FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Raspoznavanje uzoraka i strojno učenje

SEMINARSKI RAD

"Predviđanje kardiovaskularnih bolesti"

Daria Ćaćić

Osijek, 2021.

SADRŽAJ

1. UVOD

Kardiovaskularne bolesti (KVB) ili bolesti srca i krvnih žila su jedan su od vodećih uzroka smrti i invaliditeta diljem svijeta. U prosjeku umre 17.9 milijuna ljudi godišnje od ove grupe bolesti od kojih je 4 od svakih 5 osoba umrlo od srčanog ili moždanog udara. Trećina tih osoba je umrlo preuranjenom smrću u dobi ispod 70 godina. Zbog ove statistike njihovo je sprječavanje jedno od važnijih zadaća moderne medicine. Predviđanje kardiovaskularnih bolesti jedna je od metoda koje su korisne u prevenciji ili ranom otkrivanju što može omogućiti produljenje životnog vijeka. Ono se može izvesti pomoću rezultata medicinskih testova koji su dostupni većinom kada je već razvijena bolest i u naprednom stadiju. Osim tog načina predviđanja postoje pokazatelji ili rizične skupine koje može gotovo svatko sam kod kuće provjeriti. Ti faktori će biti ispitani u ovom programu. Naravno ne daju točnost predviđanja kao neki medicinski testovi, ali dovoljno da daju smjernice kako poboljšati život i obratiti se liječniku kako bi se bolest otkrila ranije. S obzirom da je projekt vezan za strojno učenje, osim obične predikcije vršit će se predikcija sa proizvoljno odabranim algoritmima strojnog učenja.

2. TIJEK RJEŠAVANJA ZADATKA

Rješavanje ovog zadatka svodi se na nekoliko koraka. Svaki će biti zasebno detaljnije komentiran sa prikladnim primjerima. Počet će se od pregleda podataka uzetih za treniranje modela pa kreiranje modela i na kraju kreiranje funkcionalnosti web stranice.

2.1. Opis korištenih podataka

Prvo korak pri rješavanju ovog projekta je naći prikladan skup podataka za treniranje. Web stranica za pristup podatcima korištenim za ovaj projekt nalazi se u dijelu poveznice i literatura. CSV datoteka sadrži 13 stupaca od kojih se prvi odnosi na ID, a zadnji na postojanje kardiovaskularne bolesti u pacijentu. Ostalih 11 odrano je za treniranje modela (ili manje, s obzirom da se proizvoljno biraju trening podatci od strane korisnika). Ti stupcu su: dob, spol, visina, težina, sistolički, dijastolički tlak, kolesterol, glukoza, pušenje, konzumacija alkohola, fizička aktivnost. Skoro svi od ovih podataka su poznati i lako mjerljivi prosječnoj osobi. Kolesterol i glukoza se obično pokažu na krvnoj slici i osobe su često svjesne problema zbog očitih smetnji. Baza sadrži 70_000 vrijednosti. Primjer podatka prikazan je u tablici ispod.

106	21865	1	160	68.0	140	90	1	1	0	0	1	1

Podatci su poredani kao redoslijedom kojim su prethodno navedeni. Dob je u danima, za spol vrijednosti 1 označavaju žensku osobu, a 2 mušku. Visina je zadana u cm, a težina u kg. Za tlak su korištene standardne jedinice mm Hg korištene na svim tlakomjerima. Kolesterol se odnosi na razine kolesterola gdje 1 predstavlja normalnu, 2 više od normalnog i 3 puno više od normalnog. Za glukozu se uzima isto mjerenje. Za preostale vrijednosti (pušenje, alkohol, fizička aktivnost) vrijedi da je vrijednost 0 ukoliko se osoba ne bavi tim radnjama, a 1 ukoliko je pušač ili konzumira alkohol ili se bavi fizičkom aktivnosti. Zadnji stupac daje vrijednost 1 ukoliko postoji kardiovaskularna bolest i 0 ukoliko ne postoji. Iz ovih podataka se vidi da je kod osobe prisutna bolest.

