# Predpoveď hodnotenia filmov

Projekt - Strojové učenie

**Autor:** Péter Dobsa

6. januára 2015

# Abstrakt

Dobsa, Péter: Predpoveď hodnotenia filmov (projekt zo strojového učenia)

Projekt je zameraný na natrénovanie algoritmu, ktorý podľa rôznych vlastností snaží predpovedať používateľské hodnotenie daného filmu. Pri trénovaní sú použité rozhodovacie stromy, rôzne prístupy ich skonštruovania a ensemble metódy: bagging a náhodné lesy.
Projekt obsahuje vlastnú implementáciu niektorých algoritmov, aj využitie voľne dostupnej knižnice scikit-learn, návrh ich aplikovania na spomínaný problém a experimentálne vyhodnotenie výsledkov.

**Kľúčové slová:** strojové učenie, rozhodovacie stromy, náhodné lesy, bagging

# 1 Úvod

Cieľom projektu bolo vybrať nejaký problém na ktoré sa dajú aplikovať myšlienky strojového učenia, zabezpečiť trénovacie a testovacie dáta, naimplementovať - prípadne vhodným spôsobom použiť nejaké existujúce - metódy na jeho riešenie a nasledne vyhodnotiť úspešnosť. Úloha, ktorú sme vybrali je preskúmanie vzťahov medzi popularitou a istými charakteristikami filmov a podľa nich predpovedať ich popularitu - presnejšie hodnotenie divákov podľa webového portálu IMDb (Internet Movie Database). IMDb je jedna z najväčších online databáz, ktorá eviduje rôzne informácie o filmoch a dovoľuje používateľom hlasovať na kvalitu na hodnotiacej škále 1 až po 10. IMDb skóre filmov sú vypočítané ako váhovaný priemer hlasov, pričom detaily váhovania nie sú verejné. Naše algoritmy snažia predpovedať práve tieto priemery / klasifikovať, či priemer bude pod alebo nad 60% (podľa inšpirácie od ďalšieho filmového portálu Rotten Tomatoes - ďalej len RT).

Projekt sa skladá z dvoch častí:

- tento písomný report obsahuje podrobný popis použitých dát, algoritmov; implementačné detaily a vyhodnocovanie úspešnosti
- elektronická príloha s použitými dátami a zdrojovými kódmi

# 2 Dáta

Dáta boli pozbierané prostredníctvom dvoch stránok: už spomínané IMDb, TMDb (The Movie Database) a Rotten Tomatoes (RT). Verejné API od IMDb je zle dokumentovaný a príliš obmedzení, takže namiesto priamych dotazov sme použili ďalšiu webovú službu: OMDb. OMDb pretransformuje neprehľadný *csv* odpoveď od IMDb na *json* formát a zároveň zahŕňa aj informácie z RT. TMDb poskytuje veľmi flexibilnú dotazovaciu knižnicu, takže zoznam filmov bol vytvorený podľa neho. (viď *datacollector.py* a *data.json*)
Keďže nás zaujímajú súčasné trendy, boli zvolené filmy z posledných troch rokov a jedine tie, o ktorých boli známe všetky atribúty, ktoré sme použili pri trénovaní: dokopy 2228 filmov. Tieto dáta sme použili pri trénovaní aj pri testovaní: trénovacia množina tvorila približne 65% všetkých dát, zvyšok bol pridelený do testovacej množiny. Približne 60% všetkých dát bol pozitívny (t.j. hodnotenie filmu nad 60%), zvyšok negatívny.
Atribúty, ktoré sme použili (v zátvorke je uvedený názov použitý v dátach):

- dĺžka filmu (runtime)
- ročné obdobie debutu (release date) - kvôli jednoduchosti sme použili trojice mesiacov
- cieľová skupina (rated) - podľa MPAA hodnotenia
- rozpočet (budget)
- žáner (genre) - rozlišujeme 10 základných žánrov: akčný, dobrodružný, dokumentárny, dráma, rodinný, fantasy, science fiction, romantický, komédia, thriller, a ešte ich kombinácie. Žánre sme reprezentovali 10-bitovým číslom: $n$-tý bit je aktívny, ak daný film patrí do $n$-tého žánra
- režisér (director) - každý režisér je reprezentovaný jedinečným id-čkom

