SUNS – Zadanie 2:
Viacvrstvový perceptrón II.

Petra Kirschová

1 ANALÝZA A SPRACOVANIE DÁT

Na načítanie a prípravu dát som použila knižnicu `pandas`. Boli k dispozícii zvlášť trénovacie (39337 záznamov) a testovacie dáta (9835 záznamov).

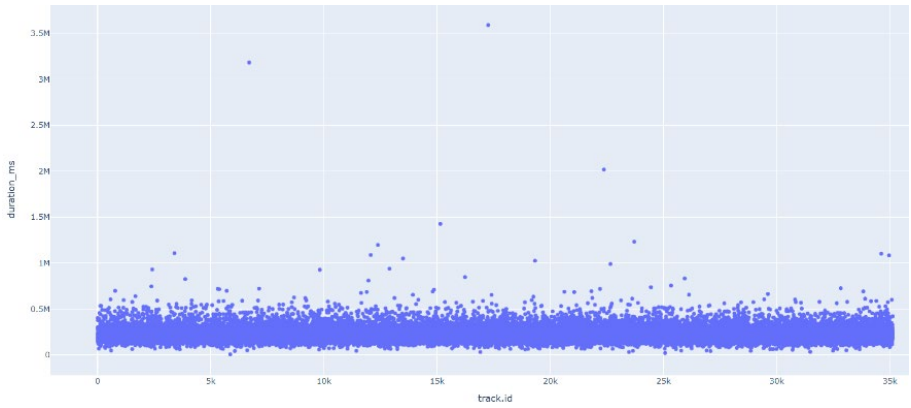
Z trénovacieho a testovacieho datasetu som odstránila:

- Stĺpce s id: *track.id, track.album.id, playlist_id*
- Stĺpec *playlist_subgenre:* pri trénovaní som zistila, že ak subgenre zostal vo vstupnej množine klasifikátora, trénovanie dosahovalo úspešnosť takmer 100%, pretože na základe subgenre sa dal ľahko určiť žáner.
  - Tieto stĺpce som odstránila kvôli tomu, že ich nepoužívam pri trénovaní.
- Riadky, ktoré obsahovali null hodnotu

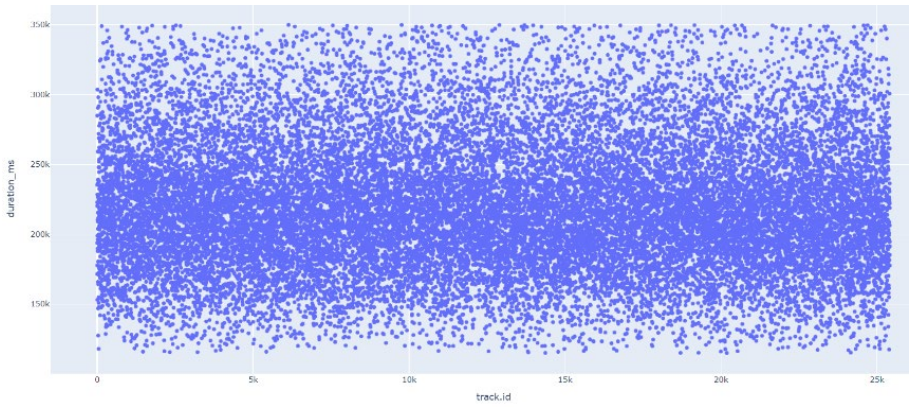
Z trénovacieho datasetu som odstránila:

- Outliers: Pri mazaní outliers som si najprv vykreslila grafy pre všetky číselné parametre (závislosť *track.id* od jednotlivých parametrov) a podľa toho som vymazala tie hodnoty, ktoré boli ďaleko od ostatných. Napríklad pre *duration_ms:*

duration_ms pred odstránením outliers:



duration_ms po odstránení outliers:



V trénovacom a testovacom datasete som nahradila:

- Žánre v stĺpci *playlist_genre* za číselné hodnoty. Priraďovanie hodnôt som robila manuálne, aby som s istotou vedela, akému žánru je aké číslo priradené (namapovaný žáner na index v poli):

```
genres = ["edm", "latin", "pop", "r&b", "rap", "rock"]
for val in genres:
      data_out.loc[
            data_out["playlist_genre"] == val, "playlist_genre"
      ] = genres.index(val)
```

- Všetky nečíselné hodnoty som skonvertovala na číselné reprezentácie:

```
for i in list_of_cols:
      data[i] = data[i].astype('category').cat.codes
```

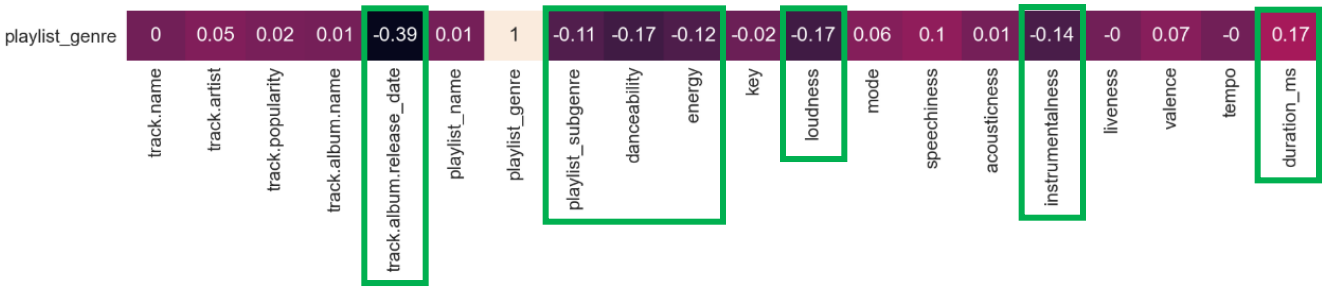
V trénovacom datasete som nahradila:

- V stĺpci *track.album.release_date* všetky dátumy rokmi, pretože sa tu nachádzali záznamy s celými dátumami a aj také, kde bol uvedený iba rok.

