Sistemi odlučivanja u medicini

Projektni zadatak

Tema: Breast Cancer

Katarina Petrović (2021/0068)

Anja Stojanović (2021/0167)

Avgust, 2024.

1. Analiza baze podataka i predprocesiranje

Prvi korak u izradi projektnog zadatka bio je analizirati bazu podataka *"04_breast_cancer_1_dataset.csv"*. Baza sadrži 10 kolona – od kojih se prva odnosi na to kojoj klasi pripada koji ispitanik, dok se ostalih 9 odnosi na različite parametre, odnosno atribute, koji su na neki način u konekciji sa odgovarajućom klasom.

Podaci koji su razmatrani u ovoj bazi podataka su:

- Broj godina – zadat u opsezima (npr. 20-29, 30-39) i pretvoren u srednju vrednost početka i kraja opsega
- Menopauza – tri karakteristične vrednosti (premeno, lt40 i ge40) koje govore o tome da li je osoba u premenopauzi ili u menopauzi u koju je ušla pre ili nakon četrdesete godine. Podaci su kodirani sa 0, 1 i 2.
- Veličina tumora – zadata u opsezima i problematično formatirana u bazi podataka (npr. opseg Oct-14 predstavlja 12-14). Podaci su kodirani brojevima od 0 do 10.
- Inv-Nodes (Involved lymph nodes) – broj limfnih čvorova koji su zahvaćeni kancerogenim ćelijama. Takođe je došlo do lošeg formatiranja koje je sređeno, a podaci su kodirani brojevima od 0 do 5. Takođe je postojao jedan outlier, za koji smo odlučile da ceo taj set podataka izbacimo iz razmatranja.
- Node-Caps - zavisno od toga da li su kancerogene ćelije prodrle kroz omotač limfnog čvora vraća vrednosti "no" ili "yes" koje su kodirane sa 0 i 1. Takođe je postojalo 8 ispitanika sa neregularnim vrednostima, koje smo izbacile iz razmatranja.
- Stepen malignosti – ukazuje na agresivnost tumora i ima vrednosti 1, 2 ili 3.
- Dojka – da li je kancerom zaražena leva ili desna dojka
- Deo dojke – izdeljeno na 5 celina (gornji levi, gornji desni, centar, donji levi i donji desni) koje su kodirane vrednostima od 1 do 5
- Zračenje – da li je osoba prethodno zračena. Podaci su dati sa "no" ili "yes" i kodirani sus a 0 ili 1.

Kao izlazni podatak, razmatra se ponovno pojavljivanje raka dojke. Postoje dve klase u koje se svrstavaju ispitanici – no-reccurence-events i reccurence-events koje su kodirane sa 0 i 1.

U ovoj bazi podataka nalazilo se 286 ispitanika.

Sledeći kod predstavlja postupak predprocesiranja kojim smo bazu podataka učinile pogodnom za dalje ispitivanje:

In [41]:
```
# print(data['Class'].unique())
data.Class = data['Class'].replace('no-recurrence-events', 0)
data.Class = data['Class'].replace('recurrence-events', 1)
#pd.set_option('future.no_silent_downcasting', True)

#print(data['Age'].unique())
data.Age = data['Age'].replace('20-29', 25)
data.Age = data['Age'].replace('30-39', 35)
data.Age = data['Age'].replace('40-49', 45)
data.Age = data['Age'].replace('50-59', 55)
data.Age = data['Age'].replace('60-69', 65)
data.Age = data['Age'].replace('70-79', 75)

#print(data['Menopause'].unique())
data.Menopause = data['Menopause'].replace('premeno', 0)
data.Menopause = data['Menopause'].replace('lt40', 1)
data.Menopause = data['Menopause'].replace('ge40', 2)

#print(data['Tumor-size'].unique())
data['Tumor-size'] = data['Tumor-size'].replace('0-4', 0)
data['Tumor-size'] = data['Tumor-size'].replace('05-Sep', 1)
data['Tumor-size'] = data['Tumor-size'].replace('Oct-14', 2)
data['Tumor-size'] = data['Tumor-size'].replace('15-19', 3)
data['Tumor-size'] = data['Tumor-size'].replace('20-24', 4)
data['Tumor-size'] = data['Tumor-size'].replace('25-29', 5)
data['Tumor-size'] = data['Tumor-size'].replace('30-34', 6)
data['Tumor-size'] = data['Tumor-size'].replace('35-39', 7)
data['Tumor-size']= data['Tumor-size'].replace('40-44', 8)
data['Tumor-size'] = data['Tumor-size'].replace('45-49', 9)
data['Tumor-size'] = data['Tumor-size'].replace('50-54', 10)

#print(data['Inv-nodes'].unique())
#print(data['Inv-nodes'].value_counts().get('24-26'))
data['Inv-nodes'] = data['Inv-nodes'].replace('0-2', 0)
data['Inv-nodes'] = data['Inv-nodes'].replace('03-May', 1)
data['Inv-nodes'] = data['Inv-nodes'].replace('06-Aug', 2)
data['Inv-nodes'] = data['Inv-nodes'].replace('09-Nov', 3)
data['Inv-nodes'] = data['Inv-nodes'].replace('Dec-14', 4)
data['Inv-nodes'] = data['Inv-nodes'].replace('15-17', 5)
data = data.drop(data[data['Inv-nodes'] == '24-26'].index)
data['Inv-nodes'] = data['Inv-nodes'].astype('int64')

#print(data['Node-caps'].unique())
#print(data['Node-caps'].value_counts().get('?'))
data = data.drop(data[data['Node-caps'] == '?'].index)
data['Node-caps'] = data['Node-caps'].replace('no', 0)
data['Node-caps'] = data['Node-caps'].replace('yes', 1)

#print(data['Deg-malig'].unique())

#print(data['Breast'].unique())
data.Breast = data['Breast'].replace('left', 0)
data.Breast = data['Breast'].replace('right', 1)
```

```
#print(data['Breast'].unique())
data.Breast = data['Breast'].replace('left', 0)
data.Breast = data['Breast'].replace('right', 1)

#print(data['Breast-quad'].unique())
#print(data['Breast-quad'].value_counts().get('?'))
data = data.drop(data[data['Breast-quad'] == '?'].index)
data['Breast-quad'] = data['Breast-quad'].replace('left_low', 1)
data['Breast-quad'] = data['Breast-quad'].replace('left_up', 2)
data['Breast-quad'] = data['Breast-quad'].replace('right_low', 3)
data['Breast-quad'] = data['Breast-quad'].replace('right_up', 4)
data['Breast-quad'] = data['Breast-quad'].replace('central', 5)

#print(data['irradiat'].unique())
data.irradiat = data['irradiat'].replace('no', 0)
data.irradiat= data['irradiat'].replace('yes', 1)

#print(data.to_string())
data.info(verbose=True)
data.describe().T

plt.figure()
data.hist(bins = 10, figsize=(12,9))
plt.show()
```

