Univerzitet u Kragujevcu

Domaći zadatak

iz predmeta Sistem za podršku odlučivanju

Tema:

Implementacija klasifikacionih algoritmima sa nadgledanim učenjem

mentori: Ognjen Pavić,
prof. dr. Tijana Geroski,
prof. dr. Nenad Filipović
student: Željko Simić 3vi/2023

Kragujevac 2024.

1 Uvod

Dat je ‘‘`Social_Network_Ads.csv`’’ dataset.[1] Namenjen za sagledanje U sebi sadrı400 instanci (uzoraka), a ima u sebi za svaki po 4 kolona atributa (posmatran kao ulazna vrednost):

- `User ID` - Ukazuje na udeljeni jedinstveni identifikator korisnika posmatranog sistema. **Biće izbačen pri sagledanju i neće biti uzet u obzir**;
- `Gender` - Ukazuje na pol korisnika, kategoričkih vrednosti (`Male, Female`);
- `Age` - Ukazuje na starost korisnika, sa kontinualnim vrednostima u domenu $[18, 60]$
- `EstimatedSalary` - Ukazuje na starost korisnika sistema, kontinualna vrednost u domenu od $[15000, 150000]$

i 1 ciljnu vrednost (posmatran kao izlaznu vrednost):

- `Purchased` - Ukazuje da li je korisnik sistema pazario proizvod za kog važi ovaj skup podataka, koji je kategorička vrednost `0, 1` - što dovodi klasifikacije koje se obavljaju budu **binarne**.

2 Metodologija

Uz tekst seminarskog rada je priložen izvorni kod (na putanji `./src/main.py`) koji će se ovde sagledati. Implementacija 1. u sebi sadrži deklaracije uvedenih modula iz kojih će biti pozivane njihove komponente u daljem radu (praktično realizovanje teorijskog rada na ovu temu[2]). Korišćen je objektni tip podataka `pandas.DataFrame`[3] kojim će biti urađeno strukturno skladištenje (sa metodom `pd.read_csv(putanja_csv_fajla_skupa_podataka)`) i manipulisano elementima skupa podataka koji je bio pomenut u uvodu. Biće izvršeno predprocesiranje u kom spada eliminisanje `User ID` atributa, enkodiranje `Gender` atributa (da bi bio prikladan za dalji rad sa klasifikatorima iz vrednosti tipa string-a u broj)[4], podela skupa podataka uz `train_test_split(...)` na trening i test skupove podataka po nekoj razmeri $8:2$ za ulazne i izlazne vrednosti klasifikacija.[5]

```
1
2  import pandas as pd
3  from sklearn.linear_model import LogisticRegression
4  from sklearn.tree import DecisionTreeClassifier
5  from sklearn.ensemble import RandomForestClassifier
6  from sklearn.naive_bayes import GaussianNB
7  from sklearn.naive_bayes import MultinomialNB
8  from sklearn.naive_bayes import BernoulliNB
9  from sklearn.naive_bayes import CategoricalNB
10 from sklearn.naive_bayes import ComplementNB
11 from sklearn.svm import SVC
12 from sklearn.neighbors import KNeighborsClassifier
13 # from sklearn.neighbors import RadiusNeighborsClassifier
14
15 from sklearn.model_selection import train_test_split
16 import seaborn as sb
17 import matplotlib.pyplot as plt
18 import numpy as np
19 from sklearn.metrics import classification_report
20 from sklearn.metrics import confusion_matrix
21 from sklearn.preprocessing import LabelEncoder
22 from sklearn.model_selection import GridSearchCV
23
24 ...
25
26 dataframe = pd.read_csv('./Social_Network_Ads.csv')
27 # storing all dataframe without user ids
28 data = dataframe.iloc[:,1:]
29
30 le = LabelEncoder()
31 le.fit(data.iloc[:, 0])
32
33 # Encoding to numerical binary values for the purposes of dealing with
      NaiveBayes classificators
34 # https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.
      LabelEncoder.html
35
36 encoded_genders = le.transform(data.iloc[:, 0])
37 data[data.columns[0]] = encoded_genders
38
39 ...
40
41 x_train, x_test, y_train, y_test = train_test_split(
42     data[data.columns[:-1]],
43     data[data.columns[-1]],
44     test_size=0.2
```

```
)
```

Implementacija 1: Predprocesiranje podataka i navođenje modula koji će se koristiti

U implementaciji 2. vrši se generisanje slika 1. (za korelaciju među features-ima po verovatonosnoj distributivnoj funkciji (PDF) na target vrednost `Purchased`), 2. (za distribuciju naspram feature-a `Age` i target vrednosti), 3. (za distribuciju naspram feature-a `EstimatedSalary`). Ovo sve je obavljeno uz metode `seaborn.pairplot` za sliku 1.[6] i `seaborn.displot` za slike 2. i 3.[7]

```
    # visualization of correlation
    sb.pairplot(data, hue="Purchased")

    # visualization of distribution against Age feature and target value
    sb.displot(data,
    hue = "Purchased",
    palette = "Spectral",
    x = data['Age'],
    kind = 'kde',
    fill = True)
# visualization of distribution against EstimatedSalary feature and target
    value
sb.displot(data,
hue = "Purchased",
palette = "Spectral",
      x = data['EstimatedSalary'],
      kind = 'kde',
      fill = True)
```

Implementacija 2: Navođenje načina vizuelizacija za sliku `1., 2. i 3.`

Na slici 1. je moguće uočiti da PDF za feature `Gender` vrednosti među sobom (`0` za `Male`, a `1` za `Female`) po `Purchased` vrednosti ukazuje na malo veću kupovnu moć posmatranog proizvoda kod muškaraca, nego kod žena.

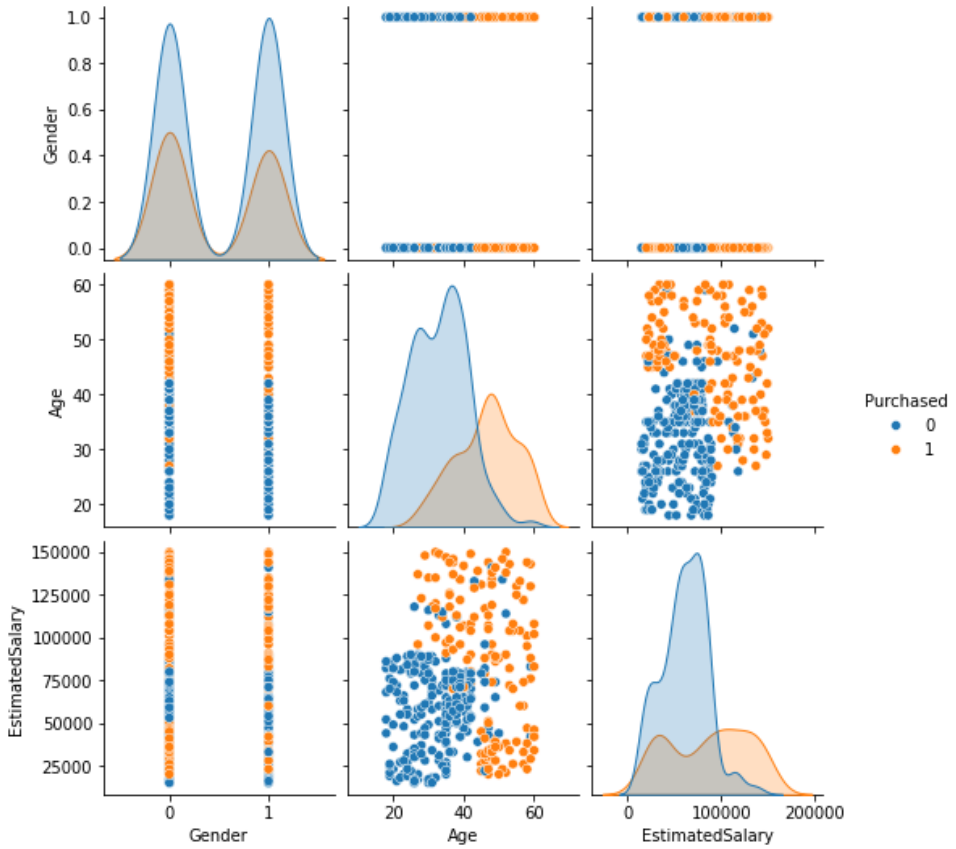
Za PDF `Gender` naspram `Age` po `Purchased` može se videti da kupovna moć ovog proizvoda je kod muškaraca i kod žena u domenu od 45+ godina, ali kod muškaraca možda malo prostranjenije po godinama.