Po úprave trénovacieho datasetu v ňom zostalo 36146 trénovacích dát, teda zmazala som zhruba 3000 záznamov, čo vzhľadom na veľkosť datasetu nehrá veľkú rolu. Z trénovacích dát som vymazala iba 1 riadok s null hodnotou, teda obsahujú 9834 záznamov.

Pre trénovací dataset som vykreslila korelačnú maticu, na ktorej som sledovala parameter *playlist_genre*. Žáner má najväčšiu koreláciu s *track.album.release_date*. Má tiež dosť veľké korelácie aj s *danceability*, *loudness*, *duration_ms*, *playlist_subgenre*, *energy* a *instrumentalness*.

Korelácie playlist_genre s ostatnými ukazovateľmi:



Pri trénovaní s rôznymi vstupnými parametrami som neskôr zistila, že sieť trénuje najlepšie, keď sa použijú všetky ukazovatele (okrem *playlist_subgenre*), nie len tie, ktoré majú s *playlist_genre* vysokú koreláciu. Pre porovnanie, presnosť trénovania s použitím iba atribútov s vysokou koreláciou vs. celého datasetu:

Vysoká korelácia

```
categorical_accuracy: 0.4336
```

Celý dataset

```
categorical_accuracy: 0.5434
```

Následne sa trénovacie aj testovacie dáta normalizovali, aby sa hodnoty údajov dostali do intervalu <0,1>. Normalizovala som všetky dáta okrem stĺpca *playlist_genre*, keďže je výstupom neurónovej siete a normalizácia je v tom prípade nepotrebná.

2 VIACVRSTVOVÝ PERCEPTRÓN

Na trénovanie klasifikátora som použila knižnice `keras` a `tensorflow` a na vykreslenie grafov knižnice `seaborn` a `matplotlib`.

2.1 ROZDELENIE DÁT

Najprv som dataset rozdelia na vstupy (X) a výstup (Y) neurónovej siete.

X: Vstupnú množinu predstavujú všetky stĺpce datasetu, okrem *playlist_genre* a *playlist_subgenre*.

Y: Výstupnú množinu tvorí stĺpec *playlist_genre*.

Dáta boli už vopred rozdelené na trénovaciu a testovaciu množinu. Pri trénovaní som z trénovacej množiny použila 20% na validáciu.

2.2 NASTAVENIE KLASIFIKÁTORA

Model perceptrónu som inicializovala ako Sequential.

```
model = tf.keras.Sequential()
```

Potom som pridala vstupnú vrstvu tvorenú 18 neurónmi (vstupná množina obsahuje 18 ukazovateľov).

```
model.add(Input(shape=(in_size,)))
```

Model obsahuje 1 skrytú vrstvu typu Dense so 100 neurónmi, kde sa používa aktivačná funkcia `'relu'`.

```
model.add(Dense(100, activation='relu'))
```

Výstupná vrstva typu Dense obsahuje 6 neurónov (pretože v stĺpci *playlist_genre* sa nachádza 6 rôznych žánrov). Vo výstupnej vrstve sa používa aktivačná funkcia `'softmax'`.

```
model.add(Dense(out_size, activation='softmax'))
```

Klasifikátoru som nastavila loss funkciu na 'categorical_crossentropy', optimizer Adam s learning rate 0.0001 a metric na `'categorical_accuracy'`. Learning rate som oproti predvolenej hodnote zmenšila kvôli použitiu batchov pri trénovaní. Väčší learning rate viedol k pretrénovaniu.

```
classifier.compile(
            loss='categorical_crossentropy',
            optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
            metrics=['categorical_accuracy'])
```

Trénovanie prebiehalo v 500 epochách s veľkosťou batchov 100. Validačné dáta tvorili 20% z trénovacej množiny.

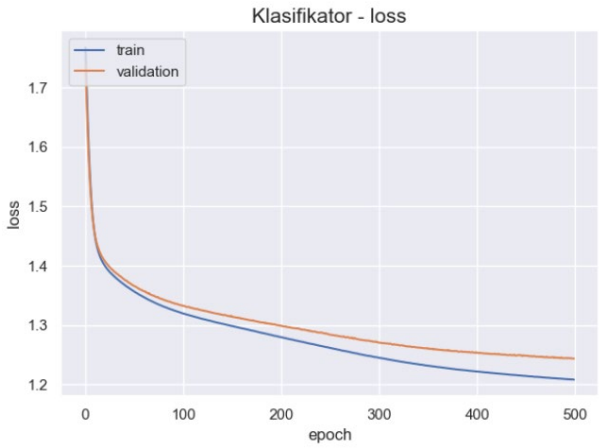
```
fit = classifier.fit(train_x_in, train_y_out, batch_size=100,epochs=500,
                     validation_split=0.2,use_multiprocessing=True)
```

