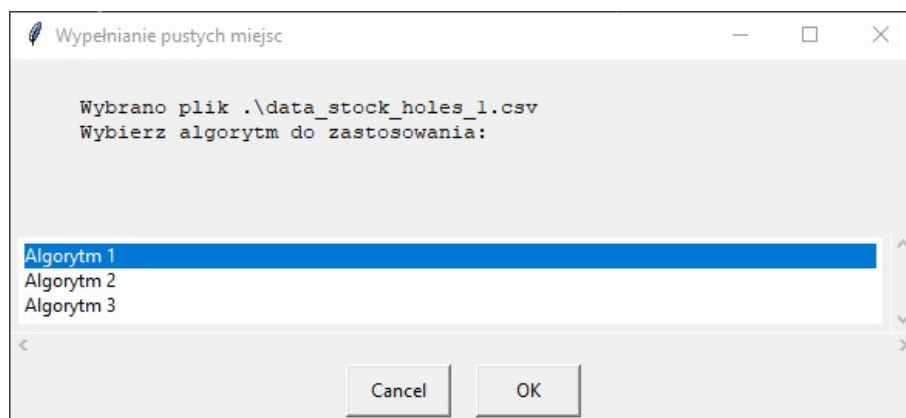
Drugi moduł służy do wypełniania brakujących wartości w zbiorze danych z użyciem wybranego algorytmu.

Najpierw należy wskazać plik który ma zostać wypełniony z użyciem okna wyboru pokazanego na rysunku 4.5. Tak jak wcześniej, lista tworzona jest na podstawie plików w folderze i przyjętej konwencji nazewnictwa plików. Wyświetlane są wyłącznie pliki posiadające sufiks “_holes_X” w nazwie, bez kolejnych sufiksów.



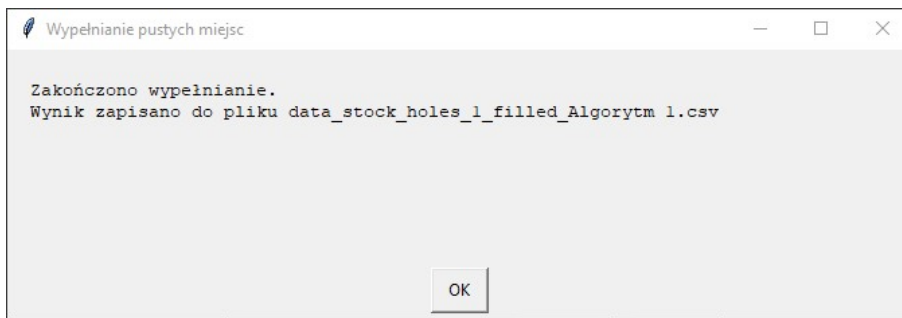
Rysunek 4.5: Okno wyboru pliku do wypełnienia

Następnie z użyciem okna jak na rysunku 4.6 wybrany zostaje algorytm który ma zostać wykorzystany do wypełniania brakujących wartości. Wyświetlona zostaje też nazwa wybranego wcześniej pliku w celu uniknięcia błędu wybrania niewłaściwego pliku.



