Seminar 1

OBJAŠNJIVA UMJETNA INTELIGENCIJA (XAI)

Antonia Šarčević

Sadržaj

Uvod

Razvoj računalne snage, napredak algoritama učenja i dostupnost velike količine podataka omogućili su razvoj modela dubokog učenja koji daju izvrsne rezultate čak i za vrlo složene probleme. U današnje vrijeme takvi modeli umjetne inteligencije pronalaze široku primjenu u raznim područjima ljudskih djelatnosti, ali su i dalje neshvatljivi prosječnom korisniku što nerijetko uzrokuje nepovjerenje te se modeli čine manje pouzdanim od dobro proučenih i inherentno objašnjivih modela poput linearne i logističke regresije ili stabala odluke. Zbog toga bi mogućnost objašnjavanja modela dubokog učenja znatno doprinijela prihvaćanju dobivenih rezultata te omogućila korištenje umjetne inteligencije i za donošenje osjetljivijih odluka. Osim toga objašnjavanjem modela možemo osigurati da model nije pristran, ali i olakšati razvoj i otklanjanje grešaka u modelu. [1][2]

Zbog toga se razvila posebna grana umjetne inteligencije, XAI (eXplainable Artificial Intelligence), koja se bavi objašnjavanjem modela strojnog učenja. U nastavku će biti detaljnije objašnjeno što je XAI, vrste metoda objašnjavanja, pregled modela koji su objašnjivi sami po sebi i nekoliko popularnijih metoda za interpretaciju.

1. XAI

XAI (eXplainable Artificial Intelligence) se odnosi na skup metoda i tehnika koje omogućuju razumljivost i transparentnost odluka koje donose modeli strojnog učenja. [3] Time se osiguravaju pravednost odluke, zaštita osjetljivih informacija, robusnost modela i uzročno-posljedične veze te doprinosi povjerenju u sustav. No također treba razumjeti da nije uvijek potrebno i poželjno objašnjavati modele strojnog učenja.

Metode interpretacije mogu se klasificirati na temelju nekoliko kriterija navedenih u nastavku.

1.1. Inherentna i naknadna objašnjivost

Inherentna objašnjivost odnosi se na modele koji su lako shvatljivi zahvaljujući njihovoj jednostavnoj strukturi poput manjih stabala odluke i rijetkih linearnih modela. S druge strane, naknadna objašnjivost podrazumijeva primjenu interpretacijske metode nakon što je model već istreniran. Takve metode mogu se primijeniti i na inherentno objašnjive modele.[4]

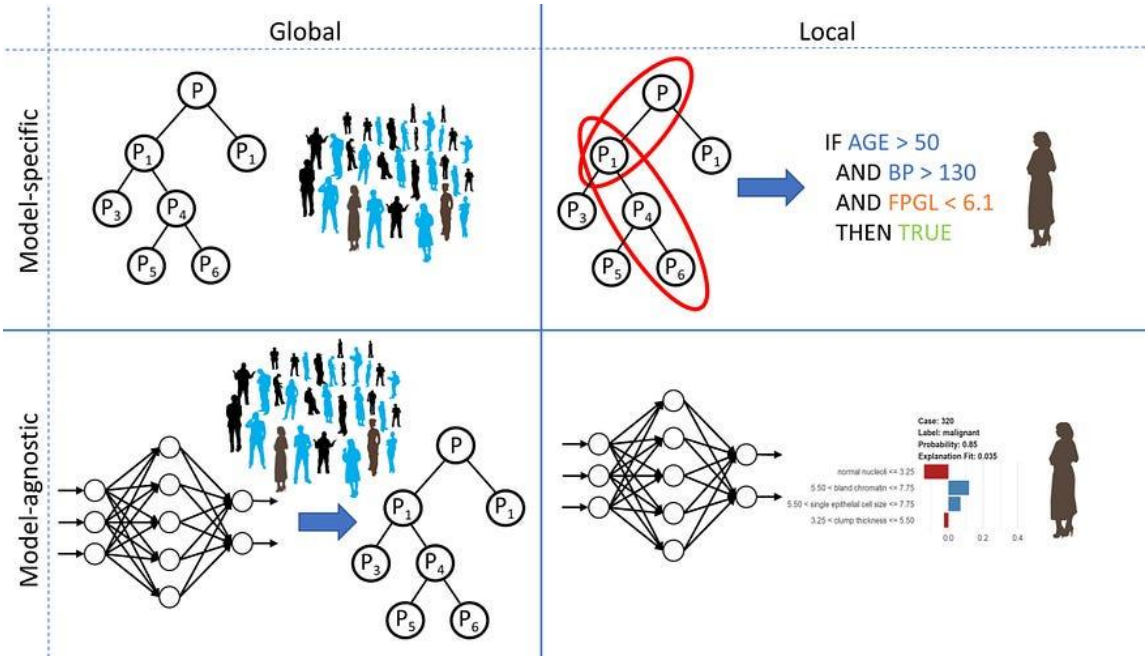
1.2. Specifične i opće metode

Specifične metode objašnjavanja ograničene su na određenu klasu modela. Primjer takve metode bila bi interpretacija težina linearnog modela koja je ograničena samo na taj skup modela ili metode specifične za neuronske mreže. S druge strane, opće metode mogu se primijeniti na bilo koji model dubokog učenja, ali naknadno, nakon treniranja modela. Takve metode se metode uglavnom baziraju na proučavanju izlaznih i ulaznih parova podataka, te nemaju pristup unutarnjim parametrima modela. [4]

1.3. Lokalne i globalne metode

Lokalne metode objašnjavaju određeni, individualni izlaz modela. Prednost lokalnog pristupa je što se čak i za složene modele lokalno predviđanje može monotono ili linearno ovisiti o određenim značajkama. Globalne metode, s druge strane, opisuju prosječno

ponašanje modela. One se primjenjuju na grupu instanci, koja se promatra kao čitav skup podataka ili primjenom lokalnih metoda na svakoj instanci. [4]



Metode objašnjavanja [5]

1.4. Rezultati interpretacijskih metoda

Interpretacijske metode mogu se podijeliti prema vrsti rezultata koje pružaju:

- **Sažeta statistika značajki** može biti sažetak za svaku značajku, poput jedne vrijednosti koja prikazuje važnost značajke, ili složenijih prikaza, kao što su interakcije između parova značajki
- **Vizualizacija sažetaka značajki** često je nužna za ispravnu interpretaciju
- **Interni parametri modela**. Interpretacija modela temelji se na lako razumljivim parametrima, poput težina u linearnim modelima ili strukture stabla odluke. Ovi parametri su specifični za model te se ponekad smatraju i sažetim statistikama značajki.
- **Točke podataka** objašnjavaju model kroz konkretne podatke, korisno za slike i tekst, ali manje za podatke s velikim brojem značajki.
- **Inherentno objašnjivi modeli.** Crne kutije mogu se objasniti pomoću aproksimativnih modela koji sami po sebi imaju jednostavnije, razumljive parametre. [4]

2. Inherentno objašnjivi modeli

Najjednostavniji način za postići objašnjivost je korištenjem modela koji su inherentno objašnjivi. Primjeri takvih modela bili bi linearna i logistička regresija te stabla odluke koji će biti detaljnije objašnjeni u nastavku.

ALGORITAM	Linearnost	Monotonost	Interakcija	Zadatak
Linearna regresija	DA	DA	NE	regr
Logistička regresija	NE	DA	NE	class
Stabla odluke	NE	Ponekad	DA	class,regr
RuleFit	DA	NE	DA	class,regr
Naivni Bayes	NE	DA	NE	class
k-najbližih susjeda	NE	NE	NE	class,regr

Tablica inherentno objašnjivih modela [4]

2.1. Linearna regresija

Linearna regresija izlaz modela računa kao sumu značajki pomnoženih s težinama što možemo zapisati kao:

$$h(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n$$

Iz ovakve hipoteze možemo jednostavno i intuitivno odrediti utjecaj pojedinih značajki na izlaz modela. Generalno možemo reći da značajke čije težine imaju veću apsolutnu vrijednost više doprinose predikciji modela. Naime važnost značajke računa se kao t-statistika, odnosno apsolutna vrijednost pripadajuće težine skalirana standardnom pogreškom.

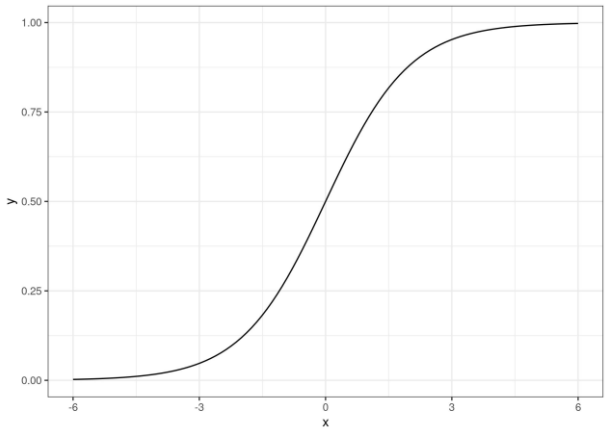
$$t_{w_i} = w_i / SE(w_i)$$

No kako bi model bio primjenjiv podaci moraju zadovoljavati određene zahtjeve poput: linearnosti, normalnosti, nezavisnosti, imati konstantne varijance i fiksirane oznake te ne smiju biti multikolinearni. Zbog relativno strogih zahtjeva ovakav model često nije dovoljno dobar za složenije probleme i skupove podataka. [4]

2.2. Logistička regresija

Model logističke regresije koristi logističku funkciju kako bi dobili izlaz između 0 i 1 koji interpretiramo kao vjerojatnost pripadnosti primjera određenoj klasi.

$$\text{logistic}(\eta) = 1 \,/\, (1+\exp(-\eta))$$



Logistička funkcija [4]

Model logističke regresije:

$$P(y_{(i)}=1) = 1/\,(1+\exp(-(\beta_0+\beta_1 x_{(i)1}+\ldots+\beta_p x_{(i)p})))$$

Zbog primjene logističke funkcije težine više ne utječu na izlaz linearno kao što je to bio slučaj za linearnu regresiju već promjena značajke za mjeru veličine mijenja omjer vjerojatnosti s faktorom $\exp(\beta)$ po izrazu:

$$\text{odds}_{x_j+1} \,/\, \text{odds}_{x_j} = \exp(\beta_j(x_j+1) - \beta_j x_j) = \exp(\beta_j)$$

koji slijedi iz:

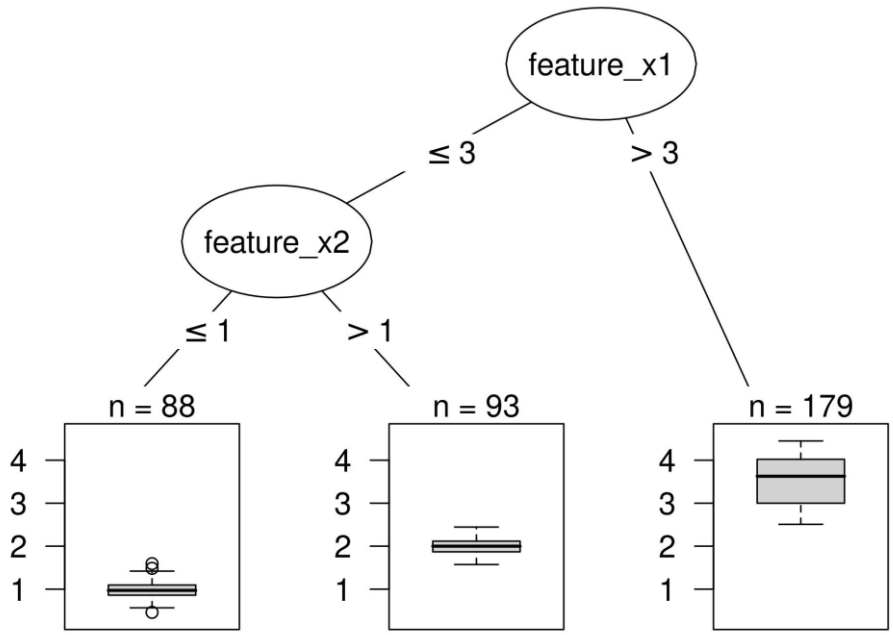
$$P(y=1) \,/\, (1-P(y=1)) = \text{odds} = \exp(\beta_0+\beta_1 x_1+\ldots+\beta_p x_p)$$

Iz toga je vidljivo da je interpretacija logističkih modela nešto zahtjevnija od interpretacije linearnih modela. [4]

2.3. Stabla odluke

Velika prednost stabala odluke je što ne zahtijevaju linearnost podataka već se baziraju na podjeli skupa podataka na maje skupove te se mogu koristiti iza klasifikaciju i za probleme regresije. Odnos izlaza y i ulaza x prikazan je formulom:

$$y = f(x) = \sum_{m=1}^{M} c_m I\{x \in R_m\}$$



Primjer stabla odluke [4]

Ovakva stabla vrlo su jednostavna za interpretaciju, počevši od korijena spušta se rema određenom podskupu podataka prema listovima koji određuju izlaz modela. Svi uvjeti na putu povezani su I operatorom.