Za PDF `Gender` naspram `EstimatedSalary` po `Purchased` može se videti da kupovna moć ovog proizvoda je kod muškaraca i kod žena u domenu od 75000+, ali da ne kupuju oba pola u domenu od [50000, 75000].

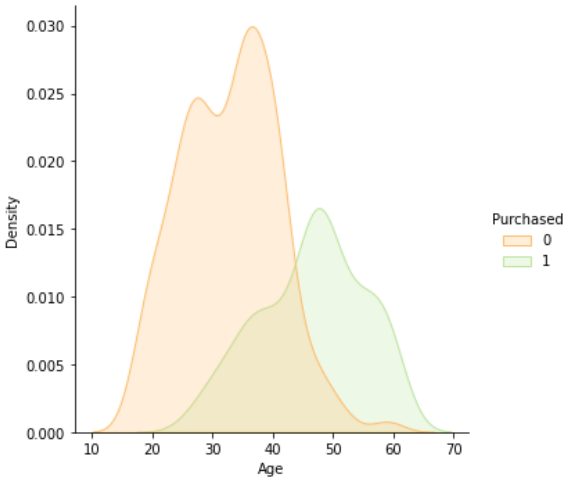
Za PDF `Age` među sobom po `Purchased` može se videti da kupovna moć ovog proizvoda je najveća negde oko 50 godina, ali i da ne kupuju najviše sa 30 godina što je i dominantnije. Što je i prikazano na slici 2.

Za PDF `Age` naspram `EstimatedSalary` po `Purchased` može se videti da kupovna moć ovog proizvoda je kod starosti [40, 60] svih raspoloživih prihoda, ali da ne kupuju domenu prihod od [0, 75000] kod onih koji su u starosti od [20, 40] godina.

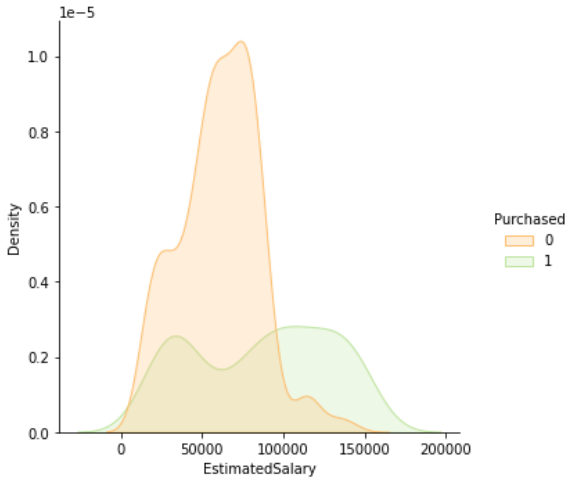
Za PDF `EstimatedSalary` među sobom po `Purchased` može se videti da kupovna moć ovog proizvoda je najveća među prihodima od 100000, ali i da ne kupuju najviše sa prihodom 75000 što je i dominantnije. Što je i prikazano na slici 3.



Slika 1: Vizuelizacija za korelaciju među features-ima po verovatonosnoj distributivnoj funkciji (PDF) na target vrednost `Purchased`



Slika 2: Vizuelizacija za distribuciju naspram feature-a `Age` i target vrednosti



Slika 3: Vizuelizacija za distribuciju naspram feature-a `EstimatedSalary` i target vrednosti

U implementaciji 3. je realizovana klasifikacija **logističkom regresijom sa optimizacijom hiperparametara** korišćenjem `LogisticRegression`[8] i `GridSearchCV`[9]. U promenlji-

vu `param_grid` smešteni su parovi ključ, vrednost za koncepte:

- `C` - podešena jačina inverzne regulacije, pozitivna decimalna vrednost. Manja vrednost jača regulacija. `np.logspace(...)` daje listu elemenata $10^{-1}, ..., 10^{-9}$ gde je stepen 10, ali izvodilac je određen $\frac{ln(\texttt{broj\_elemenata})}{ln(\texttt{stepen})}$ korakom.[10]
- `penalty` - podešeni su određeni režimi penala (kazne), l2 bi inače bio podrazumevan, ovi penali mogu da ne rade za određene rešavače.
- `solver` - podešeni rešavači, tj. algoritmi za optimizacioni problem, gde je `lbfgs` podrazumevan.
- `class_weight` - podešena pristrasnost za neku ciljanu vrednost `Purchased`-a, npr. u ovom slučaju samo za vrednost `0`, tj. nekupljenog proizvoda je manja pristrasnost - od 20%, nego kod vrednosti `1`, tj. kupljenog - od 80%.
- `l1_ration` - koristan samo u slučaju režima penala `elasticnet` u kom je korišćen kao miksing parametar kao što je u teorijskom domaćem zadatku pričano o *regularizacije elastične mreže* ρ.
- `multi_class` - moguće podrazumevano automatski, ali podešava ishode po binarnom ili po višeklasnom funkciji PDF. Priča vezana oko binarne, OVR, multinomijalne logističke regresije u teoriji.

`GridSearchCV` radi isprobavanje svakog ponaosob hiperparametra nad klasifikatorom i stratifikovan cross-validation postupak na 10 fold-ova. Izvlači se najbolji ishod evaluacija pri predviđanju test skupom nad obučenim modelom. Kasnije će biti diskutovano o analizi koja je obrađena uz `classification_report(...)` i `confusionMatrix(...)` - koji je korisnički definisana funkcija prikazana u implementaciji 4. gde se vrši iscrtavanje konfuzione matrice (pogotka ciljanih vrednosti naspram predviđenih modelom i stvarnih vrednosti iz trening skupa).

```
    print('Logistic Regression prediction:')
LR = LogisticRegression()
param_grid = {
    'C': np.logspace(-1, -9, num=10),
  'penalty': ['l1', 'l2', 'elasticnet'],
  'solver': ['lbfgs', 'liblinear', 'newton-cg', 'newton-cholesky', 'saga'],
  'class_weight' : ['balanced', 'None', {0: 0.2, 1:0.8}],
  'l1_ratio': [0, 0.1, 1],
  'multi_class': ['auto', 'ovr', 'multinomial']

}
LR_grid = GridSearchCV(estimator = LR,
                       param_grid=param_grid, verbose=0,
                       cv=10, n_jobs=-1)
LR_grid.fit(x_train,y_train)
prediction=LR_grid.predict(x_test)
print('Best case:')
print(LR_grid.best_estimator_)
print()
report = classification_report(list(y_test), prediction)
print(report)

cm=confusionMatrix(list(y_test),prediction, "Logistic regression w/
   optimization")
```

Implementacija 3: Logistička regresija sa optimizacijom hiperparametara.

```
def confusionMatrix(y_true,y_pred,title):
  cm=confusion_matrix(y_pred,y_true)
  plt.figure()
  sb.heatmap(cm, annot=True, fmt='0')
  plt.title(title)
  plt.xlabel('True Value')
  plt.ylabel('Predicted Value')
```

Implementacija 4: Funkcija za generisanje konfuzione matrice.

U implementaciji broj 5. vrši se obična **logistička regresija** bez ikakvih optimizacija hiperparametara.

```
    print(f'Logistic Regression prediction without optimization:')
    LR.fit(x_train,y_train)
    prediction=LR.predict(x_test)

    print()
    report = classification_report(list(y_test), prediction)
    print(report)
```

Implementacija 5: Logistička regresija bez ikakvih optimizacija hiperparametara.

