Klasifikácia filmového žánru

Martin Slovák

January 3, 2025

1 Abstrakt

Tento projekt sa zaoberá problematikou viac triednej klasifikácie filmových žánrov pomocou jednoduchého filmového popisu. Filmový popis by nemal obsahovať spoiler a mal by iba načrtnúť obsah filmu, pričom sa môže odrážať od svojho žánru. Preto som sa rozhodol využiť techniky prirodzeného spracovania jazyka a strojového učenia s učiteľom, aby som vyskúšal, ako presne a efektívne je možné klasifikovať filmové žánre na základe takýchto stručných textových popisov. V rámci projektu som použil dataset obsahujúci filmové popisy a žánre Action, Comedy a Horror. Vektorizáciu som robil pomocou TF-IDF a zvolené metódy boli rozhodovací strom a podobné varianty náhodný les a gradient boosting, potom logistická regresia, naivný Bayes a Support Vector Machine. Vyhodnocoval som na základe accuracy, macro average precision, overall a F1-score.

Ukázal som na základe výsledkov, že aj keď tieto metódy nedosahuje úplnú efektivitu, resp. presnú klasifikáciu filmových žánrov, ich výkon je sľubný a naznačujú potenciál.

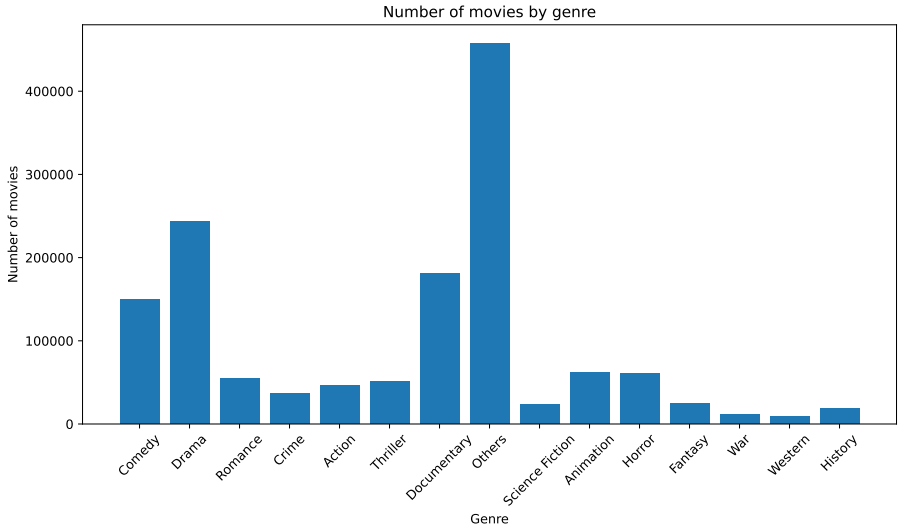
2 Úvod

2.1 Problém

Budem riešiť viac triednu klasifikáciu (strojové učenie s učiteľom), konkrétne 3 triedy a budú to žánre Comedy, Horror, Action. Jednotlivé triedy budu abecedne zakódované do číselných foriem Action = 1, Comedy = 2, Horror = 3. Trénovací dataset bude mať formu (x_i, y_i), kde $x_i \in \mathbb{R}^n$ je i-ty príklad a $y_i \in \{1, 2, 3\}$ je i-ta trieda. Cieľom bude nájsť takú hypotézu $h \in H$, že $h(x_i) = y_i$, kde H je priestor z hypotéz. Príklady x budú vo forme reťazca a pôjde o popis filmu "overview".

2.2 Dataset

Zo začiatku som skúšal získať dáta cez IMDB. Bohužiaľ je API dostupne iba cez AWS a nie je najlacnejšie. Ponúkajú zadarmo dataset na stránke `https://developer.imdb.com/non-commercial-datasets/` ale dáta neobsahujú popis filmu, takže to nie je užitočné. Ďalej existuje stránka OMDb API ale ponúka zadarmo 1000 použití na deň ale v prípade rýchlych zmien dát to nebude prospešné. Nakoniec som našiel stránku TMDB (The Movie DataBase), ktorá ponúka API ale tiež jeden človek na Kaggle zverejnil dataset s 1M záznamov o filmoch od TMDB+IMDb a obsahuje žánre a popis filmov. Dodatočne nie sú upravené, teda nejde o benchmark dáta a ak by som si zobral TMDB API a IMDb tak by to mohlo byť 1:1, takže nevidím dôvod prečo to nevyužiť. Takže môj dataset, ktorý som si vybral je `https://www.kaggle.com/datasets/alanvourch/tmdb-movies-daily-updates/data` (verzia 325). Ako prvé budem analyzovať čo všetko dataset obsahuje a čo treba pripraviť. Dataset obsahuje celkovo 1 029 720 riadkov a rozdelenie žánrov je nasledovne:



Do others patria iné žánre, napríklad Music, Mystery... ale aj zle hodnoty a preto budem musieť vyčistiť dáta.

2.3 Preprocessing I

Tento preprocessing bude slúžiť na čistenie dát od nežiadúcich hodnôt. Aj keď je v datasete veľké množstvo dát tak nie každé má vyplnené overview, genres alebo dokonca je overview v inom jazyku ako anglicky alebo sú tam duplicity.Tiež budem musieť vymazať tie riadky, ktoré nemajú žáner Comedy,Horror alebo Action.