Općenito važnost značajki u stablu odluke računa se tako da se prolaskom po putu uzima u obzir koliko je značajka smanjila entropiju u odnosu na roditeljski čvor.

Prednosti ovakvog pristupa su to što su stabla odluke idealna za opisivanje interakcija između značajki te stvara jasne grupe koje je jednostavno shvatiti i vizualizirati. No s druge strane nije dovoljno dobro rješenje za linearne podatke, te mu nedostaje zaglađenosti (oštre granice/podjele grupa) i poprilično je nestabilan model. [4]

3. Metode objašnjavanja

Jednostavniji modeli poput stabla odluke i linearnih modela mogu se jednostavno objasniti (white-box modeli), ali ne daju dobre rezultate na složenim skupovima podataka i kompleksnijim problemima. S druge strane modeli poput dubokih neuronskih mreža, dobro predviđaju i za složene probleme, ali ne znamo zašto donose pojedine odluke to jest ne daju zadovoljavajuća objašnjenja odluka, sama po sebi (black-box modeli). Zbog toga se za objašnjavanje ovakvih modela koriste ad-hoc metode koje interpretiraju rezultate neovisno o modelu. U nastavku je prikazano nekoliko popularnijih metoda za objašnjavanje modela. Sve navedene metode pridodaju određene vrijednosti značajkama ulaza zbog čega ih nazivamo atributnim metodama objašnjavanja.

3.1. LIME

LIME (Local Interpretable Model-agnostic Explanations) je kao što joj ime kaže opća metoda primjenjiva na sve vrste modela strojnog učenja, a objašnjava ponašanje modela lokalno, na nekom primjeru.

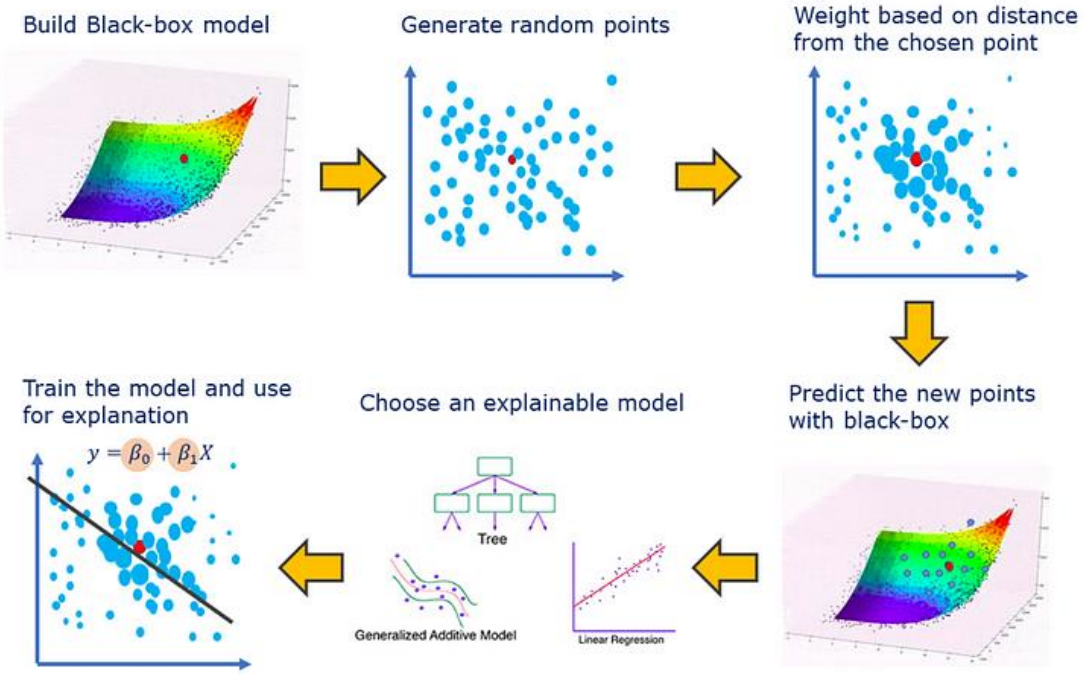
3.1.1. Algoritam

- Odabir modela i referentne točke
- Generiranje novog seta ulaznih podataka uzorkovanjem iz normalne distribucije odgovarajuće setu za treniranje
- Generiranje predviđanja korištenjem modela koji želimo objasniti
- Dodjeljivanje težina temeljenih na udaljenosti od referentne točke (koristi se RBF Kernel koji dodjeljuje veće težine bližim točkama)

$$RBF(x^{(i)}) = \exp\left(-\frac{||x^{(i)} - x^{(ref)}||^2}{kw}\right)$$

Gaussian Kernel formula, kw parametar određuje koliko je velik značajni krug značajki

- Treniranje surogat modela na generiranom skupu s težinama koji aproksimira ponašanje početnog modela u referentnoj točki i onda se on koristi za objašnjavanje. Surogat model može biti bilo koji inherentno objašnjivi model, a kao podrazumijevani model u Pythonu koristi se Ridge regresija. [6]

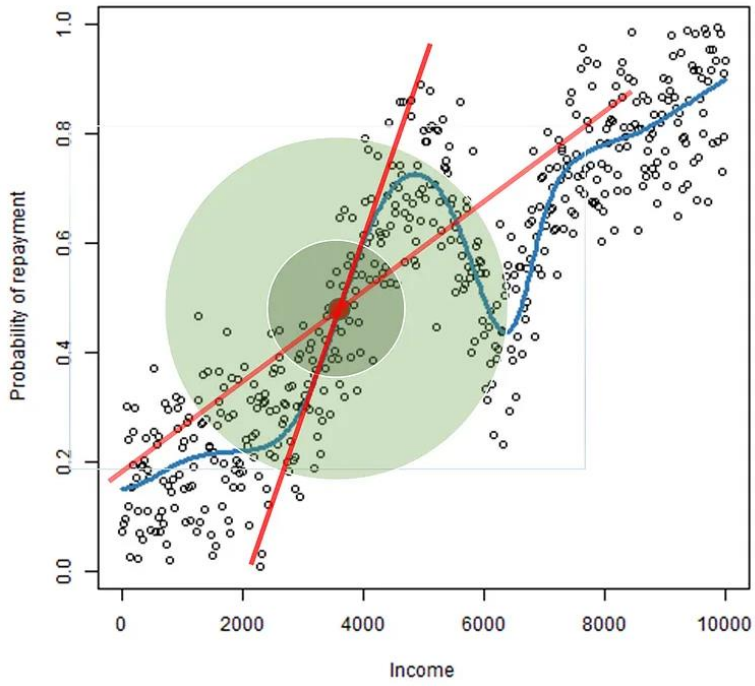


Koraci LIME algoritma [6]

3.1.2. Prednosti i nedostaci

Prednost LIME metode je njena jednostavnost i primjenjivost, naime LIME se može koristiti na različitim modelima i skupovima podataka poput tabličnih, tekstualnih i slikovnih podataka.

S druge strane problem generiranja podataka i dalje se raspravlja. Naime algoritam generira nasumične točke u prostoru, ali težine se primjenjuju samo na točke koje su dovoljno blizu referenci. Zašto onda ne generirati samo točke koje su blizu i kako odrediti na koje točke treba primijeniti težine. Ukoliko je maksimalna udaljenost prevelika, problem se neće moći uvijek dobro aproksimirati, s druge strane ograničavanje na premalu udaljenost rezultira nestabilnim modelima. [6]



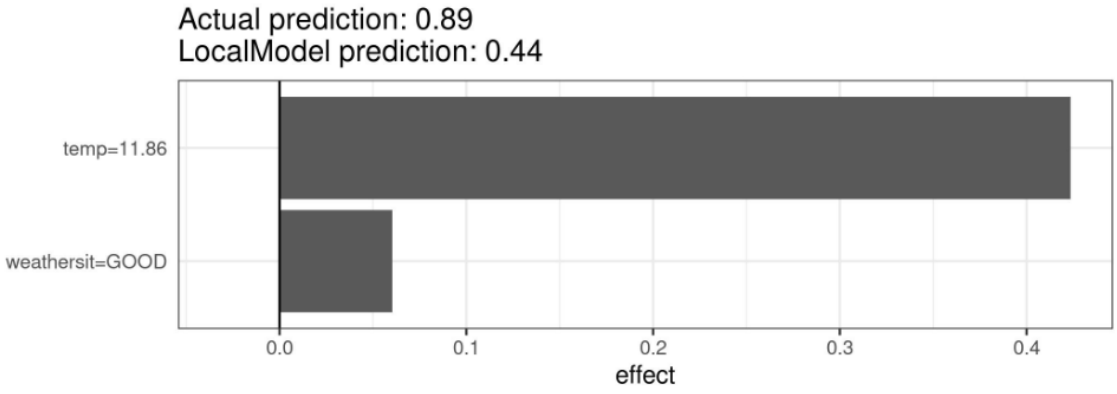
Ovisnost interpretacije o širini kernela [6]

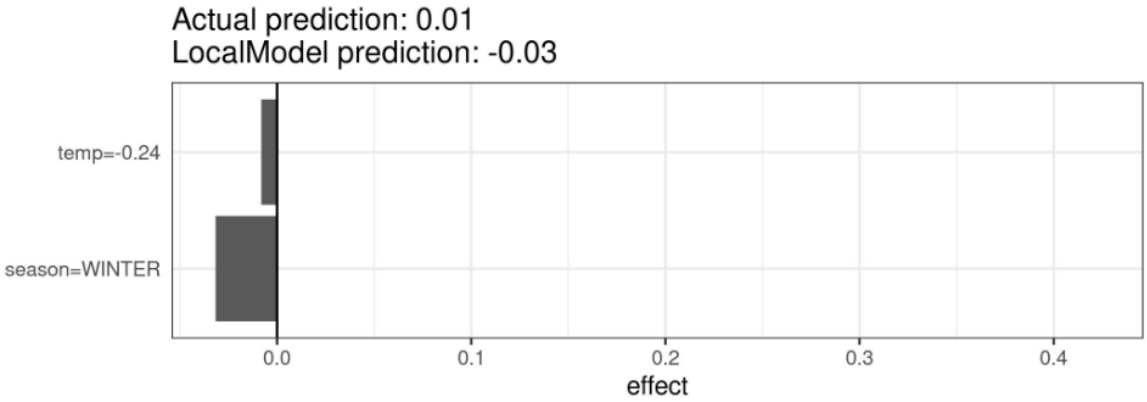
3.1.3. Primjeri

LIME podržava različite tipove podataka, a u nastavku će biti pokazani primjeri za tablične, tekstualne i slikovne podatke.

- **Tablični podaci:**

  Problem klasifikacije s podacima o najmu bicikala na temelju kojih se predviđa hoće li broj iznajmljivanja premašiti prosjek. Trenirana je slučajna šuma sa 100 stabala, te se pomoću LIME metode objašnjava predviđanje za dva primjera na temelju informacija o vremenu i temperaturi, pokazujući kako specifične značajke utječu na rezultate.





Visoka temperatura i dobro vrijeme imaju pozitivan utjecaj na predikciju. Efekt predstavlja težinu pomnoženu s vrijednošću značajke.

- **Tekstualni podaci:**

LIME za tekst funkcionira stvaranjem varijacija izvornog teksta uklanjanjem nasumičnih riječi te predstavlja svaku riječ kao binarnu (1 ako je uključena, 0 ako nije). Primjer uključuje klasifikaciju komentara na YouTubeu kao spam ili normalne. Vjerojatnost predikcije modela mijenja se ovisno o prisutnosti ili odsutnosti specifičnih riječi, pri čemu je "channel" prepoznat kao snažan indikator spama.

