**Dokumentacja AmIPhat**

**Spis treści**

[1. Przegląd 1](#_Toc2119088173)

[2. Uzasadnienie funkcjonalności 2](#_Toc214707397)

[3. Predykcja 2](#_Toc1966685765)

[4. Dataset 3](#_Toc1533949753)

[4.1 Opis wykorzystania Datasetu 3](#_Toc157465285)

[5. Konfiguracja 4](#_Toc1559770218)

[6. Model uczenia maszynowego 5](#_Toc326278440)

[7. Opis tworzenia modelu 5](#_Toc1244308466)

[8. API 5](#_Toc1758622615)

[9. Szczegóły Konfiguracji Docker 6](#_Toc1079754379)

[10. Struktura projektu 6](#_Toc1888420005)

[11. Dodatkowe Informacje 7](#_Toc1363264992)

## **1. Przegląd**

**AmIPhat** to aplikacja webowa zaprojektowana do przewidywania poziomu otyłości na podstawie danych wprowadzonych przez użytkownika. Aplikacja działa na najnowszych przeglądarkach internetowych, co zapewnia jej kompatybilność ze wszystkimi aktualnymi urządzeniami.

Główne funkcje aplikacji obejmują:

* Wprowadzenie danych użytkownika, takich jak waga, wzrost, wiek, płeć oraz poziom aktywności fizycznej.
* Przetwarzanie i analiza wprowadzonych danych w celu przewidzenia poziomu otyłości użytkownika.
* Wyświetlanie wyników wraz z zaleceniami dotyczącymi zdrowego stylu życia i ewentualnych zmian, które mogą pomóc w utrzymaniu lub poprawie zdrowia.

Aplikacja składa się z pokazowego programu wykorzystującego hosting Streamlit oraz z kontenerów Docker w celu trenowania i wykorzystania modelu. Dzięki zastosowaniu Dockera możliwe jest uruchomienie aplikacji bez konieczności doinstalowywania dodatkowych pakietów.

## **2. Uzasadnienie funkcjonalności**

AmIPhat służy wielu celom:

1. **Specjaliści z dziedziny zdrowia:**
   * Lekarze i dietetycy mogą wykorzystać aplikację do oceny ryzyka otyłości u swoich pacjentów.
   * Szybka prognoza otyłości umożliwia pracownikom służby zdrowia szybsze wdrożenie planu leczenia.
2. **Użytkownicy indywidualni:**
   * Osoby mogą samodzielnie ocenić, czy ich obecny styl życia naraża je na ryzyko otyłości.
   * Aplikacja może zachęcić użytkowników do podejmowania bardziej świadomych decyzji dotyczących diety i stylu życia.

## **3. Predykcja**

Główna funkcjonalność aplikacji polega na przewidywaniu stopnia otyłości na podstawie danych wprowadzonych przez użytkownika.

Obejmuje następujące etapy:

1. Wprowadzenie danych: Użytkownik wprowadza swoje dane za pomocą interfejsu użytkownika.
2. Przetwarzanie: Aplikacja przetwarza wprowadzone dane.
3. Analiza i obliczenia: Za pomocą modelu, aplikacja analizuje dane użytkownika i oblicza przewidywany stopień otyłości.
4. Prezentacja wyników: Wyniki prezentowane są użytkownikowi.

## **4. Dataset**

Aplikacja wykorzystuje zbiór danych "Obesity Levels" pobrany z Kaggle. Zbiór ten składa się z 16 zmiennych niezależnych oraz jednej zmiennej zależnej, reprezentującej poziom otyłości.

Poniżej przedstawione są wszystkie zmienne z datasetu:

* Gender - płeć
* Age – wiek
* Height – wzrost
* Weight – waga
* Family\_history\_with\_overweight - obecność nadwagi w historii rodzinnej
* FAVC - częstość spożycia jedzenia wysokokalorycznego
* FCVC - częstość spożycia warzyw
* NCP – liczba dziennie spożywanych posiłków głównych
* CAEC - częstość podjadania między posiłkami
* SMOKE – czy osoba jest palaczem
* CH2O - ilość dziennego spożycia wody
* SCC – czy osoba liczy spożywane kalorie
* FAF - częstość aktywności fizycznej
* TUE – czas spędzony przy korzystaniu z elektroniki
* MTRANS – główny sposób transportu
* NObesity – poziom otyłości **(Zmienna zależna)**

### **4.1 Opis wykorzystania Datasetu**

Zbiór danych "Obesity Levels" pobrany z Kaggle został załączony do repozytorium. Po wczytaniu dane są przetwarzane w celu usunięcia brakujących wartości i normalizacji. Proces ten obejmuje następująco:

* Pobieranie danych (download\_dataset.py) - odpowiada za pobranie danych z zewnętrznego źródła. Wykorzystuje bibliotekę opendatasets.
* Czyszczenie danych (clean\_data.py) - odpowiada za proces czyszczenia, usuwa duplikaty.
* Wstępne przetwarzanie (preprocess\_data.py) - dodaje dodatkową zmienną niezależną BMI wyliczoną na podstawie Height i Weight.
* Przygotowanie danych (prepare\_data.py) - przekształca dane na format, który jest wymagany do trenowania modelu (zastępuje dane kategoryczne liczbami).
* Podział na zestawy treningowe i testowe (split\_data.py) - odpowiada za podział na zestawy treningowe i testowe.

## **5. Konfiguracja środowiska Conda**

Składa się ona z dwóch głównych elementów w katalogu conf: skryptu PowerShell do tworzenia środowiska oraz pliku konfiguracyjnego.

