SYSTEM EKSPERCKI - DOKUMENTACJA

1. Cel projektu

Celem projektu było stworzenie systemu, który potrafi podjąć decyzję na jakiś temat na podstawie zestawu reguł i wiedzy pobranej od użytkownika, przy czym dziedzina problemu nie może znajdować się w kodzie. Ponadto, należało zapewnić możliwość odczytu i zapisu bazy wiedzy do pliku w wybranym formacie (w moim przypadku są to pliki csv).
Zastosowany przeze mnie algorytm to drzewo decyzyjne.

2. Opis projektu

A) plik ‘tree.py’

W tym pliku znajduje się najważniejsza część projektu - drzewo decyzyjne. Składa się ono z:
-Klasa Node - W schematycznym ujęciu drzewa decyzyjnego jest to punkt węzłowy - rozgałęziający lub końcowy, w zależności od informacji, które przechowuje. Każdy węzeł rozgałęziający przechowuje informację o zmiennej, według której podzielone są dane. Węzły rozgałęziające, które dzielą dane według zmiennej o wartościach nienumerycznych, przechowują informacje o tylu węzłach, ile unikalnych wartości ma ta zmienna (każdy kolejny węzeł został stworzony wyłącznie z porcji danych, w której wszystkie wartości tej zmiennej są identyczne). Natomiast węzły decyzyjne dzielące dane według zmiennej o wartości numerycznej przechowują informację o progu liczbowym, według którego dane zostały podzielone na te o wartościach zmiennej mniejszych bądź równych i większych, oraz o kolejnych węzłach stworzonych na podstawie tego podziału. Węzły końcowe przechowują jedynie podjętą decyzję.
-Klasa DecisionTree - klasa tworząca drzewo decyzyjne na podstawie otrzymanego zestawu danych. Przechowuje informacje pomagające w procesie budowy drzewa, a także zapisuje informacje o korzeniu drzewa. Budowa drzewa polega na dzieleniu danych na mniejsze porcje i tworzeniu kolejnych węzłów, dopóki porcja danych nie będzie czysta - wszystkie decyzje będą identyczne (lub nie zostanie osiągnięta maksymalna “głębokość” - dozwolona ilość rozgałęzień, którą stanowi ilość zmiennych niewynikowych w bazie reguł, jeśli taka opcja zostanie odblokowana w kodzie programu). Dzieje się to przez próbę podziału zestawu danych według każdej zmiennej i wybraniu tego, który powoduje największy przyrost informacji. Przyrost informacji informuje o zmianie czystości zestawu danych po podzieleniu go na mniejsze części. Zazwyczaj jest on stosowany w parze z entropią, jednak ja wykorzystałem indeks GINI (indeks nieczystości), gdyż jego obliczenie jest zdecydowanie prostsze (wzór to: 1 - suma prawdopodobieństw otrzymania możliwych rezultatów podniesionych do kwadratu, podczas gdy wzór na entropię stosuje logarytm o podstawie 2), a przynosi ten sam efekt. Ponadto, klasa ta zawiera metodę pozwalającą na obliczenie dokładności odpowiedzi drzewa - podejmuje decyzje dla każdej reguły zapisanej w bazie, a następnie porównuje je z prawidłowymi decyzjami i oblicza stosunek poprawnie podjętych decyzji do liczby wszystkich reguł w bazie.

B) plik ‘data_io.py’

W tym pliku znajdują się funkcje do zapisu i odczytu bazy reguł z pliku. Dodatkowo, funkcja pobierająca dane z pliku pomija nieprawidłowe linie - reguły z brakującymi danymi lub zbyt wieloma zmiennymi - oraz duplikaty reguł.

C) plik ‘main.py’

Jest to plik wykonawczy programu. Służy do komunikacji z użytkownikiem, pozwala mu m. in. na wczytanie bazy reguł i ewentualne uzupełnienie jej o nowe reguły lub stworzenie jej od zera, zapisanie zaktualizowanej bazy do pliku, wielokrotne podejmowanie decyzji na podstawie odpowiedzi na zadane mu pytania, a także zapewnia możliwość uzupełnienia bazy o nową regułę przez użytkownika w przypadku, gdy podjęta decyzja jest błędna.

D) biblioteki

W programie zostały wykorzystane dwie biblioteki niestandardowe:
-Pandas - biblioteka opierająca się na bibliotece NumPy. Wykorzystywana w projekcie głównie do odczytu i zapisu danych do pliku csv, a także tworzenia i uzupełniania bazy reguł przy użyciu obiektu klasy DataFrame. Używana baza reguł jest przekazywana do drzewa decyzyjnego, gdzie zostaje przetworzona na szereg danych biblioteki NumPy.
-NumPy - biblioteka szeroko wykorzystywana przeze mnie przy tworzeniu drzewa decyzyjnego. Pozwala na łatwe operowanie danymi oraz zapewnia prosty dostęp do szukanych wartości.

3. Podsumowanie

W ramach projektu stworzyłem algorytm drzewa decyzyjnego, interaktywną funkcję podejmującą decyzję na podstawie odpowiedzi na pytania zadane użytkownikowi, zapewniłem możliwość odczytu i zapisu bazy reguł do pliku w formacie csv, a także umożliwiłem użytkownikowi uzupełnienie bazy lub stworzenie jej od podstaw, realizując tym samym wszystkie założenia projektu.
W trakcie realizacji projektu trafiłem na wiele przeszkód, przez które mój pierwotny pomysł na jego wykonanie zmienił się. Początkowo nie planowałem używać innych bibliotek niż standardowe, jednak okazało się to praktycznie niemożliwe. Odczytywanie danych miało opierać się na zwykłym pobieraniu informacji z pliku csv i zapisywaniu ich do klasy Rule, która zawierałaby decyzję i słownik z wartościami pozostałych zmiennych. Jak się później okazało, realizacja była bardzo skomplikowana i nieczytelna, a dodatkowo wszystkie wartości odczytane z pliku były zapisywane jako string, co uniemożliwiało podział według zmiennej numerycznej na wartości mniejsze bądź równe i większe. Z tych powodów zrezygnowałem z tego pomysłu i zdecydowałem się na skorzystanie z biblioteki pandas, której elementem jest obiekt klasy DataFrame, będący czymś na kształt tablicy i umożliwiający przechowywanie w nim wszystkich reguł jednocześnie, dodatkowo automatycznie dobierając odpowiednie typy do zmiennych. Ponadto, planowałem cały algorytm drzewa wykonać na zwykłych funkcjach, lecz szybko okazało się, że wymagałoby to przekazywania coraz większych ilości zmiennych jako parametrów do kolejnych funkcji. Dlatego ostatecznie zdecydowałem się na utworzenie klasy DecisionTree, co pozwoliło na znaczne zredukowanie liczby parametrów wykorzystywanych metod, które miały dostęp do atrybutów drzewa, poprawiając tym samym czytelność kodu.
Uważam, że mój projekt jest dobry, ponieważ pozwala użytkownikowi na podjęcie decyzji według określonych reguł, często bez konieczności podawania wartości wszystkich zmiennych. Jest to o wiele prostsze rozwiązanie niż ręczne szukanie odpowiedzi w bazie reguł, szczególnie jeśli zawiera ona wiele pozycji.

Paweł Wysocki