U implementaciji 6. je realizovana klasifikacija **stabla odlučivanja sa optimizacijom hiperparametara** korišćenjem `DecisionTreeClassifier`[11] i `GridSearchCV`. U promenljivu `param_grid` smešteni su parovi ključ, vrednost za koncepte:

- `criterion` - mera za podešavanje funkcije određivanja kvaliteta podele.
- `max_depth` - najveća dubina stabla odluke koja je uzeta u razmatranje.
- `min_samples_leaf` - minimalan broj uzoraka u čvorovima listova u stablu, tačka podele ma koje dubine će biti uvažena ako se ustanovi toliki broj uzoraka trening skupa i sa leve i sa desne grane. Utiče na *smooth*-ovanje. Zaokruživanje broja vrši na više.
- `ccp_alpha` - parametar za primenu pri *odecanju minimalne cene kompleksnoti.*

`GridSearchCV` radi isprobavanje svakog ponaosob hiperparametra nad klasifikatorom i stratifikovan cross-validation postupak na 10 fold-ova. Izvlači se najbolji ishod evaluacija pri predviđanju test skupom nad obučenim modelom. Kasnije će biti diskutovano o analizi koja je obrađena uz `classification_report(...)` i `confusionMatrix(...)`.

```
DT = DecisionTreeClassifier()
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': np.arange(3, 15, step=6),
    'min_samples_leaf': np.arange(1, 10, step=3),
    'ccp_alpha': [0, 0.1, 0.2]
}
DT_grid = GridSearchCV(estimator = DT,
                       param_grid=param_grid, verbose=0,
                       cv=10, n_jobs=-1)
DT_grid.fit(x_train,y_train)
prediction=DT_grid.predict(x_test)
print('Best case:')
print(DT_grid.best_estimator_)
print()
report = classification_report(list(y_test), prediction)
print(report)
```

```
18
19 cm=confusionMatrix(list(y_test),prediction, "Decision Tree w/ optimization")
```

Implementacija 6: Stablo odlučivanja sa optimizacijama hiperparametara.

U implementaciji broj 7. vrši se obična **klasifikacija stabla odlučivanja** bez ikakvih optimizacija hiperparametara.

```
1     print(f'Decision Tree prediction without optimization:')
2     DT.fit(x_train,y_train)
3     prediction=DT.predict(x_test)
4
5     print()
6     report = classification_report(list(y_test), prediction)
7     print(report)
```

Implementacija 7: Stablo odlučivanja bez ikakvih optimizacija hiperparametara.

U implementaciji 8. je realizovana klasifikacija **slučajnih šuma sa optimizacijom hiperparametara** korišćenjem `RandomForestClassifier`[12] i `GridSearchCV`. U promenljivu `param_grid` smešteni su parovi ključ, vrednost za koncepte:

- `n_estimators` - broj stabala odlučivanja u šumi, podrazumeva se da bude 100.
- `max_depth` - najveća dubina stabla odluke koja je uzeta u razmatranje.
- `min_samples_leaf` - minimalan broj uzoraka u čvorovima listova u stablu, tačka podele ma koje dubine će biti uvažena ako se ustanovi toliki broj uzoraka trening skupa i sa leve i sa desne grane. Utiče na *smooth*-ovanje. Zaokruživanje broja vrši na više.
- `min_samples_split` - minimalan potreban broj uzoraka da bi se desila podela, ako nije `int` tipa onda će da vrši zaokruživanje broja naviše.
- `bootstrap` - koriste se se bootstrap uzorci pri gradnji stabala, inače je korišćen čitav skup podataka za izgradnju stabala (što je podrazumevan slučaj1).

`GridSearchCV` radi isprobavanje svakog ponaosob hiperparametra nad klasifikatorom i stratifikovan cross-validation postupak na 10 fold-ova. Izvlači se najbolji ishod evaluacija pri predviđanju test skupom nad obučenim modelom. Kasnije će biti diskutovano o analizi koja je obrađena uz `classification_report(...)` i `confusionMatrix(...)`.

```
1  print('Random Forest prediction:')
2  RF = RandomForestClassifier()
3  param_grid = {
4      'n_estimators': np.arange(start=4, stop=20, step=4),
5      'max_depth': list(range(10, 110, 50)) + [None],
6      'min_samples_split': [2, 5, 10],
7      'min_samples_leaf': [1, 2, 4],
8      'bootstrap': [True, False]
9  }
10 RF_grid = GridSearchCV(estimator = RF,
11                        param_grid=param_grid, verbose=0,
12                        cv=10, n_jobs=-1)
13 RF_grid.fit(x_train,y_train)
14 prediction=RF_grid.predict(x_test)
15 print('Best case:')
16 print(RF_grid.best_estimator_)
17 print()
```

```
report = classification_report(list(y_test), prediction)
print(report)

cm=confusionMatrix(list(y_test),prediction, "Random Forst w/ optimization")
```

Implementacija 8: Slučajna šuma sa optimizacijama hiperparametara.

U implementaciji broj 9. vrši se obična **klasifikacija slučajnih šuma** bez ikakvih optimizacija hiperparametara.

```
print(f'Random Forest prediction without optimization:')
RF.fit(x_train,y_train)
prediction=RF.predict(x_test)
report = classification_report(list(y_test), prediction)
print(report)
```

Implementacija 9: Slučajne šume bez ikakvih optimizacija hiperparametara.

U implementaciji 10. je realizovana klasifikacija **Gausov naivni Bajes sa optimizacijom hiperparametara** korišćenjem `GaussianNB`[13] i `GridSearchCV`. U promenljivu `param_grid` smešteni su parovi ključ, vrednost za koncepte:

- `var_smoothing` - podešava količinu najveće disperzije svih features-a koji doprinose stabilnosti sračunavanja.

`GridSearchCV` radi isprobavanje svakog ponaosob hiperparametra nad klasifikatorom i stratifikovan cross-validation postupak na 10 fold-ova. Izvlači se najbolji ishod evaluacija pri predviđanju test skupom nad obučenim modelom. Kasnije će biti diskutovano o analizi koja je obrađena uz `classification_report(...)` i `confusionMatrix(...)`.

```
print('Gaussian Naive Bayes prediction:')

GNB=GaussianNB()
param_grid = {
    'var_smoothing': np.logspace(1, -9, num=10)
}
GNB_grid = GridSearchCV(estimator = GNB,
            param_grid=param_grid, verbose=0,
                        cv=10, n_jobs=-1)
GNB_grid.fit(x_train,y_train)
prediction=GNB_grid.predict(x_test)
print('Best case:')
print(GNB_grid.best_estimator_)
print()
report = classification_report(list(y_test), prediction)
print(report)

cm=confusionMatrix(list(y_test),prediction, "Gaussian Naive Bayes w/
   optimization")
```

Implementacija 10: Gausov naivni Bajes sa optimizacijama hiperparametara.

U implementaciji broj 11. vrši se obična **klasifikacija Gausov naivni Bajes** bez ikakvih optimizacija hiperparametara.

```
print(f'Gaussian Naive Bayes prediction without optimization:')
GNB.fit(x_train,y_train)
prediction=GNB.predict(x_test)
report = classification_report(list(y_test), prediction)
print(report)
```

Implementacija 11: Gausov naivni Bajes bez ikakvih optimizacija hiperparametara.

U implementaciji 12. je realizovana klasifikacija **multinomijani naivni Bajes sa optimizacijom hiperparametara** korišćenjem `MultinomialNB`[14] i `GridSearchCV`. U promenljivu `param_grid` smešteni su parovi ključ, vrednost za koncepte:

- `alpha` - aditivni (Laplace/Lidstone) smoothing parametar.

`GridSearchCV` radi isprobavanje svakog ponaosob hiperparametra nad klasifikatorom i stratifikovan cross-validation postupak na 10 fold-ova. Izvlači se najbolji ishod evaluacija pri predviđanju test skupom nad obučenim modelom. Kasnije će biti diskutovano o analizi koja je obrađena uz `classification_report(...)` i `confusionMatrix(...)`.

```
print(f'Multinomial Naive Bayes prediction:')
MNB=MultinomialNB()
param_grid = {
    'alpha': np.logspace(-1, -9, num=10)
}
MNB_grid = GridSearchCV(estimator = MNB,
            param_grid=param_grid, verbose=0,
                        cv=10, n_jobs=-1)
MNB_grid.fit(x_train,y_train)
prediction=MNB_grid.predict(x_test)
print('Best case:')
print(MNB_grid.best_estimator_)
print()
report = classification_report(list(y_test), prediction)
print(report)
```

Implementacija 12: Multinomialni naivni Bajes sa optimizacijama hiperparametara.

