import numpy as np

import matplotlib . pyplot as plt

data=np.loadtxt("dataa.csv", delimiter=',', skiprows=1)

-učitavnje csv datoteke, gdje se odvajaju zarezom i preskače se prvi red

print(f"Broj osoba na kojem je vršeno mjerenje {len(data)}")

-ispis broja mjerenja

plt.scatter(x=data[:,1],y=data[:,2])

omjer gdje je x 2 stupac, a y 3 (brojanje počinje od 0)

data\_filtered=data[::50] #svaki 50. red

-ako se traži svaki 50 red

print(f"Maksimalna visina {data[:,1].max()}")

-ispis maksimalne vrijednosti 2 retka, također imamo min() i mean()

data\_male=data[data[:,0]==1]

-ako želimo odabrati tamo gdje je prvi stupac jednak 1 odnosno spol=1 muško

import numpy as np

import matplotlib as plt

import pandas as pd

data=pd.read\_csv("data\_C02\_emission.csv")

-učitavnje sa pandasom, ako nam vrijednosti nisu samo numeričke

# brisanje redova gdje barem vrijednost jedne velicine nedosta je

data.dropna(axis=0)

# brisanje stupaca gdje barem jedna vrijednost nedostaje

data.dropna(axis=1)

# brisanje dupliciranih redova

data.drop\_duplicates()

# kada se obrisu pojedini redovi potrebno je resetirati indekse retka

data =data.reset\_index(drop = True)

# provjera koliko je izostalih vrijednosti po svakom stupcu DataFramea:

print(data.isnull().sum())

# konvertiraj tipove object u kategorije

data[data.select\_dtypes(['object']).columns] = data.select\_dtypes(['object']).apply(lambda x : x.astype('category'))

print(data.info())

kategorije i uvjeti:

#) Koja tri automobila ima najvecu odnosno najmanju gradsku potrošnju? Ispišite u terminal: ´ime proizvodaca, model vozila i kolika je gradska potrošnja.

sub\_data=data[["Fuel Consumption City (L/100km)","Make", "Model"]]

print(f"3 najveća:\n{sub\_data.nlargest(3, 'Fuel Consumption City (L/100km)')}")

print(f"3 najmanja:\n{sub\_data.nsmallest(3, 'Fuel Consumption City (L/100km)')}")

#Koliko vozila ima velicinu motora izmedu 2.5 i 3.5 L? Kolika je prosjecna C02 emisija  plinova za ova vozila?

data\_engine\_subset = data[(data['Engine Size (L)'] > 2.5) & (data['Engine Size (L)'] < 3.5)]

print(f"Broj vozila sa trazenomvelicinom motora: {len(data\_engine\_subset)}")

print(f"Prosjek emisije CO2 takvih vozila: {data\_engine\_subset['CO2 Emissions (g/km)'].mean()}")

#Koliko mjerenja se odnosi na vozila proizvodaca Audi? Kolika je prosjecna emisija C02 ˇplinova automobila proizvodaca Audi koji imaju 4 cilindara?

data\_audi\_subset=data[(data['Make']=='Audi')]

print(f"Broj vozila marke audi:{len(data\_audi\_subset)}")

data\_audi\_subset\_c=data\_audi\_subset[(data\_audi\_subset['Cylinders']==4)]

print(f"Prosjecna emisija plinova audija sa 4 cilindra:{data\_audi\_subset\_c['CO2 Emissions (g/km)'].mean()}")

#Koliko je vozila s 4,6,8. . . cilindara? Kolika je prosjecna emisija C02 plinova s obzirom na ˇbroj cilindara?

data\_4cy=data[(data['Cylinders']==4)]

print(f"Broj autiju sa 4 cilindra: {len(data\_4cy)}")

data\_6cy=data[(data['Cylinders']==6)]

print(f"Broj autiju sa 6 cilindra: {len(data\_6cy)}")

data\_8cy=data[(data['Cylinders']==8)]

print(f"Broj autiju sa 8 cilindra: {len(data\_8cy)}")

Logistička regresija:

# učitavanje dataseta

data\_df = pd.DataFrame(data, columns=['num\_pregnant', 'plasma', 'blood\_pressure','triceps', 'insulin', 'BMI', 'diabetes\_function', 'age', 'diabetes']) #koriste se ocisceni podaci za dataframe

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

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

# a)

logReg\_model = LogisticRegression(max\_iter=300)

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

# b)

y\_predictions = logReg\_model.predict(X\_test)

