

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
FIIT-100241-75561

Peter Markuš

Interaktívne tutoriály v strojovom učení

Bakalárska práca

Študijný program: Informatika
Študijný odbor: 9.2.1 Informatika
Miesto vypracovania: Ústav informatiky, informačných systémov a softvérového
inžinierstva, FIIT STU, Bratislava
Vedúci práce: Mgr. Peter Laurinec
Máj 2018

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

ZADANIE BAKALÁRSKEHO PROJEKTU

Meno študenta: **Markuš Peter**

Študijný odbor: Informatika

Študijný program: Informatika

Názov projektu: **Interaktívne tutoriály v strojovom učení**

Zadanie:

Metódy strojového učenia, ktoré riešia úlohy analýzy dát, nachádzajú uplatnenie v praxi v stále väčšom meradle. Vyvijajú sa stále viac komplexnejšie a komplikovannejšie metódy, ktoré zvládajú tým pádom aj viac zložitejšie úlohy. Môžu to byť metódy zhľukovania, regresie, klasifikácie alebo aj redukcie dimenzionality. Predsudok voči tejto komplexite sa dá prekonať napríklad vizualizáciami procesov v metóde strojového učenia.

Analyzujte a navrhnite vizualizácie (napr. interaktívna webová aplikácia), ktoré vysvetlia fungovanie zvolenej metódy strojového učenia, ktorá bude slúžiť k výučbe. Váš návrh následne overte pomocou opäťovne vykonateľných dokumentov (tutoriálov), ktoré budú obsahovať analýzy zo zvolených netriviálnych prípadových štúdií. Takéto dokumenty budú obsahovať aj časti programu a interaktívne vizualizácie.

Práca musí obsahovať:

Anotáciu v slovenskom a anglickom jazyku

Analýzu problému

Opis riešenia

Zhodnotenie

Technickú dokumentáciu

Zoznam použitej literatúry

Elektronické médium obsahujúce vytvorený produkt spolu s dokumentáciou

Miesto vypracovania: Ústav informatiky, informačných systémov a softvérového inžinierstva, FIIT
STU, Bratislava

Vedúci projektu: Mgr. Peter Laurinec

Termín odovzdania práce v zimnom semestri : 12.12.2017

Termín odovzdania práce v letnom semestri: 10.5.2018

**SLOVENSKÁ TECHNICKÁ UNIVERZITA
V BRATISLAVE**

Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

1

prof. Ing. Pavol Návrat, PhD.
riaditeľ ÚSISI

Bratislava 18.9.2017

Čestne vyhlasujem, že som túto prácu vypracoval samostatne, na základe konzultácií a s použitím uvedenej literatúry.

V Bratislave, 10.5.2018

Peter Markuš

Pod'akovanie

Ďakujem vedúcemu práce Mgr. Petrovi Laurincovi za dostupnosť, podnetné pripomienky a pomoc pri získavaní dát na vyhodnotenie výsledkov. Vďaka patrí všetkým participantom tutoriálov spolu aj s tými, čo ma podporovali pri tvorbe tejto práce.

Anotácia

Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Študijný program: Informatika

Autor: Peter Markuš

Bakalárska práca: Interaktívne tutoriály v strojovom učení

Vedúci bakalárskeho projektu: Mgr. Peter Laurinec

December 2017

Cieľom bakalárskej práce je podať základný prehľad o strojovom učení a jeho používaných prístupov. Najväčší dôraz bude kladený na metódy klasifikácie, pri ktorých sa budú používať dva rôzne algoritmy zamerajúce sa na textové dátá. Hlavnou úlohou práce bude vysvetliť fungovanie týchto algoritmov používateľom čo najjednoduchším možným spôsobom prostredníctvom vykonateľnej príručky a interaktívnej webovej aplikácie. Vykonateľná príručka bude poskytovať interaktívne programové prostredie v ktorom budú riešené praktické úlohy so sprevádzajúcim textom a nápomocnými vizualizáciami. Najväčšia pozornosť bude venovaná Naivnému Bayesovskému klasifikátoru, pri ktorom budú poskytované rozsiahle interaktívne vizualizácie, umožňujúce sledovať hlbšie jeho výpočtové procesy cez webovú aplikáciu. Pomocou interakčných prvkov bude možné experimentovať s rôznymi konfiguráciami algoritmu a pozorovať vzťahy medzi vzorkami dát a ich atribútov cez grafy.

Annotation

Slovak University of Technology Bratislava

FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

Degree course: Informatics

Author: Peter Markuš

Bachelor Thesis: Interactive tutorials in machine learning

Supervisor: Mgr. Peter Laurinec

December 2017

The aim of the bachelor thesis is to provide basic overview of machine learning and its used approaches. The greatest emphasis will be placed upon classification methods, in which will be used two different algorithms focusing on the text data in particular. The main task will be to explain the processes of these algorithms to users in the simplest way possible through executable document and interactive web application. Executable document will provide an interactive programming environment in which practical tasks are solved with accompanying text and helpful visualizations. The greatest attention will be paid to the Naive Bayesian classifier, along with providing extensive interactive visualizations to allow deep tracking of its computational processes through web application. By using interaction elements, we will be able to experiment with different algorithm configurations and observe relationships between data samples and its attributes via graphs.

Obsah

1 Úvod	1
2 e-Learning v strojovom učení	5
2.1 Metriky na meranie efektivity tutoriálov	5
2.2 Nástroje pre podporu vzdelávania	6
2.2.1 Jupyter Notebook	7
2.2.2 Framework Dash	8
3 Teória strojového učenia	9
3.1 Metódy strojového učenia	9
3.2 Klasifikačný model v strojovom učení	11
3.2.1 Zhromažďovanie dát	12
3.2.2 Predspracovanie dát	12
3.2.3 Exploratívna analýza	13
3.2.4 Budovanie modelu	13
3.2.5 Evaluácia modelu	13
3.3 Zložky chýb v predikčnom modeli	17
4 Návrh riešenia	21
5 Vybrané algoritmy pre textovú klasifikáciu	23
5.1 Naivný Bayesovský klasifikátor	24
5.2 Algoritmus podporných vektorov	27
6 Použité nástroje na podporu vzdelávania	31
6.1 Tutoriál v Jupyter Notebooku	31
6.1.1 Naivný Bayesovský klasifikátor	31
6.1.2 Algoritmus podporných vektorov	32
6.2 Tutoriál s frameworkom Dash	34
6.2.1 Vlastnosti trénovacej množiny	34
6.2.2 Klasifikačné parametre a ich výsledky	35
6.2.3 Atribúty trénovacieho datasetu	37
6.2.4 Pravdepodobnosti vzoriek a slov	37

6.2.5 Klasifikačný proces pravdepodobností	39
6.2.6 Jedinečnosť slov pre každú kategóriu	40
6.2.7 Text vzorky	40
7 Zhodnotenie tutoriálov	41
7.1 Obsah a spôsob vyhodnocovania.....	41
7.2 Vyhodnotenie výsledkov	42
8 Záver.....	45
Literatúra.....	47
A Technická dokumentácia	A-1
A.1 Jupyter Notebook	A-1
A.2 Aplikácia Dash.....	A-1
B Zhodnotenie plánu práce	B-1
C Použité materiály pri testovaní.....	C-1
C.1 Tutoriál v Jupyter Notebooku	C-1
C.2 Webová aplikácia Dash.....	C-12
C.2.1 Dash príručka	C-14
C.3 Dotazníky	C-19
C.3.1 Strojové učenie s Naivným Bayesovským klasifikátorom a SVM	C-19
C.3.2 Naivný Bayesovský klasifikátor.....	C-22
D Obsah elektronického média	D-2

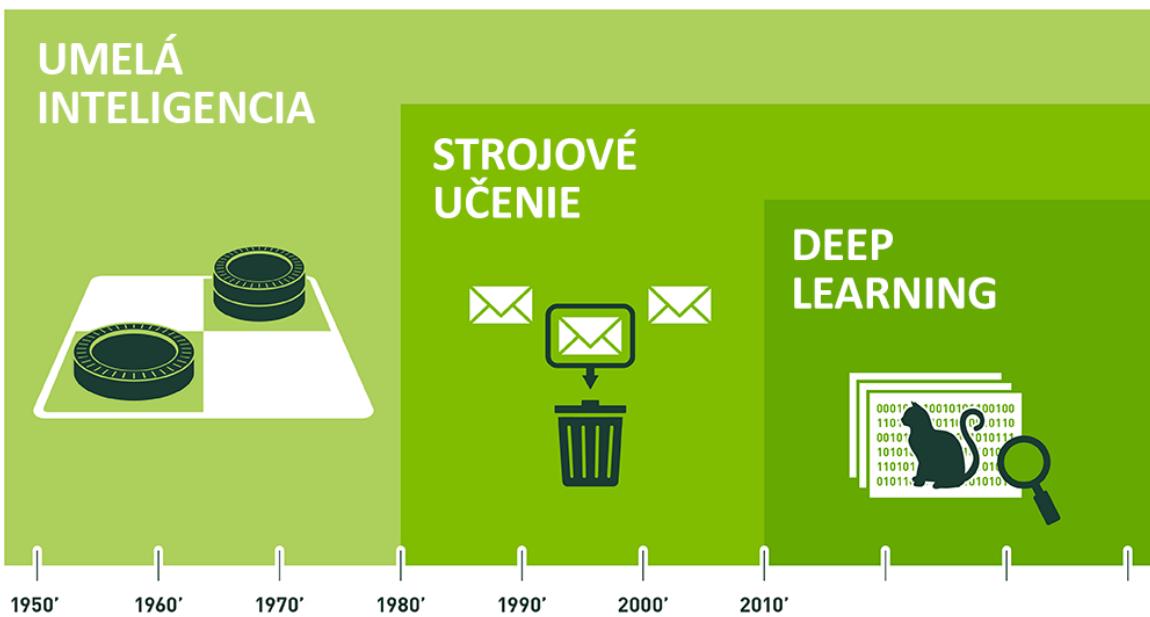
1 Úvod

V súčasnosti sa so strojovým učením stretávame čoraz viac, hoci si to ľudia nemusia vôbec uvedomovať. Vzorovým príkladom môže byť vyhľadávanie na internete, pri ktorom sa softvér naučil vhodne vyhodnocovať výber stránok podľa predstáv používateľa. Taktiež pri prezeraní emailov je to naučený spamovací filter rozlišujúci relevantné emaily od spamov. Schopnosť systém učiť sa očividne prináša množstvo nových možností riešenia komplexných problémov, ktorými sa zaoberá oblasť umelej inteligencie.

Pôvodne sa pri zstrojovaní inteligentných systémov vychádzalo z algoritmov ako hľadať riešenia v problémoch. Napríklad tak, ako pri hľadaní najkratšej cesty z bodu X do bodu Y. Ak by sme však takýmto spôsobom chceli riešiť mechanizmus pre odporúčacie systémy, ktoré sú implementované napríklad u Amazonu, Youtubu či Netflixu, boli by potrebné modifikácie algoritmov pre každého individuálneho používateľa zvlášť, čo by zrovna nebolo reálne riešenie. Jediným spôsobom ako predísť problémom takéhoto typu je zstrojiť systém, ktorý je schopný sa prispôsobovať novým podmienkam a učiť sa samostatne. Je tăžkopádne považovať systém za inteligentný bez možnosti učenia sa, preto je strojové učenie súčasťou umelej inteligencie.

Jednou z úloh umelej inteligencie je napodobňovať aj ľudskú inteligenciu. Aby sme sa k tomu približovali v čo najbližšej možnej podobe, bolo by vhodné sa inšpirovať práve aj ľudským “hardvérom”, mozgom. Mnoho vedcov verí, že prostredníctvom neurónových sietí, napodobňujúce spôsob fungovania mozgu sa dosiahnu najlepšie možné výsledky. V súčasnosti patrí medzi najpresnejšie metódy riešenia problémov v strojovom učení. Neurónové siete boli vynájdené už dávnejšie, ich potenciál sa však ukázal až neskôr pri výkonnejšej výpočtovej sile a internetu, cez ktorý bolo možné čerpať veľké množstvá dát. Boli to práve dátá, ktoré prinášali možnosť zvyšovať zložitosť procesov prostredníctvom hlbokých neurónových sietí. Mnohé známe spoločnosti ako Microsoft a Google investujú do výskumu v tejto oblasti, o ktorej sa hovorí Deep learning [1].

Umelá inteligencia čoraz viac naprieduje a je obohatovaná stále novými prístupmi k jej realizácii. Vhodným zhrnutím historických pokrokov vyjadruje nasledujúci obrázok od NVIDIAE.



Obrázok 1.1: História umelej inteligencie od NVIDIA¹

Čo je to strojové učenie? Podobne ako je to pri umelej inteligencii, ľažko je podať jeho výstižnú definíciu. Opäť sa môžu vyskytnúť mylné predstavy o učení tak, ako je to s inteligenciou. Pre jednoduchosť pod pojmom učenie v kontexte strojov si môžme predstaviť adaptívny prístup riešenia úloh v tom zmysle, že ich systém vyrieší po každom pokuse efektívnejšie. Takému učeniu zodpovedá aj jedna zo známych definícií strojového učenia od Arthurua Samuela. Jeho myšlienka spočíva v tom, že strojové učenie sa zaoberá metódami učenia sa programov tak, aby vedeli adekvátne odpovedať na nové vstupy dát bez toho, aby na ne boli explicitne naprogramovaný.

V dnešnom svete sú dáta jedným z najdôležitejších zdrojov informácií. Je ich však také veľké množstvo, že ich nie je možné spracovať manuálne alebo bežnými štatistickými metódami. Bývajú vytvorené v biznisových oblastiach ako je marketing, obchod, finančie alebo aj pri výskume v medicíne a biológii. Všetky tieto dáta je možné analyzovať a objavovať v nich skryté vlastnosti a vzťahy interpretujúce nové znalosti. V medicíne alebo biológii prostredníctvom elektronických dát napríklad môžeme objavovať skryté stránky rakovín či génov pre ich lepšie

¹ COPELAND, Michael. What's the Difference Between Artificial Intelligence, Machine Learning, and Deep Learning?. Nvidia [online]. California, USA, 2016 [cit. 2018]. Dostupné z: <http://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>

porozumenie. Pre získavanie nových znalostí z dostupných dát vznikla nová interdisciplinárna oblasť zvaná objavovanie znalostí (data mining). Používa najmä prístupy strojového učenia, dopomôct však môžu aj databázové systémy, štatistika či jednoduché vizualizačné techniky [1].

2 e-Learning v strojovom učení

Jeden z najpodstatnejších vplyvných udalostí vzdelávania nastal práve aj pri vzniku internetu. Poznatky sa stali ľahko šíriteľné a dostupné všade tam, kde by inak prístupné neboli. Bývajú sprostredkovávané na webových stránkach najmä prostredníctvom videí a dokumentov, ktoré je možné rozšíriť aj novými inovatívnejšími metódami akými sú napríklad aj interaktívne vizualizácie. Mnoho z týchto metód nieje možné použiť pri tradičnom vzdelávaní. Potenciál E-learningu v tomto smere stále rastie a stáva lacnejšou, dostupnejšou a často krát aj efektívnejšou metódou pre vzdelávanie.

Dnes je množstvo materiálov aj o strojovom učení a väčšina z nich je spracovávaných prostredníctvom internetových článkov, kurzov či tutoriálov. Kurzy sa narozenie od tutoriálov snažia pokryť širší záber z danej oblasti, pričom tutoriály sa dajú chápať ako jeho súčasti, ktoré poskytujú návod len k jednej podoblasti. Najvzornejším a v súčasnosti najlepšie ohodnocovaným príkladom internetového kurzu strojového učenia je od spoluorganizátora vzdelávacej spoločnosti Coursera Andrew Ng, ktorý záhytáva všetky podstatné témy strojového učenia. Jeho dlhoročné skúsenosti ako profesor strojového učenia priniesli kvalitný výklad prostredníctvom videí, ktoré boli obohatované aj intuitívnymi vizualizáciami. Výklad však vo všeobecnosti nieje jedinou metrikou pre meranie efektivity internetových materiálov. V prípade tutoriálov pre strojové učenie by bolo potrebné dbať na niekoľko metrík:

2.1 Metriky na meranie efektivity tutoriálov

- **použitie relevantného obsahu** pre pochopenie hlavného konceptu, pomocou ktorého bude používateľ schopný implementovať algoritmus podľa svojho problému.
- **presnosť, zrozumiteľnosť a jednoduchosť výkladu** aj prostredníctvom prezentačných pomôcok akými sú napríklad interaktívne vizualizácie.
- **Vhodné použitie štandardných nástrojov** známych na implementáciu strojového učenia pre príslušný problém. Pod nástrojmi môžeme chápať napríklad programovacie jazyky a knižnice².

² COPELAND, Michael. How to Use Training Metrics to Measure eLearning Effectiveness. *Efrontlearning* [online]. 2017 [cit. 2018]. Dostupné z: <https://www.efrontlearning.com/blog/2017/11/how-use-training-metrics-measure-effectiveness.html>

Metriky nám však samotne nemusia byť k úžitku pokial' ich nemáme ako merat'. Na sledovanie týchto metrík je preto potreba zaviesť aj niekoľko testovacích metód. Nasledovné testovacie metódy sa vzťahujú viac na kurzy ako na tutoriály, no dajú sa aplikovať aj pri tutoriáloch pokial' ich rozsah je dostačne veľký.

- **čas strávený pri úlohách** - vyskytujú sa pri úlohach problémy pri ktorých sú používateelia zdržaný neadekvátne dlho? Je úloha príliš ťažká a vyžaduje si väčšiu pozornosť a podporu?
- **čas absolvovania tutoriálu** - vyžaduje tutoriál príliš dlhý čas na jeho dokončenie? Je celkovo príliš obtiažny či ľahký?
- **miera úspešnosti tutoriálu** - koľko používateľov dokáže dokončiť tutoriál do konca?
- **miera úspešnosti úloh** - ktoré úlohy niesú používateľmi dokončené a prečo?
- **miesta odchodov používateľov** - v ktorých častiach tutoriálu odchádza väčšina používateľov?³

2.2 Nástroje pre podporu vzdelávania

V strojovom učení sa vyvíjajú stále nové netriviálne metódy pri ktorých nieje vždy intuitívne jasné ako pri nich fungujú použité algoritmy vo vnútri. Preto sa na zjednodušenie takýchto typov problémov manipulujúce s veľkým množstvom dát používajú vizualizačné techniky. Existujú rôzne konfigurácie či parametre algoritmov spolu aj s rôznymi dátami a nechat' možnosť s nimi experimentovať používateľa pri chápaní klúčových konceptov cez interaktívne vizualizácie môže zvýšiť efektivitu a pozitívnu skúsenosť s tutoriálom. Mnoho tutoriálov neponúka takúto možnosť zrejme kvôli tomu, že to vyžaduje množstvo práce s nástrojmi, ktoré niesú špecificky vytvárané na tento problém. Časom však strojové učenie nachádzalo stále väčšiu dôležitosť aj v schopnostiach vysvetlenia, preto začali vznikať knižnice a frameworky špeciálne zamerané na túto problematiku.

³ VENTURI, David. Every single Machine Learning course on the internet, ranked by your reviews. *Freecodecamp*[online]. 2017 [cit. 2018]. Dostupné z: <https://medium.freecodecamp.org/every-single-machine-learning-course-on-the-internet-ranked-by-your-reviews-3c4a7b8026c0>

2.2.1 Jupyter Notebook

Jednou z významných nástrojov pre podporu vzdelávania je Jupyter Notebook. Umožňuje integrovať vizualizácie, spustiteľný kód, obrázky, videá, text či matematické výrazy spolu do jedného dokumentu. Tento dokument slúži aj ako editovateľný program, ktorý je spustiteľný vo webovom prehliadači. Používajú sa na vytváranie či zdieľanie technického obsahu širokému publiku napr. cez blogové príspevky či odborné publikácie. Prevažne sa používajú pri vykonávaní analýzy dát spolu aj s popísanými výsledkami. Jupyter sa aj preto stal výborným prostriedkom pre výučbu v strojovom učení, pretože pracuje s veľkým množstvom dát ktoré je potrebné dobre poznať aj cez vizualizácie. Pri vzdelávaní však viniká aj pri iných príbuzných oblastí⁴.

Projekt Jupyter ponechal veľký vplyv pre rýchle testovanie, prototypovanie a prezentovanie myšlienok verejným komunitám. Jedná sa o neziskový open-source projekt stvárnený z projektu IPython v roku 2014, ktorý sa vyvíja s cieľom podporiť vedecké či edukatívne dokumenty pri ktorých je možné používať rôzne programovacie jazyky ako napr. R, Ruby, Go a Python, ktorý je jeden z najpoužívanejších programovacích jazkov v strojovom učení⁵.

Okrem interaktívnych rozhrani poskytujú aj statický pohľad na archivovaný dokument, v ktorom môžu byť výstupy spustených blokov kódov zrenderované. Takýmto spôsobom je možné zobraziť obsah bez nutnosti akýchkoľvek inštalácií na strane používateľov. Jupyter notebooky teda slúžia dvom účelom. Ponúka rozhranie pre spúšťanie kódov a slúži taktiež ako dobre formátovaný archív zobrazujúci aj výstupy týchto blokov kódu [2].

