Obsah

Obrázky

Načítajte data, predspracujte ich, natrénujte jednoduchú site a vyhodnoťte ju:

Prezrite si stĺpce v database. Podľa popisu zistite, č isa v jednotlivých stĺpcoch nachádzajú outliery a odstráňte ich.

```
---------- Min ----------
danceability            -0.74
energy                0.000197
loudness               -47.046
speechiness              -0.128
acousticness              -0.99
instrumentalness         -0.929
liveness                 -0.828
valence                  -0.631
tempo                       0.0
duration_ms            -427346.0
popularity                -43.0
number_of_artists           1.0
explicit                  False
dtype: object
```

Obrázok 2Hodnoty pred odstránením outlierov (minimum)

```
---------- Max ----------
danceability               8.375
energy                      94.1
loudness                  29.422
speechiness                0.965
acousticness               0.996
instrumentalness           0.994
liveness                   0.997
valence                    0.995
tempo                    241.423
duration_ms         1930821300.0
popularity                 82.0
number_of_artists          19.0
explicit                   True
dtype: object
```

Obrázok 11Hodnoty pred odstránením outlierov (maximum)

```
---------- Min ----------
danceability              0.0
energy                0.000197
loudness               -47.046
speechiness               0.0
acousticness              0.0
instrumentalness          0.0
liveness              0.00967
valence                   0.0
tempo                     0.0
duration_ms          -427346.0
popularity                0.0
number_of_artists         1.0
explicit                False
dtype: object
```

Obrázok 3Hodnoty po odstránením outlierov (maximu)

```
---------- Max ----------
danceability              0.975
energy                      1.0
loudness                 -0.116
speechiness               0.965
acousticness              0.996
instrumentalness          0.994
liveness                  0.997
valence                   0.995
tempo                   241.423
duration_ms         1930821300.0
popularity                82.0
number_of_artists         19.0
explicit                  True
dtype: object
```

Obrázok 4Hodnoty po odstránením outlierov (minimum)

Outliery museli byť odstránené v nasledujúcich stĺpcoch: danceability, loudness, energy, speechiness, acousticness, instrumentalness, liveness, valence, popularity.

V nasledujúcich riadkov môžeme vidieť ako som to robila.

```
df = df[(df['danceability'] <= 1) & (df['danceability'] >= 0)]
df = df[df['loudness'] <= 0]
df = df[df['energy'] <= 1]
df = df[df['speechiness'] >= 0]
df = df[df['acousticness'] >= 0]
df = df[df['instrumentalness'] >= 0]
df = df[df['liveness'] >= 0]
df = df[df['valence'] >= 0]
df = df[df['popularity'] >= 0]
```

Odstráňte stĺpce, ktoré sa nedajú použiť pri ďalšom spracovaní a null hodnoty

Najprv musela som pozrieť že koľko hodnôt chýba v daných stĺpcov.

```
******************** Missing values ********************
Lenght of dataset: 11727
danceability           0
energy                 0
loudness               0
speechiness            0
acousticness           0
instrumentalness       0
liveness               0
valence                0
tempo                  0
duration_ms            0
popularity             0
number_of_artists    121
explicit               0
name                   0
url                    0
genres                 0
filtered_genres        0
top_genre            168
emotion                0
dtype: int64
```

Obrázok 5 Chýbajúce údaje

Odstránila som top_genre lebo tam chýbalo najviac dát s tým spôsobom:

```
df_with_topgenre = df[df['top_genre'].notnull()]
df.drop(columns=['top_genre'], inplace=True)
df.dropna(inplace=True)
```

Po odstránením už nebudú chýbať údaje a krásne vidíme, že už nebude chýbať žiadny dát, viď. Ďalší obrázok.

```
******************** Missing values after removing them ********************
Lenght of dataset: 11606
danceability         0
energy               0
loudness             0
speechiness          0
acousticness         0
instrumentalness     0
liveness             0
valence              0
tempo                0
duration_ms          0
popularity           0
number_of_artists    0
explicit             0
name                 0
url                  0
genres               0
filtered_genres      0
emotion              0
dtype: int64
```

Obrázok 6 Chýbajúce hodnoty po odstránení

Nečíselné stĺpce vhodne zakódujte

```
******************** Column types ********************
danceability         float64
energy               float64
loudness             float64
speechiness          float64
acousticness         float64
instrumentalness     float64
liveness             float64
valence              float64
tempo                float64
duration_ms          float64
popularity           float64
number_of_artists    float64
explicit                bool
name                  object
url                   object
genres                object
filtered_genres       object
emotion               object
dtype: object
```

Obrázok 7 Typy stĺpcov

Najprv museli sme dozvedieť že jednotlivé stĺpce sú čísla alebo nie. S label encodingom zakódovali sme emotion, lebo je to objekt a ešte to budeme aj ďalej potrebovať. Mohli by sme to isté aj robiť aj pre explicit, name, url, genres, filtered_genress, lebo buď sú objecty alebo je explicit.

```
# Label encoding for emotion
le = LabelEncoder()
df_with_topgenre['labelEncoding'] = le.fit_transform(df_with_topgenre['emotion'])
print("*"*20, "Label encoding", "*"*20)
print(df_with_topgenre[['emotion', 'labelEncoding']].head(10))
emotions_df = df_with_topgenre['emotion']
```

Výstup je nasledovný:

```
******************** Label encoding ********************
     emotion  labelEncoding
0      happy              2
1      happy              2
2  energetic              1
3      happy              2
4      happy              2
5  energetic              1
6      happy              2
7      happy              2
8      happy              2
9  energetic              1
```

Obrázok 8 Label encoding

Pre emotional dummy encoding (one-hot) dá taký výstup:

```
******************** Dummy encoding ********************
     emotion  happy  energetic    sad   calm
0      happy   True      False  False  False
1      happy   True      False  False  False
2  energetic  False       True  False  False
3      happy   True      False  False  False
4      happy   True      False  False  False
5  energetic  False       True  False  False
6      happy   True      False  False  False
7      happy   True      False  False  False
8      happy   True      False  False  False
9  energetic  False       True  False  False
```

Obrázok 9 Dummy encoding

Kód je nasledovný:

```
# Dummy (one-hot) encoding for emotion
df_with_topgenre = pd.get_dummies(df_with_topgenres, columns=['emotion'],
prefix='', prefix_sep='')
df_with_topgenre['emotion'] = emotions_df
myemotions = list(df_with_topgenres['emotion'].unique())
show_columns = ['emotion'] + myemotions
print("*"*20, "Dummy encoding", "*"*20)
print(df_with_topgenre[show_columns].head(10))
```

Vytvorte vstupné X a výstupné y dátové množiny. Vo vhodnom pomere rozdeľte dáta na trénovaciu, validačnú a testovaciu množinu

V prvom rade som vymazal stĺpce, ktoré nebudeme potrebovať, pretože niektoré prvky stĺpcov nie sú čísla, ale iné typy prvkov.

```
df.drop(columns=['explicit'], inplace=True)
df.drop(columns=['name'], inplace=True)
df.drop(columns=['url'], inplace=True)
df.drop(columns=['genres'], inplace=True)
df.drop(columns=['filtered_genres'], inplace=True)
```

Hodnoty X a y som nastavil na základe stĺpca emócií nasledovne.

```
X = df.drop(columns=['emotion'])
y = df['emotion']
```

Dáta rozdelila som na trénovaciu, validačnú a testovaciu množinu nasledovne:

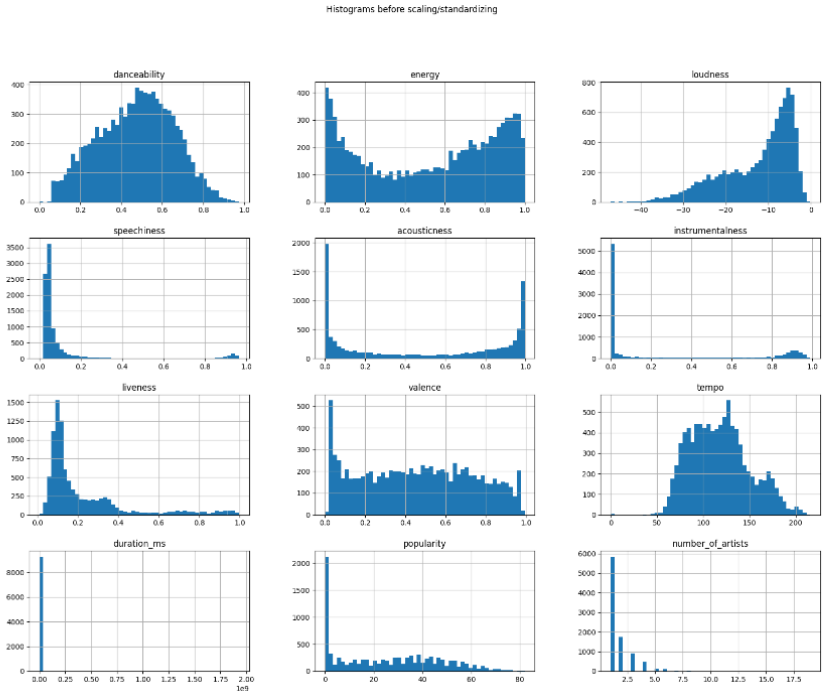
```
X_train, X_valid_test, y_train, y_valid_test = train_test_split(X, y, shuffle=True,
test_size=0.2, random_state=42)
X_valid, X_test, y_valid, y_test = train_test_split(X_valid_test, y_valid_test,
shuffle=True, test_size=0.5, random_state=42)
```

```
X_train: (9284, 12)
X_valid: (1161, 12)
X_test: (1161, 12)
y_train: (9284,)
y_valid: (1161,)
y_test: (1161,)
```