- príbeh (plot) - číslenú hodnotu za príbeh sme pridelili podľa jednoduchej lexikálnej analýzi: slová dostali skóre na základe frekvencie výskytu, pričom každý výskyt bol odmenený podľa IMDb skóra filmu. Podľa očakávania najväčsie skóre dostali všeobecne použité slová ako spojky a častice, takže sme manuálne vybrali prvých 25 podstatných mien - atribútom je počet takýchto slov v príbehu. Pre zaujímavosť uvedieme prvých 10: life, young, love, world, family, years, time, story, friends, battle

- herci (actors) - každému hercu sme pridelili skóre podľa nasledujúceho vzorca:

$$\frac{\sum_{f \in F} imdb\,skóre f}{|F|}$$

, kde $F$ je množina filmov v ktorých sa daný herec objaví.
Výsledná hodnota je potom priemerné skóre hercov filmu.

# 3 Použité metódy

Ako základný princíp učenia sme sa rozhodli použiť **rozhodovacie stromy** z viacerých dôvodov: rozhodovacie stromy sú intuitívne, dobre prehľadné, ľahko pochopiteľné dátové štruktúry, vďaka čomu ich implementácia je menej náročná ako v prípade väčšine ostatných prístupov, výsledný model je ľahko vizualizovaľný a tým pádom aj interpretovateľný a pritom sú aplikovateľné na našu situáciu: dokážu predpovedať spojité číslené hodnoty, aj klasifikovať do diskrétnych kategórií. Ďalší dôvod ich použitia je viac subjektívny: v rámci kurzu sme im venovali menšiu pozornosť a boli sme zvedaví ako dobre fungujú v praxi.
Idea rozhodovacích stromov je jednoduchá: sú to $n$-árne stromy (obvykle $n = 2$), kde cesta od koreňa až k nejakému listu reprezentuje proces klasifikácie. Každý vnútorný vrchol ukrýva v sebe test na nejaký atribút, na základe čoho sa rozhodneme v ktorom jeho nasledovníku pokračujeme, pričom každý list je jedna z trieda klasifikácie. Ľahko potom vidíme, že pre ľubvoľné dáta môžeme zostrojiť veľa rôznych (potenciálne nekonečne veľa) rozhodovacích stromov. Čím viac snažíme daný strom prispôsobiť k trénovacím dátam, tým viac sa stane komplexnejším a zároveň tým viac stráca schopnosť generalizácie. Základným ťažiskom skonštruovania rozhodovacích stromov je teda uhádnuť najmenšiu komplexitu a štruktúru, ktorá ešte dáva prijateľné výsledky. Je to známy NP-úplny problém a v praxi rozhodovacie stromy sú často vytvorené podľa rôznych heuristík. Uvedieme si teraz 3 algoritmy, ktoré sú dôležité z hľadiska tohto projektu:

- **ID3 (Iterative Dichotomiser 3)** - využíva koncept *entrópie* - veľkosti neistoty v danom systéme. Algoritmus buduje strom zhora nadol a každému vrcholu je priradená množina dát. Začína sa s koreňom, ktorá obsahuje celú trénovaciu množinu. Zoberie najinformatívňejší atribút (t.j. atribút s najmenšou entrópiou), podľa neho rozdelí množinu dát na $n$ podmnožín, kde $n$ je počet hodnôt, ktorých daný atribút môže nadobúdať, a na základe hodnoty zvoleného atribútu priraďuje ich do príslušnej množiny. Pre každú podmnožinu vytvorí nový vrchol, ktoré tvoria nasledovníkov pôvodného vrchola. To isté opakuje pre každý znovuvytvorený vrchol - s rozdielom, že už predtým použité atribúty nemôžu použiť na rozdelenie množiny dát - až kým nedostane k niektorej z terminálnych podmienok:

  1. každé dáto množiny patrí do tej istej triedy - vytvorí list pre danú triedu
  2. nemá atribút podľa čoho by dokázal rozdeliť dáta - vytvorí list s triedou, ktorá má najviac reprezentantov v množine

3. množina vrcholu je prázdna - vytvorí list s triedou, ktorá má najviac reprezentantov v množine rodiča