Jupyter notebooky sú obzväšť vhodnou voľbou pri prezentovaní konceptov strojového učenia. Učenie sa a experimentovanie s komplexnými dátami môže byť viac záživné a efektívnejšie ak s nimi môžeme interagovať cez nastavovanie určitých parametrov algoritmu cez kód a vidieť ich vplyv vo vizualizovanom obrázku. Väčšina dokumentov v Jupyteri je zrenderovaná statickým spôsobom. Existujú však aj elementy zvané ipywidjety, prostredníctvom ktorých je možné vizualizovať dátu aj dynamickým spôsobom v reálnom čase. Používateľom sa

⁴ PÉREZ, Fernando a Brian GRANGER. The state of Jupyter: How Project Jupyter got here and where we are headed. *Oreilly* [online]. 2017 [cit. 2018]. Dostupné z: <https://www.oreilly.com/ideas/the-state-of-jupyter>

⁵ SARKAR, Tirthajyoti. A very simple demo of interactive controls on Jupyter notebook. *Towardsdatascience* [online]. 2017 [cit. 2018]. Dostupné z: <https://towardsdatascience.com/a-very-simple-demo-of-interactive-controls-on-jupyter-notebook-4429cf46aab>

umožní nastavenie vstupných parametrov cez grafické rozhranie, pri ktorých sa vizualizované výstupy ihneď prispôsobia vstupom⁶.

Jupyter notebook je úspešný prezentačný prostriedok poskytujúci informácie v rôznych formách pre používateľov. Vzhľadom na komplexnosť strojového učenia je vždy dobré upriamiť pozornosť na vizualizácie. Vynaliezavosť vo vytváraní grafov kľúčových na pochopenie zásadných konceptov je jeden z najefektívnejších spôsobov ako podávať nové informácie používateľom. Preto nám niekedy siahanie na vizualizačné pomôcky v jupyter notebooku nie vždy môžu stačiť najmä ak vytvárame komplikovanejší súbor interaktívnych grafov, ktoré by napr. mohli aj medzi sebou komunikovať. Preto jedna z najznámejších firiem zaoberajúcich sa vizualizáciami dát Plotly vydala nový framework zvaný Dash.

2.2.2 Framework Dash

Používa sa pri vytváraní analytických webových aplikácií určených na vizualizáciu dát s rozsiahlymi možnosťami interaktivity vo webovom prehliadači len prostredníctvom Pythonu. Nie je teda potrebné zasahovať do aplikácie s javascriptom, CSS alebo HTML. Dash má tieto základné komponenty v sebe zabudované a abstrahuje množstvo funkcionálit z kničníc akými sú napr. plotly.js, react či flask. Hlavná výhoda Dashu preto spočíva v zladení všetkých týchto technológií a protokolov do jedného tak, aby jednotlivé funkcionality vedeli medzi sebou komunikovať. Táto abstrakcia prináša aj celkovú jednoduchosť pri implementovaní zložitejších dynamických aplikácií, na ktoré by inak štandardne bolo potrebné vytvoriť tím backendu, frontendu, data-science špecialistov či inžinierov. Tento framework je čerstvo vydaný v lete roku 2017, stále sa vyvíja a vylepšuje.

Potenciál E-learningu boduje aj pri vysvetlivujúcej sile. Poskytuje predovšetkým slobodu miesta, času a zastavovania sa podľa potreby myšlienkových pochodov v procese učenia. Jednou z ďalších nápadov na zvýšenie efektivity by mohol byť aj prostredníctvom spracovania prirodzeného jazyka. Pri riešení práce ma k výbere algoritmov viedla práve táto oblasť.

⁶ SARKAR, Tirthajyoti. A very simple demo of interactive controls on Jupyter notebook. *Towardsdatascience* [online]. 2017 [cit. 2018]. Dostupné z: <https://towardsdatascience.com/a-very-simple-demo-of-interactive-controls-on-jupyter-notebook-4429cf46aab>

3 Teória strojového učenia

Existuje množstvo definícií strojového učenia. Jednu z najpresnejších a najformálnejších priniesol v roku 1997 Tom Mitchell, ktorý hovorí, že “Počítačový program sa učí pomocou skúsenosti E s ohľadom na úlohu T a metriku výkonnosti P, pričom výkonnosť P sa v úlohe T zvyšuje prostredníctvom skúsenosti E., Ak by sme napríklad chceli predikovať výhru programu pri úlohe T, ktorý spočíva v hraní šachu, použijeme dátá resp. skúseností E prostredníctvom ktorých ak sa niečo naučil užitočné, zvýši svoju pravdepodobnosť výhry P v nasledujúcich hrách. [3]

Cieľom naučeného programu je teda schopnosť spracovávať neznáme udalosti či dátá vďaka získaným skúsenostiam. Uvedený príklad hry šachu je len jeden z mnoha rôznych typov problémov strojového učenia. Z takejto abstraktnej definície v reálnom svete nie je možné aplikovať jeden univerzálny algoritmus učenia ktorý by vedel pokryť väčšinu problémov, preto sa metódy strojového učenia delia do niekoľkých prístupov podľa typu učenia.

3.1 Metódy strojového učenia

Medzi najrozličnejšie a najpoužívanejšie prístupy patrí učenie s učiteľom (supervised learning) a učenie bez učiteľa (unsupervised learning)⁷.

- Učenie s učiteľom – program je “natrénovaný” vopred definovanou trénovacou množinou pozostávajúca zo vstupov a ich požadovanými výstupmi. Program sa naučí charakteristiky jednotlivých výstupov zo vstupov a na ich základe vygeneruje pravidlo ktoré použije pri odpovedi na nové vstupy dát s neznámym výstupom. Učenie s učiteľom sa delí podľa predikujúcich hodnôt na dva rôzne typy:
 - Klasifikácia – klasifikuje diskrétny (kategórické) hodnoty
 - Regresia – predikuje spojité (číselné) hodnoty
- Učenie bez učiteľa – programu je poskytnutý dataset bez označenia výstupných hodnôt, pri ktorom má za úlohu hľadať vzťahy a štruktúru dát na základe vstupov

⁷ NG, Andrew. Machine Learning. *Coursera* [online]. 2012, 2012 [cit. 2018]. Dostupné z: <https://www.coursera.org/learn/machine-learning>

- Zhlukovanie (clustering) – metóda hľadania súvislostí medzi vstupnými dátami tak, aby ich bolo možné podľa určitých rozličných vlastností roztriediť do skupín (zhlukov)
- Redukcia dimenzionality – metóda na redukciu počtu zvažovaných atribútov vo vstupe pri vytváraní predikčných modelov

Každý z týchto typov problémov je možné riešiť rôznymi algoritmami. Existuje ich veľké množstvo a odpoveď na otázku ktoré z nich použiť závisí od množstva faktorov ako napr.:

- Veľkosť, kvalita a štruktúra dát
- Dostupnosť výpočtového času
- Dôležitosť presnosti výsledkov
- Obtiažnosť interpretácie algoritmu

Často krát ani skúsený analytici nedokážu podľa typu problému povedať, ktorý algoritmus by bol najúspešnejší. Preto sa väčšinou z nich nejaké výberú a ich výsledky sa porovnávajú. Data science experti a vývojári na základe štúdií zostavili prehľad o vyberaní niekoľko používaných algoritmov podľa rôznych špecifických problémov. Nasledujúci ďahák zobrazuje spomenuté prístupy strojového učenia spolu s bežnými otázkami kedy sú aké algoritmy zhruba vhodné k použitiu. Na nájdenie najlepšieho algoritmu je však treba zohľadniť množstvo iných už spomínaných faktorov⁸.

⁸ LI, Hui. Which machine learning algorithm should I use?. *Sas* [online]. North Carolina, USA, 2017 [cit. 2018]. Dostupné z: <https://blogs.sas.com/content/subconsciousmusings/2017/04/12/machine-learning-algorithm-use/>



Obr. 3.1: Tahák algoritmov strojového učenia⁹

3.2 Klasifikačný model v strojovom učení

Po vybraní klasifikačného algoritmu spolu s jeho nastavením a natrénovaním trénovacou množinou sa vygeneruje klasifikačný model. Vytváranie celého optimálneho modelu pozostáva z piatich krokov¹⁰:

- Zhromažďovanie dát
- Predspracovanie dát
- Exploratívna analýza
- Budovanie modelu
- Evaluácia modelu

⁹ LI, Hui. Which machine learning algorithm should I use?. *Sas* [online]. North Carolina, USA, 2017 [cit. 2018]. Dostupné z: <https://blogs.sas.com/content/subconsciousmusings/2017/04/12/machine-learning-algorithm-use/>

¹⁰ MAYO, Matthew. A General Approach to Preprocessing Text Data. *Kdnuggets* [online]. 1997, 2017 [cit. 2018]. Dostupné z: <https://www.kdnuggets.com/2017/12/general-approach-preprocessing-text-data.html>

3.2.1 Zhromažďovanie dát

Pri vytváraní klasifikačného modelu nám k úspešnosti pomôže čo najväčší počet trénovacích dát, preto je potrebné často krát získať dátu aj z viacerých zdrojov.

3.2.2 Predspracovanie dát

Získané dátu len veľmi zriedkavo prichádzajú v takom správnom tvere, ktoré by vedel algoritmus spracovať optimálnym spôsobom, najmä vtedy keď pochádzajú z rôznych zdrojov. Preto ich potrebujeme upraviť vo fáze predspracovania dát, ktorý pozostáva z nasledovných krokov:

- Formátovanie dát
- Extrahovanie atribútov z dát
- Čistenie dát (vyplňanie chýbajúcich či vychýlených hodnôt v atribútoch) [1]

Kedže pri textovej klasifikácii vychádzame čisto len z textu, spôsob extrahovania atribútov je výrazne odlišný oproti ostatným štruktúrovaným dátam. Nevidíme žiadne priamo merateľné hodnoty, pri ktorých by sme vedeli niečo určiť, preto si musíme atribúty vytvoriť samostatným spôsobom. V predspracovaní textu sa používa **tokenizácia**, ktorá spočíva v rozdelovaní dlhých textov do viet, fráz, symbolov alebo najčastejšie slov. Na ich základe je možné vytvárať atribúty, ktorými môžu byť napríklad počty výskytujúcich sa slov v dokumente. Po tokenizácii môžu nasledovať ďalšie metódy predspracovania známe ako normalizácie [4]:

- **Odstránenie stop slov** – slová vyskytujúce sa často vo vetách, zvyknú mať len syntaktický význam a neobsahujú žiadnu významnú informáciu (and, or, you, we, with)
- **Stemming** – process odstraňovania sufíxov, prefixov, afixov zo slov na získanie základného tvaru slova (working, worked, works → work)
- **Lemmatizácia** – podobné stemmingu, líši sa schopnosťou zachytiť kánonické formy slov (good, better, cool → good)
- **Iné** – zmena veľkosti znakov, odstránenie čísel alebo ich konvertovanie do textovej podoby, odstránenie interpunkcií či medzier (často býva súčasťou tokenizácie) [4]

3.2.3 Exploratívna analýza

Cieľom je poznať dátá a zrhnúť ich hlavné charakteristiky, ktoré sú najčastejšie vyjadrované vizuálnymi alebo štatistickými metódami. Jedná sa o distribúciu dát, hľadanie anomálii a podobne. Hľadajú sa taktiež skryté vzťahy alebo korelácie medzi atribútmi. Exploratívna analýza sa zvykne používať aj pred predspracovávaním dát. Robia sa popri nej nasledujúce úkony:

- Odstránenie prebytočných atribútov (korelujúce a neobsahujúce žiadny informačný zisk)
- Normalizácia atribútov (úprava rozsahu hodnôt v atribútoch dát) [5]

3.2.4 Budovanie modelu

Ďalším krokom je vybrať algoritmus, ktorý sa použije na vykonanie požadovanej úlohy. Ako už bolo spomenuté, existuje veľké množstvo rôznych algoritmov ktoré riešia rôzne typy problémových úloh. Žiadny algoritmus nie je natoľko univerzálny, aby sa oňom dalo povedať, že je nadradený nad niektorými ostatnými algoritmami. Existujú však intuicie kedy sa aké algoritmy hodia, treba však vždy zistiť ktorý funguje najlepšie na konkrétny problém. Preto sa často v praxi algoritmy porovnávajú a zistí, ktorý má najlepšie požadované výsledky. Na to, aby sme mohli modely porovnávať, je potreba si zadefinovať metriku na meranie výkonnosti. Budovanie modelu preto práve vychádza aj z nasledujúceho kroku, evaluácie. [5]

3.2.5 Evaluácia modelu

Pri každom modeli je nutné zistiť, ako dobre bude predvídať budúce dátá. Kedže nepoznáme ich cieľové hodnoty, potrebujeme pomocou určitých metrík ohodnotiť model na dátach, pri ktorých cieľové hodnoty poznáme.

Rozdelenie trénovacej a testovacej množiny

Pri vytváraní klasifikáčného modelu potrebujeme predstavu, ako bude úspešný na dátach, ktoré nepozná, preto je vždy potrebné rozdeliť dataset na trénovaciu a testovaciu množinu. Klúčovou informáciou je, že nikdy nesmieme trénovať a testovať na tých istých dátach, pretože nám tak hrozí pretrénovanie. To nastáva práve vtedy, keď testovacie a trénovacie dátá majú veľký rozdiel vo výsledkoch úspešnosti. Model sa len akosi naučil dátu naspamäť, ale nenaučil sa správne pravidlá o nich vo všeobecnosti. Pri malom počte dát však toto rozdelenie môže spôsobiť aj to, že sa zahodí

zrovna tá unikátna časť, ktorá by mohla výrazne zlepšiť klasifikačný model, pretože sme nemali žiadne také podobné vzorky v trénovacej množine. V takýchto prípadoch sa často používa krížová validácia [6].

Krížová validácia (cross-validation)

Pointou je, že všetky dátá sa použijú na trénovanie aj testovanie, nie však naraz. Definuje sa premenná k koľko krát sa bude krížová validácia vykonávať a dátá budú rozdelené na k rôznych množín. Následne sa vždy zoberie jedna množina ktorá sa použije na testovanie a všetky ostatné sa použijú na trénovanie. Špeciálny prípad k-násobnej krížovej validácie je **leave-one-out**, kde sa používa len jedna vzorka na testovanie a všetky ostatné na trénovanie. Výsledok je tak možné potom merat' cez zpriemerovanie. Veľkou limitáciou tohto prístupu je čas, pretože k sa rovná počtu dát, preto sa najčastejšie používa nejaké zvolené k, pričom číslo k sa najčastejšie používa s číslom päť alebo desať. Táto cross-validácia sa používa len nato, aby sa zistilo, ktorý nastavený model je najlepší. Síce nebude úplne presná predstava ako sa bude správať v produkcií lebo nemáme úspešnosť nad vzorkami ktoré sme predtým ešte nikdy nevideli, máme však aspoň nejaký odhad, ktorý v mnohých prípadoch často krát pomáha. Najideálnejšie použitie krížovej validácie je preto vtedy, keď sa dataset rozdelí do troch rôznych datasetov: trénovacia, cross-validačná a testovacia. Pri testovacej dostávame reálnu predstavu ako úspešný je model pri nikdy nevidených dátach [6].

Metriky vyhodnocovania

Opäť ich existuje množstvo a závisí od domény, ktorá sa vyberie a použije. Najčastejšie sa vychádza z matici chýb (confusion matrix), ktorý je vyjadrený v nasledujúcej tabuľke [5].

	Predikcia - Positive	Predikcia - Negative
Skutočnosť - Positive	True Positive	False Negative
Skutočnosť - Negative	False Positive	True Negative

Tabuľka 3.1: Matica chýb

Hovorí nám o tom, aké rôzne stavy môžu nastať pri klasifikácii. Ak máme skutočnosť, ktorá môže mať dva stavy (pozitívna alebo negatívna), a máme taktiež dva stavy pre klasifikáciu, či je nejaká trieda pozitívna alebo negatívna, dostaneme dokopy štyri rôzne stavy. Z týchto štyroch čísel je možné počítať rôzne metriky. V zásade sa najčastejšie používa **správnosť** (accuracy),

ktorá sa vypočítava súčtom čísel na diagonále predeleným súčtom všetkých čísel. Táto metrika však má svoje obmedzenie, a to práve pri triedach, ktoré sú početnostne nevyvážené. Ak by napríklad v datasete boli dve triedy a prvá z nich by tvorila 90% a druhá len 10%, môže sa nám stať to, že by aj triviálny klasifikátor tvrdiaci že všetky vzorky patria len do prvej triedy mal až 90%-tnú úspešnosť. Takáto metrika správnosti je preto veľmi nevhodná pri majoritných triedach, pretože sa nič nedozvieme o minoritných triedach. Preto sa používajú metriky presnosti (precision) a recall.

- **Precision** - Pravdepodobnosť, že klasifikovaná vzorka do príslušnej triedy bola správne klasifikovaná (snaží sa kvantifikovať koľko klasifikácií príslušnej triedy bolo trafených správne)
- **Recall** – Pravdepodobnosť, že vzorka príslušnej triedy bola správne klasifikovaná (snaží sa povedať koľko zo všetkých pozorovaní s príslušnou triedou sa dokázalo nájsť)

Tieto dve metriky fungujú proti sebe a každá sa snaží ohodnotiť nejakú vec, ktorá je proti tomu druhému. Príkladom výberu metriky z týchto dvoch môže byť napríklad pri chorobách, kde väčšinou chorých ľudí je menší počet ako tých zdravých. Zaujímajú nás preto viac tí chorí a v takomto prípade by sme pridávali váhu metriky recall. Ak chceme optimalizovať výsledky týchto metrik, väčšinou sa stane že pri jednej sa zhorší tá druhá. Ak by sme chceli zgeneralizovať výsledok pre obe naraz, používa sa metrika **F1**, ktorá je počítaná harmonickým priemerom metrik precision a recall [5].

F1-score

$$\frac{2 * (Precision * Recall)}{Precision + Recall}$$

Pri týchto metrikách nie je možné počítať výsledok do viacerých možných tried len jedným číslom narozením od metriky správnosti pretože sa vypočítava pre každú triedu zvlášť. Na to, aby sme vedeli vyhodnocovať rôzne modely, je vhodné výsledky porovnávať len jedným číslom. Vtedy sa používa spriemerovanie.

Všetky spomínané metriky pre každú triedu sa zosumarizúvajú v tzv. **klasifikačnom reporte**. Stĺpec Support vyjadruje počet klasifikovaných vzoriek.

Kategória	Precision	Recall	F1-score	Support
news-Graphics	0.91	0.98	0.94	61
news-Forsale	0.95	0.89	0.92	61
news-Baseball	0.97	0.95	0.96	62
Avg/total	0.94	0.94	0.94	184

Obr. 3.2: Klasifikačný report troch kategórií

Ďalšia metrika ktorá sa zvykne používať je **logloss**. Používa sa v prípadoch keď nám klasifikátor vie povedať pravdepodobnosť, ktorá vyjadruje ako si je svojou predikciou istý. Na základe toho sa proporcíálnym spôsobom prostredníctvom zlogaritmizovanej pravdepodobnosti penalizuje model. Pri tomto prístupe treba však mať na pamäti, že je potrebné v niektorých prípadoch skorigovať hodnoty s extrémne veľkou chybou, pretože by mali za následok smerovať k mínus nekonečnu.

Ladenie hyperparametrov

Veľa z algoritmov má množstvo rôznych nastaviteľných parametrov zvaných ako hyperparametre, prostredníctvom ktorých môžeme zlepšovať výsledky na konkrétnych datasetoch. Preto je potrebné nájsť správnu kombináciu týchto hyperparametrov tak, aby nám dávali čo najlepšie možné výsledky. Ladenie môžeme zrealizovať ručne alebo aj automatickým spôsobom, ktorý sa delí na dve skupiny:

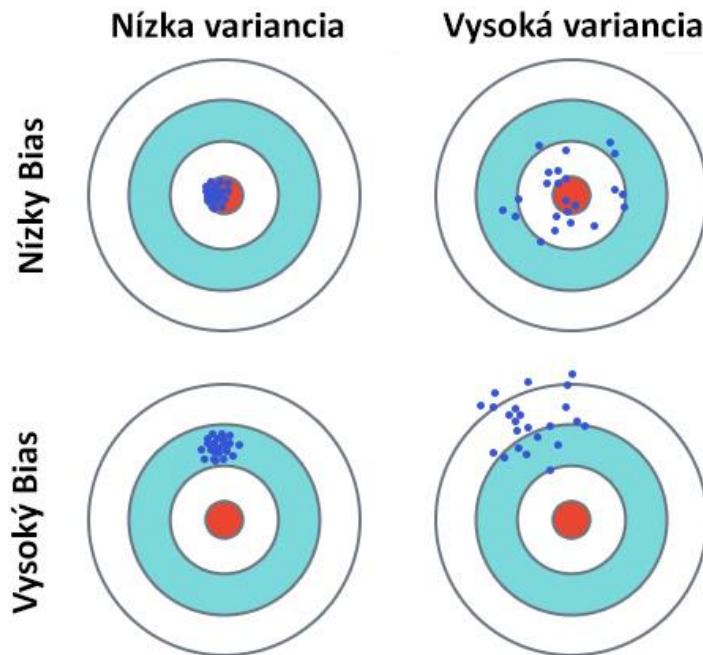
- **Informované hľadanie** – prístup hrubej sily tiež známy ako prehľadávanie podľa mriežky parametrov (grid search), kde sa skúšajú všetky rôzne kombinácie parametrov podľa zadaného rozsahu. Keď prostredníctvom neho nájdeme najúspešnejšie nastavenie hyperparametrov, je treba mať na pamäti, že výsledky pri nikdy nevidených dátach môžu byť odlišné a možno slabšie ako pri iných nastaveniach. Preto je pre tento problém preučenia vhodné si odložiť nejaké dátá a až úplne na konci sa pri nich pozrieť, ako reálne odhadujú správanie modelu.