Obrázok 10 Rozdelenie trénovacie, validačné a testovacie dáta

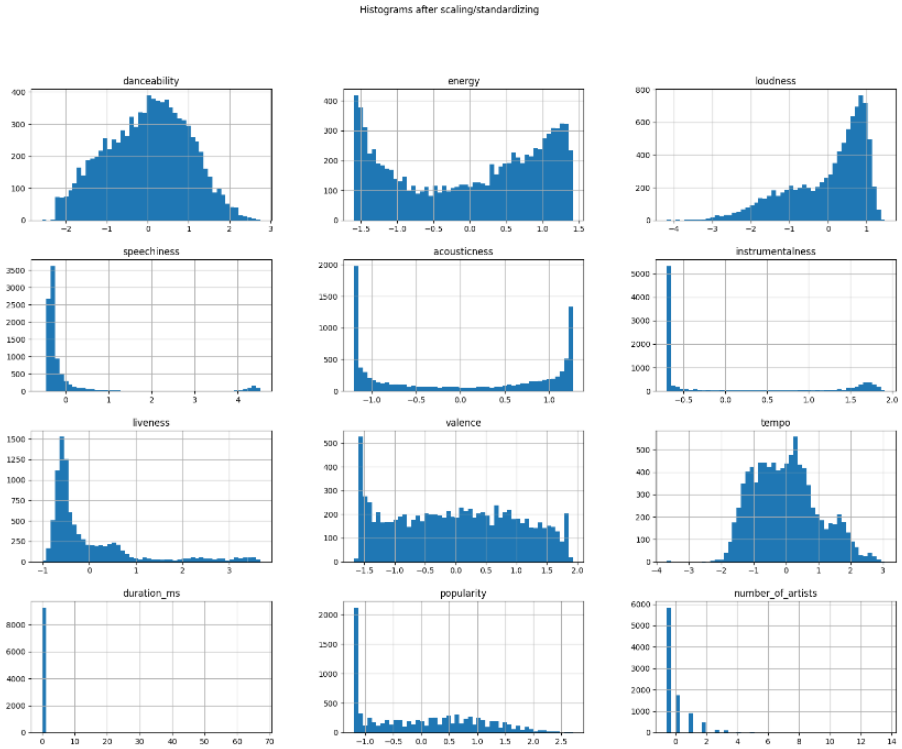
Dáta správne normalizujte, alebo škálujte

Pred škálovaním histogram vizeralo nasledovne:



Obrázok 11Histogram pred škálovaním

Po škálovaním histogram vizeralo nasledovne:



Obrázok 12Histogram po škálovaním

```
---------- Min ----------
danceability              0.000000
energy                    0.000197
loudness                -47.046000
speechiness               0.000000
acousticness              0.000000
instrumentalness          0.000000
liveness                  0.009670
valence                   0.000000
tempo                     0.000000
duration_ms          -359471.000000
popularity                0.000000
number_of_artists         1.000000
dtype: float64
```

Obrázok 13Maximum pred štandardizácie

```
---------- Max ----------
danceability          9.750000e-01
energy                1.000000e+00
loudness             -1.160000e-01
speechiness           9.650000e-01
acousticness          9.960000e-01
instrumentalness      9.910000e-01
liveness              9.970000e-01
valence               9.940000e-01
tempo                 2.159180e+02
duration_ms           1.930821e+09
popularity            8.200000e+01
number_of_artists     1.900000e+01
dtype: float64
```

Obrázok 14 Minimum pred štandardizácie

```
---------- Min ----------
danceability          -2.581676
energy                -1.588970
loudness              -4.190689
speechiness           -0.544224
acousticness          -1.197201
instrumentalness      -0.703670
liveness              -1.013139
valence               -1.663442
tempo                 -3.682719
duration_ms           -0.041501
popularity            -1.194423
number_of_artists     -0.559810
dtype: float64
```

Obrázok 15Maximum po štandardizácie

```
---------- Max ----------
danceability           2.751750
energy                 1.414767
loudness               1.484531
speechiness            4.525316
acousticness           1.268677
instrumentalness       1.917442
liveness               3.672029
valence                1.924472
tempo                  3.063932
duration_ms           67.145569
popularity             2.702533
number_of_artists     13.569615
dtype: float64
```

Obrázok 16Minimum po štandardizácie

Štandardizácia v kóde by vyzeralo nasledovne:

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_valid = scaler.transform(X_valid)
X_test = scaler.transform(X_test)

X_train = pd.DataFrame(X_train, columns=X.columns)
X_valid = pd.DataFrame(X_valid, columns=X.columns)
X_test = pd.DataFrame(X_test, columns=X.columns)
```

Natrénujte jednoduchú neurónovú sieť

V kóde nastavila som nasledovné veci:

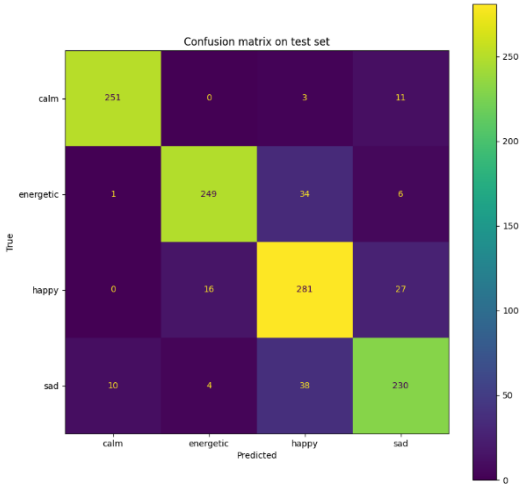
```
print("*"*20, "MLP", "*"*20)
print(f"Random accuracy: {1/len(y_train.unique())}")

clf = MLPClassifier(
    hidden_layer_sizes=(100, 100, 5, 6, 90),
    random_state=1,
    max_iter=100,
    validation_fraction=0.2,
    early_stopping=True,
    learning_rate='adaptive',
    learning_rate_init=0.001,
).fit(X_train, y_train)
```

Hodnota pre Random accuracy vyšlo: 0.25

Random accuracy: 0.25

Natrénovanú sieť vyhodnoťte na trénovacej a testovacej množine:

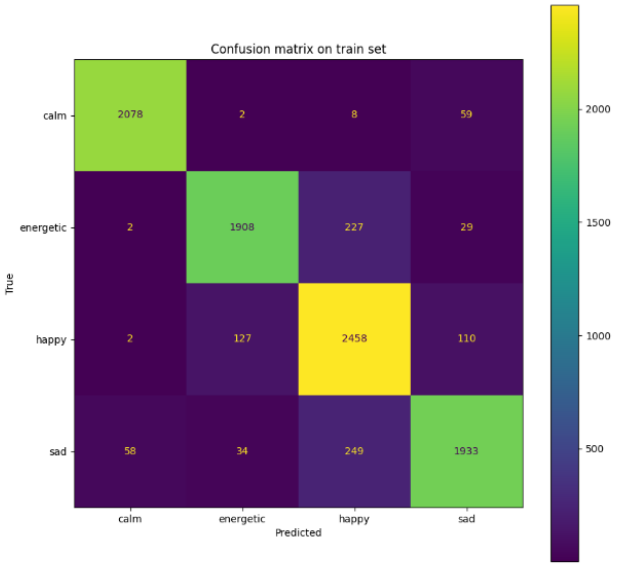
Confusion matrix pre testovacie dáta bude vyzerať nasledovne:



Obrázok 17Confusion matrix pre testovacie dáta

Že sa nám to podarilo, môžete vidieť aj na obrázku, keďže väčšina prvkov je umiestnená na hlavnej uhlopriečke. Čím viac dominuje žltkastá farba, tým presnejší výsledok sa nám podarilo dosiahnuť.

Confusion matrix pre trénovacie dáta bude vyzerať nasledovne:



Obrázok 18Confusion matrix pre trénovacie dáta