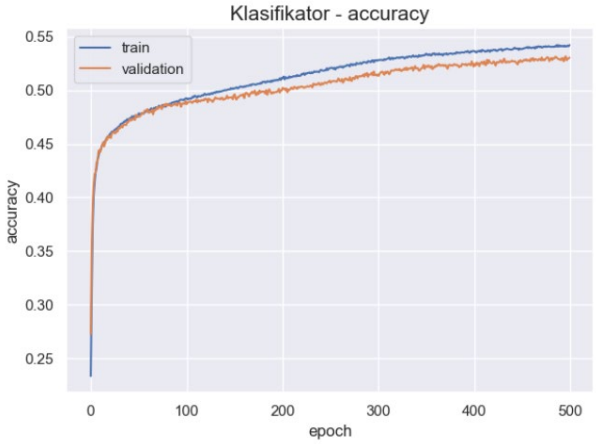
2.3 VÝSLEDKY TRÉNOVANIA

Pri horeuvedených nastaveniach modelu som dosiahla presnosť trénovania 54,2% a chybu 1,2. Presnosť validácie bola dosť blízko trénovacej – 53% a chyba 1,24.

```
loss: 1.2080 - categorical_accuracy: 0.5421 - val_loss: 1.2435 - val_categorical_accuracy: 0.5304
```

Priebežnú chybu aj presnosť som vykreslila do nasledovných grafov. Na grafoch je vidno, že chyby pri trénovaní a validácii a tiež presnosť trénovania a validácie dosahujú veľmi podobné hodnoty, teda validácia počas trénovania ukazuje to, že sieť bola trénovaná správne.



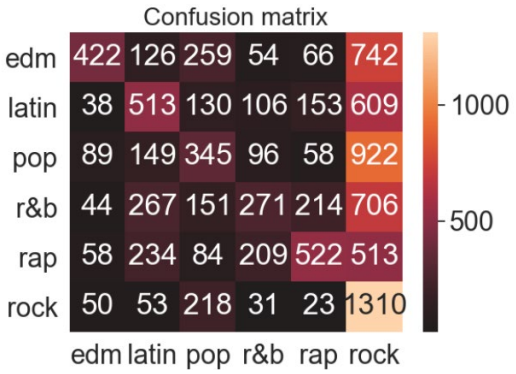


Celková úspešnosť trénovania, v priemere 50-55%, bola relatívne dobrá. Keď som sa snažila zmenou nastavení siete dosiahnuť vyššiu presnosť, spolu s presnosťou sa zvyšovala aj validačná chyba, teda vyššou presnosťou som nedosiahla spoľahlivejšie výsledky.

Vzhľadom na to, že testovacie dáta neboli upravované a obsahovali outliers, úspešnosť siete v predikcii žánrov z testovacích dát bola dosť nízka, okolo 35%.

```
loss: 1.9281 - categorical_accuracy: 0.3481
```

Pre výsledky klasifikátora som vykreslila confusion matrix. Na diagonále matice je počet správne zatriedených vzoriek do hudobných žánrov. Na opačnej diagonále je počet nesprávne zatriedených vzoriek. Z matice je vidno, že v zatrieďovaní vzoriek prevláda kategória rock, čo mohlo byť dôsledkom nevyváženého a neprečisteného testovacieho datasetu.

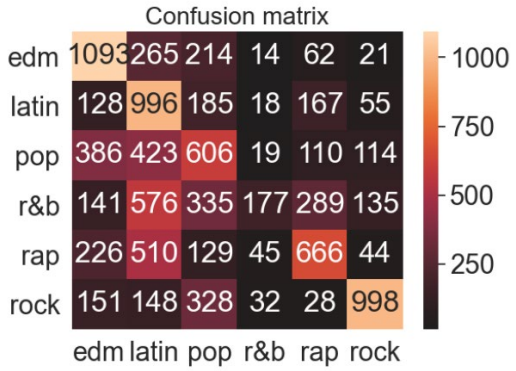


Skúsila som upraviť stĺpec *track.album.release_date* aj v testovacom datasete, teda nahradila som všetky dátumy rokmi. Po tejto úprave sa výsledky evaluácie zvýšili o vyše 10% a aj na confusion matrix bolo vidno, že sieť trénuje oveľa lepšie. Pôvodne nízke percento úspešnosti je teda spôsobené predovšetkým nekonzistentnými dátami v testovacom datasete.

Úspešnosť predikcie po upravení track.album.release_date:

```
loss: 1.3551 - categorical_accuracy: 0.4613
```

Confusion matrix po upravení track.album.release_date:



3 PRETRÉNOVANIE

Pretrénovanie siete som dosiahla zvýšením počtu neurónov v skrytej vrstve zo 100 na 500 a zväčšením parametra learning_rate z 0.0001 na 0.001. Bez použitia regularizačných metód bola presnosť predikcie výstupu z testovacích dát cca 33%. Presnosť trénovania sa vyšplhala až na takmer 70%, ale presnosť validácie zostala na 56%, teda je medzi nimi obrovský rozdiel, čo by nemalo za normálnych okolností nastať. Takisto rozdiel medzi trénovacou a validačnou chybou je veľmi veľký.