2. Informaciona dobit, koeficijent korelacije

Prvi korak pri rešavanju druge tačke projektnog zadatka je računanje informacione dobiti - IG-a svakog obeležja, što je u python kodu implementirano na sledeći način:

In [42]:
```
def calculateInfoD(col):
    un = np.unique(col)
    infoD = 0
    for u in un:
        p = sum(col == u)/len(col)
        infoD -= p*np.log2(p)
    return infoD

klasa = data.iloc[:,0]

infoD = calculateInfoD(klasa)
print('Info(D) = ' + str(infoD))
```

```
Info(D) = 0.8685339602652349
```

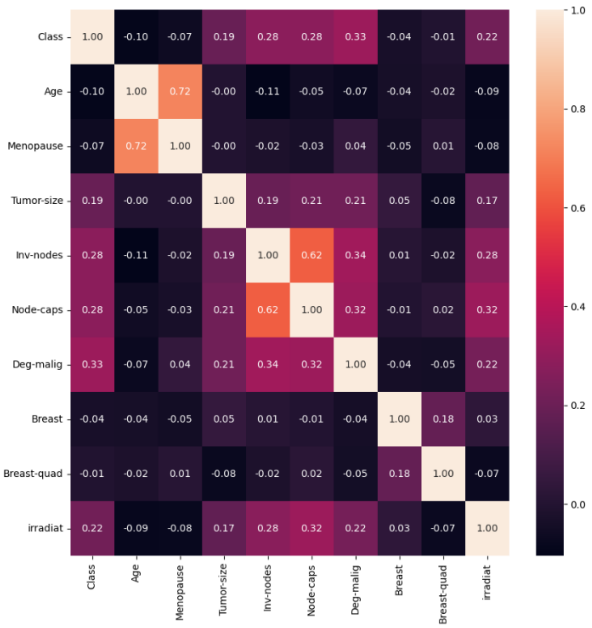
In [43]:
```
IG = []
for ob in range(1,10):
    kol = data.iloc[:, ob]
    f = np.unique(kol)

    infoDA = 0
    for i in f:
        temp = klasa[kol == i]

        infoDi = calculateInfoD(temp)
        Di = sum(kol == i)
        D = len(kol)

        infoDA += Di*infoDi/D

    IG.append([data.columns[ob], infoD - infoDA])

IGdf = pd.DataFrame(IG, columns=['obelezje', 'IG'])
IGsorted = IGdf.sort_values(by=['IG'], ascending=False, ignore_index=True)
IGsorted
```

Rezultati su sledeći:

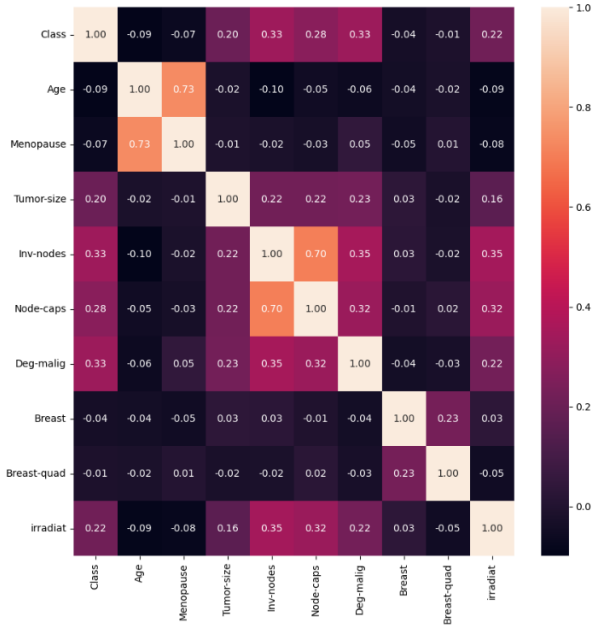
	obelezje	IG
0	Deg-malig	0.085685
1	Inv-nodes	0.076270
2	Tumor-size	0.061806
3	Node-caps	0.052764
4	irradiat	0.032220
5	Age	0.020600
6	Menopause	0.012078
7	Breast-quad	0.008378
8	Breast	0.000944

Kao što se može primetiti, informacione dobiti svih atributa su prilično male, ali je nama od većeg značaja da posmatramo koja su to najinformativnija, a koja najmanje informativna obeležja. Dva najinformativnija obeležja su DegMalig i InvNodes, što se i moglo očekivati na osnovu njihovog "fizičkog" značenja u ovom eksperimentu. Takođe uočavamo da obeležja Breast i BreastQuad nisu ni od kakvog informacionog značaja, što je u skladu sa početnom pretpostavkom da oni ne utiču na ponovno pojavljivanje raka dojke.

Korelacija podataka iz baze je ispitivana pomoću Pearson-ovog i Spearman-ovog koeficijenta korelacije. Iako su rezultati koji će biti prikazani u nastavku prilično slični, mi smo se opredelile za korišćenje Spearman-ovog koeficijenta korelacije zbog njegove opštosti.