- **Neinformované hľadanie** – prístup náhodného vyhľadávania pri ktorom sa skúšajú náhodne hodnoty hyperparametrov. Používa sa najmä pri diskrétnych hodnotách. Existujú modifikácie informovaného prehľadávania pri ktorom sa zvykne napr. používať aj simulované žíhanie [5].

3.3 Zložky chýb v predikčnom modeli

Existujú dva rôzne zložky chýb ktoré figurujú v predikčných modeloch.

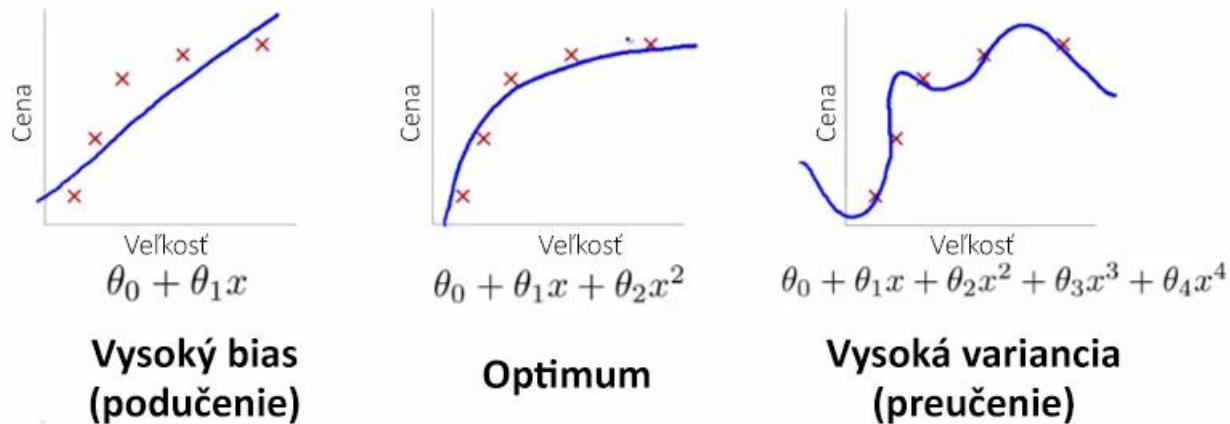
- **Bias** (výchylka), vyjadruje ako úspešne sa zohľadnili charakteristiky neznámych datasetov. Modelu chýba nejaká informácia a triafa predikujúce hodnoty mimo.
- **Variancia** (rozptyl) vyjadruje predikovaciu stabilnosť a premenlivosť modelu. Prostredníctvom nej sa indikuje, či sa pri vytváraní modelu ide správnym smerom k úspešným predikciám. Čím je variancia väčšia, tým sú predikované hodnoty viac rozptylené od reálnych hodnôt [6]



Obr. 3.3: Ilustrácia zložiek chýb bias a variancie¹¹

¹¹ MAYO, Matthew. Understanding the Bias-Variance Tradeoff: An Overview. *Kdnuggets* [online]. 2016 [cit. 2018]. Dostupné z: <https://www.kdnuggets.com/2016/08/bias-variance-tradeoff-overview.html>

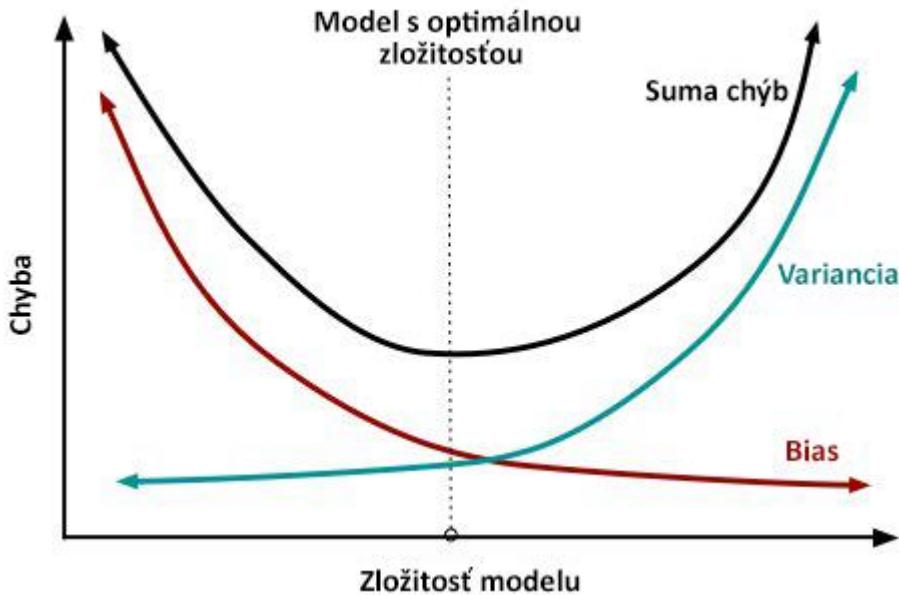
Varianciu je možné spozorovať najmä vtedy, keď úspešnosť predikovania hodnôt na trénovacej množine je výrazne lepší ako na testovacej množine. Tieto zložky chýb sú nezávislé a často krát ich obe nie je možné mať v perfektnom stave. Optimálne modely mávajú napríklad nejakú chybu na trénovacích dátach pri ktorých sa zachová bias. Ak sa potom použijú dátá ktoré predtým nikdy nevidel, tak je dostatočne jednoduchý a robustný na to, aby dával konzistentné predikcie. Snažíme sa teda nájsť vyvážený model, ktorý nie je príliš jednoduchý s vysokým bias a ani príliš komplikovaný s vysokou varianciou. Peknou ilustráciou je použitie lineárnej regresie s rôznymi stupňami polynómov, prostredníctvom ktorých sa snaží trafiť čo najviac dátových bodov.



Obr. 3.4: Ilustrácia podučenia, preučenia a optimálneho modelu¹²

Máme tu však problém v tom, že v modeli nevieme spočítavať aký má bias alebo varianciu. Vieme ale spočítavať ich sumy, prostredníctvom ktorých môžeme nachádzať nejaké optimum medzi nimi.

¹² NG, Andrew. Machine Learning. *Coursera* [online]. 2012, 2012 [cit. 2018]. Dostupné z: <https://www.coursera.org/learn/machine-learning>



Obr. 3.5: Ilustrácia počítania sumy chýb variancie a bias¹³

Existujú ďalšie spôsoby ako kontrolovať varianciu a bias prostredníctvom celej skupiny metód, ktorým sa hovorí učenie súborom metód. Základná myšlienka spočíva v tom, že pokiaľ máme viacero nezávislých modelov, tak ich skombinovaním vieme vyrobiť stabilnejší a lepší model. Týmto spôsobom sa zredukuje náhodná chyba ktorá môže vznikať len pri jednom modeli. Existuje niekoľko prístupov ktoré riešia tento problém:

- **Bagging** - výsledná predikcia vzniká hlasovaním medzi rôznymi modelmi. Výsledkom býva najčastejšie hlasovaná trieda. Nezávislosť je možné zabezpečiť používaním rôznych algoritmov s rôznymi trénovacími dátami. Rôznorodosť trénovacích dát môžeme riešiť napríklad aj vyberaním náhodných podmnožín. Taktiež sa môžu používať aj rôzne atribúty pri každom modeli.
- **Boosting** – taktiež kombinuje predikcie viacerých algoritmov. Proces boostingu však spočíva v natrénovaní algoritmu na určitých dátach, pri ktorých sa pozrie, na ktorých pozorovaniach mal nesprávnu predikciu a zvýši im váhu pre ďalší model, ktorý sa snaží tieto chyby napraviť. Pri tomto prístupe je treba dávať pozor na pretrénovanie.

¹³ MAYO, Matthew. Understanding the Bias-Variance Tradeoff: An Overview. *Kdnuggets* [online]. 2016 [cit. 2018]. Dostupné z: <https://www.kdnuggets.com/2016/08/bias-variance-tradeoff-overview.html>

- **Stacking** – má určitú podobnosť z boostingu aj baggingu. Je to model, ktorý používa predikčné výstupy z nezávislých čiastkových modelov. Ich kombináciou sa vytvára výsledný model, ktorý tieto čiastkové modely ováhuje podľa určitej funkcie [5].

Ďalšie možnosti kontroly variancie a biasu:

- Kontrola množstvom trénovacích dát. Čím ich je väčší počet, tým je väčšia šanca, že sa vygeneruje lepší model a zníži sa variancia aj bias. Možným riešením je taktiež krížová validácia. Treba však pamätať aj na to, že čím početnosť dát je väčšia, tým sa zmenšuje celkový vplyv na výsledný model a to najmä v prípade jednoduchých modelov.
- Pridávanie, odstraňovanie či vytváranie nových atribútov (**feature engineering**).
- Zmena optimalizačného kritéria pri chybovej funkcií regularizačným parametrom

4 Návrh riešenia

Cieľom práce je vytvoriť tutoriál Naivného Bayesovského klasifikátora. Kedže sa jedná o pomerne jednoduchý algoritmus, obsah tutoriálu bude obohacovaný základmi strojového učenia spolu s vysvetlením hlavnej intuície fungovania algoritmu podporných vektorov (Support Vector Machines alebo SVM).

V rámci tutoriálu sa použije interaktívne prostredie Jupyter, pri ktorom sa použije programovací jazyk Python. Tento vykonateľný dokument bude pozostávať z vysvetľujúcho textu, ktorý bude sprevádzaný aj matematickými vzorcami a tabuľkami prostredníctvom značkovacieho jazyka markdown. Popri výklade budú implementované nápomocné vizualizácie cez knižnicu matplotlib, ktorý v spolupráci s ipywidgetami bude umožňovať používateľom experimentovať a nastavovať množstvo vstupných parametrov, pri ktorých sa vizualizované výstupy grafov ihned prispôsobia vstupom. Najčastejšie používanými vstupmi budú hyperparametre algoritmu SVM, pri ktorých sa umožní sledovanie rôznych správaní sa klasifikačných modelov.

Najväčší dôraz práce sa kladie na implementáciu webovej aplikácie, ktorá bude realizovaná prostredníctvom frameworku Dash. Použijú sa pri nej rozsiahle interaktívne prvky, ktoré budú vizualizovať procesy klasifikácie textových dokumentov použitím algoritmu Naivného Bayesa. Bude primárne vytváraný ako súčasť tutoriálu v jupyter notebooku, vytvorí sa však aj príručka prostredníctvom ktorej sa vytvorí samostatný tutoriál pre algoritmus Naivného Bayesa. V tejto príručke sa opíše fungovanie aplikácie spolu so sprevádzaným textom vysvetlenia spomínaného algoritmu. Aplikácia bude obsahovať aj niekoľko návodov, ktoré budú stručne vysvetlovať ako jednotlivé grafické elementy fungujú.

5 Vybrané algoritmy pre textovú klasifikáciu

V dnešnej dobe dostupnosť informácií exponenciálne rastie najmä cez textovú formu a príkladom toho môžu byť elektronické knihy, učebné materiály, vedecké články, noviny, konverzácie na sociálnych sieťach či príspevky na rôznych portáloch. Využitie automatizovanej textovej klasifikácie nám v nich môže pomôcť rýchlym spôsobom analyzovať a vytvárať nové informácie. Do dôležitých súčasti textovej klasifikácie patrí analýza pocitov, zámeru a sentimentu. Týmito analýzami sa zaoberá oblasť spracovanie prirodzeného jazyka, ktorý má za úlohu prepojiť ľudskú komunikáciu s počítačovým porozumením. Slovenčina má v tejto sfére isté komplikácie keďže sa vyznačuje skloňovaním a inými komplikovanými vecami. Pri textoch napríklad v angličtine kde je naviac aj presnejší slovosled, sa skloňovanie či časovanie vôbec nepoužíva. Stáva sa preto vhodnejším jazykom na spomínané analýzy¹⁴.

Najbežnejšími úlohami zamerajúce sa na učenie s učiteľom v textovej klasifikácii je napríklad filtrovanie spamu či kategorizovanie emailov na základe ich obsahu. Špeciálnejším prípadom použitia by mohla byť aj identifikácia núdzovej situácie prostredníctvom analýzy miliónov online informácií, kde by sa vyžadovali metódy viacerých klasifikátorov pre dosiahnutie vysokej presnosti výslednej klasifikácie¹⁵.

Do často používaných algoritmov v textovej klasifikácii patrí Naivný Bayesovský klasifikátor. Je známy pre svoju rýchlosť ktorá sa využíva aj pri spracovaní prirodzeného jazyka a úspešnosti pri vysoko rozumných dátach. Algoritmus je jednoduchý na implementáciu a pochopenie aj pre ľudí, ktorí nemajú žiadnu predstavu o strojovom učení.

¹⁴ GUPTA, Shashank. Automated Text Classification Using Machine Learning. *Towardsdatascience* [online]. 2018 [cit. 2018]. Dostupné z:

<https://towardsdatascience.com/automated-text-classification-using-machine-learning-3df4f4f9570b>

¹⁵ GUPTA, Shashank. Automated Text Classification Using Machine Learning. *Towardsdatascience* [online]. 2018 [cit. 2018]. Dostupné z:

<https://towardsdatascience.com/automated-text-classification-using-machine-learning-3df4f4f9570b>

5.1 Naivný Bayesovský klasifikátor

Naivný Bayesovský klasifikátor je založený na Bayesovej vete, ktorá vychádza z podmienených pravdepodobností [1]. Táto veta určuje **posteriórnú pravdepodobnosť** platnosti hypotézy B za predpokladu, že hypotéza A je platná. Celý tento vzťah je možné zapísť nasledovne:

$$P(B|A) = \frac{P(A|B) P(B)}{P(A)}$$

Hypotéza B predstavuje kategórie ktoré sa predpovedajú a v hypotéze A vstupujú atribúty pozorovaní. Pre neznámu vzorku x sa vypočítajú posteriórne pravdepodobnosti pre každú z K možných kategórií:

$$P(B = k|x) = \frac{P(x|B = k) P(B = k)}{P(x)}$$

Z týchto všetkých vypočítaných posteriórnych pravdepodobností sa vyberie na klasifikáciu tá kategória, ktorá má najväčšiu hodnotu. Kedže v Bayesovom pravidle je menovateľ $P(x)$ (**marginálna pravdepodobnosť pozorovania**) vždy rovnaký pre každú kategóriu, nie je potrebné ho vypočítavať [1]. Najväčšiu časť posteriórnej pravdepodobnosti tvorí člen vierohodnosti kategórie (**likelihood**):

$$P(x|B = k)$$

Atribúty pozorovaní vzorky x podmienené kategóriou k sú v kontexte textovej klasifikácie vyjadrené modelom bag-of-words, ktorý spočíva v spočítavaní frekvencií každého slova v trénovacích vzorkách pre kategóriu do ktorej spadajú, atribúty sú teda slová a ich hodnoty sú počty výskytov v kategórii. To bude mať za následok zneusporiadanie slov v teste a naivný predpoklad nezávislosti atribútov. Majme napríklad päť trénovacích vzoriek s dvoma rôznymi ohodnoteniami (kategóriami) v nasledujúcej tabuľke.

Trénovacie vzorky	Kategória
excellent headset	Good
mic doesn't work	Bad
love this headset	Good
bad quality mic	Bad
great mic	Good

Tabuľka 4.1: Trénovacie vzorky s ohodnoteným textom

Po natrénovaní týchto vzoriek dostaneme nasledujúci bag-of-words model

Kategórie	excellent	headset	mic	doesn't	work	love	this	bad	quality	great
Good	1	2	1	0	0	1	1	0	0	1
Bad	0	0	2	1	1	0	0	1	1	0

Tabuľka 4.1: Reprezentácia textu vzoriek pre každú kategóriu cez model bag-of-words

Priebeh výpočtu likelihoodu pozostáva z vynásobenia pravdepodobností medzi všetkými atribútmi pozorovaní, ktoré vzorka x obsahuje. V jednej neznámej vzorke sa teda bude iterovať cez všetky slová, ktoré sa budú navzájom vynásobovať ich vypočítanou pravdepodobnosťou. Pravdepodobnosti jednotlivých pozorovaní atribútov sú vypočítavané počtom výskytov príslušnej hodnoty (slova) atribútu v trénovacej množine vydelenú počtom výskytov vo všetkých atribútoch v trénovacej množine.

Celý proces učenia pozostáva v spočítaní týchto frekvencií vo všetkých trénovacích vzorkách pre ich určenú kategóriu, pričom v rámci algoritmu Naivného Bayesa sa taktiež pridáva atribút s počtami vzoriek pre každú kategóriu, ktorý slúži pre výpočet apriórnej (**prior**) pravdepodobnosti $P(B)$. Tá je vyjadrená počtom trénovacích vzoriek v kategórii k vydelenú celkovým počtom trénovacích vzoriek v trénovacej množine.

Výpočet pravdepodobnosti určitej kategórie sa teda vykonáva na základe početnosti výskytov hodnôt v atribútoch v trénovacej množine. Celý process klasifikácie sa dá vyjadriť nasledovnou výpočtovou formulou.

$$\hat{y} = \underset{k \in \{1, \dots, K\}}{\operatorname{argmax}} P(B_k) \prod_{i=1}^n P(x_i | B_k)$$

V člene vierohodnosti kategórie môže dôjsť k prípadu atribútu s nulovou početnosťou a spôsobil by celkovú nulovú pravdepodobnosť. Preto je potrebné buď zadefinovať veľmi malé racionálne číslo na nahradenie, alebo pridať jeden výskyt pre všetky existujúce atribúty – **Laplaceovo vyhľadzovanie** [1].

Táto metóda klasifikácie je však použiteľná len pre diskrétné dátu. Existuje preto varianta zvaná ako **Gaussový Naivný Bayes**, ktorý sa vie vysporiadáť aj so spojitými hodnotami v atribútoch prostredníctvom normálnej distribúcie.

Naivný Bayes má napriek svojim silným predpokladom nezávislosti atribútov veľké využitie v mnohých situáciách v reálnom svete, používajú sa najmä pri klasifikácii dokumentov a filtrovaní spamu. Väčšinou nemávajú takú úspešnosť ako niektoré iné komplexnejšie algoritmy, majú však oproti nim množstvo iných výhod.

Kedy má Naivný Bayesovský klasifikátor dobré výsledky?

- keď naivný predpoklad závislosti atribútov zodpovedá poskytnutým dátam (zriedkavý prípad v praxi a stáva sa jeho hlavnou nevýhodou)
- pri dobre separovaných kategóriach, keď zložitosť modelu je menej dôležitá
- pri extrémne veľkom počte atribútov, keď zložitosť modelu je menej dôležitá

Posledné dva body sa zdajú byť odlišné, no v skutočnosti so sebou súvisia. Ako dimenzionalita datasetu raste, klesá pravdepodobnosť, že akékoľvek dva body v priestore budú blízko seba, pretože musia byť blízko v každej jednej dimenzií na to, aby boli blízko celkovo. To znamená, že dátu vo veľkých dimenziách majú tendenciu byť v priemere viac separované ako dátu v nižších dimenziách za predpokladu, že nové dimenzie skutočne pridávajú nejakú informáciu. Z tohto dôvodu môže jednoduchý Naivný Bayesovský klasifikátor práve prostredníctvom naivného predpokladu fungovať veľmi dobre a v niektorých prípadoch aj lepšie ako iné oveľa komplikovanejšie klasifikátory. Pri dostatočnom počte poskytnutých vysoko dimenzionálnych dát môže byť tento algoritmus veľmi úspešný [7].