Rysunek 4.6: Okno wyboru algorytmu

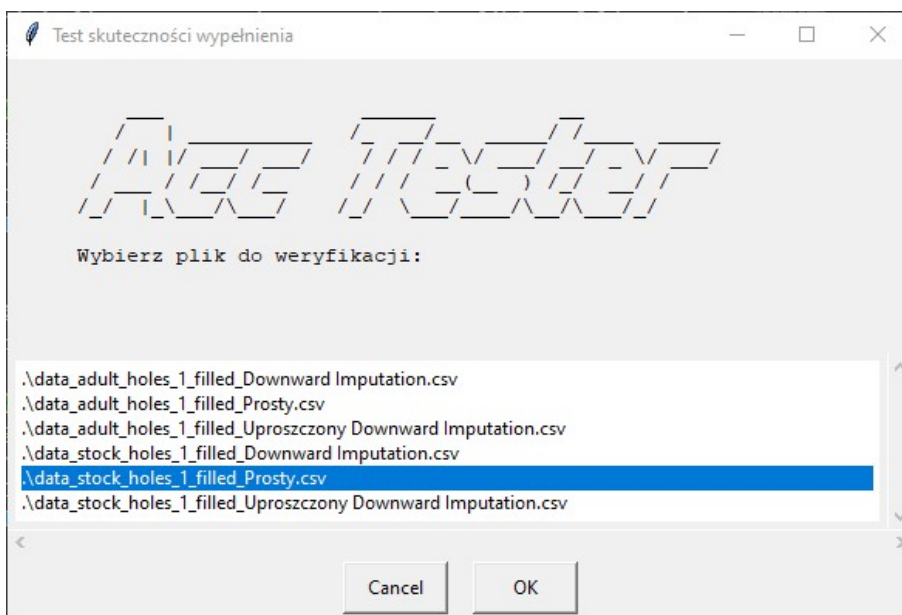
Puste miejsca zostają wypełnione z wykorzystaniem wybranego algorytmu. Dokładny sposób działania algorytmów zostanie opisany w kolejnych rozdziałach. Po zakończeniu wypełniania wyświetlone zostaje podsumowanie jak na rysunku 4.7. Nazwa pliku wynikowego tworzona jest poprzez dodanie sufiksu “_filled_Y”, gdzie Y to nazwa wybranego algorytmu.



Rysunek 4.7: Okno z podsumowaniem wypełniania

Ostatni moduł odpowiada za wygenerowanie danych, które można wykorzystać do oceny skuteczności wypełniania brakujących wartości. Jako wystarczające uznano procent skuteczności wypełniania dla danych kategorycznych i liczb całkowitych oraz średnie odchylenie bezwzględne dla wszystkich danych liczbowych. Obie wartości są obliczane dla poszczególnych wypełnionych kolumn.

Jak w przypadku poprzednich modułów, zacząć należy od wyboru pliku który ma zostać poddany analizie. Wyboru dokonuje się z użyciem okna pokazanego na rysunku 4.8. Wyświetlane są tylko pliki zawierające słowo “filled” w nazwie, ponieważ takie pliki posiadają wartości wypełnione za pomocą któregoś algorytmu z użyciem odpowiedniego modułu.



Rysunek 4.8: Okno wyboru pliku do analizy

Z użyciem wybranego pliku, oryginalnego pliku z danymi przed usunięciem losowych danych oraz pliku z informacją które dane zostały usunięte a następnie wypełnione, przeprowadzane jest obliczanie skuteczności wypełniania danych.

Obliczenie procentowej skuteczności wypełnienia przebiega następująco:

- 1) Sprawdzenie ilości wypełnianych wartości w danej kolumnie.
- 2) Zsumowanie ilości poprawnie wypełnionych danych w kolumnie poprzez porównanie wartości o współrzędnych zapisanych w pliku tworzonym podczas przygotowywania danych.
- 3) Zastosować wzór 4.1:

$$ACC = \frac{m}{n} \times 100\% \quad (4.1)$$

gdzie: ACC – procentowa skuteczność wypełnienia kolumny, m – ilość poprawnie wypełnionych wartości, n – ilość wypełnionych wartości.

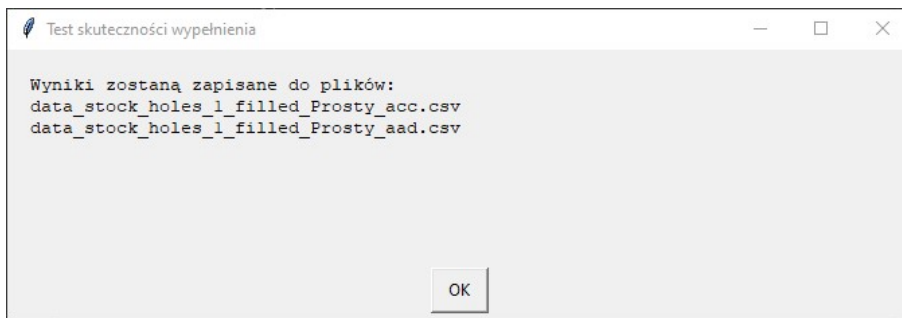
Aby obliczyć średnie odchylenie bezwzględne należy zastosować wzór 4.2:

$$AAD = \frac{\sum_{i=1}^n |x_i - \hat{x}_i|}{n} \quad (4.2)$$

gdzie: AAD – średnie odchylenie bezwzględne dla danej kolumny, n – ilość wypełnionych wartości w kolumnie, x_i – wartość wypełnionego i -tego elementu kolumny, \hat{x}_i – oryginalna wartość i -tego elementu kolumny.

