

Fakultet inženjerskih nauka
Univerziteta u Kragujevcu

Tema:

*Klasifikacioni algoritmi sa nadgledanim učenjem na primeru Social
Network Ads*

student:
Nikola Mitrevski
400/2021

predmetni profesor:
dr Nenad Filipović
predmetni asistent:
Tijana Šušteršić

Kragujevac 2021.

Sadržaj:

1	Uvod	3
2	Logistic Regression	4
2.1	Uvoz potrebnih biblioteka i SNA skupa podataka	4
2.2	Podela podataka na nezavisne promenljive i zavisnu promenljivu.....	4
2.3	Podela podataka na skup za treniranje i na skup za testiranje	4
2.4	Normalizacija podataka	5
2.5	Treniranje i testiranje modela	6
2.6	Kreiranje konfuzione matrice	7
2.7	Metrike senzitivnosti i specifičnosti	7
2.8	Vizuelizacija rezultata trening skupa	8
2.9	Vizuelizacija rezultata test skupa	9
3	Decision Tree	10
3.1	Uvoz potrebnih biblioteka i SNA skupa podataka	10
3.2	Podela podataka na nezavisne promenljive i zavisnu promenljivu.....	10
3.3	Podela podataka na skup za treniranje i na skup za testiranje	10
3.4	Normalizacija podataka	11
3.5	Treniranje i testiranje modela	11
3.6	Kreiranje konfuzione matrice	12
3.7	Metrike senzitivnosti i specifičnosti	12
3.8	Vizuelizacija rezultata trening skupa	13
3.9	Vizuelizacija rezultata test skupa	14
4	Random Forest	15
4.1	Uvoz potrebnih biblioteka i SNA skupa podataka	15
4.2	Podela podataka na nezavisne promenljive i zavisnu promenljivu.....	15
4.3	Podela podataka na skup za treniranje i na skup za testiranje	15
4.4	Normalizacija podataka	16
4.5	Treniranje i testiranje modela	16
4.6	Kreiranje konfuzione matrice	17
4.1	Metrike senzitivnosti i specifičnosti	18
4.2	Vizuelizacija rezultata trening skupa	19
4.3	Vizuelizacija rezultata test skupa	20
5	Naive Bayes Classifier	21
5.1	Uvoz potrebnih biblioteka i SNA skupa podataka	21
5.2	Podela podataka na nezavisne promenljive i zavisnu promenljivu.....	21

5.3	Podela podataka na skup za treniranje i na skup za testiranje	21
5.4	Normalizacija podataka	22
5.5	Treniranje i testiranje modela	22
5.6	Kreiranje konfuzione matrice	23
5.7	Metrike senzitivnosti i specifičnosti	23
5.8	Vizuelizacija rezultata trening skupa	24
5.9	Vizuelizacija rezultata test skupa	25
6	Support Vector Machine	26
6.1	Uvoz potrebnih biblioteka i SNA skupa podataka	26
6.2	Podela podataka na nezavisne promenljive i zavisnu promenljivu.....	26
6.3	Podela podataka na skup za treniranje i na skup za testiranje	26
6.4	Normalizacija podataka	27
6.5	Treniranje i testiranje modela	27
6.6	Kreiranje konfuzione matrice	28
6.7	Metrike senzitivnosti i specifičnosti	28
6.8	Vizuelizacija rezultata trening skupa	29
6.9	Vizuelizacija rezultata test skupa	30
7	K Nearest Neighbour	31
7.1	Uvoz potrebnih biblioteka i SNA skupa podataka	31
7.2	Podela podataka na nezavisne promenljive i zavisnu promenljivu.....	31
7.3	Podela podataka na skup za treniranje i na skup za testiranje	31
7.4	Normalizacija podataka	32
7.5	Treniranje i testiranje modela	32
7.6	Kreiranje konfuzione matrice	33
7.7	Metrike senzitivnosti i specifičnosti	33
7.8	Vizuelizacija rezultata trening skupa	34
7.9	Vizuelizacija rezultata test skupa	35
8	Literatura	36

1 Uvod

U ovom radu biće reči o nekoliko klasifikacionih algoritama(Logistic Regression, Decision Tree, Random Forest, Naive Bayes Classifier, Support Vector Machine i K Nearest Neighbour) koji su primenjeni na Social Network Ads(SNA) skupu podataka.

Cilj klasifikacije nad ovim skupom podataka je određivanje ciljne publike za konkretan oglas, odnosno maksimiziranje stope učestalosti klikova na oglas.

SNA skup podataka sadrži neke informacije o korisnicima(User ID, Gender, Age, EstimatedSalary, Purchased) neke društvene mreže.

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

Slika 1 Prikaz prvih pet reda iz SNA dataset-a

Kolona purchased(kupljeno) je vektor logičkih vrednosti koji opisuje da li je svaki pojedinac iz skupa podataka na kraju klinuo na oglas.

2 Logistic Regression

2.1 Uvoz potrebnih biblioteka i SNA skupa podataka

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('../input/Social_Network_Ads.csv')
```

2.2 Podela podataka na nezavisne promenljive i zavisnu promenljivu

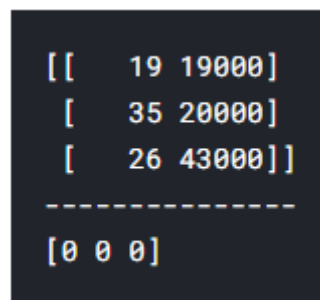
SNA skup podataka je potrebno podeliti na nezavisne promenljive(X) i zavisnu promenljivu(y), kako bi se mogao odrediti efekat nezavisnih promenljivih na zavisnu promenljivu.

Sledeće linije koda dele SNA skup podataka na nezavisne promenljive(3 i 4 kolona(2 i 3 indeks)) i zavisnu promenljivu(5 kolona(4 indeks)):

```
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

Sledeće linije koda štampaju prva tri reda iz X i y skupa:

```
print(X[:3, :])
print('-'*15)
print(y[:3])
```



```
[[ 19 19000]
 [ 35 20000]
 [ 26 43000]]
-----
[0 0 0]
```

Slika 2 Prikaz prva tri reda iz X i y skupa

2.3 Podela podataka na skup za treniranje i na skup za testiranje

Da bi se model mogao naučiti, potrebno je podeliti podatke na skup podataka za treniranje i na skup podataka za testiranje modela.