Výhody Naivného Bayesovské klasifikátora:

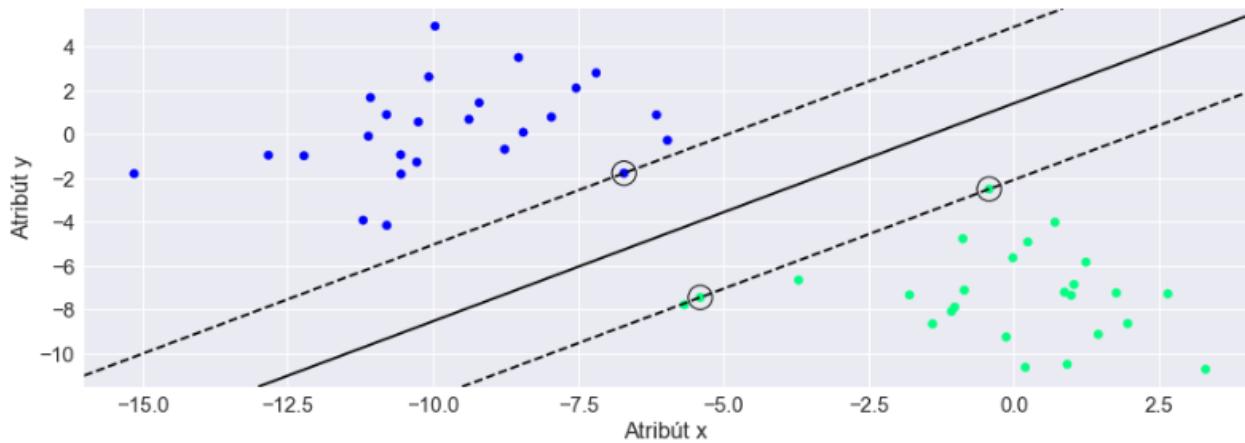
- je extrémne rýchly na trénovanie a klasifikovanie
- ľahko pochopiteľný a jednoduchý na implementáciu
- neobsahuje takmer žiadne nastaviteľné parametre
- vystačí si s malým množstvom trénovacích dát

Vychádzaním z týchto výhod je tento klasifikátor dobrou voľbou obzvlášť pri potrebe rýchleho inicializovania základného modelu klasifikácie

5.2 Algoritmus podporných vektorov

Jednou z ďalších úspešných algoritmov pre textovú klasifikáciu je algoritmus podporných vektorov, ktorý bol pred neurónovými sietami najpopulárnejší učiaci sa algoritmus pre jeho vysokú presnosť.

Algoritmus podporných vektorov slúži na vytvorenie diskriminatívneho modelu, ktorý sa snaží nájsť nadrovinu oddelujúcu dátu pre dve rôzne kategórie. Na jej základe sa potom rozhoduje, do ktorej kategórie bude neznáma vzorka klasifikovaná. Jeden z jednoduchších modelov je možné vizualizovať nasledujúcim obrázkom [7].



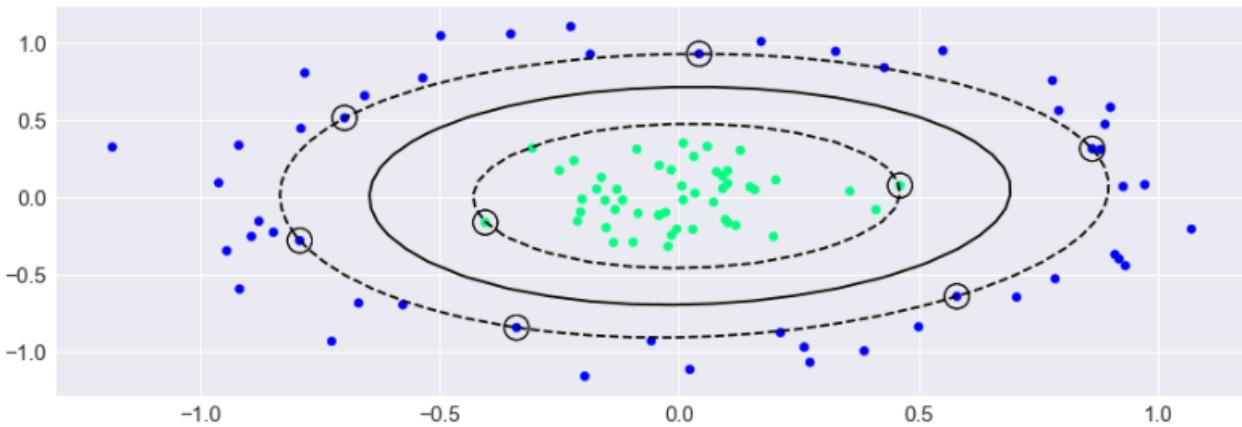
Obr. 5.1: Model SVM pri lineárne separovateľných dátach

V našom prípade dvojdimenzionálneho priestoru atribútov plná čiara predstavuje oddelovaciu nadrovinu (hyperplane), ktorá maximalizuje rozpätie (margin) medzi kategóriami prostredníctvom troch vyznačených dátových bodov známe ako podporné vektory (support vectors) tvoriace cestu. Tieto body sú kľúčové prvky k vytvoreniu klasifikačného modelu a je preto od nich odvodené meno samotného algoritmu [7].

Kľúčom k úspechu tohto klasifikátora je skutočnosť, že je založený len na umiestnení týchto podporných vektorov, akékoľvek iné body nachádzajúce sa ďaleko od okrajov vôbec nemenia výsledný tvar klasifikačného modelu. Problém však nastáva pri ďažšom rozpoložení dát s malým rozpäťím medzi kategóriami ktorý sice je maximalizovaný a oddeluje trénovacie dátá perfektným spôsobom, ale často krát to neplatí pri predikciách u neznámych vzorkách, pretože neberie do úvahy frekvencie dát na určitých miestach nachádzajúcich sa za okrajmi cesty. Model sa príliš snaží správne kategorizovať trénovacie dátá, stráca tým však generalizáciu riešenia u

neznámych dát a trpí preučením. Preto v takýchto prípadoch je potrebné nastaviť regularizačný parameter pre zgeneralizovanie chyby pre testovacie dát. Táto tolerancia určitej chyby v trénovaní bude mať za následok vytvorenia cesty s väčším rozpätím, v ktorom bude zahrnuté väčšie množstvo podporných vektorov [7].

V našom prípade išlo o lineárnu separáciu dvojdimenzionálnych dát. Na to, aby sme vedeli oddeliť aj nelineárne rozpoloženie dát je potrebné použiť kernelovú funkciu transformujúcu dátu do viacozmerného priestoru tak, aby ich bolo možné oddeliť. Tento kernelový trik nám umožňuje rozdelovať dátu analogickým spôsobom tak, ako to bolo pri dvojrozmerných dátach. Nelineárne oddelenie je možné vizualizovať v dvojrozmernom priestore nasledovne:



Obr. 5.2: Model SVM pri nelineárne separovateľných dátach

Bežne sa používajú tieto tri kernelové funkcie: lineárna, polynomiálna a radiálna bázová funkcia (Radial Basis Function alebo RBF). V prípade použitia RBF či polynomiálneho kernelu existuje hyperparameter γ (gamma), prostredníctvom ktorého vieme nastaviť do akej miery chceme tvarovať komplexné tvary tak, aby zodpovedali trénovacím dátam. Čím je teda tento hyperparameter väčší, tým je model náchylnejší k preučeniu.

Algoritmus v tejto základnej podobe vykonáva binárnu klasifikáciu, pri ktorej vie klasifikovať len dve rôzne kategórie. Na to, aby sme mohli klasifikovať väčší počet kategórií, je treba problém transformovať na viacero binárnych klasifikačných podproblémov. Používajú sa tieto dva prístupy:

- **One-vs-all** - počas trénovania sa vytvorí klasifikátor pre každú kategóriu, pričom vzorky príslušných kategórií sú pozitívne a všetky ostatné sú negatívne. Pri klasifikácii sa vyberie ten klasifikátor, ktorý má najširšie okrajové rozdelenie.

- **One-vs-one** - počas trénovania sa vytvára klasifikátor pre každý možný pár kategórií. V čase predikcie sa používa schéma hlasovania a kategória s najvyšším počtom predpovedí sa použije ako výsledok. Výhoda tohto prístupu nad one-vs-all spočíva v lepšej vyváženosti výsledných rozdelení medzi kategóriami, nevýhodou je ale oveľa pomalšie trénovanie keďže obsahuje celkovo $K(K - 1)/2$ klasifikátorov¹⁶. V praxi sa väčšinou používa prvý prístup.

Výhody algoritmu SVM:

- veľká rýchlosť klasifikácie
- kernelový trik zabezpečuje univerzálnosť modelu pre rôznorodé dátu
- kompaktný model zaberajúci málo pamäti prostredníctvom závislosti od pomerne málo podporných vektorov
- narozdiel od väčšiny iných algoritmov fungujú veľmi dobre s vysoko rozmernými dátami ktoré môžu aj presahovať celkový počet vzoriek [7]

Nevýhody algoritmu SVM:

- pomalá rýchlosť trénovania pri veľkých datasetoch
- finálny model je ťažko interpretovateľný
- nájdenie správnych parametrov kernelu nie je jednoduché najmä pri veľkom počte dát

Vzhľadom na tieto charakteristiky sa odporúča SVM používať najmä vtedy, keď máme veľký počet dostupných dát a výsledky iných algoritmov nestačia. Viniká najmä keď máme dostupnú vysokú výpočtovú silu pre natrénovanie a nastavovanie hyperparametrov. Úspešný je aj pri problémoch textovej klasifikácie práve aj kvôli tomu, že dáta bývajú vo viacdimenzionálom priestore lineárne separovateľné [7].

¹⁶ SINGH, Sadanand. Understanding Support Vector Machine via Examples. *Sadanand-singh.github* [online]. 2017 [cit. 2018]. Dostupné z: <https://sadanand-singh.github.io/posts/svmpython/>

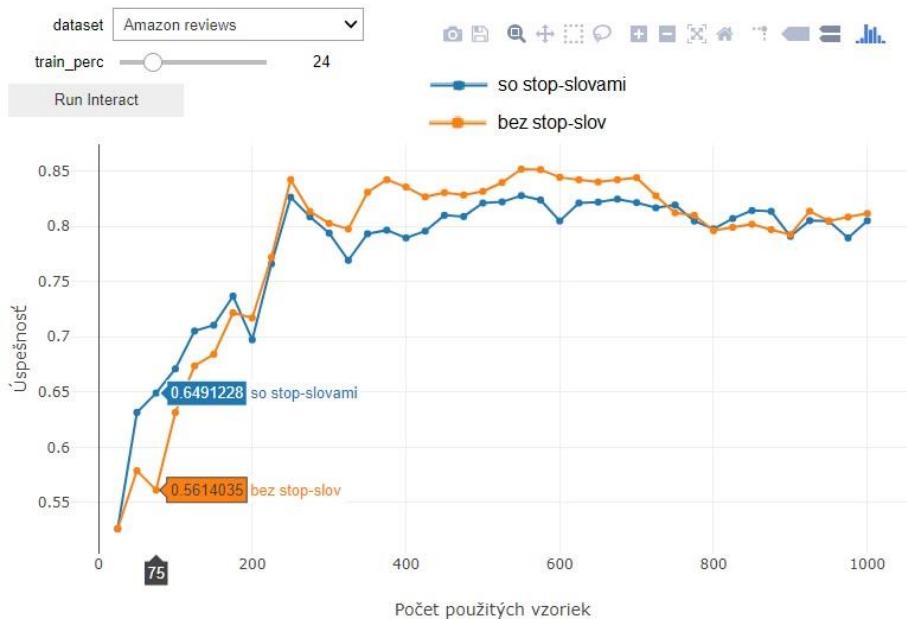
6 Použité nástroje na podporu vzdelávania

6.1 Tutoriál v Jupyter Notebooku

Vykonateľný tutoriál bol realizovaný prostredníctvom Jupyter notebooku, pri ktorom bol text podávaný aj pre ľudí, ktorí nemajú skúsenosť so strojovým učením. V priebehu vysvetlovania vybraných algoritmov sa preto spomínajú aj fundamentálne pojmy pre porozumenie základnej terminológie.

6.1.1 Naivný Bayesovský klasifikátor

Popri vysvetlovaní algoritmu Naivného Bayesa sa ozrejmila základná terminológia strojového učenia. Na konci boli implementované funkcie, prostredníctvom ktorých bolo možné spúšťať vizualizáciu úspešnosti klasifikácie na krátkych textových dátach, pri ktorých mohol používateľ zadať pomer trénovacích a testovacích dát, prípadne vybrať jeden z troch rôznych datasetov. Kedže sa jednalo o textovú klasifikáciu, vizualizovali sa rozdiely v úspešnosti s aplikovaním stop slov a bez nasledovným spôsobom.



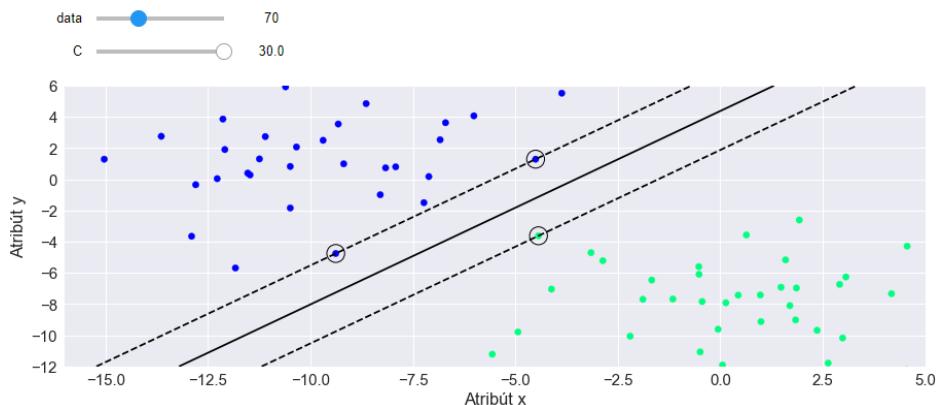
Obr. 7.1: Úspešnosť Naivného Bayesovského klasifikátora
vzhľadom na počet použitých vzoriek

Pre pochopenie vplyvu apriórnych pravdepodobností na výslednú pravdepodobnosť sa okrem krátkych textov použil aj dataset s dlhým textom. Vizualizácia tohto vplyvu je realizovaná vo webovej aplikácii dash, ktorá sa spomenie neskôr. V tejto časti tutoriálu bolo možné prostredníctvom dlhých textov poukázať na problém extrémne malých vypočítaných pravdepodobností. Aplikovaním funkcie logaritmus bolo možné vidieť zásadný rozdiel úspešnosti klasifikácie. Následne sa zosumarizovali výhody, nevýhody, kedy máva dobré výsledky a rôzne modifikácie tohto algoritmu.

6.1.2 Algoritmus podporných vektorov

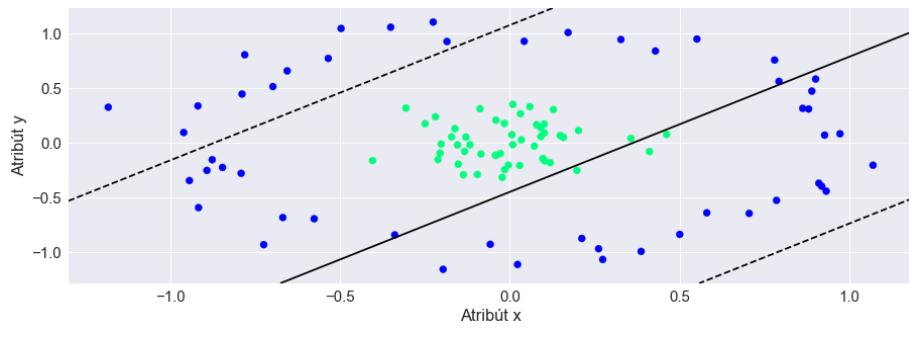
Na rozhodnutie o výbere druhého algoritmu vplýval kontext problému a typ klasifikačného modelu. V našom kontexte textovej klasifikácie má algoritmus SVM vysokú úspešnosť a je preto vhodný aj na porovnanie s Naivným Bayesovským klasifikátorom. Taktiež narozdiel od generatívneho modelu Naivného Bayesa sa dal intuitívnym spôsobom poukázať princíp diskriminatívneho modelu, ktorý naviac ponúka viac príležitostí vizualizácie.

Použili sa interaktívne vizualizácie, prostredníctvom ktorých bolo umožnené nastaviť počet trénovacích dát spolu aj s regularizačným parametrom C. Pri každom prenastavení vstupov sa výsledný vizualizačný graf okamžite mení.

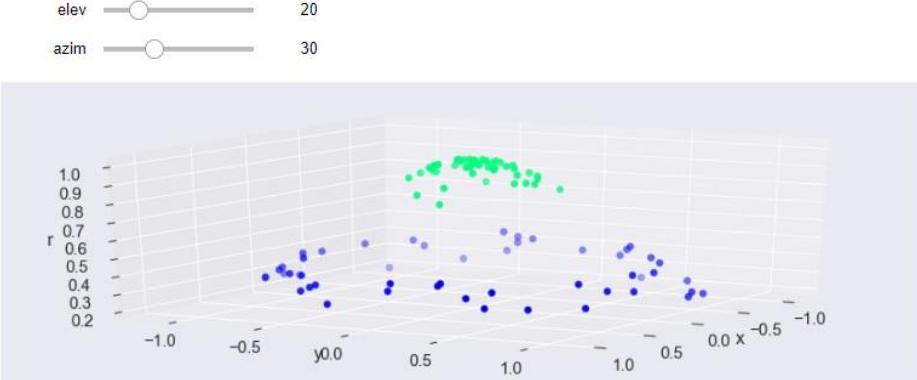


Obr. 7.2: Vizualizácia s nastaviteľným počtom trénovacích dát a hyperparametrom C

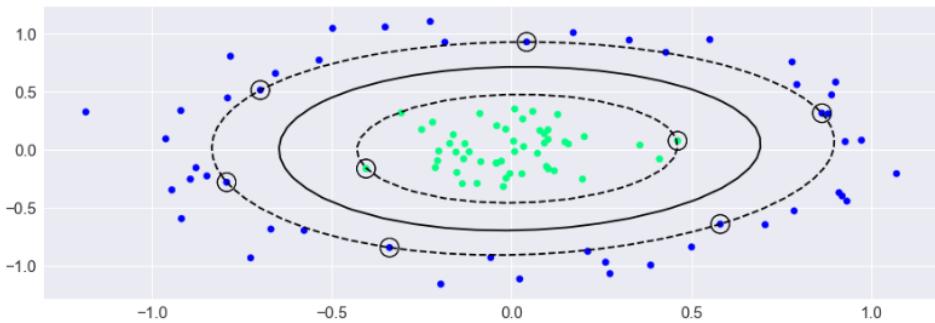
Nasledujúci prvý obrázok poukazuje na problém nelineárne separovateľných dát, druhý sa snaží poukázať na možné riešenie problému z editovateľného pohľadu trojrozmernej perspektívy, pri ktorom je možné dátá oddeliť plochou. Tretí obrázok zobrazuje výslednú nelineárnu separáciu dát použitím kernelu s radiálnej bázovou funkciou.



Obr. 7.3: Problém nelineárne separovateľných dát

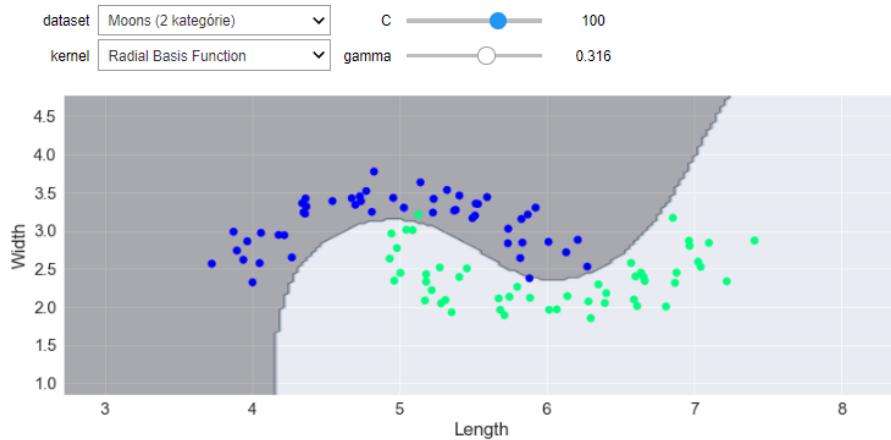


Obr. 7.4: Zobrazenie dát v trojrozmernom priestore



Obr. 7.5: Riešenie pomocou kernelu RBF

V tutoriály sa prostredníctvom takýchto typov obrázkov vysvetlila hlavná intuícia fungovania algoritmu SVM. Nasledovná interaktívna vizualizácia ponúka možnosť výberu troch rôznych datasetov, kernelov a hyperparametrov cez ktoré bolo možné pozorovať rôzne tvary klasifikačných modelov.