Dodatočne je formát CSV (Comma Separated Values) a niektoré hodnoty nie sú dané ako prázdne medzi ,, ale sú vynechané a z toho dôvodu má niektorý riadok namiesto ID vyplnený title a podobne. Preto som si to prekonvertoval do stĺpcových hodnôt v **Exceli** .xlsx a pomocou filtra som skúmal dáta a vymazal určité časti a zvyšok som doriešil pomocou Pythonu. Preto odporúčam pracovať iba s predspracovaním datasetom, ktorý bude v prílohe, lebo, ak sa samostatne spustí prepare_movies.py tak môže tam byť malý rozdiel dát a potom aj vo výsledku. Ukážka analýzy dát:

- Počet filmov, kde nie je uvedený popis: 183 273
- Počet filmov, kde nie je uvedený žáner: 298 061
- Počet filmov, kde sú duplikáty popisu filmu, pričom popis je uvedený: 18556 (napr. je hodnota popisu filmu "Movie Details")

A to som nezapočítal žánre, ktoré sú zle napísane atď...

2.4 Preprocessing II

Tento preprocessing bude nasledovať po prvom a ide o Natural Language Processing. Ešte predtým než začnem využívať rôzne metódy tak si musím pripraviť vstup do dostatočnej podoby. NLP podľa definície je teoreticky motivovaný rozsah na analýzu a reprezentáciu prirodzene sa vyskytujúcich textov na jednej alebo viacerých úrovniach lingvistickej analýzy za účelom dosiahnutia spracovania ľudského jazyka pre rad úloh alebo aplikácií. [13]. Úlohou je teda pripraviť text do takej formy, aby s tým vedel narábať model. Budem postupovať následovne, pričom využijem knižnice NLTK [17] a langid [12]:

1. **Odstránenie interpunkčných znamienok**. Tieto znaky neprispievajú modelu a ak by som ich nevymazal tak by sa mohli zhoršiť ďalšie kroky spracovania textu. Napríklad (hello world again) by pri rozložení na jednotlivé tokeny bolo ["(hello", "world", "again)"] a nastal by problém pri ďalšom spracovaní, pretože slovo "(hello" nie je správne. Toto bude platiť aj pre HTML tagy, ktoré sa tam môžu vyskytnúť.
2. **Odstránenie znakov, ktoré nie sú ASCII**. Budem sa snažiť držať iba anglické slová a preto znaky, ktoré nie sú ASCII sa budú zle vysvetľovať.
3. **Ponechanie iba znakov z abecedy**. Teraz sa budem snažiť odstrániť čísla, pretože nemajú žiaden prínos.
4. **Prevedenie všetkých písmen v texte na malé**. Slová "Hello" a "hello" už len, keď vložíme do asociatívneho poľa tak budeme mať dva odlišné kľúče a pritom pre človeka sú to rovnaké slová, len jedno môže hovoriť o začiatku vety. Podobne by to fungovalo aj pri spracovaní textu. Preto musím zaistiť, aby boli slová rovnaké a to tak, že prevediem všetky písmena na malé.

5. **Odstránenie Stop words**. Stop word sú také slová, ktoré sa považujú za bezvýznamne a nedávajú nám informačnú hodnotu. Tieto slová síce tvoria základ jazyka, ale neprispievajú výrazne k obsahu textu a preto ich môžem odstrániť.

6. **Aplikácia lemmatizácie**. Lemmatizacia je určenie lemmatu teda základného slovného tvaru, pričom sa pri tomto procese sa využíva slovník. Napríklad slovo "running" by sa zmenilo na "run". Takto viem docieliť, že iné filmové popisy budú mať väčšiu podobnosť.

Toto je môj postup a takto zaistím, že vstupný text bude v požadovanej forme na ďalšie spracovanie. Tiež je dobré dodať, že okrem lemmatizácie existuje aj metóda, ktorá sa vola **Stemming**. Ide o metódu, pri ktorej sa hľadá kmeň slova tak, že sa odstránia morfologické koncovky a predpony. Slová ako fisher, fished, fishing môžu vrátiť tvar fish ale napríklad slovo arguing vráti argu, čo nie je slovo. Toto je jedna nevýhoda stemming a preto to ani neplánujem používať.

2.5 Počet dát

Kvôli tomu, že mám veľa dát a nie sú rovnomerné vyvážené tak si určím hornú hranicu k filmov pre daný žáner. Ďalej budem musieť podľa niečoho zoradiť. Počet slov bol môj prvý nápad lebo teoreticky čím viacej slov ten popis obsahuje tak tým kvalitnejší ten text môže byť ale nebola to pravda lebo dlhý text môže hovoriť všetko okrem žánru a zase malý text ak má unikátne slová, ktoré pomôžu ku klasifikácií. Toto vyriešim pomocou metódy TF-IDF (Term Frequency-Inverse Document Frequency), ktorá slúži na vyhodnotenie dôležitosti jednotlivých slov v rámci textu vzhľadom na celý dataset. Popis ma vysokú hodnotu, ak obsahuje viac informatívnych a unikátnych slov a naopak nižšiu, ak obsahuje prevažne bežné alebo často sa opakujúce slová. TF (Term Frequency) meria, ako často sa dané slovo vyskytuje v konkrétnom popise.