Po zakończeniu obliczeń, wyświetlane jest podsumowanie jak na rysunku 4.9, a wynik obliczeń zapisywany jest w dwóch plikach

- wyniki obliczania procentowej skuteczności jest zapisywany w pliku o nazwie tworzonej przez dodanie do nazwy analizowanego pliku sufiksu "_aad",
- wyniki obliczania średniego odchylenia bezwzględnego jest zapisywany w pliku o nazwie tworzonej przez dodanie do nazwy analizowanego pliku sufiksu "_acc".



Rysunek 4.9: Okno z podsumowaniem

4.2. Opis implementacji algorytmów

Implementowane algorytmy opierają się o kolejne rozwinięcia idei zaprezentowanej w [28]. Polega ona na sformułowaniu problemu wypełniania problemu wypełniania brakujących wartości jako problemu decyzyjnego. Wynikają z tego następujące założenia:

- kolumny wypełniane są pojedynczo,
- samo wypełnianie przeprowadzane jest przez system decyzyjny,
- algorytm ma za zadanie wybrać kolejność wypełniania kolumn oraz określić jakie kolumny będą zawarte w zbiorze danych uczących systemu decyzyjnego.

Działanie poszczególnych algorytmów zostanie zaprezentowane na przykładowej tabeli 4.1 przedstawiającej zbiór danych:

Tabela 4.1: Tabela wyjściowa

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	*	*	x	*	*	*	x	*	*	*	x	*	*	*
2	x	*	*	*	*	*	*	*	*	*	*	x	*	*
3	*	*	*	*	x	*	*	*	*	*	*	x	*	*
4	x	*	x	*	x	*	x	x	*	*	x	x	*	*
5	*	*	*	*	x	*	*	x	x	*	*	*	*	*
6	*	*	*	*	x	*	x	x	*	*	*	*	*	*
7	*	*	*	*	x	*	*	*	*	*	*	*	*	*
8	*	*	*	*	*	*	*	x	*	*	x	*	x	*
9	*	*	*	*	x	*	*	*	x	*	*	*	x	*
10	*	*	*	*	*	*	x	*	x	*	*	*	*	*
11	*	x	*	*	x	*	*	*	*	*	*	*	*	*

gdzie: * – istniejąca wartość, x – brakująca wartość.

4.2.1. Algorytm pierwszy

Procedura dla algorytmu pierwszego wygląda następująco:

- 1) Tworzony jest zbiór kolumn z brakującymi wartościami X . Dla przykładowej tabeli zbiór będzie wyglądał następująco:

$$X = \{A, B, C, E, G, H, I, K, L, M\}.$$

- 2) Kolumny w zbiorze X są następnie sortowane od kolumn zawierających najmniejszą ilość brakujących wartości do zawierających ich najwięcej, dając ciąg η . W przypadku kolumn o takiej samej ilości brakujących danych kolejność nie ma znaczenia. Dla przykładowej tabeli zbiór ten będzie wyglądał następująco:

$$\eta = (B, M, C, A, L, K, I, H, G, E).$$

- 3) Pierwszy element w ciągu η zostaje wybrany jako kolumna która zostanie wypełniona. W przykładzie będzie to element B .
- 4) Tabela zostaje rozdzielona na dwie: w jednej znajdują się rekordy w których w kolumnie wybranej do wypełnienia znajdują się dane, a w drugiej rekordy w

których w wybranej tabeli danych brakuje. Dla przykładowej tabeli będzie to wyglądało następująco, gdzie tabela 4.2 przedstawia tabelę z danymi w wybranej kolumnie, natomiast tabela 4.3 przedstawia tabelę z brakiem danych w wybranej kolumnie.