Obr. 7.6: Vizualizácia nastaviteľnosti rôznych konfigurácií SVM

6.2 Tutoriál s frameworkom Dash

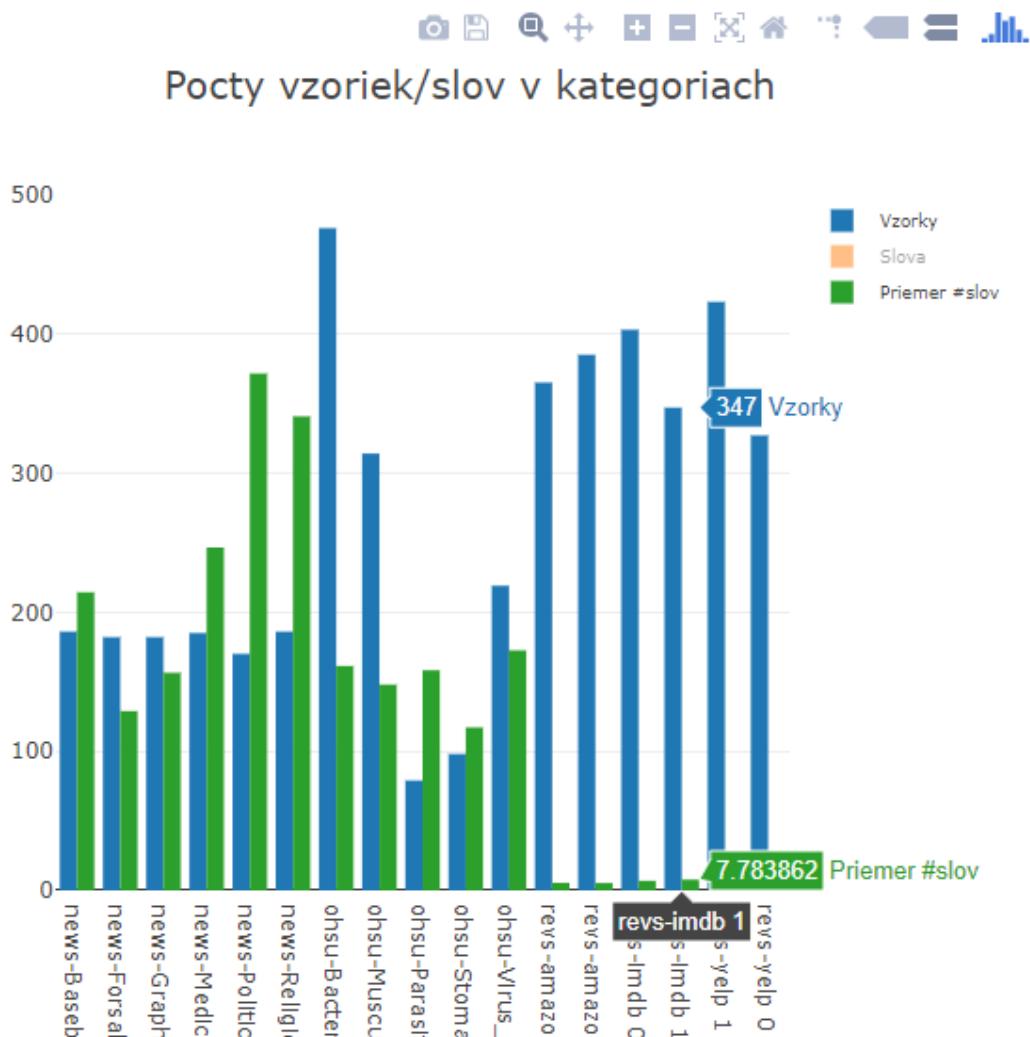
Webová aplikácia začína s textom, ktorý sumarizuje celý proces Naivného Bayesovského klasifikátora v kontexte textovej klasifikácie. Po tomto úvode nasleduje niekoľko sekcií:

6.2.1 Vlastnosti trénovacej množiny

Táto sekcia slúži na zosumarizovanie základných vlastností všetkých kategórií v trénovacej množine.

Na to, aby sme mohli sledovať kedy sa algoritmu darí a kedy nie, boli použité tri rôzne typy datasetov s rozličnými vlastnosťami a obtiažnosťami klasifikácie. Prvý z nich je 20news-18828 obsahujúci dlhšie texty s ľahko rozpoznávateľnými kategóriami akými sú napríklad grafika, politika a šport. Druhým je ohsumed s ľažko rozpoznávateľnými kategóriami s typmi chorôb, kde terminológia slov je podobná. Tretí obsahuje tri podčasti datasetov pozostávajúce z krátkych textov recenzií filmov, produktov a reštaurácií. Prostredníctvom poskytnutého grafu je pre každú kategóriu z trénovacej množiny zobrazený počet trénovacích vzoriek, pričom je možné zobraziť aj počet v nich vyskytujúcich sa slov a priemerný počet slov na jednu vzorku.

Vlastnosti trenovacej mnoziny



Obr. 7.7: Sekcia vlastností trénovacej množiny

6.2.2 Klasifikačné parametre a ich výsledky

Táto sekcia umožňuje vybrať a kombinovať kategórie z trénovacej množiny, z ktorej sa zoberú trénovacie vzorky na trénovanie a testovacie vzorky pre klasifikáciu. Klasifikátor teda nebude brať žiadne iné kategorické vzorky z trénovacej množiny. Pre vizualizácie spomínané v nasledujúcich sekciách je potrebné vybrať vždy len tri kategórie. Po vybraní kategórií sa nastaví riešenie nulových frekvencií slov prostredníctvom racionálneho čísla, ktorý sa neustále zobrazuje a aktualizuje cez vybranú hodnotu v exponencionálne rastúcom slideri. Na koniec spustíme klasifikáciu tlačidlom “Klasifikuj”, prostredníctvom ktorého sa nám na základe vybraných kategórií a parametrov zobrazí

klasifikačný report, počet testovacích a správne klasifikovaných vzoriek spolu aj s metrikou správnosti. Použité datasety sú rozdelené do pomeru 3:1 trénovacích a testovacích dát.

Klasifikacne parametre a ich vysledky

Pouzité datasety (news, ohsu, revs) sú rozdelene do **trenovacieho(75%)** a **testovacieho(25%)** datasetu.

Vyberte tri kategorie z ktorých sa vyberu trenovacie vzorky na trenovanie a testovacie vzorky pre klasifikaciu.

The screenshot shows a horizontal list of three categories: "news-Graphics", "news-Forsale", and "news-Baseball". Each category is preceded by a small "X" icon and followed by a blue rectangular button containing the category name. To the right of the list is a "close" button (X) and a dropdown arrow.

Moznosti riesenia nulovych frekvencii slov



Klasifikacny report [?]

Category	Precision	Recall	Average	F1-score	Support
news-Graphics	0.91	0.98	0.95	0.94	61
news-Forsale	0.95	0.89	0.92	0.92	61
news-Baseball	0.97	0.95	0.96	0.96	62
Avg/total	0.94	0.94	0.94	0.94	184

173 spravnych klasifikacii z celkovych 184 vzoriek

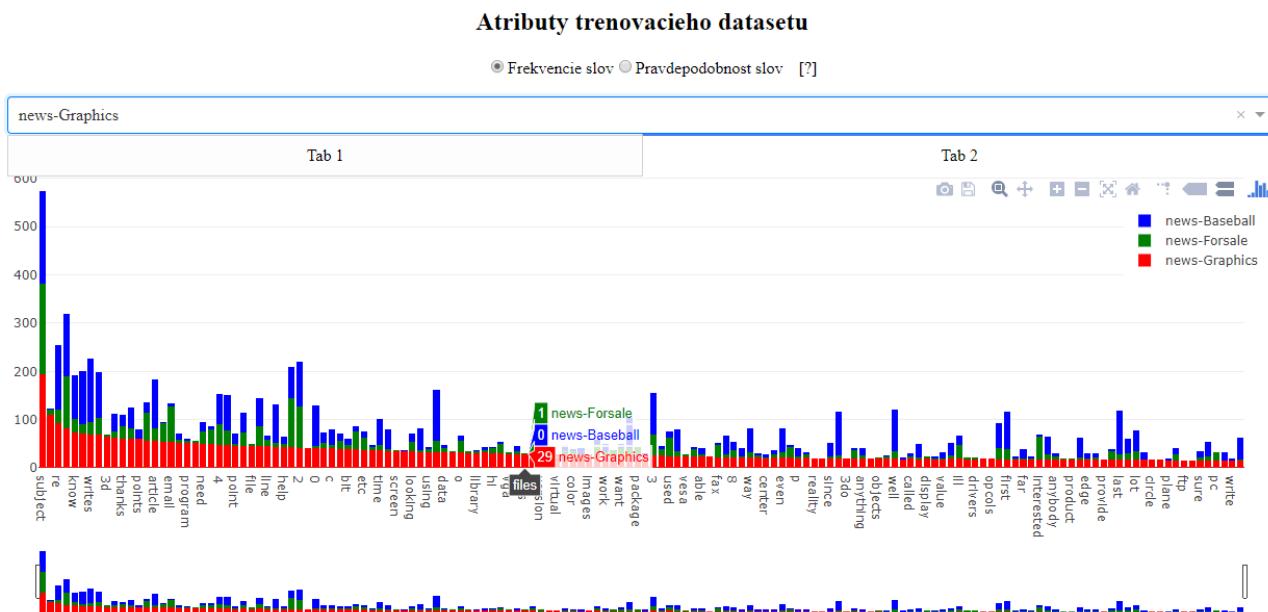
Uspesnosť: 0.94

Obr. 7.8: Sekcia klasifikačných parametrov a ich výsledkov

6.2.3 Atribúty trénovacieho datasetu

Táto sekcia zobrazuje najviac frekventované slová podľa vybranej kategórie cez dropdown.

Kedže ich existuje veľké množstvo, zobrazili sa z nich len 150 najfrekventovanejších s usporiadaním od najväčšieho po najmenší. Graf je možné zobraziť dvoma rôznymi spôsobmi cez taby. Prvý ukazuje ktoré kategórie prekonali početnosti slov vybranej kategórie tak, že presiahnu jej čiaru v príslušnom slove. Druhý tab ukazuje početnosti každej kategórie dokopy v jednom stĺpci. Kedže vypočítavanie pravdepodobností v Naivnom Bayesovskom klasifikátore nezáleží len nad počtom slov, vytvorila sa možnosť zobrazenia pravdepodobnostných hodnôt nad dropdownom.



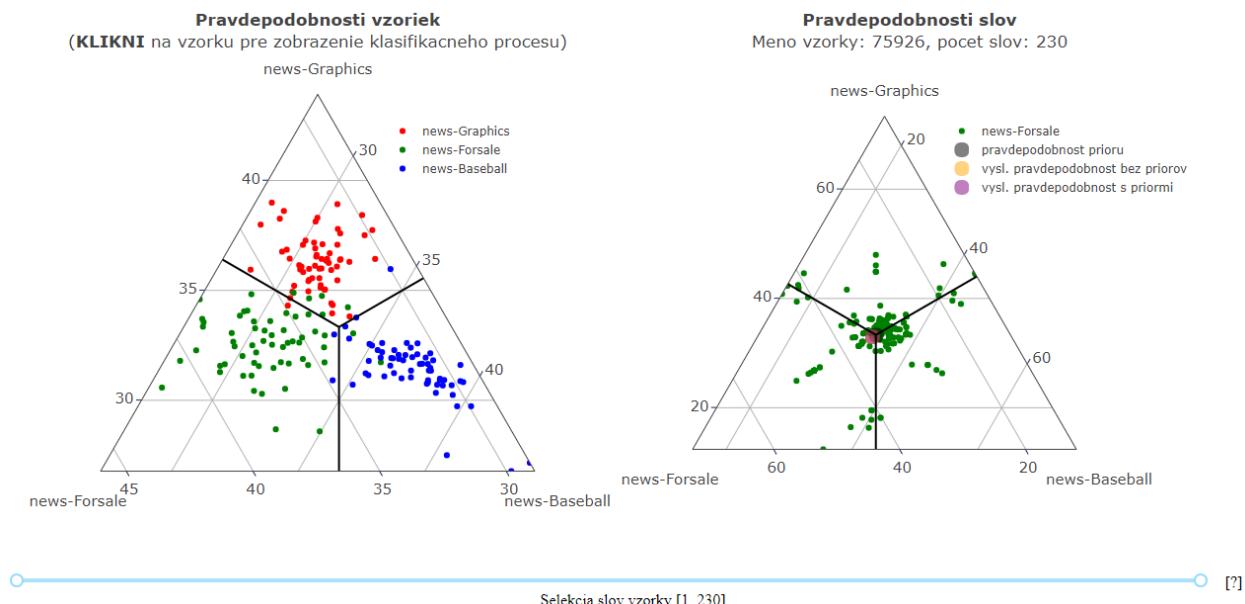
Obr. 7.9: Sekcia atribútov vybraného trénovacieho datasetu

6.2.4 Pravdepodobnosti vzoriek a slov

Táto sekcia zobrazuje prostredníctvom terárnych grafov výsledné pravdepodobnosti vzoriek spolu aj s pravdepodobnosťami jednotlivých slov vybranej vzorky.

Na to, aby sme mohli určiť interpretovateľné pravdepodobnosti pre každú kategóriu a získať tak aj s akou istotou boli všetky vzorky klasifikované, je potrebné ich najprv upraviť do zobrazovateľného tvaru pre ternárny graf. Pri vzorkách s dlhým textom je najprv potrebné aplikovať funkciu logaritmus pre každú vypočítanú pravdepodobnosť. Dostávame tým záporné hodnoty, ktoré potrebujeme pre každú kategóriu ideálne namapovať do percentálneho rozsahu. Na

to sa použila mapovacia funkcia $((2 * \text{avgPerc}) - i) * 33.33) / \text{avgPerc}$, kde avgPerc je priemer kategorických pravdepodobnostných hodnôt a i je zlogaritmizovaná pravdepodobnosť, ktorá sa ide mapovať. Výsledné percentuálne pravdepodobnosti medzi kategóriami sú však v častých prípadoch veľmi podobné a pri každom z nich sa hodnoty pohybujú v rozsahu 28% až 38% najmä vtedy, keď sa používajú vzorky s dlhším textom. Následkom je veľká hustota vzoriek na jednom mieste v terárnom grafe a množstvo nevyužitého percentuálneho priestoru. Preto sa tento problém vyrieší vyhľadaním najmenšej mapovanej hodnoty pre každú kategóriu. Tieto hodnoty sa použijú ako hranice, na základe ktorých sa vykoná priblíženie grafu, pri ktorom budeme môcť stále vidieť všetky dátové vzorky.



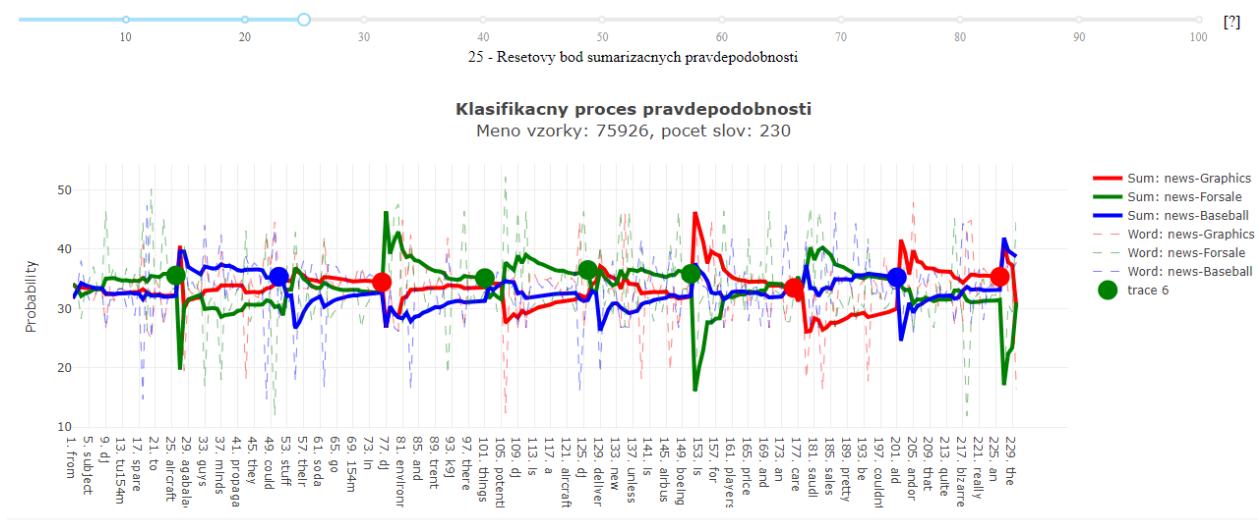
Obr. 7.10: Sekcia pravdepodobností vzoriek a slov

Na obrázku vľavo vidíme vzorky farbou vyjadrujúce ku ktorým kategóriám patria. Podobne ako vizualizujeme všetky klasifikované vzorky, môžeme vizualizovať všetky slová v jednej vzorke ktorá sa vyberie cez kliknutie na graf. Zobrazí sa nám napravo graf, cez ktorý dostaneme predstavu o každom slove aký mal konkrétny vplyv na výsledok cez miesto, na ktorom sa nachádza. Farba slov tentokrát vyjadruje klasifikovanú kategóriu vybranej vzorky. Okrem pravdepodobností slov sú zobrazené pravdepodobnosti prioru, výsledných pravdepodobností bez prioru a s priorom (ktorú je možné vidieť aj pri hoveru vzoriek v ľavom grafe). Tieto body v grafe nám umožnia sledovať vplyv priorov na výslednú klasifikáciu. Kedže neznáme vzorky v našom prípade pozostávajú

väčšinou z dlhých textov, vplyvy priorov nie sú dostatočne vidieť. Preto sa aj prostredníctvom slideru nižšie pridala možnosť vybrať rozsah použitých slov cez výberu počiatočného a posledného slova vo vzorke. Tento výber intuitívne mení aj výslednú klasifikáciu vyjadrenú cez farbu použitých slov.

6.2.5 Klasifikačný proces pravdepodobnosti

Táto sekcia umožňuje sledovať proces vypočítavaných pravdepodobností po každej jednej iterácii slova vo vzorke.



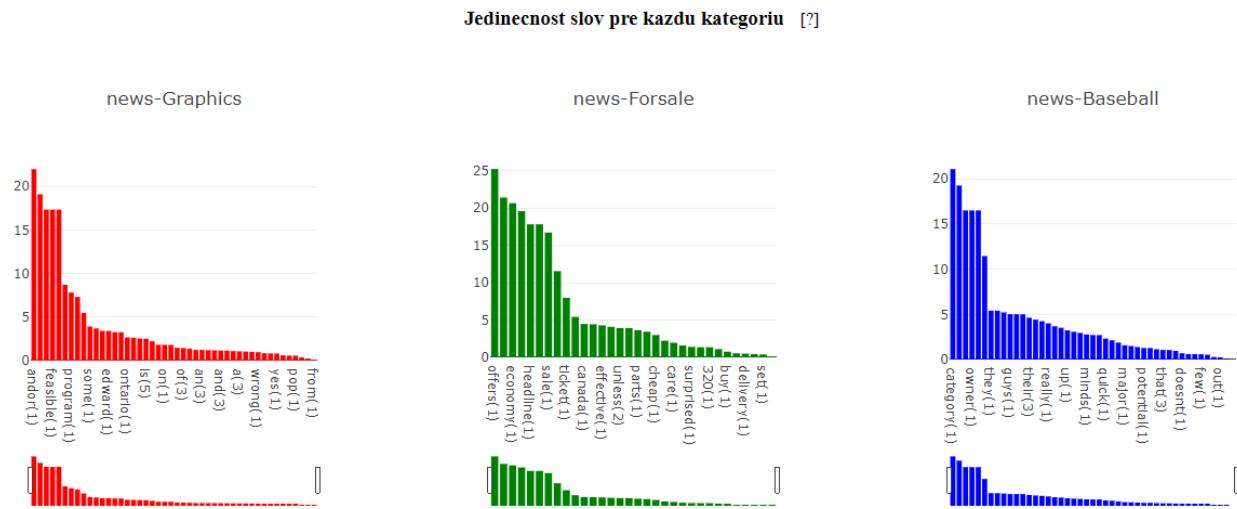
Obr. 7.11: Sekcia klasifikačného procesu pravdepodobnosti

Čiarkované čiary predstavujú pravdepodobnosti samostatných slov ktoré boli vypočítané už v spomínanom pravom ternárnom grafe. Plná čiara vyjadruje sumarizujúcu pravdepodobnosť, ktorá bere do úvahy všetky predošlé slová v jednom segmente slov. Segmente sú oddelované tučným bodom, ktorý farbou vyjadruje kategóriu, ku ktorej by bol príslušný segment slov klasifikovaný. Po každom tomto bode sa tieto sumarizujúce pravdepodobnosti resetujú na to, aby bolo vidieť priebeh výpočtov cez celú vzorku. Túto dĺžku segmentov je možné nastaviť prostredníctvom slideru vyššie od grafu. Takýmto spôsobom môžeme mať predstavu, ku ktorým kategóriám čiastky textového dokumentu viac korešpondovali.

6.2.6 Jedinečnosť slov pre každú kategóriu

Táto sekcia nám zobrazuje jedinečnosť jednotlivých slov vo vzorke pre každú jednu kategóriu.