$$TF(t,d) = \frac{Number\ of\ times\ term\ t\ appears\ in\ overview\ d}{Total\ number\ of\ terms\ in\ overview\ d}$$

IDF vypočíta ako vzácne je slovo v celom datasete.

$$IDF(t,D) = \log(\frac{Total\ number\ of\ overviews\ in\ the\ dataset\ N}{Number\ of\ overviews\ containing\ term\ t})$$

Skóre sa potom vypočíta ako

$$TF - IDF(t,d,D) = TF(t,d) \times IDF(t,D)$$

Toto skóre zoradím od najväčšieho a vyberiem iba top k filmov pre každý žáner. Ako prvé som si musel pozrieť ako veľký je nepomer a prispôsobím sa na určité číslo, ktoré bolo na konci $k = 10000$ filmov pre každý žáner.

Potom som roztriedil dataset na trénovaci/testovací/validačný v pomere 60:20:20, pričom boli žánre vyvážené.

2.6 Forma vstupných dát

Keď už mám pripravený čistý text pre každý film tak by som ho mohol dať ako vstupné dáta x_i. Ale to by tak nemalo fungovať, pretože aby bol text použiteľný pre rôzne metódy prostredníctvom počítačov, musí byť najprv transformovaný na číselné hodnoty [5]. Takéto transformovanie sa volá vektorizácia textu. Najprv si text rozdelím na samostatné jednotky, ktoré sa volajú tokeny, teda jedno slovo napríklad "this is sentence" →["this", "is", "sentence"] a potom použijem určitú metódu. Súčastnosti existuje mnoho metód ako napríklad Bag of Words [10], TF-IDF, Word2Vec [2] alebo BERT (Bidirectional Encoder Representations from Transformers)[4]. Vybral som si TF-IDF vektorizovanie pomocou Scikit-learn knižnice, pretože BERT moc dlho trval kým sa vykoná vektorizácia a nie je garantované, že bude fungovať úplne super a Words2Vec mal horšie výsledky ako TF-IDF. Dodatočne bag of words je podobne ako TF-IDF len nezachováva informáciu o dôležitosti slov. Jednoduchý príklad vektorizácie:
Majme 1. text "Hello world", 2. text "Hello world again" a 3. text "This sentence has no meaning". Aplikujem TF-IDF a dostanem:

Dokument	**Slovo**	**TF-IDF hodnota**
0	2	0.7071
0	7	0.7071
1	2	0.5179
1	7	0.5179
1	0	0.6809
2	6	0.4472
2	5	0.4472
2	1	0.4472
2	4	0.4472
2	3	0.4472

Table 1: Ukážka TF-IDF.

3 Aktuálny state of art

Klasifikácia filmových žánrov nie je unikátna problematika a mnoho ľudí sa ju snažilo riešiť s rôznymi prístupmi a definíciami problému. Táto úloha sa dá zadefinovať ako binárna klasifikácia alebo multiclass, prípadne multilabel, teda film môže mať viacero žánrov. Na vstupe je video a snažili sa to vyriešiť cez konvolučné siete [19]. Prípadne filmový plagát a použili hlboké neuronové siete [1]. Takže vstup a metóda riešenia je na kreativite človeka ale aby som sa držal svojho problému tak existujú aj články, kde to ľudia riešia pomocou filmového popisu a ako metódu použili neuronové siete [6], kde sa hodnoty f1-score, precision a recall pohybovali okolo ≈60. Ďalej cez metódy ako Support Vector Classifier, Naive Bayes, Logistic Regression + používali aj TF-IDF a mali dosť dobré hodnoty ako napríklad accuracy 95 ale to mal multi-label problém

[11].

Neplánujem tu všetko rozpisovať kto, čo robil ale tieto články som našiel cez Google Scholar a ešte existujú aj mimo toho a hlavne prevažne sú multi-label ale na tejto problematike je dôležité to, že každý mal rôzne vstupy a teda, ak niekto dosiahne accuracy 95 tak to nemusím hneď aj ja dosiahnuť.

4 Metódy

Metódy budú strojové učenie s učiteľom a ich implementáciu budem vykonávať pomocou knižnice Scikit-learn a hľadanie optimálnych hyperparametrov pomocou techniky grid search [9].

4.1 Naive Bayes

Túto metódu sme sa síce neučili ale bude vhodná na moju problematiku. Naive Bayes patrí do skupiny pravdepodobnostných klasifikátorov a predpokladá, že všetky atributy sú navzájom nezávislé. Je založený na základe Bayesovej vety

$$P(C_k|x) = \frac{P(C_k)P(x|C_k)}{P(x)}$$

, kde p$(C_k|x_1, ..., x_n)$ je priradená pravdepodobnosť pre každú z K možných výstupov alebo tried C_k, pričom problém má byť klasifikovaný. x $= (x_1, ..., x_n)$ je vektor, ktorý obsahuje n vlastností. Je vhodný na prácu s textom a jedno známe využitie je "spam filtering".

