POLITECHNIKA WARSZAWSKA

Wydział Elektroniki i Technik Informacyjnych

Instytut Systemów Elektronicznych

Piotr Tąkiel

nr albumu: 212512

PRACA DYPLOMOWA MAGISTERSKA

Tytuł pracy

Praca wykonana pod kierunkiem

dr inż. Roberta Nowaka

Warszawa, 11 października 2014

Streszczenie

Tytuł pracy

Streszczenie

Imię i nazwisko: Piotr Tąkiel

Specjalność: Inżynieria Systemów Informacyjnych

Data urodzenia: 12 stycznia 1990

Data rozpoczęcia studiów: 1 października 2009

ŻYCIORYS

Życiorys

Spis treści

1. Wstęp . . . 5

1.1 O pracy . . . 5

1.2 Cel i zakres pracy . . . 5

1.2.1 Cel pracy . . . 5

1.2.2 Zakres pracy . . . 5

2. Wprowadzenie do problemu klasyfikacji . . . 6

3. Założenia i wymagania systemu . . . 7

3.1 Wymagania funkcjonalne . . . 7

3.1.1 Przetwarzanie i klasyfikacja . . . 7

3.1.2 Przeszukiwanie . . . 8

3.1.3 Opis systemu komputerowego . . . 11

3.2 Wymagania niefunkcjonalne . . . 12

3.3 Wymagania ograniczeń . . . 13

3.4 Podsumowanie wymagań . . . 13

4. Projekt systemu komputerowego . . . 15

4.1 Projekt narzędzia . . . 15

4.2 Projekt interfejsu sieciowego . . . 15

4.3 Projekt testów . . . 15

4.3.1 Testy narzędzia . . . 15

4.3.2 Testy interfejsu . . . 15

5. Elementy implementacji systemu . . . 16

5.1 Implementacja narzędzia . . . 16

5.2 Implementacja interfejsu webowego . . . 16

6. Testy systemu . . . 17

6.1 Testy narzędzia . . . 17

6.2 Testy interfejsu webowego . . . 17

7. Wyniki badań . . . 18

8. Wnioski 19

Bibliografia 20

1. Wstęp

1.1 O pracy

1.2 Cel i zakres pracy

1.2.1 Cel pracy

1.2.2 Zakres pracy

2. Wprowadzenie do problemu klasyfikacji

3. Założenia i wymagania systemu

Zdefiniowanie wymagań ma bardzo duże znaczenie podczas realizacji projektu informatycznego. Właściwe sformułowanie wymagań uszczegóławia wizję systemu oraz jest źródłem wiedzy o pożądanych możliwościach i cechach rozwiązania. Dodatkowo, prawidłowa identyfikacja wymagań nakreśla cel projektu i, co za tym idzie, pozwala uniknąć potrzeby wprowadzania kosztownych poprawek.

Wymagania dzielimy na funkcjonalne, jakościowe (niefunkcjonalne) oraz wymagania ograniczeń. Do pierwszej grupy zaliczamy wymagania opisujące możliwości, jakie projektowany system daje użytkownikom. Przykładem wymagania funkcjonalnego może być klasyfikowanie danych przy użyciu algorytmu drzew decyzyjnych. Wymagania jakościowe skupiają się na ilościowych regułach oceniania systemu, np. czy baza danych potrafi odzyskać utracone dane w mniej niż jedną sekundę. Ostatnia kategoria wymagań precyzuje granice rozwiązania.

3.1 Wymagania funkcjonalne

Zgodnie z tym, co przedstawiłem w rozdziale poświęconym celowi pracy, tworzona przeze mnie aplikacja służy ułatwieniu zadań związanych z klasyfikacją danych. Aplikacja ma za zadania wykonywać operacje, które dla użytkownika zajęły by bardzo dużo czasu. Do czynności takich można zaliczyć wstępne przetwarzanie danych oraz ich późniejszą klasyfikację, jak również wyszukiwanie kombinacji metod oraz parametrów, których użycie prowadzi do najlepszej jakości klasyfikacji. Takie sformułowanie wymagań prowadzi do wyłonienia dwóch podstawowych funkcjonalności projektowanego rozwiązania.