```
MLP accuracy on train set:  0.9023050409306334
MLP accuracy on test set:  0.8708010335917312
```

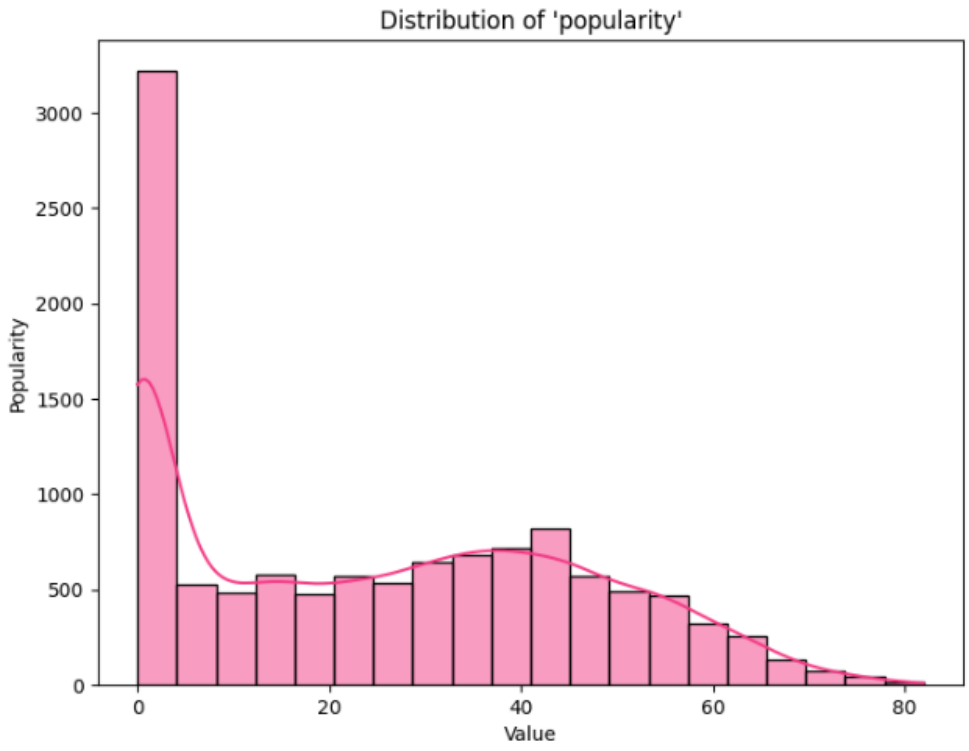
Obrázok 19MLP accuracy pre trénovacie a testovacie dáta

Kód, s ktorým sa mi podarilo dosiahnuť tento výsledok, je nasledovný:

```
y_pred = clf.predict(X_train)
print('MLP accuracy on train set: ', accuracy_score(y_train, y_pred))
cm_train = confusion_matrix(y_train, y_pred)
y_pred = clf.predict(X_test)
print('MLP accuracy on test set: ', accuracy_score(y_test, y_pred))
cm_test = confusion_matrix(y_test, y_pred)
class_names = list(le.inverse_transform(clf.classes_))
disp = ConfusionMatrixDisplay(confusion_matrix=cm_train,
display_labels=class_names)
fig, ax = plt.subplots(figsize=(10,10))
disp.plot(ax=ax)
disp.ax_.set_title("Confusion matrix on train set")
disp.ax_.set(xlabel='Predicted', ylabel='True')
plt.show()
disp = ConfusionMatrixDisplay(confusion_matrix=cm_test, display_labels=class_names)
fig, ax = plt.subplots(figsize=(10,10))
disp.plot(ax=ax)
disp.ax_.set_title("Confusion matrix on test set")
disp.ax_.set(xlabel='Predicted', ylabel='True')
plt.show()
```

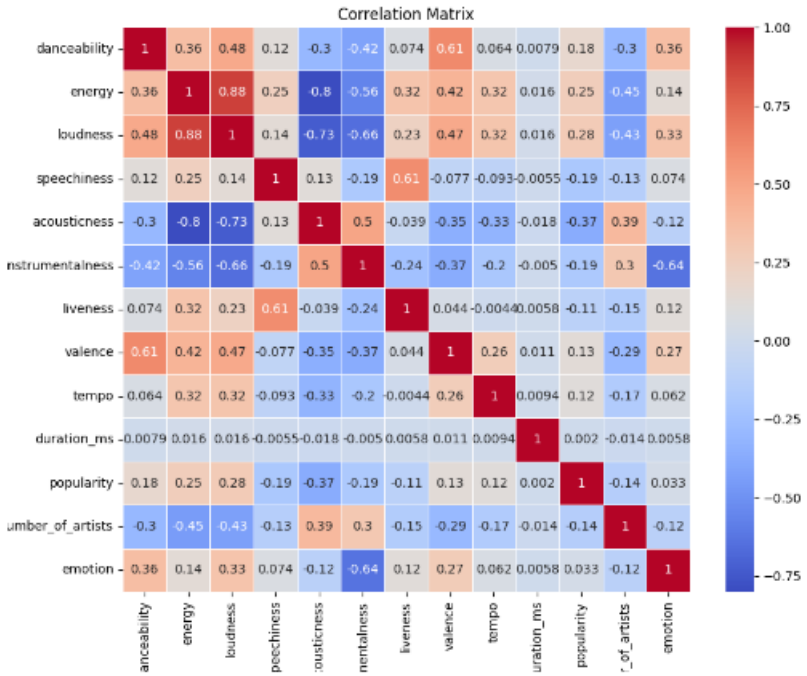
Analýza datasetu pomocou EDA

V prvom rade som začal jednoduchým grafom, ktorý je histplot a skúma popularity. Tu ma zaujímalo, aké sú najvyššie a najnižšie čísla.



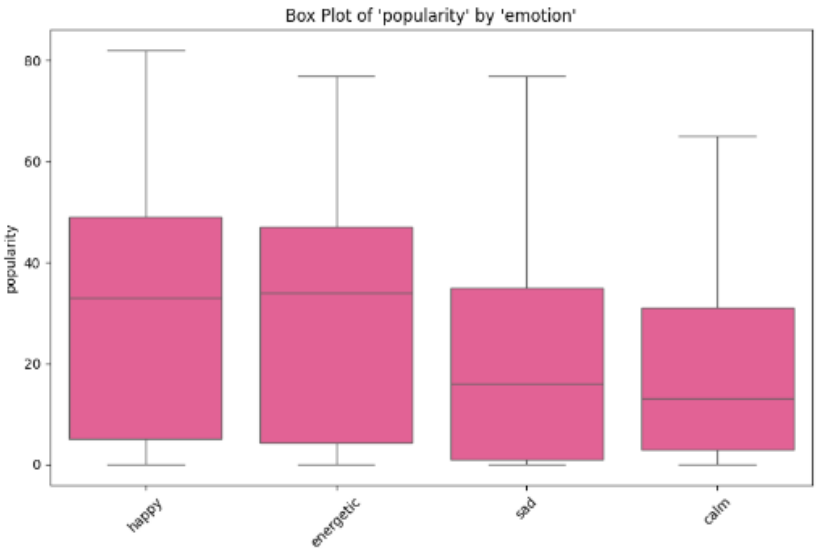
Obrázok 20 Graf pre popularity

Ďalej je Correlation Matrix:



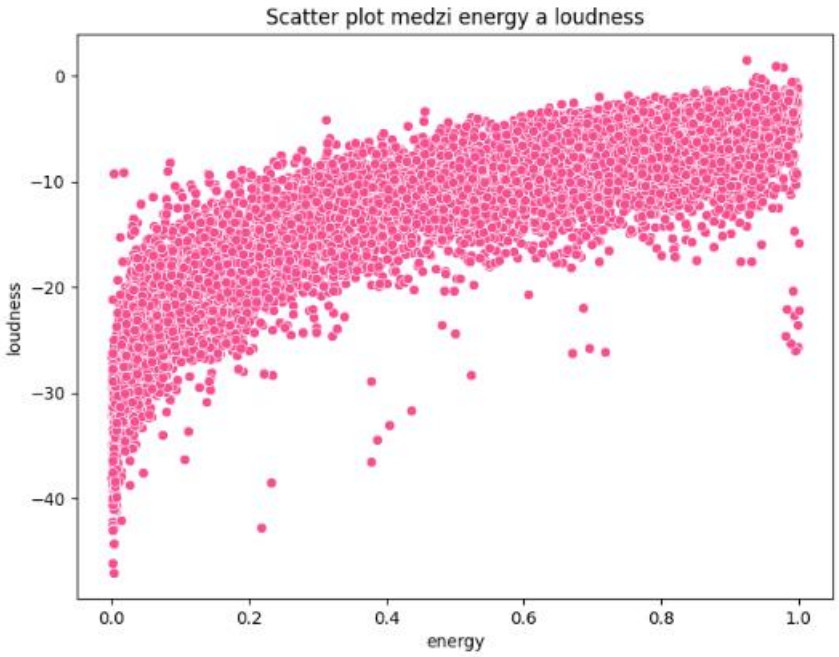
Obrázok 21Confusion matrix

Nasledujóuci graf bude typu boxplot. Pomocou tohto grafu skúmam, aké populárne sú jednotlivé kategórie emócií.



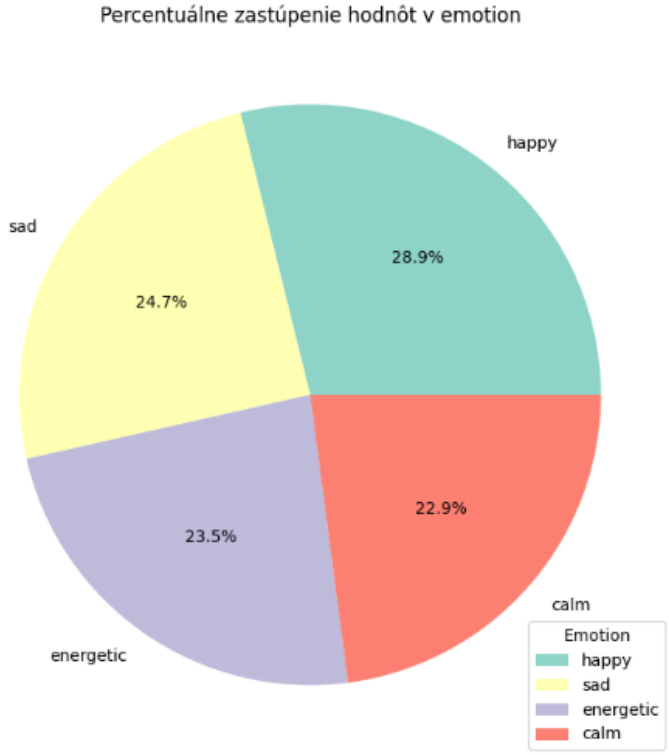
Obrázok 22 Boxplot

Scatter plot medzi energy a loudness:



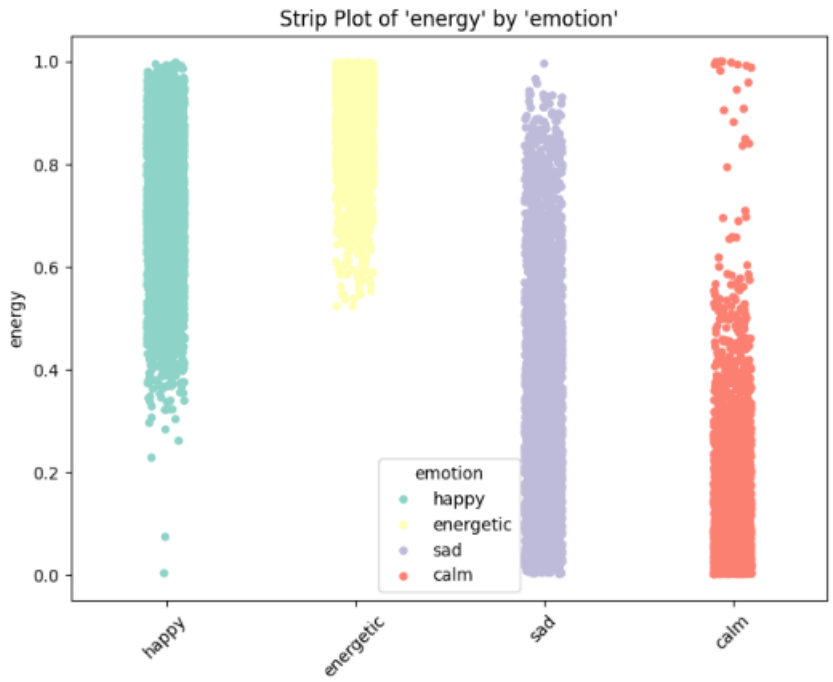
Obrázok 23Scatter plot

Percentuálne zastúpenie hodnôt v emotion:



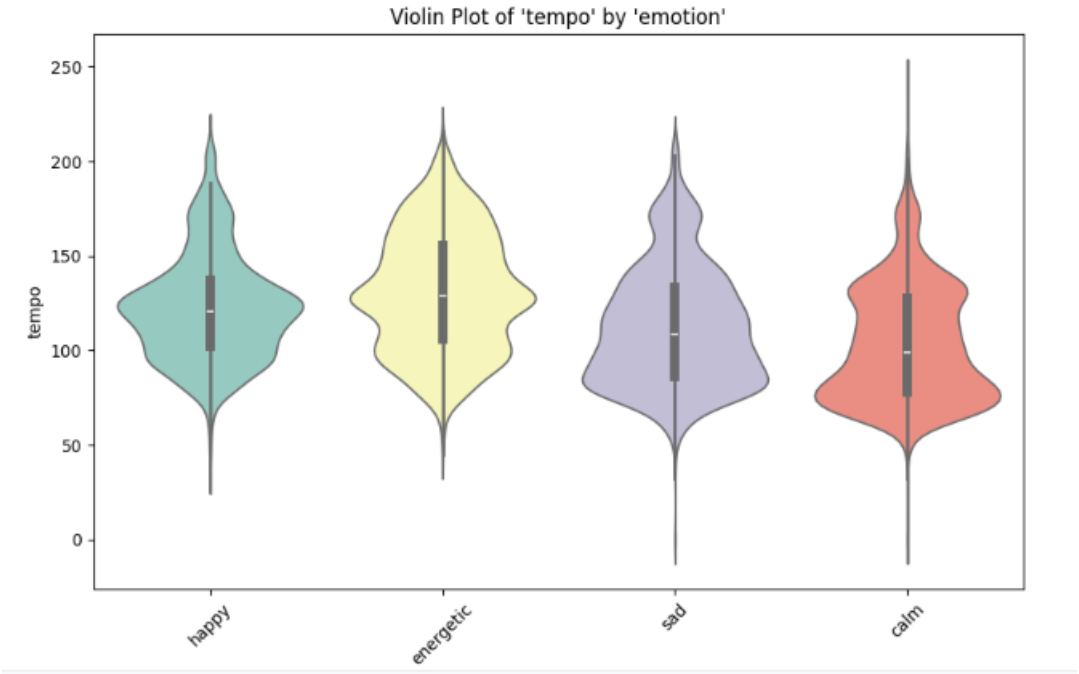
Obrázok 24 Pie plot

Strip plot pre emotion pomocou energy:



Obrázok 25Strip plot

Violin plot pre emotion pomocou tempo:

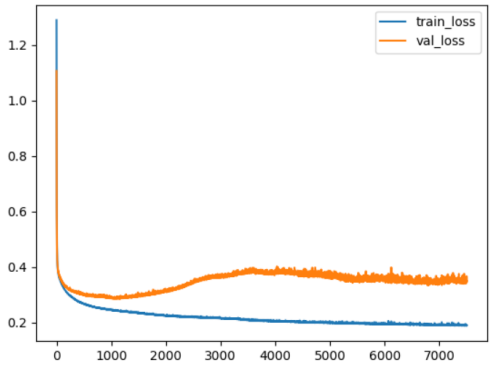