	SADRŽAJ	**RAZRED**
267	PSY is a good guy	0
173	For Christmas Song visit my channel! ;)	1

For	**Christmas**	**Song**	**visit**	**my**	**channel!**	**;)**	**VJV.**	**TEŽINA**
1	0	1	1	0	0	1	0.17	0.57
0	1	1	1	1	0	1	0.17	0.71
1	0	0	1	1	1	1	0.99	0.71
1	0	1	1	1	1	1	0.99	0.86
0	1	1	1	0	0	1	0.17	0.57

Slučaj	Vjerojatnost labele	Značajka	Težina značajki
1	0.1701170	is	0.000000
1	0.1701170	good	0.000000
1	0.1701170	a	0.000000
2	0.9939024	channel!	6.180747
2	0.9939024	;)	0.000000
2	0.9939024	visit	0.000000

- **Slikovni podaci:**

Za slike LIME ne mijenja pojedinačne piksele, već segmentira slike u takozvane superpiksele (grupe sličnih piksela). Primjer uključuje klasifikaciju slike kruha koristeći Googleov Inception V3 model, s objašnjenjima za oznake "Bagel" i "Strawberry." Zeleni segmenti povećavaju vjerojatnost klase, dok crveni smanjuju, pružajući vizualni uvid u to kako LIME ističe ključna područja na slici.

** Svi primjeri preuzeti su iz [4]

3.2. Shapely values

Ova metoda koristi se za prikaz važnosti pojedinih značajki za donošenje odluke, a bazira se na izračunu Shapelyjeve vrijednosti. Ideja potječe iz teorije kooperativnih igara i temelji se na podjeli zajedničkog dobitka ovisno o zaslugama pojedinih igrača. Ukoliko zadatak modela promatramo kao igru, a izlaz kao dobitak možemo igrače povezati s značajkama.

Shapelyjeve vrijednosti računaju se kao prosjek marginalnih doprinosa pojedinih značajki koji odgovaraju razlici u izlazu modela za ulaz sa i bez promatrane značajke za svaki podskup skupa preostalih značajki. Svakom doprinosu pridodaju se težine ovisno o udjelu značajki koje obuhvaćaju.

Formula za Shapelyjeve vrijednost za značajku i s funkcijom doprinosa v:

$$\phi_i(v) = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [v(S \cup i) - v(S)]$$

$$v_x(S) = \int f(x_1, ..., x_F) d\mathbb{P}_{x \notin S} - \mathbb{E}_X[f]$$

Pri čemu S označava broj značajki u određenom podskupu, a F ukupan broj značajki. Iz izraza za težinu (multinomijalni koeficijent) da metoda više kažnjava podskupove koji čiji je broj elemenata udaljeniji od 0 ili F. [9]

3.2.1. Algoritam

- Enumerirati sve moguće podskupove značajki za svaki primjer
- Izračunati predikcije za svaki podskup
- Pronaći marginalni doprinos značajke unutar svake predikcije podskupa
- Prosječno vrednovanje tih doprinosa kako bi se dobila Shapleyjeve vrijednost za tu značajku

3.2.2. Prednosti i nedostaci

Jedna od najvećih prednosti ove metode je njena pravednost u smislu da garantira teoretski pravedan način za raspodjelu zasluga svakoj značajki. Osmi toga ova metoda ne ovisi o modelu što ju čini široko primjenjivom.

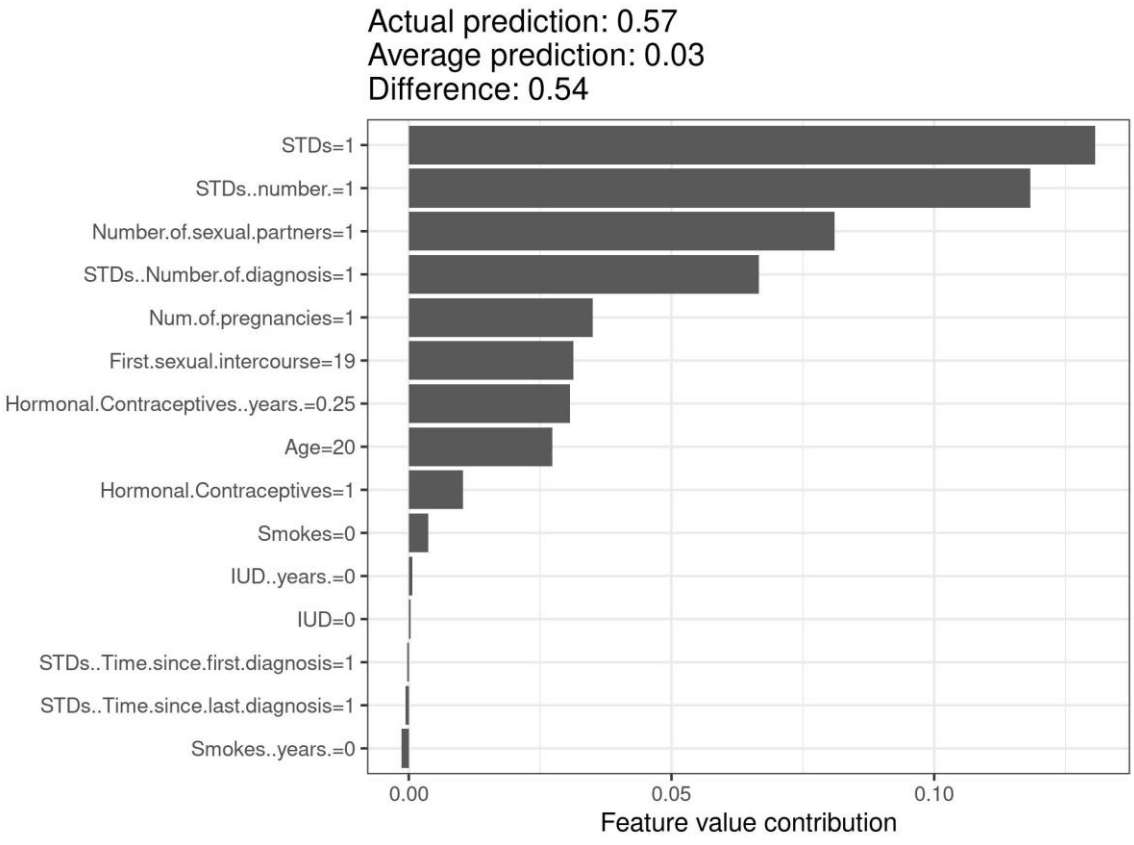
S druge strane složenost ovakvog postupka raste eksponencijalno ovisno o broju značajki, što metodu čini nepraktičnom za modele s velikom dimenzijom ulaza.

U praksi se često koriste aproksimativne metode poput SHAP (SHapley Additive exPlanations) koje ubrzavaju izračun Shapleyjeve vrijednosti, a zadržavaju većinu interpretabilnosti.

3.2.3. Primjeri

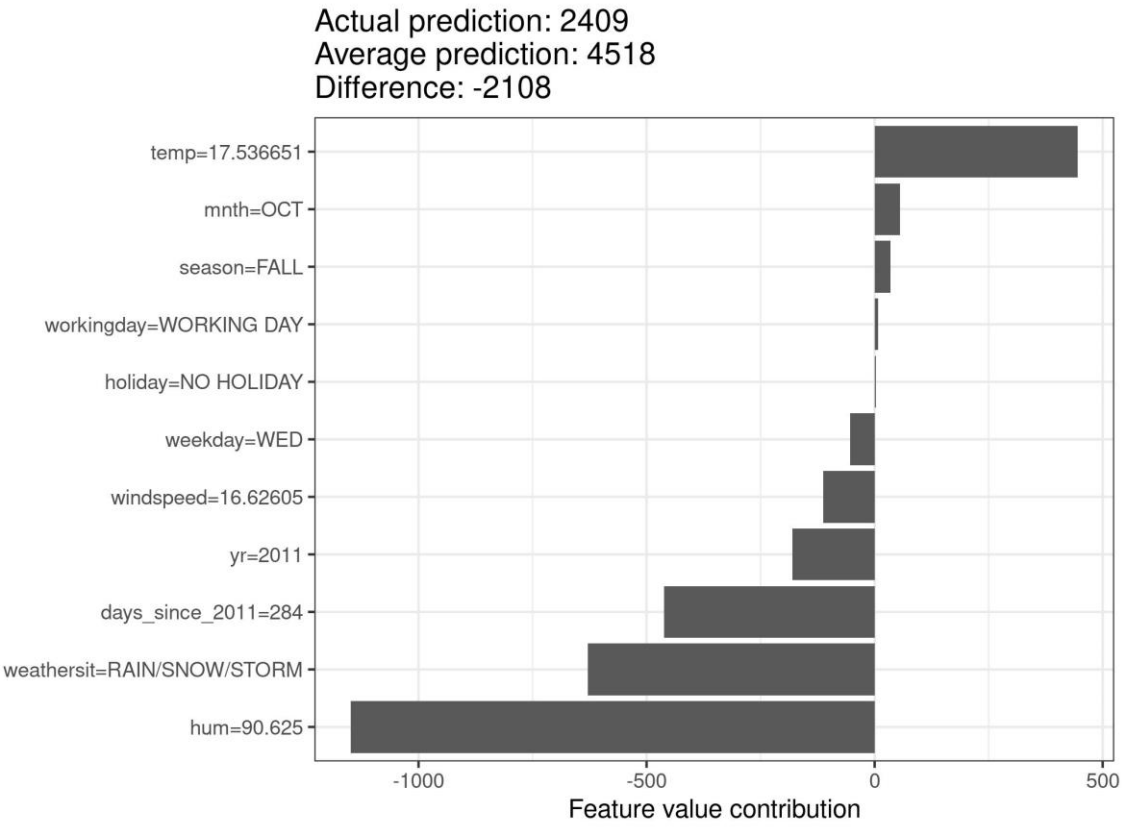
Interpretacija Shapleyjeve vrijednosti za značajku j je sljedeća: Vrijednost j-te značajke pridonijela je predikciji za određeni primjer s vrijednošću ϕ_j u usporedbi s prosječnom predikcijom za cijeli skup podataka. Shapleyjeva vrijednost primjenjiva je i na klasifikaciju (kada radimo s vjerojatnostima) i na regresiju.

Analiza predikcije pomoću modela slučajne šume za procjenu rizika raka vrata maternice:



Za ženu iz skupa podataka s predikcijom od 0,57, rizik od raka je 0,54 iznad prosječne predikcije (0,03). Najveći doprinos povećanju vjerojatnosti imala je dijagnosticirana spolno prenosiva bolest (STD). Zbroj doprinosa odgovara razlici između stvarne i prosječne predikcije (0,54).

Kod skupa podataka o najmu bicikala:



Za određeni dan (dan 285), predviđena je vrijednost od 2409 bicikala, što je -2108 ispod prosjeka (4518). Vremenski uvjeti i vlažnost dali su najveći negativni doprinos, dok je temperatura imala pozitivan utjecaj. Zbroj Shapleyjevih vrijednosti daje razliku između stvarne i prosječne predikcije (-2108).

Važno je točno interpretirati Shapleyjevu vrijednost: ona predstavlja prosječni doprinos određene značajke predikciji unutar različitih kombinacija značajki. Shapleyjeva vrijednost nije jednostavno razlika u predikciji ako bismo značajku izostavili iz modela.

** Svi primjeri preuzeti su iz [4]

3.3. Saliency maps (Pixel Attribution)

Saliency mape su tehnika objašnjavanja strojnog učenja koja se koristi u računalnom vidu kako bi se vizualiziralo koji dijelovi slike najviše doprinose odluci neuronske mreže. Ove mape pokazuju relevantne dijelove slike ističući piksele koji imaju najveći utjecaj na odluku modela. To omogućuje razumijevanje onoga što model smatra važnim za donošenje određene odluke. [11]

Metode poput LIME i Shapely values manipuliraju djelom ulaza kako bi generirali rješenja. S druge strane neke metode poput Sailency mapa računaju gradijent izlaza u odnosu na ulazne značajke. To jest izračunava se kako se promjena pojedinog piksela odražava na promjenu izlaza modela. Rezultat se uglavnom prikazuje u obliku mapi, gdje su važnija područja označena svjetlijim ili istaknutijim bojama. [4]

Sailency mape [11]

3.3.1. Algoritam

- Primijeniti prolaz unaprijed na zadanu sliku to jest izračun izlaza istreniranog modela
- Unatražnom propagacijom kroz mrežu računa se gradijent izlazne vrijednosti u odnosu na svaki piksel.

$$E_{grad}(I_0) = \frac{\delta S_c}{\delta I}|_{I=I_0}$$

- Vizualizacija gradijenata. Moguće je prikazati apsolutne vrijednosti gradijenta za pojedini piksel ili naglasiti negativne i pozitivne doprinose odvojeno [4]

3.3.2. Prednosti i nedostaci

Velika prednost Sailency mapa je vizualizacija objašnjenja što ga čini lakšim za shvatiti. Osim toga metode bazirane na gradijentima su brže od metoda koje manipuliraju ulaz kao LIME i Shapely values.