Existuje viacero verzií, pričom každá sa používa na iný typ dát. **Gaussovský** naivný Bayes sa používa na kontinuálne dáta, pričom sa predpokladá, že používa Gaussovo rozdelenie. Čo v mojom prípade nie je vhodné. Naopak **Multinomiálny** naivný Bayes je vhodný na diskrétne dáta a preto je vhodný na počty výskytov slov v texte. Je vyjadrený ako

$$p(x|C_k) = \frac{(\sum_{i=1}^{n} x_i)!}{\Pi_{i=1}^{n} x_i!} \Pi_{i=1}^{n} p_{ki}^{x_i},$$

kde $p_{ki} = p(i|C_k)$. Ešte existuje **Bernoulliho** naivný Bayes ale ten je vhodný pre binárne dáta.

V knižnici Scikit-learn sa táto metóda volá MultinomialNB [16] a obsahuje hyperparameter rýchlosť učenia α.

4.2 Support Vector Machine

SVM je algoritmus primárne využívaný na klasifikáciu a regresnú analýzu. Funguje na princípe hľadania optimálnej hyperroviny, ktorá najlepšie oddeľuje triedy v priestore. Táto hyperrovina je umiestnená tak, aby maximalizovala vzdialenosť medzi najbližšími bodmi z rôznych tried. Táto vzdialenosť sa volá margin.

SVM môže byť lineárná. d-dimenzionálna hyperrovina môže byť napísaná ako množina x bodov, ktoré spĺňajú

$$w^T x - b = 0,$$

kde w je normálový vektor hyperroviny. Ak sú dáta lineárne separovateľné tak existuje taká hyperrovina, ktorá dokáže úplne oddeliť dve triedy dát bez akejkoľvek chyby. Matematicky minimalizujeme w, b

$$\frac{1}{2}||w||^2$$

za podmienky

$$y_i(w^T x_i - b) \geq 1$$

, kde y_i je trieda a x_i je dátový bod.

Ak by dáta neboli lineárne separovateľné tak si zavedieme novú premenné ξ, ktorá pridáva flexibilitu modelu a dovolí niektorým bodom byť na zlej strane, prípadne na hyperrovine a premennú C, ktorá určuje kompromis medzi zväčšovaním veľkosti marginu a zabezpečením, že dátové body x_i ležia na správnej strane. Teraz minimalizujeme w,b, ξ

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i$$

za podmienky

$$y_i(w^T x_i - b) \geq 1 - \xi_i, \xi_i \geq 0.$$

SVM môže byť aj nelineárne pomocou kernelového triku čo je metóda, ktorá umožňuje transformáciu dát do novej dimenzie pomocou kernelovej funkcie. Napr. polynomiálna kernelová funkcia je

$$k(x_i, x_j) = (x_i \cdot x_j)^d.$$

Ja sa budem zaoberať multiclass SVM a pretože SVM bol navrhnutý na binárne klasifikované tak si rozdelím multiclass na viacero binárnych. Techniky môžu byť

- **One-versus-rest** funguje na princípe zvolenia jednej triedy a zvyšné zlúčim a spravím binárnu klasifikáciu. Víťaz je ten čo má najlepšie skóre, teda najlepší výstup. Pre moje 3 triedy 0,1,2 by to vyzeralo ako 0 a (1 ∪ 2), 1 a (0 ∪ 2), 2 a (0 ∪ 1).
- **One-versus-one** funguje tak, že si vytvorím všetky dvojice (0, 1), (0, 2), (1, 2) a každý binárny klasifikátor predpovie pre jednu z dvojíc a pridá jej hlas. Trieda s najväčším počtom hlasov vyhráva, teda je konečná predikcia.

Knižnica Scikit-learn má metódu SVC (Support Vector Classifier) a implementuje one-versus-one metódu na multiclass [20] a obsahuje hyperparametre ako sú C, typ kernelovej funkcie, stupeň polynómu, gammu a coef0. Ale kvôli náročnej časovej zložitosti budem optimalizovať iba kernelovú funkciu, C a gammu.

4.3 Rozhodovací strom

Ide o algoritmus, ktorý sa najčastejšie využíva na klasifikáciu a podporuje multiclass. Stromová hierarchická štruktúra, ktorá rozkladá dáta na základe podmienok v uzloch, až kým sa nedosiahne listový uzol, ktorý predstavuje výslednú predikciu.

Pri tvorbe stromu sa vyberie atribút, ktorý najlepšie rozdeľuje dáta. Na takéto rozhodnutie sa používa napr. entropia, ktorá určuje mieru informačnej hodnoty atribútu. Entropia H pre množinu dát X má všeobecný vzorec

$$H(X) = -\sum_{i=1}^{|X|} P(x_i) \log_2 P(x_i),$$

alebo Gini G(X):

$$G(X) = 1 - \sum_{i=1}^{|H|} P(x_i)^2.$$

Okrem rozhodovacieho stromu plánujem použiť aj náhodný les čo je vytvorenie viacerých rozhodovacích stromov a Gradient Boosting, ktorá postupne pridáva rozhodovacie stromy. Každý nový strom je optimalizovaný tak, aby zlepšil chyby predchádzajúcich stromov. Tieto všetky techniky sa v Scikit-learn volajú DecisionTreeClassifier [3] , RandomForestClassifier [18] a GradientBoostingClassifier [7]. Pre rozhodovací strom budem hľadať parametre max_depth a criterion (entropia, Gini, log_loss), pri náhodnom lese budem optimalizovať aj n_estimators a pre GradientBoosting aj learning rate.