Tabela 4.2: Tabela wydzielona z oryginalnej, zawierająca wyłącznie rekordy z danymi w rozważanej kolumnie

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	*	*	x	*	*	*	x	*	*	*	x	*	*	*
2	x	*	*	*	*	*	*	*	*	*	*	x	*	*
3	*	*	*	*	x	*	*	*	*	*	*	x	*	*
4	x	*	x	*	x	*	x	x	*	*	x	x	*	*
5	*	*	*	*	x	*	*	x	x	*	*	*	*	*
6	*	*	*	*	x	*	x	x	*	*	*	*	*	*
7	*	*	*	*	x	*	*	*	*	*	*	*	*	*
8	*	*	*	*	*	*	*	x	*	*	x	*	x	*
9	*	*	*	*	x	*	*	*	x	*	*	*	x	*
10	*	*	*	*	*	*	x	*	x	*	*	*	*	*

Tabela 4.3: Tabela wydzielona z oryginalnej, zawierająca wyłącznie rekordy z brakiem danych w rozważanej kolumnie

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
11	*	x	*	*	x	*	*	*	*	*	*	*	*	*

- 5) Pierwsza z tabel - w przykładzie tabela 4.2 - posłuży do uczenia modelu uczenia maszynowego. Wypełniana kolumna służy jako cel, a pozostałe jako dane uczące.
- 6) Następnie nauczony model zostaje wykorzystany do wypełnienia brakujących wartości w drugiej tabeli, w przykładzie tabeli 4.3.
- 7) Na koniec tabele są łączone w jedną co w przykładzie skutkuje tabelą 4.4, a algorytm jest powtarzany aż do wypełnienia wszystkich brakujących wartości.

Tabela 4.4: Tabela prezentująca wygląd danych po pojedynczym przejściu głównej pętli algorytmu

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	*	*	x	*	*	*	x	*	*	*	x	*	*	*
2	x	*	*	*	*	*	*	*	*	*	*	x	*	*
3	*	*	*	*	x	*	*	*	*	*	*	x	*	*
4	x	*	x	*	x	*	x	x	*	*	x	x	*	*
5	*	*	*	*	x	*	*	x	x	*	*	*	*	*
6	*	*	*	*	x	*	x	x	*	*	*	*	*	*
7	*	*	*	*	x	*	*	*	*	*	*	*	*	*
8	*	*	*	*	*	*	*	x	*	*	x	*	x	*
9	*	*	*	*	x	*	*	*	x	*	*	*	x	*
10	*	*	*	*	*	*	x	*	x	*	*	*	*	*
11	*	*	*	*	x	*	*	*	*	*	*	*	*	*

4.2.2. Algorytm drugi

Algorytm drugi jest rozwinięciem pierwszego. Różni się sposobem wyboru kolumn mających pełnić rolę danych uczących. Eliminuje on z tego zbioru kolumny które również posiadają brakujące dane. Oznacza to, że kolumny wypełniane w pierwszej kolejności wypełniane są z użyciem silnika uczenia maszynowego uczącego się na mniejszej ilości danych, lecz lepszej jakości.

Procedura dla algorytmu drugiego wygląda następująco:

- 1) Tworzony jest zbiór kolumn z brakującymi wartościami X . Dla przykładowej tabeli zbiór będzie wyglądał następująco:

$$X = \{A, B, C, E, G, H, I, K, L, M\}.$$

- 2) Kolumny w zbiorze X są następnie sortowane od kolumn zawierających najmniejszą ilość brakujących wartości do zawierających ich najwięcej, dając ciąg η . W przypadku kolumn o takiej samej ilości brakujących danych kolejność nie ma znaczenia. Dla przykładowej tabeli zbiór ten będzie wyglądał następująco:

$$\eta = (B, M, C, A, L, K, I, H, G, E).$$

- 3) Pierwszy element w ciągu η zostaje wybrany jako kolumna która zostanie wypełniona. W przykładzie będzie to element B .
- 4) Tabela zostaje podzielona na dwie. W jednej znajdują się kolumny z pustymi wartościami, oprócz tej która ma zostać wypełniona. W drugiej znajdują się kolumny pełne oraz ta wyznaczona do wypełnienia. Tabele 4.5 i 4.6 przedstawiają rozdzielanie tabeli dla rozważanego przykładu:

Tabela 4.5: Tabela wydzielona z oryginalnej, zawierająca tylko kolumny z brakującymi danymi, z wyjątkiem wyznaczonej do wypełnienia