S druge strane teško je raspoznati je li objašnjenje doista točno te se pokazalo da je metoda nestabilna. Naime za male promjene u slici koje ne utječu na klasifikaciju došlo je do velikih promjena u objašnjenju modela. Sve to ovu metodu čini vrlo nepouzdanom. [4]

4. Primjeri

U nastavku će biti prikazani primjeri(s kôdom) korištenja navedenih metoda za objašnjavanje modela treniranih na skupovima podataka iz područja medicine.

4.1. LIME i Shapely values na skupu podataka o srčanim bolestima

U ovom djelu bit će prikazan problem klasifikacije stupnja prisutnosti srčanih bolesti kod pacijenata pomoću inherentno objašnjivog modela stabla odluke i modela slučajne šume koji će biti pojašnjen metodama LIME i Shapely values.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns

from ucimlrepo import fetch_ucirepo

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from pydl85 import DL85Classifier
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

from lime.lime_tabular import LimeTabularExplainer
import shap

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

4.1.1. Pregled skupa podataka

Radi se o multivarijatnoj bazi podataka iz područja zdravlja i medicine, koja je namijenjena klasifikacijskim zadacima. Značajke unutar baze su različitih tipova, uključujući kategorijske, cijelobrojne i realne vrijednosti. Baza sadrži ukupno 303 instance s 76 atributa, no svi objavljeni eksperimenti koriste podskup od 14 atributa (13 značajki i klasifikacija).

U istraživanjima strojnog učenja najčešće se koristi Cleveland baza podataka, a cilj (oznaka "goal") odnosi se na prepoznavanje prisutnosti srčanih bolesti kod pacijenta. Vrijednost cilja može biti cijeli broj od 0 (odsutnost bolesti) do 4 (prisutnost bolesti u različitim stupnjevima). Eksperimenti s Cleveland bazom uglavnom se fokusiraju na razlikovanje između prisutnosti (vrijednosti 1, 2, 3, 4) i odsutnosti (vrijednost 0) bolesti.

```
# fetch dataset
heart_disease = fetch_ucirepo(id=45)

# data (as pandas dataframes)
X = heart_disease.data.features
y = heart_disease.data.targets

print(heart_disease.variables)
```

	name	role	type	demographic	description	units	missing_values
0	age	Feature	Integer	Age	None	years	no
1	sex	Feature	Categorical	Sex	None	None	no
2	cp	Feature	Categorical	None	None	None	no
3	trestbps	Feature	Integer	None	resting blood pressure (on admission to the ho...	mm Hg	no
4	chol	Feature	Integer	None	serum cholestoral	mg/dl	no
5	fbs	Feature	Categorical	None	fasting blood sugar > 120 mg/dl	None	no
6	restecg	Feature	Categorical	None	None	None	no
7	thalach	Feature	Integer	None	maximum heart rate achieved	None	no
8	exang	Feature	Categorical	None	exercise induced angina	None	no
9	oldpeak	Feature	Integer	None	ST depression induced by exercise relative to ...	None	no
10	slope	Feature	Categorical	None	None	None	no
11	ca	Feature	Integer	None	number of major vessels (0-3) colored by flour...	None	yes
12	thal	Feature	Categorical	None	None	None	yes
13	num	Target	Integer	None	diagnosis of heart disease	None	no

```
print(X)
```

```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0     63    1   1       145   233    1        2      150      0      2.3
1     67    1   4       160   286    0        2      108      1      1.5
2     67    1   4       120   229    0        2      129      1      2.6
3     37    1   3       130   250    0        0      187      0      3.5
4     41    0   2       130   204    0        2      172      0      1.4
..   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
298   45    1   1       110   264    0        0      132      0      1.2
299   68    1   4       144   193    1        0      141      0      3.4
300   57    1   4       130   131    0        0      115      1      1.2
301   57    0   2       130   236    0        2      174      0      0.0
302   38    1   3       138   175    0        0      173      0      0.0

     slope   ca  thal
0        3  0.0   6.0
1        2  3.0   3.0
2        2  2.0   7.0
3        3  0.0   3.0
4        1  0.0   3.0
..     ...  ...   ...
298      2  0.0   7.0
299      2  2.0   7.0
300      2  1.0   7.0
301      2  1.0   3.0
302      1  NaN   3.0

[303 rows x 13 columns]
```

```
print(y)
```

```
     num
0      0
1      2
2      1
3      0
4      0
..   ...
298    1
299    2
300    3
301    1
302    0

[303 rows x 1 columns]
```

```
print('Missing values:\n' + str(np.isnan(X).sum()))
print('\nDuplicated: ' + str(X.duplicated().sum()))
```

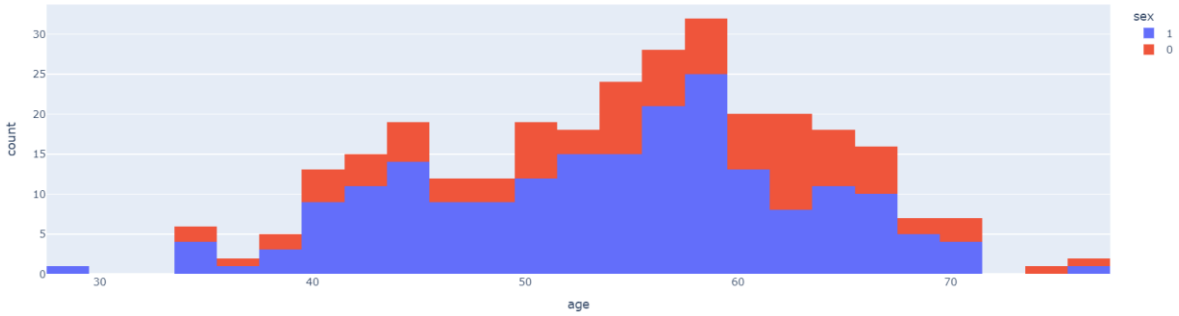
```
Missing values:
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          4
thal        2
dtype: int64

Duplicated: 0
```

```
valid_rows = ~np.isnan(X).any(axis=1)

X = X[valid_rows]
y = y[valid_rows]

print(X.shape, y.shape)
print('Missing values:\n' + str(np.isnan(X).sum()))
```

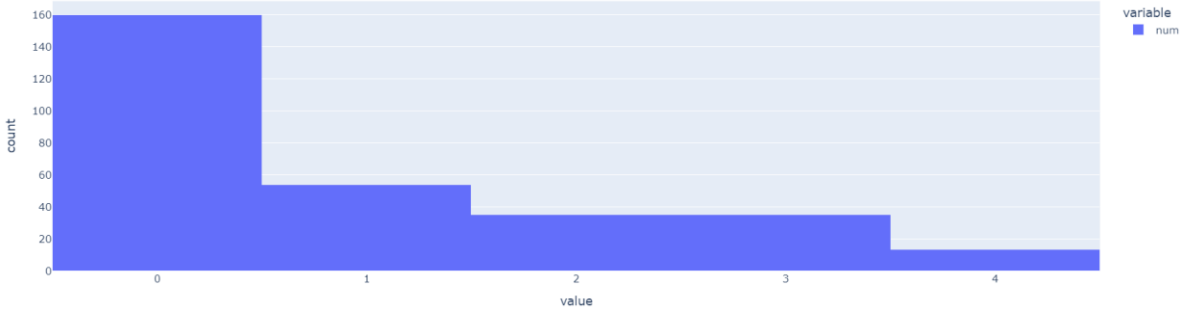
```
(297, 13) (297, 1)
Missing values:
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
dtype: int64
```

```
fig = px.histogram(data_frame=X, x='age', color= 'sex')
fig.show()

# 1-male, 0-female
X['sex'].value_counts()
```

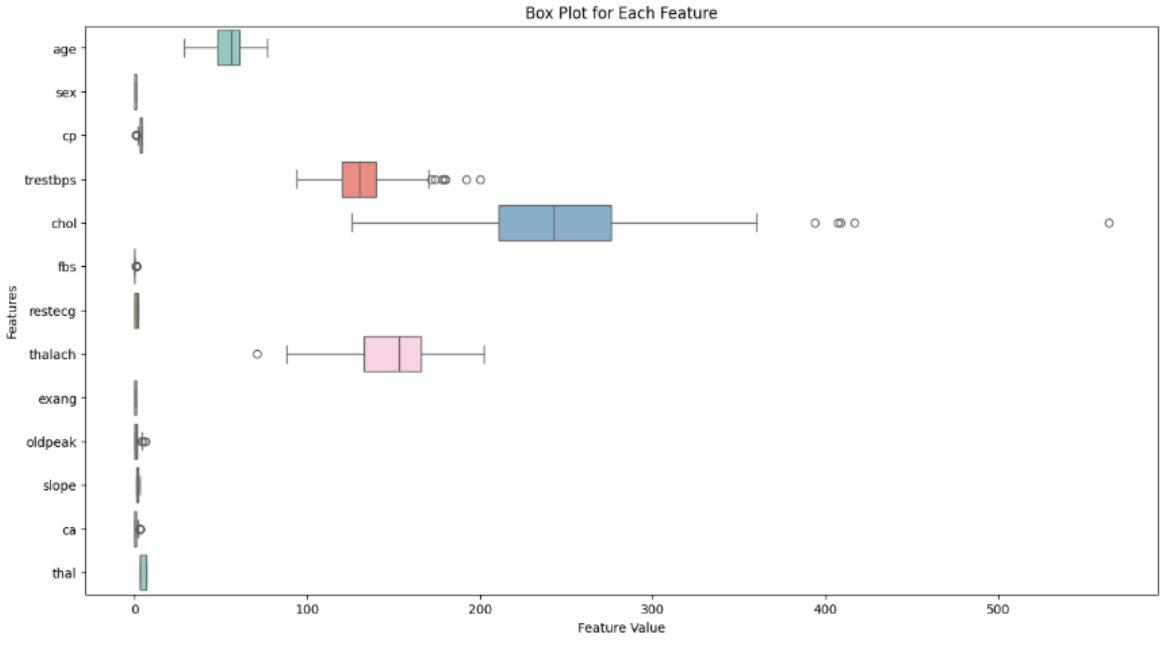
```
sex
1    201
0     96
Name: count, dtype: int64
```




```
fig = px.histogram(data_frame=y)
fig.show()
```




```
def box_plot(X):
    plt.figure(figsize=(15, 8))
    sns.boxplot(data=X, orient="h", palette="Set3")
    plt.title("Box Plot for Each Feature")
    plt.xlabel("Feature Value")
    plt.ylabel("Features")
    plt.show()
box_plot(X)
```



```
X_train, X_test, y_train, y_test = train_test_split(X, y.values.ravel(), test_size=0.25, random_state=42)

encoder = OneHotEncoder(handle_unknown='ignore')
X_train_bin = encoder.fit_transform(X_train)
X_test_bin = encoder.transform(X_test)
```

4.1.2. Stablo odluke

```
dl85 = DL85Classifier(max_depth=3, min_sup=5)
dl85.fit(X_train_bin.toarray(), y_train)

y_pred_dl85 = dl85.predict(X_test_bin.toarray())
print(y_pred_dl85)
```

```
[0, 2, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 2, 0, 0, 0, 3, 0, 0, 3, 0, 3, 0, 1, 0, 1, 3, 0, 0, 0, 0,
3, 0, 0, 0, 3, 0, 3, 0, 0, 1, 3, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 3, 0, 0, 1, 2, 3, 3, 0, 1]
```