3.1.1 Przetwarzanie i klasyfikacja

Pierwszą z głównych funkcjonalności jest przetwarzanie oraz klasyfikacja danych zgodnie z nastawami użytkownika. Odbiorca aplikacji powinien mieć pełną kontrolę nad tym w jaki sposób dane zostaną zmodyfikowane (o ile w ogóle) oraz jak ma przebiegać klasyfikacja. Powołując się na to, co opisałem we rozdziale o klasyfikacji, użytkownik powinien być w stanie:

1. Wybrać metodę radzenia sobie z brakującymi danymi w wierszach

2. Wskazać kolumny, które powinny zostać usunięte ze zbioru danych
3. Wskazać kolumny z danymi, które powinny zostać poddane normalizacji
4. Określić metodę kwantyfikcji danych w kolumnach wraz z parametrami
5. Określić metodę klasyfikacji danych wraz z parametrami

Spośród wyżej wymienionych operacji tylko pierwsza i ostatnia powinny być zawsze określone. Wynika to z faktu, że wiele algorytmów klasyfikacji zakłada obecność wszystkich danych i nie uwzględnia sytuacji, w których pewnych wartości brakuje. Inaczej jest w przypadku pozostałych czynności. Usuwanie, normalizacja oraz kwantyfikacja kolumn mogą mieć zasadniczy wpływ na przebieg całego algorytmu, ale nie są konieczne.

Należy zwrócić uwagę na wymaganie dotyczące kwantyfikacji wartości w kolumnach. Aplikacja powinna pozwalać na wykorzystanie wielu różnych algorytmów łączenia jednocześnie. Dla każdego wybranego kwantyfikatora użytkownik powinien móc wskazać grupę kolumn poddanych łączeniu oraz parametry algorytmu.

Rezultatem działania aplikacji powinna być wartość walidacji krzyżowej na przetworzonych danych z użyciem określonego algorytmu.

3.1.2 Przeszukiwanie

Drugą podstawową funkcjonalnością jest wyszukiwanie najlepszej metody przetwarzania i klasyfikacji określonych danych. Ogromna rozmaitość dostępnych ustawień całego algorytmu sprawia, że zazwyczaj udaje nam się przejrzeć jedynie niewielki fragment wszystkich możliwości. Z tego powodu dla ludzi zadanie to wiąże się często z dobieraniem kombinacji metod oraz ich parametrów „na chybił trafił”. Czynność, która dla wielu ludzi okazuje się czasochłonna i nużąca, może być w znacznie szybszy sposób zrealizowana przez komputer.

Przeszukiwanie polegać będzie zatem na dobieraniu parametrów całego algorytmu maksymalizujących wartość funkcji celu - walidacji krzyżowej. Aby uniknąć przeuczenia i nadmiernego dopasowania ustawień do wprowadzonych danych, przeszukiwanie wykorzystywać będzie jedynie część danych. Wielkość tej części użytkownik będzie mógł ustalić samodzielnie w postaci liczby procentowej. Domyślną wartością tej części uczącej będzie 90%. Pozostałe dane zostaną użyte do ostatecznego określenia jakości parametrów algorytmu - odbędzie się to ponownie przy użyciu walidacji krzyżowej. Jak widać zatem, dane zostaną podzielone na zbiór uczący dla poszukiwań oraz na zbiór testowy. Zbiór uczący będzie dzielony na podzbiory uczące i testowe podczas określania jakości parametrów algorytmu w każdej iteracji.

