Универзитет у Крагујевцу
Факултет инжењерских наука

Основи машинског и дубоког учења

Семинарски рад:
Маркетинг

Студент:
Михаило Јаћимовић 636/2018

Предметни професори:
Проф. Др. Владимир Миловановић

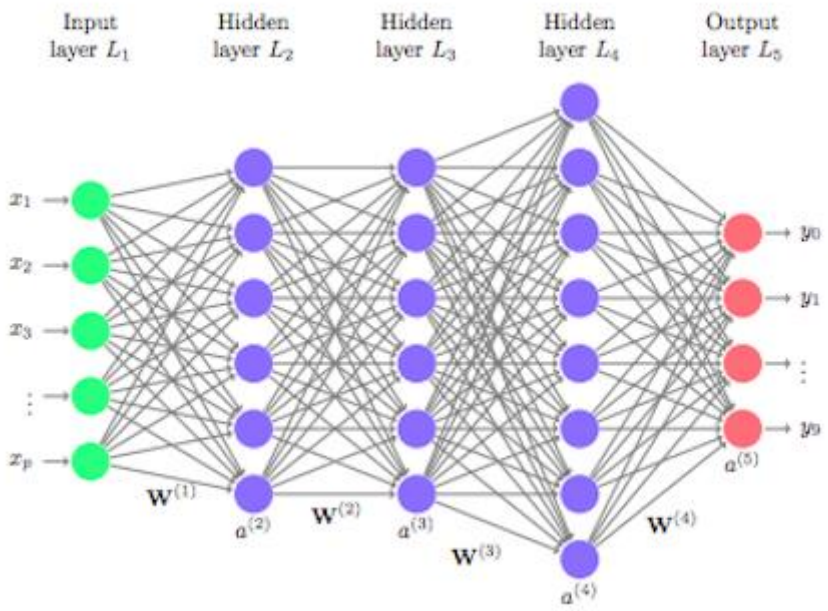
Крагујевац, Јун 2022.

Садржај

1. Поставка задатка

1.1. Циљ пројектног задатка

Циљ семинарског рада је да се направи програм која ће на основу улазних података који се добијају испитивањем купаца у Сан Франциску прорачунавати просечну плату коју дати купац заради у току једне године. Купац добија одређен број питања на који треба да одговори, ако не жели, испитаник не мора да одговори на сва питања. На крају фазе прикупљања података, укупно је испитано 8993 особе. Сваки купац је требало да одговори на четрнаест питања.
Програм представља приказ рада једне мреже за дубоко учење. На основу прикупљених података, правимо одговарајући модел. Када дати модел истренирамо, наш програм би требало да са високом сигурношћу може да предвиди просечну плату купца на основу улазних података.



Слика 1 – Изглед једне мреже за дубоко учење

1.2. Опис листе података

Након што смо прикупили податке, потребно их је исправно класификовати како би наш програм могао да их обради.
Наша листа података се састоји од четрнаест колона и 8993 реда. Сваки ред представља одговоре једног испитаника и јединствен је. Првих тринаест колона представљају улазне податке у наш систем илити атрибуте система, док последња колона представља излаз из система. На основу улазних података, наша мрежа ће предвиђати податке који су нама дати у последњој колони.
Наши улазни подаци су:

1. пол
2. брачни статус
3. године
4. образовање
5. занимање
6. колико година испитаник живи у Сан Франциску
7. дуална зарада
8. број чланова породице
9. број чланова породице млађих од 18 година
10. кућни статус
11. тип објекта где купац живи
12. етничка група
13. језик

Наш излазни податак је:

1. просечна годишња зарада

Index	Sex	laritalStati	Age	Education	)ccupatio	YearsInSf	ualIncom	eholdMer	Under18	seholdSta	peOfHom	thnicClas	Language	Income
878	2	1	7	2	5	5	3	2	0	1	1	7	1	2
879	1	1	7	3	8	5	3	2	0	1	1	7	1	2
880	2	1	6	3	5	?	3	2	0	1	1	7	1	5
881	2	4	5	3	6	?	1	3	1	1	2	7	1	3
882	1	1	6	2	1	?	3	2	0	1	1	7	1	5
883	2	1	4	6	1	5	2	3	1	1	1	7	1	9
884	1	5	4	5	1	?	1	1	0	1	1	7	2	7
885	2	1	4	6	9	5	3	2	0	1	1	7	1	9
886	1	1	4	4	3	5	2	2	0	1	1	7	1	8
887	1	5	1	2	6	5	1	3	1	3	1	7	1	1
888	2	1	4	3	5	1	3	4	2	2	1	7	1	5
889	1	1	7	3	4	4	2	2	0	1	4	7	1	4
890	2	1	6	3	4	4	2	2	0	1	4	7	1	4
891	2	1	5	6	1	?	2	3	0	1	1	5	2	3

Слика 2 – Изглед скупа података

2. Израда модела

За потребе овог пројекта коришћен је програмски језик Python. У делу 2.1. описане су библиотеке које су коришћене у реализацији датог задатка, у делу 2.2. описан је начин претпроцесирања података. У одељку 2.3. описан је проблем преучења(overfit) и начини на који он може да се превазиђе. Одељак 2.4. је посвећен матрици конфузије и метрици.

2.1. Python библиотеке

За решавање датог задатка коришћене су следеће python библиотеке:

1. pandas – користи се за манипулацију подацима и анализу
2. matplotlib – користи се за визуализацију података
3. numpy – користи се за рад са матрицама и низовима
4. tensorflow – библиотека за дубоко учење
5. sklearn – библиотека за машинско учење
6. imblearn – користи се за рад са неуравнотеженим подацима

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
from tensorflow import keras
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, BaggingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report,precision_score, p
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense
from imblearn.over_sampling import SMOTE
from sklearn.utils import class_weight
```

Слика 3 – Листа свих коришћених библиотека

2.2. Опис претпроцесирање података

Пошто су подаци достављени у excel фајлу, прво што је потребно урадити јесте учитавање података у облику матрице у пајтону.
Ту нам помаже библиотека pandas која има могућност да ради са подацима из excel табеле.

```
"Load dataset"
dataFrame = pd.read_csv("dataset.csv")

columns = list(dataFrame.columns)
print(columns)
```

Слика 4 – Учитавање података из excel фајла

Сада је креиран један dataFrame у којем се налазе сви подаци и над којима сада може да се манипулише у пајтону.
Командом print(columns) штампа се први ред, односно листа назива наших колона.

```
['Sex', 'MaritalStatus', 'Age', 'Education', 'Occupation',
'YearsInSf', 'DualIncome', 'HouseholdMembers', 'Under18',
'HouseholdStatus', 'TypeOfHome', 'EthnicClass', 'Language', 'Income']
```

Слика 5 – Листа назива колона

Пошто подаци нису комплетни, потребно је пронаћи сва поља која су празна и доделити им одређене вредности. У нашем случају, вредности које им се додељују представљају медијану дате колоне.
Медијана представља број који је граница између горње и доње половине одређеног низа вредности.
Уместо медијане могле су се користити неке друге вредности као замена за дата поља.

```
for col in columns:
   dataFrame[col] = dataFrame[col].replace("?", np.nan)
   dataFrame[col] = dataFrame[col].astype('float32')
   dataFrame[col].fillna(dataFrame[col].median(), inplace=True)
```

Слика 6 – Проналажење празних поља и додела вредности

Како би класификација боље радила, спајају се излази који су близу један другог.