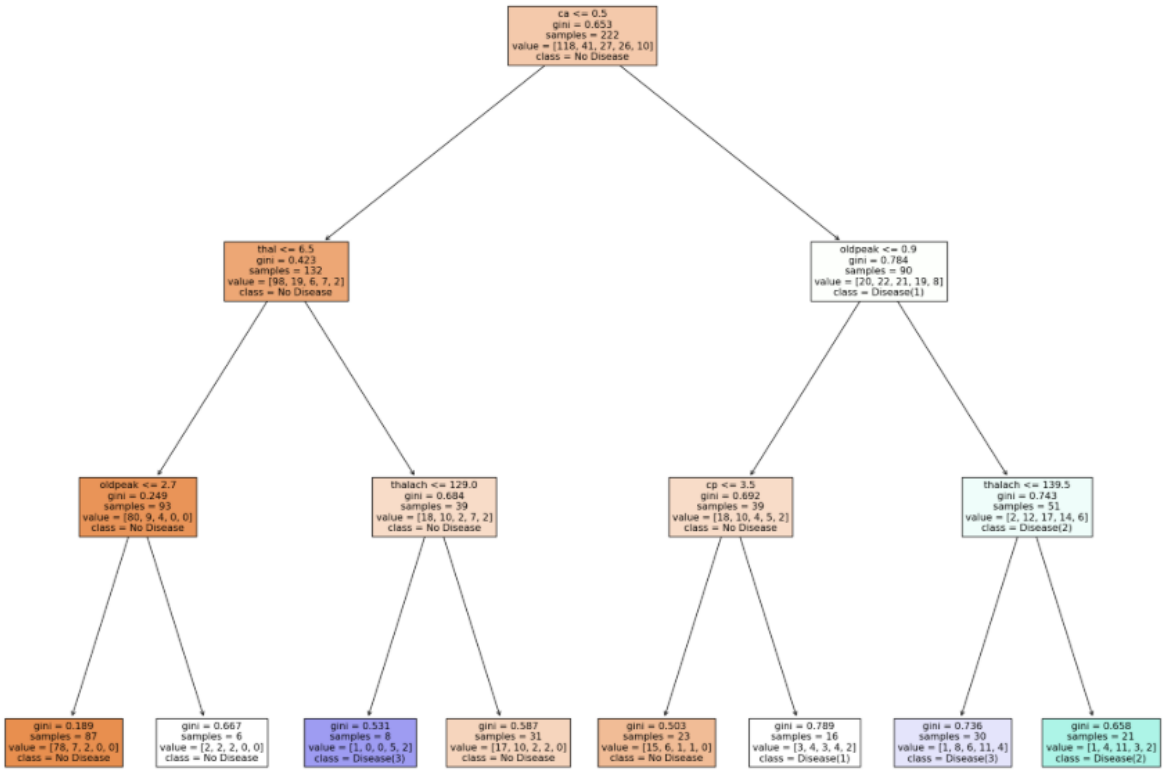
```
dt = DecisionTreeClassifier(max_depth=3, random_state=42)
dt.fit(X_train, y_train)

y_pred_dt = dt.predict(X_test)
print(y_pred_dt)
```

```
[0 1 0 3 0 2 0 2 0 0 0 0 0 1 0 0 0 0 3 0 2 2 0 0 3 0 3 0 0 0 0 0 3 0 0 0 0
 0 2 1 0 3 3 3 0 0 0 0 2 0 0 2 0 0 1 3 0 0 2 0 0 0 2 0 0 0 0 0 3 1 1 3 2 0
 2]
```

4.1.2.1Vizualizacija

```
text_representation = tree.export_text(dt)
print(text_representation)

fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(dt,
                   feature_names=X.columns,
                   class_names=['No Disease', "Disease(1)", "Disease(2)", "Disease(3)", "Disease(4)"],
                   filled=True)
```

```
|--- feature_11 <= 0.50
|   |--- feature_12 <= 6.50
|   |   |--- feature_9 <= 2.70
|   |   |   |--- class: 0
|   |   |--- feature_9 >  2.70
|   |   |   |--- class: 0
|   |--- feature_12 >  6.50
|   |   |--- feature_7 <= 129.00
|   |   |   |--- class: 3
|   |   |--- feature_7 >  129.00
|   |   |   |--- class: 0
|--- feature_11 >  0.50
|   |--- feature_9 <= 0.90
|   |   |--- feature_2 <= 3.50
|   |   |   |--- class: 0
|   |   |--- feature_2 >  3.50
|   |   |   |--- class: 1
|   |--- feature_9 >  0.90
|   |   |--- feature_7 <= 139.50
|   |   |   |--- class: 3
|   |   |--- feature_7 >  139.50
|   |   |   |--- class: 2
```



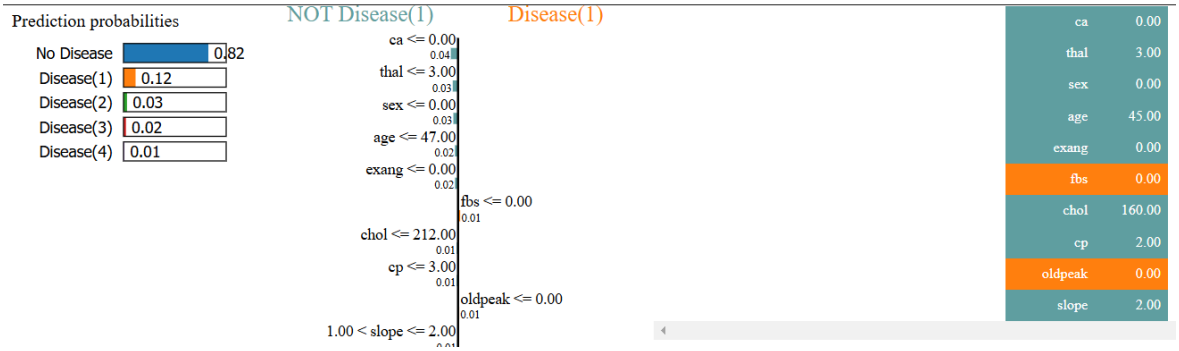
4.1.3. Slučajna šuma

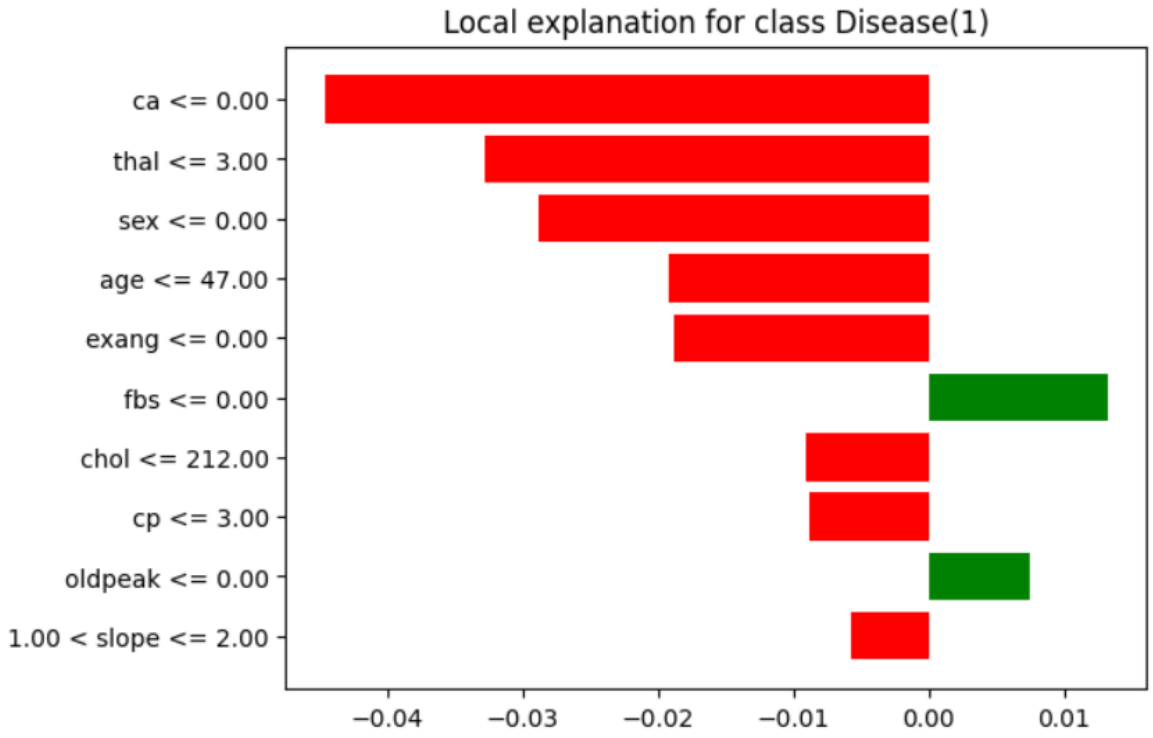
```
rf = RandomForestClassifier(n_estimators=100, max_depth=5, min_samples_split=10, min_samples_leaf=5, random_state=42)
rf.fit(X_train, y_train)

y_pred_rf = rf.predict(X_test)
print(y_pred_rf)
```

```
[0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 3 0 0 0 0 0 3 0 0 0 0
 0 2 0 0 2 3 0 0 0 0 0 0 0 0 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 2 2 0
 2]
```

4.1.3.1LIME

```
explainer = LimeTabularExplainer(X_train.values,
                                 feature_names = X.columns,
                                 class_names = ['No Disease','Disease(1)','Disease(2)','Disease(3)','Disease(4)'],
                                 mode = 'classification')

explanation = explainer.explain_instance(
    data_row=X_test.iloc[0].values,
    predict_fn=lambda x: rf.predict_proba(pd.DataFrame(x, columns=X_train.columns))
)

explanation.show_in_notebook(show_table=True, show_all=False)

fig = explanation.as_pyplot_figure()
plt.show()
```

