Pokretanje primjera iz uvodnog dijela laboratorijske vježbe

Na prvoj laboratorijskoj vježbi iz neuronskih mreža, studenti se upoznaju sa problemom klasifikacije, kao jednim od najčešćih problema iz prakse. Kroz zadatke, studenti rješavaju jednostavne probleme klasifikacije nad tekstualnim ulaznim podacima, koristeći Keras razvojni okvir.

Prikazivanje osnovnih informacija o skupu podataka poput minimalne vrijednosti, maksimalne vrijednosti, srednje vrijednosti, standardne devijacije, i tako dalje.

```
1 import pandas as pd
2 data = pd.read_csv('dataset.csv')
3 data.describe()
```

Pandas modul omogućava i crtanje histograma za svaki od atributa:

```
1 import pandas as pd
2 data = pd.read_csv('dataset.csv')
3 data.hist()
4
```

Primjer vršenja normalizacije

```
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split

#pripremanje testnih podataka
X, Y = np.arange(10).reshape((5, 2)), range(5)
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.33, random_state=42)

#Normalizacija
scaler = MinMaxScaler().fit(x_train)
x_train = scaler.fit_transform(x_train)
x_test = scaler.fit_transform(x_test)

x_train, x_test
```

```
(array([[0.66666667, 0.66666667],
        [0.        , 0.        ],
        [1.        , 1.        ]]), array([[0., 0.],
        [1., 1.]]))
```

Primjer vršenja standardizacije

```
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

#pripremanje testnih podataka
X, Y = np.arange(10).reshape((5, 2)), range(5)
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.33, random_state=42)

#Standardizacija
scaler = StandardScaler().fit(x_train)
x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)
```

```
13
14 x_train, x_test
```

```
    (array([[ 0.26726124,  0.26726124],
            [-1.33630621, -1.33630621],
            [ 1.06904497,  1.06904497]]), array([[-0.53452248, -0.53452248],
            [ 1.87082869,  1.87082869]]))
```

Primjer za model koji bi mogao poslužiti za binarnu klasifikaciju:

```
1 from keras import models
2 from keras import layers
3
4 model = models.Sequential()
5 model.add(layers.Dense(8, activation='relu', input_shape=(4000,)))
6 model.add(layers.Dense(8, activation='relu'))
7 model.add(layers.Dense(1, activation='sigmoid'))
8 model
```

```
    <keras.engine.sequential.Sequential at 0x7f30522d6bd0>
```

Postavljanje funckije gubitka, optimizatora kao i metrike koja će se pratiti na modelu

```
1 #Adam optimizator sa podrazumijevanim postavkama
2 model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
3
4 #eksplicitno podesavanje istog optimizatora
5 '''from tensorflow import keras
6 opt = keras.optimizers.Adam(learning_rate=0.01)
7 model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])'''
8
```

Zadatak 1 - Binarna klasifikacija - Klasifikacija vina

a) Učitati podatke za crno i bijelo vino koji se nalaze u CSV datotekama winequality-red.csv i winequality-white.csv u varijable red i white respektivno. Pri ovome postaviti parametar separator da bude znak tačka-zarez (;). U odgovarajućim DataFrame objektima dodati novu kolonu label koja će imati vrijednost 0 za podatke bijelog vina, a vrijednost 1 za podatke crnog vina. Nakon ovoga spojiti ih u jedan DataFrame objekat wines.

```
import pandas as pd
import numpy as np

#ucitavanje podataka, kao i postavljanje separatora
red = pd.read_csv("winequality-red.csv",sep=";")
white = pd.read_csv("winequality-white.csv",sep=";")

#dodavanje kolone "label", gdje je crvenom vinu label = 1, a bijelom = 0
red["label"] = 1
white["label"] = 0

#spajanje u jedan DataFrame objekat
wines = pd.concat([white,red],ignore_index=True)
wines


```

b) Izvršiti prikaz osnovnih podataka spojenog skupa podataka korištenjem **describe** metode te nacrtati histograme korištenjem **hist** metode. Šta možete zaključiti o podacima?

```
wines.describe()