```
for i in range(0, len(dataFrame['Income'])):
    if dataFrame['Income'][i] == 1 or dataFrame['Income'][i] == 2:
        dataFrame['Income'][i] = 0
    elif dataFrame['Income'][i] == 3 or dataFrame['Income'][i] == 4:
        dataFrame['Income'][i] = 1
    elif dataFrame['Income'][i] == 5 or dataFrame['Income'][i] == 6:
        dataFrame['Income'][i] = 2
    elif dataFrame['Income'][i] == 7 or dataFrame['Income'][i] == 8:
        dataFrame['Income'][i] = 3
    elif dataFrame['Income'][i] == 9:
        dataFrame['Income'][i] = 4
```

Слика 7 – Спајање излаза

Да не би дошло до ситуације да су подаци небалансирани, приликом поделе на тренинг, валидациони и тестни скуп, врши се насумично мешање података пре него што се поделе у дате скупове.

```
dataFrame = dataFrame.sample(frac=1, random_state=10)

X = dataFrame[columns[0:13]].values
y = dataFrame[columns[13]].values
```

Слика 8 – Насумично мешање

Функцијом train_test_split дели се скуп података на тренинг, валидациони и тестни скуп података. Након што се то уради, скалирају се улазни подаци, односно врши се нормализација улазних података на одређен скуп вредности, како би тренинг био бржи.

```
X_train, X_valid_test, y_train, y_valid_test = train_test_split(X, y, test_size= 0.30, random_state=10
x_valid, x_test, y_valid, y_test = train_test_split(X_valid_test, y_valid_test, test_size= 0.50, rando

scaler = StandardScaler()
X_train = pd.DataFrame(scaler.fit_transform(X_train))
x_valid = pd.DataFrame(scaler.fit_transform(x_valid))
X_test = pd.DataFrame(scaler.fit_transform(x_test))
```

Слика 9 – Подела улазних података на тренинг, валидациони и тестни скуп и нормализација података

2.3. Модел и проблем преучења

Када се заврши са претпроцесирањем података, све је спремно за креирање модела за дати проблем.
За креирање модела користе се библиотеке tensorflow и keras.
Модел који је коришћен има улазни слој са 13 улаза(за сваки атрибут по један улаз), има 5 скривених слојева, сваки слој као активациону функцију има функцију ReLU, за спречавање преучења коришћена је dropout метода, где се у одређеним слојевима, одређен број неурона насумично не користи приликом учења.
На последњем слоју користи се softmax као активациона функција и имамо 5 излаза.

```
model = keras.Sequential([
    keras.layers.Dense(128, input_dim=13, activation='relu'),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dropout(0.3),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dropout(0.3),
    keras.layers.Dense(5, activation='softmax'),
    ])
```

Слика 10 – Изглед модела мреже

Као функцију губитака користи се categorical crossentropy, а adam(adaptive moment estimation) користи се као оптимизатор.

```
model.compile(
    loss='categorical_crossentropy',
    #loss='mse',
    optimizer='adam',
    metrics=['accuracy']
    )
```

Слика 11 – Додатна кофигурација модела

Како је као функцију губитака одабрана categorical crossentropy потребно је да се onehot-yjy излазни подаци.
Након тога модел је спреман за тренирање. Изабрано је 50 као број епоха за тренирање.

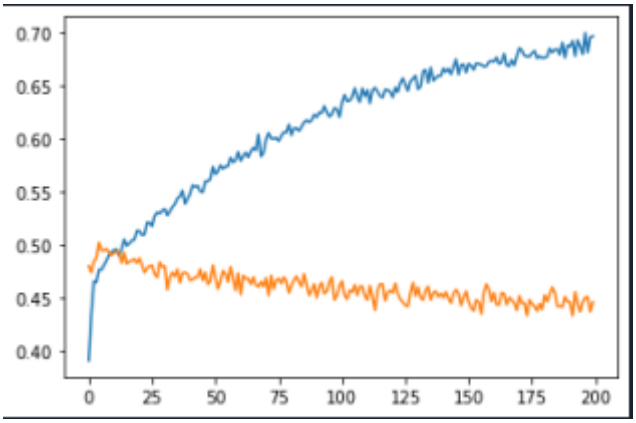
```
y_train_enc = pd.get_dummies(y_train)
y_valid_enc = pd.get_dummies(y_valid)

