Dokumentacja Techniczna Projektu AI: heart-attack-prediction

Wprowadzenie

Cel Projektu

Celem projektu **heart-attack-prediction** jest stworzenie aplikacji, która potrafi przewidywać ryzyko wystąpienia ataku serca na podstawie dostarczonych parametrów pacjenta. Dzięki wykorzystaniu zaawansowanych algorytmów uczenia maszynowego, aplikacja analizuje dane medyczne i dostarcza dokładne prognozy, które mogą wspierać decyzje medyczne i wczesną interwencję.

Znaczenie Przewidywania Ataku Serca

Ataki serca są jedną z głównych przyczyn zgonów na świecie. Wczesne wykrycie osób z wysokim ryzykiem może znacząco poprawić wyniki leczenia i zmniejszyć śmiertelność. Przewidywanie ryzyka ataku serca może pomóc w:

- Wczesnej identyfikacji pacjentów wymagających intensywniejszej opieki.
- Personalizacji planów leczenia.
- Zwiększeniu świadomości pacjentów na temat ich stanu zdrowia i konieczności zmiany stylu życia.

Opis Aplikacji

Aplikacja **heart-attack-prediction** przyjmuje zestaw parametrów medycznych pacjenta i na ich podstawie prognozuje ryzyko wystąpienia ataku serca. Interfejs użytkownika pozwala na łatwe wprowadzanie danych, a zaawansowane algorytmy w backendzie przetwarzają te dane, generując wyniki w czasie rzeczywistym.

Struktura Projektu

Projekt został zorganizowany w sposób modułowy, z podziałem na różne komponenty odpowiedzialne za poszczególne części procesu przewidywania ryzyka ataku serca. Struktura ta obejmuje:

1. **Frontend**: Zrealizowany za pomocą Streamlit, umożliwia interakcję z użytkownikiem.

2. **Backend**: Oparty na FastAPI, łączy warstwę frontendową z backendową i obsługuje logikę aplikacji. Wykorzystuje Kedro do zarządzania przepływem danych i logiką przetwarzania. AutoGluon do trenowania ML oraz Weights & Biases do monitorowania procesu trenowania.
3. **Baza Danych**: do przechowywania danych wykorzystywane jest baza danych PostgreSQL.

Użyty zbiór danych

Dane są w formacie klucz: wartość.

- **Age**: Wiek pacjenta
- **Sex**: Płeć pacjenta
- **exang**: Dławica wywołana wysiłkiem (1 = tak; 0 = nie)
- **ca**: Liczba głównych naczyń (0-3)
- **cp**: Typ bólu w klatce piersiowej
    - Wartość 1: typowa dławica piersiowa
    - Wartość 2: nietypowa dławica piersiowa
    - Wartość 3: ból nie związany z sercem
    - Wartość 4: bezobjawowy
- **trtbps**: Ciśnienie krwi w spoczynku (w mm Hg)
- **chol**: Cholesterol w mg/dl pobierany za pomocą czujnika BMI
- **fbs**: Poziom cukru na czczo > 120 mg/dl (1 = prawda; 0 = fałsz)
- **rest_ecg**: Wyniki elektrokardiograficzne w spoczynku
    - Wartość 0: normalny
    - Wartość 1: mający nieprawidłowość fali ST-T (odwrócenie fali T i/lub uniesienie lub obniżenie ST o > 0,05 mV)
    - Wartość 2: wykazujący prawdopodobne lub pewne przerosty lewej komory serca według kryteriów Estes'a
- **thalach**: Maksymalne osiągnięte tętno
- **target**:
    - Wartość 0: mniejsze ryzyko ataku serca
    - Wartość 1: większe ryzyko ataku serca

Technologie Użyte do Tworzenia Projektu

Ogólna Architektura

Projekt został umieszczony w kontenerach Docker, z podziałem na frontend, backend oraz bazę danych.

Frontend

Frontend został wykonany przy użyciu frameworku Streamlit, co umożliwia interakcję użytkownika z aplikacją w przystępny i intuicyjny sposób.

Backend

Backend składa się z kilku kluczowych komponentów:

- **FASTAPI**: Służy do połączenia warstwy frontend z warstwą backend, obsługując żądania HTTP i dostarczając odpowiedzi do frontendu.
- **Kedro**: Odpowiada za zarządzanie pipeline'em danych i logiką aplikacji. Kedro zapewnia strukturyzowane podejście do tworzenia pipeline'ów przetwarzania danych, umożliwiając modularność i ponowne wykorzystanie kodu. Pipeliny odpowiadają za:
    - **Przetwarzania danych**: Odpowiada za walidację i przekształcanie danych wejściowych.
    - **Trenowania modeli**: Integruje AutoGluon do automatycznego trenowania modeli ML.
- **AutoGluon**: Wykorzystywany w pipeline Kedro do automatycznego trenowania modeli ML.
- **WanDB**: (Weights & Biases) Używane do śledzenia procesu trenowania modelu, monitorowania wyników i zarządzania eksperymentami.