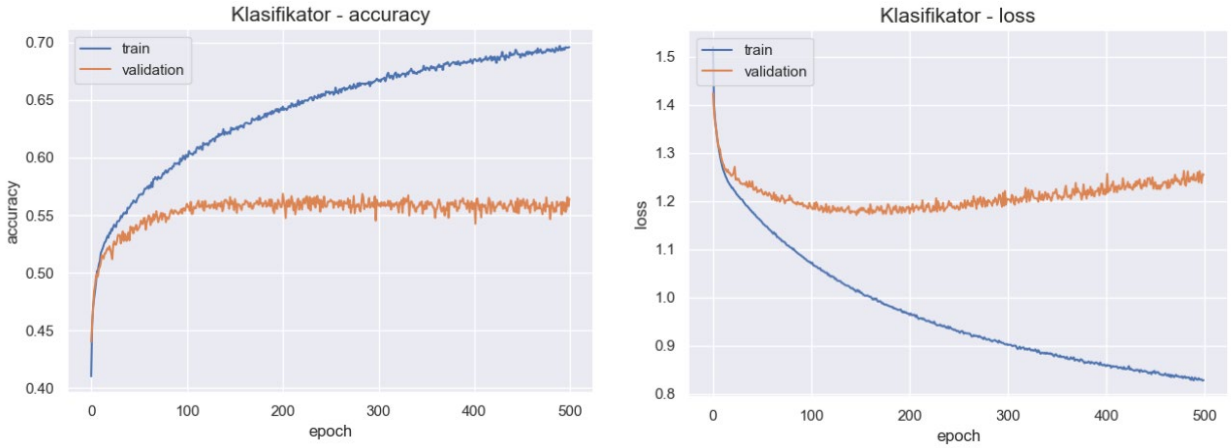
Predikcia výstupu:

```
loss: 2.8621 - categorical_accuracy: 0.3346
```

Výsledky trénovania a validácie:

```
loss: 0.8286 - categorical_accuracy: 0.6959 - val_loss: 1.2543 - val_categorical_accuracy: 0.5643
```

Priebeh presnosti a chyby pri trénovaní a validácii:



3.1 L1/L2 REGULARIZÁCIA

Princípom L1/L2 regularizácie je pridanie penalty váham v kriteriálnej funkcii. Pomocou tejto regularizácie sa mi podarilo rozdiely medzi trénovacími a validačnými chybami výrazne zmenšiť.

Pri L1/L2 regularizácii som model siete vytvorila nasledovne, s tým, že som menila typ regularizácie l1, l2 a l1_l2:

```
model = tf.keras.Sequential()
model.add(Input(shape=(in_size,)))
model.add(Dense(500,
activation='relu',kernel_regularizer='l1'/'l2'/'l1_l2'))
model.add(Dense(out_size, activation='softmax'))
```

kernel_regularizer='l1':

Najprv som použila iba L1 regularizáciu, t.j. penalta pre váhy bola vypočítaná pomocou absolútnych hodnôt váh. V tomto prípade sa výsledky validácie a trénovania k sebe veľmi priblížili, priebeh validačnej a trénovacej chyby bol takmer totožný. Výsledná presnosť trénovania a predikcie výstupných žánrov boli však nižšie, ako v pôvodnom modeli.

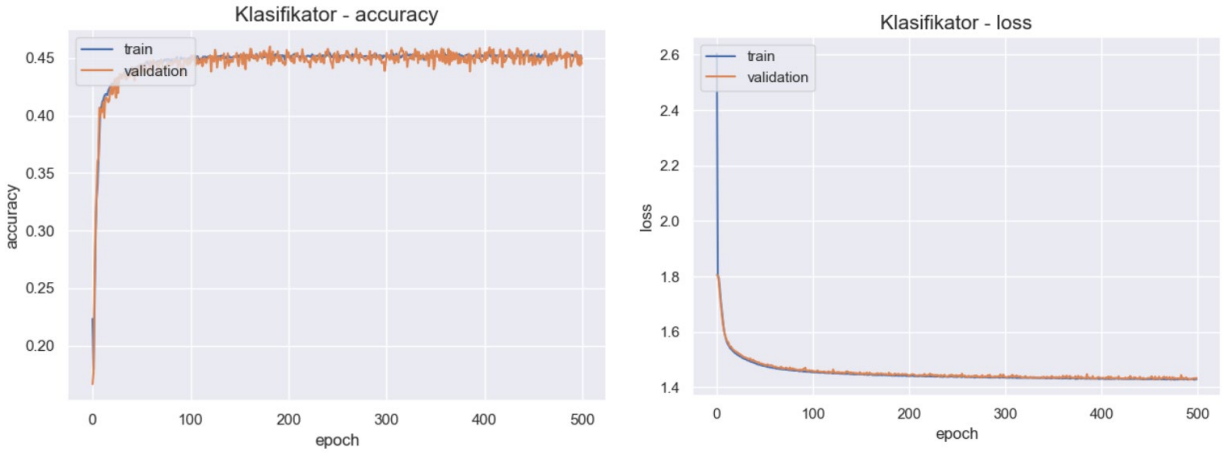
Predikcia výstupu:

```
loss: 2.0135 - categorical_accuracy: 0.3024
```

Výsledky trénovania a validácie:

```
loss: 1.4294 - categorical_accuracy: 0.4497 - val_loss: 1.4346 - val_categorical_accuracy: 0.4450
```

Priebeh presnosti a chyby pri trénovaní a validácii:



kernel_regularizer='l2':

Pri L2 regularizácii, sa penalta vypočítava sumou druhých mocnín váh. Výsledky tu boli tiež relatívne dobré, validačné grafy síce nemali taký plynulý priebeh, ako pri L1, ale nakoniec výsledok skonvergoval na cca 55% úspešnosť trénovania, teda konečné hodnoty boli lepšie, ako pri L1.