history = model.fit(X_train, y_train_enc, validation_data = (x_valid,y_valid_enc), epochs = 50)
```

Слика 12 – Тренирање модела

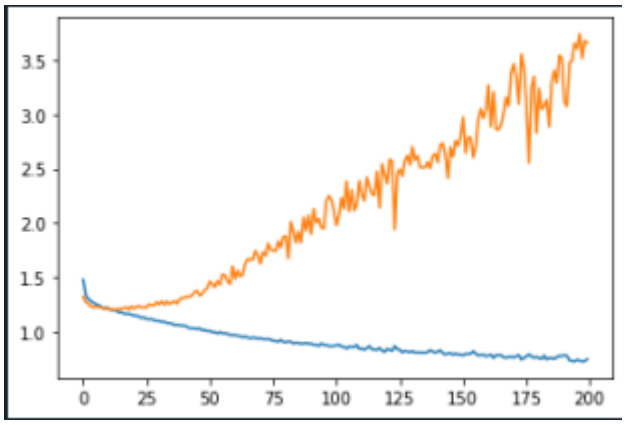
Када се истренира модел врши се валидација и предикција модела.
Предикција се врши на тестном скупу података. За модел најбољи резултат је 42% тачних предикција на тестном скупу, док је на тренинг скупу тај проценат знатно већи, што значи да је дошло да проблема преучења.

```
val_loss: 3.2985 - val_accuracy: 0.4366
Epoch 196/200
197/197 [==============================] - 0s 2ms/step - loss: 0.7381 - accuracy: 0.6799 -
val_loss: 3.5768 - val_accuracy: 0.4455
Epoch 197/200
197/197 [==============================] - 1s 3ms/step - loss: 0.7146 - accuracy: 0.6999 -
val_loss: 3.5688 - val_accuracy: 0.4500
Epoch 198/200
197/197 [==============================] - 1s 3ms/step - loss: 0.7481 - accuracy: 0.6815 -
val_loss: 3.6080 - val_accuracy: 0.4507
Epoch 199/200
197/197 [==============================] - 1s 3ms/step - loss: 0.7161 - accuracy: 0.6956 -
val_loss: 3.6365 - val_accuracy: 0.4366
Epoch 200/200
197/197 [==============================] - 0s 2ms/step - loss: 0.7153 - accuracy: 0.6969 -
val_loss: 3.9281 - val_accuracy: 0.4455
43/43 [==============================] - 0s 1ms/step
Model accuracy score is: 0.4262416604892513
```

Слика 13 – Вредност тачности модела



Слика 14 – Плавом бојом је представљена тачност на тренинг скупу, док је наранџастом бојом представљена тачност на валидационом скупу



Слика 15 – Плавом бојом је представљена функција губитака на тренинг скупу, док је наранџастом бојом представљена функција губитака на валидационом скупу

Како је дошло до преучења, потребно је искористити неке од следећих метода, да би модел вршио бољу предикцију:

1. Смањити комплексност модела
2. Користити регуларизацију
3. Користити ансабл методе
4. Рано стајање
5. Користити више података
6. Нормализација података
7. Претпроцесирање података
8. Небалансирани улазни подаци

Ако је модел превише комплексан може доћи до преучења, где ће се модел сконцентрисати на учење појединости сваког тренинг примера и неће правити добру генерализацију.
Смањењем комплексности модела, у нашем случају, броја скривених слојева и броја неурона, постижу се бољи резултати.
Подаци су претпроцесирани и извршена је нормализација пре поделе на тренинг, валидациони и тестни скуп.
Додата је dropout метода као вид регуларизације приликом учења, такође, додата је L2 регуларизација, али моделу није повећана тачност.
Коришћењем ансабл метода постигнути су исти резултати као и приликом коришћења неуронске мреже.
Рано стајање – надгледа се учење, тренутак када модел почиње да overfit-ује прекида се са тренирањем.
Улазни подаци нису небалансирани, свака класа садржи приближно једнак број примера.