Całkowita liczba wszystkich możliwych kombinacji metod przetwarzania i klasyfikacji zależy wykładniczo od liczby atrybutów (kolumn) danych. Przykładowo, liczba możliwych kombinacji atrybutów, które aplikacja podda normalizacji, wynosi 2^n, gdzie n oznacza liczbę kolumn. Aby uniknąć długiego czasu przeszukiwania związanego z eksplozją wykładniczą, użytkownik powinien być w stanie określić podzbiór wszystkich kombinacji, które program ma przeanalizować.

Dobierając metodę określania podprzestrzeni poszukiwań, należy osiągnąć kompromis pomiędzy precyzyjnością metody oraz stopniem jej skomplikowania. Dokładna metoda mogłaby wymagać od użytkownika bezpośredniego wprowadzania opisów kombinacji, które aplikacja miałaby obsłużyć. Jednakże przy gigantycznej liczbie wszystkich możliwych wariantów, rozwiązanie to nie znalazłoby zastosowania. Skrajną alternatywą mogłoby być pytanie użytkownika, czy przeszukiwanie powinno uwzględniać określone operacje, np. usuwanie kolumn lub ich kwantyfikację. Odpowiedzi „tak lub nie" byłyby wygodne dla odbiorcy, ale odbierałyby jakikolwiek wpływ na proces wyszukiwania.

Zaproponowane przeze mnie rozwiązanie stanowi złoty środek pomiędzy dwoma skrajnymi metodami. Polega ono na podziale przestrzeni poszukiwań na podprzestrzenie i na precyzowaniu każdej z osobna.

Na samym początku użytkownik powinien wybrać metody radzenia sobie z brakującymi danymi, na przykład: $\{\text{usuń wiersze}, \text{wartość średnia}\}$.

W następnej kolejności użytkownik powinien określić zbiór atrybutów R_{attr}, spośród których aplikacja może wybrać te, które zostaną usunięte. Dodatkowo użytkownik powinien określić zbiór liczb $\mathbf{R}$:

$$\mathbf{R} = \{r_1, r_2, \ldots, r_n\}$$

taki, że każda liczba r_i oznacza liczbę atrybutów, które algorytm będzie mógł usunąć jednocześnie ze zbioru R_{attr}.

Analogicznie powinno przebiegać określanie szczegółów normalizacji atrybutów. Użytkownik powinien wybrać zbiór atrybutów N_{attr} oraz wprowadzić zbiór liczb $\mathbf{N}$, którego każdy element oznaczać będzie liczbę atrybutów normalizowanych jednocześnie.

Opis podprzestrzeni przeszukiwań związanej z kwantyfikacją okazuje się bardziej skomplikowany. Podobnie jak poprzednio, odbiorca powinien wskazać zbiór atrybutów poddanych kwantyfikacji Q_{attr}. Następnie użytkownik powinien określić zbiór algorytmów kwantyfikacji

Q, które mogą być użyte. W kolejnym kroku należy wprowadzić zbiór liczb Q_{pass}. Każdy element zbioru oznaczać będzie liczbę jednoczesnych przebiegów kwantyfikacji. W dalszej kolejności odbiorca określi liczbę n_{max} oznaczającą maksymalną liczbę atrybutów, które mogą być łączone przez algorytmy ze zbioru **Q**. Na samym końcu osoba będzie musiała opisać podzbiór wszystkich kombinacji parametrów, jakie mogą przyjąć wybrane wcześniej algorytmy kwantyfikacji. Aby uprościć to zadanie, użytkownik wprowadzi tylko jedną liczbę g, oznaczającą *granularność* podprzestrzeni parametrów.

Wiele algorytmów często przyjmuje parametry początkowe należące do określonych dziedzin. Przykładami takich dziedzin mogą być: wartości „tak" i „nie", zbiór liczb rzeczywistych od 0.01 do 100. W ogólnym przypadku nie jest możliwe przeglądanie wszystkich kombinacji parametrów dla dowolnego algorytmu – jest ich zbyt dużo lub ich wartości są nieprzeliczalne. Z tego powodu, w mojej pracy przeszukiwanie przestrzeni ograniczam do jej wybranych elementów. Naturalnym podejściem do tego problemu jest wybór takich wartości z każdej dziedziny, które byłby dla niej reprezentatywne.