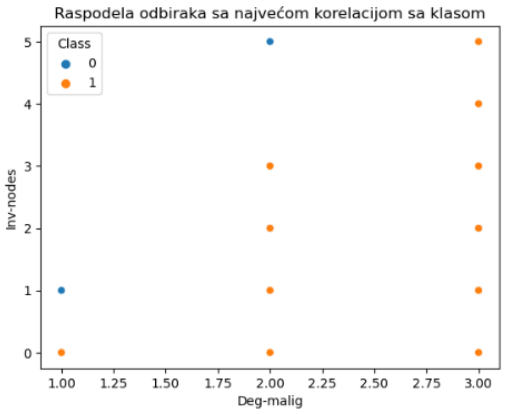
Pearson-ov koeficijent korelacije



Spearman-ov koeficijent korelacije

Na osnovu ovih podataka moguce je uvideti i odnose između pojedinih obeležja, a ne samo odnose obeležja sa klasom. Zaključujemo da su obeležja Age i Menopause u višoj korelaciji, kao i NodeCaps i InvNodes.

Takođe, ispitale smo raspodelu odbiraka onih obeležja koja su najinformativnija (tj. najviše korelisana sa klasom) i dobile sledeće rezultate:

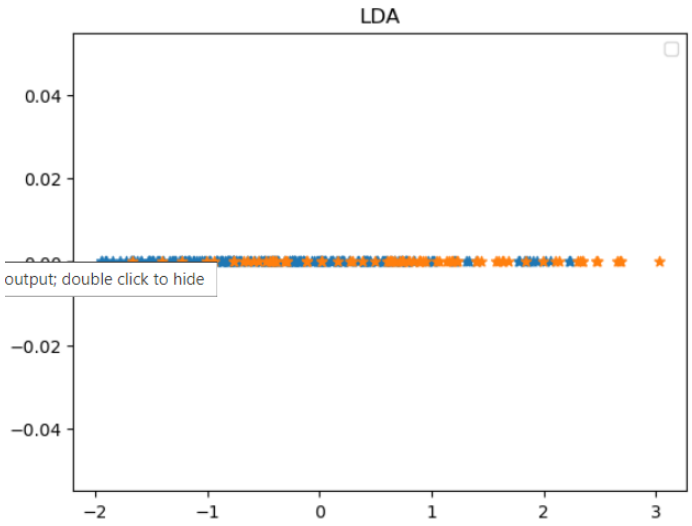


Uočavamo da ispitanici sa većim brojem zahvaćenih limfnih čvorova i manjim stepenom malignosti pripradaju klasi 0, odnosno onih kod kojih se rak nije vraćao, dok ispitanici sa velikim stepenom malignosti i manjim brojem zahvaćenih limfnih čvorova pripadaju klasi 1 – onih kojima se rak dojke vraćao.

3. LDA metoda za redukciju dimenzija

LDA metoda za redukciju dimenzija je zasnovana na izračunavanju vrednosti 3 matrice – unutarklasnog rasejanja Sw, međuklasnog rasejanja Sb i miksovane Sm.

U našem slučaju, s obzirom da je broj klasa 2, moguce je smanjiti dimenzionalnost problema najmanje na jednu dimenziju. Koristile smo ugrađenu LDA funkciju na osnovu koje smo dobile sledeći grafik nakon redukcije na jednu dimenziju:



Na grafiku vidimo da podaci nisu linearno ni kvadratno separabilni što bi uzrokovalo probleme u klasifikaciji parametarskim klasifikatorom.

Takođe, indeks informativnosti na osnovu procenta varijanse je u potpunosti zadržan i iznosi 100% nakon izvršene redukcije.

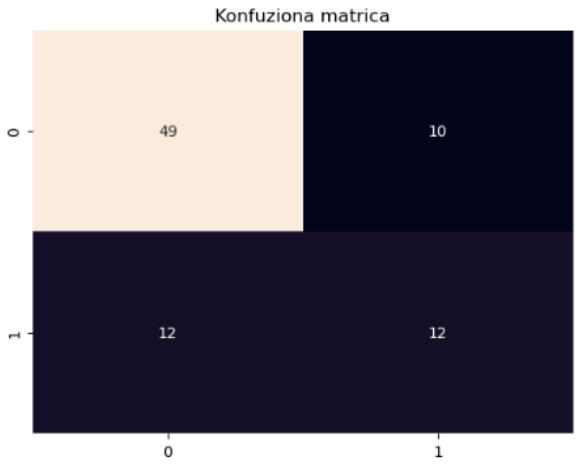
4. Projektovanje parametarskog klasifikatora -> Projektovanje klasifikatora na bazi testiranja hipoteza

Redukcija dimenzija iz tačke 3 nije dala zadovoljavajuću separabilnost klasa, tako da umesto parametarskog klasifikatora projektujemo klasifikator na bazi testiranja hipoteza. U nastavku je kod u kom se prvenstveno određuju matematičko očekivanje i varijansa najinformativnijih podataka (InvNodes i DegMalig) na osnovu trening skupa koji iznosi 70% ukupnih podataka, a onda se na osnovu njih formiraju funkcije gustine verovatnoće. Za klasifikaciju koristimo Bajesov test minimalne verovatnoće greške.

