```
# Inicializace prostredi, nacteni funkci

import os
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, confusion_matrix as cm
import seaborn as sn
from sklearn import preprocessing
import pandas as pd
import matplotlib.pyplot as plt


## Data k nauceni neuronky

# Inicializace cesty k souboru
path = os.path.join(os.getcwd(), 'data_train.csv')

# Nacteni dat
data_train = pd.read_csv (path)

# Vypis prvnich 5 radku souboru pro kontrolu
print(data_train.head())

## Zakladni analyza dat

# Inicializace histogramu
data_train.hist(figsize=(10,10))

# Zobrazeni histogramu
plt.show()


# Kontrola uplnosti dat

missing_values_count = data_train.isnull().sum()
missing_values_count[0:]


# Ziska pocet sloupcu (data jsou neco jako matice... data.shape vrati radky
, sloupce -> 0 vrati radky, 1 vrati sloupce)
cols = data_train.shape[1]

# Inicializuje promennou X, slozenou ze sloupcu z dat, krome posledního
sloupce
X = data_train.iloc[:,0:cols-1]

# Inicializuje promennou Y, slozenou z posledniho sloupce (Y je zavisla pro
menna)
y = data_train.iloc[:,cols-1:cols]
```

```
# Transformovani a vypsani zavisle promnenne Y
le = preprocessing.LabelEncoder()
y = y.apply(le.fit_transform)
y

# Transformovani a vypsani zavisle promnenne Y
le = preprocessing.LabelEncoder()
y = y.apply(le.fit_transform)
y

## Rozdeli data na trenovaci a testovaci

# X_train se vyuzije k trenovani
# Y_train take uci neuronku
# X_test slouzi k testovani a ziskani predikce
# Y_test slouzi k overeni predikce a vyhodnoceni uspesnosti
# nahodny vyber radku z trenovacich dat a rozdeleni je v pomeru 70% : 30% -
> TRAIN : TEST
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, r
andom_state=42)

# Pripraveni dat pro neuronku
scaler = preprocessing.StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

# Inicializace klasifikatoru (neuronky)
# Neuronka ma 5 vrstev s osmi neurony v kazde verstve a probiha tam 1000 it
eraci
mlp = MLPClassifier(hidden_layer_sizes=(8, 8, 8, 8, 8), max_iter=1000)

# Fitrovani dat (trenovani)
mlp.fit(X_train, y_train.values.ravel())


# Ziskani predikce vlozenim testovacich dat do nacvicene neuronky
predictions = mlp.predict(X_test)
print(predictions)


# Vezme testovaci radky a prida k nim sloupce spravne vysledky a predikce
cols = X_test.shape[1]
original_result = pd.DataFrame(data = y_test, columns = data_train.columns[
:])
train_result = pd.DataFrame(data = X_test, columns = data_train.columns[0:c
ols])

train_result.insert(cols, 'Original Result', original_result)
train_result.insert(cols + 1, 'Predictions', predictions)
train_result.head()
## Vypis vysledku predikce
```

```
# Inicializace cesty, kam my byt vysledek ulozen
pathTrain = os.path.join(os.getcwd(), 'train_prediction.csv')

# exportuje data frame do csv a ulozi na urcenou cestu
train_result.to_csv(pathTrain, index=False)


# Zjisti presnost neuronky (porovnani testovacich vysledku a predikce)
accuracy_score(y_test, predictions)


## Matice zamnen (Confusion Matrix)

# Pro matici je potreba zase puvodnich vysledku testovacich dat a jejich pr
edikce
predictions_translated = le.inverse_transform(predictions)
y_translated = le.inverse_transform(y_test.values.ravel())

ax = plt.subplot()
sn.set(font_scale=1.8)
sn.heatmap(cm(y_test, predictions), annot=True, annot_kws={"size": 16}, ax=
ax)

ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix')
ax.xaxis.set_ticklabels(le.classes_, rotation=45)
ax.yaxis.set_ticklabels(le.classes_, rotation=45)


## Data testovaci - nemaji vyhodnocene data pro overeni

path = os.path.join(os.getcwd(), 'data_test.csv')
data_test = pd.read_csv (path)
print(data_test.head())


# Tyto data nemaji posledni sloupec s vysledky, tedy uz X tvori vsechny slo
upce
cols = data_test.shape[1]
X_data_test = data_test.iloc[:,0:cols]


scaler = preprocessing.StandardScaler()
scaler.fit(X_data_test)

X_data_test = scaler.transform(X_data_test)
```

```
# Neuronka je uz natrenovana z trenovacich dat a tedy staci aplikovat na te
stovaci data a ziskat predikci
predictions = mlp.predict(X_data_test)
print(predictions)


# Pridani predikce a testovacim datum (jako posledni sloupec)
cols = data_test.shape[1]
data_test.insert(cols, 'Predictions', predictions)
data_test.head()


# Vytvoreni data framu positive a negative slozenim z predikce a id radku
final_result = data_test.iloc[:,0:1]
final_result.insert(1, 'Predictions', predictions)
positive = final_result[final_result['Predictions'].isin([1])]
negative = final_result[final_result['Predictions'].isin([0])]


# Exportovani positive a negative data framu do csv souboru

pathPositive = os.path.join(os.getcwd(), 'positive_result.csv')
positive.to_csv(pathPositive, index=False)
pathNegative = os.path.join(os.getcwd(), 'negative_result.csv')
negative.to_csv(pathNegative, index=False)
```