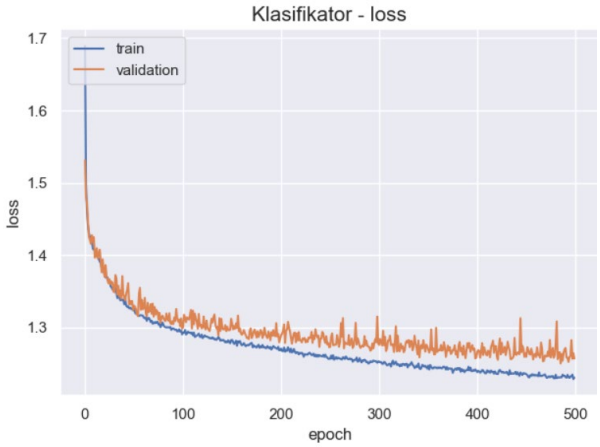
Predikcia výstupu:

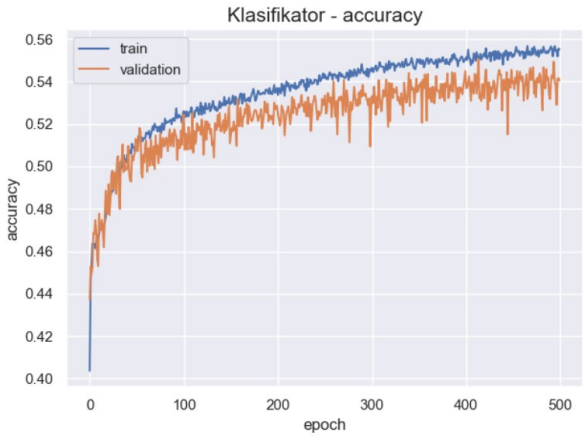
```
loss: 2.1878 - categorical_accuracy: 0.3254
```

Výsledky trénovania a validácie:

```
loss: 1.2306 - categorical_accuracy: 0.5552 - val_loss: 1.2570 - val_categorical_accuracy: 0.5405
```

Priebeh presnosti a chyby pri trénovaní a validácii:





kernel_regularizer='l1_l2':

Pri nastavení súčasne L1 aj L2 boli výsledky takmer rovnaké, ako pri použití iba samostatného L1.

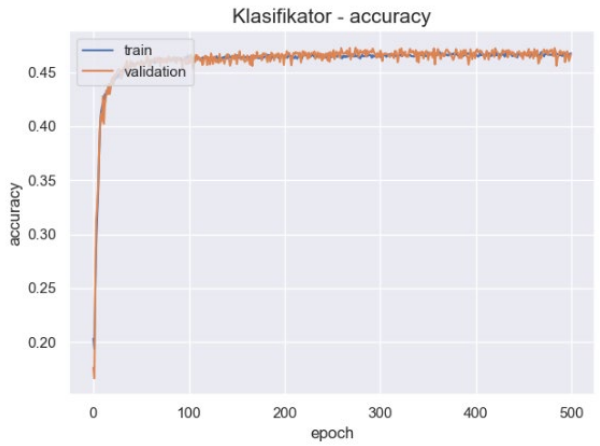
Predikcia výstupu:

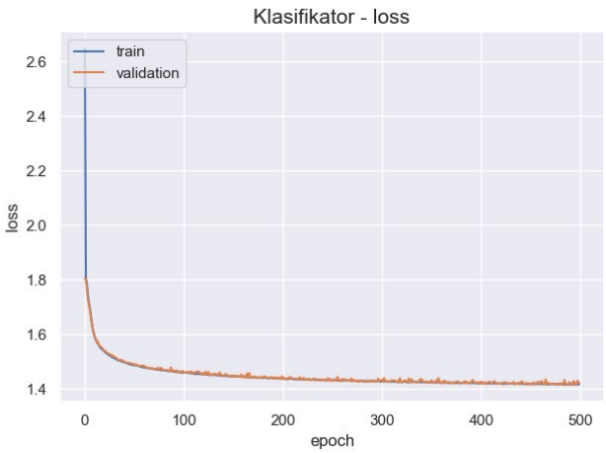
```
loss: 2.0809 - categorical_accuracy: 0.3049
```

Výsledky trénovania a validácie:

```
loss: 1.4171 - categorical_accuracy: 0.4678 - val_loss: 1.4217 - val_categorical_accuracy: 0.4665
```

Priebeh presnosti a chyby pri trénovaní a validácii:





Z porovnania všetkých grafov vyplýva, že pri použití samostatného L1 alebo L1 spolu s L2 dostaneme pre tento model takmer také isté výsledky, takže sú rovnako dobré. Samostatná L2 zlepšila konečné výsledky trénovacej úspešnosti, no trénovanie pri L1 alebo L1+L2 malo stabilnejší priebeh.

3.2 DROPOUT

Pri dropout regularizácii nastáva vynechávanie neurónov pri trénovaní. Dropout vrstvu som pridala medzi skrytú a výstupnú vrstvu. Vyskúšala som viacero nastavení parametra dropoutu (percent, koľko neurónov bude náhodne vynechaných).

```
model = tf.keras.Sequential()
model.add(Input(shape=(in_size,)))
model.add(Dense(500, activation='relu'))
model.add(Dropout([parameter]))
model.add(Dense(out_size, activation='softmax'))
```

Dropout(0.5)

Najlepšie výsledky som dosiahla pri 0.5 dropoute kedy sa validačná presnosť a chyba dosť priblížila k trénovacej a aj celková presnosť trénovania bola relatívne dobrá, cca 57%. Sieť pri tejto regulácii dosahuje presnosť predikcie výstupu 32.5%.