In [50]:
```
data_new = data[['Deg-malig', 'Inv-nodes', 'Class']]
X_new = data_new.iloc[:,:-1]
y_new = data_new.iloc[:, -1]

X_new_norm = X_new - np.mean(X_new, axis=0)
X_new_norm /= np.std(X_new, axis=0)

X1_new = X_new_norm.loc[y_new == 0, :]
X2_new = X_new_norm.loc[y_new == 1, :]

N1 = X1_new.shape[0]
N2 = X2_new.shape[0]

N1trening = int(0.7*N1)
X1_trening = X1_new.iloc[:N1trening, :]
X1_test = X1_new.iloc[N1trening:, :]

N2trening = int(0.7*N2)
X2_trening = X2_new.iloc[:N2trening, :]
X2_test = X2_new.iloc[N2trening:, :]

M1p = np.mean(X1_trening, axis=0)
S1p = np.cov(X1_trening.T)
#print('Matematicko ocekivanje K1: ' + str(M1p))
#print('Kovarijaciona matrica K1: \n' + str(S1p))

M2p = np.mean(X2_trening, axis=0)
S2p = np.cov(X2_trening.T)
#print('Matematicko ocekivanje K2: ' + str(M2p))
#print('Kovarijaciona matrica K2: \n' + str(S2p))

def izracunaj_fgv(x, m, s):
    det = np.linalg.det(s)
    inv = np.linalg.inv(s)
    x_mu = x - m

    fgv_const = 1/np.sqrt(2*np.pi*det)
    fgv_rest = np.exp(-0.5*x_mu.T@inv@x_mu)
    return fgv_const*fgv_rest

p1 = N1trening / (N1trening + N2trening)
p2 = N2trening / (N1trening + N2trening)
T = np.log(p1/p2)

odluka1 = np.zeros((N1-N1trening, 1))
for i in range(N1-N1trening):
    x1 = X1_test.iloc[i, :]
    f1 = izracunaj_fgv(x1, M1p, S1p)
    f2 = izracunaj_fgv(x1, M2p, S2p)
    h1 = -np.log(f1) + np.log(f2)
    if h1 < T:
        odluka1[i] = 0
    else:
        odluka1[i] = 1
```

```
odluka2 = np.zeros((N2-N2trening, 1))
for i in range(N2-N2trening):
    x2 = X2_test.iloc[i, :]
    f1 = izracunaj_fgv(x2, M1p, S1p)
    f2 = izracunaj_fgv(x2, M2p, S2p)
    h2 = -np.log(f1) + np.log(f2)
    if h2 < T:
        odluka2[i] = 0
    else:
        odluka2[i] = 1

odluka = np.append(odluka1, odluka2, axis=0)
Xtest = np.append(X1_test, X2_test, axis=0)
Ytest = np.append(np.zeros((N1-N1trening, 1)), np.ones((N2-N2trening, 1)))

from sklearn.metrics import confusion_matrix
conf_mat = confusion_matrix(Ytest, odluka)
plt.figure()
sns.heatmap(conf_mat, annot=True, fmt='g', cbar=False)
plt.show()

tacnost = 100*np.trace(conf_mat)/np.sum(conf_mat)
print(tacnost)
```

Rezultati testa su sledeći:



Tačnost klasifikatora je 73.5%.

5. Neparametarska klasifikacija

U ovom delu razmatrrale smo KNN neparametarski klasifikator. Ovaj klasifikator razmatra svoje najbliže susede (u Euklidskom smislu) i odbirak pridružuje onoj klasi čijih pripadnika ima više među k najbližih suseda.

Kod koji implementira KNN klasifikaciju za k=10 prikazan je na sledećoj slici (skup podataka je podeljen na 30% testirajući skup, 70% obučavajući skup):

In [51]:
```
X = data.drop('Class', axis=1).values
Y = data['Class'].values
```

In [52]:
```
from sklearn.model_selection import train_test_split
Xtrening, Xtest, Ytrening, Ytest = train_test_split(X, Y, train_size=0.7, random_state=42, stratify=Y)
```

In [53]:
```
from sklearn.metrics import accuracy_score

k = 10
pred = np.zeros((Xtest.shape[0],))
for i in range(Xtest.shape[0]):
    tren_test = Xtest[i, :]
    dist = np.zeros((Xtrening.shape[0],))
    for j in range(Xtrening.shape[0]):
        tren_trening = Xtrening[j, :]
        dist[j] = np.sqrt(np.sum((tren_test - tren_trening)**2))

    idx = np.argsort(dist)[:k]
    najblizi_klase = Ytrening[idx]

    val, cnt = np.unique(najblizi_klase, return_counts=True)

    pred[i] = val[np.argmax(cnt)]

acc = accuracy_score(Ytest, pred)
print('Tacnost klasifikacije je : ' + str(acc*100) + '%.')
```

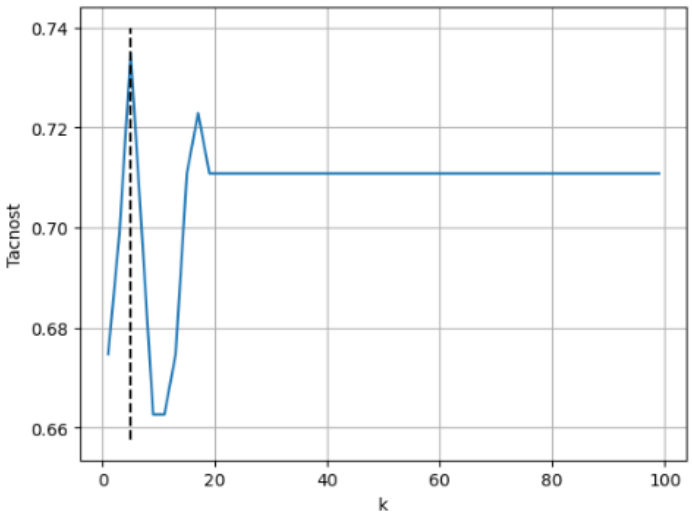
```
Tacnost klasifikacije je : 67.46987951807229%.
```

Prilikom ispitivanja za koju vrednost k se dostiže maksimalna tačnost modela, implementiran je sledeći kod koji koristi ugrađenu KNN funkciju:

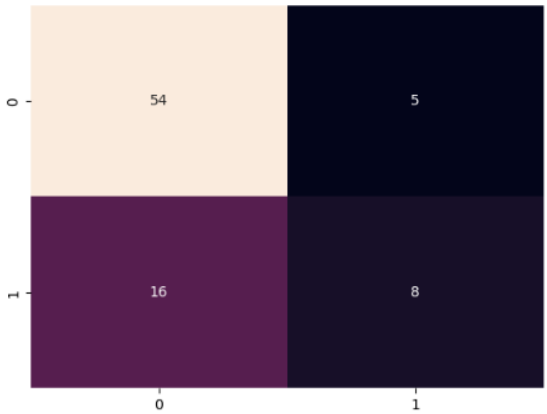
In [55]:
```
from sklearn.neighbors import KNeighborsClassifier as KNN
from sklearn.metrics import accuracy_score

neighbours = np.arange(1, 100, 2)
acc = np.zeros((neighbours.size))
for i in range(neighbours.size):
    knn_model = KNN(n_neighbors=neighbours[i])
    knn_model.fit(Xtrening, Ytrening)
    y_pred_knn = knn_model.predict(Xtest)
    acc[i] = accuracy_score(Ytest, y_pred_knn)
```

In [56]:
```
bestAcc = np.max(acc)
bestK = neighbours[np.argmax(acc)]

plt.figure()
plt.plot(neighbours, acc)
plt.plot([bestK, bestK], [np.min(acc)-0.005, np.max(acc)+0.005], 'k--')
plt.grid()
plt.show()

print('Najbolja tacnost se dobija za K = ' + str(bestK) + ' i iznosi ' + str(bestAcc*100) + '%.')
```

Grafik tačnosti u zavisnosti od broja suseda prikazan je na sledećoj slici:



Maksimalna tačnost KNN klasifikatora dobijena je za k=5 i iznosi 73.5%.