Obrázok 26Violin plot

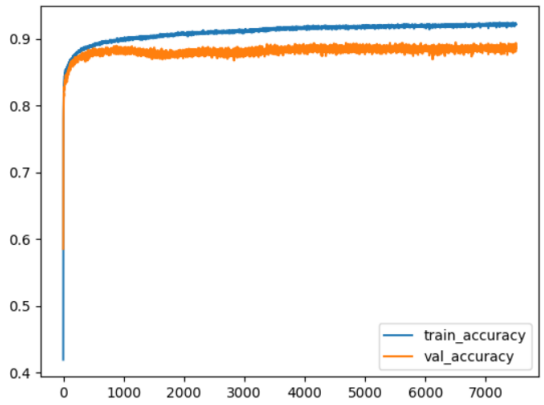
Natrénovanie neurónový sieť

Zvoľte architektúru a nastavenia hyperparametrov tak, aby ste dosiahli pretrénovanie. Demonštrujte pomocou grafov priebehu trénovanie, vyhodnotenia úspešností a konfúznej matice.

Prvý pokus. Mala som nastavené epochs na 7500, batch_size na 32 a learning_rate na 0.0002

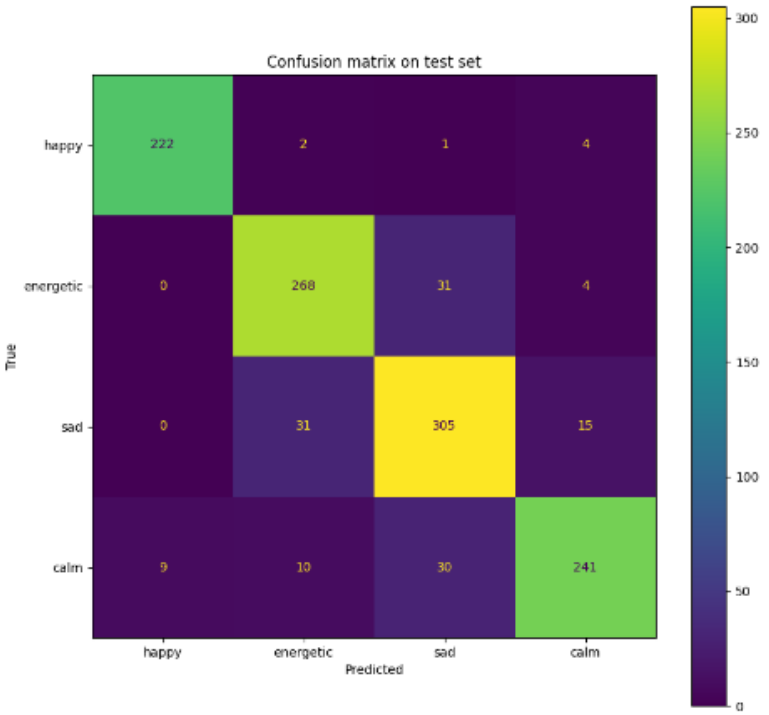


Obrázok 28Loss 1. pokus



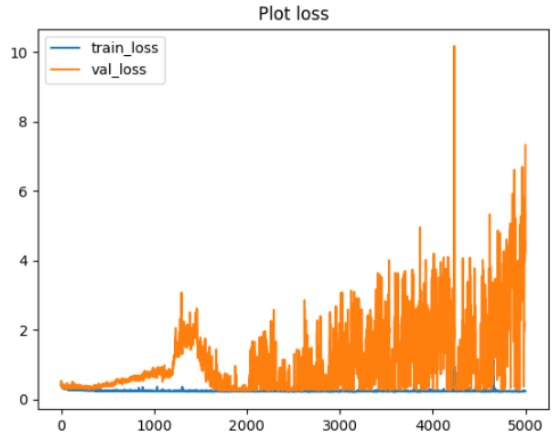
Obrázok 27Accuracy 1. pokus

Pri prvom pokuse Confusion matrix on test set dal dobré hodnoty, lebo výrazne vidíme hlavný diagonál. Žltá farba dominuje na hlavnej diagonále, a to znamená že na hlavnej diagonále máme najväčšie čísla, dostali sme dobrý výsledok.

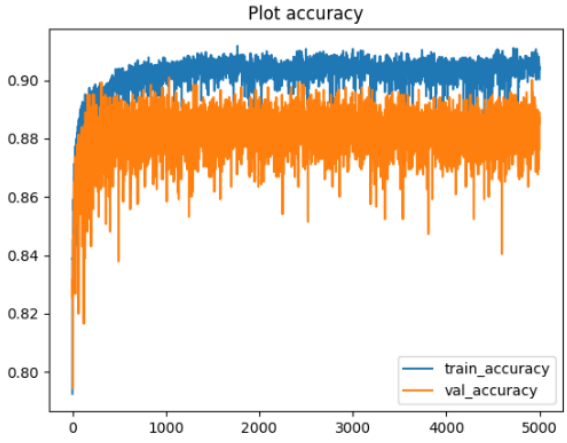


Obrázok 29Konfúzna matica 1. pokus

Pri druhom pokuse mala som hodnoty takto nastavené, že epochs na 5000, batch_size na 32 a learning_rate na 0.02. Nasledované grafy budú oveľa horšie vyzerať ako predošlé.

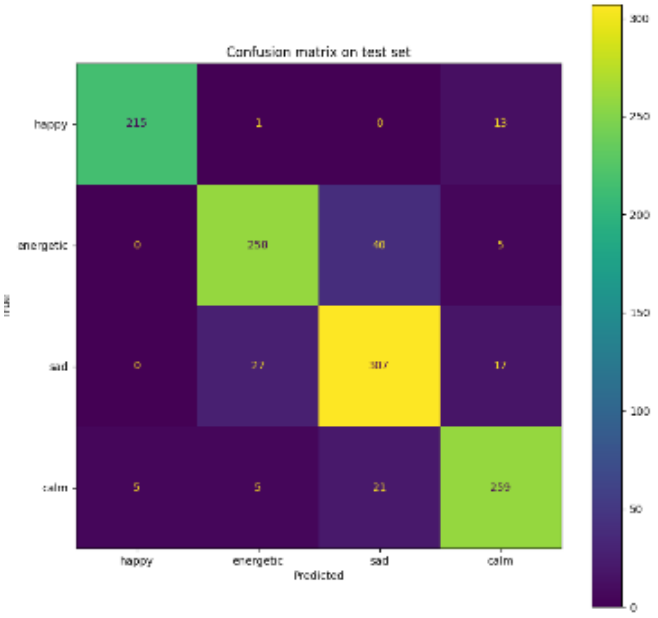


Obrázok 31Loss 2. pokus



Obrázok 30Accuracy 2. pokus

Pri 2. pikuse konfúzna matica bude celkom istá ako pri prechádzajúcej pokuse.

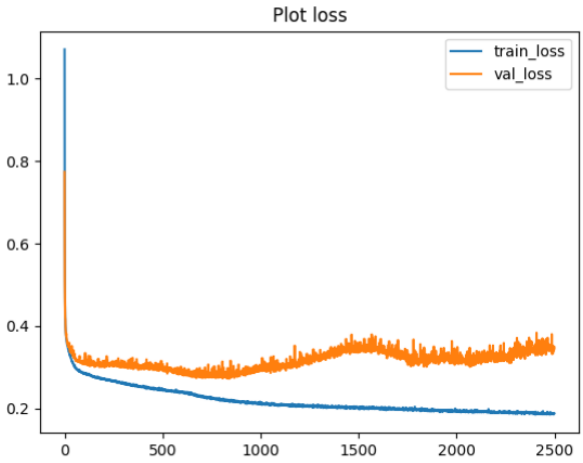


Obrázok 32Konfúzna matica 2. pokus

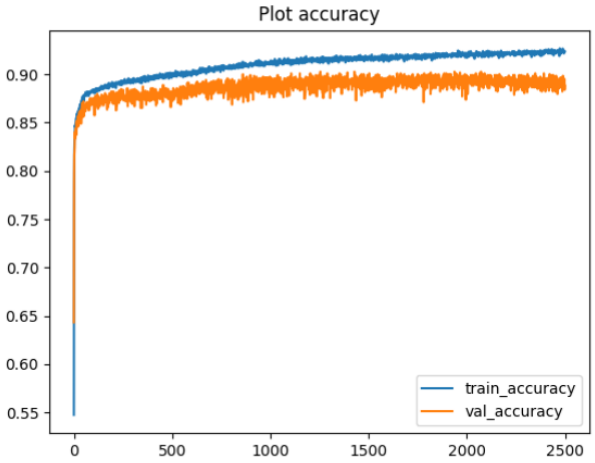
```
******************** Test accuracy ********************
Test accuracy: 0.8858
******************** Train accuracy ********************
Train accuracy: 0.9039
```

Obrázok 33Test a Train accuracy

Pri treťom pokuse mala som hodnoty takto nastavené, že epochs na 2500, batch_size na 42 a learning_rate na 0.001.

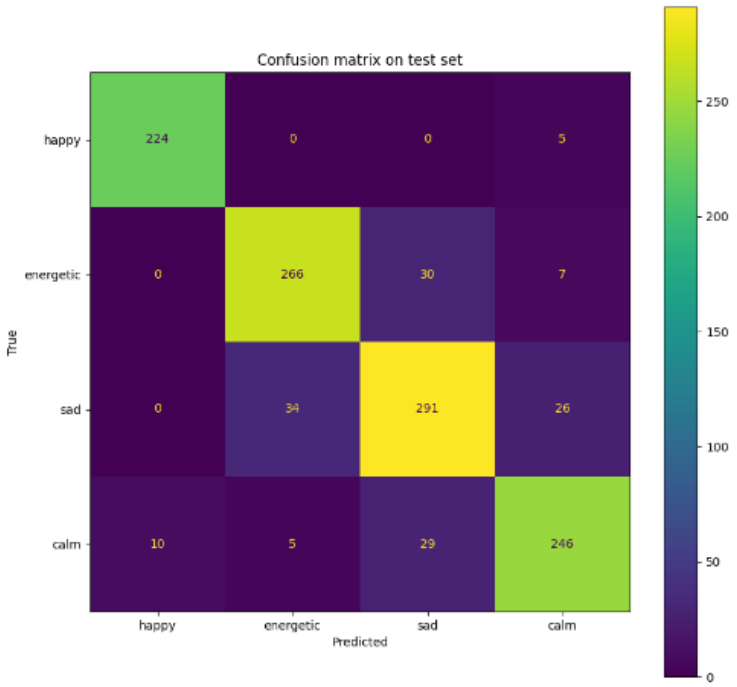


Obrázok 35Loss 3. pokus



Obrázok 34Accuracy 3. pokus

Kongruenčná matica bude vyzerať nasledovne:



Obrázok 36Kongruenčná matica 3. pokus