Baza Danych

Dane są zapisywane w bazie danych PostgreSQL, która jest umieszczona w osobnym kontenerze.

Opis funkcjonalności aplikacji heart-attack-prediction

Aplikacja **heart-attack-prediction** wykorzystuje FastAPI do interakcji z użytkownikiem, umożliwiając zarządzanie danymi i uruchamianie predykcji dotyczących ryzyka ataku serca. Poniżej znajduje się opis dostępnych funkcji aplikacji wraz z odpowiadającymi im endpointami.

Endpointy i ich funkcjonalności

1. **Pobieranie danych ze zbioru**

```python
@router.get("/heart_data/{dataset_name}",
response_model=List[HeartDataSet])
async def get_heart_data(dataset_name: str):
    return get_dataset(dataset_name)
```

   - **Endpoint**: GET /heart_data/{dataset_name}
   - **Opis**: Pobiera dane ze wskazanego zbioru danych i zwraca je w formacie JSON.

2. **Przewidywanie ryzyka ataku serca**

```python
@router.post("/predict", response_model=int)
async def predict(data: HeartPrediction):
    return get_prediction(data)
```

   - **Endpoint**: POST /predict
   - **Opis**: Przyjmuje dane pacjenta w formacie JSON, przetwarza je i zwraca prognozę ryzyka ataku serca na podstawie wytrenowanego modelu AutoGluon.

3. **Usuwanie zbioru danych**

```
@router.delete("/delete_dataset/{dataset_name}")
async def delete_dataset_endpoint(dataset_name: str):
    try:
        delete_dataset(dataset_name)
        return {"message": f"Dataset {dataset_name} deleted
successfully"}
    except HTTPException as e:
        raise e
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

   - **Endpoint**: DELETE /delete_dataset/{dataset_name}
   - **Opis**: Usuwa wszystkie dane ze wskazanego zbioru danych, zapisując pustą tabelę.

4. **Dodawanie wiersza do zbioru danych**

```
@router.post("/add_row/{dataset_name}")
async def add_row_to_dataset_endpoint(dataset_name: str, row: dict):
    try:
        add_row_to_dataset(dataset_name, row)
        return {"message": f"Row added to dataset {dataset_name}
successfully"}
    except HTTPException as e:
        raise e
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

   - **Endpoint**: POST /add_row/{dataset_name}
   - **Opis**: Dodaje nowy wiersz do wskazanego zbioru danych. W przypadku błędu zwraca odpowiedni komunikat.

5. **Usuwanie wiersza ze zbioru danych**

```
@router.delete("/remove_row/{dataset_name}/{row_id}")
async def remove_row_from_dataset_endpoint(dataset_name: str,
row_id: int):
    try:
        remove_row_from_dataset(dataset_name, row_id)
        return {"message": f"Row {row_id} removed from dataset
{dataset_name} successfully"}
    except HTTPException as e:
        raise e
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

   - **Endpoint**: DELETE /remove_row/{dataset_name}/{row_id}
   - **Opis**: Usuwa wskazany wiersz ze zbioru danych na podstawie ID wiersza. W przypadku błędu zwraca odpowiedni komunikat.

6. **Uruchamianie pipeline'u**

```
@router.post("/run_pipelines")
async def run_pipelines_endpoint():
    try:
        run_pipeline()
        return {"message": "Pipelines executed and new model
generated successfully"}
    except HTTPException as e:
        raise e
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

   - **Endpoint**: POST /run_pipelines
   - **Opis**: Uruchamia pipeline przetwarzania danych i trenowania modelu. W przypadku błędu zwraca odpowiedni komunikat.