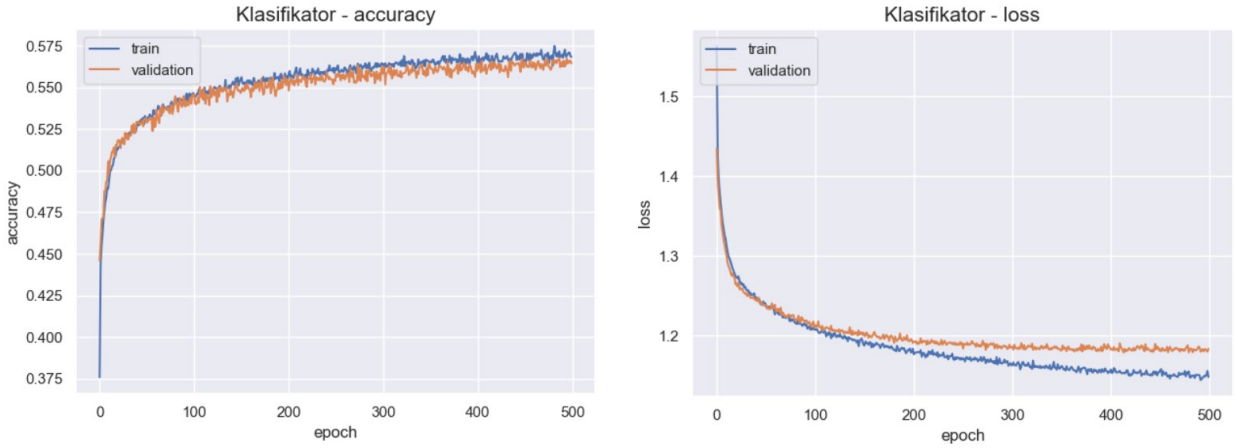
Predikcia výstupu:

```
loss: 2.5278 - categorical_accuracy: 0.3257
```

Výsledky trénovania a validácie:

```
loss: 1.1480 - categorical_accuracy: 0.5685 - val_loss: 1.1835 - val_categorical_accuracy: 0.5646
```

Priebeh presnosti a chyby pri trénovaní a validácii:



Dropout(0.1)

Dropout(0.1) bol príliš malý a rozdiel trénovacích a validačných výsledkov sa výrazne nezmenšil.

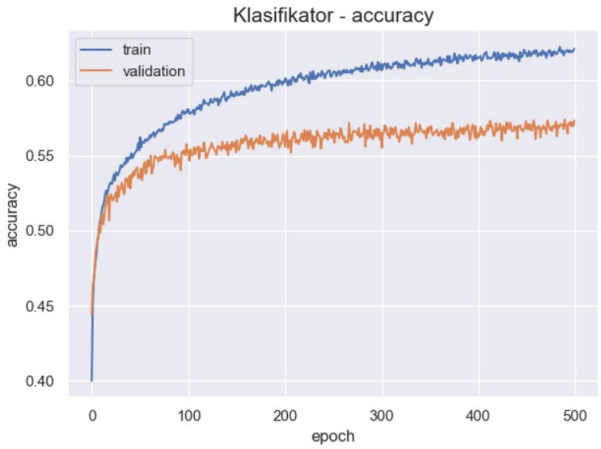
Predikcia výstupu:

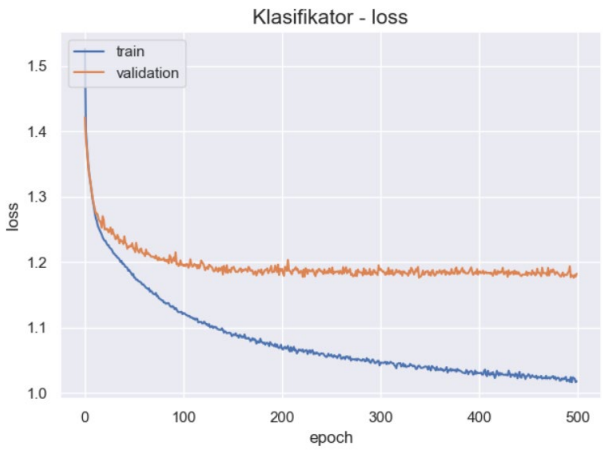
```
loss: 2.8872 - categorical_accuracy: 0.3202
```

Výsledky trénovania a validácie:

```
loss: 1.0188 - categorical_accuracy: 0.6182 - val_loss: 1.1784 - val_categorical_accuracy: 0.5668
```

Priebeh presnosti a chyby pri trénovaní a validácii:





Dropout(0.9)

Dropout(0.9) bol zas príliš veľký, čo viedlo k tomu, že sa presnosť a chyba validácie a trénovania od seba začali znovu vzďaľovať.

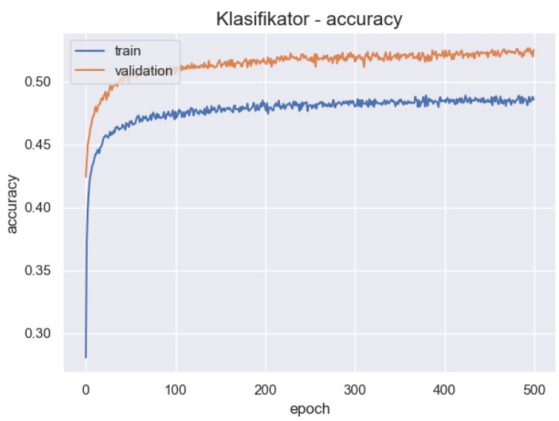
Predikcia výstupu:

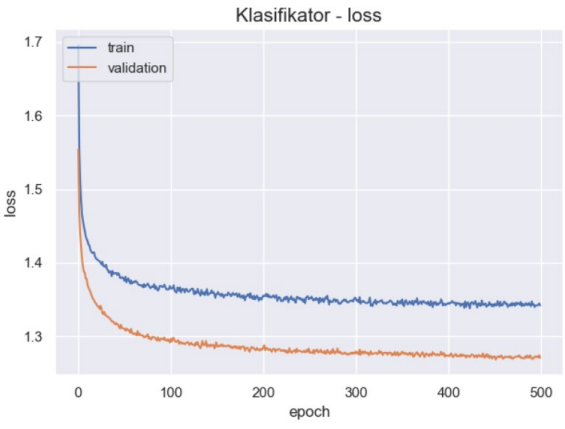
```
loss: 2.2064 - categorical_accuracy: 0.3276
```

