Univerzitet „Džemal Bijedić“ u Mostaru

Fakultet Informacijskih Tehnologija

Predmet: Umjetna inteligencija

HealthPredict – AI agent za procjenu rizika od srčanih bolesti

Dokumentacija

Profesor:
prof.dr.sc. Nina Bijedić

Student:
Nejra Muminović, IB220043

Mostar, 2025

Svrha i cilj projekta

Cilj projekta HealthPredict je razvoj AI agenta koji omogućava procjenu rizika od srčanih bolesti na osnovu medicinskih i demografskih podataka pacijenata. Sistem je namijenjen kao pomoćni alat za brzu i informativnu procjenu zdravstvenog rizika, a ne kao zamjena za stručnu medicinsku dijagnozu.

Projekt je inspiriran činjenicom da su kardiovaskularne bolesti jedan od vodećih uzroka smrtnosti u svijetu, te da rana procjena rizika može imati ključnu ulogu u prevenciji i pravovremenom liječenju. Korištenjem historijskih medicinskih podataka i algoritama mašinskog učenja, HealthPredict omogućava korisnicima da dobiju uvid u vjerovatnoću postojanja srčanih oboljenja.

Svrha projekta je izrada web aplikacije koja koristi algoritme mašinskog učenja za analizu faktora kao što su starost, spol, tip angine, krvni pritisak, holesterol, EKG rezultati i drugi relevantni parametri, te na osnovu njih predviđa rizik od srčanih bolesti.

Tehnologije

Za realizaciju AI agenta HealthPredict korištene su sljedeće tehnologije:

- Python – korišten za backend logiku, obradu podataka, treniranje modela i predikciju.
- Flask – korišten za izradu REST API-ja i komunikaciju između frontenda i backend-a.
- Random Forest Classifier – algoritam mašinskog učenja korišten za klasifikaciju rizika od srčanih bolesti.
- Logistic Regression – korišten kao dodatni model tokom treniranja i evaluacije.
- JavaScript – korišten za izradu frontend dijela aplikacije i komunikaciju s API-jem.
- HTML & CSS – korišteni za izradu korisničkog interfejsa.
- Pandas – korišten za obradu i manipulaciju medicinskih podataka.
- Scikit-learn – biblioteka korištena za treniranje, skaliranje i evaluaciju ML modela.
- CSV dataset (Heart Disease Dataset) – korišten kao izvor podataka za treniranje modela.

Zašto Random Forest Classifier?

Random Forest Classifier je odabran zbog svoje sposobnosti da:

- efikasno radi s većim brojem karakteristika,
- bude otporan na šum u podacima,
- pruži dobru tačnost bez potrebe za kompleksnim podešavanjem,
- omogući uvid u važnost pojedinih karakteristika koje utiču na predikciju.

Zahvaljujući ovim osobinama, Random Forest je pogodan za medicinske podatke koji često sadrže varijacije i nelinearne odnose.

Funkcionalnosti

AI agent HealthPredict nudi sljedeće funkcionalnosti:

- Predikcija rizika od srčanih bolesti
  Korisnik unosi medicinske podatke (starost, spol, tip bola u prsima, krvni pritisak, holesterol, EKG rezultat, puls, itd.), a sistem vraća procjenu rizika (nizak ili visok rizik) zajedno sa vjerovatnoćom.

- Prikaz najvažnijih faktora
  Sistem prikazuje koje karakteristike su najviše utjecale na odluku modela.

- Dodavanje novih pacijenata
  Korisnici mogu dodavati nove podatke o pacijentima u bazu podataka.

- Ponovno treniranje modela (Retraining)
  Nakon dodavanja novih podataka, model se može ponovno trenirati kako bi se poboljšala tačnost i prilagodio novim informacijama.

- Višejezična podrška (Bosanski / Engleski)
  Korisnički interfejs omogućava promjenu jezika aplikacije.