7. **Importowanie danych z pliku CSV**

```python
@router.post("/import_data/{dataset_name}")
async def import_data_endpoint(dataset_name: str, file: UploadFile =
File(...)):
    try:
        file_content = await file.read()

        if not file_content:
            raise HTTPException(status_code=400, detail="Uploaded
file is empty")

        file_location = f"heart-attack-
prediction/data/01_raw/{file.filename}"
        os.makedirs(os.path.dirname(file_location), exist_ok=True)

        with open(file_location, "wb") as file_object:
            file_object.write(file_content)

        if not os.path.exists(file_location) or
os.path.getsize(file_location) == 0:
            raise HTTPException(status_code=400, detail="Failed to
save uploaded CSV or file is empty")

        with open(file_location, "r") as file_check:
            saved_file_content = file_check.read()

        append_data_to_dataset(dataset_name, file_location)
        return {"message": f"Data appended to dataset {dataset_name}
successfully"}
    except HTTPException as e:
        raise e
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

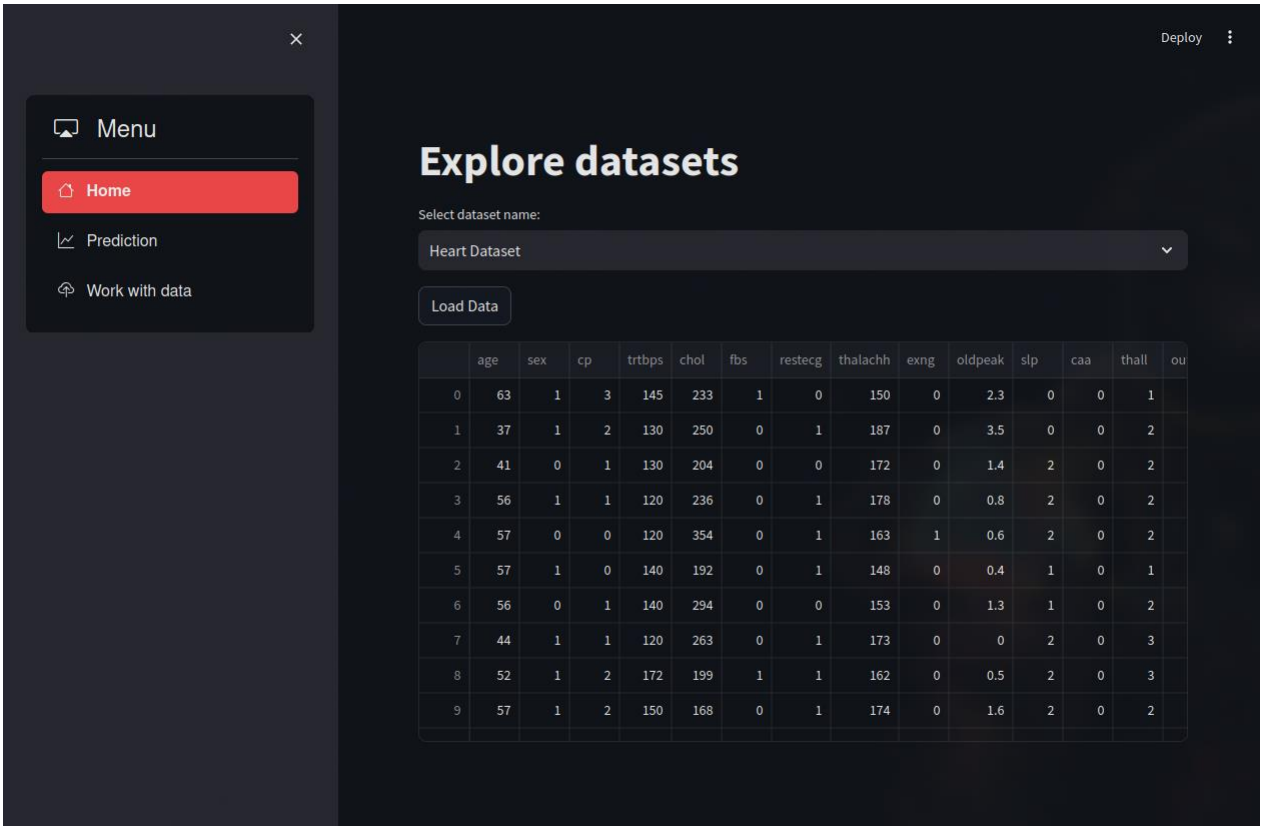
- **Endpoint**: POST /import_data/{dataset_name}
- **Opis**: Dodaje dane z pliku CSV do istniejącego zbioru danych. W przypadku błędów (np. pusty plik, błędy parsowania) zwraca odpowiedni komunikat.

Opis interakcji użytkownika z aplikacją za pomocą Streamlit

Aplikacja **heart-attack-prediction** oferuje interfejs użytkownika wykonany w Streamlit, który pozwala na łatwe zarządzanie danymi oraz korzystanie z funkcji predykcyjnych. Aplikacja składa się z trzech głównych sekcji: Home, Prediction oraz Work with Data. Poniżej przedstawiono szczegółowy opis interakcji użytkownika w każdej z tych sekcji.

Strony aplikacji w Streamlit

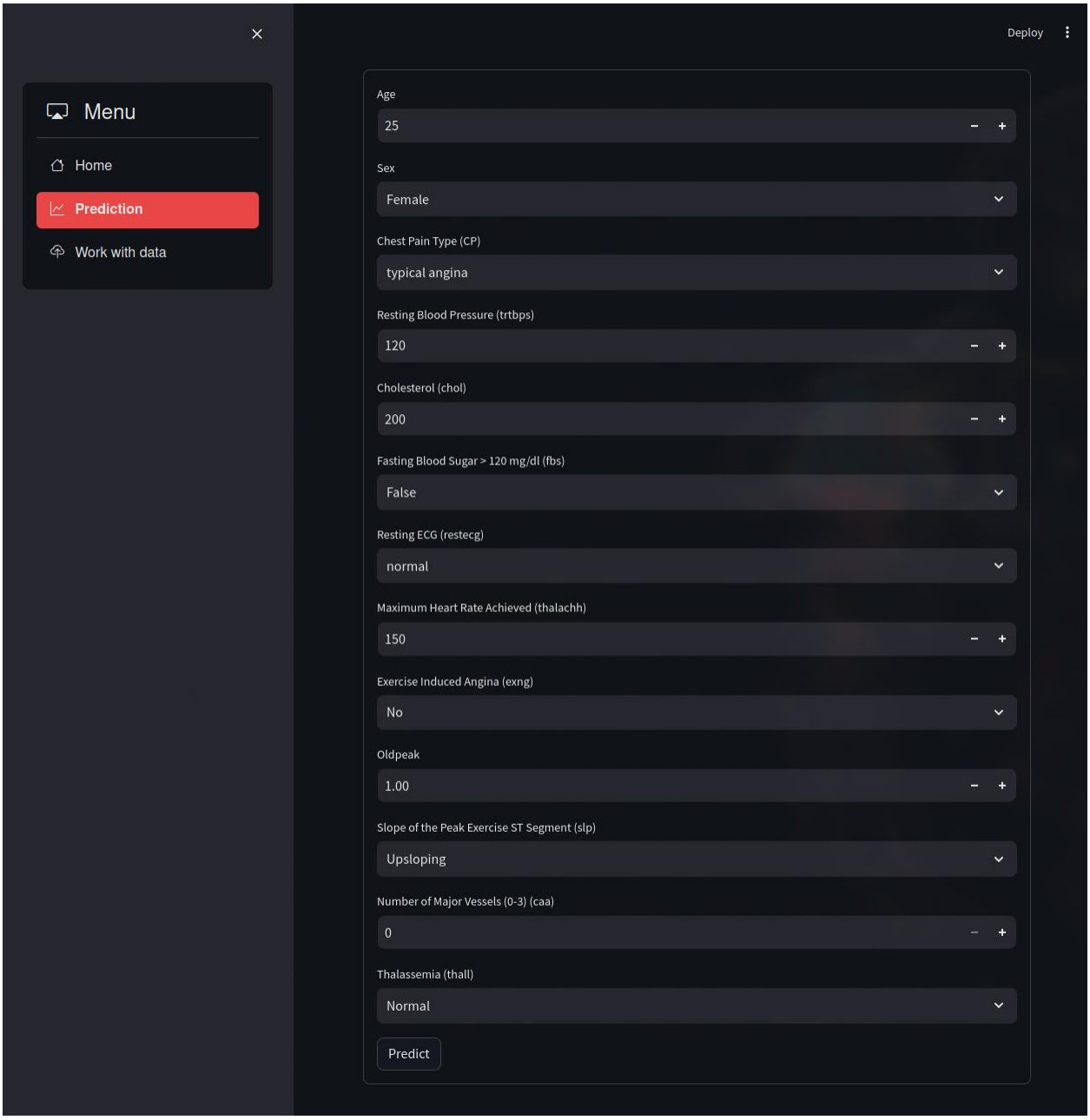
1. **Home**



	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	ou
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	
7	44	1	1	120	263	0	1	173	0	0	2	0	3	
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2	

