

**WYDZIAŁ GEOLOGII, GEOFIZYKI I OCHRONY ŚRODOWISKA**

KATEDRA GEOINFORMATYKI I INFORMATYKI STOSOWANEJ

Projekt inżynierski

*Pakiet w języku Python do optymalizacji procesów analizy danych tabelarycznych oraz modelowania regresyjnego  
i klasyfikacyjnego z wykorzystaniem AI*

*Python package for optimizing processes of tabular data analysis, regression and classification modeling using AI*

Autor: *Filip Andrzej Hałys*

Kierunek studiów: Geoinformatyka

Opiekun pracy: *Dr inż. Monika Chuchro*

Kraków, 2024

**Spis treści**

[1. WSTĘP 3](#_Toc180519948)

[1.1. Cel 3](#_Toc180519949)

[1.2. Motywacja 3](#_Toc180519950)

[2. TECHNOLOGIA I METODY 4](#_Toc180519951)

[3. OGÓLNY OPIS PAKIETU 5](#_Toc180519952)

[4. SZCZEGÓŁOWY OPIS POSZCZEGÓLNYCH SUBPAKIETÓW 6](#_Toc180519953)

[5. PRZYKŁADY UŻYCIA 7](#_Toc180519954)

[6. PODSUMOWANIE, DYSKUSJA, WNIOSKI 8](#_Toc180519955)

[7. LITERATURA 9](#_Toc180519956)

# WSTĘP

Język Python jest jednym z najczęściej aktualnie wykorzystywanych języków programowania. W wielu rankingach znajduje się nawet na 1 miejscu; chociażby według TIOBE INDEX nieprzerwanie od ponad roku Python dzierży miano najbardziej popularnego języka programowania. Swoją pozycję zawdzięcza niewątpliwie dzięki prostej składni, szerokiemu spektrum zastosowań, czy rozwojowi sztucznej inteligencji. Relację pomiędzy rozwojem sztucznej inteligencji i językiem Python można określić jako mutualizm. Zdecydowanie tak szerokie wykorzystywanie modeli AI w dzisiejszym świecie jest spowodowane łatwością ich tworzenia, czy implementacji. W tworzeniu tego typu rozwiązań znacznie pomaga język Python i dedykowane dla niego pakiety (między innymi PyTorch, scikit-learn, czy Keras). Z drugiej strony Python również korzysta z przyspieszającego tempa rozwoju AI. Dzięki stworzonym pakietom do tworzenia sieci neuronowych, dużych modeli językowych, czy rozwiązywania problemów uczenia maszynowego Python znacznie zyskał na popularności. Wymienione wcześniej pakiety, to tylko przykłady gotowych rozwiązań, z których użytkownik może korzystać w celu budowy modelów AI. Takich pakietów istnieje bardzo dużo, co więcej atutem Pythona jest możliwość tworzenia własnych, personalizowanych pakietów dedykowanych dla określonych problemów.

# Cel

Podstawowym celem tego projektu inżynierskiego było utworzenie własnego, personalizowanego pakietu. Jako temat przewodni pakietu wybrano szeroko pojęty proces analizy danych oraz powiązany z nim proces modelowania regresyjnego i klasyfikacyjnego, za pomocą wybranych algorytmów uczenia maszynowego. Skupiono się głównie na optymalizacji tychże procesów. Poprzez optymalizację rozumie się ułatwienie, przyspieszenie i poukładanie składowych tychże procesów.

## Motywacja

Głównym czynnikiem wpływającym na wybór tego tematu była ciekawość, w jaki sposób przebiega proces tworzenia pakietu od samego początku do finalnego etapu, w którym użytkownik jest w stanie ów pakiet wykorzystać.

# TECHNOLOGIA I METODY

W tej sekcji skupiono się na wykorzystanych technologiach w projekcie, a także opisie użytych algorytmów uczenia maszynowego.

## Środowisko programistyczne

Do implementacji kodu posłużono się środowiskiem PyCharm od firmy JetBrains. Jest to jedno z dwóch głównych narzędzi wykorzystywanych do tworzenia oprogramowania w języku Python obok programu Microsoft Studio Code. Wybrano akurat to środowisko, gdyż w przeciwieństwie do innych zapewnia ono pomoc przy pisaniu dobrze wyglądającego i czytelnego kodu.