U implementaciji broj 13. vrši se obična **klasifikacija multinomijalni naivni Bajes** bez ikakvih optimizacija hiperparametara.

```
MNB.fit(x_train,y_train)
prediction=MNB.predict(x_test)
print(f'Multinomial Naive Bayes prediction without optimization:')
report = classification_report(list(y_test), prediction)
print(report)
```

Implementacija 13: Multinomijalni naivni Bajes bez ikakvih optimizacija hiperparametara.

U implementaciji 14. je realizovana klasifikacija **Bernuli naivni Bajes sa optimizacijom hiperparametara** korišćenjem `BernoulliNB`[15] i `GridSearchCV`. U promenljivu `param_grid` smešteni su parovi ključ, vrednost za koncepte:

- `alpha` - aditivni (Laplace/Lidstone) smoothing parametar.

`GridSearchCV` radi isprobavanje svakog ponaosob hiperparametra nad klasifikatorom i stratifikovan cross-validation postupak na 10 fold-ova. Izvlači se najbolji ishod evaluacija pri predviđanju test skupom nad obučenim modelom. Kasnije će biti diskutovano o analizi koja je obrađena uz `classification_report(...)` i `confusionMatrix(...)`.

```
print(f'Bernoulli Naive Bayes prediction:')
BNB=BernoulliNB()
param_grid = {
    'alpha': np.logspace(-1, -9, num=10)
}
BNB_grid = GridSearchCV(estimator = BNB,
            param_grid=param_grid, verbose=0,
                        cv=10, n_jobs=-1)
BNB_grid.fit(x_train,y_train)
```

```
prediction=BNB_grid.predict(x_test)
print('Best case:')
print(BNB_grid.best_estimator_)
print()
report = classification_report(list(y_test), prediction)
print(report)
```

Implementacija 14: Bernulijev naivni Bajes sa optimizacijama hiperparametara.

U implementaciji broj 15. vrši se obična **klasifikacija Bernulijev naivni Bajes** bez ikakvih optimizacija hiperparametara.

```
print(f'Bernoulli Naive Bayes prediction without optimization:')
BNB=BernoulliNB()
BNB.fit(x_train,y_train)
prediction=BNB.predict(x_test)

print()
report = classification_report(list(y_test), prediction)
print(report)
```

Implementacija 15: Bernulijev naivni Bajes bez ikakvih optimizacija hiperparametara.

U implementaciji 16. je realizovana klasifikacija **kategorički naivni Bajes sa optimizacijom hiperparametara** korišćenjem `CategoricalNB`[17] i `GridSearchCV`. U promenljivu `param_grid` smešteni su parovi ključ, vrednost za koncepte:

- `alpha` - aditivni (Laplace/Lidstone) smoothing parametar.

`GridSearchCV` radi isprobavanje svakog ponaosob hiperparametra nad klasifikatorom i stratifikovan cross-validation postupak na 10 fold-ova. Izvlači se najbolji ishod evaluacija pri predviđanju test skupom nad obučenim modelom. Kasnije će biti diskutovano o analizi koja je obrađena uz `classification_report(...)` i `confusionMatrix(...)`.

```
print(f'Categorical Naive Bayes prediction:')

CatNB=CategoricalNB()
param_grid = {
    'alpha': np.logspace(-1, -9, num=10)
}
CatNB_grid = GridSearchCV(estimator = CatNB,
            param_grid=param_grid, verbose=0,
                        cv=10, n_jobs=-1)
CatNB_grid.fit(x_train,y_train)
prediction=CatNB_grid.predict(x_test)
print('Best case:')
print(CatNB_grid.best_estimator_)
print()
report = classification_report(list(y_test), prediction)
print(report)


CatNB.fit(x_train,y_train)
prediction=CatNB.predict(x_test)
```

Implementacija 16: Kategorički naivni Bajes sa optimizacijama hiperparametara.

U implementaciji broj 17. vrši se obična **klasifikacija kategorički naivni Bajes** bez ikakvih optimizacija hiperparametara.

```

print(f'Categorical Naive Bayes prediction without optimization:')
```

```
CatNB.fit(x_train,y_train)
prediction=CatNB.predict(x_test)

print()
report = classification_report(list(y_test), prediction)
print(report)
```

Implementacija 17: Kategorički naivni Bajes bez ikakvih optimizacija hiperparametara.

U implementaciji 18. je realizovana klasifikacija **multinomijani naivni Bajes sa optimizacijom hiperparametara** korišćenjem `ComplementNB`[16] i `GridSearchCV`. U promenljivu `param_grid` smešteni su parovi ključ, vrednost za koncepte:

- `alpha` - aditivni (Laplace/Lidstone) smoothing parametar.

`GridSearchCV` radi isprobavanje svakog ponaosob hiperparametra nad klasifikatorom i stratifikovan cross-validation postupak na 10 fold-ova. Izvlači se najbolji ishod evaluacija pri predviđanju test skupom nad obučenim modelom. Kasnije će biti diskutovano o analizi koja je obrađena uz `classification_report(...)` i `confusionMatrix(...)`.

```
print(f'Complement Naive Bayes prediction:')

CNB=ComplementNB()
param_grid = {
    'alpha': np.logspace(-1, -9, num=10)
}
CNB_grid = GridSearchCV(estimator = CNB,
            param_grid=param_grid, verbose=0,
                        cv=10, n_jobs=-1)
CNB_grid.fit(x_train,y_train)
prediction=CNB_grid.predict(x_test)
print('Best case:')
print(CNB_grid.best_estimator_)
print()
report = classification_report(list(y_test), prediction)
print(report)
```

Implementacija 18: Komplementarni naivni Bajes sa optimizacijama hiperparametara.

U implementaciji broj 19. vrši se obična **klasifikacija komplement naivni Bajes** bez ikakvih optimizacija hiperparametara.

```
print(f'Complement Naive Bayes prediction without optimization:')
CNB.fit(x_train,y_train)
prediction=CNB.predict(x_test)

print()
report = classification_report(list(y_test), prediction)
print(report)
```

Implementacija 19: Komplement naivni Bajes bez ikakvih optimizacija hiperparametara.

U implementaciji 20. je realizovana klasifikacija **mašina potpornih vektora (SVM) sa optimizacijom hiperparametara** korišćenjem `SVC`[18] i `GridSearchCV`. U promenljivu `param_grid` smešteni su parovi ključ, vrednost za koncepte:

- `C` - parametar inverzne regularizacije, pozitivna vrednost, korenovana je u `l2` režimu rada.
- `gamma` - koeficijent režima kernel-a `rbf, poly, sigmoid`. Vrednosti koji mogu biti unete `scale` (pa je ona $\frac{1}{\texttt{broj\_featuresa * disperzija\_ulaza}}$), `auto` (pa je ona $\frac{1}{\texttt{broj\_featuresa}}$), nenegativna vrednost

- `kernel` - `rbf` je podrazumevana vrednost, ali se specifikuje koji se algoritam koristi.

`GridSearchCV` radi isprobavanje svakog ponaosob hiperparametra nad klasifikatorom i stratifikovan cross-validation postupak na 3 fold-ova. Izvlači se najbolji ishod evaluacija pri predviđanju test skupom nad obučenim modelom. Kasnije će biti diskutovano o analizi koja je obrađena uz `classification_report(...)` i `confusionMatrix(...)`.

```
print(f'SVM prediction:')

SVM=SVC()
param_grid = {
    'C': [0.1, 10],
    'gamma': [1, 0.01],
    'kernel': ['rbf', 'linear', 'sigmoid']
}
SVM_grid = GridSearchCV(estimator = SVM,
            param_grid=param_grid, verbose=1,
                        cv=3, n_jobs=-1)
SVM_grid.fit(x_train,y_train)
prediction=SVM_grid.predict(x_test)
print('Best case:')
print(SVM_grid.best_estimator_)
print()
report = classification_report(list(y_test), prediction)
print(report)
cm=confusionMatrix(list(y_test),prediction, "SVM w/ optimization")
```

Implementacija 20: SVM sa optimizacijama hiperparametara.

U implementaciji broj 21. vrši se obična **klasifikacija SVM** bez ikakvih optimizacija hiperparametara.

```
print(f'SVM prediction without optimization:')
SVM.fit(x_train,y_train)
prediction=SVM.predict(x_test)

print()
report = classification_report(list(y_test), prediction)
print(report)
```

Implementacija 21: SVM bez ikakvih optimizacija hiperparametara.