	A	C	E	G	H	I	K	L	M
1	*	x	*	x	*	*	x	*	*
2	x	*	*	*	*	*	*	x	*
3	*	*	x	*	*	*	*	x	*
4	x	x	x	x	x	*	x	x	*
5	*	*	x	*	x	x	*	*	*
6	*	*	x	x	x	*	*	*	*
7	*	*	x	*	*	*	*	*	*
8	*	*	*	*	x	*	x	*	x
9	*	*	x	*	*	x	*	*	x
10	*	*	*	x	*	x	*	*	*
11	*	*	x	*	*	*	*	*	*

Tabela 4.6: Tabela wydzielona z oryginalnej, zawierająca tylko kolumny bez brakujących danych i kolumnę wyznaczoną do wypełnienia

	B	D	F	J	N
1	*	*	*	*	*
2	*	*	*	*	*
3	*	*	*	*	*
4	*	*	*	*	*
5	*	*	*	*	*
6	*	*	*	*	*
7	*	*	*	*	*
8	*	*	*	*	*
9	*	*	*	*	*
10	*	*	*	*	*
11	x	*	*	*	*

- 5) Tabela z kolumną do wypełnienia znowu zostaje rozdzielona. W jednej znajdują się rekordy w których w kolumnie wybranej do wypełnienia znajdują się dane, a w drugiej rekordy w których w wybranej tabeli danych brakuje. Dla przykładowej tabeli będzie to wyglądało następująco, gdzie tabela 4.7 przedstawia tabelę z danymi w wybranej kolumnie, natomiast tabela 4.8 przedstawia tabelę z brakiem danych w wybranej kolumnie.

Tabela 4.7: Tabela wydzielona z oryginalnej, zawierająca tylko kolumny bez brakujących danych i kolumnę wyznaczoną do wypełnienia

	B	D	F	J	N
1	*	*	*	*	*
2	*	*	*	*	*
3	*	*	*	*	*
4	*	*	*	*	*
5	*	*	*	*	*
6	*	*	*	*	*
7	*	*	*	*	*
8	*	*	*	*	*
9	*	*	*	*	*
10	*	*	*	*	*

Tabela 4.8: Tabela wydzielona z oryginalnej, zawierająca tylko kolumny bez brakujących danych i kolumnę wyznaczoną do wypełnienia

	B	D	F	J	N
11	x	*	*	*	*

- 6) Pierwsza z tabel - w przykładzie tabela 4.7 - posłuży do uczenia modelu uczenia maszynowego. Wypełniana kolumna służy jako cel, a pozostałe jako dane uczące.
- 7) Następnie nauczony model zostaje wykorzystany do wypełnienia brakujących wartości w drugiej tabeli, w przykładzie tabeli 4.8.
- 8) Następnie tabele są łączone spowrotem w jedną, przywracając oryginalne rozmiary. W pierwszej kolejności łączone są ze sobą tabele które brały udział w uczeniu i wypełnianiu. W przykładzie będą to tabele 4.7 i 4.8 tworząc 4.9.

Tabela 4.9: Tabela wynikająca z połączenia tabel biorących udział w nauczaniu i wypełnianiu

	B	D	F	J	N
1	*	*	*	*	*
2	*	*	*	*	*
3	*	*	*	*	*
4	*	*	*	*	*
5	*	*	*	*	*
6	*	*	*	*	*
7	*	*	*	*	*
8	*	*	*	*	*
9	*	*	*	*	*
10	*	*	*	*	*
11	*	*	*	*	*

- 9) Na koniec dołączana jest wcześniej utworzona tabela zawierająca wyłączne kolumny z brakującymi wartościami. W przykładzie oznacza to łączenie tabel 4.9 i 4.5, uzyskując tabelę 4.10.

Tabela 4.10: Tabela prezentująca wygląd danych po pojedynczym przejściu głównej pętli algorytmu

	B	D	F	J	N	A	C	E	G	H	I	K	L	M
1	*	*	*	*	*	*	x	*	x	*	*	x	*	*
2	*	*	*	*	*	x	*	*	*	*	*	*	x	*
3	*	*	*	*	*	*	*	x	*	*	*	*	x	*
4	*	*	*	*	*	x	x	x	x	x	*	x	x	*
5	*	*	*	*	*	*	*	x	*	x	x	*	*	*
6	*	*	*	*	*	*	*	x	x	x	*	*	*	*
7	*	*	*	*	*	*	*	x	*	*	*	*	*	*
8	*	*	*	*	*	*	*	*	*	x	*	x	*	x
9	*	*	*	*	*	*	*	x	*	*	x	*	*	x
10	*	*	*	*	*	*	*	*	x	*	x	*	*	*
11	*	*	*	*	*	*	*	x	*	*	*	*	*	*