Entrópia množiny $S$ je definovaná nasledujúcim vzťahom:

$$H(S) = -\sum_{x \in X} p(x) \log_2 p(x)$$

, kde $X$ je množina klasifikačných tried, ktoré sa objavia v $S$, a $p(x)$ je pomer počtu dát s triedou $x$ k veľkosti $S$.
Informatívnosť atribútu $A$ vzhľadom na množinu $S$ je definovaná ako:

$$I(A, S) = H(S) - \sum_{t \in T} p(t) H(t)$$

, kde $T$ je množina podmnožín, ktorých vytvoríme z množiny $S$ podľa atribútu $A$, $p(t)$ je pomer veľkosti $t$ k veľkosti $S$

- Hlavnou nevýhodou ID3 je to, že predpokladá, že hodnota každého atribútu je diskrétna. Existuje jeho vylepšená verzia, ktorá dokáže pracovať aj so spojitými atribútmi: algoritmus **C4.5**. Rozdelenie v tomto prípade funguje nájdením prahovej hodnoty a všetky dáta s menšou hodnotou sa dostanú do ľavého syna, ostatné do pravého (prípadne množinu reálnych čísel môžeme rozdeliť na $n$ intervalov a podľa nich rozdelit dáta na $n$ podmnožín). Nakoniec C4.5 prekonvertuje výsledný strom na množinu $if-then$ pravidiel - tzv. *ruleset*. Je ale stále len použiteľné na diskrétnu klasifikáciu.

- **CART (Classification and Regression Trees)** je veľmi podobný algoritmu C4.5, ale už podporuje spojité cieľové hodnoty (regresiu) a na rozdiel od C4.5 nevytvorí rule set a výsledný strom je vždy binárny.

Použitie rozhodovacích stromov nie je vždy spoľahlivý spôsob predpovedania hodnôt: veľmi malé zmeny v trénovacej množine môžu zapríčiniť veľké zmeny vo výslednom strome, a na druhej strane aj skonštruovanie stromu je často nedeterministická operácia, keďže v implementáciach niektoré rozhodnutia (napr. určovanie prahovej hodnoty v C4.5) sú často stochastické. Na zmiernenie tohto problému existujú metódy, ktoré finálny výsledok počítajú na základe viacerých rozhodovacích stromov (les rozhodovacích stromov) - tzv. *ensemble* metódy:

- **bagging** - z pôvodnej trénovacej množiny vytvorí $m$ nových trénocích množín $x_m$ veľkosti $k$ ($k < n$, kde $n$ je veľkosť pôvodnej množiny) náhodným výberom, pričom ten istý príklad sa môže vyskytnúť viac ako jedenkrát. Následne pre každú množinu natrénuje rozhodovací strom $t_m$ a výsledok určuje ako priemer pridikcií týchto stromov:

$$h_{bag} = \frac{1}{m} \sum_{i=1}^{m} t_i(x_i)$$

  V prípade klasifikácie sa rozhodne podľa počtu hlasov.

- **náhodný les (random forest)** - alebo *atribút bagging*, funguje na rovnakom princípe ako bagging, s tým rozdieľom, že pri vytváraní stromu v kroku rozdelenia sa vždy prenáša len nejaká náhodná podmnožina atribútov, za účelom nájdenia najsilnejších prediktorov.

# 4 Implementácia

V projekte porovnávame 4 rôzne algoritmy implementované v programovacom jazyku Python (verzia 3.4.2) prostredníctvom knižnice *scikit-learn* (http://scikit-learn.org/). Implementácia algoritmov sa nachádza v súbore *learn.py*, pričom samotné testovanie v *main.py* (návod na spustenie v priloženom *readme.txt*).
Algoritmy:

- vlastná, naivná implementácia rozhodovacieho stromu (trieda *dtree_custom*) pomocou algoritmu C4.5, bez transformácie na množinu pravidiel (t.j. počas trénovania sa vytvorí strom, ktorý je potom priamo použitý na testovanie). Skonštruovanie stromu je plne deterministické, pre rovnakú trénovaciu množinu algoritmus vždy vytvorí rovnaký binárny strom, ktorý klasifikuje dáta do dvoch tried (pozitívna alebo negatívna), pričom predpokladá, že atribúty sú spojité. Ako prahovú hodnotu rozdelenia používa priemer hodnôt daného atribútu.