4.4 Logistická regresia

Algoritmus určený na klasifikáciu. Jej typ určuje koľko tried bude klasifikovať. Napríklad Binomiálna logistická regresia vykonáva binárnu klasifikáciu a Multinomiálna vykonáva klasifikáciu pre 3 a viac tried. Preto v mojom prípade budem využívať multinomiálnu klasifikáciu. Na rozdiel od binomiálnej verzie, ktorá používa sigmoid funkciu na výpočet pravdepodobností, multinomiálna logistická regresia používa softmax funkciu. Pre K tried vyzerá funkcia

$$P(y = k|x) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}},$$

kde $z_k = w_k^T x + b_c$ je lineárna kombinácia váh w_k a vstupov x pre triedu k.

Scikit-learn obsahuje metódu LogisticRegression [14] a budem hľadať optimálne hyperparametre C, penalty a solver.

4.5 Technické problémy

Pôvodne som mal dataset veľkosti $k = 30000$ ale kvôli tomu, že SVC od Scikit-learn počíta kvadratické programovanie a časová zložitosť je medzi $O(n_{features} \times$

$n^2_{samples}$) a $O(n_{features} \times n^2_{samples})$ [21] čo nie je optimálne pre moju veľkosť. Ďalší problém bol, že GradientBoostingClassifier od Scikit-learn tiež nebol optimálny a chcel som skúsiť alternatívnu metódu XGBoost [22], ktorá je známa tým, že je rýchlejšia ako GradientBoostingClassifier a umožňovala využitie CUDA. Keď som skúsil CUDA tak mi to dávalo chybu, že som run out of memory. Toto som asi aj mohol čakať pre moju GTX 1050 Ti, pretože má iba 4 GB pamäte a tieto výpočty si vyžadujú viac. Tak som ponechal CPU ale, keď som skúšal hĺbku stromu viac než 7 tak mi to dávalo rýchle výsledky ale neboli to čísla, iba NaN. Tu je príklad výstupu počas hľadania vhodných parametrov:

```
[CV 2/5] END ...learning_rate=0.01, max_depth=3;, score
    =0.589 total time= 3.9min
[CV 1/5] END ...learning_rate=0.01, max_depth=3;, score
    =0.590 total time= 4.0min
[CV 4/5] END ...learning_rate=0.01, max_depth=3;, score
    =0.597 total time= 4.0min
[CV 5/5] END ...learning_rate=0.01, max_depth=3;, score
    =0.611 total time= 4.0min
[CV 3/5] END ...learning_rate=0.01, max_depth=3;, score
    =0.623 total time= 4.1min
[CV 5/5] END .....learning_rate=0.05, max_depth=7;, score
   =nan total time= 1.0min
```

Dôvod prečo mi to dávalo NaN môže byť tým, že to žiada veľmi veľa pamäte (16 GB RAM možno nestačí) a namiesto toho, aby to dalo error tak to vráti NaN alebo nedokáže optimálne vypočítať vstupnú riedku maticu, ktorá vznikne pomocou TF-IDF. Dôvod som už hlbšie neriešil a zostal som pri GradientBoostingClassifier ale počkal som si dlhšie. Jedno trénovanie môže trvať aj 50 min ale aspoň to vypočíta pre väčšie hĺbky ([CV 4/5] END ..learning_rate=0.01, max_depth=30;, score=0.710 total time=56.6min).

Zvyšné metódy mi pekne rozbehlo bez akýchkoľvek problémov.

5 Vyhodnotenie

5.1 Metriky

Metriky som si zvolil podľa článku METRICS FOR MULTI-CLASS CLASSIFICATION: AN OVERVIEW [8], kde autori uvádzajú rôzne metriky na moju problematiku. Metriky, ktoré použijem

- **Accuracy** vyjadruje pomer správne klasifikovaný a celkový počet vzoriek, teda vyjadruje pravdepodobnosť, že predikcia modelu bude správna pre náhodne vybraný prípad.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Confusion matrix** je tabuľka, ktorá sumarizuje výsledky klasifikácie. Riadky sú skutočné (actual) triedy a stĺpce sú predpovedané triedy, pričom sú rovnako zoradené pre riadky a stĺpce. Z toho dôvodu sú správne klasifikované prípady na hlavnej diagonále a zlé mimo.