4.2.3. Algorytm trzeci

Algorytm trzeci jest kolejnym rozwinięciem drugiego. On również skupia się głównie na zmianie logiki stojącej za doбором kolumn do wykorzystania w uczeniu silnika. Dodatkowo zmieniany jest sposób wyboru kolejności w jakiej wypełniane są kolumny. W przeciwieństwie do poprzedniego algorytmu, kolejność wypełniania kolumn nie jest dyktowana przez samą ilość brakujących wartości, a ilość brakujących wartości występujących w tych samych rekordach. Ma to na celu jak najbardziej efektywne uzyskanie informacji mających znaczenie przy wypełnianiu kolejnych kolumn. Również w tym celu kolumny o najmniejszym wpływie dla wypełniania innych kolumn wypełniane są w sposób prosty, to znaczy w oparciu o najczęściej występującą wartość lub średnią wartość w kolumnie, zależnie od typu danych.

Procedura dla algorytmu trzeciego wygląda następująco:

- 1) Tworzony jest zbiór kolumn z brakującymi wartościami X . Dla przykładowej tabeli zbiór będzie wyglądał następująco:

$$X = \{A, B, C, E, G, H, I, K, L, M\}$$

- 2) Ze zbioru X wybierana jest kolumna z największą liczbą brakujących wartości.

Ta kolumna kolumna jest usuwana ze zbioru X i dodawana do ciągu η , natomiast liczba brakujących wartości w tej kolumnie dodawana jest do ciągu δ . W przykładzie jest to kolumna E z 7 brakującymi elementami, a zbiór i ciągi będą wyglądały następująco:

$$X = \{A, B, C, G, H, I, K, L, M\}, \eta = (E), \delta = (7)$$

- 3) Następnie z tak powstałego zbioru X zostaje wybrana kolumna mająca najwięcej wspólnych z poprzednio wybraną kolumną brakujących wartości w rekordach. W przypadku gdy liczba wspólnych braków jest równa dla kilku kolumn, można wybrać dowolną z nich. Wybrana kolumna usuwana jest ze zbioru X i dodawana do ciągu η , a liczba wspólnych brakujących wartości do ciągu δ . W przykładzie jest to H , z 3 wspólnymi elementami. Poskutkuje to zbiorem i ciągami:

$$X = \{A, C, G, I, K, L, M\}, \eta = (E, H), \delta = (7, 3)$$

- 4) Powyższy krok jest powtarzany aż do wyczerpania elementów zbioru X . W ten sposób tworzone są 2 ciągi. Jeden zawiera kolumny w uzyskanej w ten sposób kolejności, oznaczony jako η , oraz drugi mu odpowiadający oznaczony δ , zawierający wartości liczbowe oznaczające liczbę wspólnych z poprzednim elementem brakujących wartości. Te ciągi dla przykładowej tabeli będą wyglądać następująco:

$$\eta = (E, H, G, K, C, L, A, I, M, B), \delta = (7, 3, 2, 2, 2, 1, 2, 0, 1, 0)$$

- 5) Jeśli ciąg δ zawiera na końcu wartości zerowe to są one usuwane z tego ciągu, to odpowiadające im kolumny zostają wypełnione w sposób prosty i usuwane z ciągu η . W przykładzie poskutkuje to ciągami:

$$\eta = (E, H, G, K, C, L, A, I, M), \delta = (7, 3, 2, 2, 2, 1, 2, 0, 1)$$

- 6) Tabela zostaje rozdzielona na podstawie zawartości ciągu η . Jedna tabela wynikowa zawiera kolumny zawarte w tym ciągu, a druga pozostałe, zawierająca wyłącznie kompletne dane. Tabele dla przykładu będą wyglądać następująco, gdzie tabela 4.11 zawiera kolumny z ciągu η , a tabela 4.12 pozostałe:

Tabela 4.11: Tabela zawierająca kolumny zawarte w ciągu η

	A	C	E	G	H	I	K	L	M
1	*	x	*	x	*	*	x	*	*
2	x	*	*	*	*	*	*	x	*
3	*	*	x	*	*	*	*	x	*
4	x	x	x	x	x	*	x	x	*
5	*	*	x	*	x	x	*	*	*
6	*	*	x	x	x	*	*	*	*
7	*	*	x	*	*	*	*	*	*
8	*	*	*	*	x	*	x	*	x
9	*	*	x	*	*	x	*	*	x
10	*	*	*	x	*	x	*	*	*
11	*	*	x	*	*	*	*	*	*

Tabela 4.12: Tabela zawierająca kolumny nie zawarte w ciągu η

	B	D	F	J	N
1	*	*	*	*	*
2	*	*	*	*	*
3	*	*	*	*	*
4	*	*	*	*	*
5	*	*	*	*	*
6	*	*	*	*	*
7	*	*	*	*	*
8	*	*	*	*	*
9	*	*	*	*	*
10	*	*	*	*	*
11	*	*	*	*	*

- 7) Ze zbioru η usuwana jest kolumna będąca ostatnim elementem ciągu i dokładana do tabeli zawierającej kompletne dane. Ta kolumna zostanie wypełniana. Tabela dzielona jest na dwie, jedną zawierającą rekordy w których wypełniana kolumna zawiera dane, i drugą w której wypełniana kolumna danych nie posiada. Pierwsza z nich posłuży jako dane treningowe dla modelu uczenia maszynowego, a druga zostanie z jego użyciem wypełniona. W rozpatrywanym przypadku przenoszona kolumną będzie M , więc ciągi po jej usunięciu będą wyglądały następująco:

$\eta = (E, H, G, K, C, L, A, I)$, $\delta = (7, 3, 2, 2, 2, 1, 2, 0)$ Z kolei tabela zawierająca dane treningowe będzie wyglądała jak 4.13 , z kolei tabela do wypełnienia jak 4.14.

Tabela 4.13: Tabela zawierająca kolumny nie zawarte w ciągu η

	B	D	F	J	N	M
1	*	*	*	*	*	*
2	*	*	*	*	*	*
3	*	*	*	*	*	*
4	*	*	*	*	*	*
5	*	*	*	*	*	*
6	*	*	*	*	*	*
7	*	*	*	*	*	*
10	*	*	*	*	*	*
11	*	*	*	*	*	*

Tabela 4.14: Tabela zawierająca kolumny nie zawarte w ciągu η

	B	D	F	J	N	M
8	*	*	*	*	*	x
9	*	*	*	*	*	x

- 8) Następnie uzyskane po wypełnieniu tabele są łączone w jedną, a powyższa procedura polegająca na przenoszeniu i wypełnianiu ostatniej kolumny z ciągu η jest powtarzana aż do jego wyczerpania i tym samym wypełnienia wszystkich

kolumn. Tabela 4.15 przedstawia tabelę po połączeniu tabel 4.13 i 4.14, gotową do dodania kolejnej tabeli z ciągu η

Tabela 4.15: Tabela gotowa do powtórzenia procedury dodania i wypełnienia kolumny

	B	D	F	J	N	M
1	*	*	*	*	*	*
2	*	*	*	*	*	*
3	*	*	*	*	*	*
4	*	*	*	*	*	*
5	*	*	*	*	*	*
6	*	*	*	*	*	*
7	*	*	*	*	*	*
8	*	*	*	*	*	*
9	*	*	*	*	*	*
10	*	*	*	*	*	*
11	*	*	*	*	*	*