Kod

Učitavanje podataka

Podaci se učitavaju iz CSV datoteke korištenjem Pandas biblioteke. Dataset sadrži medicinske i demografske podatke pacijenata relevantne za procjenu srčanih bolesti.

```
6   def load_data(path="data/heart.csv"):
7       df = pd.read_csv(path)
8       return df
```

Procesiranje podataka

Podaci se čiste i pripremaju za treniranje modela:

- uklanjaju se nedostajuće vrijednosti,
- kreira se ciljna varijabla (0 – nema bolesti, 1 – postoji rizik),
- vrši se enkodiranje kategorijskih varijabli,
- numeričke vrijednosti se skaliraju korištenjem StandardScaler.

```python
def preprocess(df):
    df = df.dropna()

    df["target"] = df["num"].apply(lambda x: 1 if x > 0 else 0)
    y = df["target"]

    X = df.drop(["num", "target", "id", "dataset"], axis=1)

    X["sex"] = X["sex"].map({"Male":1, "Female":0})
    X["fbs"] = X["fbs"].map({True:1, False:0})
    X["exang"] = X["exang"].map({True:1, False:0})

    X = pd.get_dummies(X, columns=["cp","restecg","slope","thal"], drop_first=True)

    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    return X_scaled, y, scaler, X.columns.tolist()
```

Podaci se prije treniranja modela prolaze kroz fazu procesiranja. Uklanjaju se redovi sa nedostajućim vrijednostima, zatim se kreira ciljna varijabla gdje vrijednost 1 označava prisustvo rizika od srčanih bolesti, dok 0 označava odsustvo rizika. Kategorijske varijable se enkodiraju korištenjem binarnog mapiranja i one-hot enkodiranja, dok se numeričke vrijednosti skaliraju pomoću StandardScaler metode radi poboljšanja performansi modela.

Treniranje modela

Model se trenira korištenjem Random Forest Classifier algoritma. Tokom treniranja čuvaju se:

- istrenirani model,
- scaler,
- lista feature-a.

Ovi podaci se koriste kasnije tokom predikcije kako bi ulazni podaci bili pravilno obrađeni.

```python
print("Treniranje RandomForest sa GridSearch...")
rf = RandomForestClassifier(random_state=42)
param_grid = {
    "n_estimators": [100, 200],
    "max_depth": [None, 6, 10],
    "min_samples_split": [2, 5]
}
grid = GridSearchCV(rf, param_grid, cv=5, scoring="roc_auc", n_jobs=-1)
grid.fit(X_train, y_train)
best_rf = grid.best_estimator_
print("Random Forest metrics:", evaluate_model(best_rf, X_test, y_test))
print("Best RF params:", grid.best_params_)

os.makedirs("models", exist_ok=True)
joblib.dump({"model": best_rf, "scaler": scaler, "features": feature_names}, MODEL_PATH)
print(f"Model spremljen u {MODEL_PATH}")
```

Ovaj dio koda služi za treniranje Random Forest Classifier modela i optimizaciju njegovih hiperparametara pomoću GridSearchCV metode. Najprije se inicijalizira Random Forest model uz korištenje parametra random_state=42, čime se osigurava ponovljivost rezultata. Zatim se definiše skup hiperparametara koje će GridSearch ispitivati, uključujući broj stabala u modelu (n_estimators), maksimalnu dubinu stabla (max_depth) i minimalan broj uzoraka potreban za dijeljenje čvora (min_samples_split). GridSearchCV se koristi za sistematsko testiranje svih kombinacija ovih parametara kroz 5-struku unakrsnu validaciju, pri čemu se kao kriterij za odabir najboljeg modela koristi ROC AUC metrika. Nakon treniranja, bira se najbolji model sa optimalnim parametrima, vrši se evaluacija njegovih performansi na testnom skupu podataka, a zatim se istrenirani model zajedno sa scalerom i listom feature-a trajno sprema na disk kako bi se kasnije mogao koristiti za predikciju rizika od srčanih bolesti.