- **Precision & recall**. Precision vyjadruje pomer hodnôt, ktoré náš model hovorí, že sú pozitívne a v skutočnosti sú pozitívne, teda ako veľmi môžeme dôverovať, ak predpovedá jednotlivca ako pozitívneho.

$$Precision = \frac{TP}{TP + FP}$$

  Recall hovorí o schopnosti modelu nájsť všetky pozitívne príklady v dátach. Vysoká hodnota znamená, že model robí málo FN chýb.

$$Recall = \frac{TP}{TP + FN}$$

- **F1-Score** metrika kombinuje hodnoty precision a recall do jedného čísla pomocou harmonického priemeru. Ide teda o kompromis medzi precision a recall.

$$F1 - Score = (\frac{2}{precision^{-1} + recall^{-1}}) = 2 \cdot (\frac{precision \cdot recall}{precision + recall})$$

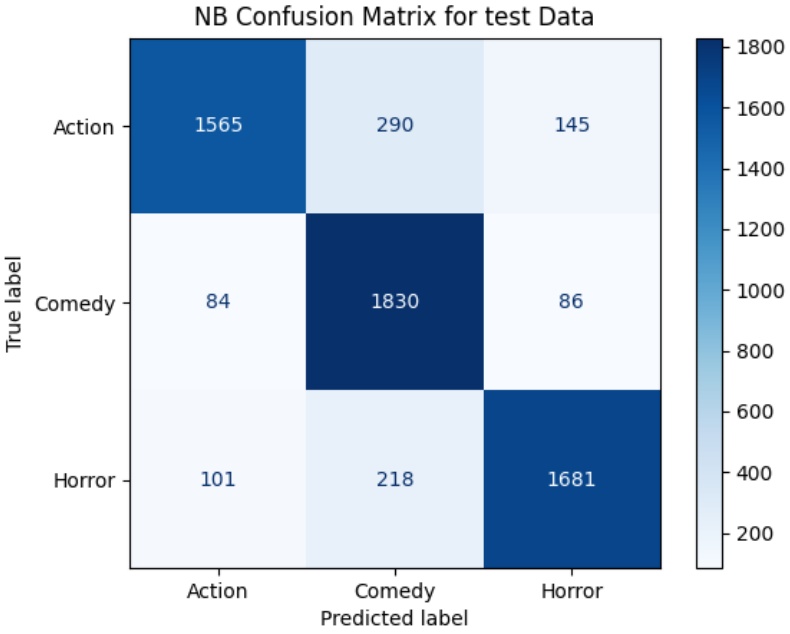
Precision, recall a F1-score sa vypočíta pre každý žáner samostatne a potom sa spraví makro priemer (priemer bez pridanej váhy pre jednotlivé žánre). Tieto hodnoty metrík budú založené na **testovacích dátach**, pričom budem kontrolovať hodnoty aj na validačných dátach, aby som zistil či nejde o underfitting/overfitting.

5.2 Vyhodnotenie Naive Bayes

Optimálny parameter alpha je 0.355. Výsledné hodnoty sú

- Accuracy : 0.846
- Macro Precision: 0.852
- Macro Recall: 0.846
- Macro F1-Score: 0.845

Confusion matrix je



5.3 Vyhodnotenie SVC

Optimálne parametre sú C=10, solver = 'RBF' (Radial Basis Function), ktorá je definovaná ako:

$$K(x_i, x_j) = \exp\left(-\gamma ||x_i - x_j||^2\right)$$

, kde γ je hyperparameter, ktorý som tiež hľadal a je nastavený ako "scale" hodnota čo sa vypočíta ako:

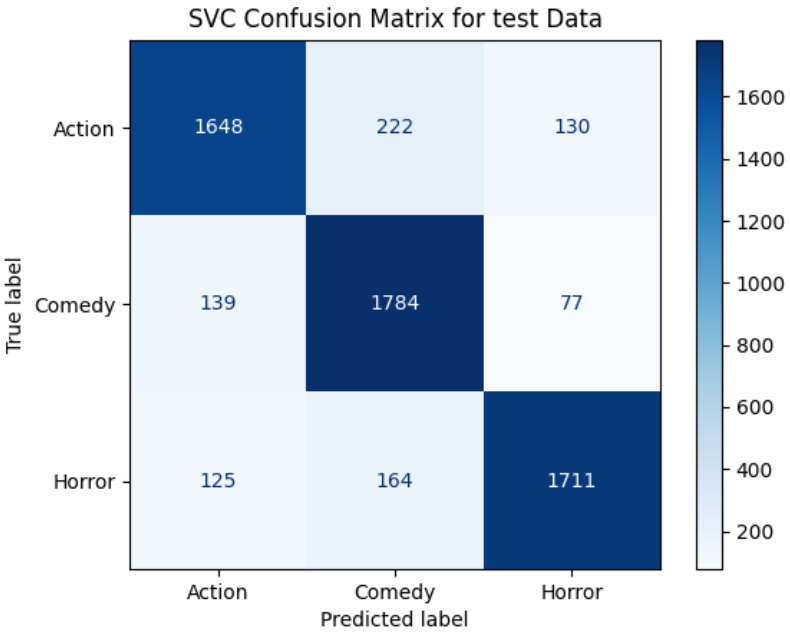
$$\gamma = \frac{1}{n\_features \cdot Var(X)}$$

Pre tieto parametre dostávam výsledok:

- Accuracy : 0.854
- Macro Precision: 0.856
- Macro Recall: 0.854
- Macro F1-Score: 0.854

Tiež je dobré dodať, že validačné hodnoty su skoro rovnaké, takže nie je tu príznak overfitting. Maximálne underfitting podľa toho ako súdime úspešnosť modelov.

Confusion matrix je následovná



Najmenej chybných označení urobil žáner horror a skutočnosti išlo o comedy.