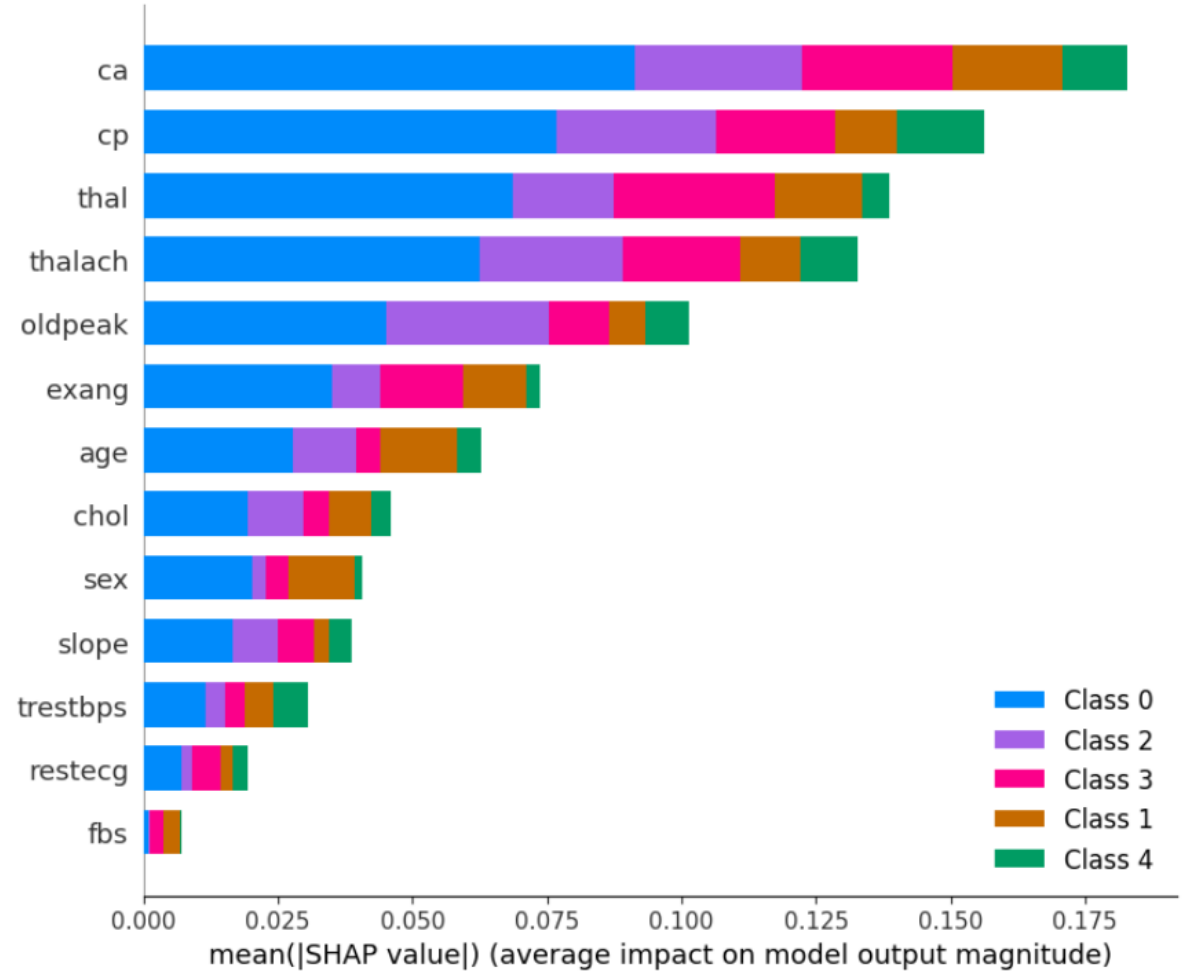




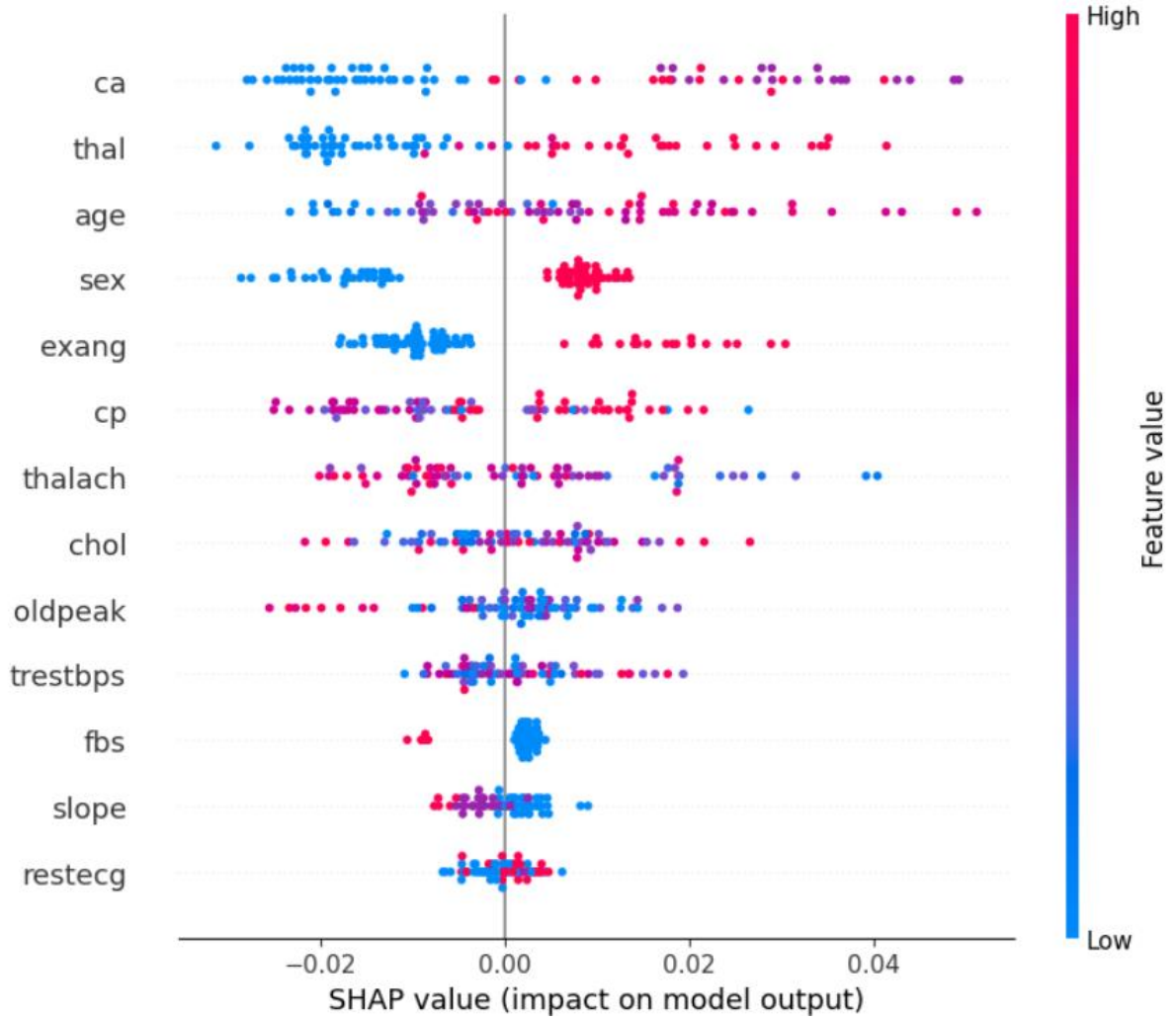
4.1.3.2Shapley values

```
shap.initjs()
explainer = shap.TreeExplainer(rf)
shap_values = explainer.shap_values(X_test)

print("Variable Importance Plot - Global Interpretation")
figure = plt.figure()
shap.summary_plot(shap_values, X_test)
```

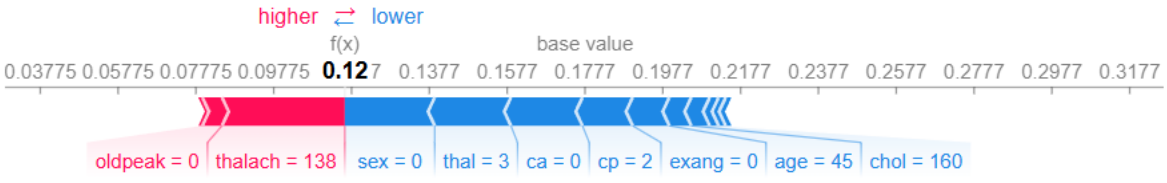
Variable Importance Plot - Global Interpretation



```
shap.summary_plot(shap_values[1], X_test)
```




```
explainer = shap.TreeExplainer(rf)
shap_values = explainer.shap_values(X_test.iloc[0])
shap.initjs()
shap.force_plot(explainer.expected_value[1], shap_values[1], X_test.iloc[0])
```



4.1.4. Evaluacija

4.1.4.1Stablo odluke

```
print("Accuracy:", accuracy_score(y_test, y_pred_dl85))
print("Classification Report:\n", classification_report(y_test, y_pred_dl85, zero_division = 0))
print(confusion_matrix(y_test, y_pred_dl85))
```

```
Accuracy: 0.6266666666666667
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.95      0.84        42
           1       0.40      0.15      0.22        13
           2       0.33      0.12      0.18         8
           3       0.29      0.44      0.35         9
           4       0.00      0.00      0.00         3

    accuracy                           0.63        75
   macro avg       0.35      0.34      0.32        75
weighted avg       0.56      0.63      0.57        75

[[40  0  1  1  0]
 [ 6  2  1  4  0]
 [ 2  1  1  4  0]
 [ 3  2  0  4  0]
 [ 2  0  0  1  0]]
```

```
print("Accuracy:", accuracy_score(y_test, y_pred_dt))
print("Classification Report:\n", classification_report(y_test, y_pred_dt, zero_division = 0))
print(confusion_matrix(y_test, y_pred_dt))
```

```
Accuracy: 0.5866666666666667
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.86      0.81        42
           1       0.33      0.15      0.21        13
           2       0.18      0.25      0.21         8
           3       0.36      0.44      0.40         9
           4       0.00      0.00      0.00         3

    accuracy                           0.59        75
   macro avg       0.33      0.34      0.33        75
weighted avg       0.55      0.59      0.56        75

[[36  1  4  1  0]
 [ 7  2  1  3  0]
 [ 2  2  2  2  0]
 [ 1  1  3  4  0]
 [ 1  0  1  1  0]]
```

4.1.4.2Slučajna šuma

```
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
print("Classification Report:\n", classification_report(y_test, y_pred_rf, zero_division = 0))
print(confusion_matrix(y_test, y_pred_rf))
```

```
Accuracy: 0.5866666666666667
Classification Report:
              precision    recall  f1-score   support

           0       0.69      1.00      0.82        42
           1       0.00      0.00      0.00        13
           2       0.25      0.25      0.25         8
           3       0.00      0.00      0.00         9
           4       0.00      0.00      0.00         3

    accuracy                           0.59        75
   macro avg       0.19      0.25      0.21        75
weighted avg       0.41      0.59      0.48        75

[[42  0  0  0  0]
 [10  0  2  1  0]
 [ 4  1  2  1  0]
 [ 3  2  4  0  0]
 [ 2  0  0  1  0]]
```

4.2. Saliency mape na skupu podataka o raku kože

U nastavku će biti prikazana klasifikacija slika pigmentiranih lezija konvolucijskom neuronskom mrežom (CNN) te primjena Saliency mapa za objašnjavanje tog modela.

```
import pandas as pd

import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import numpy as np
import pandas as pd
import tensorflow as tf
import keras
import sklearn.metrics as metrics
from imblearn.over_sampling import RandomOverSampler
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Flatten, Dense, MaxPool2D
```

4.2.1. Pregled skupa podataka

HAM10000 ("Human Against Machine with 10000 training images") je skup 10015 dermatoskopskih slika iz različitih populacija, snimljenih i pohranjenih različitim metodama.

Slučajevi uključuju reprezentativnu kolekciju svih važnih dijagnostičkih kategorija pigmentiranih lezija: aktiničke keratoze i intraepitelni karcinom / Bowenova bolest (akiec), bazocelularni karcinom (bcc), benigne keratoze (solarne lentigine / seboreične keratoze i keratoze slične lišaju planusu, bkl), dermatofibrom (df), melanom (mel), melanocitni nevusi (nv) i vaskularne lezije (angiomi, angioqueratomi, piogeni granulomi i hemoragije, vasc).

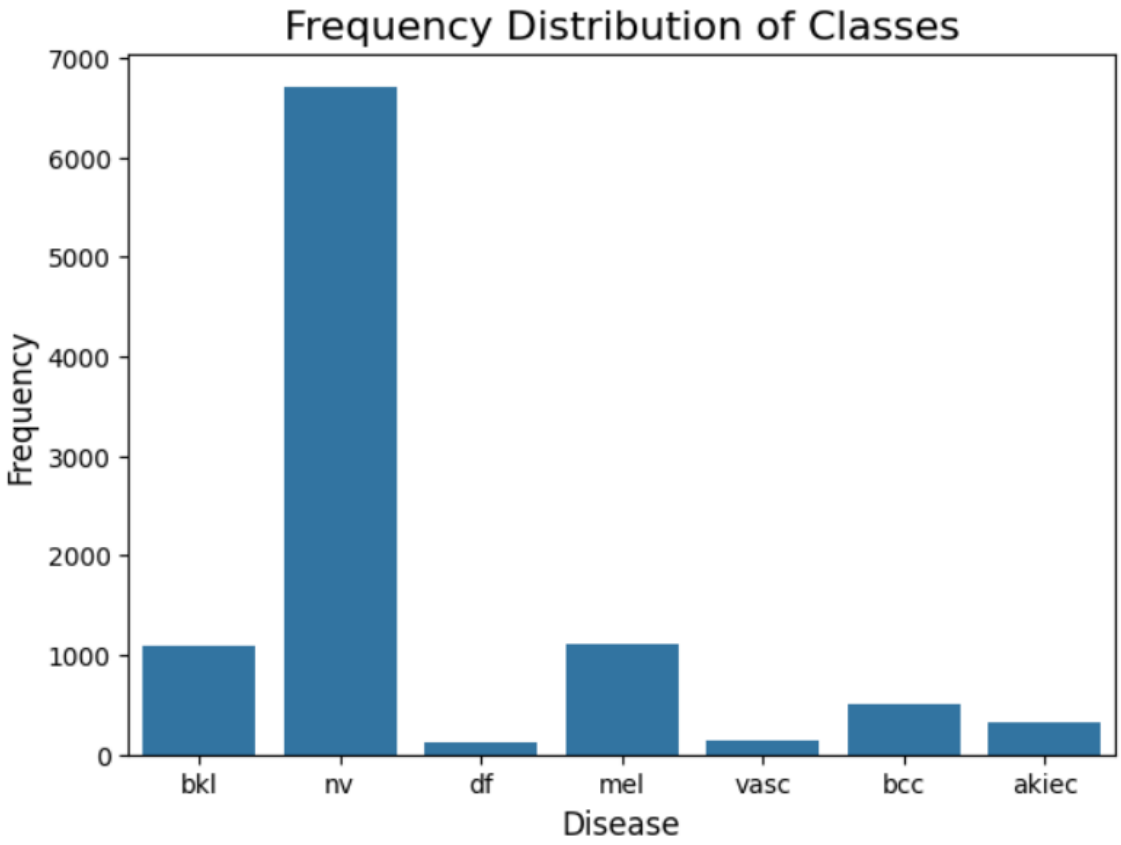
```
dataset_images_RGB = pd.read_csv("data/hmnist_28_28_RGB.csv")
print(dataset_images_RGB.head(3))
print('Slike: ', dataset_images_RGB.shape)
```

```
   pixel0000  pixel0001  pixel0002  pixel0003  pixel0004  pixel0005  \
0        192        153        193        195        155        192
1         25         14         30         68         48         75
2        192        138        153        200        145        163

   pixel0006  pixel0007  pixel0008  pixel0009  ...  pixel2343  pixel2344  \
0        197        154        185        202  ...        173        124
1        123         93        126        158  ...         60         39
2        201        142        160        206  ...        167        129

   pixel2345  pixel2346  pixel2347  pixel2348  pixel2349  pixel2350  \
0        138        183        147        166        185        154
1         55         25         14         28         25         14
2        143        159        124        142        136        104

   pixel2351  label
0        177      2
1         27      2
2        117      2

[3 rows x 2353 columns]
Slike:  (10015, 2353)
```