Skrypt create\_env\_conda.ps1 automatyzuje proces tworzenia środowiska Conda. Wykorzystuje plik environment.yml, który zawiera wszystkie zależności potrzebne do uruchomienia projektu.

Plik environment.yml zawiera wszystkie zależności, które są wymagane do uruchomienia projektu. Dzięki temu możemy łatwo odtworzyć środowisko na różnych komputerach, zapewniając spójność wersji bibliotek i narzędzi.

## **6. Model uczenia maszynowego**

Aplikacja wykorzystuje model drzewa decyzyjnego, popularny algorytm uczenia maszynowego. Model ten jest przydatny do analizy danych zarówno kategorycznych, jak i numerycznych, dostarczając jasnych i interpretowalnych wyników.

## **7. Opis tworzenia modelu**

* Trenowanie modelu (training\_model.py) - trenuje model klasyfikacyjny na podstawie dostarczonych danych treningowych
* Ocena Modelu (evaluate\_model) - przeprowadza predykcję wykorzystując testowy zbiór.
* Zapis Modelu (save\_model) - zapisuje wytrenowany model.
* Wyników Modelu (show\_model\_data) - wyświetla dokładność modelu oraz raport klasyfikacji.

## **8. API**

W tej sekcji opisane są pliki i moduły związane z implementacją API dla projektu. API zostało zaimplementowane przy użyciu frameworku FastAPI oraz obsługuje serwery FastAPI i Streamlit.

* FastAPI jest używane do tworzenia serwera API, który obsługuje zapytania HTTP i zwraca odpowiedzi na podstawie wytrenowanego modelu uczenia maszynowego.
* Streamlit jest używane do tworzenia interfejsu webowego, który umożliwia interakcję z modelem uczenia maszynowego w przeglądarce.

## **9. Szczegóły Konfiguracji Docker**

Aplikacja może być uruchomiona w kontenerze Docker. Plik Dockerfile zawiera instrukcje do budowania obrazu Docker. Instrukcje dotyczące zbudowania i uruchomienia kontenerów znajdują się w pliku README.

Kontenery mogą pobierać i zapisywać dane na zewnątrz poprzez montowanie folderów podczas uruchomienia.

## **10. Struktura projektu**

Projekt jest zorganizowany w następujący sposób:

* **API:**
  + fastapi: Pliki serwera FastAPI.
    - libs:
      * load\_model.py: pakiet wczytujący model uczenia maszynowego
    - ml\_model: folder z którego wczytywany jest model
    - app.py: skrypt uruchamiający API
    - Dockerfile: konfiguracja kontenera Docker dla API
    - example\_request.json: przykładowy payload do zapytania do API
    - requirements.txt: lista pakietów potrzebnych do zbudowania kontenera
  + streamlit: Pliki serwera Streamlit dla interfejsu webowego.
    - app.py: moduł uruchamiający aplikacje streamlit
    - model.joblib: model
    - requirements.txt: plik konfiguracyjny dla streamlit
* Training:
  + **dataset:**
    - [ObesityDataSet\_raw\_and\_data\_sinthetic.csv](https://github.com/BrainTireFire/PJA-SUML-14C-GR4/blob/docs/dataset/obesity-levels/ObesityDataSet_raw_and_data_sinthetic.csv): Pliki datasetu "Obesity Levels".
  + **models:**
    - model.joblib: Zserializowany model drzewa decyzyjnego.
  + **src:**
    - main.py: Główny skrypt do uruchomienia aplikacji.
    - modules: Kod źródłowy do przetwarzania danych i zadań z zakresu nauki o danych.
      * data\_processing: Skrypty do przetwarzania datasetu.
        + clean\_data.py: Oczyszcza surowe dane.
        + download\_dataset.py: Pobiera dataset z Kaggle.
        + prepare\_data.py: Przygotowuje dane do treningu.
        + preprocess\_data.py: Przetwarza dane.
        + split\_data.py: Dzieli dane na zbiory treningowe i testowe.
      * data\_science: Skrypty do treningu i ewaluacji modelu uczenia maszynowego.
        + evaluate\_model.py: Ewaluacja wytrenowanego modelu.
        + save\_model.py: Zapisuje wytrenowany model.
        + show\_model\_data.py: Wizualizacja danych modelu.
        + training\_model.py: Trenuje model uczenia maszynowego.
* **conf:**
  + create\_env\_conda.ps1: Skrypt do tworzenia środowiska Conda.
  + environment.yml: Plik konfiguracyjny dla środowiska Conda.
* **docs**:
  + Dokumentacja\_AmlPhat\_1\_1.docx: Plik z dokumentacją projektu.
* .gitignore: Określa, które pliki i katalogi mają być ignorowane w kontroli wersji.
* README.md: Zawiera przegląd i instrukcje dotyczące projektu.

## **11. Dodatkowe Informacje**

Aby uzyskać więcej informacji na temat instalacji, konfiguracji i użytkowania aplikacji, zapoznaj się z plikiem README.md w repozytorium projektu.

Pliki odpowiadające za dokumentacje projektu znajdują się w folderze docs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data | Autor | Zakres | Zatwierdził | Wersja |
| 01.06.2024 | Norbert Isański | Sporządzenie dokumentacji projektowej | Maciej Kawęcki | 1.0 |
| 07.06.2024 | Maciej Romaniuk | Poprawki do dokumentu | Maciej Kawęcki | 1.1 |