**OSNOVE STROJNOG UČENJA – LABOSI**

**LV1**

* Otvaranje jednostavnih tekstualnih datoteka
* Uvjeti, catch try blokovi, rstrip i split na riječima
* Prosječan broj riječi, min, max, srednja vrijednost, broj riječi

**LV2**

* import numpy as np
* Numpy array
* Rad s numpy poljima, matematičke operacije
* Generator slučajnih brojeva
* Matplotlib – kreiranje statičkih i dinamičkih vizualizacija
* data = np.loadtxt("data.csv", skiprows = 1, delimiter = ',')
* učitavanje cs datoteke u numpy array

**ZADATAK 2:** Rad s csv datotekom u obliku numpy arraya

Dohvaćanje stupaca, preskakanje redova, sve jasno

**ZADATAK 3:**

img = plt.imread("road.jpg")

* dohvaćanje slike
* izmjena slike po zahtijevima iz zadatka

**ZADATAK 4:**

* numpy funkcije zeros i ones kako bi kreirali polja
* numpy funkcije hstack i vstack

**LV3**

* pandas biblioteka
* data = pd . read\_csv ( "data\_C02\_emission.csv")
* dohvaća se uglatim zagradama
* metode DataFrameova po željenim stupcima
* primjer 3.5. radi sa stupcima
* **DUPLICIRANE VRIJEDNOSTI PRIMJER 3.7.**
* **PLOTOVI SU OD 3.8. PA NADALJE**
* **DRUGI ZADATAK JE VIZUALIZACIJA**

**LV4**

* Linearni modeli za regresiju
* Veličine x i y
* Skup za učenje modela(trening skup) i skup za testiranje modela (testni skup)
* x\_train, x\_test, y\_train, y\_test = train\_test\_split(
* x, y, test\_size=0.2, random\_state=1)
* Linearni regresijski model i funkcije ( **MinMaxScaler, fit, LinearRegression)**
* **SVE IMPORTAT – bilbioteka sklearn**
* Evaluacija modela: .predict
* NA ISPITU KORISTITI .TONUMPY koji pretvara dataframe stupce u numpy array

**LV5**

**NADZIRANO UČENJE KLASIFIKACIJA**

* scikit-learn biblioteka
* Najvažnije metode ove klase su:   
  • .fit(X,y)za procjenu parametara modela na temelju podataka za ucenje, ˇ   
  • .predict(X) za izracunavanje izlaza modela na temelju ulaznih vrijednosti ulaznih veli ˇ cina, ˇ• .predict\_proba(X) za izracunavanje vjerojatnosti klase na temelju ulaznih vrijednosti ulaznih ˇ velicina
* **Matrica zabune, tocnost, preciznost, odziv**
* **Plotovi na skupu za ucenje i treniranje**

**LV6**

**KLASIFIKACIJA PRIMJENOM ALGORITMA K NAJBLIŽIH I SVM (KNN)**

**U ZADATKU IMA .TONUMPY I SVE**

**LV7**

* **K SREDNJIH VRIJEDNOSTI**

**LV8 – UMJETNE NEURONSKE MREŽE (ISPIT)**

* IZRADA MREŽE - SLOJEVI
* PODEŠAVANJE PARAMETARA TRENIGNA
* POKRETANJE MREŽE UČENJA
* SPREMANJE MODLEA
* UČITAVANJE MODELA

**GLAVNE FUNKCIJE ZA ISPIT**

* data = np.loadtxt('', delimiter=',', skiprows=0) – **učitavanje dataseta u numpy**
* data\_df = pd.DataFrame(data)- **csv u pandas DataFrame**
* print(f'Broj dupliciranih: {data\_df.duplicated().sum()}')
* print(f'Broj izostalih: {data\_df.isnull().sum()} ')
* data\_df = data\_df.drop\_duplicates()