Sledeće linije koda dele podatke na trening skup i na test skup:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.25, random_state = 0)
```

Sledeće linije koda štampaju prva tri reda iz novodobijenih skupova:

```
print(X_train[:3])
print('-'*15)
print(y_train[:3])
print('-'*15)
print(X_test[:3])
print('-'*15)
print(y_test[:3])
```

```

[[ 44 39000]
 [ 32 120000]
 [ 38 50000]]
-----
[0 1 0]
-----
[[ 30 87000]
 [ 38 50000]
 [ 35 75000]]
-----
[0 0 0]

```

Slika 3 Prikaz prva tri reda iz trening i test skupa

2.4 Normalizacija podataka

Da bi se dobili najtačniji rezultati, potrebno je primeniti normalizaciju podataka (skaliranje karakteristika).

Sledeće linije koda normalizuju podatke iz trening i test skupa:

```

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

```

Sledeće linije koda štampaju prva tri reda iz trening i test skupa, nakon normalizacije podataka:

```

print(X_train[:3])
print('-'*15)
print(X_test[:3])

```

```

[[ 0.58164944 -0.88670699]
 [-0.60673761  1.46173768]
 [-0.01254409 -0.5677824  ]]
-----
[[-0.80480212  0.50496393]
 [-0.01254409 -0.5677824  ]
 [-0.30964085  0.1570462  ]]

```

Slika 4 Prikaz prva tri reda iz trening i test skupa, nakon normalizacije podataka

2.5 Treniranje i testiranje modela

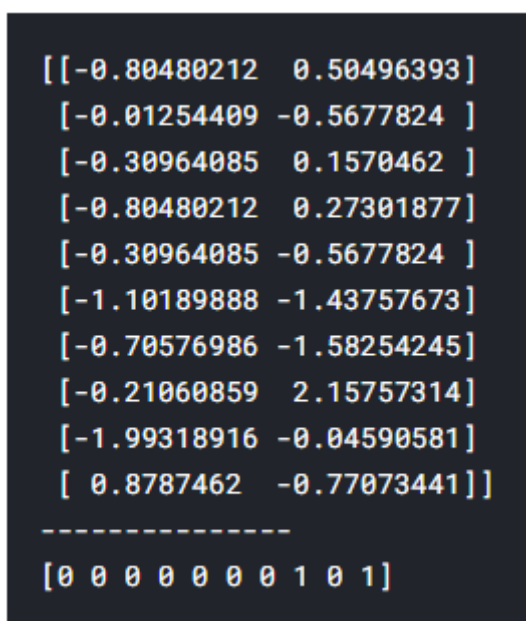
Sledeće linije koda kreiraju, treniraju i testiraju objekat klase LogisticRegression:

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0, solver='lbfgs' )
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

Konstruktoru klase LogisticRegression je prosleđen argument random_state s vrednošću 0, da bi se uvek dobili isti rezultati.

Sledeće linije koda štampaju prvih deset redova nakon predviđanja modela:

```
print(X_test[:10])
print('-'*15)
print(y_pred[:10])
```

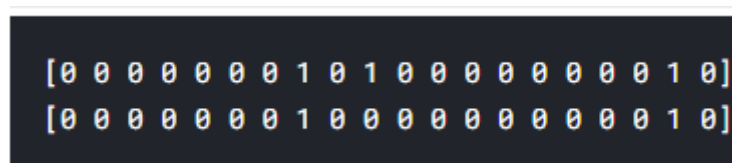


```
[[-0.80480212  0.50496393]
 [-0.01254409 -0.5677824 ]
 [-0.30964085  0.1570462 ]
 [-0.80480212  0.27301877]
 [-0.30964085 -0.5677824 ]
 [-1.10189888 -1.43757673]
 [-0.70576986 -1.58254245]
 [-0.21060859  2.15757314]
 [-1.99318916 -0.04590581]
 [ 0.8787462  -0.77073441]]
-----
[0 0 0 0 0 0 0 1 0 1]
```

Slika 5 Prikaz prvih deset redova nakon predviđanja modela

Sledeće linije koda štampaju prvih dvadeset redova željenih i stvarnih izlaza:

```
print(y_pred[:20])
print(y_test[:20])
```



```
[0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0]
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0]
```

Slika 6 Prikaz prvih dvadeset redova željenih i stvarnih izlaza

Sa slike vidimo da dolazi do određene greške prilikom predviđanja modela.

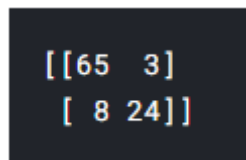
2.6 Kreiranje konfuzione matrice

Sledeće linije koda kreiraju konfuzionu matricu:

```
from sklearn.metrics import confusion_matrix  
cm = confusion_matrix(y_test, y_pred)
```

Sledeće linije koda štampaju konfuzionu matricu:

```
print(cm)
```



```
[[65  3]  
 [ 8 24]]
```

Slika 7 Prikaz konfuzione matrice

Sa slike se vidi da je 89 tačnih predviđanja i 11 netačnih predviđanja, što znači da je model postigao ocenu tačnosti od 89%.

2.7 Metrike senzitivnosti i specifičnosti

Sledeće linije koda izračunavaju senzitivnost i specifičnost:

```
#####from confusion matrix calculate sensitivity and specificity  
sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])  
specificity = cm[1,1]/(cm[1,0]+cm[1,1])
```

Sledeće linije koda štampaju vrednosti za senzitivnost i specifičnost:

```
print('Sensitivity : ', sensitivity)  
print('Specificity : ', specificity)
```

```
Sensitivity : 0.9558823529411765  
Specificity : 0.75
```

Slika 8 Prikaz senzitivnosti i specifičnosti

2.8 Vizuelizacija rezultata trening skupa

Sledeće linije koda vizuelizuju rezultat trening skupa:

```
# Visualizing the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.6, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
               c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



Slika 9 Grafički prikaz rezultata trening skupa

Sa grafika se vidi da kada se starost i procenjena plata povećavaju, svaki pojedinac ima veću verovatnoću da će biti zelen(klikne na oglas).

2.9 Vizuelizacija rezultata test skupa

Sledeće linije koda vizuelizuju rezultat test skupa:

```
# Visualizing the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.6, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



Slika 10 Grafički prikaz rezultata test skupa

3 Decision Tree

3.1 Uvoz potrebnih biblioteka i SNA skupa podataka

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('../input/Social_Network_Ads.csv')
```

3.2 Podela podataka na nezavisne promenljive i zavisnu promenljivu

SNA skup podataka je potrebno podeliti na nezavisne promenljive(X) i zavisnu promenljivu(y), kako bi se mogao odrediti efekat nezavisnih promenljivih na zavisnu promenljivu.

Sledeće linije koda dele SNA skup podataka na nezavisne promenljive(3 i 4 kolona(2 i 3 indeks)) i zavisnu promenljivu(5 kolona(4 indeks)):

```
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

Sledeće linije koda štampaju prva tri reda iz X i y skupa:

```
print(X[:3, :])
print('-'*15)
print(y[:3])
```

```
[[ 19 19000]
 [ 35 20000]
 [ 26 43000]]
-----
[0 0 0]
```

Slika 11 Prikaz prva tri reda iz X i y skupa

3.3 Podela podataka na skup za treniranje i na skup za testiranje

Da bi se model mogao naučiti, potrebno je podeliti podatke na skup podataka za treniranje i na skup podataka za testiranje modela.

Sledeće linije koda dele podatke na trening skup i na test skup:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.25, random_state = 0)
```

Sledeće linije koda štampaju prva tri reda iz novodobijenih skupova:

```
print(X_train[:3])
print('-'*15)
print(y_train[:3])
print('-'*15)
print(X_test[:3])
print('-'*15)
print(y_test[:3])
```

```

[[ 44 39000]
 [ 32 120000]
 [ 38 50000]]
-----
[0 1 0]
-----
[[ 30 87000]
 [ 38 50000]
 [ 35 75000]]
-----
[0 0 0]

```

Slika 12 Prikaz prva tri reda iz trening i test skupa

3.4 Normalizacija podataka

Da bi se dobili najtačniji rezultati, potrebno je primeniti normalizaciju podataka (skaliranje karakteristika).