```
******************** Test accuracy ********************
Test accuracy: 0.8755
******************** Train accuracy ********************
Train accuracy: 0.9282
```

Obrázok 37Test a Train accuracy

Kód s čím som nastavila jednotlivé premenné je nasleddovné:

```
model = Sequential()
model.add(Dense(24, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(15, activation='relu'))
model.add(Dense(4, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.001),
metrics=['accuracy'])
history = model.fit(x=X_train, y=y_train, validation_data=(X_valid, y_valid),
epochs=2500, batch_size=42)
```

Odstráňte pretrénovanie tak, že do trénovania zavediete EarlyStopping pre skoré zastavenie trénovania

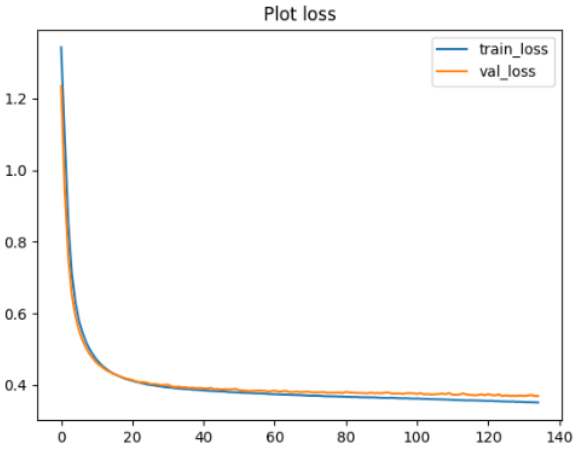
Pomocou EarlyStopping môžeme zastaviť trénovanie. Do kódu bolo potrebné pridať iba jeden riadok, ktorý vyzerá takto:

```
earlystopping = EarlyStopping(monitor='val_loss', patience=10, verbose=1,
restore_best_weights=True)
```

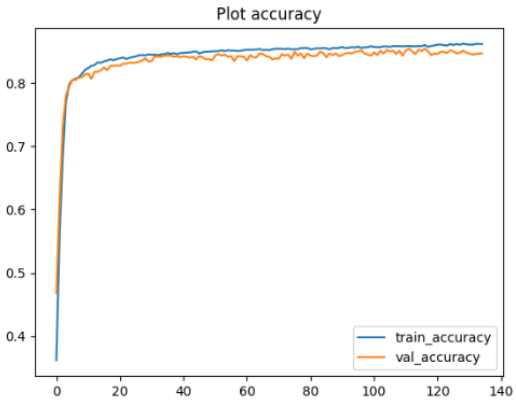
V predchádzajúcej úlohe bola definovaná premenná. Musíme túto premennú trochu zmeniť, čo urobíme takto:

```
history = model.fit(x=X_train, y=y_train, validation_data=(X_valid, y_valid),
epochs=2500, batch_size=32,callbacks=[earlystopping])
```

Mala som hodnoty takto nastavené, že epochs na 2500, batch_size na 32 a learning_rate na 0.0002.



Obrázok 38 Loss a EarlyStopping



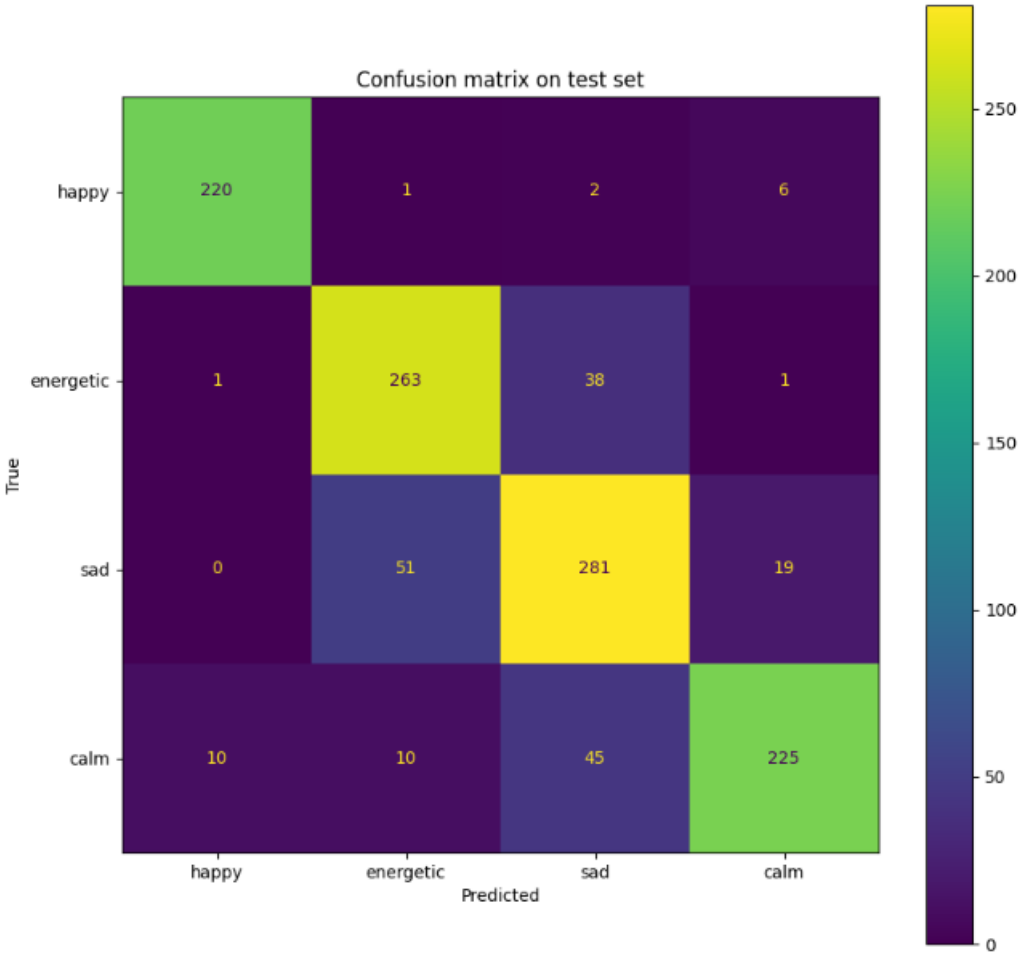
Obrázok 39Accuracy a EarlyStopping

Ako môžete vidieť na grafoch, pomocou EarlyStopping sa namiesto 2500 epoch program zastavil na 135 epochách.