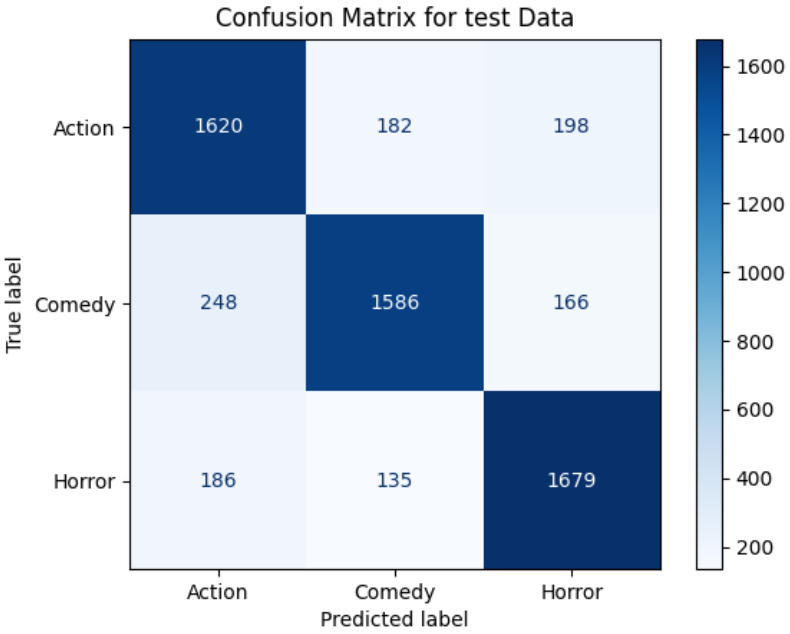
5.4 Vyhodnotenie rozhodovacích stromov

Pretože v tejto kategórií plánujem vyhodnotiť až 3 metódy tak pre tu najlepšiu potom zobrazím confusion matrix. Pre rozhodovací strom bol optimálny parameter max_depth = 100, pre náhodný les max_depth = 200, n_estimators = 500 a criterion = 'gini'. Pre gradient boosting je max_depth = 10 a learning_rate = 0.5. Pre tieto parametre dostávam hodnoty:

	Accuracy	Macro Precision	Macro Recall	Macro F1-Score
Rozhodovací strom	0.625	0.626	0.625	0.625
Náhodný les	0.808	0.810	0.808	0.808
Gradient Boosting	0.814	0.814	0.814	0.814

Table 2: Porovnanie rozhodovacieho stromu, náhodného lesu a gradient boosting

Najlepší model je gradient boosting, ktorý ma accuracy o ≈ 0.006 lepšiu ako náhodný les, ale jeho trénovanie trvalo veľmi dlho. Confusion matrix vyzerá:



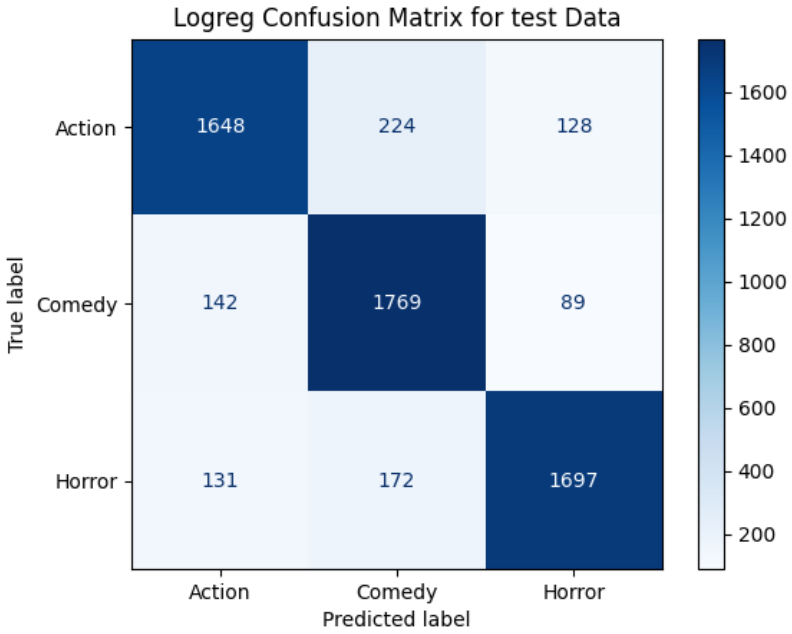
5.5 Vyhodnotenie logistickej regresie

Optimálne parametre sú C=10, penalty je l2 a solver je lbfgs (Limited-memory Broyden-Fletcher-Goldfarb-Shanno), čo je variant kvázi-Newtonovej metódy.

Pre tieto parametre dostávam výsledok:

- Accuracy : 0.852
- Macro Precision: 0.853
- Macro Recall: 0.852
- Macro F1-Score: 0.852

Confusion matrix je



5.6 Porovnanie

	Accuracy	Macro Precision	Macro Recall	Macro F1-Score
Naivne Bayes	0.846	0.852	0.846	0.845
SVC	0.854	0.856	0.854	0.854
Rozhodovací strom	0.625	0.626	0.625	0.625
Náhodný les	0.808	0.810	0.808	0.808
Gradient Boosting	0.814	0.814	0.814	0.814
Logistická regresia	0.852	0.853	0.852	0.852

Table 3: Porovnanie metód

Najlepší model je SVC a najhorší je náhodný les, ktorý možno nie je vhodný na tuto problematiku. Ale jeho vylepšená verzia náhodný les to vedela zlepšiť o dosť.