2.2. Model strojnog učenja

Problem predviđanja kvb klasičan je problem binarne klasifikacije. Za algoritame učenja izabrani su 4 algoritma koji su dali najbolje rezultate. To su logistička regresija, k najbližih susjeda, random forest i gradient boosting. Kako bi stranica bila interaktivnija izabrano je više algoritama i dozvoljava se odabir algoritma za predikciju rezultata. Algoritmi su napisani unutar funkcija koje se pozivaju ukoliko je algoritam odabran. Treniranje modela unutar funkcija može se vidjeti na slici ispod.

```
def RFC_fun(X_train, X_test, Y_train, Y_test, user_input):
      RFC = RandomForestClassifier(n_estimators = 120, max_depth = 10,
min_samples_split = 10)
      RFC.fit(X_train, Y_train)
      st.write(str(accuracy_score(Y_test, RFC.predict(X_test)) * 100) + '%')
      prediction = RFC.predict(user_input)
      return prediction

def LR_fun(X_train, X_test, Y_train, Y_test, user_input):
      LR = LogisticRegression()
      LR.fit(X_train, Y_train)
      st.write(str(accuracy_score(Y_test, LR.predict(X_test)) * 100) + '%')
      prediction = LR.predict(user_input)
      return prediction

def kN_fun(X_train, X_test, Y_train, Y_test, user_input):
    kN = KNeighborsClassifier(n_neighbors = 40)
    kN.fit(X_train, Y_train)
    st.write(str(accuracy_score(Y_test, kN.predict(X_test)) * 100) + '%')
    prediction = kN.predict(user_input)
    return prediction

def GBC_fun(X_train, X_test, Y_train, Y_test, user_input):
    Grad_Boosting = GradientBoostingClassifier( n_estimators = 100,
max_depth = 1, learning_rate = 1,
                                            random_state = 0)
    Grad_Boosting.fit(X_train, Y_train)
    st.write(str(accuracy_score(Y_test, Grad_Boosting.predict(X_test)) * 100)
+ '%')
    prediction = Grad_Boosting.predict(user_input)
    return prediction
```

2.2.1 Definiranje funkcija za treniranje modela

2.3. Korišteni framework i prikaz web stranice

Kao framework uzet je *Streamlit*. Vrlo je jednostavan za korištenje i zadovoljava sve predviđene potrebe, otvorenog je koda i predviđen za rad sa strojnim učenjem i podatkovnim znanostima zbog čega je savršen za ovaj projekt. Korištenjem Streamlit-a uklanja se potreba kreiranja html-a što olakšava sveukupan posao. Kako bi se stranici lako pristupilo postavljena je na Heroku pomoću nekoliko jednostavnih koraka nakon završetka koda. Stranica za pristup aplikaciji dana je u dijelu *Poveznice i literatura*. Za korištenje Streamlit-a potrebno je prvo uvesti biblioteku *streamlit*. Naslov se zadaje funkcijom title.

```
st.title("Cardiovascular disease prediction")
```

2.3.1 Postavljanje naslova stranice

Za prikaz teksta postoji više funkcija. Neke od korištenih ovdje su write, header, subheader, markdown.

```
st.markdown("**To choose attributes, click on the arrow in the upper left corner
of the screen**")
st.subheader("Data information: ")

st.write("Systolic blood pressure is higher number.")
st.write("Diastolic blood pressure is lower number.")
st.write("Cholesterol levels are: 1 - normal, 2 - above normal, 3 - well above
normal")
st.write("Glucose levels are: 1 - normal, 2 - above normal, 3 - well above
normal")
```