## Środowisko wirtualne

W projekcie wykorzystano również możliwość utworzenia własnego środowiska wirtualnego. Zainicjowano je za pomocą narzędzia „anaconda3”. Jest to popularne narzędzie wspomagające zachowanie ładu i porządku w projektach. Za jego pomocą tworzenie własnych środowisk dedykowanych pod konkretny problem czy projekt jest proste i bardzo szybkie.

Na początku w aplikacji Anaconda Navigator utworzono środowisko o nazwie ENV-FOR-ET i zapisano je w lokalizacji ‘~\anaconda3\envs\ENV-FOR-ET’. Następnie dostosowano interpreter Pythona w Pycharmie, w taki sposób aby kompilowany kod wywoływał się przy pomocy utworzonego wcześniej środowiska. Po konfiguracji interpretera utworzono plik typu yaml; environment.yml. Celem jego stworzenia było ułatwienie zarządzania pakietami wewnątrz środowiska wirtualnego. Poniżej na Rys. 1 zaprezentowano jego zawartość:



Rys. 1 – zawartość pliku environemnt.yml

W kolejnym etapie zainstalowano narzędzie conda-lock. Jego zdaniem jest blokowanie wersji pakietów, co może być pomocne przy próbie odtworzenia jeden do jeden środowiska, np. na innym urządzeniu. Po instalacji zablokowano wersje pakietów. W celu instalacji i stworzenia blokady wykorzystano polecenia:

*conda install -c conda-forge conda-lock* – instalacja narzędzia cond-lock  
*conda-lock lock --file environment.yml* – utworzenie blokady

Początkowo nie zadeklarowano wszystkich niezbędnych pakietów w sekcji ‘dependencies’. Kolejne pakiety dołączano do środowiska wraz z rozwojem projektu przy pomocy poleceń:

*conda activate base* – aktywowanie domyślnego środowiska Condy

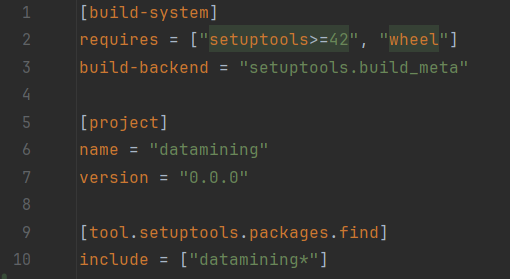
*conda-lock -f environment.yml –* stworzenie na nowo pliku conda-lock.yml

*conda-lock install --name ENV-FOR-ET conda-lock.yml* – instalacja pakietów za pomocą pliku conda-lock.yml

## Tworzenie pustego pakietu

Po przygotowaniu środowiska do pracy utworzono pusty pakiet. W tym celu wykonano trzy kroki. Po pierwsze utworzono pusty folder, będący głównym folderem pakietu. Nadano mu nazwę „datamining”. Następnie utworzono plik pyproject.toml, którego zawartość zaprezentowano na Rys. 2:

*Rys. 2 – zawartość pliku pyproject.toml*



W pliku tym zadeklarowano nazwę pakietu (zgodną z nazwą folderu) oraz jego wersję (0.0.0). Wskazano również, iż zawartość pakietu jest kompatybilna z zawartością utworzonego w kroku pierwszym folderu. Finalnie wywołano polecenie: ‘pip install -e .’ w celu instalacji pakietu.

Od tego momentu każda napisana funkcja lub klasa obiektów wewnątrz folderu datamining stała się składową tegoż pakietu.

## System kontroli wersji

W trakcie tworzenia pakietu wykorzystywano system kontroli wersji GIT. Na początku utworzono repozytorium wykorzystując portal Github. Następnie sukcesywnie wraz z rozwojem pakietu dodawano do niego wprowadzane zmiany. Pozwoliło to na skuteczne i uporządkowane zarządzanie rozwojem projektu. Utworzenie repozytorium miało również dodatkowy cel; umożliwić udostępnienie funkcji wchodzących w skład pakietu innym użytkownikom.