Granularność liniową g_l dla zbioru $\{a, \ldots, b\}$ definiuję jako liczność zbioru $\mathbf{L} = \{l_1, l_2, \ldots, l_n\}$ takiego, że:

$$l_1 = a$$
$$l_n = b$$
$$l_{i+1} - l_i = c$$

gdzie c jest stałą, a $i \in \{1, \ldots, n-1\}$. Intuicyjnie granularność liniowa to liczba od 1 większa od liczby podziałów zbioru na przedziały o jednakowej długości.

Granularność wykładniczą g_w dla zbioru $\{a, \ldots, b\}$ definiuję jako liczność zbioru $\mathbf{W} = \{w_1, w_2, \ldots, w_n\}$ takiego, że:

$$w_1 = a$$
$$w_n = b$$
$$w_{i+1} = w_i * q$$

gdzie q jest stałą, a $i \in \{1, \ldots, n-1\}$. Posługując się intuicją, granularność wykładniczą można rozumieć jako liczbę o 1 większą od liczby podziałów zbioru na przedziały należące do różnych rzędów wielkości. Zbiory **L** i **W** nazywam *zbiorami granularnymi*.

Granularnością nazywam granularność dziedziny każdego parametru algorytmu. Przeszukiwanie przestrzeni parametrów algorytmu sprowadzam zatem do przeglądania tylko tych

wartości, na jakie wskazuje granularność. Konkretniej oznacza to przeglądanie elementów iloczynu kartezjańskiego zbiorów granularnych każdego parametru algorytmu. Metoda ta wymaga wcześniejszego określenia dziedzin parametrów, ale nie stanowi to problemu, ponieważ zazwyczaj są one znane. Posługiwanie się granularnością pozwala zatem na ilościowe określenie dokładności przeszukiwań bez względu na to, z ilu parametrów dany algorytm korzysta.

Po zakończeniu ustalania szczegółów kwantyfikacji użytkownikowi pozostanie ustawienie procesu klasyfikowania danych. Odbiorca aplikacji będzie musiał wskazać algorytmy, których działanie ma zostać sprawdzone, jak również granularność dla tych algorytmów.

Podsumowując wymagania dotyczące przeszukiwania przestrzeni kombinacji, użytkownik będzie mógł:

1. Wskazać metody radzenia sobie z brakującymi danymi.
2. Wytypować możliwe kombinacje atrybutów przeznaczonych do usunięcia
3. Ustalić możliwe kombinacje atrybutów przeznaczonych do normalizacji
4. Określić kombinacje algorytmów łączenia atrybutów oraz ich parametry
5. Wskazać zbiór algorytmów klasyfikacji oraz ich parametry

3.1.3 Opis systemu komputerowego

Biorąc pod uwagę różnorodność popularnie używanych systemów operacyjnych, projektowany system nie powinien wymuszać na użytkowniku żadnej konkretnej platformy. Projektowane rozwiązanie powinno działać równie dobrze zarówno na systemach z rodziny MS Windows, jak i Linux (na przykład). Jednocześnie aplikacja powinna umożliwiać przetwarzanie, klasyfikowanie danych oraz przeszukiwanie w prosty i intuicyjny sposób. W tym celu system powinien oferować przejrzysty i nieskomplikowany graficzny interfejs użytkownika. Interfejs musi pozwalać na wprowadzanie danych do przetwarzania oraz na łatwe ustawianie parametrów opisanych w poprzednich częściach rozdziału.