```
y_train_enc = pd.get_dummies(y_train)
y_valid_enc = pd.get_dummies(y_valid)

stopping = EarlyStopping(monitor='val_loss', patience=25)
```

Слика 15 – Функција за рано стајање

```
val_loss: 1.2917 - val_accuracy: 0.4781
Epoch 31/200
197/197 [==============================] - 0s 2ms/step - loss: 1.0894 - accuracy: 0.5342 -
val_loss: 1.2569 - val_accuracy: 0.4700
Epoch 32/200
197/197 [==============================] - 0s 2ms/step - loss: 1.0760 - accuracy: 0.5380 -
val_loss: 1.3073 - val_accuracy: 0.4707
Epoch 33/200
197/197 [==============================] - 0s 2ms/step - loss: 1.0706 - accuracy: 0.5409 -
val_loss: 1.2916 - val_accuracy: 0.4722
Epoch 34/200
197/197 [==============================] - 0s 2ms/step - loss: 1.0683 - accuracy: 0.5455 -
val_loss: 1.3012 - val_accuracy: 0.4715
Epoch 35/200
197/197 [==============================] - 0s 2ms/step - loss: 1.0709 - accuracy: 0.5388 -
val_loss: 1.3186 - val_accuracy: 0.4759
43/43 [==============================] - 0s 1ms/step
Model accuracy score is: 0.4603409933283914
```

Слика 16 – Прецизност модела након додате функције EarlyStopping

Модел постиже бољу тачност након додавања функције earlyStopping, која износи 46%.

Смањивањем комплексности модела повећана је тачност погађања резултата на 50%.

```
model = keras.Sequential([
    keras.layers.Dense(26, input_dim=13, activation='relu'),
    keras.layers.Dense(15, activation='relu'),
    #keras.layers.Dense(64, activation='relu'),
    #keras.layers.Dropout(0.3),
    #keras.layers.Dense(9, activation='relu'),
    #keras.layers.Dense(9, activation='relu'),
    #keras.layers.Dropout(0.3),
    keras.layers.Dense(5, activation='softmax'),
    ])
```

Слика 17 – Модел након смањивања комплексности

```
val_loss: 1.2065 - val_accuracy: 0.4885
Epoch 37/200
197/197 [==============================] - 0s 2ms/step - loss: 1.1423 - accuracy: 0.5166 -
val_loss: 1.2074 - val_accuracy: 0.4900
Epoch 38/200
197/197 [==============================] - 0s 2ms/step - loss: 1.1417 - accuracy: 0.5228 -
val_loss: 1.2065 - val_accuracy: 0.4818
Epoch 39/200
197/197 [==============================] - 0s 2ms/step - loss: 1.1422 - accuracy: 0.5237 -
val_loss: 1.2092 - val_accuracy: 0.4885
Epoch 40/200
197/197 [==============================] - 0s 2ms/step - loss: 1.1401 - accuracy: 0.5268 -
val_loss: 1.2063 - val_accuracy: 0.4915
Epoch 41/200
197/197 [==============================] - 0s 2ms/step - loss: 1.1396 - accuracy: 0.5214 -
val_loss: 1.2060 - val_accuracy: 0.4855
43/43 [==============================] - 0s 1ms/step
Model accuracy score is: 0.4936990363232024
```

Слика 18 – Тачност новог модела

Ансамбл методе које су коришћене у задатку, како би се повећала тачност предвиђања јесу:

1. DecisionTreeClassifier
2. RandomForestClassifier
3. BaggingClassifier

Ни за један случај тачност се није повећала.