# OGÓLNY OPIS PAKIETU

Pakiet nazwano ‘datamining’. W jego skład wchodzi dokładnie 6 podmodułów:

* **\_errors** – podmoduł pomocniczy, niewidzialny z perspektywy użytkownika, głównie przeznaczony do wyświetlania informacji o błędach popełnionych przez użytkownika w momencie tworzenia obiektu klasy lub wywoływania funkcji,
* **check** – podmoduł służący do zaznajamiania się z ramką danych przez użytkownika, przechowujący funkcje wyświetlające podstawowe statystyki badanych danych,
* **preprocessing** – niewielki podmoduł odpowiedzialny za przeprowadzanie preprocessingu, czyli wstępnego ‘czyszczenia’ danych,
* **normalizations** – rozbudowany podmoduł przechowujący funkcje normalizacyjne zestaw danych
* **regression** – podmoduł zbudowany z dwóch dużych klas, dzięki którym możliwe jest optymalizowanie tworzenia wydajnych modeli regresji liniowej
* **classification** – podmoduł zbudowany z jednej klasy, dzięki której możliwe jest tworzenie obiektów w postaci dobrej jakości klasyfikatorów

Tutaj jeszcze: wykorzystane pakiety (pandas, numpy itp), opis tego że wszędzie dodawano komentarze i nazwy zmiennych, klas , opisy itp. Są w języku ang bo jest to dominujący język w branży

# SZCZEGÓŁOWY OPIS POSZCZEGÓLNYCH PODMODUŁÓW

## PODMODUŁ ‘\_errors’

Podmoduł ten utworzono w celu przechowywania metod obsługujących błędy w przekazywaniu parametrów przez użytkownika podczas tworzenia instancji klas lub wywoływania innych metod znajdujących się w obrębie całego pakietu. Innymi słowy podmoduł ten jest podmodułem pomocniczym pozwalającym na zatrzymywanie działania metody bądź tworzenia instancji klasy. Zatrzymanie to odbywa się w momencie gdy użytkownik jako argument do wywoływanej metody lub tworzonej klasy poda parametr o innym typie niż oczekiwany. Wewnątrz tego podmodułu utworzono dwie funkcje; jedną w postaci dekoratora, drugą w postaci zwykłej metody.

Pierwsza z nich (dekorator) „\_validate\_method\_argument\_types” pozwala na wyświetlenie błędu o treści: „Argument <nazwa argumentu> must be of type <oczekiwany typ>, got <aktualny typ> instead.”. Dzięki podniesieniu błędu o takiej treści użytkownik popełniający błąd przy wywoływaniu metody wchodzącej w skład pakietu jest w stanie zweryfikować, który podany przez niego argument ma niepoprawny typ. Podniesienie błędu powoduje bezwarunkowe zatrzymanie tej metody i niewykonywanie jej dalszego działania.

Drugą metodę „\_validate\_class\_argument\_types” zaimplementowano nieco inaczej, nie jako dekorator, a jako zwykłą funkcję. Przyjmuje ona jeden argument w postaci słownika, w którego skład wchodzą klucze i wartości. Klucze określają nazwę argumentu, wartości natomiast są dwuelementową listą, w skład której wchodzi oczekiwany typ (element listy o indeksie 0) i przekazany typ przez użytkownika (element listy o indeksie 1). Zarówno klucz, jak i typy argumentów przekazane w liście są przekazywane jako tekst (string). Funkcja ta wywoływana jest w konstruktorze klasy. Użytkownik podając nieodpowiedni typ jednego z parametrów podczas tworzenia instancji klasy (w której zaimplementowano wywołanie tej metody pomocniczej) jako wynik otrzyma błąd o analogicznej treści jak w przypadku powyższego dekoratora. Stworzenie instancji się nie powiedzie.

# PRZYKŁADY UŻYCIA

# PODSUMOWANIE, DYSKUSJA, WNIOSKI

# LITERATURA

https://www.tiobe.com/tiobe-index/