4.3. Napotkane problemy

4.4. Testy algorytmów na wybranych źródłach danych

5. Podsumowanie i wnioski końcowe

Załączniki

Literatura

- [1] <https://www.python.org/>. Dostęp 26.02.2023.
- [2] <https://pandas.pydata.org/>. Dostęp 26.02.2023.
- [3] <https://numpy.org/>. Dostęp 26.02.2023.
- [4] <https://scikit-learn.org/stable/>. Dostęp 26.02.2023.
- [5] <https://easygui.sourceforge.net/>. Dostęp 26.02.2023.
- [6] <https://archive-beta.ics.uci.edu/>. Dostęp 26.02.2023.
- [7] <https://www.kaggle.com/datasets>. Dostęp 26.02.2023.
- [8] Geurts L., Meertens L., Pemberton S.: ABC Programmer's Handbook. Bosko Books, Londyn 2005.
- [9] <https://www.artima.com/articles/the-making-of-python>. Dostęp 26.02.2023.
- [10] Lutz M.: Learning Python, 5th Edition. O'Reilly Media Inc, Sebastopol 2013.
- [11] <https://www.redhat.com/en/topics/middleware/what-is-ide>. Dostęp 26.02.2023.
- [12] <https://code.visualstudio.com/>. Dostęp 26.02.2023.
- [13] Johnson B.: Visual Studio Code: End-to-End Editing and Debugging Tools for Web Developers. John Wiley & Sons, Indianapolis 2019.
- [14] <https://github.com/>. Dostęp 26.02.2023.
- [15] Chacon S., Straub B.: Pro Git 2nd ed. Apress, 2014.
- [16] Laster B.: Professional Git. John Wiley & Sons, Indianapolis 2017.
- [17] Petrou T.: Pandas Cookbook: Recipes for Scientific Computing, Time Series Analysis and Data Visualization using Python. Packt Publishing, Birmingham 2017.
- [18] <https://pypi.org/project/Numeric/>. Dostęp 26.02.2023.
- [19] <https://pypi.org/project/numarray/>. Dostęp 26.02.2023.

- [20] Ranjani J., Sheela A., Meena K. P.: "Combination of NumPy, SciPy and Matplotlib/PyLab -a good alternative methodology to MATLAB - A Comparative analysis"2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT), Chennai 2019
- [21] McKinney W.: Python for Data Analysis. O'Reilly Media Inc, Sebastopol 2012.
- [22] <https://docs.python.org/3/library/tkinter.html>. Dostęp 26.02.2023.
- [23] <https://datasetsearch.research.google.com/>. Dostęp 26.02.2023.
- [24] Noy N., Burgess M., Brickley D.: Google Dataset Search: Building a search engine for datasets in an open Web ecosystem. WebConf'2019, San Francisco 2019
- [25] <https://archive-beta.ics.uci.edu/dataset/2/adult>. Dostęp 26.02.2023.
- [26] www.kaggle.com/datasets/mattiuzc/stock-exchange-data. Dostęp 26.02.2023.
- [27] <https://pypi.org/project/auto-py-to-exe/>. Dostęp 26.02.2023.
- [28] Brański A.: Wybrane zagadnienia informatyki stosowanej. Oficyna Wydawnicza Politechniki Rzeszowskiej, Rzeszów 2020.

STRESZCZENIE PRACY DYPLOMOWEJ MAGISTERSKIEJ
IMPLEMENTACJA WYBRANYCH ALGORYTMÓW
WYPEŁNIANIA BRAKUJĄCYCH WARTOŚCI, DLA STRUMIENI
DUŻYCH ZBIORÓW DANYCH

Autor: inż. Krzysztof Lang, nr albumu: EF-148853

Opiekun: dr Michał Piętał

Słowa kluczowe: bazy danych, brakujące wartości, wypełnianie, duże zbiory danych

Dla poprawnej analizy danych ważna jest jej kompletność. Istnieje wiele sposobów radzenia sobie z brakującymi danymi. Najprostsze metody opierające się między innymi na średniej bądź najczęściej występującej wartości w wielu przypadkach mogą negatywnie wpłynąć na skuteczność analizy. Niniejsza praca ma na celu przeanalizować skuteczność uzupełniania brakujących danych z użyciem bardziej zaawansowanych metod opierających się na wykorzystaniu uczenia maszynowego. Te metody mają na celu wypełnić brakujące dane wartościami dużo bardziej zbliżonymi do rzeczywistych, minimalizując negatywny wpływ na skuteczność późniejszej analizy danych.

MSC THESIS ABSTRACT
IMPLEMENTATION OF SELECTED MISSING VALUE IMPUTATION
ALGORITHMS FOR LARGE DATA SETS

Author: Krzysztof Lang, BSc, code: EF-148853

Supervisor: Michał Piętał, PhD

Key words: databases, missing values, imputation, big data

For correct data analysis, completeness is important. There are many ways to deal with missing data. The simplest methods based on, among other things, the average or the most frequently occurring value in many cases can negatively affect the effectiveness of the analysis. This thesis aims to analyze the effectiveness of imputing missing data using more advanced methods based on the use of machine learning. These methods are designed to impute missing data with values much closer to the actual data, minimizing the negative impact on the effectiveness of subsequent data analysis.