Predikcija rizika srčanih bolesti

Na osnovu unesenih podataka, backend API obrađuje zahtjev i vraća:

- predikciju (nizak ili visok rizik),
- vjerovatnoću rizika,
- najutjecajnije karakteristike.

```python
def predict(model, scaler, features, data: dict):
    X = pd.DataFrame([data])

    # poravnaj feature-e
    X = X.reindex(columns=features, fill_value=0)

    X_scaled = scaler.transform(X)
    proba = model.predict_proba(X_scaled)[0][1]
    pred = int(proba >= 0.5)

    return {
        "prediction": pred,
        "probability": float(proba)
    }
```

Funkcija predict prima istrenirani model, scaler, listu feature-a i podatke unesene od strane korisnika. Ulazni podaci se prvo pretvaraju u Pandas DataFrame, a zatim se poravnavaju prema feature-ima korištenim tokom treniranja modela, pri čemu se nedostajuće vrijednosti popunjavaju nulom. Nakon toga, podaci se skaliraju pomoću prethodno sačuvanog StandardScaler objekta. Random Forest Classifier zatim izračunava vjerovatnoću rizika od srčane bolesti, na osnovu koje se formira konačna predikcija (0 – nizak rizik, 1 – visok rizik). Funkcija vraća rezultat u JSON formatu koji sadrži predikciju i vjerovatnoću rizika.

Dodavanje novih podataka

Novi podaci o pacijentima se validiraju, pohranjuju i koriste za buduće treniranje modela.

```python
@app.route("/add", methods=["POST"])
def add_data():
    try:
        df = load_data()
        data = request.json

        numeric_fields = ["age","sex","trestbps","chol","fbs","thalch","exang","oldpeak","ca"]
        for field in numeric_fields:
            if field not in data or data[field] in ["", None]:
                return jsonify({"status": "error",
                                "message_bs": "Molimo popunite sva obavezna polja.",
                                "message_en": "Please fill all required fields."}), 400
            data[field] = float(data[field])

        string_fields = ["cp","restecg","slope","thal"]
        for field in string_fields:
            if field not in data or data[field] in ["", None]:
                return jsonify({"status": "error",
                                "message_bs": "Molimo popunite sva obavezna polja.",
                                "message_en": "Please fill all required fields."}), 400

        data.setdefault("id", len(df) + 1)
        data.setdefault("dataset", "new")
        data.setdefault("num", 0)

        df = pd.concat([df, pd.DataFrame([data])], ignore_index=True)
        df.to_csv("data/heart.csv", index=False)

        return jsonify({"status": "success",
                        "message_bs": "Pacijent uspješno dodan.",
                        "message_en": "Patient added successfully."})

    except Exception as e:
        print("Add Data Error:", e)
        return jsonify({"status": "error",
                        "message_bs": f"Greška: {str(e)}",
                        "message_en": f"Error: {str(e)}"}), 500
```

Ova funkcija */add* omogućava dodavanje novih podataka o pacijentima u backend sistem. Prvo učitava postojeći skup podataka, a zatim prima JSON podatke sa zahtjeva. Funkcija provjerava da li su sva obavezna numerička i kategorijska polja popunjena i vrši konverziju numeričkih vrijednosti u odgovarajući tip. Nakon validacije, dodaju se interne kolone poput id, dataset i num kako bi novi zapis bio kompatibilan sa postojećim podacima. Na kraju, novi pacijent se dodaje u DataFrame i trajno pohranjuje u CSV datoteku, čime podaci postaju dostupni za buduće treniranje modela. U slučaju greške, funkcija vraća odgovarajući JSON odgovor sa opisom problema.

Ponovno treniranje modela

Korisnik može pokrenuti ponovno treniranje modela koristeći prošireni skup podataka, čime se osigurava kontinuirano poboljšanje tačnosti predikcija.

```python
@app.route("/retrain", methods=["POST"])
def retrain():
    try:

        df = load_data()
        df = df.dropna()
        if df.empty:
            return jsonify({
        "status": "error",
        "message_bs": "Nije moguće retrenirati model jer nema validnih korisnika.",
        "message_en": "Cannot retrain the model because there are no valid users."
            })


        os.system("python train_model.py")
        global model, scaler, features
        model, scaler, features = load_model()

        return jsonify({
            "status": "success",
            "message_bs": "Model je uspješno retreniran i ponovno učitan.",
            "message_en": "Model retrained and reloaded successfully."
        })

    except Exception as e:
        print("Retrain Error:", e)
        return jsonify({
            "status": "error",
            "message_bs": f"Greška prilikom retreniranja: {str(e)}",
            "message_en": f"Error retraining model: {str(e)}"
        }), 500
```