2.3.2 Korištenje Streamlit funkcija za ispis

Write funkcija je za pisanje običnog teksta, subheader za postavljanj svojevrsnog poglavlja, a markdown omogućuje formatiranje teksta unutar markdown sintakse. Unošenje korisničkih podataka vrši se u sidebar-u prvo označavanjem checkbox-a za atribute kojima se želi trenirati model. Nakon označavanja atributa prikazuje se slider u kojemu se odabire vrijednost koju se želi unijeti. Za sve ovo u python-u je napravljena funkcija koja ukoliko je označen checkbox izvršava niz radnji kako bi se spremio uneseni podatak ili ukoliko nije označen spremilo vrijednost *false* u prikladnom polju koji se kasnije predaje pri podjeli trening podataka. Ukoliko atribut postoji to polje ima vrijednost *true*.

```
def get_user_input():

    numbers = [False]
    features = 0
    data = False

    col_1 = st.sidebar.checkbox('Select age')
    if col_1:
        age = st.sidebar.slider('20 - 90', 20, 90, 55, key = 1)
        age = age * 365
        data = {'age' : age}
        features = pd.DataFrame(data, index = [0])
        numbers = [True]
```

2.3.3 Definiranje korisnične funkcije i uvjeta za prvi atribut

Unutar funkcije slider definiraju se maksimalna, minimalna i početna vrijednost. Slično se odvija za sve ostale atribute, samo što ima provjeravanje dodatnog uvjeta ukoliko nije odabran nijedan atribut prije toga.

```
    col_2 = st.sidebar.checkbox('Select sex')
    if col_2:
        sex = st.sidebar.slider('1 - female, 2 - male', 1, 2, 1, key = 2)
        if data:
            features['sex'] = sex
            numbers.append(True)
        else:
            data = {'sex' : sex}
            features = pd.DataFrame(data, index = [0])
            numbers.append(True)
    else:
        numbers.append(False)
```

2.3.4 Uvjet za sve atribute nakon drugog, uključujući drugi

Svi korisnički podatci spremaju se u DataFrame pošto se modeli i treniraju na tom obliku. Na kraju funkcija predaje DataFrame sa korisničkim podatcima, niz kojim se određuju atributi korišteni za treniranje i informacija postoji li i jedan odabran atribut ili ne.

```
user_input, numbers, data = get_user_input()
```

2.3.5 Korištenje prethodno definirane funkcije za dohvaćanje unesenih podataka

U glavnom dijelu stranice nudi se pomoću selectbox izbor algoritma za predikciju i button za aktivaciju.

```
option = st.selectbox("Choose clasification algorithm", ("Logistic
Regression", "Random Forest", "k-Nearest Neighbors", "Gradient
Boosting"))
button = st.button("Get prediction")
```

2.3.6 Opcije selectbox-a i dohvaćanje stanja button-a

Izvođenje predikcije vrši se ukoliko je button pritisnut.

```
if button:
    if data == False:
        st.markdown("**Please choose at least 1 attribute.**")
    else:
        heart_df = pd.read_csv ("cardio_train.csv", index_col = 0 )

        X = heart_df.iloc[:, numbers]
        Y = heart_df.iloc[:, -1]

        X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size
= 0.25, random_state = 0)
```

2.3.7 Učitavanje i podjela podataka pri pritisku button-a

Učitavanje se radi tek prilikom pritiska button-a pošto se radi o jako velikoj csv datoteci i svaki put kada dođe do promjene poput označavanja atributa stranica se ponovo učitava što usporava cjelokupnu stranicu. Nakon učitavanja podataka u DataFrame vrši se podjela na x i y te podjela na test i train. Vrši se provjeravanje odabranog algoritma i na kraju ispisivanje rezultata.

```
if (prediction == 0):
            st.markdown("**Predicted that there is no cardiovascular disease.**")
        else:
            st.markdown("**Predicted that cardiovascular disease is present.**")

        st.write(prediction)
```