RAD SA DUPLICIRANIM I IZOSTALIM PODACIMA

* data\_df = data\_df.dropna(axis=0) **i onda** data\_df = pd.DataFrame(data) za kreiranje date s očišćenim podacima
* **min, max, scatter LV1 LV2 LV3**
* data = pd.read\_csv('data\_C02\_emission.csv')
* input\_variables = ['Fuel Consumption City (L/100km)', 'Fuel Consumption Hwy (L/100km)',
* 'Fuel Consumption Comb (L/100km)', 'Fuel Consumption Comb (mpg)', 'Engine Size (L)', 'Cylinders']
* output\_variable = ['CO2 Emissions (g/km)']
* x = data[input\_variables].to\_numpy()
* y = data[output\_variable].to\_numpy()

**TRAIN TEST SPLIT:** X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, random\_state=5)

LEGISTIČKA REGRESIJA: logReg\_model = LogisticRegression(max\_iter=300)

logReg\_model.fit(X\_train, y\_train)

MATRICA ZABUNE: disp = ConfusionMatrixDisplay(confusion\_matrix(y\_test, y\_predictions))

disp.plot()

plt.show()

PARAMETRI: print (" Tocnost : " , accuracy\_score ( y\_test , y\_pred ) )

print(" Preciznost : ", precision\_score (y\_test, y\_pred))

print(" Odziv: ", recall\_score(y\_test, y\_pred))

**KORIŠTENJE ČISTOG DATAFRAMEA SA POTREBNIM ULAZNIM I IZLAZNIM PODACIMA**

data\_df = pd.DataFrame(data, columns=['',]

X = data\_df.drop(columns=['']).to\_numpy()

y = data\_df[''].copy().to\_numpy()

**DODAVANJE LAYERA**

model = keras.Sequential()

model.add(layers.Input(shape=(784,)))

model.add(layers.Dense(units=100, activation="relu"))

model.add(layers.Dense(units=50, activation="relu"))

model.add(layers.Dense(units=10, activation="softmax"))

model.summary()

#podesavanje parametara treninga

model.compile(loss="categorical\_crossentropy", optimizer="adam", metrics=["accuracy",])

#POKRETANJE UČENJA MREŽE

history = model.fit(X\_train\_reshaped , y\_train\_encoded, batch\_size=32, epochs=20, validation\_split=0.1)

#evaluacija i ispis

score = model.evaluate(X\_test\_reshaped, y\_test\_encoded, verbose=0)

for i in range(len(model.metrics\_names)):

    print(f'{model.metrics\_names[i]} = {score[i]}')

#predict i matrica zabune

y\_predictions = model.predict(X\_test\_reshaped)  #vraca za svaki primjer vektor vjerojatnosti pripadanja svakoj od 10 klasa (softmax) (10 000,10)

y\_predictions = np.argmax(y\_predictions, axis=1)  #vraća polje indeksa najvecih elemenata u svakom pojedinom retku (1d polju) (0-9) (10 000,) - 1d polje

cm = confusion\_matrix(y\_test, y\_predictions)    #zbog prethodnog koraka, usporedba s y\_test, a ne encoded

disp = ConfusionMatrixDisplay(confusion\_matrix=cm)

disp.plot()

plt.show()

#spremanje modela

model.save('Model/')

#ucitavanje modela

model = load\_model('Model/')

**DOHVAĆANJE PODATAKA:  
  
NUMPY**

* plt.scatter(x=data[:, 7], y=data[:, 5])

**PANDAS**print({data\_df[data\_df[8]==1][5].min()}')

print({data\_df[data\_df[8]==1][5].max()}')

print({data\_df[data\_df[8]==1][5].mean()}')

# e) Izvršite evaluaciju mreže na testnom skupu podataka.

model = load\_model('Model/')

score = model.evaluate(X\_test, y\_test, verbose=0)

for i in range(len(model.metrics\_names)):

    print(f'{model.metrics\_names[i]} = {score[i]}')

# f) Izvršite predikciju mreže na skupu podataka za testiranje. Prikažite matricu zabune za skup podataka za testiranje.

y\_predictions = model.predict(X\_test)

y\_predictions = np.around(y\_predictions).astype(np.int32)

cm = confusion\_matrix(y\_test, y\_predictions)

disp = ConfusionMatrixDisplay(confusion\_matrix=cm)

disp.plot()

plt.show()