- scikit-learn implementácia rozhodovacieho stromu (*dtree*). Knižnica používa algoritmus CART, obsahuje teda 2 verzie štruktúry: jedna na regresiu a jedna na klasifikáciu (*DecisionTreeClassifier* a *DecisionTreeRegressor*), pričom vytvorenie stromu nie je deterministické: pre rovnaké dáta môže vytvoriť rôzne stromy. Knižnica navyše umožňuje nastavať niektré vlastnosti výsledného stromu, ako maximálna hĺbka (max_depth), minimálny počet príkladov v listoch (min_samples_leaf) a minimálny počet príkladov v každej množine po rozdelení (min_samples_split).

Taktiež sme otestovali už spomínané ensemble metódy:

- bagging - aj pre vlastnú, aj pre scikit implementáciu, t.j. dokopy 3 verzie

- scikit verzia náhodných lesov pre regresiu a klasifikáciu (*RandomForestClassifier* a *RandomForestRegressor*)

# 5 Výsledky

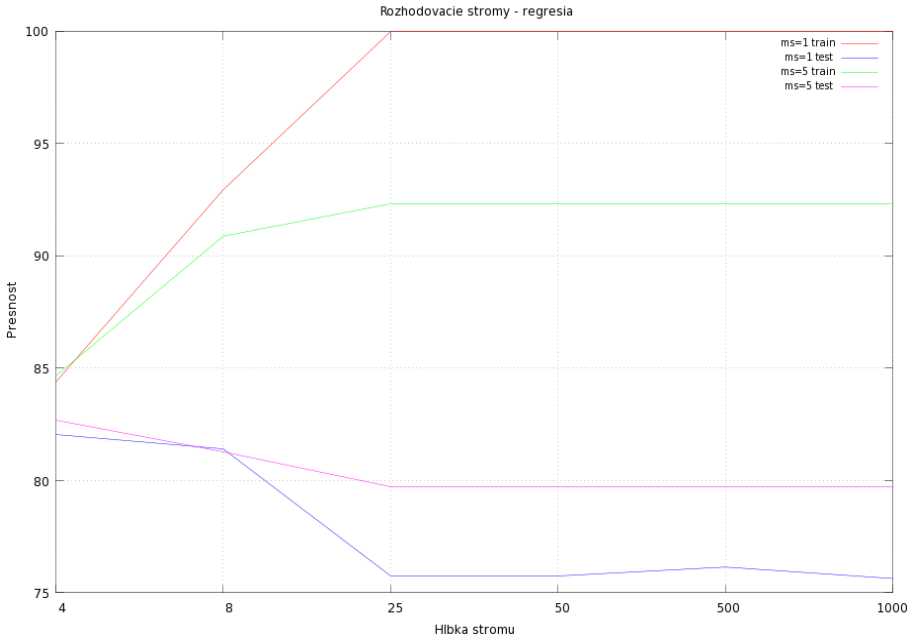
Algoritmy sú otestované s rôznymi nastaveniami parametrov, pri rozhodovacích stromoch to znamená maximálne povolenú hĺbku (md) a minimálny počet príkladov v listoch / pri rozdelení (ms - vždy dostali rovnakú hodnotu), pričom pri baggingu máme veľkosť lesa (fc) a počet príkladov v trénovacích množinách (sc). Navyše väčšina algoritmov podporuje regresiu a klasifikáciu, použitý režim je označený písmenom $r$, resp. $c$ a každú verziu sme otestovali aj na trénovacích aj na testovacích dátach. Je tu uvedená veľká časť dosiahnutých výsledkov, celý zoznam nájdete v prílohe *results.txt*.
Vlastná implementácia rozdhodovacieho stromu dokázala klasifikovať trénovacie dáta s presnosťou 85.29%, pričom testovacie dáta s presnosťou 80.25%. Podľa očakávania, scikit stromy s vhodnými parametrami dosiahli oveľa väčšiu presnosť. Z tabuľky 1 a 2 môžeme vidieť, že povolená hĺbka a požadovaný počet príkladov naozaj ovplyvňuje výsledok: zložitejšie stromy s väčšou hĺbkou preučia dáta - ak to skombinujeme s tým, že dovolíme stromu vytvoriť list aj pre jeden príklad, perfektne predpovie trénovacie dáta, pričom má problemy s testovacími.
Stromy spravidla nemali väčšiu hĺbku ako 25 (príklad neobmedzeného stromu - t.j. žiadne obmedzenie na hĺbku a počet príkladov - je v prílohe *treeunlim.pdf*, má hĺbku 18), takže v tabuľke nad týmto číslom sú väčšinou rovnaké výsledky.