Sledeće linije koda normalizuju podatke iz trening i test skupa:

```

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

```

Sledeće linije koda štampaju prva tri reda iz trening i test skupa, nakon normalizacije podataka:

```

print(X_train[:3])
print('-'*15)
print(X_test[:3])

```

```

[[ 0.58164944 -0.88670699]
 [-0.60673761  1.46173768]
 [-0.01254409 -0.5677824 ]]
-----
[[-0.80480212  0.50496393]
 [-0.01254409 -0.5677824 ]
 [-0.30964085  0.1570462 ]]

```

Slika 13 Prikaz prva tri reda iz trening i test skupa, nakon normalizacije podataka

3.5 Treniranje i testiranje modela

Sledeće linije koda kreiraju, treniraju i testiraju objekat klase DecisionTreeClassifier:

```

from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion='entropy', random_state=0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

```

Konstruktoru klase DecisionTreeClassifier je prosleđen argument random_state s vrednošću 0, da bi se uvek dobili isti rezultati.

Sledeće linije koda štampaju prvih deset redova nakon predviđanja modela:

```

print(X_test[:10])
print('-'*15)
print(y_pred[:10])

```

```

[[-0.80480212  0.50496393]
 [-0.01254409 -0.5677824 ]
 [-0.30964085  0.1570462 ]
 [-0.80480212  0.27301877]
 [-0.30964085 -0.5677824 ]
 [-1.10189888 -1.43757673]
 [-0.70576986 -1.58254245]
 [-0.21060859  2.15757314]
 [-1.99318916 -0.04590581]
 [ 0.8787462  -0.77073441]]
-----
[0 0 0 0 0 0 0 1 0 0]

```

Slika 14 Prikaz prvih deset redova nakon predviđanja modela

Sledeće linije koda štampaju prvih dvadeset redova željenih i stvarnih izlaza:

```

print(y_pred[:20])
print(y_test[:20])

```

```

[0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 0 1 0]
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0]

```

Slika 15 Prikaz prvih dvadeset redova željenih i stvarnih izlaza

Sa slike vidimo da dolazi do određene greške prilikom predviđanja modela.

3.6 Kreiranje konfuzione matrice

Sledeće linije koda kreiraju konfuzionu matricu:

```

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

```

Sledeće linije koda štampaju konfuzionu matricu:

```

print(cm)

```

```

array([[62,  6],
       [ 3, 29]])

```

Slika 16 Prikaz konfuzione matrice

Sa slike se vidi da je 91 tačnih predviđanja i 9 netačnih predviđanja, što znači da je model postigao ocenu tačnosti od 91%.

3.7 Metrike senzitivnosti i specifičnosti

Sledeće linije koda izračunavaju senzitivnost i specifičnost:

```

#####from confusion matrix calculate sensitivity and specificity
sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
specificity = cm[1,1]/(cm[1,0]+cm[1,1])

```

Sledeće linije koda štampaju vrednosti za senzitivnost i specifičnost:

```

print('Sensitivity : ', sensitivity)
print('Specificity : ', specificity)

```

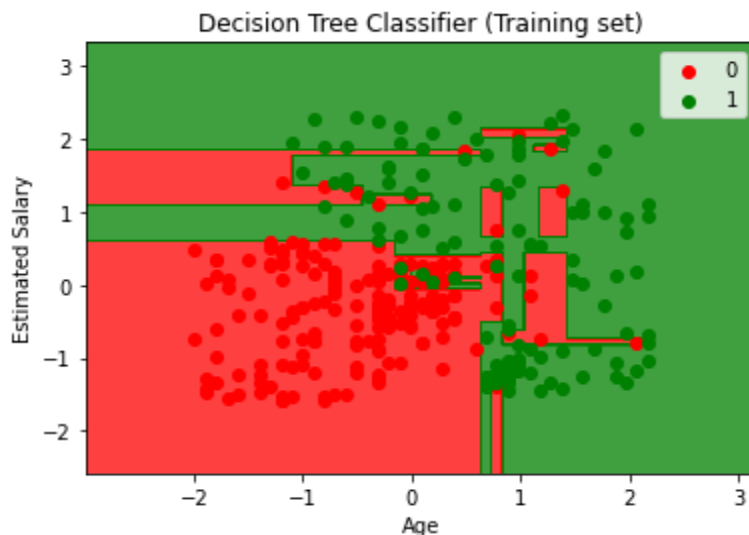
Sensitivity : 0.9117647058823529
Specificity : 0.90625

Slika 17 Prikaz senzitivnosti i specifičnosti

3.8 Vizuelizacija rezultata trening skupa

Sledeće linije koda vizuelizuju rezultat trening skupa:

```
# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
               c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Decision Tree Classifier (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



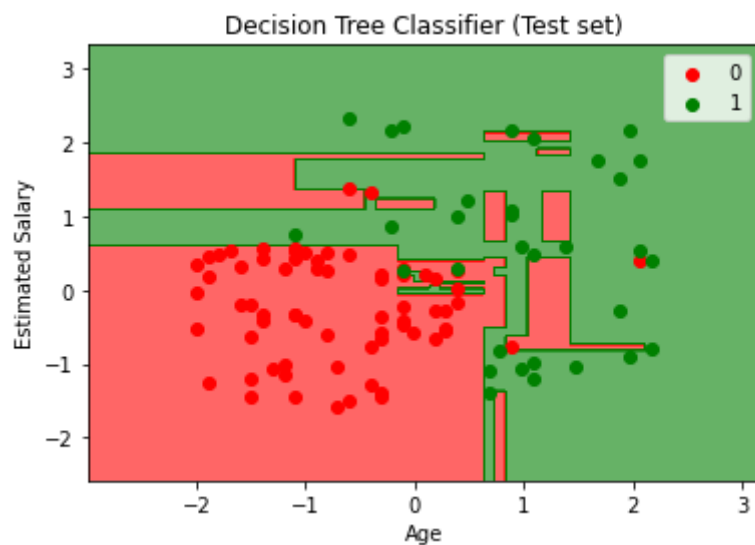
Slika 18 Grafički prikaz rezultata trening skupa

Sa grafika se vidi da kada se starost i procenjena plata povećavaju, svaki pojedinac ima veću verovatnoću da će biti zelen(klikne na oglas).

3.9 Vizuelizacija rezultata test skupa

Sledeće linije koda vizuelizuju rezultat test skupa:

```
# Visualizing the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                      np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.6, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
               c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Logistic Regression (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



Slika 19 Grafički prikaz rezultata test skupa

4 Random Forest

4.1 Uvoz potrebnih biblioteka i SNA skupa podataka

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('../input/Social_Network_Ads.csv')
```

4.2 Podela podataka na nezavisne promenljive i zavisnu promenljivu

SNA skup podataka je potrebno podeliti na nezavisne promenljive(X) i zavisnu promenljivu(y), kako bi se mogao odrediti efekat nezavisnih promenljivih na zavisnu promenljivu.

Sledeće linije koda dele SNA skup podataka na nezavisne promenljive(3 i 4 kolona(2 i 3 indeks)) i zavisnu promenljivu(5 kolona(4 indeks)):

```
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

Sledeće linije koda štampaju prve tri vrste iz X i y skupa:

```
print(X[:3, :])
print('-'*15)
print(y[:3])
```

```
[[ 19 19000]
 [ 35 20000]
 [ 26 43000]]
-----
[0 0 0]
```

Slika 20 Prikaz prva tri reda iz X i y skupa

4.3 Podela podataka na skup za treniranje i na skup za testiranje

Da bi se model mogao naučiti, potrebno je podeliti podatke na skup podataka za treniranje i na skup podataka za testiranje modela.

Sledeće linije koda dele podatke na trening skup i na test skup:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.25, random_state = 0)
```

Sledeće linije koda štampaju prva tri reda iz novodobijenih skupova:

```
print(X_train[:3])
print('-'*15)
print(y_train[:3])
print('-'*15)
print(X_test[:3])
print('-'*15)
print(y_test[:3])
```



```

[[ 44 39000]
 [ 32 120000]
 [ 38 50000]]
-----
[0 1 0]
-----
[[ 30 87000]
 [ 38 50000]
 [ 35 75000]]
-----
[0 0 0]

```

Slika 21 Prikaz prva tri reda iz trening i test skupa

4.4 Normalizacija podataka

Da bi se dobili najtačniji rezultati, potrebno je primeniti normalizaciju podataka (skaliranje karakteristika).

Sledeće linije koda normalizuju podatke iz trening i test skupa:

```

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

```

Sledeće linije koda štampaju prva tri reda iz trening i test skupa, nakon normalizacije podataka:

```

print(X_train[:3])
print('-'*15)
print(X_test[:3])

```

```

[[ 0.58164944 -0.88670699]
 [-0.60673761  1.46173768]
 [-0.01254409 -0.5677824  ]]
-----
[[-0.80480212  0.50496393]
 [-0.01254409 -0.5677824  ]
 [-0.30964085  0.1570462  ]]

```

Slika 22 Prikaz prva tri reda iz trening i test skupa, nakon normalizacije podataka

4.5 Treniranje i testiranje modela

Sledeće linije koda kreiraju, treniraju i testiraju objekat klase RandomForestClassifier:

```

from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion =
'entropy', random_state = 0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

```

Konstruktoru klase RandomForestClassifier je prosleđen argument random_state s vrednošću 0, da bi se uvek dobili isti rezultati.

Sledeće linije koda štampaju prvih deset redova nakon predviđanja modela:

```

print(X_test[:10])
print('-'*15)

```

```
print(y_pred[:10])
```

```
[[-0.80480212  0.50496393]
 [-0.01254409 -0.5677824 ]
 [-0.30964085  0.1570462 ]
 [-0.80480212  0.27301877]
 [-0.30964085 -0.5677824 ]
 [-1.10189888 -1.43757673]
 [-0.70576986 -1.58254245]
 [-0.21060859  2.15757314]
 [-1.99318916 -0.04590581]
 [ 0.8787462  -0.77073441]]
-----
[0 0 0 0 0 0 0 1 0 1]
```

Slika 23 Prikaz prvih deset redova nakon predviđanja modela

Sledeće linije koda štampaju prvih dvadeset redova željenih i stvarnih izlaza:

```
print(y_pred[:20])
print(y_test[:20])
```

```
[0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 1 0 1 0]
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0]
```

Slika 24 Prikaz prvih dvadeset redova željenih i stvarnih izlaza

Sa slike vidimo da dolazi do određene greške prilikom predviđanja modela.

4.6 Kreiranje konfuzione matrice

Sledeće linije koda kreiraju konfuzionu matricu:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

Sledeće linije koda štampaju konfuzionu matricu:

```
print(cm)
```

```
[[63  5]
 [ 4 28]]
```

Slika 25 Prikaz konfuzione matrice

Sa slike se vidi da je 91 tačnih predviđanja i 9 netačnih predviđanja, što znači da je model postigao ocenu tačnosti od 91%.

4.1 Metrike senzitivnosti i specifičnosti

Sledeće linije koda izračunavaju senzitivnost i specifičnost:

```
#####from confusion matrix calculate sensitivity and specificity  
sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])  
specificity = cm[1,1]/(cm[1,0]+cm[1,1])
```

Sledeće linije koda štampaju vrednosti za senzitivnost i specifičnost:

```
print('Sensitivity : ', sensitivity)  
print('Specificity : ', specificity)
```

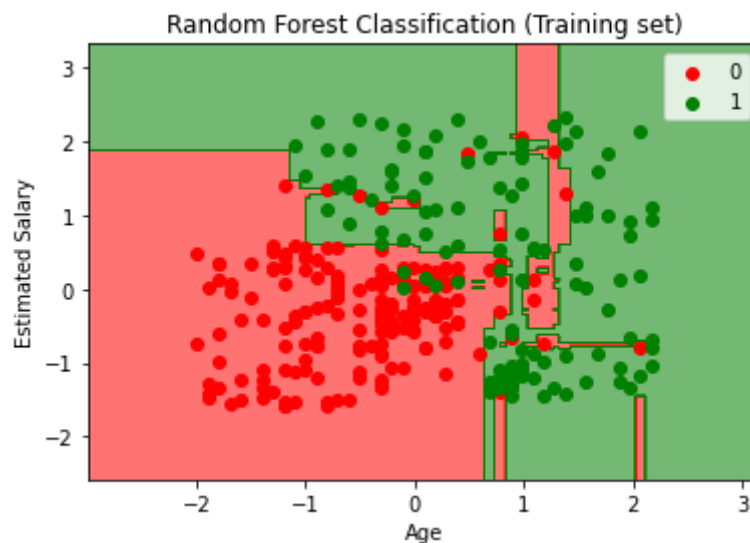
```
Sensitivity : 0.9264705882352942  
Specificity : 0.875
```

Slika 26 Prikaz senzitivnosti i specifičnosti

4.2 Vizuelizacija rezultata trening skupa

Sledeće linije koda vizuelizuju rezultat trening skupa:

```
# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                      np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.55, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Random Forest Classification (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