# c)

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

disp.plot()

plt.show()

# broj TN je 89, TP 36, FN 18 i FP 11, model često osobe koje imaju dijabetes proglasi da nemaju - greška, nedovoljno komentirano

# d)

print(f'Tocnost: {accuracy\_score(y\_test, y\_predictions)}')

print(f'Preciznost: {precision\_score(y\_test, y\_predictions)}')

print(f'Odziv: {recall\_score(y\_test, y\_predictions)}')

Neuronska mreža:

# učitavanje podataka:

data\_df = pd.DataFrame(data, columns=['num\_pregnant', 'plasma', 'blood\_pressure',   'triceps', 'insulin', 'BMI', 'diabetes\_function', 'age', 'diabetes']) #koriste se ocisceni podaci za dataframe

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

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

# a)

model = keras.Sequential()

model.add(layers.Input(shape=(8,)))

model.add(layers.Dense(units=12, activation="relu"))

model.add(layers.Dense(units=8, activation="relu"))

model.add(layers.Dense(units=1, activation="sigmoid"))

model.summary()

# b)

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

# c)

history = model.fit(X\_train, y\_train, batch\_size=10, epochs=150, validation\_split=0.1)

# d)

model.save('Model/')

# e)

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)

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()

Stupičasti grad sa postocima:

# Create a bar chart

labels = ['Male', 'Female']

percentages = [male\_survivors\_percentage, female\_survivors\_percentage]

plt.bar(labels, percentages)

plt.xlabel('Gender')

plt.ylabel('Survival Percentage')

plt.title('Percentage of Male and Female Survivors')

# Display the chart

plt.show()

KNN:

data = pd.read\_csv("titanic.csv")

data.dropna(inplace=True)

X = data[['Pclass', 'Sex', 'Fare', 'Embarked']]

y = data['Survived']

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.4, random\_state=42)

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

numeric\_features = ['Fare']

categorical\_features = ['Pclass', 'Sex', 'Embarked']

preprocessor = ColumnTransformer(

    transformers=[

        ('numeric', StandardScaler(), numeric\_features),

        ('categorical', OneHotEncoder(), categorical\_features)

    ])

X\_train = preprocessor.fit\_transform(X\_train)

X\_test = preprocessor.transform(X\_test)

from sklearn.neighbors import KNeighborsClassifier

import matplotlib.pyplot as plt

import numpy as np

knn = KNeighborsClassifier(n\_neighbors=5)

knn.fit(X\_train, y\_train)

# Visualize decision boundary

def plot\_decision\_boundary(model, X, y):

    h = 0.02  # step size in the mesh

    x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1

    y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1

    xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h), np.arange(y\_min, y\_max, h))

    Z = model.predict(np.c\_[xx.ravel(), yy.ravel()])

    Z = Z.reshape(xx.shape)

    plt.contourf(xx, yy, Z, alpha=0.8)

    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k')

    plt.xlabel('Feature 1')

    plt.ylabel('Feature 2')

    plt.title('KNN Decision Boundary')

    plt.show()

# Assuming X has 2 features for visualization purposes

X\_train\_vis = X\_train[:, :2]

plot\_decision\_boundary(knn, X\_train\_vis, y\_train)

train\_accuracy = knn.score(X\_train, y\_train)

test\_accuracy = knn.score(X\_test, y\_test)

print(f"Training accuracy: {train\_accuracy}")

print(f"Testing accuracy: {test\_accuracy}")

from sklearn.model\_selection import cross\_val\_score

k\_values = [3, 5, 7, 9, 11]

cv\_scores = []

for k in k\_values:

    knn = KNeighborsClassifier(n\_neighbors=k)

    scores = cross\_val\_score(knn, X\_train, y\_train, cv=5)

    cv\_scores.append(scores.mean())

best\_k = k\_values[np.argmax(cv\_scores)]

print(f"Best K value: {best\_k}")

Ako treba neku vrijednost string u neki broj

data\_df.drop\_duplicates()

data\_df=data\_df.reset\_index(drop=True)

data\_df.loc[data\_df.Sex=="male", "Sex"]=0

data\_df.loc[data\_df.Sex=="female", "Sex"]=1

data\_df.loc[data\_df.Embarked=="S", "Embarked"]=0

data\_df.loc[data\_df.Embarked=="C", "Embarked"]=1

data\_df.loc[data\_df.Embarked=="Q", "Embarked"]=2

print(len(data\_df))