Projektowane narzędzie powinno radzić sobie z odczytem danych zgromadzonych w popularnych formatach plików, np. MS Excel lub CSV (*Comma Separated Values*). Z racji tego, że wprowadzane dane poddawane są algorytmom klasyfikacji, plik powinien posiadać wyszczególnioną kolumnę zawierającą informacje o kategorii obserwacji. Nagłówek tej kolumny powinien nosić nazwę *Index*.

Zadaniem systemu jest ułatwienie odnajdywania ustawień algorytmu klasyfikacji. Typowa sesja pracy z aplikacją polegać powinna na wyszukiwaniu najlepszych ustawień „na chybił trafił" lub automatycznie (przeszukiwanie). W obu wariantach pożyteczne byłoby przechowywanie informacji o wynikach wcześniejszych operacji. Mając łatwy dostęp do wcześniejszych rezultatów, użytkownik mógłby w prostszy sposób wnioskowań na temat skuteczności użytych metod. Biorąc to pod uwagę, aplikacja powinna zapisywać informacje o wynikach przeprowadzonych operacji. W przypadku przetwarzania i klasyfikacji, system zapamiętywałby wprowadzone parametry oraz wynik walidacji krzyżowej. W przypadku procesu przeszukiwania, aplikacja zapisywałaby dwie rzeczy. Po pierwsze, opis podprzestrzeni przeszukiwań. Po drugie, najlepszy znaleziony wynik oraz opis metody, która do niego prowadzi. Aplikacja powinna umożliwiać również usuwanie wybranych zapisów.

Biorą pod uwagę możliwie długi czas przeszukiwania, system komputerowy musi pozwalać na wykonywanie tej operacji w tle. Po uruchomieniu przeszukiwania użytkownik powinien być w stanie wykonywać inne operacje w systemie. Stąd też kolejne wymaganie – system powinien umożliwiać przeprowadzanie wielu przeszukiwań jednocześnie. Oprócz tego, system powinien pozwalać odbiorcy na podglądanie aktualnego postępu operacji. W dowolnym momencie użytkownik powinien móc dowiedzieć się, jaka część wszystkich wybranych kombinacji została już sprawdzona.

Dodatkowo system nie powinien wymagać dostępu do Internetu. Aplikacja powinna być w stanie wykonać wszystkie konieczne obliczenia na pojedynczym komputerze, nie korzystając z sieci.

Poza wyżej wymienioną funkcjonalnością system komputerowy powinien oferować nieskomplikowaną metodę instalacji.

3.2 Wymagania niefunkcjonalne

Określenie wymagań jakościowych jest bardzo istotne, ponieważ definiują one zakres prawidłowej funkcjonalności systemu.

System komputerowy powinien umożliwiać wykonywanie co najmniej 20 operacji przeszukiwań jednocześnie. Wymaganie to pokrywa większość scenariuszy użycia operacji przeszukiwania. Nie spodziewamy się bowiem, aby użytkownik uruchamiał ponad kilkadziesiąt operacji w tym samym czasie.

Aplikacja powinna pozwalać na zapisanie co najmniej 100000 wyników klasyfikacji oraz 100000 wyników przeszukiwań. Takie wymaganie gwarantuje, że odbiorca, będący wielkim

entuzjastą uczenia maszynowego, mógłby dokonywać 50 badań każdego typu codziennie przez ponad pięć lat[1].

Ze względu na to, że posiadane przeze mnie dane zostały dostarczone w postaci pliku MS Excel, aplikacja powinna potrafić odczytywać pliki tego typu. Odczyt danych z innych rodzajów plików nie jest konieczny.

Ostatnim wymaganiem jest określenie liczby algorytmów klasyfikacji i łączenia, jakie aplikacja będzie oferowała. Aby umożliwić użytkownikowi szeroki wybór, aplikacja pozwalać będzie na skorzystanie z co najmniej pięciu algorytmów klasyfikacji oraz tej samej liczby metod klasteryzacji.