Konfuziona matrica KNN klasifikatora za optimalan broj suseda

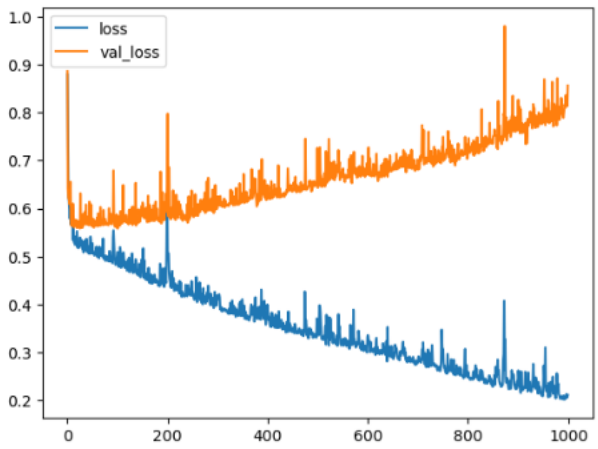
6. Neuralne mreže

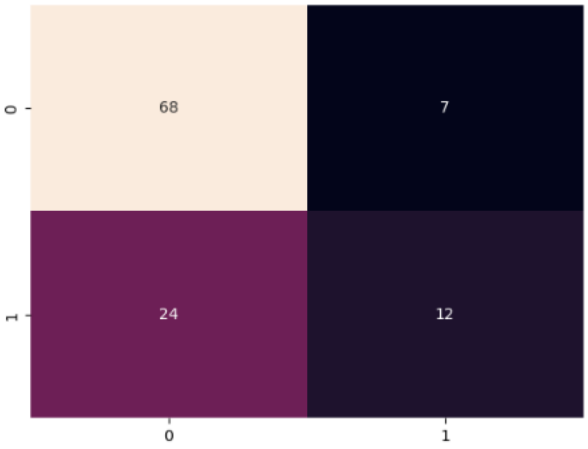
U ovom delu, zadatak je bio treniranje i klasifikacija podataka pomoću različitih struktura neuralnih mreža. Kako bismo došli do optimalne strukture, prvo ćemo pokazati loše izbore za strukturu neuralne mreže (previše kompleksna neuralna mreža – preobučavanje ili nedovoljno kompleksna neuralna mreža).

Prilikom formiranja neuralne mreže, korišćena je biblioteka keras. Za sve skrivene slojeve aktivaciona funkcija je ReLU (rectified linear unit), dok je za izlazni sloj aktivaciona funkcija unipolarni sigmoid.

Neuralna mreža sa jednim skrivenim slojem sa 350 neurona:

U ovom slučaju došlo je do preobučavanja neuralne mreže što se ogleda u tome da je mreža naviknuta na trening skup podataka i da greska pri njihovoj klasifikaciji teži 0, dok greska pri klasifikaciji novih (test podataka) raste tokom epoha. U nastavku je prikazana konfuziona matrica i grafik performanse neuralne mreže sa ovakvom strukturom.

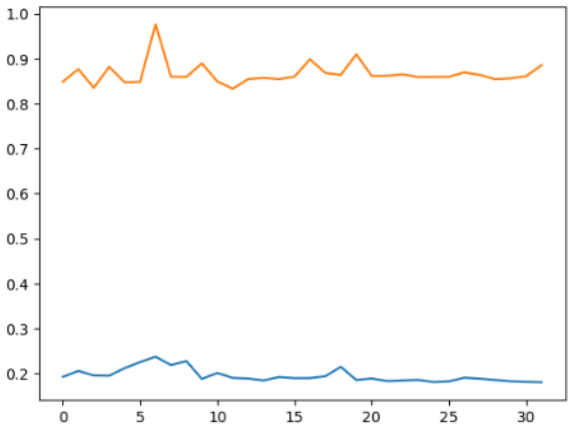


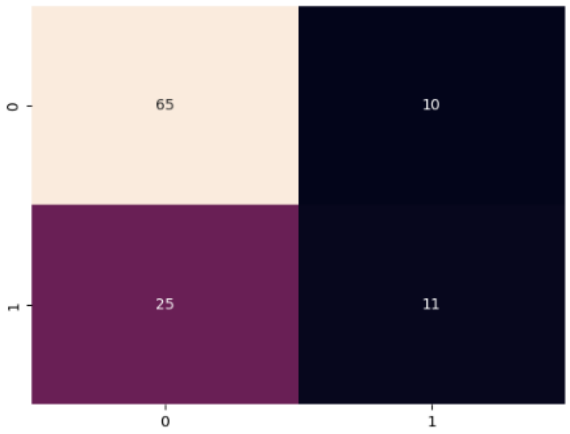


Tačnost mreže na trening skupu je 90%, a na test skupu 72%.

Problem preobučavanja rešen je dvema metodama – ranim zaustavljanjem i regularizacijom.

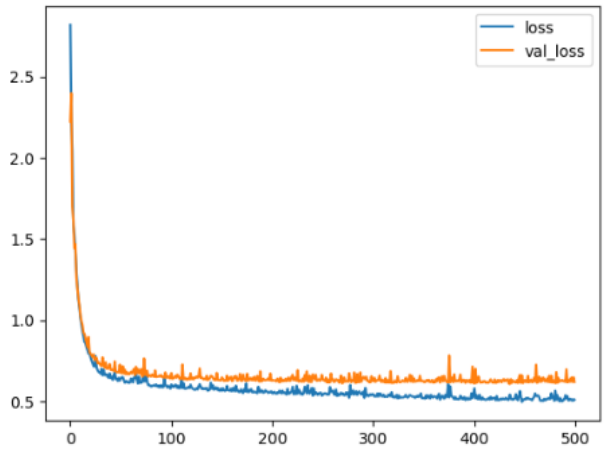
Rano zaustavljanje podrazumeva da se obučavanje mreže zaustavlja onog trenutka kada greska na validacionom skupu krene da raste. U našem slučaju to se dešava na samom pocetku treniranja mreže, pa se metod zaustavlja nakon 30-ak epoha.