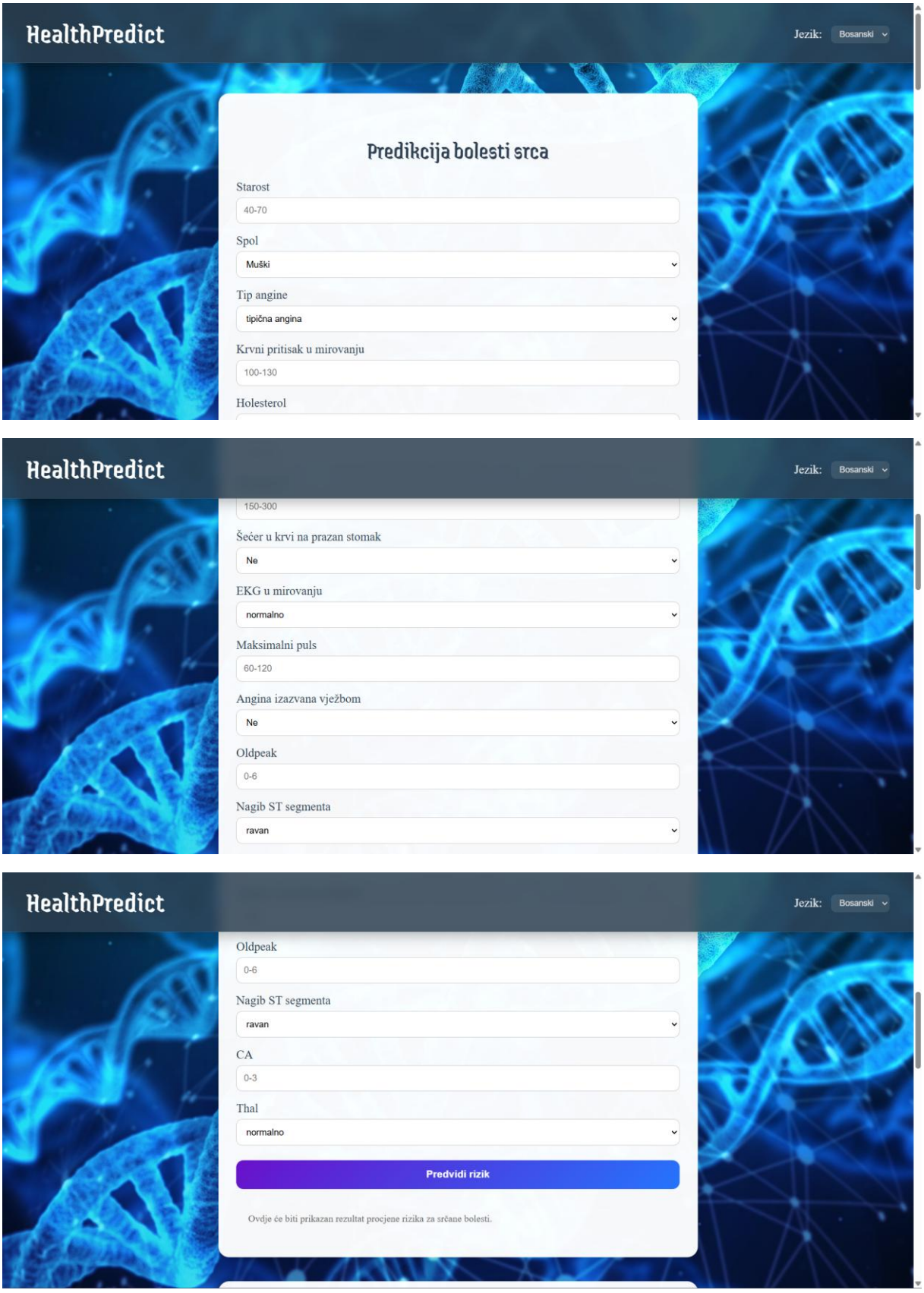
Ova funkcija */retrain* u Flask API-ju omogućava ponovno treniranje modela na ažuriranom skupu podataka. Kada korisnik pošalje POST zahtjev na ovaj endpoint:

- Funkcija učitava postojeći dataset pozivom `load_data()` i uklanja sve redove sa nedostajućim vrijednostima (`NaN`).
- Provjerava se da li dataset sadrži validne podatke; ako ne, vraća se greška u JSON formatu.
- Ako su podaci validni, pokreće se skripta `train_model.py` koja trenira model koristeći Random Forest algoritam.
- Nakon što se treniranje završi, funkcija ponovno učitava model, scaler i listu feature-a u memoriju pomoću `load_model()`.
- Na kraju, funkcija vraća JSON odgovor s informacijom da je model uspješno retreniran i ponovno učitan.

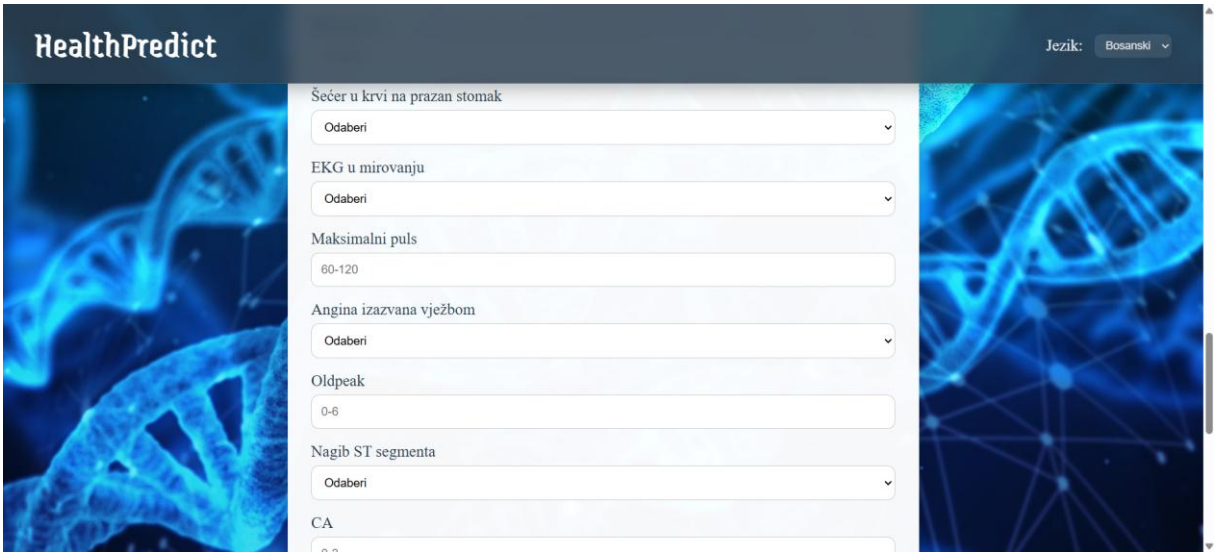
U slučaju greške tokom procesa, funkcija hvata iznimku i vraća JSON odgovor sa detaljima o problemu.