U implementaciji 22. je realizovana klasifikacija **k-Najbližih suseda (kNN) sa optimizacijom hiperparametara** korišćenjem `KNeighborsClassifier`[19] i `GridSearchCV`. U promenljivu `param_grid` smešteni su parovi ključ, vrednost za koncepte:

- `n_neighbors` - korisnički definisan broj suseda korišćeni za upitne uzorke, podrazumeva se da bude 5.
- `leaf_size` - veličina listova korišćen pri radu sa *KDTree i BallTree* algoritmima, ali u ovom slučaju je algoritam automatski određen, podrazumeva se da je 30. Može doprineti brzini izrade upita, pogodnosti memorijskog skladištenja drveća.
- `p` - izvodilac stepena metrika Minkovskog, može se pretvoriti p=1 menheten distancu, u p=2 euklidsku distancu, itd. Mora biti pozitivna.
- `weights` - funkcija težina korišćena pri predviđanju, `uniform` je kada svi susedi teže isto i podrazumevana je, `distance` - daje se veća pristrasnost bližim susedima,

- `metric` - podrazumevana je `minkovski`, korišćena pri određivanju distanci.

`GridSearchCV` radi isprobavanje svakog ponaosob hiperparametra nad klasifikatorom i stratifikovan cross-validation postupak na 5 fold-ova. Izvlači se najbolji ishod evaluacija pri predviđanju test skupom nad obučenim modelom. Kasnije će biti diskutovano o analizi koja je obrađena uz `classification_report(...)` i `confusionMatrix(...)`.

```
print(f'KNN prediction:')
estimator_KNN = KNeighborsClassifier(algorithm='auto')
parameters_KNN = {
    'n_neighbors': (1,10),
    'leaf_size': (20,40),
    'p': (1,2),
    'weights': ('uniform', 'distance'),
    'metric': ('minkowski', 'chebyshev')
}
# with GridSearch
grid_search_KNN = GridSearchCV(
    estimator=estimator_KNN,
    param_grid=parameters_KNN,
    scoring = 'accuracy',
    n_jobs = -1,
    cv = 5
)
grid_search_KNN.fit(x_train,y_train)
prediction=grid_search_KNN.predict(x_test)
print('Best case:')
print(grid_search_KNN.best_estimator_)
print()
report = classification_report(list(y_test), prediction)
print(report)
cm=confusionMatrix(list(y_test),prediction, "KNN w/ optimization")
```

Implementacija 22: kNN sa optimizacijama hiperparametara.

U implementaciji broj 23. vrši se obična **klasifikacija kNN** bez ikakvih optimizacija hiperparametara.

```
print(f'KNN prediction without optimization:')
estimator_KNN = KNeighborsClassifier(algorithm='auto')
estimator_KNN.fit(x_train,y_train)
prediction=estimator_KNN.predict(x_test)
report = classification_report(list(y_test), prediction)
print(report)
```

Implementacija 23: kNN bez ikakvih optimizacija hiperparametara.

3 Rezultati

Dati skup podataka ne mora podlezati skaliranjima i skup je stabilan. Na slici 1. je moguće uočiti da nema mnogo prilika da se dogodi overfitting pri klasifikaciji. U daljem tekstu biće obrađene diskusije oko rezultata obavljenih evaluacija nad modelima i sa (gde će biti ukazano na parametre koji su najboljeg mogućeg ishoda) i bez optimizacija hiperparametara. Proračun metrika po svakoj ciljanoj vrednosti vrši se sagledanjem vrednosti koje su ključne. Senzitivitnosti (recall) ističe koliko validacijom model dobro predviđa u skladu 2 uparena skupa. Dok specifičnost ističe koliko se za 2 uparena skupa poklopljeno loše predviđa po modelu. Preciznost govori koliko je model sposoban da poklopi činjeničnu evaluaciju naspram nečinjenične sa obrascem skupa (test skupa) predvidjanog nad trening skupom. Tačnost u ovom slučaju govori koliko činjenični skup je pogodan naspram modela. F1-score računat po formuli $2 * \frac{precision * recall}{precision + recall}$ koji je kombinacija senzitivnosti i preciznosti. Takođe se daju statističke vrednosti:

1. koliki je krajnji accuracy,
2. macro avg - metrike (preciznost, senzitivnost, f1) se izračunavaju prosekom podjednakog udela svake vrednosti ponaosob,[20][21]
3. weighted avg - metrike se računaju tako što na prosek utiče težinski udeo po zastupljenosti vrednosti klasifikatornog atributa ponaosob.

Formiranje konfuzione matrice koja služi za prikaz zastupljenosti pri poklapanju skupa podataka nakon predikcija sa onim koji su dati pre predikcija, prikaz konfuzione matrice, zasebna izdvajanja metrika za svaku klasu zasebno je obavljena.

3.1 Logistička regresija

U navedenom izlazu 1. data je analiza evaluacije klasifikatora optimizovanog hiperparametrima koji ishoduju najbolji slučaj:

```
C=0.0016681005372000592,
class_weight='balanced',
l1_ratio=0,multi_class='multinomial',
solver='newton-cg'
```

Tačnost koja je ukupna dobijena je 88%. Preciznost 96% za vrednost klase 0 i 71% za vrednost klase 1, senzitivnost 86% za vrednost klase 0 i 91% za vrednost klase 1, F1-score 91% za vrednost klase 0 i 80% za vrednost klase 1.

```
1     Best case:
2     LogisticRegression(C=0.0016681005372000592, class_weight='balanced',
      l1_ratio=0,
3     multi_class='multinomial', solver='newton-cg')
4
5
6                precision    recall  f1-score   support
7             0       0.96      0.86      0.91        58
8             1       0.71      0.91      0.80        22
9
10    accuracy                           0.88        80
11   macro avg       0.84      0.89      0.85        80
12 weighted avg       0.89      0.88      0.88        80
```

Izlaz 1: Logistička regresija sa optimizacijom hiperparametara

Za izlaz 2. tačnost koja je ukupna dobijena je 73%. Preciznost 72% za vrednost klase 0 i 0% za vrednost klase 1, senzitivnost 100% za vrednost klase 0 i 0% za vrednost klase 1, F1-score 84% za vrednost klase 0 i 0% za vrednost klase 1.

```
              precision    recall  f1-score   support
           0       0.72      1.00      0.84        58
           1       0.00      0.00      0.00        22

    accuracy                           0.73        80
   macro avg       0.36      0.50      0.42        80
weighted avg       0.53      0.72      0.61        80
```

Izlaz 2: Logistička regresija bez optimizacije hiperparametara

S obzirom da je izlaz 1. bolji od izlaza 2. daje se prikaz matrice konfuzije za njega na slici 4.

Slika 4: Matrica konfuzije za logističku regresiju sa optimizacijom hiperparametrima

3.2 Stablo odlučivanja

U navedenom izlazu 3. data je analiza evaluacije klasifikatora optimizovanog hiperparametrima koji ishoduju najbolji slučaj: `ccp_alpha=0.1, max_depth=3`.