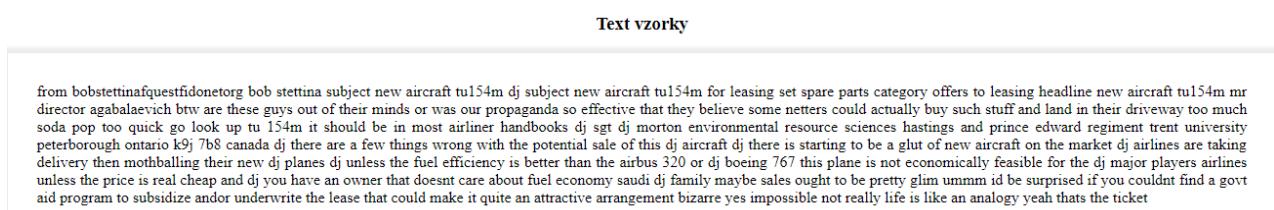
Jedinečnosť slova sa vypočítava percentuálnym rozdielom medzi najvyššou a druhou najvyššou kategorickou pravdepodobnosťou slova. V grafoch sú taktiež vidieť aj hodnoty v zátvorkách indikujúce počet vyskytujúcich slov vo vzorke.



Obr. 7.12: Sekcia jedinečností slov pre každú kategóriu

6.2.7 Text vzorky

Zobrazuje text vybranej vzorky spolu s vybraným rozsahom slov



Pri tejto aplikácii sa zvažovali nasledovné vylepšenia:

- pridanie parametra pre načítanie dát s použitím stop slov
 - použitie princípu jedinečnosti slov pre zdetekovanie slov, ktoré nemali žiadny významný vplyv vo výslednej klasifikácii. Tieto slová by bolo možné priradiť do množiny stop slov
 - podobne ako pri sekcií jedinečnosti slov by sa vytvorila sekcia, ktorá by zobrazovala vplyv slov na výslednú klasifikáciu cez jedinečnosť a zároveň aj početnosť slov vo vzorke.

7 Zhodnotenie tutoriálov

Pre vyhodnotenie tutoriálov bolo potrebné vytvoriť dva rôzne tutoriály. Vo vykonateľnom tutoriály realizovanom v jupyteri bola pokrytá základná teória strojového učenia popri ktorej sa vysvetlovali algoritmy Naivného Bayesa a SVM, pričom pri webovej aplikácii sa používal len algoritmus Naivného Bayesa. Kedže webová aplikácia bola primárne vytvorená ako súčasť vykonateľnej príručky, vytvorila sa ďalšia príručka opisujúca fungovanie aplikácie spolu aj s textom vysvetlenia tak, aby pokryla obsah algoritmu Naivného Bayesa vo vykonateľnom tutoriály.

Takto bolo možné porovnávať výsledky tutoriálov, pričom tutoriály boli stavané aj pre ľudí bez skúseností so strojovým učením.

7.1 Obsah a spôsob vyhodnocovania

Pre otestovanie ľudí sa vytvoril dotazník prostredníctvom ktorého sa zisťovalo presné chápanie kľúčových pojmov pre porozumenie celého klasifikačného procesu algoritmu Naivného Bayesa. Najčastejšie sa teda používali otázky s likelihoodom, priorom a Bayesovým pravidlom.

Priebeh testovania účastníkov pozostával najprv v zisťovaní znalostí pred absolvovaním tutoriálu cez vyplnenie dotazníka. Po absolvovaní tutoriálu vyplnili ten istý dotazník, ktorý mal rozdielne len to, že mal preusporiadane odpovede.

Pred testovaním bolo účastníkom povedané nepoužívať žiadne iné externé zdroje, pomocou ktorých by vedeli zodpovedať na jednotlivé otázky.

Pri tutoriály s webovou aplikáciou nasadenou na školskom serveri sa pre získanie dostatočného počtu ľudí použili verejne dostupný ľudia cez internet, pre ktorých bolo potrebné vytvoriť preloženú verziu dotazníka do angličtiny spolu aj s príručkou pre webovú aplikáciu.

Pri tutoriály s jupyterom boli používané aj otázky z algoritmu SVM, pri ktorom bolo kľúčové pochopiť časti zo základnej teórie strojového učenia. Kedže tieto informácie neboli zahrnené v tutoriály s webovou aplikáciou, neuvádzali sa kvôli nemožnosti porovnania.

7.2 Vyhodnotenie výsledkov

Účastník	Dotazník pred	Dotazník po	Zlepšenie
1.	0	14	14
2.	0	6	6
3.	6	18	12
4.	12	21	9
5.	0	10	10
6.	2	7	5
7.	0	15	15
8.	7	12	5
9.	5	16	11
10.	3	11	6
Priemer	3,5	13	9,3

Tabuľka 7.1: Výsledky testovania s jupyter vykonateľnou príručkou

Účastník	Dotazník pred	Dotazník po	Zlepšenie
1.	10	18	8
2.	11	20	9
3.	6	15	9
4.	0	11	11
5.	0	6	6
6.	0	12	12
7.	3	13	10
8.	0	14	14
9.	4	17	13
10.	0	13	13
11.	0	15	15
12.	0	6	6
Priemer	2,83	13,3	10,5

Tabuľka 7.2: Výsledky testovania s webovou aplikáciou a jej príručkou

V dotazníkoch bolo možné získať maximálny počet bodov 22. Počet účastníkov bolo 9 pre tutoriál s vykonateľnou príručkou v jupyteri, pri tutoriály s webovou aplikáciou sa našlo 12 účastníkov. Pre užitočnosti samotných tutoriálov sa spočítali aj bodové zlepšenia po ich absolvovaní. Kedže bol počet účastníkov na tutoriáloch odlišný, na porovnanie sa použilo zpriemerovanie bodov.

Na základne zpriemerovaných výsledkov zlepšenia je možné vidieť, že účastníci tutoriálu s webovou aplikáciou úspešnejšie zodpovedali otázky v dotazníku. Je treba zohľadniť aj obtiažnosť otázok, ktoré neboli úplne jednoduché najmä pre tých, čo sa stretávajú s touto problematikou prvý krát. Preto mälokto sa našiel s vysokým počtom bodov po absolvovaní tutoriálov. Vo výsledkoch pre výhodnotenie je treba brat' ohľad aj na pomerne malý počet účastníkov. Bolo by potrebné zapojiť väčšie množstvo ľudí na to, aby sme mohli urobiť stabilnejší a dôverihodnejší záver.

Taktiež treba mať na pamäti, že webová aplikácia bola navrhovaná ako rozšírenie tutoriálu v jupyteri, ktorá mala zobrazovať a interpretovať správanie sa procesov algoritmu. Bolo preto obtiažne vytvárať otázky tak, aby cez funkcionality týchto dvoch navrhovaných tutoriálov pokryli ten istý výučbový obsah. Obsah príručky pre webovú aplikáciu bol preto koncipovaný tak, aby pomohol zodpovedať na otázky priamejším spôsobom.

Napriek tomu nám tieto výsledky testovania a zapojenie webovej aplikácie do tutoriálov pomohli potvrdiť využiteľnosť interpretácie procesov vo vnútri algoritmov strojového učenia.

8 Záver

Práca bola zameraná na analýzu a návrh vizualizácií, ktoré vysvetlujú a popisujú správanie procesov vo zvolených metódach strojového učenia, ktoré by mohli slúžiť ako prostriedok pre výučbu. Na začiatku sa spomenuli metriky na meranie efektivity tutoriálov a používané nástroje pre podporu vzdelávania v strojovom učení. Nasledovala teoreticky spracovaná téma strojového učenia, pri ktorej bola zvolená problematika klasifikačných metód. Vysvetluje sa celý proces vytvárania klasifikačného modelu, pričom najväčší dôraz bol kladený na krok evaluácie, po ktorom nasledovali aj prístupy riešenia problémov preučenia a podučenia cez varianciu a bias.

V ďalšej časti bola venovaná pozornosť dvom vybraným algoritmom, ktoré s vysokou úspešnosťou riešia problém textovej klasifikácie. Pri algoritme podporných vektorov sa vysvetlovali len hlavné intuítie fungovania, detailnejší opis bol poskytnutý pre Naivný Bayesovský klasifikátor. Opis algoritmov končil krátkou sumarizáciou výhod spolu s stým, kedy je ktorý vhodné použiť.

Nasledoval návrh riešenia tutoriálu pre vybrané algoritmy cez vykonateľnú príručku poskytujúcu aj interaktívne vizualizácie. Najväčší dôraz bol však kladený na implementáciu interaktívnej webovej aplikácie, ktorá mal za úlohu vizualizovať procesy algoritmu Naivného Bayesa pri textovej klasifikácii. Tieto vizualizácie však priamo nevysvetlovali fungovanie samotného algoritmu, slúžili predovšetkým ako prostriedok pre interpretáciu výsledkov, ku ktorým sa dopracoval. Webová aplikácia bola preto navrhovaná ako rozšírenie tutoriálu.

Cez nutnú potrebu porovnania výsledkov medzi vykonateľnou príručkou a webovou aplikáciou v dvoch rôznych tutoriáloch bolo preto problematické vyhodnotenie v nasledujúcej časti. Vytvorila sa však príručka, ktorá opisuje fungovanie aplikácie so sprevádzaným textom vysvetlenia spomínaného algoritmu. Táto metóda sa následne porovnávala s vykonateľnou príručkou v častiach algoritmu Naivného Bayesa. Ukázalo sa, že webová aplikácia so svojou rozšírenou schopnosťou interpretácie algoritmu mala pri testovaní lepšie výsledky. Pre stabilnejšie a dôverihodnejšie výsledky by však bolo potrebné zozbierat' väčšie množstvo odpovedí.

Literatúra

- [1] NÁVRAT, P. -- BIELIKOVÁ, M. -- BEŇUŠKOVÁ, Ľ. -- KAPUSTÍK, I. -- UNGER, M. *Umelá inteligencia*. Bratislava: STU v Bratislave, 2002. 393 s. ISBN 80-227-1645-6.
- [2] O'HARA, Keith J., Douglas BLANK a James MARSHALL. *Computational Notebooks for AI Education* [online]. 2015 [cit. 2018]. Dostupné z:
<http://science.slc.edu/jmarshall/papers/flairs2015.pdf>
- [3] MITCHELL, Tom M. *Machine Learning*. New York: McGraw-Hill, 1997. ISBN 00-704-2807-7.
- [4] SHARMA, Dharmendra a Suresh JAIN. Evaluation of Stemming and Stop Word Techniques on Text Classification Problem. *International Journal of Scientific Res International Journal of Scientific Research in Com earch in Computer Science and Engineering* [online]. 2015 [cit. 2018]. ISSN 2320-7639. Dostupné z:
<https://pdfs.semanticscholar.org/c5d1/db4509ea7fe81f6c9f0a5f2db8339ef7cb2d.pdf>
- [5] RASCHKA, Sebastian. *Python Machine Learning*. Birmingham, UK: Packt Publishing, 2015. ISBN 978-1-78355-513-0.
- [6] GOODFELLOW, Ian, Yoshua BENGIO a Aaron COURVILLE. *Deep Learning: Machine Learning Basics* [online]. MIT Press, 2016 [cit. 2018]. Dostupné z:
<http://www.deeplearningbook.org>
- [7] VANDERPLAS, Jacob T. *Python data science handbook: essential tools for working with data*. Sebastopol, CA: O'Reilly Media, 2016. ISBN 978-1491912058.

A Technická dokumentácia

Pri nasledovných dvoch nástrojov je potrebné nainštalovať programovací jazyk Python.

A.1 Jupyter Notebook

Na spúšťanie vykonateľných príručiek je potrebné nainštalovať prostredie Jupyter. Na inštaláciu všetkých použitých balíčkov sa v príkazovom riadku zadá *pip install*, za ktorým nasleduje meno balíčka:

- jupyter
- scipy
- sklearn
- matplotlib
- seaborn

Samotné prostredie jupyter je možné spustiť príkazom *jupyter notebook*, pri ktorom sa nám otvorí webový prehliadač. Následne sa vyberie ipynb súbor s názvom nbsvm_tutorial. V priebehu tutoriálu je potrebné spustiť všetky bloky kódov cez *shift+enter*.

Väčšinu použitých balíčkov je možné nainštalovať aj pomocou distribúcie Anaconda dostupnej na <https://www.anaconda.com/download/>.

A.2 Aplikácia Dash

Na inštaláciu Dashu a ich všetkých použitých rozšírení sa v príkazovom riadku podobne ako pri jupyteri zadá *pip install* s menom z týchto nasledovných modulov:

- dash==0.21.0
- dash-renderer==0.12.1
- dash-html-components==0.10.0
- dash-core-components==0.22.1
- plotly --upgrade
- dash-table-experiments
- dash-core-components==0.13.0-rc4
- numpy

Dash aplikáciu je možné spustiť cez príkaz `python dashWEB_sk.py`. Po niekoľkých sekundách ju bude možné zobraziť vo webovom prehliadači na adrese <http://127.0.0.1:8050/>.

Pre použitie iných textových datasetov je možné modifikovať súbor dashREADER.py, v ktorom sú nastaviteľné parametre na načítavanie dát z určitého priečinka v hlavnej funkcií main. Formát pre načítavanie textových dokumentov je nasledovný – datasetový priečinok obsahuje podpriečinky menom vyjadrujúce kategóriu, v ktorých sa vyskytujú korešpondujúce textové dokumenty.

Nastaviteľné parametre pre načítavanie dát:

- dataFolder – meno súboru z ktorého sa budú načítať dátá (string)
- pickPercentage – percentuálny výber textových dokumentov pre každú kategóriu (0-100)
- ratio – percentuálny výber trénovacích dát, zvyšok dát sa použije na testovanie (0-100)
- randomness – použitie náhodného výberu dát (boolean)
- applyMinMax – nájdenie kategórie s najmenším počtom dát a jeho použitie pre výber počtu dát aj v ostatných kategóriách
- stopWords – aplikovanie stop slow (boolean)

Boli používané datasety ohsumed, 20Newsgroups¹⁷ a sentiment labelled sentences¹⁸. Pri implementácii kódu vo vykonateľnej príručke sa vychádzalo z knihy Python data science handbook¹⁹.

¹⁷ TEXT CATEGORIZATION CORPORA. Dostupné z: <http://disi.unitn.it/moschitti/corpora.htm>

¹⁸ 'From Group to Individual Labels using Deep Features', Kotzias et. al., KDD 2015. dostupné z: <https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences>

¹⁹ VANDERPLAS, Jacob T. Python data science handbook: essential tools for working with data. Sebastopol, CA: O'Reilly Media, 2016. ISBN 978-1491912058.

B Zhodnotenie plánu práce

Február	Dokončenie analýzy, oboznamovanie sa s funkciaľitou Dash frameworku, vytváranie návrhov realizovačných vizualizácií v Dashi
Marec	Implementácia Dash webovej aplikácie. Vytvorenie tutoriálu v Jupyter Notebooku
Apríl	Vyhodnocovanie implementácií a ich doladenie na základe výsledkov. Testovanie a dopísanie práce
Máj	Vyhodnocovanie dotazníkov a finalizácia práce

V letnom semestri bola implementácia Dash webovej aplikácie predĺžená kvôli náročnosti navrhovaných vizualizácií, ktoré boli vymýšlané aj za behu pre vylepšenie. Zaobchádzalo sa s elementami, ktoré neboli oficiálne vydané a ani zcela dokumentované. Táto implementácia bola finalizovaná až v polovici apríla. Nastalo posunutie plánu a v priebehu druhej polovice apríla sa vytváral tutoriál v jupyter notebooku. Po zistení potreby porovnania dvoch rôznych tutoriálov bolo nutné prispôsobiť príručku na formu tutoriálu, čo spôsobovalo ďalšie časové komplikácie spolu s vytváraním otázok pre dotazník. Až začiatkom mája sa začalo testovanie a dopisovanie bakalárskej práce. Posledné časti práce bolo potrebné finalizovať urýchleným spôsobom.

C Použité materiály pri testovaní

C.1 Tutoriál v Jupyter Notebooku

Strojové učenie s Naivným Bayesovským klasifikátorom

Naivný Bayesovský klasifikátor je algoritmus založený na Bayesovom pravidle ktorý predpovedá kategórie vzorky (vzorkou môže byť napríklad recenzia ktorá nadobúda kategóriu hodnotenia Good alebo Bad). Spôsob klasifikácie prebieha výpočtom pravdepodobnosti cez všetky možné kategórie do ktorých môže vzorka patriť, pričom kategória s najväčšou pravdepodobnosťou sa použije ako výsledná predpoveď. Na to, aby takéto typy algoritmov (**s učiteľom**) vedeli niečo predpovedať, potrebujú sa najprv niečo naučiť prostredníctvom **trénovacej množiny** (training dataset), ktorá pozostáva zo vzoriek s priradenou kategóriou do ktorej patria. Pre jednoduchosť mu poskytneme nasledovný malý dataset s 5 vzorkami textového ohodnotenia určitého produktu a úlohou bude klasifikovať vzorku s neznámou kategóriou hodnotenia Good/Bad:

Text vzorky	Kategória
excellent headset	Good
mic doesn't work	Bad
love this headset	Good
bad quality mic	Bad
great mic	Good

Na to, aby algoritmy strojového učenia vedeli prečítať vzorky, potrebujú tzv. **atribúty** (features). Atribúty sú vlastnosti vzoriek na základe ktorých sa algoritmus rozhoduje vo svojich predikciach (v medicíne by tieto vlastnosti mohli byť napr. vek, pohlavie a váha). Po nastavení atribútov a natrenovaní algoritmu sa vygeneruje **machine learning model**, ktorý je pripravený predikovať neznáme vzorky. Výber podstatných atribútov a prípadne vytváranie nových (**feature engineering**) pre tieto modely je často krát náročný proces ktorý vyžaduje doménovú znalosť dát, umožňujú však zásadne ovplyvniť efektívitu algoritmov až natoľko, že môžu prekonáť iné sofistikovanejšie algoritmy ktoré štandardne dávajú lepšie výsledky.

Naivný Bayes v našom prípade používa na reprezentáciu textových dát model **bag-of-words**, ktorý sa často krát používa aj pri spracovaní prirodzeného jazyka (natural language processing). Tento model spočíva v spočítavaní frekvencií každého slova vo vzorkách pre kategóriu do ktorej spadajú, atribúty sú teda slová a ich hodnoty sú počty výskytov v kategórii. To bude mať za následok zneusporiadanie slov v teste a s každou vzorkou sa teda zaobchádzza len ako so súborom početnosti slov. Celý proces učenia pozostáva v spočítaní týchto frekvencií vo všetkých trénovacích vzorkách pre ich určenú kategóriu, pričom sa v rámci algoritmu Naivného Bayesa taktiež pridáva atribút s počtom vzoriek pre každú kategóriu. Výsledný model sa dá ilustrovať pomocou nasledovnej tabuľky:

Kategória	excellent	headset	mic	doesnt	work	love	this	bad	quality	great	počet vzoriek
Good	1	2	1	0	0	1	1	0	0	1	3
Bad	0	0	2	1	1	0	0	1	1	0	2

Bayesové pravidlo

Spôsob vypočítavania pravdepodobnosti pre všetky kategórie je založený na Bayesovom pravidle vychádzajúci z teórie pravdepodobnosti:

$$P(A | B) = \frac{P(B | A) \times P(A)}{P(B)}$$

$P(A | B)$ - $\frac{P(A \cap B)}{P(B)}$ - pravdepodobnosť kategórie A, ktorá je podmienená javom B (atribúmi) - posterior

$P(B | A)$ - $\frac{P(B \cap A)}{P(A)}$ - pravdepodobnosť B atribútov, ktorá je podmienená javom A (kategóriou) - likelihood

$P(A)$ - pravdepodobnosť kategórie A - prior

$P(B)$ - pravdepodobnosť atribútov B

V kontexte textovej klasifikácie vypočítanie pravdepodobnosti kategórie Good pri vzorke obsahujúcej text *excellent headset mic quality* je možné vyjadriť nasledovnou rovnicou:

$$P(\text{Good} | \text{excellent headset mic quality}) = \frac{P(\text{excellent headset mic quality} | \text{Good}) \times P(\text{Good})}{P(\text{excellent headset mic quality})}$$

Ak hľadáme kategóriu s najvyššou pravdepodobnosťou, nepotrebuje počítať pravdepodobnosť $P(\text{excellent headset mic quality})$ pretože je rovnaký pre všetky kategórie a výsledok nezmení. V našom prípade teda ide len o porovnanie výsledných pravdepodobností týchto dvoch výrazov:

$$P(\text{Good} | \text{excellent headset mic quality}) = P(\text{excellent headset mic} | \text{Good}) \times P(\text{Good})$$

$$P(\text{Bad} | \text{excellent headset mic quality}) = P(\text{excellent headset mic} | \text{Bad}) \times P(\text{Bad})$$

Použitím spomínaného modelu **bag-of-words** rozobjememe text na slová a dostaneme nasledovný výraz likelihood pre kategóriu Good:

$$P(\text{excellent headset mic quality} | \text{Good}) = P(\text{excellent} | \text{Good}) \times P(\text{headset} | \text{Good}) \times P(\text{mic} | \text{Good}) \times P(\text{quality} | \text{Good})$$

Priebeh klasifikácie algoritmu Naivného Bayesa

Naívita tohto klasifikátora spočíva práve vo vypočítavaní pravdepodobnosti z každých týchto slov zvlášť, čím sa vytvára silný predpoklad, že slová medzi sebou nie sú v žiadnom vzťahu. Klasifikátor sa už teda nepozera na celú vetu, ale len na ich jednotlivé slová. Tako je možné celý **klasifikačný proces vyjadriť** nasledovnou formulou:

$$\hat{y} = \underset{k \in 1, \dots, K}{\operatorname{argmax}} \overbrace{P(C_k)}^{\text{prior}} \overbrace{\prod_{i=1}^n P(w_i | C_k)}^{\text{likelihood}}$$

C_k – aktuálne spracovaná kategória

w_i – aktuálne vypočítavané slovo

K – počet kategórií

n – počet slov v neznámej vzorke

Zhrnutie celého procesu: Hľadáme **argmax** najpravdepodobnejšiu kategóriu \hat{y} do ktorej môže patriť neznáma vzorka. Pre každú z K možných kategórií sa vypočíta pravdepodobnosť ktorá pozostáva z vynásobenia týchto dvoch pravdepodobností:

- prior $P(C_k)$ - pravdepodobnosť vybrania vzorky s kategóriou C_k zo všetkých vzoriek v trénovacej množine ($\frac{\text{počet vzoriek v } C_k \text{ kategórii}}{\text{počet vzoriek vo všetkých kategóriach}}$)
- likelihood $\prod_{i=1}^n P(w_i | C_k)$ - pravdepodobnosť pozostávajúca z vynásobení pravdepodobností medzi všetkými n slovami vyskytujúcimi sa vo vzorke ktorá sa predikuje. Tieto pravdepodobnosti slov sú vypočítávané týmto vzťahom $\frac{\text{počet slov } w_i \text{ v kategórii } C_k}{\text{celkový počet slov v kategórii } C_k}$
 - pri likelihoode môže dôjsť k prípadu slova s nulovou početnosťou, čo spôsobí celkovú nulovú pravdepodobnosť likelihoodu. Preto je potrebné buď nahradiť tieto pravdepodobnosti racionálnym číslom alebo pridať jeden výskyt pre všetky existujúce slová (Laplace smoothing)

Po aplikovaní Laplace smoothingu bude náš model spomenutý na začiatku obsahovať v každej kategórií jedno slovo navyše.