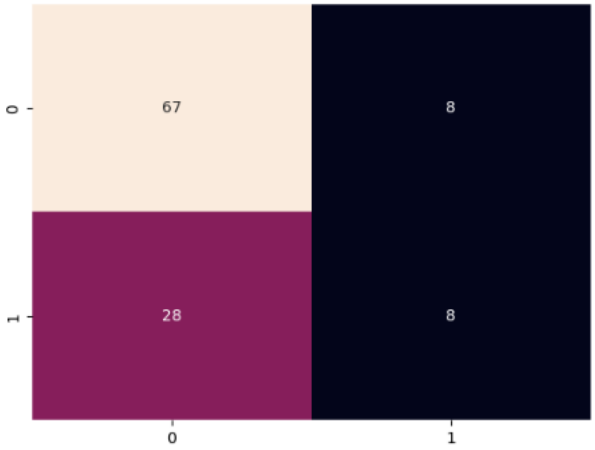




Tačnost nakon ranog zaustavljanja je: 93% na trening skupu, 68% na test skupu.

Regularizacija je metoda koja kažnjava postojanje velikih težina u neuralnoj mreži sa ciljem da male promene u ulaznim podacima ne dovode do velikih razlika u odlučivanju. U našem slučaju, koristili smo L2 regularizaciju sa parametrom Λ=0.08. U nastavku su dobijeni rezultati.



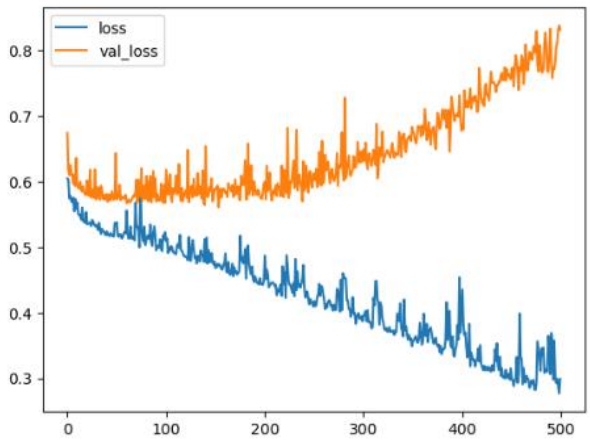


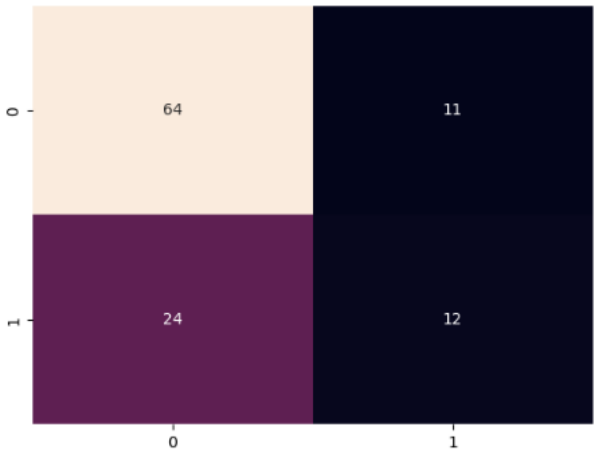
Tačnost nakon regularizacije je: 79% na trening skupu, 66% na test skupu.

Zaključujemo da je preobučavanje u nekoj meri regulisano iako je baza podataka generalno malo informativna.

Neuralna mreža sa tri skrivena sloja sa 100, 60 i 20 neurona:

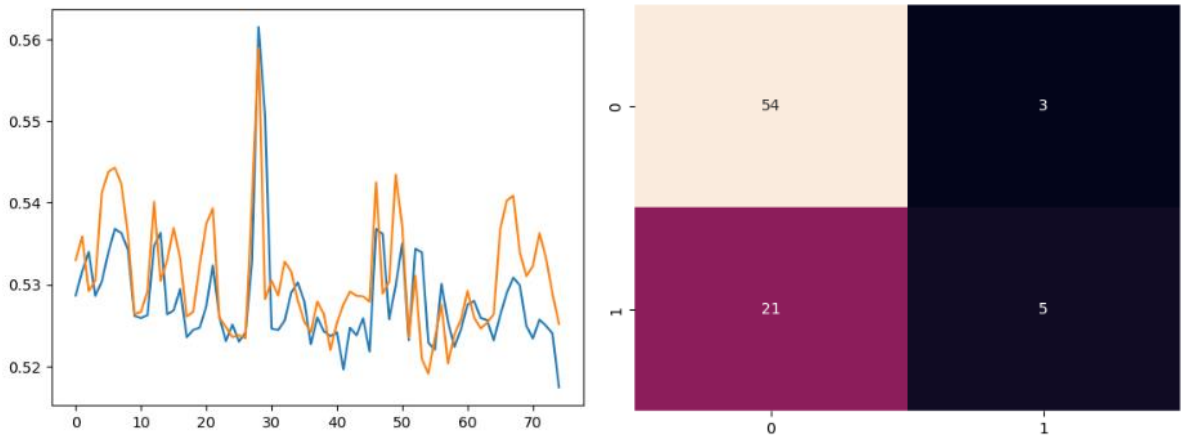
Kao i u prethodnom slučaju, došlo je do preobučavanja mreže jer je struktura mreže previse kompleksna za zadati problem. Prikazani su podaci koji ukazuju na preobučavanje:





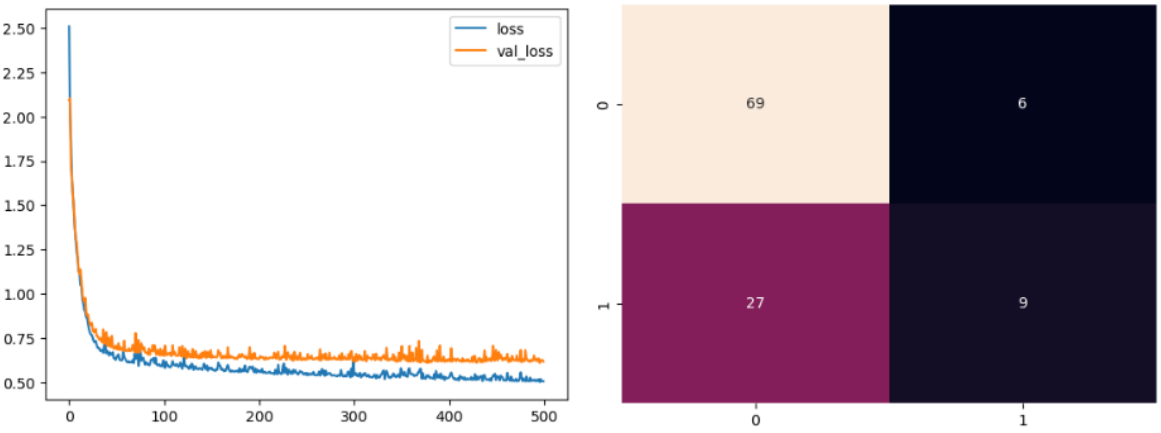
Tačnost ove mreže je: 89% na trening skupu, 68% na test skupu.

Kako bi sprečili preobučavanje, ponovo primenjujemo rano zaustavljanje i regularizaciju. Rezultati nakon ranog zaustavljanja su:



Tačnost nakon ranog zaustavljanja je: 88% na trening skupu, 67% na test skupu.

Rezultati nakon regularizacije su:

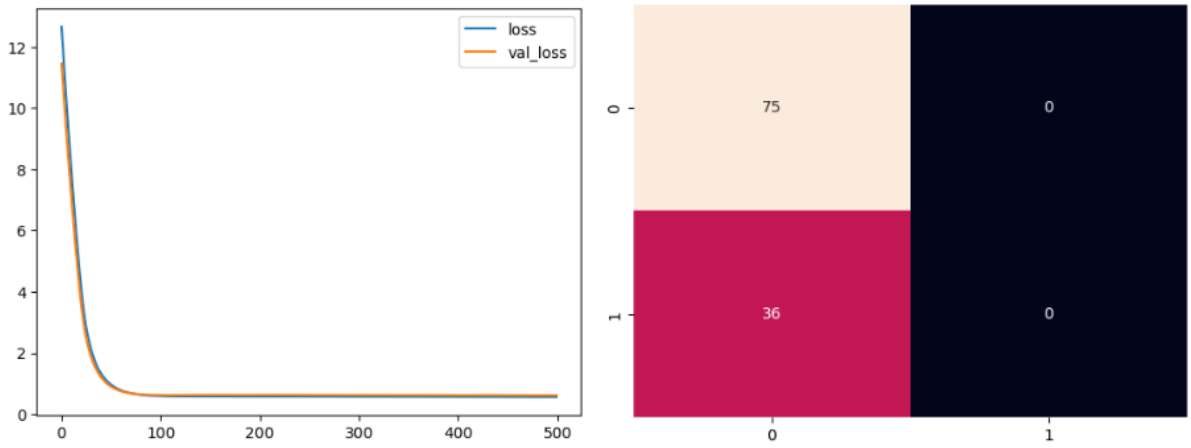


Tačnost nakon regularizacije je: 80% na trening skupu, 68% na test skupu.

Nakon ovih ispitivanja, vidimo da regularizacija i rano zaustavljanje rešavaju problem preobučavanja. Suštinski, iako mreža ima kompleksnu strukturu, modelu je na ovaj način zabranjeno da stvara težine koje dodatno usložnjavaju problem. Na ovaj način, granica klasifikacije ostaje jednostavnija i otpornija na nove ulazne podatke.

Neuralna mreža sa nedovoljnim brojem neurona

Kao kontrast prethodnim slučajevima, napravile smo neuralnu mrežu sa jednim skrivenim slojem sa 5 neurona koja nije dovoljno kompleksna da klasifikuje podatke iz date baze podataka. Iako se performanse trening i test skupa ne razlikuju, tačnost ove mreže znatno je manja u odnosu na prethodne slučajeve.



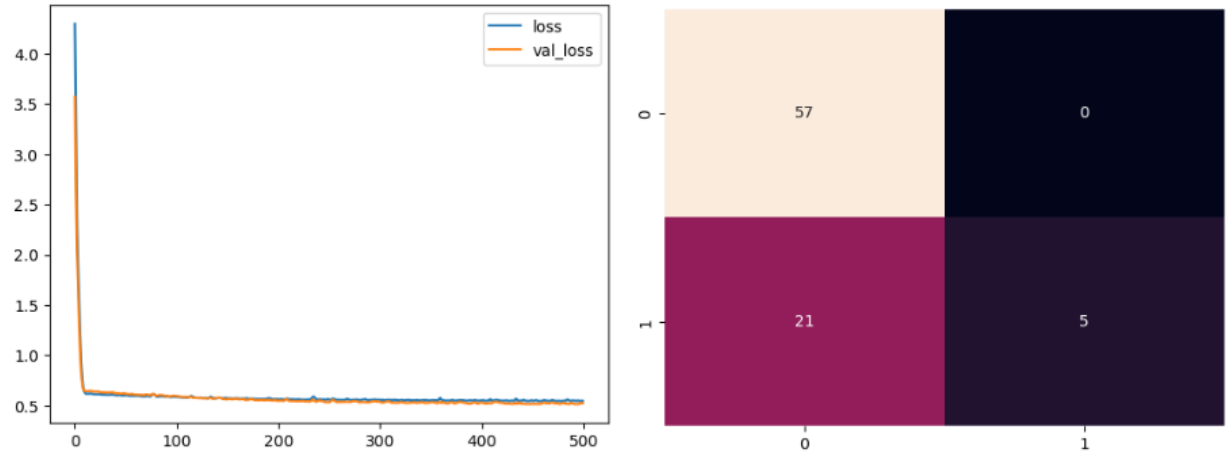
Tačnost mreže je: 73% za trening skup, 67% za test skup.