```

```
1 wines.hist(figsize=(15,20))
```

c) Iz wines izdvojiti X - karakteristike i y - labele. Zatim korištenjem funkcije train_test_split podijeliti podatke na one za treniranje i one za testiranje pri čemu 20% podataka trebaju biti podaci za testiranje;

```
#izdvoji/pripremi/razvrstaj podatke za train i test split
x = wines.drop("label",axis=1)
y = wines["label"]

```

```
#razdvajanje podataka na skupove za treniranje i za testiranje
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

```

d) Formirati sekvencijalni Keras model koji će imati 2 Dense sloja sa po 8 neurona i relu aktivacijskom funkcijom. Prvom sloju kao input_shape parametar proslijediti vrijednost (12,) s obzirom na to da skup podataka ima 12 značajki. Treći sloj, koji je u ovom slučaju izlazni sloj, postaviti da također bude Dense, ali sa samo jednim neuronom;

```
from keras import models
from keras import layers
model = models.Sequential()
model.add(layers.Dense(8, activation = "relu", input_shape=(12,)))
model.add(layers.Dense(8, activation = "relu"))
model.add(layers.Dense(1, activation = "sigmoid"))

```

e) Kompajlirati model tako da koristi adam optimizator, za funkciju gubitka koristiti binary_crossentropy, a kao metriku odabrati accuracy;

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

```

f) Model istrenirati na 20 epoha, sa veličinom batch-a 16.

```
history = model.fit(x_train, y_train, epochs=20, batch_size=16, validation_split=0.2)

```

```
Epoch 1/20
260/260 [==============================] - 1s 3ms/step - loss: 0.4484 - accuracy: 0.8126
Epoch 2/20
260/260 [==============================] - 1s 2ms/step - loss: 0.2159 - accuracy: 0.9236
Epoch 3/20
260/260 [==============================] - 1s 2ms/step - loss: 0.1898 - accuracy: 0.9389
Epoch 4/20
260/260 [==============================] - 1s 2ms/step - loss: 0.1832 - accuracy: 0.9408
Epoch 5/20
260/260 [==============================] - 1s 2ms/step - loss: 0.1762 - accuracy: 0.9427
Epoch 6/20
260/260 [==============================] - 1s 2ms/step - loss: 0.1701 - accuracy: 0.9437
Epoch 7/20
260/260 [==============================] - 1s 2ms/step - loss: 0.1644 - accuracy: 0.9454
Epoch 8/20
260/260 [==============================] - 1s 2ms/step - loss: 0.1597 - accuracy: 0.9476
Epoch 9/20
260/260 [==============================] - 1s 2ms/step - loss: 0.1540 - accuracy: 0.9473
```

```
Epoch 10/20
260/260 [==============================] - 1s 3ms/step - loss: 0.1460 - accuracy: 0.9512
Epoch 11/20
260/260 [==============================] - 1s 3ms/step - loss: 0.1339 - accuracy: 0.9541
Epoch 12/20
260/260 [==============================] - 1s 2ms/step - loss: 0.1144 - accuracy: 0.9598
Epoch 13/20
260/260 [==============================] - 1s 2ms/step - loss: 0.1055 - accuracy: 0.9644
Epoch 14/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0962 - accuracy: 0.9666
Epoch 15/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0930 - accuracy: 0.9690
Epoch 16/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0870 - accuracy: 0.9721
Epoch 17/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0823 - accuracy: 0.9735
Epoch 18/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0809 - accuracy: 0.9731
Epoch 19/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0779 - accuracy: 0.9767
Epoch 20/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0767 - accuracy: 0.9759
```