Tabuľka 1: Rozhodovacie stormy - klasifikácia, trénovacie / testovacie dáta

| | **ms=1** | **ms=5** | **ms=25** | **ms=50** | **ms=500** |
|---|---|---|---|---|---|
| **md = 4** | 92.88/92.05 | 92.88/91.92 | 92.47/92.17 | 91.78/91.66 | 91.78/91.66 |
| **md = 8** | 97.09/89.10 | 95.09/90.12 | 92.47/92.17 | 91.78/91.66 | 91.78/91.66 |
| **md = 25** | 100.0/88.71 | 95.09/90.25 | 92.47/92.17 | 91.78/91.66 | 91.78/91.66 |
| **md = 50** | 100.0/88.33 | 95.09/90.51 | 92.47/92.17 | 91.78/91.66 | 91.78/91.66 |
| **md = 500** | 100.0/88.33 | 95.09/90.51 | 92.47/92.17 | 91.78/91.66 | 91.78/91.66 |
| **md = $\infty$** | 100.0/88.71 | 95.09/90.25 | 92.47/92.17 | 91.78/91.66 | 91.78/91.66 |

Tabuľka 2: Rozhodovacie stormy - regresia, trénovacie / testovacie dáta

| | **ms=1** | **ms=5** | **ms=25** | **ms=50** | **ms=500** |
|---|---|---|---|---|---|
| **md = 4** | 84.39/82.05 | 84.66/82.69 | 84.66/83.20 | 84.73/82.56 | 56.69/58.33 |
| **md = 8** | 92.95/81.41 | 90.88/81.28 | 87.70/85.12 | 87.08/83.97 | 56.69/58.33 |
| **md = 25** | 100.0/75.76 | 92.33/79.74 | 87.70/85.12 | 87.08/83.97 | 56.69/58.33 |
| **md = 50** | 100.0/75.76 | 92.33/79.74 | 87.63/85.12 | 87.08/83.97 | 56.69/58.33 |
| **md = 500** | 100.0/76.15 | 92.33/79.74 | 87.70/85.12 | 87.08/83.97 | 56.69/58.33 |
| **md = $\infty$** | 100.0/75.64 | 92.33/79.74 | 87.63/85.12 | 87.08/83.97 | 56.69/58.33 |




Tabuľka 3: Bagging - vlastná implementácia stromov, trénovacie / testovacie dáta

| | **fc=500** | **fc=1000** | **fc=1500** |
|---|---|---|---|
| **sc = 500** | 87.70 / 84.35 | 87.15 / 82.56 | 87.01 / 82.82 |
| **sc = 1000** | 87.91 / 84.23 | 87.15 / 83.07 | 87.29 / 83.07 |
| **sc = 1500** | 87.43 / 84.10 | 87.15 / 82.94 | 87.01 / 82.94 |

Tabuľka 4: Bagging - scikit stromy, klasifikácia, fc = 500, sc = 1000, trénovacie / testovacie dáta

| | **ms=1** | **ms=5** | **ms=25** | **ms=50** | **ms=500** |
|---|---|---|---|---|---|
| **md = 4** | 92.54/92.05 | 92.61/92.05 | 92.47/92.17 | 91.78/91.66 | 59.32/58.58 |
| **md = 8** | 96.47/92.69 | 94.19/92.56 | 91.98/91.79 | 91.78/91.66 | 59.32/58.58 |
| **md = 25** | 99.58/93.20 | 94.33/92.17 | 92.54/92.30 | 91.78/91.66 | 59.32/58.58 |
| **md = 50** | 99.72/93.46 | 94.33/92.30 | 92.40/92.17 | 91.78/91.66 | 59.32/58.58 |
| **md = 500** | 99.79/93.20 | 94.47/92.43 | 92.54/92.30 | 91.78/91.66 | 59.32/58.58 |
| **md = $\infty$** | 99.51/92.82 | 94.40/92.17 | 92.47/92.17 | 91.78/91.66 | 59.32/58.58 |

Tabuľka 5: Bagging - scikit stromy, regresia, fc = 500, sc = 1000, trénovacie / testovacie dáta