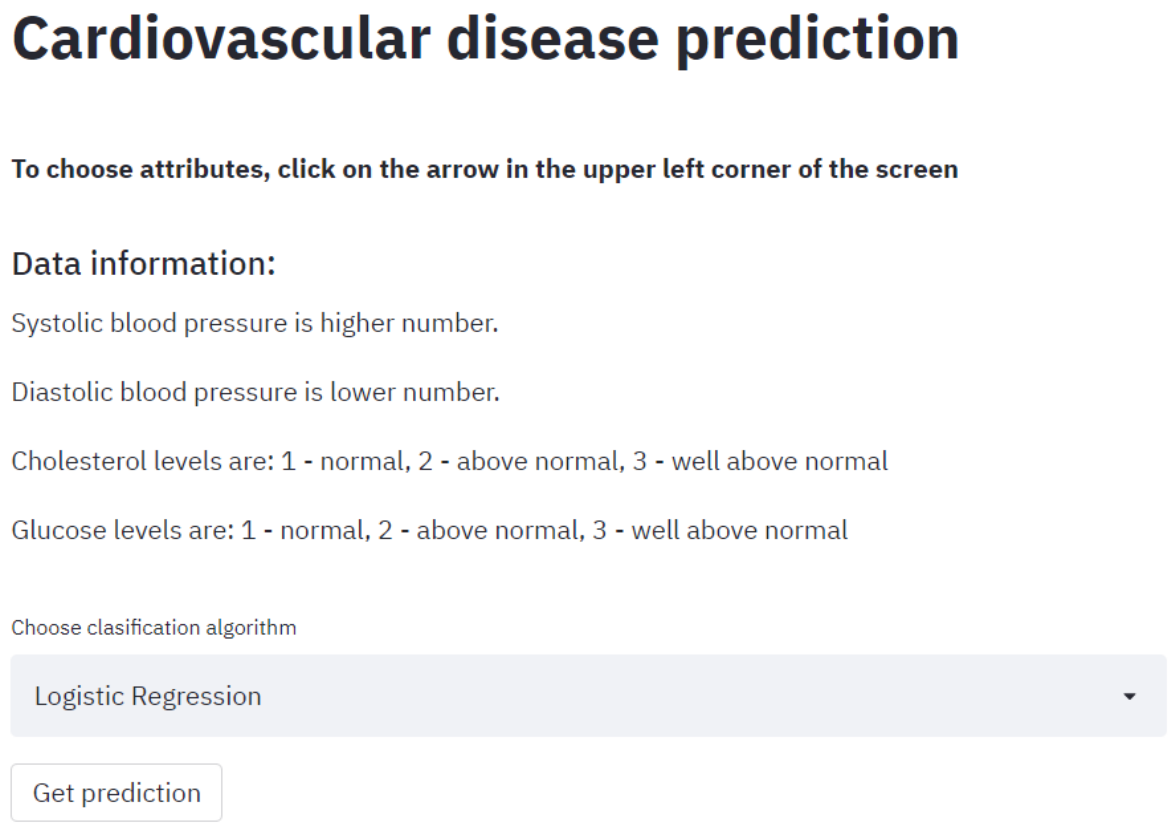
2.3.8 Ispisivanje rezultata predikcije

Nakon dovršetka koda stranici je moguće pristupiti pomoću cmd ili anaconda prompt za korisnike Anaconda-e. Kako bi bilo još lakše pristupiti potrebno je postaviti stranicu na Heroku. Prije postavljanja na Heroku potrebno je postaviti kod i podatke na GitHub pošto se sve vrši tim putem. Uz to i nekoliko datoteka koje sadržavaju podatke o bibliotekama, shell naredbe i naredbe za pokretanje. Nakon kreiranja svih potrebnih datoteka kreira se aplikacija na Heroku i povezuje sa GitHub-om nakon

čega ju je samo potrebno deploy-ati. Stranica sa detaljnijim uputama se nalazi u dijelu Poveznice i literatura.