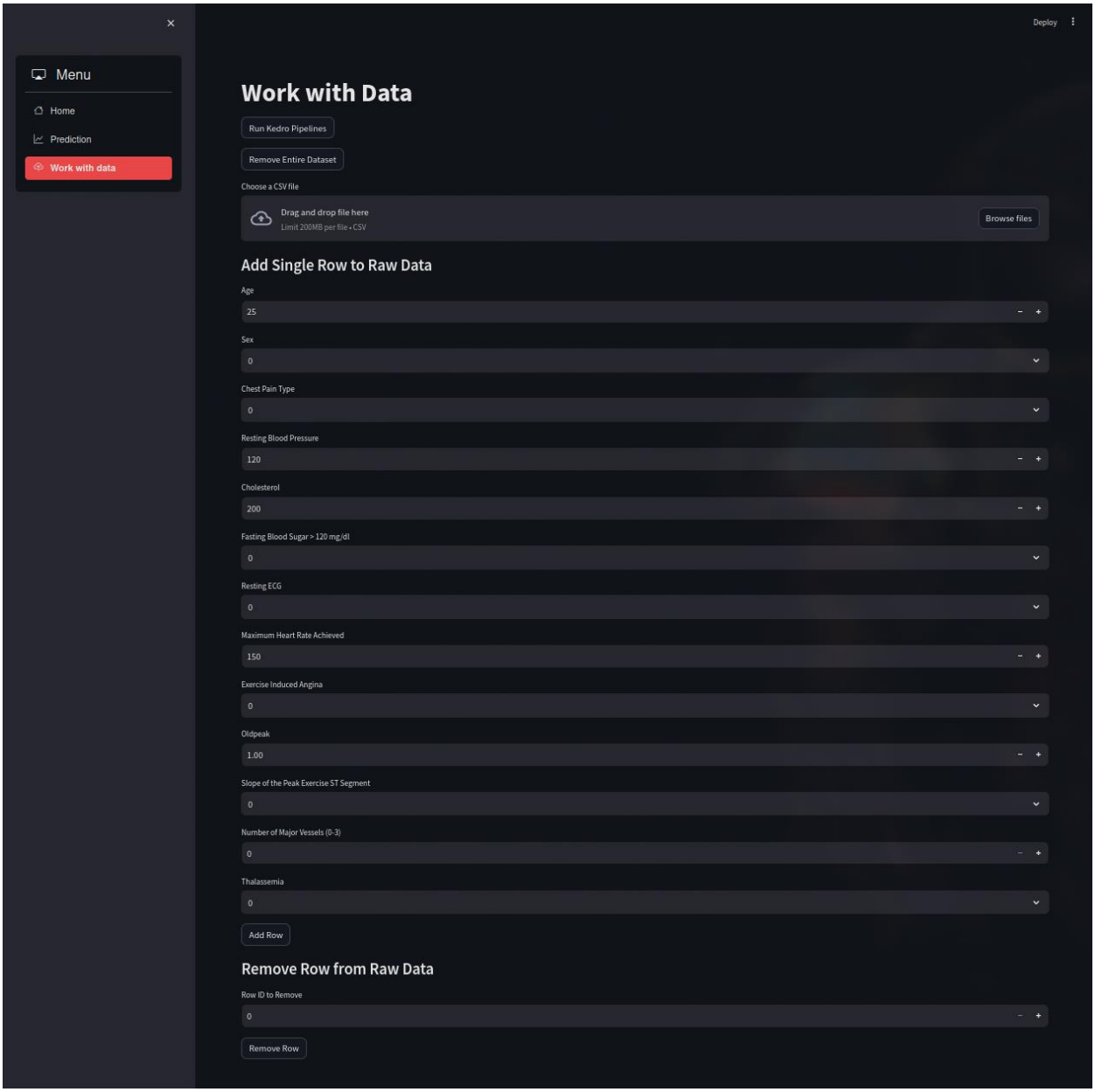
Sekcja Home umożliwia użytkownikowi przeglądanie dostępnych zestawów danych. Użytkownik wybiera nazwę zestawu danych z listy rozwijanej, a następnie ładuje dane, klikając przycisk "Load Data". Wyniki są wyświetlane w formie tabeli.

2. Prediction



Sekcja Prediction pozwala użytkownikowi wprowadzić parametry medyczne pacjenta i uzyskać prognozę ryzyka ataku serca. Użytkownik wypełnia formularz, a następnie klika "Predict", aby wysłać dane do backendu i otrzymać wynik.

3. **Work with Data**



Sekcja Work with Data umożliwia użytkownikowi zarządzanie danymi. Użytkownik może:

- Uruchomić pipeline'y Kedro, klikając przycisk "Run Kedro Pipelines".
- Usunąć cały zbiór danych, klikając przycisk "Remove Entire Dataset".
- Zaimportować nowy plik CSV, korzystając z przycisku "Import New CSV".
- Dodać pojedynczy wiersz do surowych danych za pomocą formularza.
- Usunąć wiersz z surowych danych na podstawie ID wiersza.

Diagram przedstawiający uproszczoną architekturę aplikacji.