```
from matplotlib import pyplot as plt
acc = history.history['accuracy']
loss_values = history.history['loss']
val_loss_values = history.history['val_loss']
epochs = range(1, len(acc) + 1)
plt.plot(epochs, loss_values, 'bo', label='Training loss')
plt.plot(epochs, val_loss_values, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
```

g) Ponoviti postupak, no ovaj put prije treniranja izvršiti standardizaciju, odnosno skaliranje atributa korištenjem StandardScaler objekat iz sklearn.preprocessing modula. Kolika je sada tačnost nakon treniranja? Uporediti rezultate sa onim bez skaliranja

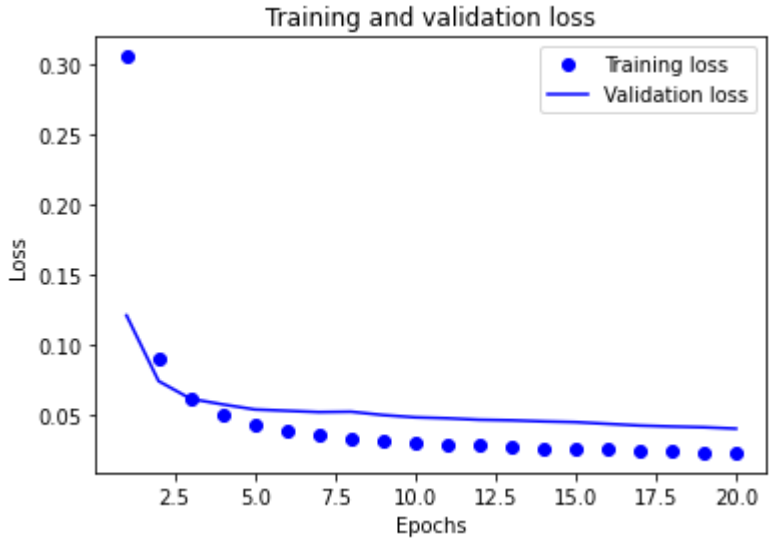
```
red = pd.read_csv("winequality-red.csv",sep=";")
white = pd.read_csv("winequality-white.csv",sep=";")
X = wines.drop("label",axis=1)
y = wines["label"]
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.2, random_state=42)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=20, batch_size=16, validation_split=0.2)

```

```
Epoch 1/20
260/260 [==============================] - 1s 3ms/step - loss: 0.3049 - accuracy: 0.9290
Epoch 2/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0904 - accuracy: 0.9755
Epoch 3/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0613 - accuracy: 0.9827
Epoch 4/20
260/260 [==============================] - 1s 3ms/step - loss: 0.0496 - accuracy: 0.9868
Epoch 5/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0428 - accuracy: 0.9892
Epoch 6/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0382 - accuracy: 0.9911
Epoch 7/20
260/260 [==============================] - 1s 3ms/step - loss: 0.0351 - accuracy: 0.9923
Epoch 8/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0327 - accuracy: 0.9935
Epoch 9/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0312 - accuracy: 0.9933
Epoch 10/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0299 - accuracy: 0.9933
Epoch 11/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0290 - accuracy: 0.9947
Epoch 12/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0279 - accuracy: 0.9947
Epoch 13/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0271 - accuracy: 0.9952
Epoch 14/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0262 - accuracy: 0.9959
Epoch 15/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0257 - accuracy: 0.9952
Epoch 16/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0252 - accuracy: 0.9954
Epoch 17/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0243 - accuracy: 0.9954
```

```
Epoch 18/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0238 - accuracy: 0.9957
Epoch 19/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0232 - accuracy: 0.9954
Epoch 20/20
260/260 [==============================] - 1s 2ms/step - loss: 0.0228 - accuracy: 0.9957
```

```
from matplotlib import pyplot as plt
acc = history.history['accuracy']
loss_values = history.history['loss']
val_loss_values = history.history['val_loss']
epochs = range(1, len(acc) + 1)
plt.plot(epochs, loss_values, 'bo', label='Training loss')
plt.plot(epochs, val_loss_values, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f30524466d0>
```



h) Model evaluirati nad testnim skupom podataka. Kolika je postignuta tačnost modela?

```
results = model.evaluate(x_test, y_test)
print("Postignuta je tacnost: ",results[1])
```

```
41/41 [==============================] - 0s 4ms/step - loss: 31.2891 - accuracy: 0.7585
Postignuta je tacnost:  0.7584615349769592
```