```
Epoch 135: early stopping
```

Obrázok 40Early Stopping

Matica zmätku bude v tomto cvičení vyzerať takto:



Obrázok 41Konfúzna matica a Early Stopping

Na obrázku môžete vidieť aj to, že pomocou Early Stopping sa nám podarilo dosiahnuť celkom dobré výsledky. Je vidieť, že na hlavnej uhlopriečke viac dominuje žltá farba, z toho vieme aj to, že čím vyššie čísla sú na hlavnej uhlopriečke, tak nám vyšiel celkom dobrý výsledok.

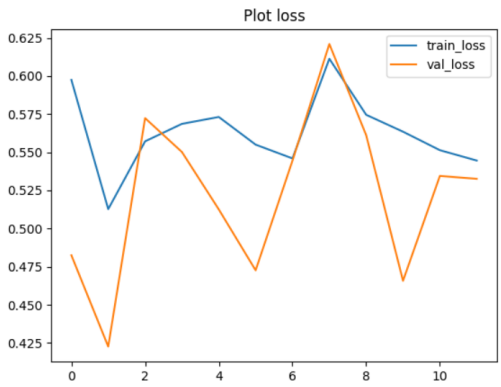
```
********** Test accuracy **********
Test accuracy: 0.8431
********** Train accuracy **********
Train accuracy: 0.8615
```

Obrázok 42Test a Train accuracy a Early Stopping

Zmena Hyperparametrov

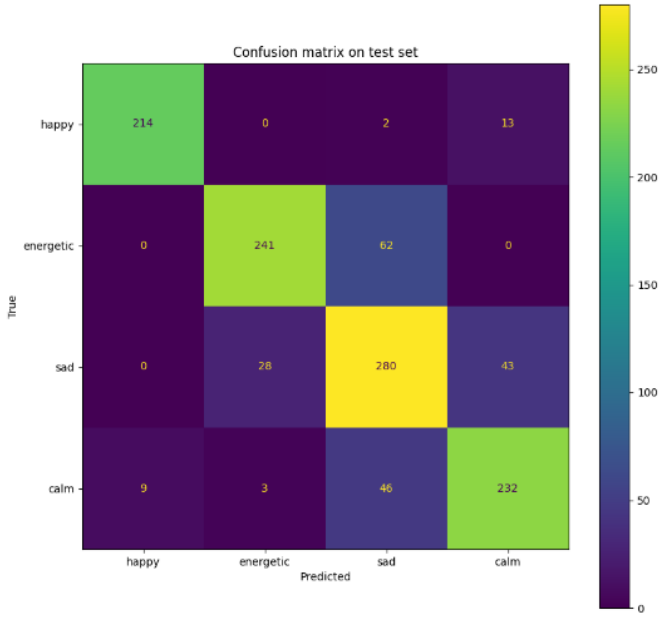
	Learning rate	Epochs	Batch size	Train accuracy	Test accuracy	Early Stopping	Počet neuróny
1. pokus	0.2	2500	32	0.8325	0.8244	12	24+15+4
2. pokus	0.0002	2500	10	0.8779	0.8679	150	24+15+4
3. pokus	0.001	700	50	0.8787	0.8636	100	24+15+4
4. pokus	0.5	100	32	0.2888	0.2992	25	24+15+4
5. pokus	0.1	100	32	0.8485	0.8389	21	24+10+4

1. Pokus



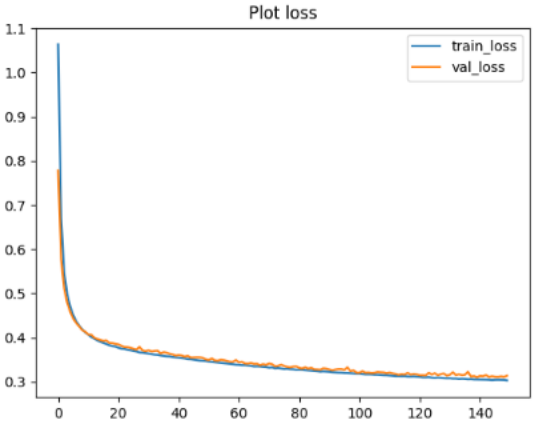
Obrázok 44Loss 1. pokus zmena

Obrázok 43Accracy 1. pokus zmena

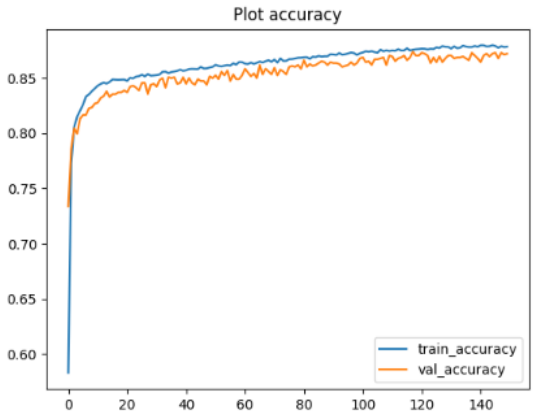


Obrázok 45Konfúzna matica 1. pokus zmena

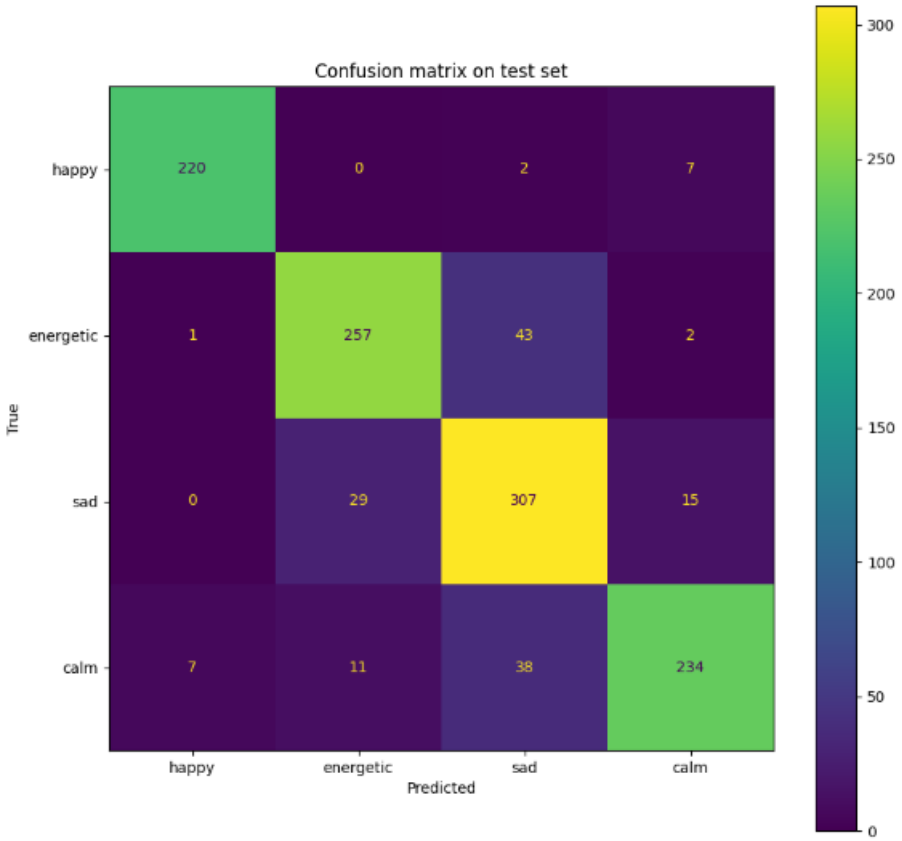
2. Pokus



Obrázok 47Loss 2. pokus zmena

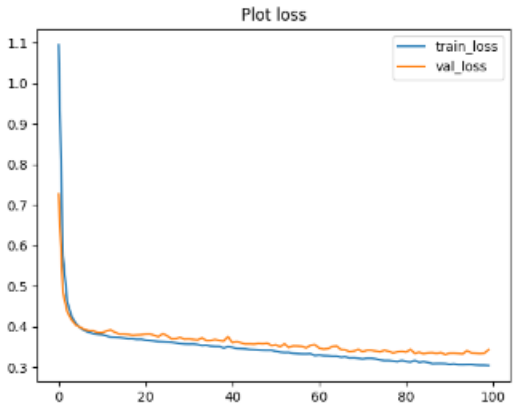


Obrázok 46 Accuracy 2. pokus zmena

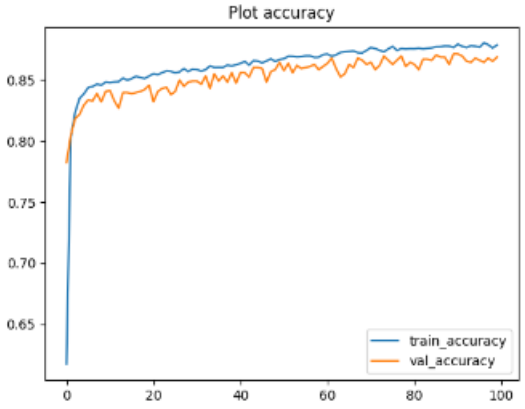


Obrázok 48Konfúzna matica 2. pokus zmena

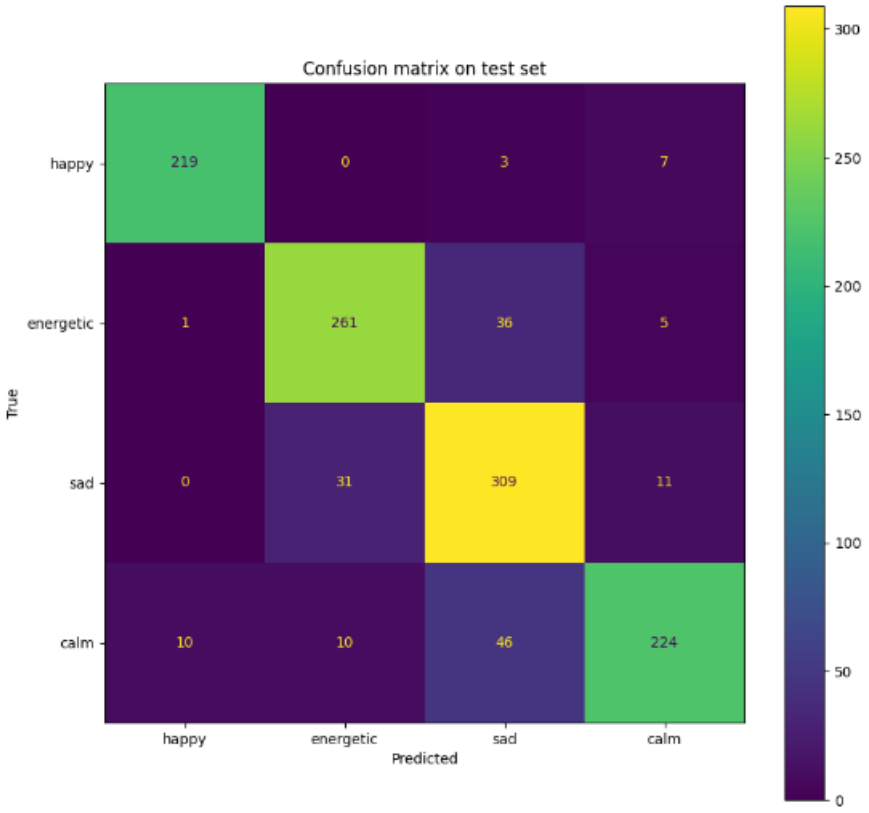
3. Pokus



Obrázok 50Loss 3. pokus zmena

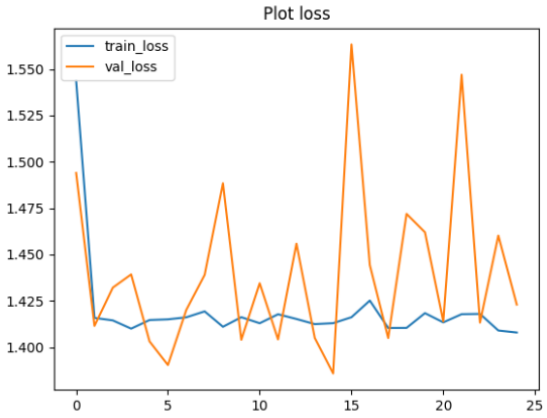


Obrázok 49 Accuracy 3. pokus zmena

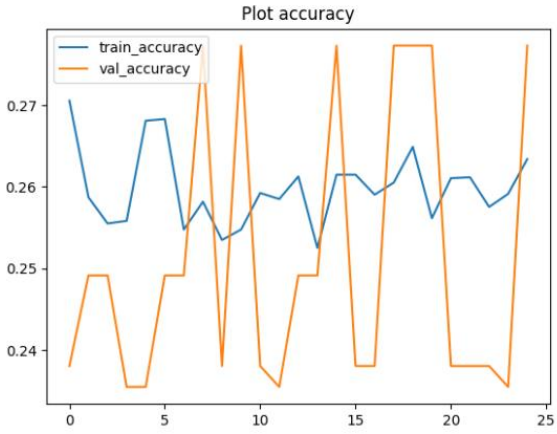


Obrázok 51Konfúzna matica 3. pokus zmena

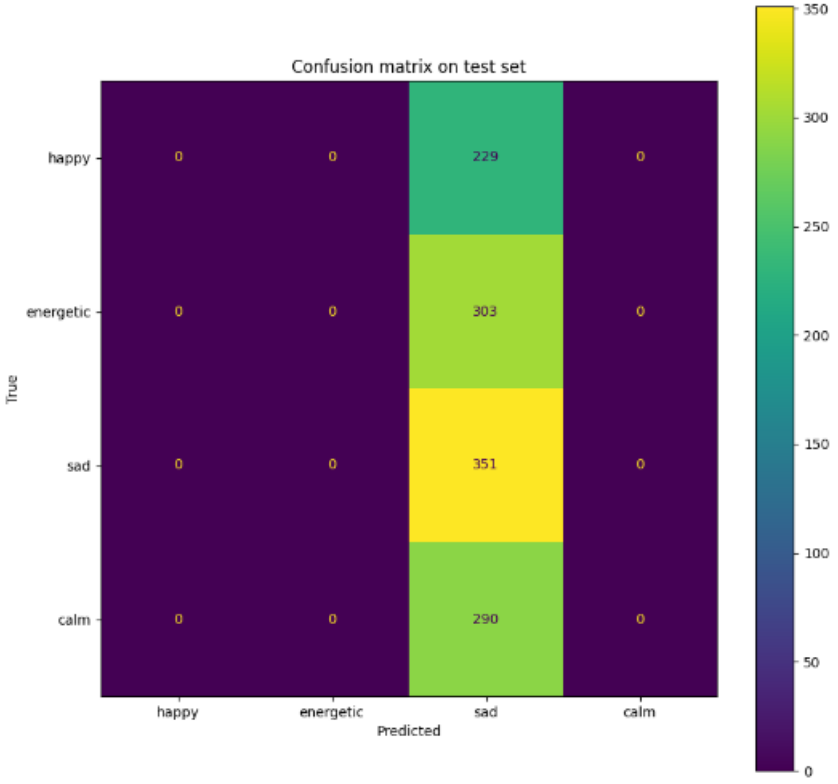
4. Pokus



Obrázok 53Loss 4. pokus zmena