Kategória	excellent	headset	mic	doesnt	work	love	this	bad	quality	great	počet vzoriek
Positive	1+1	2+1	1+1	0+1	0+1	1+1	1+1	0+1	0+1	1+1	3
Negative	0+1	0+1	2+1	1+1	1+1	0+1	0+1	1+1	1+1	0+1	2

Celý priebeh výpočtu pravdepodobností oboch kategórií na základe nášho modelu pri vzorke **excellent headset mic quality** bude vyzerať takto:

$$P(\text{Good} | \text{excellent headset mic quality}) = \frac{\overbrace{\frac{1+1}{7+10} \times \frac{2+1}{7+10} \times \frac{1+1}{7+10} \times \frac{0+1}{7+10}}^{\text{likelihood}} \times \overbrace{\frac{3}{5}}^{\text{prior}}}{\frac{3}{5}} = 0.000517$$

$$P(\text{Bad} | \text{excellent headset mic quality}) = \frac{0+1}{6+10} \times \frac{0+1}{6+10} \times \frac{2+1}{6+10} \times \frac{1+1}{6+10} \times \frac{2}{5} = 0.000293$$

Výslednou predikciou bude kategória Good s najvyššou vypočítanou pravdepodobnosťou.

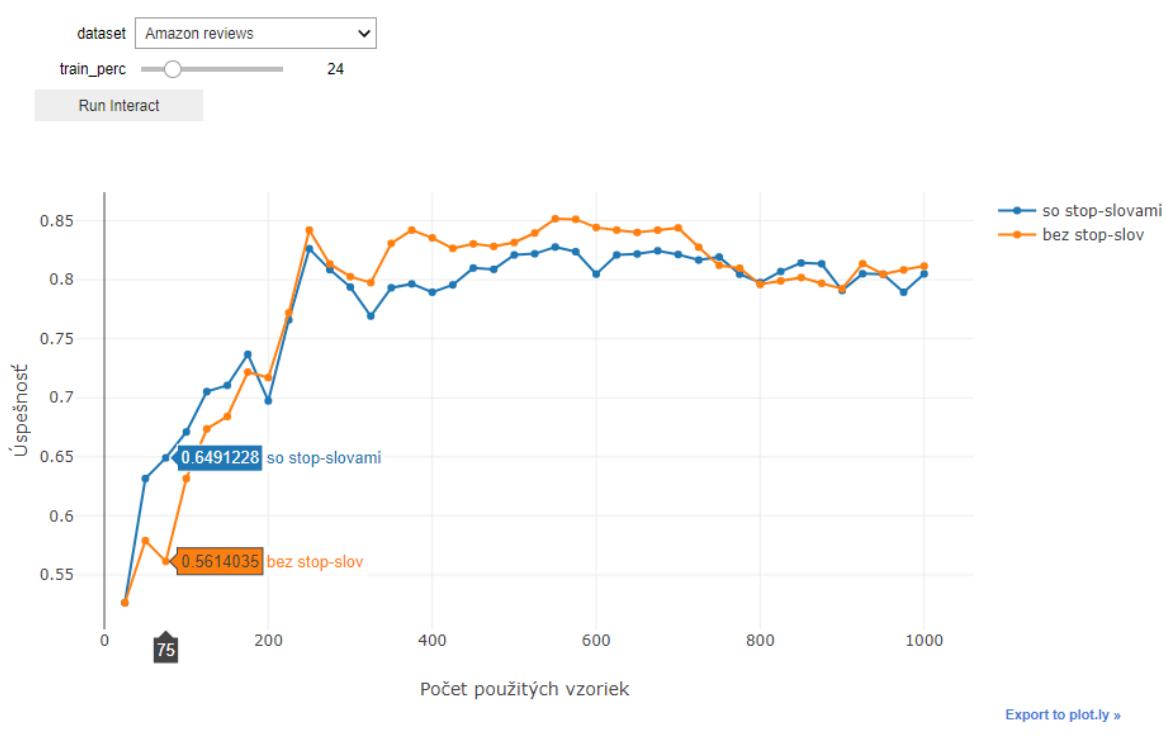
Techniky vylepšenia modelu:

- Stopwordy - použitie stopwordov spočíva v ignorovaní určitých slov, ktoré nemusia obsahovať informačný zisk (the, and, with, a, or, you, we)
- Stemming - proces odstraňovania sufiksov, prefixov, affixov zo slov na získanie základného tvaru slova (working, work, worked, works -> work)
- Lematizovanie - podobné stemmingu, liši sa schopnosťou zachytia kánonické formy slov (good, better, cool -> good)
- N-gramy - počítanie sekvenčí slov napr. v dvojiciach (excellent quality, headset mic)
- TF-IDF - penalizovanie takých slov, ktoré sa často krát vyskytujú vo väčšine vzoriek, používa sa pre nájdenie stopwordov

V nasledujúcom kóde sú implementované funkcie na klasifikovanie ktoré sa dajú spúštať s rôznymi datasetmi pri ktorých je možné nastaviť percentuálny pomer trénovacích a testovacích dát na ktorých sa otestuje klasifikácia. Keďže sa používajú pomerne jednoduché datasety, defaultne nastavený pomer je 24:76 (trénovacie:testovacie) pre lepšie zvíditeľnenie rozdielu medzi výsledkami s použitím aj bez použitia stopwordov. Výstupom interakcie je graf s vypočítanou úspešnosťou pri rôznych počtoch použitých vzoriek, ktoré boli rozdelované do trénovacích a testovacích množín.

```
In [2]: def classifySample(likelihood, priors, contentSample):
    maxPC = -1E6,
    for category in likelihood:
        p = priors[category] # počet všetkých vzoriek v kategórii
        n = sum(likelihood[category].values()) # počet všetkých slov v kategórii
        for word in contentSample:
            p *= (likelihood[category][word]+1) / n # výpočet pravdepodobnosti slova s Laplace smoothingom
        if p > maxPC[0]:
            maxPC = p, category
    return maxPC[1]

stopWords = ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'your', 'yours', 'yourself', 'yourselves', 'he',
'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine']
ipyw.interact_manual(multipleClassify, dataset={'Amazon reviews': 'amazon_cells_labelled.txt', 'Yelp reviews': 'yelp_labelled.txt',
'Imdb reviews': 'imdb_labelled.txt'}, train_perc=(4, 100, 4), # počet vzoriek v každom datasete je 1000
stopWords = ipyw.fixed(stopWords));
```



Je pochopiteľné, že čím je väčší pomer trénovacej množiny, tým je vidieť lepšie výsledky a toto relativne platí aj pri počtoch použitých vzoriek. Pri vyhľadávaní približnej úspešnosti algoritmov strojového učenia sa v praxi často krát používa pomer 80:20 dát. Z grafu je taktiež zrejmé, že aplikovanie stopwordov nemusí vždy pomôcť a často krát úspešnosť záleží na konkrétnych datasetoch. Okrem spomenutých vylepšení modelu boli použité ďalšie iné úpravy pri načítavaní textu, napríklad vymazávanie interpukcií a transformácia veľkých písmen na malé. Existuje množstvo možností predspracovania textu a pri každom datasete môžu vyuhovať iné spracovania. V strojovom učení je táto fáza známa pod názvom **predspracovanie dát** (data preprocessing). Po získavaní dát od rôznych zdrojov často krát dátá niesú v tom istom formáte, vyskytujú sa v nich chyby a je treba ich transformovať do spoločného formátu aby ich vedel algoritmus čítať.

```
In [3]: likelihood, priors, content = jr.readFile('amazon_cells_labelled.txt', 800, 200, stopWords)
priors
Out[3]: Counter({'revs-amazon 0': 391, 'revs-amazon 1': 409})
```

Vplyv priorov

Použité datasety majú približne rovnaký počet vzoriek s negatívnym a pozitívnym ohodnotením. Čím sú tieto rozdiely menšie medzi všetkými možnými kategóriami, tým menší vplyv budú mať prioru na výsledok. Keďže v klasifikačnom procese v nich spočíva len jedno vynásobenie s likelihoodom, pri dokumentoch s väčším počtom násobených slov tvoriacich likelihood začnú prioru ešte zásadnejšie strácať vplyv na výslednú pravdepodobnosť. Vizualizácia vplyvu týchto priorov na výsledok je možné vidieť vo webovej aplikácii. Obsahuje 3 rôzne datasety s rôznymi vlastnosťami, z ktorých je možné vyberať a kombinovať ich príslušné kategórie pre natrénovanie a následnú klasifikáciu.

Skúsme teraz klasifikovať dataset pozostávajúci z 3 rôznych kategórií článkového textu.

```
In [4]: with open('jupydash-likelihood', 'rb') as handle:
    likelihood = pickle.loads(handle.read())
with open('jupydash-priors', 'rb') as handle:
    priors = pickle.loads(handle.read())
with open('jupydash-content', 'rb') as handle:
    content = pickle.loads(handle.read())
print(priors) # vypisanie kategórií s početnosťou vzoriek
print(classifyTestSet(likelihood, priors, content)) # uspesnosť klasifikacie
Counter({'ohsu-Virus_Diseases': 131, 'news-Graphics': 108, 'news-Politics': 102})
0.6173913043478261
```

Vzhľadom na počet kategórií a ich obťaženosť klasifikácie spočívajúcej z veľmi odlišných domén je 61,7% úspešnosť veľmi slabá. Problém v našej klasifikačnej funkcií spočíva v tom, že násobovaním veľkého počtu pravdepodobnosti slov v likelihoode sa približuje k tak extrémne malým hodnotám, že ich programovací jazyk zaokrúhlí na nulu. Preto je potrebné modifikovať klasifikačnú funkciu tak, že aplikujeme funkciu logaritmus pre každú vypočítanú pravdepodobnosť. Násobenie pravdepodobnosti slov sa vo forme logaritmov transformuje na sčítovanie a hodnoty prestanú naberáť extrémne malé čísla. Výsledná formula celého procesu klasifikácie bude vyzerat nasledovne:

$$\hat{y} = \operatorname{argmax}_{k \in 1, \dots, K} \underbrace{\log P(C_k)}_{\text{prior}} + \underbrace{\sum_{i=1}^n \log[P(w_i | C_k)]}_{\text{likelihood}}$$

```
In [5]: def classifySample(likelihood, priors, contentSample):
    maxPC = -1E6,
    for category in likelihood:
        p = math.log(priors[category]) # zlogaritmovaná pravdepodobnosť prioru
        n = sum(likelihood[category].values())
        for word in contentSample:
            p += math.log((likelihood[category][word]+1) / n) # zlogaritmovaná pravdepodobnosť slova
        if p > maxPC[0]:
            maxPC = p, category
    return maxPC[1]
```

```
In [6]: classifyTestSet(likelihood, priors, content)
Out[6]: 0.9565217391304348
```

Iné varianty klasifikátorov vychádzajúce z Naivného Bayesa

Spomenutá metóda klasifikácie je použiteľná len pre diskrétné dátá. Existujú rôzne varianty algoritmov ktoré sa vedia vysporiať aj so spojitými dátami (obsahujúce racionálne čísla v atribútoch). Tieto varianty je možné nájsť v Pythonovej knižnici **scikit-learn**, ktorý je najpoužívanejšou knižnicou pre používanie algoritmov strojového učenia. V prípade Naivného Bayesa obsahuje 3 rôzne varianty: **Gaussian**, **Multinomial** a **Bernoulli Naive Bayes**.

Naivný Bayes má napriek svojim silným predpokladom nezávislosti atribútov veľké využitie v mnohých situáciach v reálnom svete, používajú sa najmä pri klasifikácii dokumentov a filtrovanie spamu. Väčšinou nemávajú takú úspešnosť ako niektoré iné komplexnejšie algoritmy, majú avšak oproti nim množstvo iných výhod.

Kedy má Naivný Bayesovský klasifikátor dobré výsledky?

- keď naivný predpoklad závislosti atribútov zodpovedá poskytnutým dátam (zriedkavý prípad v praxi a stáva sa jeho hlavnou nevýhodou)
- priobre separovaných kategóriách, keď zložitosť modelu je menej dôležitá
- pri extrémne veľkom počte atribútov, keď zložitosť modelu je menej dôležitá

Posledné dva body sa zdajú byť odlišné, no v skutočnosti so sebou súvisia: Ako **dimenzionalita** (početnosť atribútov) datasetu raste, klesá pravdepodobnosť, že akékolvek dva body v priestore budú blízko seba, pretože musia byť blízko v každej jednej dimenzií na to, aby boli blízko celkovo. To znamená, že dátá vo veľkých dimenziách majú tendenciu byť v priemere viac separované ako dátá v nižších dimenziach za predpokladu, že nové dimenzie skutočne pridávajú nejakú informáciu. Z tohto dôvodu môže jednoduchý Naivný Bayesovský klasifikátor práve prostredníctvom naivného predpokladu fungovať veľmi dobre a v niektorých prípadoch aj lepšie ako iné oveľa komplikovanejšie klasifikátory. Pri dostatočnom počte poskytnutých vysoko dimenzionálnych dát môže byť tento algoritmus veľmi úspešný.

Výhody Naivného Bayesovského klasifikátora

- je extrémne rýchly na trénovanie a klasifikovanie
- ľahko pochopiteľný a jednoduchý na implementáciu
- neobsahuje takmer žiadne nastaviteľné parametre
- vystačí si s malým množstvom trénovacích dát

Vychádzaním z týchto výhod je tento klasifikátor dobrou voľbou obzvlášť pri potrebe rýchleho inicializovania základného modelu klasifikácie.

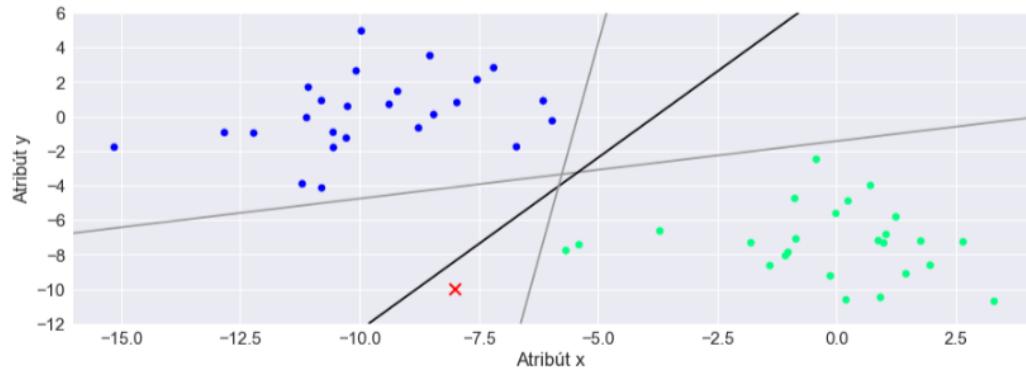
Naivný Bayesovský klasifikátor opisuje rozdelenie každej kategórie zvlášť a používa tiež generatívne modely tak, aby pravdepodobnosť určili kategórie pre neznáme vzorky. Pri ďalšom algoritme sa zameriame na **diskriminatívny model**, ktorý spočíva v tom, že namiesto modelovania rozdelení pre každú kategóriu nájdeme oddelovače kategórií v dvoch či viac dimenzionálnych atribútových priestoroch. To znamená, že nebude treba vypočítavať pravdepodobnosti pre každú kategóriu, stačí sa v prípade dvoch kategórií spýtať, na ktorej z dvoch kategorických strán leží neznáma vzorka priamo v priestore cez jej atribúty.

Support Vector Machines (SVM)

Jednou z najintuitívnejších príkladov ako riešiť klasifikačný problém cez diskriminatívny model je pomocou algoritmu Support Vector Machines. Tento diskriminatívny klasifikátor sa snaží nájsť čiary alebo krivky oddelujúce 2 kategórie na základe ktorej sa rozhodne, do ktorej kategórie bude neznáma vzorka predikovaná.

Predstavme si jednoduchý dataset s dvojdimenzionálnymi dátami (s dvomi atribútmi), pozostávajúci z dvoch ľahko oddeliteľných kategórií. Nebolo by problém nájsť správne oddelujúcu čiaru medzi nimi, problém však nastáva v tom, ako nájsť tú správnu. Existuje ich nekonečné množstvo a každá z nich môže zásadne ovplyvniť úspešnosť predikcie neznámych vzoriek. V nasledujúcom obrázku je možné vidieť niekolko čiar oddelujúcich trénovacie dátu spolu aj s práve jednou takou neznámyou vzorkou ktorá je vyznačená červeným X.

```
In [7]: X, y = make_blobs(n_samples=50, centers=2, random_state=9, cluster_std=2) # vytvorenie datasetu
yfit = np.linspace(-12, 6) # vytvor cisla na intervale <-1,3.5>
plt.figure(figsize=(15,5)) # velkosť grafu
plt.scatter(X[:, 0], X[:, 1], c=y, s=40, cmap='winter') # trenovacie body
plt.plot([-8, [-10], 'x', color='red', markeredgewidth=2, markersize=10) # testovaci bod
for m, b, c in [[3, 4.3, '0.6'], [0.5, -3.8, '0.0'], [0.1, -5.42, '0.6']]:
    plt.plot(m * yfit + b, yfit, c) # oddelovace
plt.xlabel('Atribút x')
plt.ylabel('Atribút y')
plt.xlim(-16, 4)
plt.ylim(-12, 6);
```

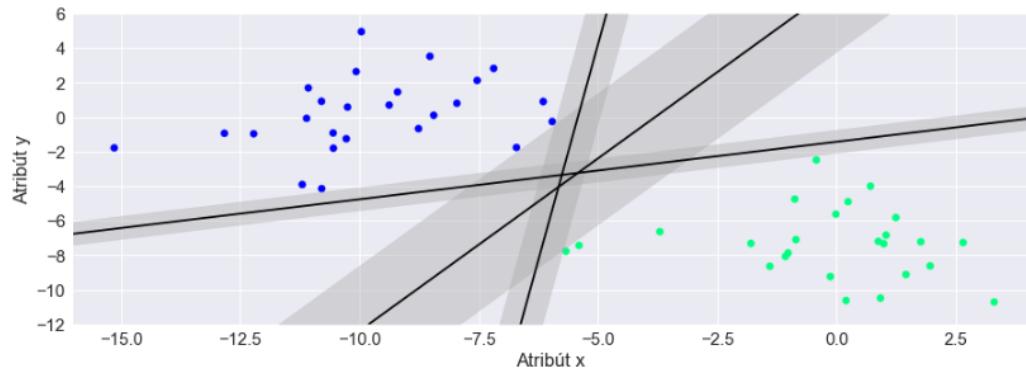


Zo znázornených troch čiar je vidieť, že čierna čiara tieto dátu oddeluje najlepším spôsobom. Ako ju však máme nájsť a podľa čoho? Je zrejmé, že takáto jednoduchá intuícia kreslenia čiar medzi kategóriami nestačí a hlavnou myšlienkom SVM algoritmu je práve nájsť túto najlepšie oddelovaciu čiaru.