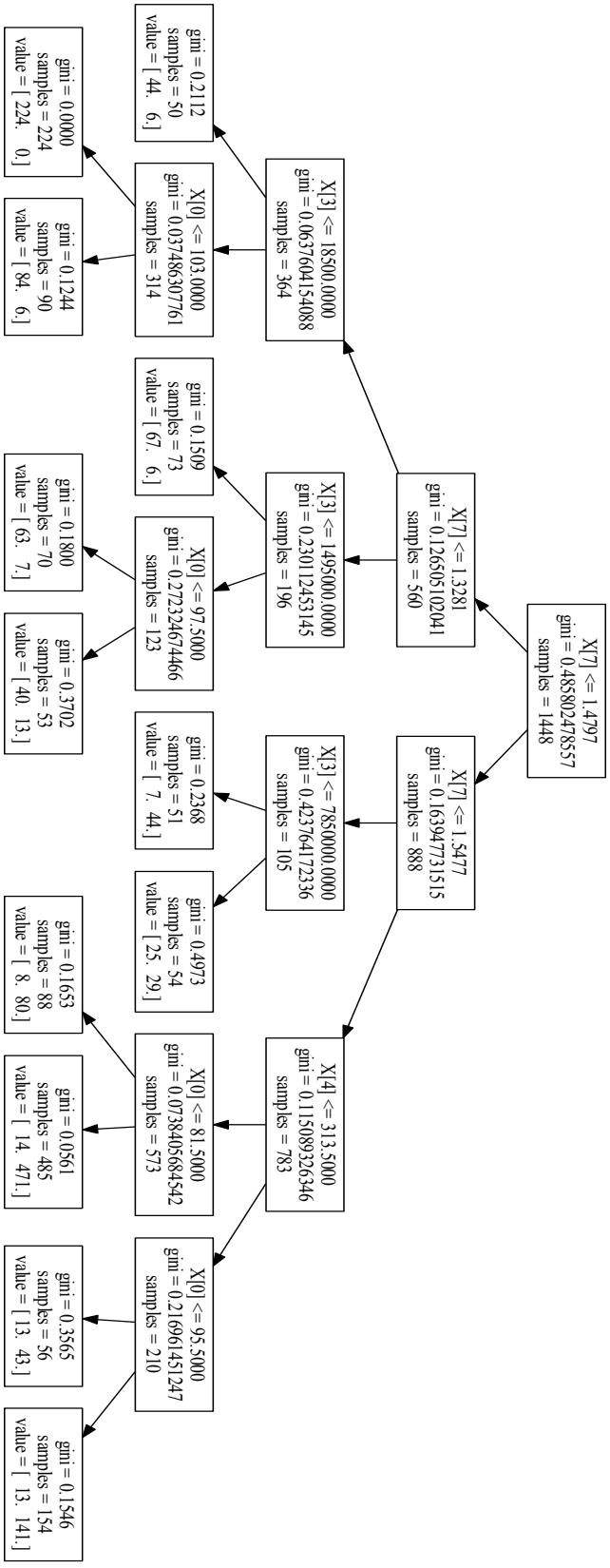
| | **ms=1** | **ms=5** | **ms=25** | **ms=50** | **ms=500** |
|---|---|---|---|---|---|
| **md = 4** | 86.53/84.23 | 86.74/84.35 | 86.18/84.10 | 84.87/84.87 | 46.82/48.46 |
| **md = 8** | 93.16/85.12 | 91.78/85.25 | 88.53/84.74 | 85.70/84.23 | 46.89/48.46 |
| **md = 25** | 96.40/85.51 | 92.67/85.51 | 88.53/84.61 | 85.91/84.48 | 46.47/47.94 |
| **md = 50** | 96.06/85.64 | 92.74/85.76 | 88.60/84.74 | 85.70/84.61 | 46.75/48.71 |
| **md = 500** | 96.33/86.02 | 92.88/85.76 | 88.46/84.61 | 85.77/84.23 | 46.68/48.58 |
| **md = $\infty$** | 96.13/85.64 | 92.95/85.76 | 88.67/84.74 | 85.70/84.48 | 46.75/48.46 |