Tačnost koja je ukupna dobijena je 89%. Preciznost 98% za vrednost klase 0 i 72% za vrednost klase 1, senzitivnost 86% za vrednost klase 0 i 95% za vrednost klase 1, F1-score 92% za vrednost klase 0 i 82% za vrednost klase 1.

```
Decision Tree prediction:
Best case:
DecisionTreeClassifier(ccp_alpha=0.1, max_depth=3)


              precision    recall  f1-score   support
           0       0.98      0.86      0.92        58
           1       0.72      0.95      0.82        22

    accuracy                           0.89        80
   macro avg       0.85      0.91      0.87        80
```

```
weighted avg       0.91      0.89      0.89        80
```

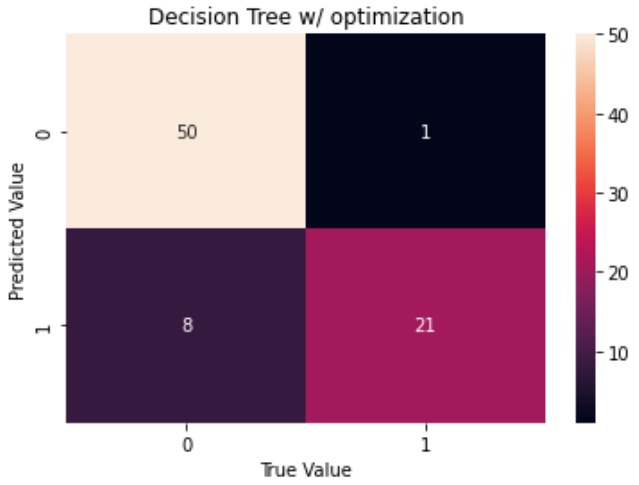
Izlaz 3: Stablo odlučivanja sa optimizacijom hiperparametara

Za izlaz 4. tačnost koja je ukupna dobijena je 88%. Preciznost 96% za vrednost klase 0 i 71% za vrednost klase 1, senzitivnost 86% za vrednost klase 0 i 91% za vrednost klase 1, F1-score 91% za vrednost klase 0 i 80% za vrednost klase 1.

```
Decision Tree prediction without optimization:

              precision    recall  f1-score   support
           0       0.96      0.86      0.91        58
           1       0.71      0.91      0.80        22

    accuracy                           0.88        80
   macro avg       0.84      0.89      0.85        80
weighted avg       0.89      0.88      0.88        80
```

Izlaz 4: Stablo odlučivanja bez optimizacija hiperparametara

S obzirom da je izlaz 3. bolji od izlaza 4. daje se prikaz matrice konfuzije za njega na slici 5.



Slika 5: Matrica konfuzije za stablo odlučivanja sa optimizacijom hiperparametrima

3.3 Slučajna šuma

U navedenom izlazu 5. data je analiza evaluacije klasifikatora optimizovanog hiperparametrima koji ishoduju najbolji slučaj:

`max_depth=10, min_samples_split=10, n_estimators=12`.

Tačnost koja je ukupna dobijena je 89%. Preciznost 98% za vrednost klase 0 i 72% za vrednost klase 1, senzitivnost 86% za vrednost klase 0 i 95% za vrednost klase 1, F1-score 92% za vrednost klase 0 i 82% za vrednost klase 1.

```
   Random Forest prediction:
Best case:
RandomForestClassifier(max_depth=10, min_samples_split=10, n_estimators=12)

              precision    recall  f1-score   support
           0       0.98      0.86      0.92        58
```

```
           1       0.72      0.95      0.82        22

    accuracy                           0.89        80
   macro avg       0.85      0.91      0.87        80
weighted avg       0.91      0.89      0.89        80
```

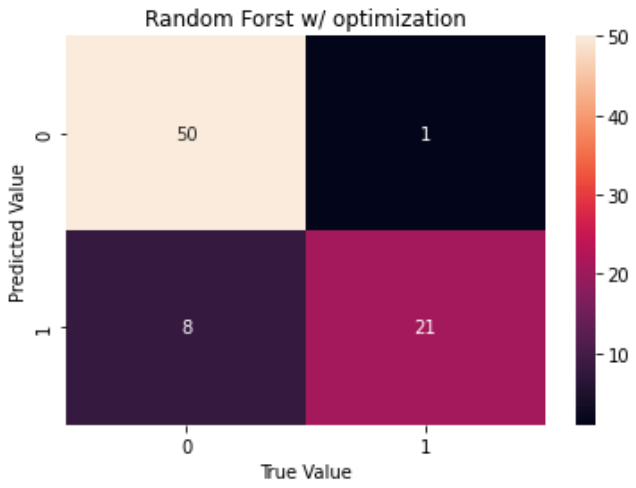
Izlaz 5: Slučajne šume sa optimizacijom hiperparametara

Za izlaz 6. tačnost koja je ukupna dobijena je 89%. Preciznost 98% za vrednost klase 0 i 72% za vrednost klase 1, senzitivnost 86% za vrednost klase 0 i 95% za vrednost klase 1, F1-score 92% za vrednost klase 0 i 82% za vrednost klase 1.

```
Random Forest prediction without optimization:
              precision    recall  f1-score   support
           0       0.98      0.86      0.92        58
           1       0.72      0.95      0.82        22

    accuracy                           0.89        80
   macro avg       0.85      0.91      0.87        80
weighted avg       0.91      0.89      0.89        80
```

Izlaz 6: Slučajne šume bez optimizacija hiperparametara

Ista je analiza ishoduje za obe evaluacije da bude ista, pa će u obzir biti uzeta mojim odabirom prvi slučaj na slici 6.



Slika 6: Matrica konfuzije za slučajnu šumu sa optimizacijom hiperparametrima

3.4 Naivni Bajes

3.4.1 Gausov Naivni Bajes

U navedenom izlazu 7. data je analiza evaluacije klasifikatora optimizovanog hiperparametrima koji ishoduju najbolji slučaj koji nije poznat po datom `var_smoothing`-u.

Tačnost koja je ukupna dobijena je 90%. Preciznost 95% za vrednost klase 0 i 79% za vrednost klase 1, senzitivnost 91% za vrednost klase 0 i 86% za vrednost klase 1, F1-score 93% za vrednost klase 0 i 83% za vrednost klase 1.

```
Gaussian Naive Bayes prediction:
Best case:
```

```
GaussianNB()
              precision    recall  f1-score   support

           0       0.95      0.91      0.93        58
           1       0.79      0.86      0.83        22

    accuracy                           0.90        80
   macro avg       0.87      0.89      0.88        80
weighted avg       0.90      0.90      0.90        80
```

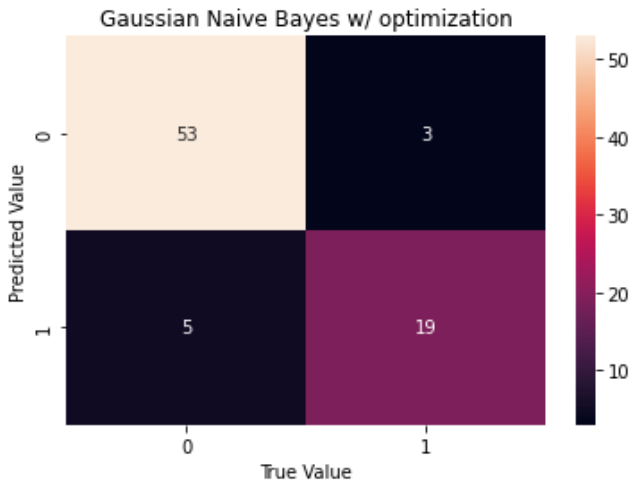
Izlaz 7: Gausov naivni Bajes sa optimizacijom hiperparametara

Za izlaz 8. tačnost koja je ukupna dobijena je 90%. Preciznost 95% za vrednost klase 0 i 79% za vrednost klase 1, senzitivnost 91% za vrednost klase 0 i 86% za vrednost klase 1, F1-score 93% za vrednost klase 0 i 83% za vrednost klase 1.

```
Gaussian Naive Bayes prediction without optimization:

              precision    recall  f1-score   support

           0       0.95      0.91      0.93        58
           1       0.79      0.86      0.83        22

    accuracy                           0.90        80
   macro avg       0.87      0.89      0.88        80
weighted avg       0.90      0.90      0.90        80
```

Izlaz 8: Gausov naivni Bajes bez optimizacija hiperparametara

Ista je analiza ishoduje za obe evaluacije da bude ista, pa će u obzir biti uzeta mojim odabirom prvi slučaj slici 7.