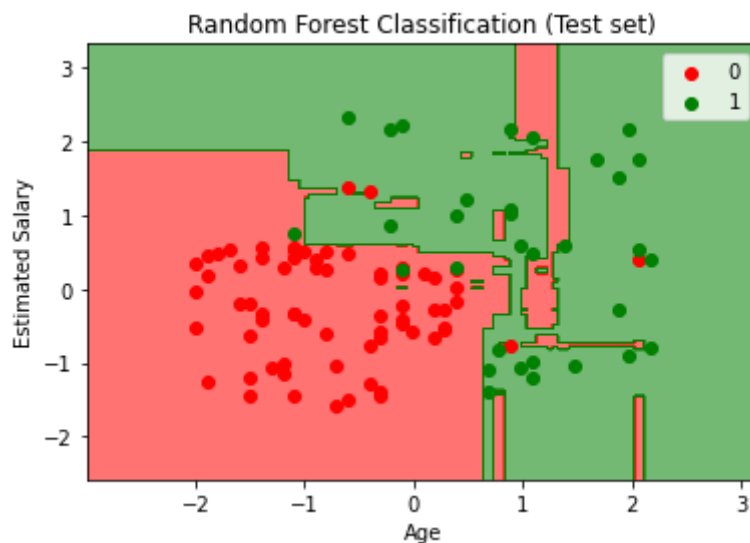
Slika 27 Grafički prikaz rezultata trening skupa

Sa grafika se vidi da kada se starost i procenjena plata povećavaju, svaki pojedinac ima veću verovatnoću da će biti zelen (klikne na oglas).

4.3 Vizuelizacija rezultata test skupa

Sledeće linije koda vizuelizuju rezultat test skupa:

```
# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.55, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Random Forest Classification (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



Slika 28 Grafički prikaz rezultata test skupa

5 Naive Bayes Classifier

5.1 Uvoz potrebnih biblioteka i SNA skupa podataka

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('../input/Social_Network_Ads.csv')
```

5.2 Podela podataka na nezavisne promenljive i zavisnu promenljivu

SNA skup podataka je potrebno podeliti na nezavisne promenljive(X) i zavisnu promenljivu(y), kako bi se mogao odrediti efekat nezavisnih promenljivih na zavisnu promenljivu.

Sledeće linije koda dele SNA skup podataka na nezavisne promenljive(3 i 4 kolona(2 i 3 indeks)) i zavisnu promenljivu(5 kolona(4 indeks)):

```
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

Sledeće linije koda štampaju prva tri reda iz X i y skupa:

```
print(X[:3, :])
print('-'*15)
print(y[:3])
```

```
[[ 19 19000]
 [ 35 20000]
 [ 26 43000]]
-----
[0 0 0]
```

Slika 29 Prikaz prva tri reda iz X i y skupa

5.3 Podela podataka na skup za treniranje i na skup za testiranje

Da bi se model mogao naučiti, potrebno je podeliti podatke na skup podataka za treniranje i na skup podataka za testiranje modela.

Sledeće linije koda dele podatke na trening skup i na test skup:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.25, random state = 0)
```

Sledeće linije koda štampaju prva tri reda iz novodobijenih skupova:

```
print(X_train[:3])
print('-'*15)
print(y_train[:3])
print('-'*15)
print(X_test[:3])
print('-'*15)
print(y_test[:3])
```

```

[[ 44 39000]
 [ 32 120000]
 [ 38 50000]]
-----
[0 1 0]
-----
[[ 30 87000]
 [ 38 50000]
 [ 35 75000]]
-----
[0 0 0]

```

Slika 30 Prikaz prva tri reda iz trening i test skupa

5.4 Normalizacija podataka

Da bi se dobili najtačniji rezultati, potrebno je primeniti normalizaciju podataka (skaliranje karakteristika).

Sledeće linije koda normalizuju podatke iz trening i test skupa:

```

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

```

Sledeće linije koda štampaju prva tri reda iz trening i test skupa, nakon normalizacije podataka:

```

print(X_train[:3])
print('-'*15)
print(X_test[:3])

```

```

[[ 0.58164944 -0.88670699]
 [-0.60673761  1.46173768]
 [-0.01254409 -0.5677824 ]]
-----
[[-0.80480212  0.50496393]
 [-0.01254409 -0.5677824 ]
 [-0.30964085  0.1570462 ]]

```

Slika 31 Prikaz prva tri reda iz trening i test skupa, nakon normalizacije podataka

5.5 Treniranje i testiranje modela

Sledeće linije koda kreiraju, treniraju i testiraju objekat klase GaussianNB:

```

from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

```

Sledeće linije koda štampaju prvih deset redova nakon predviđanja modela:

```

print(X_test[:10])
print('-'*15)
print(y_pred[:10])

```

```

[[-0.80480212  0.50496393]
 [-0.01254409 -0.5677824 ]
 [-0.30964085  0.1570462 ]
 [-0.80480212  0.27301877]
 [-0.30964085 -0.5677824 ]
 [-1.10189888 -1.43757673]
 [-0.70576986 -1.58254245]
 [-0.21060859  2.15757314]
 [-1.99318916 -0.04590581]
 [ 0.8787462  -0.77073441]]
-----
[0 0 0 0 0 0 0 1 0 1]

```

Slika 32 Prikaz prvih deset redova nakon predviđanja modela

Sledeće linije koda štampaju prvih dvadeset redova željenih i stvarnih izlaza:

```

print(y_pred[:20])
print(y_test[:20])

```

```

[0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0]
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0]

```

Slika 33 Prikaz prvih dvadeset redova željenih i stvarnih izlaza

Sa slike vidimo da dolazi do određene greške prilikom predviđanja modela.

5.6 Kreiranje konfuzione matrice

Sledeće linije koda kreiraju konfuzionu matricu:

```

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

```

Sledeće linije koda štampaju konfuzionu matricu:

```

print(cm)

```

```

[[65  3]
 [ 7 25]]

```

Slika 34 Prikaz konfuzione matrice

Sa slike se vidi da je 90 tačnih predviđanja i 10 netačnih predviđanja, što znači da je model postigao ocenu tačnosti od 90%.