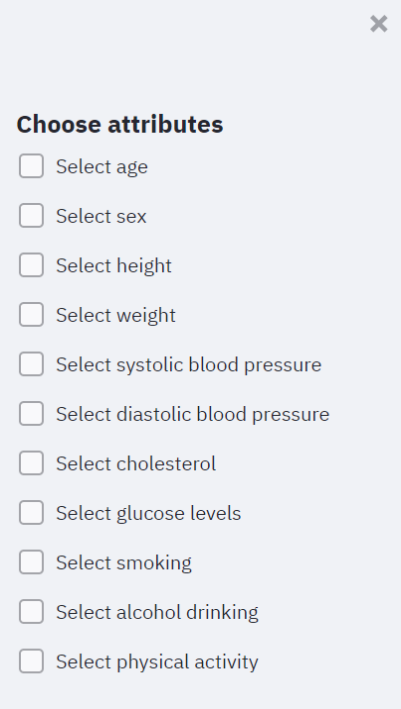
3. RJEŠENJE

Početna stranica nakon učitavanja stranice izgleda otprilike ovako.



3.1 Prvotni prikaz na stranici nakon učitavanja

Nude se dodatne informacije kako bi se bolje razumjeli atributi. Atributima se pristupa klikom na strelicu u gornjem lijevom kutu. Nakon čega se po strani prikazuje sljedeće.



3.2 Mogućnost odabira atributa pomoću checkbox-a

Klikom na atribut prikazuje se slidebar za zadavanje vrijednosti atributa.

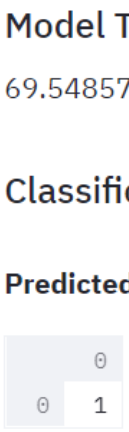
3.3 Unos podatka pomoću slider-a

Nakon odabira željenih atributa (u ovom slučaju će se svi odabrati) prikazuju se unesenu podatci dok se čeka rezultat predikcije.

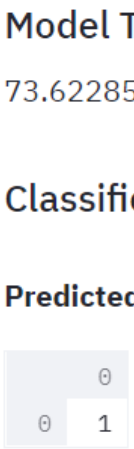
	age	sex	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active
0	20075	1	155	115	130	90	2	2	0	0	0

3.4 Prikaz korisnički unesenih podataka

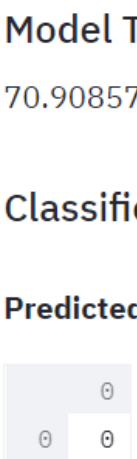
Nakon nekoliko sekundi dobiva se rezultat. Bit će prikazan za sve algoritme kako bi se usporedila točnost.