Slika 7: Matrica konfuzije za Gausov naivni Bajes sa optimizacijom hiperparametrima

3.4.2 Multinomijalni naivni Bajes

U navedenom izlazu 9. data je analiza evaluacije klasifikatora optimizovanog hiperparametrima koji ishoduju najbolji slučaj: `alpha=0.1`

Tačnost koja je ukupna dobijena je 73%. Preciznost 74% za vrednost klase 0 i 50% za vrednost klase 1, senzitivnost 95% za vrednost klase 0 i 14% za vrednost klase 1, F1-score 83% za vrednost klase 0 i 21% za vrednost klase 1.

```
Multinomial Naive Bayes prediction:
Best case:
MultinomialNB(alpha=0.1)

              precision    recall  f1-score   support
           0       0.74      0.95      0.83        58
           1       0.50      0.14      0.21        22

    accuracy                           0.73        80
   macro avg       0.62      0.54      0.52        80
weighted avg       0.68      0.72      0.66        80
```

Izlaz 9: Multinomijalni naivni Bajes sa optimizacijom hiperparametara

Za izlaz 10. tačnost koja je ukupna dobijena je 73%. Preciznost 74% za vrednost klase 0 i 50% za vrednost klase 1, senzitivnost 95% za vrednost klase 0 i 14% za vrednost klase 1, F1-score 83% za vrednost klase 0 i 21% za vrednost klase 1.

```
Multinomial Naive Bayes prediction without optimization:

              precision    recall  f1-score   support
           0       0.74      0.95      0.83        58
           1       0.50      0.14      0.21        22

    accuracy                           0.73        80
   macro avg       0.62      0.54      0.52        80
weighted avg       0.68      0.72      0.66        80
```

Izlaz 10: Multinomijalni naivni Bajes bez optimizacije hiperparametara

Gausov naivni Bajes se pokazao kao bolji Naivni Bajes klasifikator, pa tako da se samo posmatra matrica konfuzije za njega.

3.4.3 Bernulijev naivni Bajes

U navedenom izlazu 11. data je analiza evaluacije klasifikatora optimizovanog hiperparametrima koji ishoduju najbolji slučaj: `alpha=0.1`

Tačnost koja je ukupna dobijena je 73%. Preciznost 72% za vrednost klase 0 i 0% za vrednost klase 1, senzitivnost 100% za vrednost klase 0 i 0% za vrednost klase 1, F1-score 84% za vrednost klase 0 i 0% za vrednost klase 1.

```
Bernoulli Naive Bayes prediction:
Best case:
BernoulliNB(alpha=0.1)

              precision    recall  f1-score   support
           0       0.72      1.00      0.84        58
           1       0.00      0.00      0.00        22

    accuracy                           0.73        80
   macro avg       0.36      0.50      0.42        80
weighted avg       0.53      0.72      0.61        80
```

Izlaz 11: Bernuli naivni Bajes sa optimizacijom hiperparametara

Za izlaz 12. tačnost koja je ukupna dobijena je 73%. Preciznost 72% za vrednost klase 0 i 0% za vrednost klase 1, senzitivnost 100% za vrednost klase 0 i 0% za vrednost klase 1, F1-score 84% za vrednost klase 0 i 0% za vrednost klase 1.

```
Bernoulli Naive Bayes prediction without optimization:

```

```
              precision    recall  f1-score   support
           0       0.72      1.00      0.84        58
           1       0.00      0.00      0.00        22

    accuracy                           0.73        80
   macro avg       0.36      0.50      0.42        80
weighted avg       0.53      0.72      0.61        80
```

Izlaz 12: Bernuli naivni Bajes bez optimizacija hiperparametara

Gausov naivni Bajes se pokazao kao bolji Naivni Bajes klasifikator, pa tako da se samo posmatra matrica konfuzije za njega.

3.4.4 Kategorički naivni Bajes

U navedenom izlazu 13. data je analiza evaluacije klasifikatora optimizovanog hiperparametrima koji ishoduju najbolji slučaj: `alpha=0.1`

Tačnost koja je ukupna dobijena je 82%. Preciznost 92% za vrednost klase 0 i 64% za vrednost klase 1, senzitivnost 83% za vrednost klase 0 i 82% za vrednost klase 1, F1-score 87% za vrednost klase 0 i 72% za vrednost klase 1.

```
Best case:
CategoricalNB(alpha=0.1)

              precision    recall  f1-score   support
           0       0.92      0.83      0.87        58
           1       0.64      0.82      0.72        22

    accuracy                           0.82        80
   macro avg       0.78      0.82      0.80        80
weighted avg       0.85      0.82      0.83        80
```

Izlaz 13: Kategorički naivni Bajes sa optimizacijom hiperparametara

Tačnost koja je ukupna dobijena je 81%. Preciznost 88% za vrednost klase 0 i 65% za vrednost klase 1, senzitivnost 86% za vrednost klase 0 i 68% za vrednost klase 1, F1-score 87% za vrednost klase 0 i 67% za vrednost klase 1.

```
Categorical Naive Bayes prediction without optimization:

              precision    recall  f1-score   support
           0       0.88      0.86      0.87        58
           1       0.65      0.68      0.67        22

    accuracy                           0.81        80
   macro avg       0.76      0.77      0.77        80
weighted avg       0.82      0.81      0.81        80
```

Izlaz 14: Kategorički naivni Bajes bez optimizacijom hiperparametara

Gausov naivni Bajes se pokazao kao bolji Naivni Bajes klasifikator, pa tako da se samo posmatra matrica konfuzije za njega.

3.4.5 Komplementarni naivni Bajes

U navedenom izlazu 15. data je analiza evaluacije klasifikatora optimizovanog hiperparametrima koji ishoduju najbolji slučaj: `alpha=0.1`

Tačnost koja je ukupna dobijena je 57%, **što je najgori ishod nekog klasifikatora zasnovanog na Naivnom Bajesu, a i u celom radu**. Preciznost 68% za vrednost klase 0 i 46% za vrednost klase 1, senzitivnost 57% za vrednost klase 0 i 58% za vrednost klase 1, F1-score 62% za vrednost klase 0 i 51% za vrednost klase 1.

```
    Complement Naive Bayes prediction:
    Best case:
    ComplementNB(alpha=0.1)

                  precision    recall  f1-score   support
               0       0.68      0.57      0.62        49
               1       0.46      0.58      0.51        31

        accuracy                           0.57        80
       macro avg       0.57      0.58      0.57        80
    weighted avg       0.60      0.57      0.58        80
```

Izlaz 15: Komplementarni naivni Bajes sa optimizacijom hiperparametara

Za izlaz 16. tačnost koja je ukupna dobijena je 57%. Preciznost 68% za vrednost klase 0 i 46% za vrednost klase 1, senzitivnost 57% za vrednost klase 0 i 58% za vrednost klase 1, F1-score 62% za vrednost klase 0 i 51% za vrednost klase 1.

```
Complement Naive Bayes prediction without optimization:

                  precision    recall  f1-score   support

               0       0.68      0.57      0.62        49
               1       0.46      0.58      0.51        31

        accuracy                           0.57        80
       macro avg       0.57      0.58      0.57        80
    weighted avg       0.60      0.57      0.58        80
```

Izlaz 16: Komplementarni naivni Bajes bez optimizacije hiperparametara

Gausov naivni Bajes se pokazao kao bolji Naivni Bajes klasifikator, pa tako da se samo posmatra matrica konfuzije za njega.

3.5 Mašine potpornih vektora (SVM)

U navedenom izlazu 19. data je analiza evaluacije klasifikatora optimizovanog hiperparametrima koji ishoduju najbolji slučaj: `C=0.1, gamma=1, kernel='linear'`

Tačnost koja je ukupna dobijena je 85%. Preciznost 94% za vrednost klase 0 i 68% za vrednost klase 1, senzitivnost 84% za vrednost klase 0 i 86% za vrednost klase 1, F1-score 89% za vrednost klase 0 i 76% za vrednost klase 1.

```
SVM prediction:
Best case:
SVC(C=0.1, gamma=1, kernel='linear')

              precision    recall  f1-score   support
           0       0.94      0.84      0.89        58
           1       0.68      0.86      0.76        22

    accuracy                           0.85        80
   macro avg       0.81      0.85      0.83        80
weighted avg       0.87      0.85      0.85        80
```

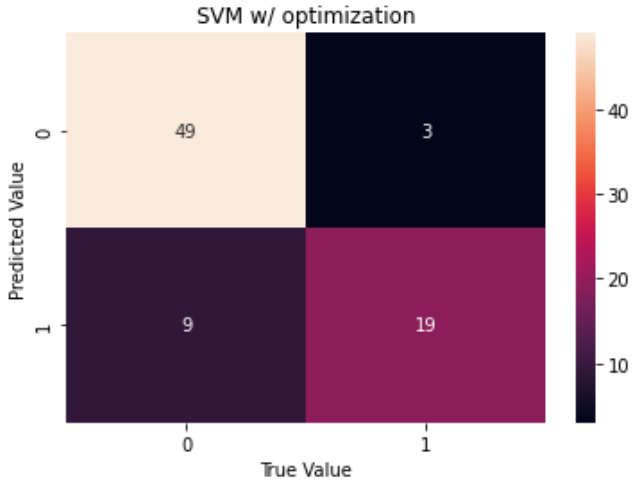
Izlaz 17: SVM sa optimizacijom hiperparametara

Za izlaz 20. tačnost koja je ukupna dobijena je 81%. Preciznost 82% za vrednost klase 0 i 77% za vrednost klase 1, senzitivnost 95% za vrednost klase 0 i 45% za vrednost klase 1, F1-score 88% za vrednost klase 0 i 57% za vrednost klase 1.

```
SVM prediction without optimization:

              precision    recall  f1-score   support
           0       0.82      0.95      0.88        58
           1       0.77      0.45      0.57        22

    accuracy                           0.81        80
   macro avg       0.80      0.70      0.73        80
weighted avg       0.81      0.81      0.80        80
```

Izlaz 18: SVM bez optimizacije hiperparametara

S obzirom da je izlaz 19. bolji od izlaza 20. daje se prikaz matrice konfuzije za njega na slici 8.