5.7 Metrike senzitivnosti i specifičnosti

Sledeće linije koda izračunavaju senzitivnost i specifičnost:

```

#####from confusion matrix calculate sensitivity and specificity
sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
specificity = cm[1,1]/(cm[1,0]+cm[1,1])

```

Sledeće linije koda štampaju vrednosti za senzitivnost i specifičnost:

```

print('Sensitivity : ', sensitivity)
print('Specificity : ', specificity)

```

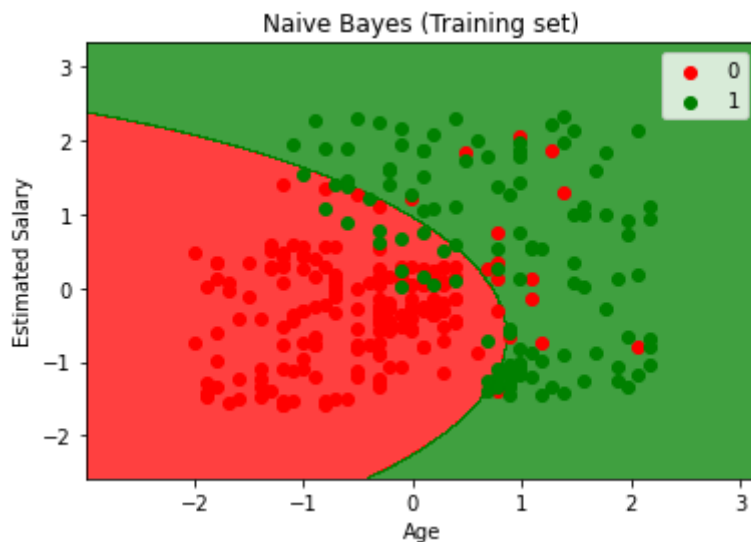

Sensitivity : 0.9558823529411765
Specificity : 0.78125

Slika 35 Prikaz senzitivnosti i specifičnosti

5.8 Vizuelizacija rezultata trening skupa

Sledeće linije koda vizuelizuju rezultat trening skupa:

```
# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Naive Bayes (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



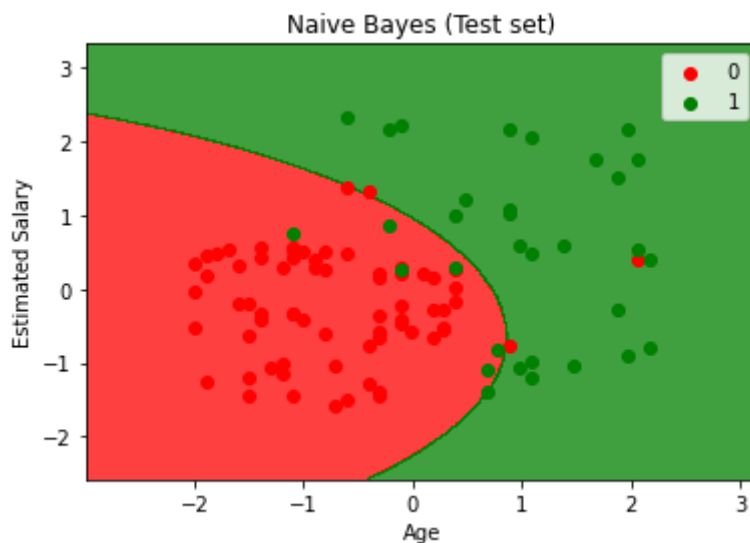
Slika 36 Grafički prikaz rezultata trening skupa

Sa grafika se vidi da kada se starost i procenjena plata povećavaju, svaki pojedinac ima veću verovatnoću da će biti zelen(klikne na oglas).

5.9 Vizuelizacija rezultata test skupa

Sledeće linije koda vizuelizuju rezultat test skupa:

```
# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Naive Bayes (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



Slika 37 Grafički prikaz rezultata test skupa

6 Support Vector Machine

6.1 Uvoz potrebnih biblioteka i SNA skupa podataka

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('../input/Social_Network_Ads.csv')
```

6.2 Podela podataka na nezavisne promenljive i zavisnu promenljivu

SNA skup podataka je potrebno podeliti na nezavisne promenljive(X) i zavisnu promenljivu(y), kako bi se mogao odrediti efekat nezavisnih promenljivih na zavisnu promenljivu.

Sledeće linije koda dele SNA skup podataka na nezavisne promenljive(3 i 4 kolona(2 i 3 indeks)) i zavisnu promenljivu(5 kolona(4 indeks)):

```
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

Sledeće linije koda štampaju prva tri reda iz X i y skupa:

```
print(X[:3, :])
print('-'*15)
print(y[:3])
```

```
[[ 19 19000]
 [ 35 20000]
 [ 26 43000]]
-----
[0 0 0]
```

Slika 38 Prikaz prva tri reda iz X i y skupa

6.3 Podela podataka na skup za treniranje i na skup za testiranje

Da bi se model mogao naučiti, potrebno je podeliti podatke na skup podataka za treniranje i na skup podataka za testiranje modela.

Sledeće linije koda dele podatke na trening skup i na test skup:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.25, random_state = 0)
```

Sledeće linije koda štampaju prva tri reda iz novodobijenih skupova:

```
print(X_train[:3])
print('-'*15)
print(y_train[:3])
print('-'*15)
print(X_test[:3])
print('-'*15)
print(y_test[:3])
```

```

[[ 44 39000]
 [ 32 120000]
 [ 38 50000]]
-----
[0 1 0]
-----
[[ 30 87000]
 [ 38 50000]
 [ 35 75000]]
-----
[0 0 0]

```

Slika 39 Prikaz prva tri reda iz trening i test skupa

6.4 Normalizacija podataka

Da bi se dobili najtačniji rezultati, potrebno je primeniti normalizaciju podataka (skaliranje karakteristika).

Sledeće linije koda normalizuju podatke iz trening i test skupa:

```

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

```

Sledeće linije koda štampaju prva tri reda iz trening i test skupa, nakon normalizacije podataka:

```

print(X_train[:3])
print('-'*15)
print(X_test[:3])

```

```

[[ 0.58164944 -0.88670699]
 [-0.60673761  1.46173768]
 [-0.01254409 -0.5677824  ]]
-----
[[-0.80480212  0.50496393]
 [-0.01254409 -0.5677824  ]
 [-0.30964085  0.1570462  ]]

```

Slika 40 Prikaz prva tri reda iz trening i test skupa, nakon normalizacije podataka

6.5 Treniranje i testiranje modela

Sledeće linije koda kreiraju, treniraju i testiraju objekat klase SVC:

```

# Fitting SVM to the Training set using Kernel as linear.
from sklearn.svm import SVC
classifier = SVC(kernel='linear', random_state=0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

```

Konstruktoru klase SVC je prosleđen argument `random_state` s vrednošću 0, da bi se uvek dobili isti rezultati.

Sledeće linije koda štampaju prvih deset redova nakon predviđanja modela:

```

print(X_test[:10])
print('-'*15)

```

```
print(y_pred[:10])
```

```
[[-0.80480212  0.50496393]
 [-0.01254409 -0.5677824 ]
 [-0.30964085  0.1570462 ]
 [-0.80480212  0.27301877]
 [-0.30964085 -0.5677824 ]
 [-1.10189888 -1.43757673]
 [-0.70576986 -1.58254245]
 [-0.21060859  2.15757314]
 [-1.99318916 -0.04590581]
 [ 0.8787462  -0.77073441]]
-----
[0 0 0 0 0 0 0 1 0 0]
```

Slika 41 Prikaz prvih deset redova nakon predviđanja modela

Sledeće linije koda štampaju prvih dvadeset redova željenih i stvarnih izlaza:

```
print(y_pred[:20])
print(y_test[:20])
```

```
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0]
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0]
```

Slika 42 Prikaz prvih dvadeset redova željenih i stvarnih izlaza

6.6 Kreiranje konfuzione matrice

Sledeće linije koda kreiraju konfuzionu matricu:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

Sledeće linije koda štampaju konfuzionu matricu:

```
print(cm)
```

```
[[66  2]
 [ 8 24]]
```

Slika 43 Prikaz konfuzione matrice

Sa slike se vidi da je 90 tačnih predviđanja i 10 netačnih predviđanja, što znači da je model postigao ocenu tačnosti od 90%.

6.7 Metrike senzitivnosti i specifičnosti

Sledeće linije koda izračunavaju senzitivnost i specifičnost:

```
#####from confusion matrix calculate sensitivity and specificity
sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
specificity = cm[1,1]/(cm[1,0]+cm[1,1])
```

Sledeće linije koda štampaju vrednosti za senzitivnost i specifičnost:

```
print('Sensitivity : ', sensitivity)
print('Specificity : ', specificity)

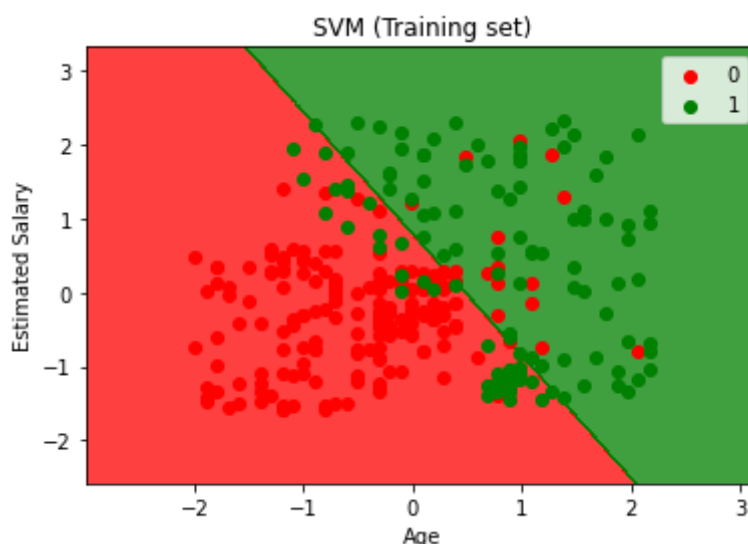
Sensitivity : 0.9705882352941176
Specificity : 0.75
```

Slika 44 Prikaz senzitivnosti i specifičnosti

6.8 Vizuelizacija rezultata trening skupa

Sledeće linije koda vizuelizuju rezultat trening skupa:

```
# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                      np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
               c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('SVM (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



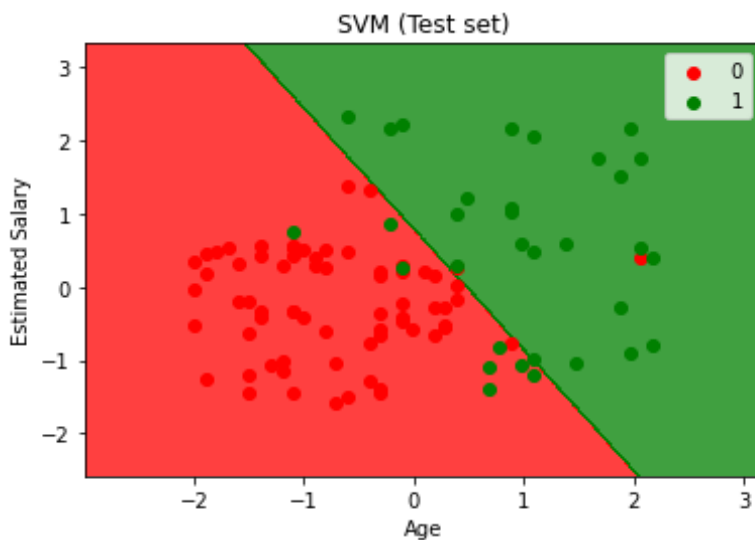
Slika 45 Grafički prikaz rezultata trening skupa

Sa grafika se vidi da kada se starost i procenjena plata povećavaju, svaki pojedinac ima veću verovatnoću da će biti zelen(klikne na oglas).

6.9 Vizuelizacija rezultata test skupa

Sledeće linije koda vizuelizuju rezultat test skupa:

```
# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                      np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('SVM (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



Slika 46 Grafički prikaz rezultata test skupa

7 K Nearest Neighbour

7.1 Uvoz potrebnih biblioteka i SNA skupa podataka

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('../input/Social_Network_Ads.csv')
```

7.2 Podela podataka na nezavisne promenljive i zavisnu promenljivu

SNA skup podataka je potrebno podeliti na nezavisne promenljive(X) i zavisnu promenljivu(y), kako bi se mogao odrediti efekat nezavisnih promenljivih na zavisnu promenljivu.

Sledeće linije koda dele SNA skup podataka na nezavisne promenljive(3 i 4 kolona(2 i 3 indeks)) i zavisnu promenljivu(5 kolona(4 indeks)):

```
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

Sledeće linije koda štampaju prva tri reda iz X i y skupa:

```
print(X[:3, :])
print('-'*15)
print(y[:3])
```

```
[[ 19 19000]
 [ 35 20000]
 [ 26 43000]]
-----
[0 0 0]
```

Slika 47 Prikaz prva tri reda iz X i y skupa

7.3 Podela podataka na skup za treniranje i na skup za testiranje

Da bi se model mogao naučiti, potrebno je podeliti podatke na skup podataka za treniranje i na skup podataka za testiranje modela.

Sledeće linije koda dele podatke na trening skup i na test skup:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.25, random_state = 0)
```

Sledeće linije koda štampaju prva tri reda iz novodobijenih skupova:

```
print(X_train[:3])
print('-'*15)
print(y_train[:3])
print('-'*15)
print(X_test[:3])
print('-'*15)
print(y_test[:3])
```



```

[[ 44 39000]
 [ 32 120000]
 [ 38 50000]]
-----
[0 1 0]
-----
[[ 30 87000]
 [ 38 50000]
 [ 35 75000]]
-----
[0 0 0]

```

Slika 48 Prikaz prva tri reda iz trening i test skupa

7.4 Normalizacija podataka

Da bi se dobili najtačniji rezultati, potrebno je primeniti normalizaciju podataka (skaliranje karakteristika).

Sledeće linije koda normalizuju podatke iz trening i test skupa:

```

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

```

Sledeće linije koda štampaju prva tri reda iz trening i test skupa, nakon normalizacije podataka:

```

print(X_train[:3])
print('-'*15)
print(X_test[:3])

```

```

[[ 0.58164944 -0.88670699]
 [-0.60673761  1.46173768]
 [-0.01254409 -0.5677824 ]]
-----
[[-0.80480212  0.50496393]
 [-0.01254409 -0.5677824 ]
 [-0.30964085  0.1570462 ]]

```

Slika 49 Prikaz prva tri reda iz trening i test skupa, nakon normalizacije podataka

7.5 Treniranje i testiranje modela

Sledeće linije koda kreiraju, treniraju i testiraju objekat klase KNeighborsClassifier:

```

# Fitting classifier to the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

```

Konstruktoru klase KNeighborsClassifier je prosleđen argument random_state s vrednošću 0, da bi se uvek dobili isti rezultati.