6 Záver

Ako je vidieť v tabuľke 3 tak hodnoty sa točili okolo ≈ 0.8. Toto bude znakom obmedzenia vstupu, aby model vedel lepšie pracovať ale prečo? Keď som skúmal dáta tak som si všimol film Mars Attacks![15] Ide o žáner Comedy a jeho popis je " fleet of Martian spacecraft surrounds the world's major cities and all of humanity waits to see if the extraterrestrial visitors have, as they claim, "come in peace." U.S. President James Dale receives assurance from science professor Donald Kessler that the Martians' mission is a friendly one. But when a peaceful exchange ends in the total annihilation of the U.S. Congress, military men call for a full-scale nuclear retaliation." a, keď som to skúsil dať do SVC modelu tak mi to vrátilo žáner Action. Ono to ale aj fakt tak znie a to bol ten problém. Niektoré popisy filmov vôbec neznejú ako ich skutočný žáner a preto sa nedokáže model dobre naučiť ale toto bola práve tá vec, ktorú som chcel zistiť. Nevybral som si tento projekt s tým, že pôjdem dať accuracy 0.99... Ale 0.854 je slušná hodnota a podobne to mali aj v niektorých článkoch, takže nepovažujem to za zlyhanie. Ak by som začal odznova tak by som nič nemenil v rámci tohto predmetu ale určite by som skúsil neuronové siete a iné pokročilé metódy. A na záver som veľmi spokojný s výberom projektu a dosť ma to bavilo si vyskúšať trochu NLP.

7 Appendix

Všetky súbory budú na mojom GitHube [`https://github.com/Martines1/machine-learning-projekt`]. Je tam pripravený dataset, ktorý si prešiel spracovaním a jednotlivé .py súbory, ktoré odkazujú na metódy sú v spustiteľnej forme. Kompletný dataset, ktorý nie je spracovaný tam neplánujem dať, kvôli jeho veľkosti ($\approx$ 520MB) ale je uvedený odkaz na stiahnutie.

References

[1] Wei-Ta Chu and Hung-Jui Guo. "Movie genre classification based on poster images with deep neural networks". In: *proceedings of the workshop on multimodal understanding of social, affective and subjective attributes*. 2017, pp. 39–45.

[2] Kenneth Ward Church. "Word2Vec". In: *Natural Language Engineering* 23.1 (2017), pp. 155–162.

[3] *DecisionTreeClassifier - scikit-learn*. `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html`. Accessed: 2025-01-01.

[4] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: `1810.04805 [cs.CL]`. URL: `https://arxiv.org/abs/1810.04805`.

[5] Roman Egger. "Text representations and word embeddings: Vectorizing textual data". In: *Applied data science in tourism: Interdisciplinary approaches, methodologies, and applications*. Springer, 2022, pp. 335–361.

[6] Ali Mert Ertugrul and Pinar Karagoz. "Movie genre classification from plot summaries using bidirectional LSTM". In: *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*. IEEE. 2018, pp. 248–251.

[7] *GradientBoostingClassifier - scikit-learn*. `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html`. Accessed: 2025-01-01.

[8] Margherita Grandini, Enrico Bagli, and Giorgio Visani. "Metrics for multi-class classification: an overview". In: *arXiv preprint arXiv:2008.05756* (2020).

[9] *GridSearchCV - scikit-learn*. `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`. Accessed: 2025-01-01.

[10] *IBM What is bag of words*. `https://www.ibm.com/think/topics/bag-of-words`. Accessed: 2025-01-01.

[11] Sanjay Kumar et al. "Movie genre classification using binary relevance, label powerset, and machine learning classifiers". In: *Multimedia Tools and Applications* 82.1 (2023), pp. 945–968.

[12] *langid library*. `https://github.com/saffsd/langid.py`. Accessed: 2025-01-01.

[13] Elizabeth D Liddy. "Natural language processing". In: (2001).

[14] *LogisticRegression - scikit-learn*. `https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LogisticRegression.html`. Accessed: 2025-01-01.

[15] *Mars Attacks! (1996) - The Movie DataBase*. https://www.themoviedb.org/movie/75-mars-attacks. Accessed: 2025-01-01.

[16] *MultinomialNB - scikit-learn*. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html. Accessed: 2025-01-01.

[17] *NLTK library*. https://www.nltk.org/. Accessed: 2025-01-01.

[18] *RandomForestClassifier - scikit-learn*. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. Accessed: 2025-01-01.

[19] Gabriel S Simões et al. "Movie genre classification with convolutional neural networks". In: *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2016, pp. 259–266.

[20] *SVC - scikit-learn*. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. Accessed: 2025-01-01.

[21] *SVC complexity - scikit-learn*. https://scikit-learn.org/stable/modules/svm.html#complexity. Accessed: 2025-01-01.

[22] *XGBoost library*. https://xgboost.readthedocs.io/en/stable/. Accessed: 2025-01-01.