Výsledky trénovania a validácie:

```
loss: 1.3421 - categorical_accuracy: 0.4855 - val_loss: 1.2707 - val_categorical_accuracy: 0.5254
```

Priebeh presnosti a chyby pri trénovaní a validácii:





Z porovnania rôznych veľkostí dropoutu vyplýva, že najlepšie výsledky trénovania dáva zlatá stredná cesta. Príliš veľký dropout, ani príliš malý nedal tak dobré výsledky, ako pri dropoute 0.5. Dropout nemal vplyv na úspešnosť predikcie žánrov z testovacích dát.

3.3 BATCH NORMALIZATION

Batch normalizácia sa používa na normalizáciu váh počas trénovania. Vrstvy s normalizáciou som pridala pred a za skrytú vrstvu. Batch normalizáciu som vyskúšala použiť pre rôzne aktivačné funkcie.

```
model = tf.keras.Sequential()
model.add(Input(shape=(in_size,)))
model.add(BatchNormalization())
model.add(Dense(500, activation='relu'/'tanh'/'sigmoid'))
model.add(BatchNormalization())
model.add(Dense(out_size, activation='softmax'))
```

activation='relu'

Najprv som použila batch normalizáciu spolu s aktivačnou funkciou 'relu'. Chyby a presnosti trénovania a validácie sa vôbec nezlepšili, skôr sa rozdiely medzi trénovacími a testovacími výsledkami ešte viac zväčšili. Presnosť predpovedaného výstupu bola porovnateľná s ostatnými regularizačnými metódami.

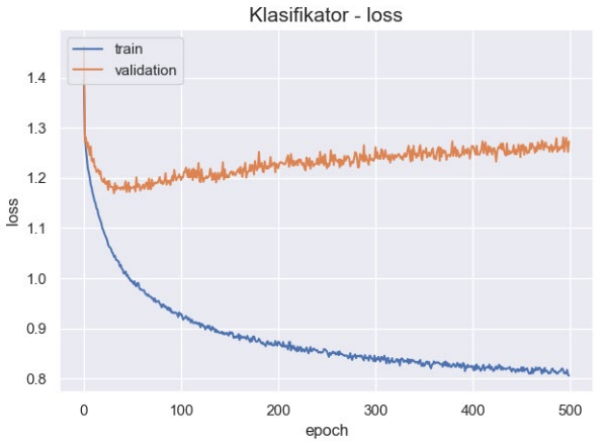
Predikcia výstupu:

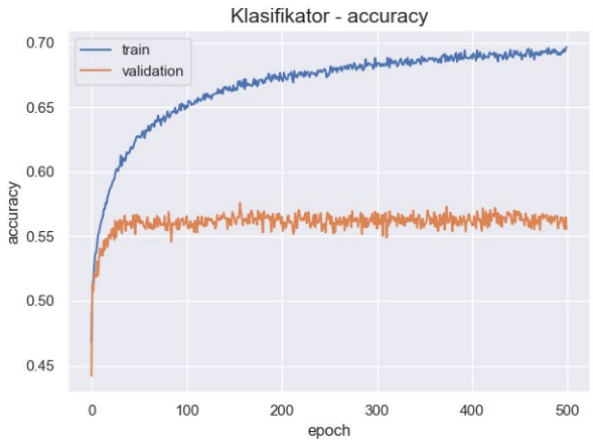
```
loss: 2.5669 - categorical_accuracy: 0.3298
```

Výsledky trénovania a validácie:

```
loss: 0.8057 - categorical_accuracy: 0.6965 - val_loss: 1.2731 - val_categorical_accuracy: 0.5557
```

Priebeh presnosti a chyby pri trénovaní a validácii:





activation='tanh'

Pri použití aktivačnej funkcie 'tanh' v skrytej vrstve sa grafy validácie a trénovania k sebe začali približovať, aj keď výsledok nebol optimálny, pretože sieť je stále pretrénovaná. Výsledok z evaluácie predikovaného výstupu klasifikátora bol pri tomto type aktivačnej funkcie oveľa horší, ako v ostatných prípadoch.

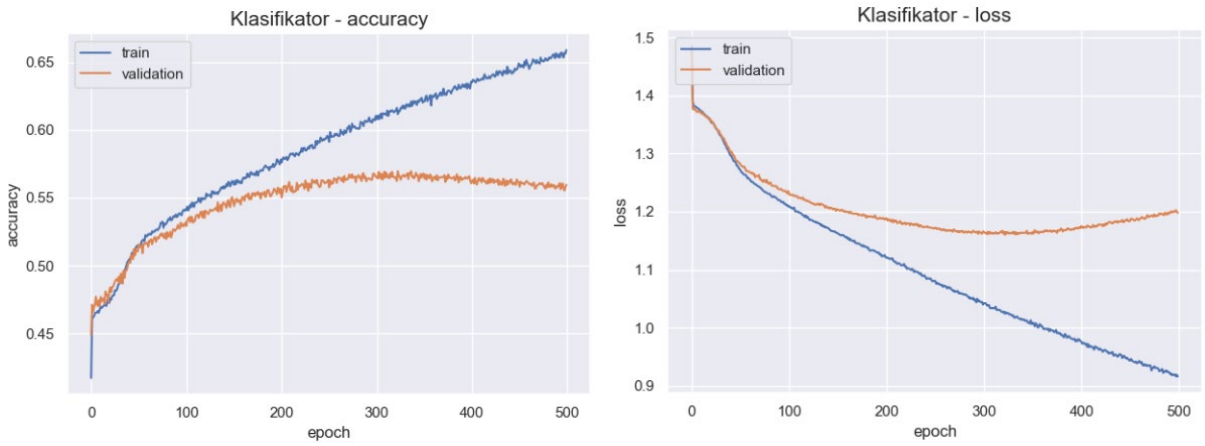
Predikcia výstupu:

```
loss: 2.6868 - categorical_accuracy: 0.2921
```