```
#model = DecisionTreeClassifier()
#model = RandomForestClassifier(n_estimators=10)
model = BaggingClassifier(KNeighborsClassifier(), max_samples=0.5, max_features=
model.fit(X_train,y_train)
y_pred = model.predict(X_test)

print(accuracy_score(y_test, y_pred))
#print(accuracy_score(y_train, y_pred))
print(confusion_matrix(y_test, y_pred))
```

Слика 19 – Ансамбл методе

2.4. Матрица конфузије и метрике

Матрица конфузије или матрица грешке, представља табелу која визуализује перформансе нашег алгоритма. Обично се користи у надгледаном учењу, у класификационим проблемима. Сваки ред представља једну класу, док свака колона представља једну предвиђану класу.
Алгоритам ради боље ако су вредности ван главне дијагонале приближно једнаке нули.

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

Слика 20 – Изглед матрице конфузије

```
[[295  40  30  12   1]
 [ 73  55  65  26   3]
 [ 47  50  91  84   3]
 [ 49  14  73 176  29]
 [ 19   1  12  68  33]]
```

Слика 21 – Матрица конфузије модела

Уз помоћ матрице конфузије могу да се израчунају различите метрике чијом анализом може да се постигне креирање бољег модела.
У задатку, метрике које се рачунају су:

1. тачност – представља пропорцију тачних предикција кроз укупан број података
2. прецизност – од свих позитивних предикција, који проценат је стварно позитиван
3. сензитивност – од свих позитивних вредности, које су стварно позитивне
4. специфичност – вероватноћа негативног резултата, под условом да је резултат стварно негативан
5. f1 резултат – представља хармонијску средину прецизности и сензитивности

$$
\begin{aligned}
precision &= \frac{TP}{TP + FP} \\
recall &= \frac{TP}{TP + FN} \\
F1 &= \frac{2 \times precision \times recall}{precision + recall} \\
accuracy &= \frac{TP + TN}{TP + FN + TN + FP} \\
specificity &= \frac{TN}{TN + FP}
\end{aligned}
$$

Слика 22 – Формуле за израчунавање метрика

	precision	recall	f1-score
class 0	0.60	0.78	0.67
class 1	0.34	0.28	0.31
class 2	0.36	0.28	0.31
class 3	0.46	0.50	0.48
class 4	0.51	0.32	0.39

Слика 23 – Прецизност, сензитивност и f1-резултат модела

```
{'class 0': 0.7777777777777778, 'class 1': 0.28378378378378377, 'class 2': 0.28, 'class 3':
0.5043988269794721, 'class 4': 0.3157894736842105}
```

Слика 24 – Тачност модела

3. Закључак

Задатак који је био описан у овом раду, служи да представи рад једне неуронске мреже. Прво је било потребно прикупити податке и направити базу података.
Подаци као такви нису довољни да се направи исправан модел, зато је следећи корак био претпроцесирање улазних вредности(попуњавање празних поља,...).
Затим је креиран одговарајући модел и истрениран на тренинг скупу, а провера тачности модела је вршена на тестном скупу.
Модел може да се поправи тако да даје боље резултате. Неки од начина да се то постигне су:

1. Прикупљање већег броја података
2. Отклањање непотребних атрибута
3. Применом дубљих неуронских мрежа
4. Применом неких других алгоритама машинског учења

4. Литература

1. Stuart Russell, Peter Norvig – Artificial Intelligence, a modern approach
2. https://keras.io/
3. https://docs.python.org/3/
4. https://en.wikipedia.org/wiki/Confusion_matrix
5. https://numpy.org/doc/stable/
6. https://towardsdatascience.com/demystifying-confusion-matrix-confusion-9e82201592fd
7. https://scikit-learn.org/stable/
8. https://towardsdatascience.com/handling-overfitting-in-deep-learning-models-c760ee047c6e