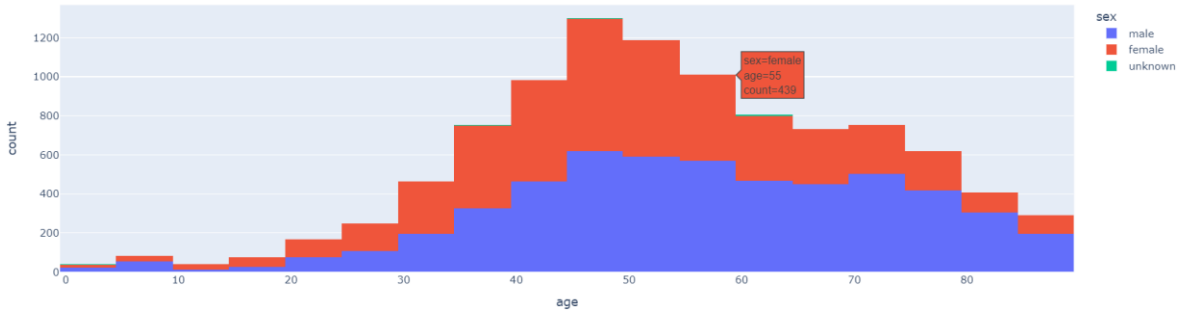
```
dataset_meta = pd.read_csv("data/HAM10000_metadata.csv")
print(dataset_meta.head(3))
print('Metapodaci: ', dataset_meta.shape)
```

```
     lesion_id      image_id   dx dx_type   age   sex localization
0  HAM_0000118  ISIC_0027419  bkl   histo  80.0  male        scalp
1  HAM_0000118  ISIC_0025030  bkl   histo  80.0  male        scalp
2  HAM_0002730  ISIC_0026769  bkl   histo  80.0  male        scalp
Metapodaci:  (10015, 7)
```

```
bar, ax = plt.subplots(figsize=(7, 5))
sns.countplot(x = 'dx', data = dataset_meta)
plt.xlabel('Disease', size=12)
plt.ylabel('Frequency', size=12)
plt.title('Frequency Distribution of Classes', size=16)
plt.show()
```




```
fig = px.histogram(data_frame=dataset_meta, x='age', color= 'sex')
fig.show()
```



4.2.2. CNN (Convolutional Neural Network)

```
num_classes = 7
batch_size = 128
epochs = 10

img_rows = 28
img_cols = 28
```

```
images = dataset_images_RGB.drop(['label'], axis=1)
labels = dataset_images_RGB['label']
```

```
oversample = RandomOverSampler()
images, labels = oversample.fit_resample(images, labels)
```

```
images = np.array(images)
images = images.reshape(-1, 28, 28, 3)
print('Shape of images: ', images.shape)
```

```
Shape of images:  (46935, 28, 28, 3)
```

```
x_train, x_test, y_train, y_test = train_test_split(images, labels,
random_state=1, test_size=0.20)

y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=(img_rows, img_cols, 3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.40))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        896

 max_pooling2d (MaxPooling2  (None, 13, 13, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 5, 5, 64)          0
 g2D)

 dropout (Dropout)           (None, 5, 5, 64)          0

 flatten (Flatten)           (None, 1600)              0

 dense (Dense)               (None, 128)               204928

 dropout_1 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 7)                 903

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        896

 max_pooling2d (MaxPooling2  (None, 13, 13, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 5, 5, 64)          0
 g2D)

 dropout (Dropout)           (None, 5, 5, 64)          0

 flatten (Flatten)           (None, 1600)              0

 dense (Dense)               (None, 128)               204928

 dropout_1 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 7)                 903

=================================================================
Total params: 225223 (879.78 KB)
Trainable params: 225223 (879.78 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
callback = tf.keras.callbacks.ModelCheckpoint(filepath='CNN/cnn-RGB.keras', monitor='val_acc', mode='max', verbose=1)
model.compile(loss=keras.losses.categorical_crossentropy,
optimizer='adam', metrics=['accuracy'])
```

```
history = model.fit(x_train, y_train, batch_size=batch_size,
epochs=epochs, validation_split=0.2, callbacks=[callback])
```

```
Epoch 1/10
235/235 [==============================] - ETA: 0s - loss: 2.7896 - accuracy: 0.3393
Epoch 1: saving model to CNN\cnn-RGB.keras
235/235 [==============================] - 10s 40ms/step - loss: 2.7896 - accuracy: 0.3393 - val_loss: 1.4271 - val_accuracy: 0.4442
Epoch 2/10
235/235 [==============================] - ETA: 0s - loss: 1.4210 - accuracy: 0.4320
Epoch 2: saving model to CNN\cnn-RGB.keras
235/235 [==============================] - 9s 40ms/step - loss: 1.4210 - accuracy: 0.4320 - val_loss: 1.2047 - val_accuracy: 0.5352
Epoch 3/10
235/235 [==============================] - ETA: 0s - loss: 1.2810 - accuracy: 0.4819
Epoch 3: saving model to CNN\cnn-RGB.keras
235/235 [==============================] - 10s 41ms/step - loss: 1.2810 - accuracy: 0.4819 - val_loss: 1.1361 - val_accuracy: 0.5722
Epoch 4/10
235/235 [==============================] - ETA: 0s - loss: 1.2138 - accuracy: 0.5134
Epoch 4: saving model to CNN\cnn-RGB.keras
235/235 [==============================] - 10s 43ms/step - loss: 1.2138 - accuracy: 0.5134 - val_loss: 1.0285 - val_accuracy: 0.5883
Epoch 5/10
235/235 [==============================] - ETA: 0s - loss: 1.1361 - accuracy: 0.5489
Epoch 5: saving model to CNN\cnn-RGB.keras
235/235 [==============================] - 10s 41ms/step - loss: 1.1361 - accuracy: 0.5489 - val_loss: 0.9605 - val_accuracy: 0.6406
Epoch 6/10
235/235 [==============================] - ETA: 0s - loss: 1.0383 - accuracy: 0.5948
Epoch 6: saving model to CNN\cnn-RGB.keras
235/235 [==============================] - 10s 41ms/step - loss: 1.0383 - accuracy: 0.5948 - val_loss: 0.8683 - val_accuracy: 0.6933
Epoch 7/10
235/235 [==============================] - ETA: 0s - loss: 0.9763 - accuracy: 0.6176
Epoch 7: saving model to CNN\cnn-RGB.keras
235/235 [==============================] - 10s 41ms/step - loss: 0.9763 - accuracy: 0.6176 - val_loss: 0.7642 - val_accuracy: 0.7142
Epoch 8/10
235/235 [==============================] - ETA: 0s - loss: 0.9349 - accuracy: 0.6405
Epoch 8: saving model to CNN\cnn-RGB.keras
235/235 [==============================] - 9s 40ms/step - loss: 0.9349 - accuracy: 0.6405 - val_loss: 0.7199 - val_accuracy: 0.7628
Epoch 9/10
235/235 [==============================] - ETA: 0s - loss: 0.8848 - accuracy: 0.6580
Epoch 9: saving model to CNN\cnn-RGB.keras
235/235 [==============================] - 10s 43ms/step - loss: 0.8848 - accuracy: 0.6580 - val_loss: 0.6555 - val_accuracy: 0.7668
Epoch 10/10
235/235 [==============================] - ETA: 0s - loss: 0.8086 - accuracy: 0.6900
Epoch 10: saving model to CNN\cnn-RGB.keras
235/235 [==============================] - 10s 42ms/step - loss: 0.8086 - accuracy: 0.6900 - val_loss: 0.5675 - val_accuracy: 0.8025
```

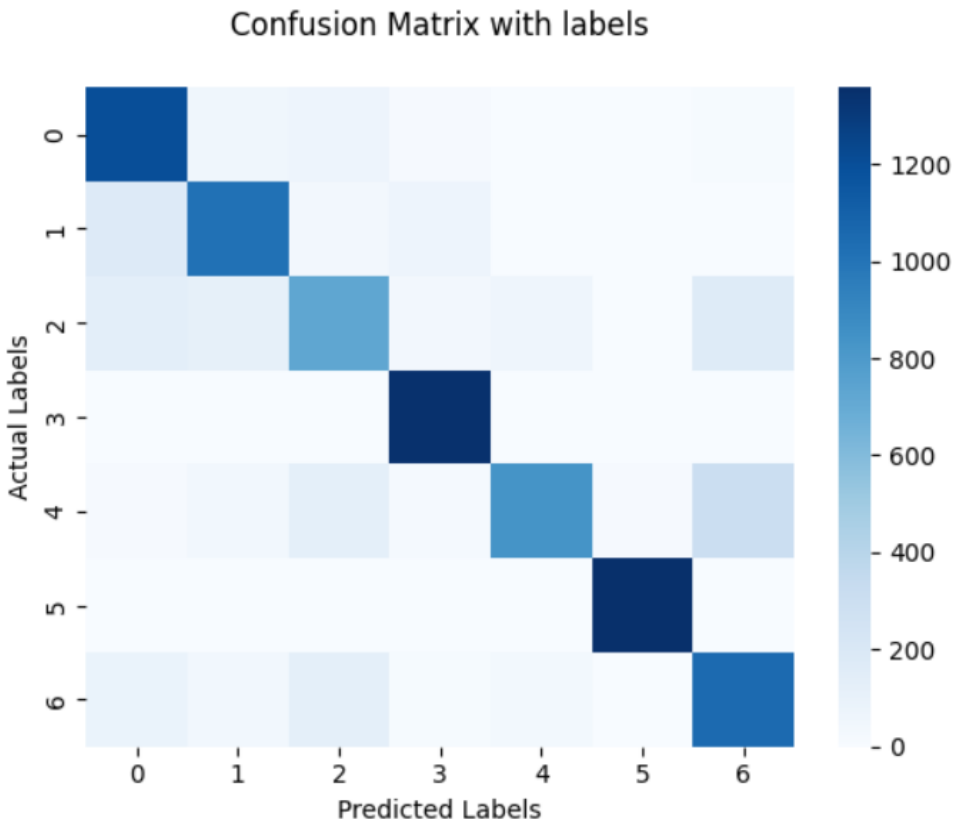
4.2.3. Evaluacija

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Summary: Loss over the test dataset: %.2f, Accuracy: %.2f' %(score[0], score[1]))
```

```
Summary: Loss over the test dataset: 0.58, Accuracy: 0.80
```

```
y_pred = model.predict(x_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis =1)
confusion_matrix = metrics.confusion_matrix(y_true=y_true,
y_pred=y_pred_classes )

ax = sns.heatmap(confusion_matrix, fmt='', cmap='Blues')
ax.set_title('Confusion Matrix with labels\n');
ax.set_xlabel('Predicted Labels')
ax.set_ylabel('Actual Labels')
plt.show()
```