Slika 8: Matrica konfuzije za SVM sa optimizacijom hiperparametrima

3.6 K-Najbližih suseda (kNN)

U navedenom izlazu 21. data je analiza evaluacije klasifikatora optimizovanog hiperparametrima koji ishoduju najbolji slučaj:

`leaf_size=20, metric='chebyshev', n_neighbors=1, p=1`

Tačnost koja je ukupna dobijena je 86%. Preciznost 94% za vrednost klase 0 i 70% za vrednost klase 1, senzitivnost 86% za vrednost klase 0 i 86% za vrednost klase 1, F1-score 90% za vrednost klase 0 i 78% za vrednost klase 1.

```
KNN prediction:
Best case:
KNeighborsClassifier(leaf_size=20, metric='chebyshev', n_neighbors=1, p=1)

              precision    recall  f1-score   support
           0       0.94      0.86      0.90        58
           1       0.70      0.86      0.78        22

    accuracy                           0.86        80
```

```
   macro avg       0.82      0.86      0.84        80
weighted avg       0.88      0.86      0.87        80
```

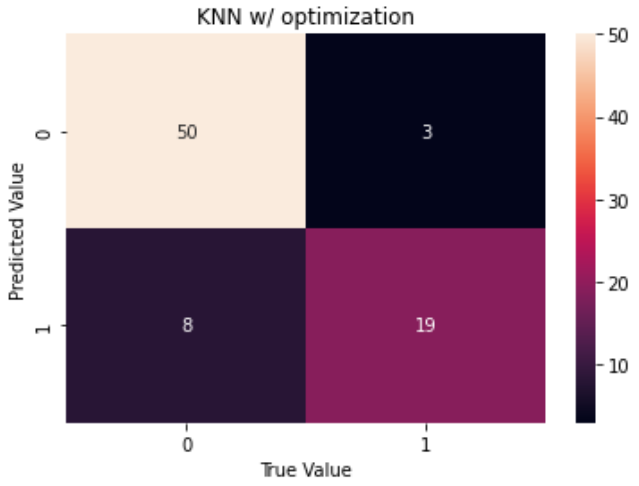
Izlaz 19: kNN sa optimizacijom hiperparametara

Za izlaz 22. tačnost koja je ukupna dobijena je 86%. Preciznost 93% za vrednost klase 0 i 72% za vrednost klase 1, senzitivnost 88% za vrednost klase 0 i 82% za vrednost klase 1, F1-score 90% za vrednost klase 0 i 77% za vrednost klase 1.

```
KNN prediction without optimization:

              precision    recall  f1-score   support
           0       0.93      0.88      0.90        58
           1       0.72      0.82      0.77        22

    accuracy                           0.86        80
   macro avg       0.82      0.85      0.83        80
weighted avg       0.87      0.86      0.87        80
```

Izlaz 20: kNN bez optimizacije hiperparametara

S obzirom da je izlaz 21. bolji od izlaza 22. daje se prikaz matrice konfuzije za njega na slici 9.



Slika 9: Matrica konfuzije za kNN sa optimizacijom hiperparametrima

4 Zaključak

Predstavljeni slučajevi tačnosti za svaki model klasifikacije na tabeli 1. može se uočiti da je najbolji model Gausov naivni Bajes. Sa druge strane, je najgori komplementarni Naivni Bajes.

Takođe je moguće uočiti da su optimizacije hiperparametrima bile beznačajne kod modela slučajnih šuma, Gausovog, multinomijalnog, Bernulijevog, komplementarnog naivnog Bajesa.

Klasifikator	**Tačnost sa GridSearchCV**	**Tačnost bez GCV**
Logistička Regresija	88%	73%
Stablo odlučivanja	88%	73%
Slučajne šume	89%	89%
Gaus naivni Bajes	90%	90%
Multinomijani naivni Bajes	73%	73%
Bernulijev naivni Bajes	73%	73%
Kategorički naivni Bajes	82%	81%
Komplementarni naivni Bajes	57%	57%
SVM	85%	81%
KNN	86%	86%

Tabela 1: Prikaz tačnosti klasifikatora sa i bez primene optimizacije hiperparametara.

Literatura

[1] Social Network Ads, `https://www.kaggle.com/datasets/akram24/social-network-ads`, Datum poslednjeg pristupa: 3. april 2024.

[2] Ž. Simić, (2024), "Teorijski izveštaj o klasifikacionim algoritmima sa nadgledanim učenjem", Univerzitet u Kragujevcu

[3] DataFrame, `https://pandas.pydata.org/docs/reference/frame.html`, Datum poslednjeg pristupa: 3. april 2024.

[4] sklearn.preprocessing.LabelEncoder, `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html`, Datum poslednjeg pristupa: 3. april 2024.

[5] sklearn.model_selection.train_test_split, `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`, Datum poslednjeg pristupa: 3. april 2024.

[6] seaborn.pairplot, `https://seaborn.pydata.org/generated/seaborn.pairplot.html`, Datum poslednjeg pristupa: 3. april 2024.

[7] seaborn.displot, `https://seaborn.pydata.org/generated/seaborn.displot.html`, Datum poslednjeg pristupa: 3. april 2024.

[8] sklearn.linear_model.LogisticRegression, `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression`, Datum poslednjeg pristupa: 3. april 2024.

[9] sklearn.model_selection.GridSearchCV, `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`, Datum poslednjeg pristupa: 3. april 2024.

[10] numpy.logspace, `https://numpy.org/doc/stable/reference/generated/numpy.logspace.html`, Datum poslednjeg pristupa: 3. april 2024.

[11] sklearn.tree.DecisionTreeClassifier, `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier`, Datum poslednjeg pristupa: 3. april 2024.

[12] sklearn.ensemble.RandomForestClassifier, `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`, Datum poslednjeg pristupa: 3. april 2024.

[13] sklearn.naive_bayes.GaussianNB, `https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html#sklearn.naive_bayes.GaussianNB`, Datum poslednjeg pristupa: 3. april 2024.

[14] sklearn.naive_bayes.MultinomialNB, `https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html`, Datum poslednjeg pristupa: 3. april 2024.

[15] sklearn.naive_bayes.BernoulliNB, `https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html`, Datum poslednjeg pristupa: 3. april 2024.

[16] sklearn.naive_bayes.ComplementNB, `https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.ComplementNB.html`, Datum poslednjeg pristupa: 3. april 2024.

[17] sklearn.naive_bayes.CategoricalNB, `https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.CategoricalNB.html`, Datum poslednjeg pristupa: 3. april 2024.

[18] sklearn.svm.SVC, `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC`, Datum poslednjeg pristupa: 3. april 2024.

[19] sklearn.neighbors.KNeighborsClassifier, `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html`, Datum poslednjeg pristupa: 3. april 2024.

[20] Machine Learning Model Evaluation Metrics part 2: Multi-class classification, `https://www.mariakhalusova.com/posts/2019-04-17-ml-model-evaluation-metrics-p2/`, Datum poslednjeg pristupa: 3. april 2024.

[21] Classification metrics: Multiclass and multilabel classification, `https://scikit-learn.org/stable/modules/model_evaluation.html#multiclass-and-multilabel-classification`, Datum poslednjeg pristupa: 3. april 2024.