Optimalna struktura neuralne mreže

Nakon testiranja raznih vrednosti za broj skrivenih slojeva i broj neurona po sloju, došle smo do krajnjeg rezulata da je optimalna struktura neuralne mreže za klasifikaciju podataka iz date baze sledeća:

1. Skriveni sloj sa 14 neurona
2. Skriveni sloj sa 6 neurona
3. Izlazni sloj

Rezultati dobijeni obučavanjem i klasifikacijom podataka na ovoj strukturi mreže su sledeći:



Tačnost strukture je: 72% na trening skupu, 74% na test skupu.

Možemo uočiti da se performanse ove neuralne mreže ne razlikuju preterano na trening i test skupu (u pogledu gubitka) i da daju zadovoljavajuću tačnost.

Za data ispitivanja, korišćeni su sledeći programski kodovi sa manjim izmenama:

```
model = Sequential()
model.add(Dense(100,activation='relu',input_dim=X.shape[1],))
model.add(Dense(60,activation='relu',input_dim=X.shape[1],))
model.add(Dense(20,activation='relu',input_dim=X.shape[1],))
model.add(Dense(1,activation='sigmoid'))

model.compile('adam',loss='binary_crossentropy', metrics=['accuracy'])
model.summary()


max_epoha = 500
history = model.fit(x=Xtrening, y=Dtrening,epochs=max_epoha, batch_size=32, shuffle=32,verbose=0, validation_data=(Xtest,Dtest))

Ypred = model.predict(Xtest)
Ypred = np.round(Ypred)
conf_mat = confusion_matrix(Dtest,Ypred)

plt.figure()
sns.heatmap(conf_mat,annot=True,fmt='g',cbar=False)
plt.show()

plt.figure()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['loss', 'val_loss'])

YpredTrening = model.predict(Xtrening,verbose=0)
YpredTrening = np.round(YpredTrening)
YpredTest = model.predict(Xtest,verbose=0)
YpredTest = np.round(YpredTest)

accTrening = accuracy_score(Dtrening,YpredTrening)
print(accTrening*100)
accTest = accuracy_score(Dtest,YpredTest)
print(accTest*100)
```

In [30]:
```
from keras.callbacks import EarlyStopping

es = EarlyStopping(monitor='val_loss', mode='min', patience=20, verbose=1)

history = model.fit(x=Xtrening, y=Dtrening,
                    epochs=500,
                    batch_size=32,
                    validation_data=(Xtest, Dtest),
                    shuffle=True,
                    callbacks=[es],
                    verbose=0,
                   )

plt.figure()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.show()

Ypred = model.predict(Xtest)
Ypred = np.round(Ypred)
conf_mat = confusion_matrix(Dtest,Ypred)


plt.figure()
sns.heatmap(conf_mat,annot=True,fmt='g',cbar=False)
plt.show()

YpredTrening = model.predict(Xtrening,verbose=0)
YpredTrening = np.round(YpredTrening)
YpredTest = model.predict(Xtest,verbose=0)
YpredTest = np.round(YpredTest)

accTrening = accuracy_score(Dtrening,YpredTrening)
print(accTrening*100)
accTest = accuracy_score(Dtest,YpredTest)
print(accTest*100)
```

In [31]:
```
from keras.regularizers import l2

model = Sequential()
model.add(Dense(200,activation='relu',input_dim=X.shape[1],kernel_regularizer=l2(0.08)))
model.add(Dense(1,activation='sigmoid'))

model.compile('adam',loss='binary_crossentropy', metrics=['accuracy'])
model.summary()


max_epoha = 500
history = model.fit(x=Xtrening, y=Dtrening,epochs=max_epoha, batch_size=32, shuffle=32,verbose=0, validation_data=(Xtest,Dtest))

Ypred = model.predict(Xtest)
Ypred = np.round(Ypred)
conf_mat = confusion_matrix(Dtest,Ypred)

plt.figure()
sns.heatmap(conf_mat,annot=True,fmt='g',cbar=False)
plt.show()

plt.figure()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['loss', 'val_loss'])

YpredTrening = model.predict(Xtrening,verbose=0)
YpredTrening = np.round(YpredTrening)
YpredTest = model.predict(Xtest,verbose=0)
YpredTest = np.round(YpredTest)

accTrening = accuracy_score(Dtrening,YpredTrening)
print(accTrening*100)
accTest = accuracy_score(Dtest,YpredTest)
print(accTest*100)
```

Zaključak

Nakon svih ispitivanja baze podataka, došle smo do zaključka da data baza podataka zahteva značajno više ispitanika i atributa na osnovu kojih bi se moglo predvideti ponovno pojavljivanje raka dojke.

U kontekstu informativnosti, shvatamo da su najinformativnija obeležja stepen malignosti i broj limfnih čvorova zahvaćenih kancerogenim ćelijama. Takođe, shvatamo da su informacije kao što su da li je kancer na levoj ili desnoj dojki ili u kom delu dojke se nalazi najmanje informativne. Iz korelacione analize dolazimo do zaključka da su godine i menopauza u velikoj meri korelisane, kao i broj limfnih čvorova i postojanje omotača oko limfnih čvorova.

LDA metodom uspele smo da svedemo podatke na jednu dimenziju, ali time nismo dobili zadovoljavajuću informativnost, pa smo umesto parametarskog klasifikatora koristile Bajesov test minimalne verovatnoće greške i ostvarile tačnost klasifikacije od 73.5%.

KNN metoda je dala najbolje rezultate za vrednost parametra k=5 i time smo uspele da ostvarimo tačnost neparametarske klasifikacije od 73.5%.

Prilikom ispitivanja optimalne strukture neuralne mreže, pokazale smo da previše kompleksne mreže dovode do preobučavanja i uspele smo da sprečimo preobučavanje ranim zaustavljanjem i regularizacijom. Takođe, pokazale smo da neuralne mreže sa nedovoljnim brojem čvorova ne mogu klasifikovati podatke sa zadovoljavajućom tačnošću. Kao optimalnu strukturu predložile smo neuralnu mrežu sa 2 skrivena sloja i jednim izlaznim i ostvarile tačnost od 74%.