```
294/294 [==============================] - 1s 4ms/step
```



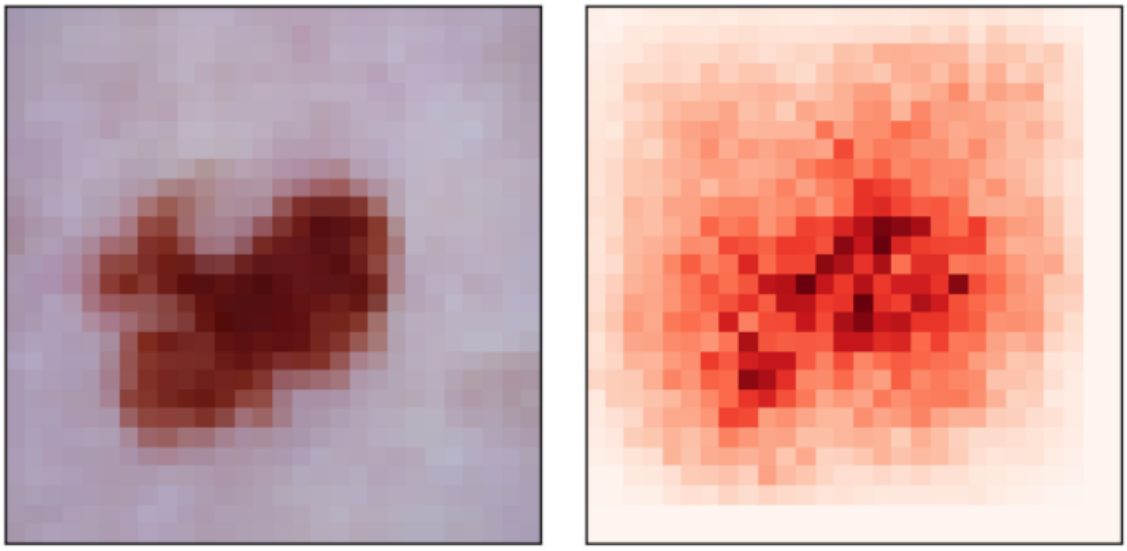
4.2.4. Saliency maps (Pixel Attribution)

```
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.preprocessing import image
from tensorflow.keras import backend as K
from tf_keras_vis.saliency import Saliency
from tf_keras_vis.utils import normalize
from vis.utils import utils

from tf_keras_vis.utils.scores import CategoricalScore

import random
```

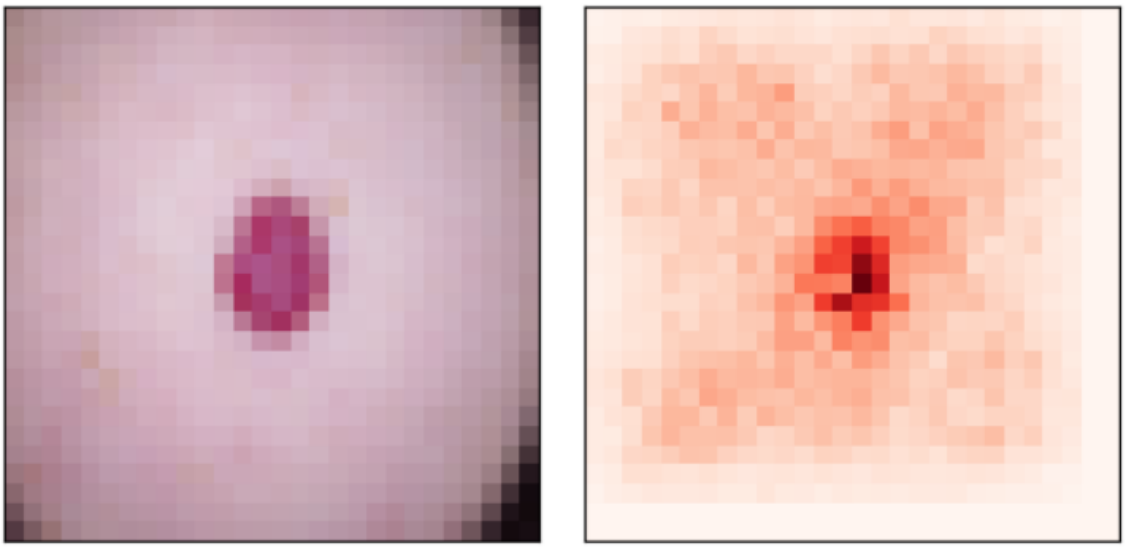
```
layer_idx = utils.find_layer_idx(model, model.layers[-1].name)
model.layers[-1].activation = tf.keras.activations.linear
model = utils.apply_modifications(model)
```

```python
def saliencyPlot(img_index = 0):
    x = x_test[img_index]
    x = x.reshape((1,) + x.shape)
    x = np.array(x, dtype=np.float32)

    prediction = model.predict(x)
    y_prediction = np.argmax(prediction)
    y_true = np.argmax([y_test[img_index]])
    print(f"Model output: {prediction}")
    print(f"Model prediction: {y_prediction}")
    print(f"True clasification: {y_true}")

    score = CategoricalScore([y_prediction])

    saliency = Saliency(model, clone=False)

    saliency_map = saliency(score, x, smooth_samples=20)
    saliency_map = normalize(saliency_map)

    subplot_args = {
        'nrows': 1,
        'ncols': 2,
        'figsize': (6, 3),
        'subplot_kw': {'xticks': [], 'yticks': []}
    }

    f, ax = plt.subplots(**subplot_args)
    ax[0].imshow(x_test[img_index])
    ax[1].imshow(saliency_map[0], cmap='Reds')
    plt.tight_layout()
    plt.show()

for i in range(5):
    saliencyPlot(random.randrange(0, len(x_test)))
```

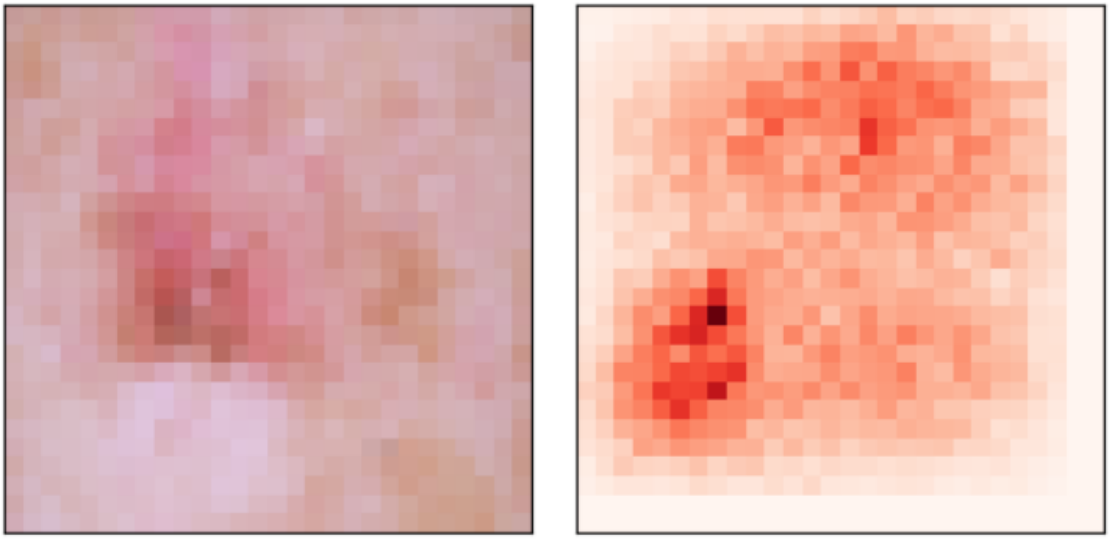
```
1/1 [==============================] - 0s 16ms/step
Model output: [[ -5.529092  -6.287302  13.48935  -20.948671  20.702822 -16.64197
   22.525131]]
Model prediction: 6
True clasification: 6
```



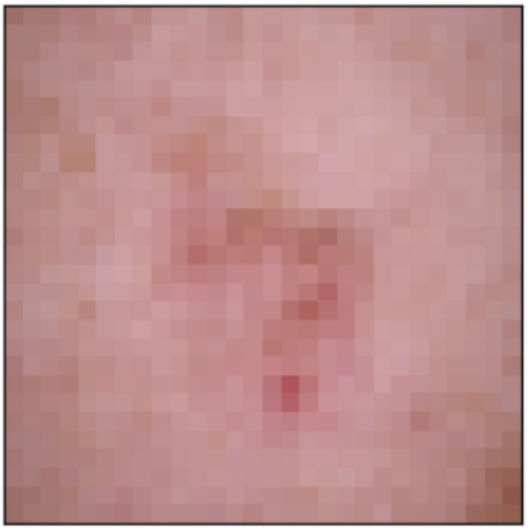
```
1/1 [==============================] - 0s 17ms/step
Model output: [[-14.7710285   2.7037752  -3.7271929  -6.8785734   1.9419974  13.436104
   -2.2827864]]
Model prediction: 5
True clasification: 5
```

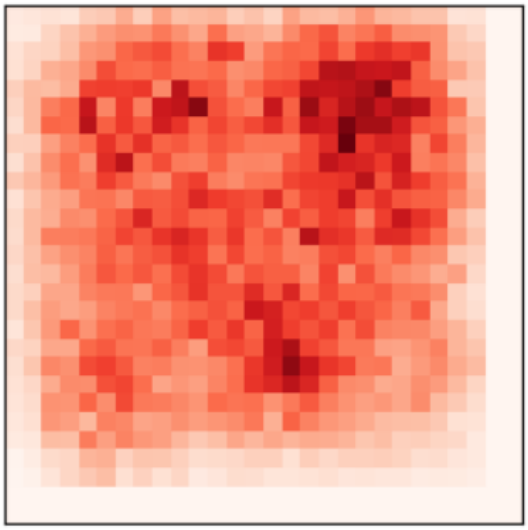


```
1/1 [==============================] - 0s 15ms/step
Model output: [[ 2.5476182  2.0607042  0.9720427 -1.2882347 -1.3925024 -2.9985223
   0.1737361]]
Model prediction: 0
True clasification: 0
```

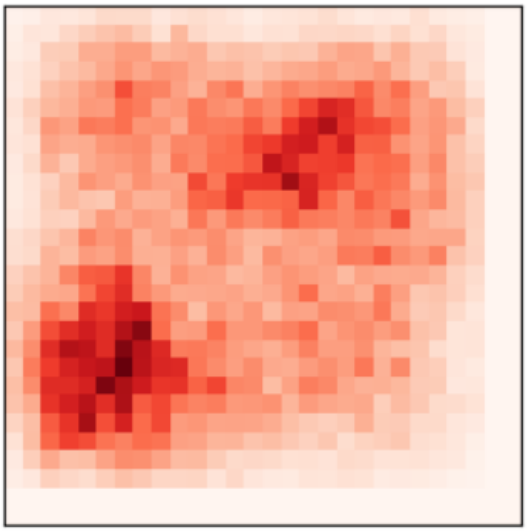


```
1/1 [==============================] - 0s 16ms/step
Model output: [[ 2.343579   1.5000801  0.9288622  1.1681467 -1.5176104 -4.371785
  -1.5361764]]
Model prediction: 0
True clasification: 0
```





```
1/1 [==============================] - 0s 17ms/step
Model output: [[ 2.1948698   0.42213526  3.3318202  -3.0356596  -0.38792053 -4.226078
   1.4361802 ]]
Model prediction: 2
True clasification: 2
```



Literatura

[1] https://www.sciencedirect.com/science/article/pii/S1566253523001148 (29. 10. 2024)

[2] Linardatos, P.; Papastefanopoulos, V.; Kotsiantis, S. Explainable AI: A Review ofMachine Learning Interpretability Methods. Entropy 2021, 23, 18 https://dx.doi.org/0.3390/e23010018

[3] https://www.geeksforgeeks.org/explainable-artificial-intelligencexai/ (29. 10. 2024)

[4] https://christophm.github.io/interpretable-ml-book/ (29. 10. 2024)

[5] https://towardsdatascience.com/explainable-machine-learning-9d1ca0547ae0 (29. 10. 2024)

[6] https://towardsdatascience.com/lime-explain-machine-learning-predictions-af8f18189bfe (29. 10. 2024)

[7] https://www.geeksforgeeks.org/introduction-to-explainable-aixai-using-lime/ (29. 10. 2024)

[8] https://www.kaggle.com/code/prashant111/explain-your-model-predictions-with-lime (29. 10. 2024)

[9] https://medium.com/data-reply-it-datatech/explainable-ai-shapley-values-and-lime-5f14d42147b3 (29. 10. 2024)

[10] https://www.kaggle.com/code/prashant111/explain-your-model-predictions-with-shapley-values?scriptVersionId=28578470 (29. 10. 2024)

[11] https://medium.com/@bijil.subhash/explainable-ai-saliency-maps-89098e230100 (30. 10. 2024)