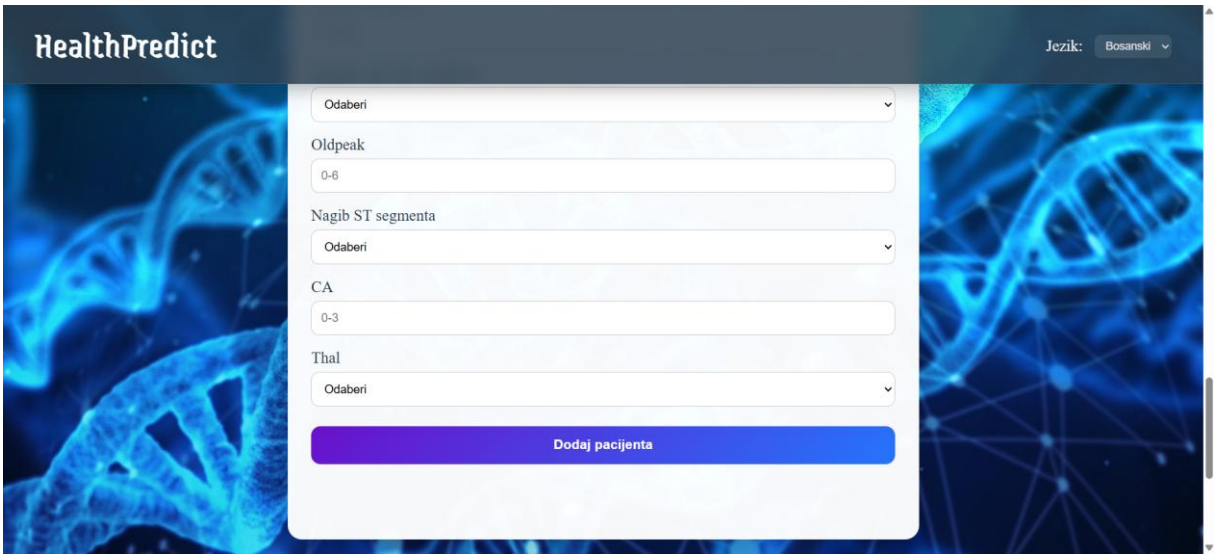
Korisnički interfejs

Interfejs za predikciju rizika od srčanih bolesti:

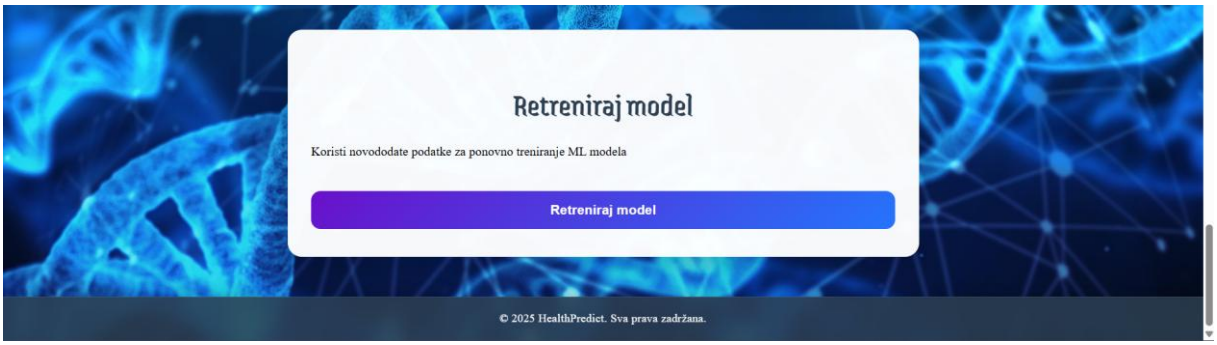


Interfejs za dodavanje novih podataka:





Interfejs za ponovno treniranje modela:



Zaključak

Aplikacija HealthPredict predstavlja moćan alat za procjenu rizika od srčanih bolesti koristeći podatke o pacijentima i mašinsko učenje. Korištenjem Random Forest algoritma, aplikacija omogućava preciznu predikciju rizika, pruža uvid u najutjecajnije karakteristike svakog pacijenta i olakšava dodavanje novih podataka radi kontinuiranog poboljšanja modela.

Integracija jednostavnog korisničkog sučelja, mogućnosti višejezične podrške i funkcionalnosti ponovnog treniranja modela osigurava da korisnici – bilo da su zdravstveni stručnjaci ili sami pacijenti – mogu donositi informirane odluke temeljem pouzdanih predikcija.

Ova aplikacija pokazuje značajan potencijal u primjeni AI tehnologija u zdravstvu, čineći procjenu rizika dostupnijom, bržom i preciznijom.