3.3 Wymagania ograniczeń

Projektując oprogramowanie, należy uwzględnić obecność wielu czynników ograniczających. Do czynników ograniczających można zaliczyć ograniczenia finansowe, związane z określonym budżetem przeznaczonym na realizację projektu lub jego określonej fazy. Innym rodzajem ograniczenia jest specyfika środowiska uruchomieniowego. Aplikacje wykonujące przetwarzanie rozproszone mogą wymagać określonej architektury sieci, podczas gdy niektóre systemy wbudowane potrafią działać poprawnie tylko na dedykowanych platformach sprzętowych. Ograniczenia nałożone na projekt mają często bezpośredni wpływ na architekturę końcowego rozwiązania i dlatego są równie istotne jak wymagania funkcjonalne i jakościowe.

Podstawowym ograniczeniem w mojej pracy środowisko uruchomieniowe – jest nim pojedynczy komputer PC. Poprawne działanie aplikacji nie wymaga pracy wielu komputerów, wystarczyć powinna jedna maszyna. Wymaganie to gwarantuje, że odbiorca będzie mógł korzystać z systemu np. na laptopie, będąc w podróży.

3.4 Podsumowanie wymagań

Tabela 3.1 stanowi zestawienie wymagań projektowych opisanych w tym rozdziale.

[1] Pięćdziesiąt operacji przez trzysta sześćdziesiąt pięć dni w roku przez pięć lat daje $50 * 365 * 5 = 91250$ przeprowadzonych operacji, czyli nieco mniej niż 100000.

Tablica 3.1. Wymagania projektowe

Rodzaj wymagania	Opis wymagania
Funkcjonalne	Przetwarzanie i klasyfikacja zgodnie z ustawieniami użytkownika
Funkcjonalne	Przeszukiwanie przestrzeni kombinacji metod przetwarzania i klasyfikacji zgodnie z ustawieniami użytkownika
Funkcjonalne	Praca pod systemami operacyjnymi z rodziny MS Windows i Linux
Funkcjonalne	Prosty i intuicyjny interfejs graficzny
Funkcjonalne	Odczyt danych posiadających wyszczególnioną kolumnę kategorii
Funkcjonalne	Zapisywanie rezultatów przetwarzania: ustawień algorytmu i wyniku walidacji krzyżowej
Funkcjonalne	Zapisywanie rezultatów przeszukiwania: ustawień przeszukiwania oraz nastaw najlepszej znalezionej metody oraz wyniku walidacji krzyżowej dla tej metody
Funkcjonalne	Uruchamianie wielu operacji przeszukiwania w tle
Funkcjonalne	Aktualizowanie informacji o postępie procesu przeszukiwania
Funkcjonalne	Nieskomplikowana metoda instalacji
Funkcjonalne	Aplikacja nie wymaga dostępu do Internetu
Niefunkcjonalne	Wykonywanie 20 operacji przeszukiwania jednocześnie
Niefunkcjonalne	Zapisywanie co najmniej 100000 wyników klasyfikacji oraz 100000 wyników przeszukiwań
Niefunkcjonalne	Obsługa formatu MS Excel
Niefunkcjonalne	Wybór spośród co najmniej 5 algorytmów klasyfikacji
Niefunkcjonalne	Wybór spośród co najmniej 5 algorytmów łączenia atrybutów
Ograniczeń	Aplikacja przeznaczona na pojedynczy komputer PC

4. Projekt systemu komputerowego

W tym rozdziale skupiam się nad projektem systemu komputerowego ułatwiającego zadania klasyfikacji danych.

4.1 Projekt narzędzia

4.2 Projekt interfejsu sieciowego

4.3 Projekt testów

4.3.1 Testy narzędzia

4.3.2 Testy interfejsu

5. Elementy implementacji systemu

5.1 Implementacja narzędzia

5.2 Implementacja interfejsu webowego

6. Testy systemu

6.1 Testy narzędzia

6.2 Testy interfejsu webowego

7. Wyniki badań

8. Wnioski

Bibliografia