Výsledky trénovania a validácie:

```
loss: 0.9158 - categorical_accuracy: 0.6590 - val_loss: 1.1976 - val_categorical_accuracy: 0.5596
```

Priebeh presnosti a chyby pri trénovaní a validácii:



activation='sigmoid'

Keď som ako aktivačnú funkciu v skrytej vrstve použila **'sigmoid'**, priebeh chyby a presnosti sa čiastočne zlepšil, grafy pre validáciu a trénovanie sú k sebe bližšie.

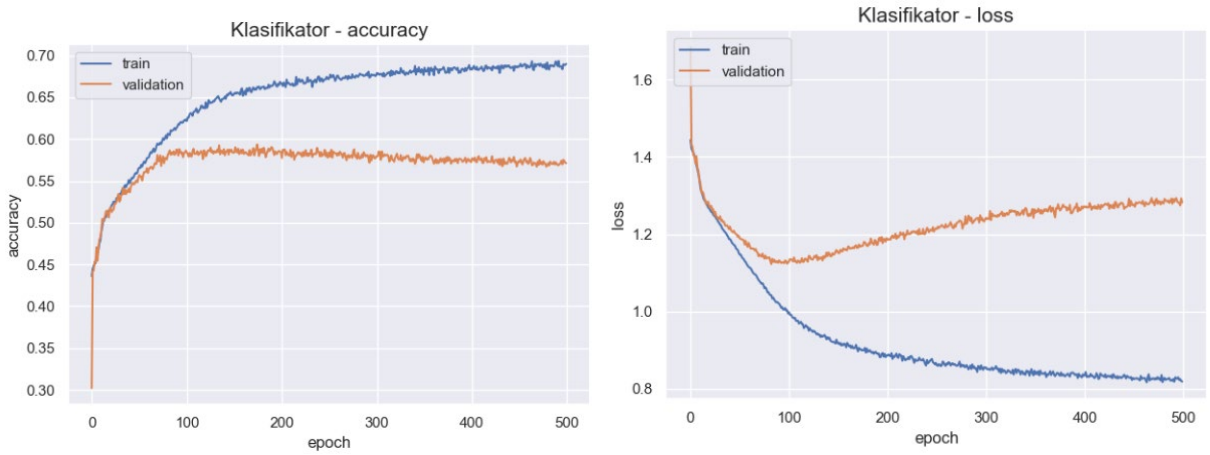
Predikcia výstupu:

```
loss: 4.3146 - categorical_accuracy: 0.3032
```

Výsledky trénovania a validácie:

```
loss: 0.8190 - categorical_accuracy: 0.6899 - val_loss: 1.2824 - val_categorical_accuracy: 0.5712
```

Priebeh presnosti a chyby pri trénovaní a validácii:



Z grafov vyplýva, že batch normalization dala spomedzi všetkých regularizačných metód najhoršie výsledky pre rôzne typy aktivačných funkcií a teda pre môj model nie je vhodná. Použila by som ju iba ak s nejakou inou regularizačnou technikou.

4 SVM KLASIFIKÁTOR

Na implementáciu SVM klasifikátora som použila knižnicu `sklearn` a na vykreslenie confusion matrix knižnice `seaborn` a `matplotlib`.

4.1 ROZDELENIE DÁT

Najprv som dataset rozdelia na vstupy (X) a výstup (Y) neurónovej siete.

X: Vstupnú množinu predstavujú všetky stĺpce datasetu, okrem *playlist_genre* a *playlist_subgenre*.

Y: Výstupnú množinu tvorí stĺpec *playlist_genre*.

Dáta boli už vopred rozdelené na trénovaciu a testovaciu množinu.

4.2 NASTAVENIE KLASIFIKÁTORA

SVM klasifikátor som inicializovala pomocou `svm.SVC` funkcie (SVM = Support Vector Classification), kde som nastavila kernel typ na `'rbf'` a zastavovaciu podmienku – toleranciu na 0.00001.

```
classifier = svm.SVC(kernel="rbf", tol=0.00001, verbose=1)
```

4.3 VÝSLEDKY TRÉNOVANIA

V porovnaní s vlastným klasifikátorom, ktorý dosahoval približne 35% presnosť, SVM klasifikátor dosiahol trochu horšie výsledky, zhruba 32%. Výsledky trénovania sú zobrazené na classification report. Vykreslila som tiež confusion matrix, rovnako ako pri prvom klasifikátore spomedzi kategórií prevláda rock. Matica je veľmi podobná, ako pri keras klasifikátore.

Classification report

	precision	recall	f1-score	support
0	0.547	0.298	0.386	1669
1	0.343	0.346	0.345	1549
2	0.258	0.115	0.159	1658
3	0.355	0.130	0.190	1653
4	0.493	0.251	0.332	1620
5	0.255	0.787	0.385	1685
accuracy			0.322	9834
macro avg	0.375	0.321	0.300	9834
weighted avg	0.375	0.322	0.299	9834

Confusion matrix