Tabuľka 6: Náhodné lesy, klasifikácia, trénovacie / testovacie dáta

| | **ms=1** | **ms=5** | **ms=25** | **ms=50** | **ms=500** |
|---|---|---|---|---|---|
| **md = 4** | 92.26/91.41 | 91.98/91.66 | 91.64/90.51 | 91.78/91.79 | 59.32/58.58 |
| **md = 8** | 95.51/91.79 | 93.85/91.92 | 92.26/91.92 | 91.78/91.66 | 59.32/58.58 |
| **md = 25** | 99.72/93.07 | 94.12/92.30 | 91.78/91.66 | 91.85/91.15 | 59.32/58.58 |
| **md = 50** | 99.79/92.82 | 93.85/92.30 | 91.91/91.92 | 91.78/90.89 | 59.32/58.58 |
| **md = 500** | 99.51/93.33 | 93.92/91.92 | 91.98/91.92 | 91.85/91.66 | 59.32/58.58 |
| **md = $\infty$** | 99.30/93.33 | 93.43/92.56 | 91.98/91.92 | 91.71/91.53 | 59.32/58.58 |

Tabuľka 7: Náhodné lesy, regresia, trénovacie / testovacie dáta

| | **ms=1** | **ms=5** | **ms=25** | **ms=50** | **ms=500** |
|---|---|---|---|---|---|
| **md = 4** | 85.70/83.84 | 86.74/83.84 | 86.04/84.10 | 84.04/83.33 | 41.09/41.66 |
| **md = 8** | 94.06/85.38 | 91.50/85.51 | 87.84/84.10 | 85.08/84.23 | 41.09/41.66 |
| **md = 25** | 99.72/93.07 | 94.12/92.30 | 91.78/91.66 | 91.85/91.15 | 41.09/41.66 |
| **md = 50** | 97.58/86.28 | 92.26/84.74 | 88.32/84.35 | 85.42/84.35 | 41.09/41.66 |
| **md = 500** | 97.99/84.74 | 92.67/85.38 | 87.70/84.61 | 85.01/84.23 | 41.09/41.66 |
| **md = $\infty$** | 98.06/84.48 | 92.74/85.89 | 88.32/84.74 | 84.46/83.71 | 41.09/41.66 |




Ukážka výsledného stromu pri md=4, ms=50

Môžeme si ešte všimnúť, že nad 25 ms výsledky sú rovnaké bez ohľadu na hĺbku stromu - pravdepodobne vďaka tomu, že z testovacích dát nie je až tak veľa, pričom listy vyžadujú dosť príkladov a výsledné stromy vždy majú menšiu výšku ako 4 - čo je naše testovacie minimum - t.j. sú rovnaké. Taktiež z regersie vidíme, že ak nastavime príliš vysokú hodnotu ms, strom nevie učiť z dát a tým pádom ani generalizovať pre neznáme dáta. Zaujímavo, toto sa neprejaví pri klasifikácii - aspoň pre jeden strom, ak si pozreme výsledky baggingu, pri vysokom ms už les stromov neklasifikuje úspešne. Zjave sa to závisí od trénovacej množiny.
Ensemble metódy vylepšujú skoro v každom prípade skóre jedného stromu (okrem vtedy, keď ms je 500), čo nie je prekvapivé. Náhodné lesy ale dosiahli približne rovnaké výsledky ako bagging, čo implikuje, že nie je silná korelácia medzi jednotlivými atribútmi. Pokúsili sme si vynechať niektoré n-tice atribútov, čo tiež potrvdil túto hypotézu: jediný silný prediktor je skóre hercov. Jeho vynechanie zapríčiní zhoršenie výsledku o 30-40%, pričom absencia ostatných atribútov spôsobuje len malé zmeny - resp. žiadne.

# 6 Záver

Myslíme si, že projekt svoje ciele úspešne splnil, výstupom sú algoritmy, ktoré s pomerne vysokou presnosťou - vlastné implementácie okolo 80%, implementácie prostredníctvom knižnice okolo 90% - dokážu predpovedať takú neistú informáciu, ako priemerné používateľské hodnotenie filmov - aspoň na naších testovacích dátach.
Plány do budúcosti:

- pracovali sme s pomerne malou množinou príkladov: rozšíriť trénovaciu a testovaciu množinu a zisiť ako to ovplyvňuje presnosť
- náhodné lesy nás presvedčili, že použité atribúty nie sú úplne ideálne: odstrániť nadbytočné atribúty a nahradiť ich s inými (napr. či daný film je pokračovanie série)
- vyskúšať iné metódy strojového učenia na riešenie tohto problému a porovnať ich s rozhodovacími stromami

**Zdroje:** scikit-learn dokumentácia algoritmov, prednášky, wiki stránka predmetu