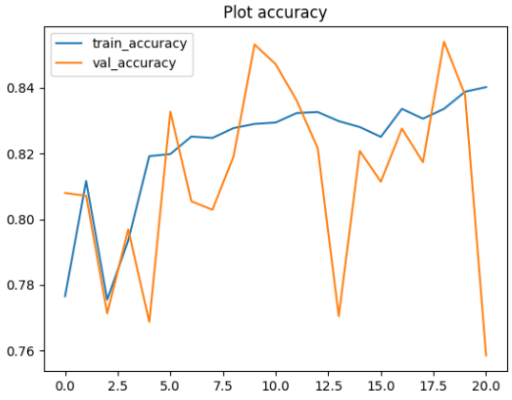
Obrázok 52Accuracy 4. pokus zmena



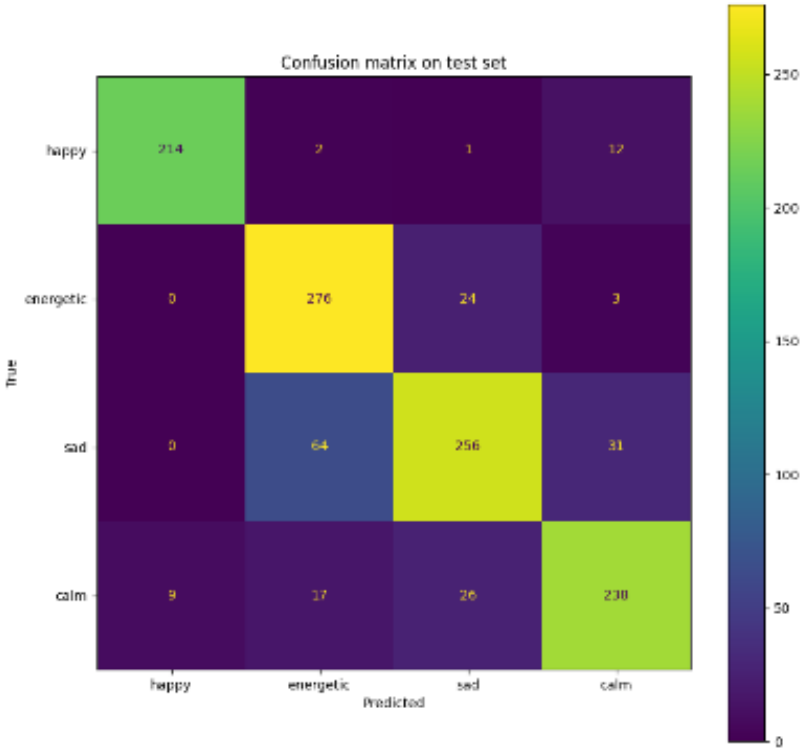
Obrázok 54Konfúzna matica 4. pokus zmena

5. Pokus

Obrázok 56Loss 5. pokus zmena



Obrázok 55 Accuracy 5. pokus zmena



Obrázok 57Konfúzna matica 5. pokus zmena

Zhrnutie tretej časti

Na základe mojich skúseností som si uvedomil, že čím menší je miera učenia, tým lepšiu hodnotu získame. Ak je teda napríklad miera učenia 0.1, dostanem horší výsledok, ako keď je miera učenia napríklad 0.00001. Zmenil som počet epochs, learning_rate a batch_size, len v poslednom bode to zostalo rovnaké, až na jeden prípad, kedy som zmenil aj počet neurónov.

Mali by sme tiež spomenúť, koľko z toho, čo presne nastavujem pri nastavovaní buniek neurónov. Prvý: 24, input_dim=X_train.shape[1], aktivácia='relu', druhý 15, aktivácia='relu', tretí 4, aktivácia='softmax'. Pri treťom sme to určite museli nastaviť na 4, lebo inak by to nešlo.

Ďalšie nastavenia, ktoré som urobil potom: model.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.0002), metrics=['accuracy']) a ešte jedno dôležité nastavenie bolo history = model.fit(x=X_train , y=y_vlak, validation_data=(X_valid, y_valid), epochs=100, batch_size=32, callbacks=[earlystopping]). Zmenil som hodnoty v týchto dvoch riadkoch, aby som zistil, čo by bolo najlepšie. V tomto prípade Adam nie je mužské meno, ale z keras.src.optimizers importujeme Adam, súčasť Kerasu.

Ďalej je dôležité spomenúť, že pred ukončením programu napíšem do konzoly nasledovné:

```
******************** Test accuracy ********************
Test accuracy: 0.8440
******************** Train accuracy ********************
Train accuracy: 0.8582
37/37 [==============================] - 0s 1ms/step
y_test: (1173, 4)
y_train: (9377, 4)
```

Obrázok 58Výpis na konci