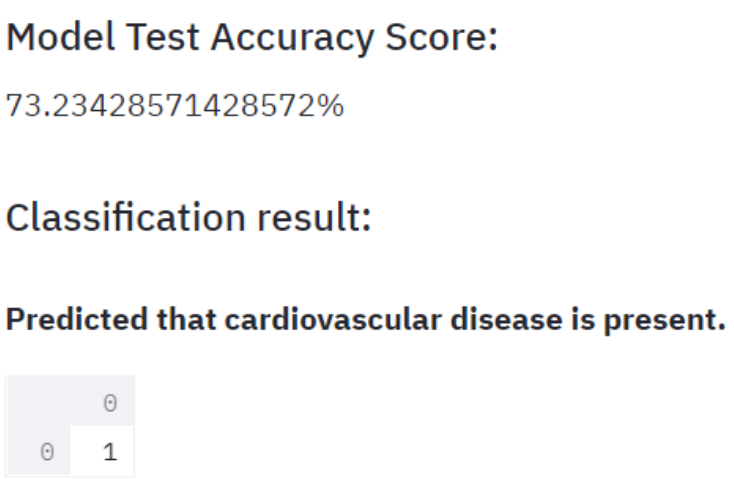
3.5 Rezultat za logističku regresiju



3.6 Rezultat za random forest algoritam



3.7 Rezultat za kNN algoritam



3.8 Rezultat za Gradient Boosting algoritam

Iako rezultati nisu predvidjeli isto može se zaključiti da je najvjerojatnije prisutna bolest. Najbolje rezultate daju random forest (73,6%) i gb (73,2%) od kojih je random forest dao malo bolje rezultate, ali je malo sporiji od gb. Najlošije su dali kNN (70,9%) i logistička regresija (69,5%) od kojih je kNN dao malo bolje rezultate, ali pri odabiru velikog broja susjeda (40). U suprotnom za mal broj poput 5 daje lošije rezultate. Oba ova algoritma iako su lošiji brži su od preciznijih algoritama. Rezultat od 73% točnosti nije baš dobar, ali govori dosta o tome koliko se može dobro predvidjeti bolest temeljeno na uzetim podatcima. Pošto ih je puno (70_000) ne radi se o manjku podataka za treniranje nego se može zaključiti da se ne može vrlo precizno predvidjeti bolest pomoću samo ovih podataka, ali mogu dati smjernice za kvalitetniji život jer je očito da imaju utjecaj na razvoj bolesti. Kako bi se poboljšao rezultat za nekoliko postotaka možda je moguće bolje narediti algoritme, ali neće se dobiti zadovoljavajući rezultat od 95%, najviše bi se dobilo nekoliko postotaka tim pute. Za bolje rezultate potrebno je odabrati druge podatke ili ih dodati postojećim. Naravno izborom manje atributa smanjuje se i točnost predikcije. Ukoliko nije izabran ni jedan atribut prikazuje se prikladna poruka koja o tome obavještava.

3.9 Prikaz ukoliko nije odabran ni jedan atribut

4. ZAKLJUČAK

Zbog učestalosti kardiovaskularnih bolesti njihovo predviđanje moglo bi biti od velike pomoći u prevenciji ili njihovom ranom otkrivanju. Predviđanje je moguće izvesti pomoću strojnog učenja treniranjem modela na prikladim podatcima. Za treniranje je korišten velik broj podataka koje skoro svaka osoba ima mogućnost izračunati i Python programski jezik za razvoj koda. Za prikaz stranice koristi se Python framework Streamlit koji je vrlo jednostavan i savršen za korištenje u svrhe strojnog učenja i podatkovnih znanosti. Kako bi se lakše koristila web aplikacija je postavljena na Heroku. Unošenjem atributa aplikacija vrši i ispisuje predikciju postojanja bolesti. Omogućava i izbor atributa nad kojima se trenira model kako bi se mogao pratiti utjecaj pojedinih atributa na ishod, ali točnost se smanjuje sa smanjenjem broja atributa. Također nudi se izbor između 4 algoritama inače dobrih za probleme binarne klasifikacije. Najbolja točnost daje random forest algoritam i iznosi 73,5% što pokazuje da iako ovi podatci imaju utjecaja na pojavu bolesti potrebno je još informacija kako bi se moglo točnije predvidjeti i koristiti u medicinskoj dijagnostici. S ovim postotkom mogu se samo dati smjernice za zdraviji život, posebno osobama koje su genetski sklone ovom tipu bolesti.

5. POVEZNICE I LITERATURA

Podatci za učenje modela:
https://www.kaggle.com/sulianova/cardiovascular-disease-dataset

Informacije o kvb:
https://www.who.int/health-topics/cardiovascular-diseases/#tab=tab_1
https://www.hzjz.hr/aktualnosti/kardiovaskularne-bolesti/

Heroku aplikacija:
https://cardiovascular-disease-pred.herokuapp.com/

GitHub za pristup kodu:
https://github.com/DariaC1/Heart_disease_prediction_RUSU.git

Upute za postavljanje Streamlit aplikacije na Heroku:
https://towardsdatascience.com/a-quick-tutorial-on-how-to-deploy-your-streamlit-app-to-heroku-874e1250dadd