Sledeće linije koda štampaju prvih deset redova nakon predviđanja modela:

```

print(X_test[:10])
print('-'*15)

```

```
print(y_pred[:10])
```

```
[[-0.80480212  0.50496393]
 [-0.01254409 -0.5677824 ]
 [-0.30964085  0.1570462 ]
 [-0.80480212  0.27301877]
 [-0.30964085 -0.5677824 ]
 [-1.10189888 -1.43757673]
 [-0.70576986 -1.58254245]
 [-0.21060859  2.15757314]
 [-1.99318916 -0.04590581]
 [ 0.8787462  -0.77073441]]
-----
[0 0 0 0 0 0 0 1 0 1]
```

Slika 50 Prikaz prvih deset redova nakon predviđanja modela

Sledeće linije koda štampaju prvih dvadeset redova željenih i stvarnih izlaza:

```
print(y_pred[:20])
print(y_test[:20])
```

```
[0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0]
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0]
```

Slika 51 Prikaz prvih dvadeset redova željenih i stvarnih izlaza

7.6 Kreiranje konfuzione matrice

Sledeće linije koda kreiraju konfuzionu matricu:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

Sledeće linije koda štampaju konfuzionu matricu:

```
print(cm)
```

```
[[64  4]
 [ 3 29]]
```

Slika 52 Prikaz konfuzione matrice

Sa slike se vidi da je 93 tačnih predviđanja i 7 netačnih predviđanja, što znači da je model postigao ocenu tačnosti od 93%.

7.7 Metrike senzitivnosti i specifičnosti

Sledeće linije koda izračunavaju senzitivnost i specifičnost:

```
#####from confusion matrix calculate sensitivity and specificity
sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
specificity = cm[1,1]/(cm[1,0]+cm[1,1])
```

Sledeće linije koda štampaju vrednosti za senzitivnost i specifičnost:

```
print('Sensitivity : ', sensitivity)
print('Specificity : ', specificity)

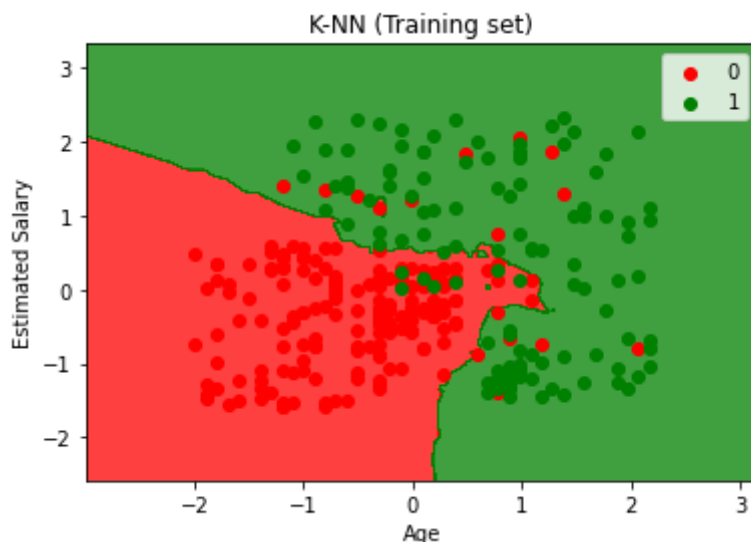
Sensitivity : 0.9411764705882353
Specificity : 0.90625
```

Slika 53 Prikaz senzitivnosti i specifičnosti

7.8 Vizuelizacija rezultata trening skupa

Sledeće linije koda vizuelizuju rezultat trening skupa:

```
# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
               c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('K-NN (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



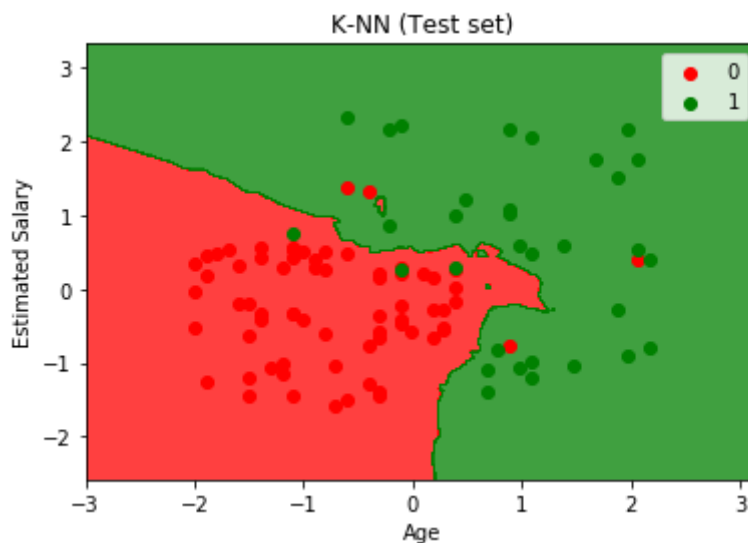
Slika 54 Grafički prikaz rezultata trening skupa

Sa grafika se vidi da kada se starost i procenjena plata povećavaju, svaki pojedinac ima veću verovatnoću da će biti zelen(klikne na oglas).

7.9 Vizuelizacija rezultata test skupa

Sledeće linije koda vizuelizuju rezultat test skupa:

```
# Visualising the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('K-NN (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



Slika 55 Grafički prikaz rezultata test skupa

8 Literatura

- [1] Kaggle -> Logistic Regression: Social Network Ads, link: <https://www.kaggle.com/johnduva/logistic-regression-social-network-ads>, 14.11.2021 (21:48)
- [2] Kaggle -> Decision Tree Classifier for Social Network Ads, link: <https://www.kaggle.com/farhanmd29/decision-tree-classifier-for-social-network-ads>, 14.11.2021 (21:48)
- [3] Kaggle -> Random Forest Classifier for Social Network Ads, link: <https://www.kaggle.com/farhanmd29/random-forest-classifier-for-social-network-ads>, 14.11.2021 (21:49)
- [4] Kaggle -> Naive Bayes Classifier on Social Media Ads, link: <https://www.kaggle.com/olusesiadebisi/naive-bayes-classifier-on-social-media-ads>, 14.11.2021 (21:50)
- [5] Kaggle -> SVM Model for Social Network Ads, link: <https://www.kaggle.com/farhanmd29/svm-model-for-social-network-ads>, 14.11.2021 (21:51)
- [6] Kaggle -> K-NN Model for Social Network Ads, link: <https://www.kaggle.com/farhanmd29/k-nn-model-for-social-network-ads>, 14.11.2021 (21:51)