Maximalizovanie hraníc dvoch kategórií

Intuícia pre nájdenie najlepšej oddelovacej čiary spočíva v hľadaní čiary s najväčšou hrubkou (cesta), ohraničenou najbližšie vyskytujúcimi sa bodmi vzoriek. V prípade našich čiar by tieto hrúbky vyzerali nasledovne.

```
In [8]: yfit = np.linspace(-12, 6)
plt.figure(figsize=(15,5))
plt.scatter(X[:, 0], X[:, 1], c=y, s=40, cmap='winter')
for m, b, d in [[3, 4.3, 2], [0.5, -3.8, 1.9], [0.1, -5.42, 0.45]]:
    xfit = m * yfit + b
    plt.plot(xfit, yfit, '-k')
    plt.fill_between(xfit, yfit - d, yfit + d, color="#AAAAAA", alpha=0.4)
plt.xlabel('Atribút x')
plt.ylabel('Atribút y')
plt.xlim(-16, 4)
plt.ylim(-12, 6);
```

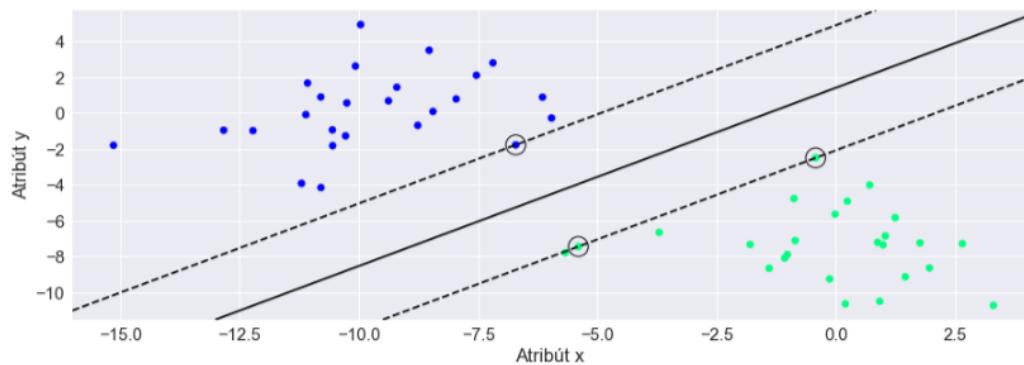


Pozrime sa teraz ako túto čiaru nájde samotný SVM algoritmus. Použijeme modul z knižnice scikit-learn pri ktorom nastavíme tzv. kernel linear pre lineárne (čiarové) oddelenie kategórií, význam parametru C bude vysvetlený neskôr. Vo výstupe nasledujúceho kódu je možné vidieť všetky možné nastaviteľné parametre.

```
In [9]: from sklearn.svm import SVC
model = SVC(kernel='linear', C=100)
model.fit(X, y)

Out[9]: SVC(C=100, cache_size=200, class_weight=None, coef0=0.0,
            decision_function_shape='ovr', degree=3, gamma='auto', kernel='linear',
            max_iter=-1, probability=False, random_state=None, shrinking=True,
            tol=0.001, verbose=False)
```

```
In [10]: plt.figure(figsize=(15,5))
plt.xlim(-16, 4);
plt.xlabel('Atribút x')
plt.ylabel('Atribút y')
plt.scatter(X[:, 0], X[:, 1], c=y, s=40, cmap='winter')
plot_svc_decision_function(model);
```



Plná čiara predstavuje oddeľovaciu čiaru (**hyperplane**), ktorá maximalizuje rozpäťie (**margin**) medzi kategóriami prostredníctvom troch vyznačených dátových bodov (**podporné vektor alebo support vectors**) tvoriace cestu. Tieto body sú kľúčové prvky k vytvoreniu klasifikačného modelu a preto je od nich odvodené meno samotného algoritmu. V scikit-learnerne sa tieto podporné vektory v klasifikátore ukladajú do atribútu `support_vectors_`:

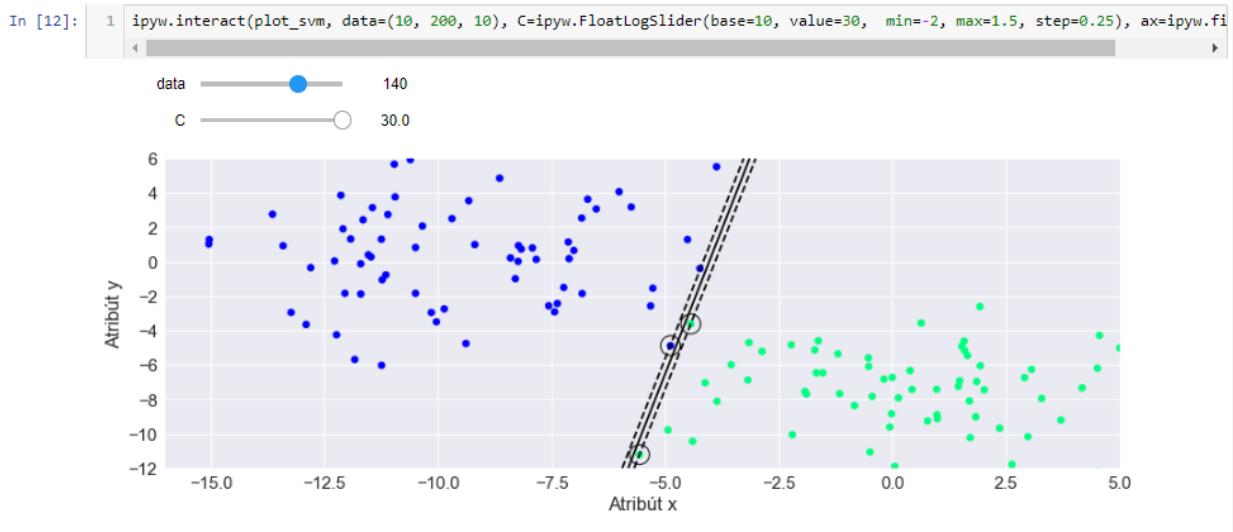
```
In [11]: model.support_vectors_

Out[11]: array([[-6.70705767, -1.7639505 ],
               [-0.42401368, -2.48919677],
               [-5.39599847, -7.43877392]])
```

Kľúčom k úspechu tohto klasifikátora je skutočnosť, že je založený len na umiestnení týchto podporných vektorov, akékoľvek iné body nachádzajúce sa ďaleko od okrajov vôbec nemenia výsledný tvar klasifikačného modelu. Problém však nastáva pri ľahšom rozložení dát s malým rozpätím medzi kategóriami ktorý sice je maximalizovaný a oddeľuje trénovacie dátá perfektným spôsobom, ale často krát to neplatí pri predikciach u neznámych vzorkov, pretože neberie do úvahy frekvencie dát na určitých miestach nachádzajúcich sa za okrajmi. Model sa príliš snaží správne kategorizovať trénovacie dátá, stráca tým však generalizáciu riešenia u neznámych dát a trpí tzv. **preučením**.

Z výstupu nasledujúceho kódu je možné meniť počet dát spolu s penalizačným alebo regularizačným parametrom C, cez ktorý je možné zgeneralizovať chybu pre testovacie dátá. Pri vysokej hodnote C sa algoritmus snaží kategorizovať trénovacie dátá čo najpresnejšie a spôsobuje nájdenie čiary s malým rozpätím. Naopak nízke hodnoty vytvárajú väčšie rozpätie a tolerujú určitú chybu v trénovaní. Zoberie sa tým do úvahy väčšie množstvo podporných vektorov naokolo pomocou ktorého bude výsledný model viac zgeneralizovaný.

Prikladom preučenia je najviac vidieť medzi použitím počtu dát 130-140 s veľkým parametrom C. Jedna (modrá) vzorka tam zásadným spôsobom zmenila celý klasifikačný model. Prostredníctvom zníženia parametra C je možné upraviť tento model a zohľadniť viaceru bodov na okolo.

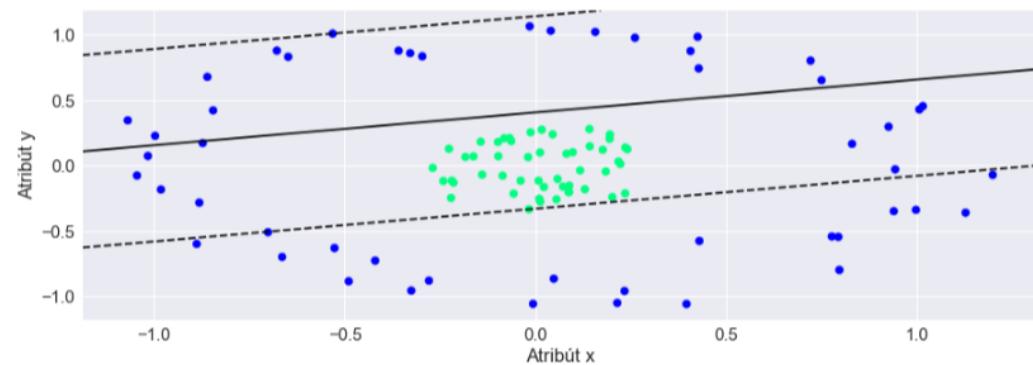


Čo keď máme dátá ktoré vytvárajú iný tvar rozdelenia kategórií?

In [13]:

```
from sklearn.datasets.samples_generator import make_circles
X, y = make_circles(100, factor=.2, noise=.1)

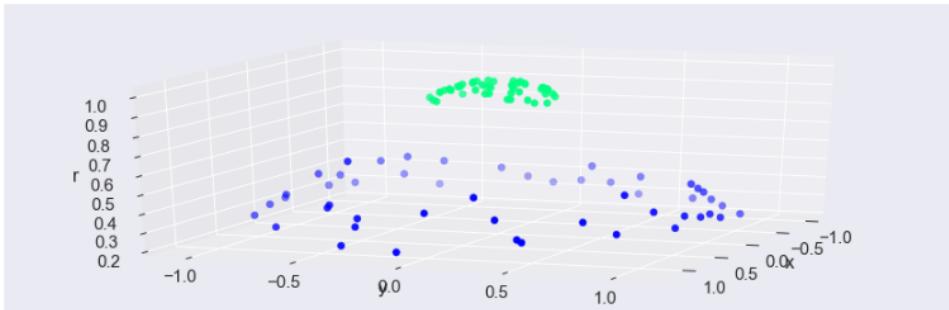
clf = SVC(kernel='linear').fit(X, y)
plt.figure(figsize=(15,5))
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='winter')
plt.xlabel('Atribút x')
plt.ylabel('Atribút y')
plot_svc_decision_function(clf, plot_support=False);
```



Je jasné vidieť, že žiadna lineárna diskriminácia nebude nikdy schopná oddeliť takéto rozdelenie dát. Skúme sa však na ne pozreť z trojrozmernej perspektívy prostredníctvom nasledujúceho grafu.

```
In [14]: 1 from mpl_toolkits import mplot3d
2 r = np.exp(-(X ** 2).sum(1))
3
4 def plot_3D(elev=20, azim=20, X=X, y=y):
5     fig = plt.figure(figsize=(15,5))
6     ax = fig.add_subplot(1, 1, 1, projection='3d')
7     ax = plt.subplot(projection='3d')
8     ax.scatter3D(X[:, 0], X[:, 1], r, c=y, s=40, cmap='winter')
9     ax.view_init(elev=elev, azim=azim)
10    ax.set_xlabel('x')
11    ax.set_ylabel('y')
12    ax.set_zlabel('r')
13 ipyw.interact(plot_3D, elev=(0, 90, 10), azim=(0, 90, 10), X=ipyw.fixed(X), y=ipyw.fixed(y));
```

elev
azim



Použitie viacrozmerného priestoru

Je jasné, že s touto dodatočnou dimenziou sa už tieto dátajú jednoducho rozdeliť prostredníctvom taktiež navýšenej dimenzie oddelovača - plochy, ktorá by sa ideálne mohla nachádzať v $r=0.7$. Ak by tieto dátu boli ešte komplikovanejšie rozložené, potrebovali by sme použiť ešte väčší dimenzionálny priestor. Taký sa už ale ľahko vizualizuje a niesu ešte rozdeluje.

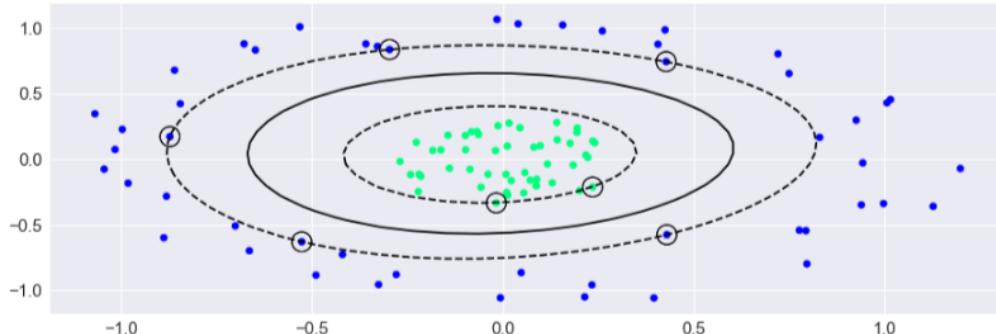
Hlavný trik SVMiek spočíva práve v tom, že vie prevádzkať atribúty do nekonečneho rozmerného priestoru prostredníctvom tzv. **kernelového triku** tak, aby ich vedel následne rozdeľovať analogickým spôsobom ako pri dvojrozmerných dát. Jeden zo známych používaných kerneľov je kernel RBF (radial basis function), ktorý je možné v scikit-learn nastaviť pri inicializovaní SVM modelu.

```
In [15]: clf = SVC(kernel='rbf', C=1E6)
clf.fit(X, y)

Out[15]: SVC(C=1000000.0, cache_size=200, class_weight=None, coef0=0.0,
              decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
              max_iter=-1, probability=False, random_state=None, shrinking=True,
              tol=0.001, verbose=False)
```

Pomocou tohto kernelizovaného SVM modelu je klasifikátor naučený separovať kategórie nelineárny spôsobom, ktorý je ilustrovaný na nasledujúcom obrázku.

```
In [16]: fig = plt.figure(figsize=(15,5))
plt.scatter(X[:, 0], X[:, 1], c=y, s=40, cmap='winter')
plot_svc_decision_function(clf)
plt.scatter(clf.support_vectors_[:, 0], clf.support_vectors_[:, 1], s=300, lw=1, facecolors='none', edgecolors='k');
```



Klasifikácia viacerých kategórií

Doteraz sme používali len binárnu klasifikáciu pre 2 rôzne kategórie. Na to, aby sme mohli predikovať väčší počet kategórií je treba problém transformovať na viacero binárnych klasifikačných podproblémov. Používajú sa nasledovné 2 prístupy:

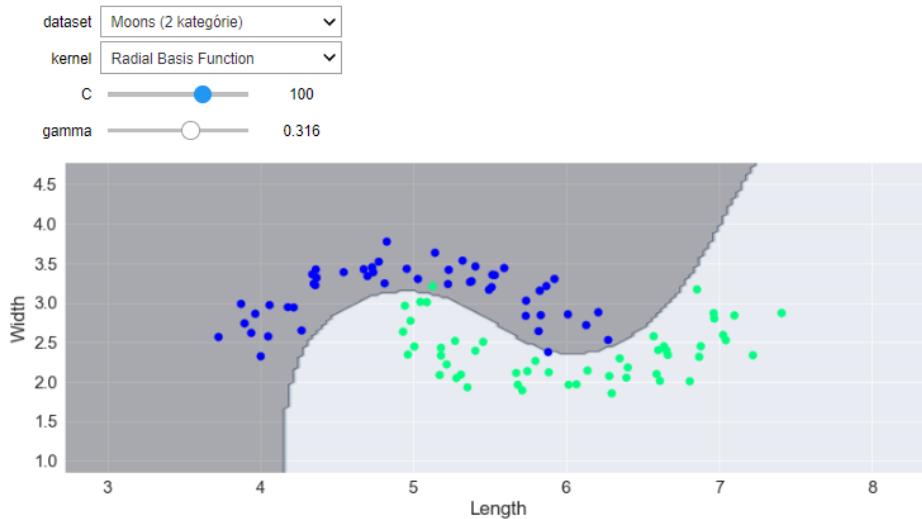
1. **One-vs-all** - počas trénovalia sa vytvoriť klasifikátor pre každú kategóriu, pričom vzorky príslušných kategórií sú pozitívne a všetky ostatné sú negatívne. Pri klasifikácii sa vyberie ten klasifikátor, ktorý má najširšie okrajové rozdelenie.
2. **One-vs-one** - počas trénovalia sa vytvára klasifikátor pre každý možný páru kategórií. V čase predikcie sa používa schéma hlasovania a kategória s najvyšším počtom predpovedí sa použije ako výsledok. Výhoda tohto prístupu nad one-vs-all spočíva v lepšej využitosti výsledných rozdielov medzi kategóriami, nevýhodou je ale oveľa pomalšie trénovalie keďže obsahuje celkovo $K(K-1)/2$ klasifikátorov. V praxi sa väčšinou používa prvý prístup.

Hyperparametre

Pri každom datasete na ktorý chceme aplikovať algoritmus strojového učenia je potrebné vyhľadať vhodné parametre pre čo najoptimálnejší model. Tieto parametre sa nazývajú **hyperparametre**. Najstandardnejším hyperparametrom je už spomínaný regularizačný parameter ktorý sa často používa pri väčšine algoritmov strojového učenia. V prípade SVM existuje ešte jeden hyperparameter γ (gamma), ktorý je nastaviteľný pri RBF kerneli. Prostredníctvom tohto parametra vieme nastaviť do akéj miery chceme tvarovať komplexné tvary tak, aby zodpovedali trénovaliacim dátam. Čím je väčší, tým je model náchylniejsí k preučeniu.

Z výstupu nasledujúceho kódu je možné nastavovať tieto parametre a pozorovať ich vplyv na vytváranie SVM klasifikačných modelov. Použijeme pri tom známy dataset iris ktorý sa naša na scikit-learnu.

```
In [17]: 1 ipyw.interact(funkcia, dataset={'Iris (2 kategórie)': 2, 'Iris (3 kategórie)': 3, 'Moons (2 kategórie)': 1},
2                         kernel=['Radial Basis Function': 'rbf', 'Linear': 'linear'],
3                         C=ipyw.FloatLogSlider(base=10, value=10, min=-2, max=4, step=0.5),
4                         gamma=ipyw.FloatLogSlider(base=10, value=0.1, min=-4, max=2, step=0.5)); # Len pre RBF
```



Jeden z prístupov hľadania vhodných hyper parametrov pre zlepšenie úspešnosti na konkrétnom datasete je možné manuálne, intuitívne či hrubou silou. Pri vyhľadávaní hrubou silou sa používa tzv. GridSearch, ktorý prehľadáva mriežky všetkých možných parametrov pre určité algoritmy

SVM algoritmus bol pred neurónovými sietami najpopulárnejší klasifikačný algoritmus v strojovom učení. Osvedčil sa kvôli svojej vysokej presnosti a flexibilite použitia pri rôznorodých dát. Okrem klasifikácie sa taktiež používa aj pri regresii kde sa namiesto klasifikácie kategórie predikuje číselná hodnota. Dobre vysvetlený a matematicky odvodnený SVM je možné vidieť v prednáške od Patricka Winstona https://www.youtube.com/watch?v=_PwhiWxHK8o

Výhody:

- veľká rýchlosť klasifikácie
- kernelový trik zabezpečuje univerzálnosť modelu pre rôznorodé dátu
- kompaktný model zaberajúci málo pamäti prostredníctvom závislosti od pomerne málo podporných vektorov
- narodenie od väčšiny iných algoritmov fungujú veľmi dobre s vysoko rozmeronymi dátami ktoré môžu aj presahovať celkový počet vzoriek

Nevýhody:

- pomalá rýchlosť trénovalia pri veľkých datasetoch
- finálny model je ťažko interpretovateľný
- nájdenie správnych parametrov kernelu nie je jednoduché najmä pri veľkom počte dát

Vzhľadom na tieto charakteristiky sa odporúčajú SVM ktorá používajú najmä vtedy, keď nám výsledky iných algoritmov s menšími požiadavkami na ladenie hyperparametrov nestačia. Toto platí najmä pri veľkom počte dát. Ak máme dostupnú vysokú výpočtovú silu pre natrénovalie a nastavovanie hyperparametrov, môže táto metóda viesť k vynikajúcim výsledkom.

C.2 Webová aplikácia Dash

Textova klasifikacia s Naivnym Bayesovskym klasifikatorom

Naivny Bayesovsky klasifikator je algoritmus strojového učenia využívajúci Bayesove pravidlo založené na teórii pravdepodobnosti pre klasifikovanie kategórii vzoriek.

Proces učenia spociva v dodavaní trenovacích dat - vzorky (textové dokumenty) s priradenou kategóriou. Spocítajú sa frekvencie každého slova vo vzorkach pre ich priradeniu kategórii a taktiež sa spocítajú aj vzorky pri každej kategórii.

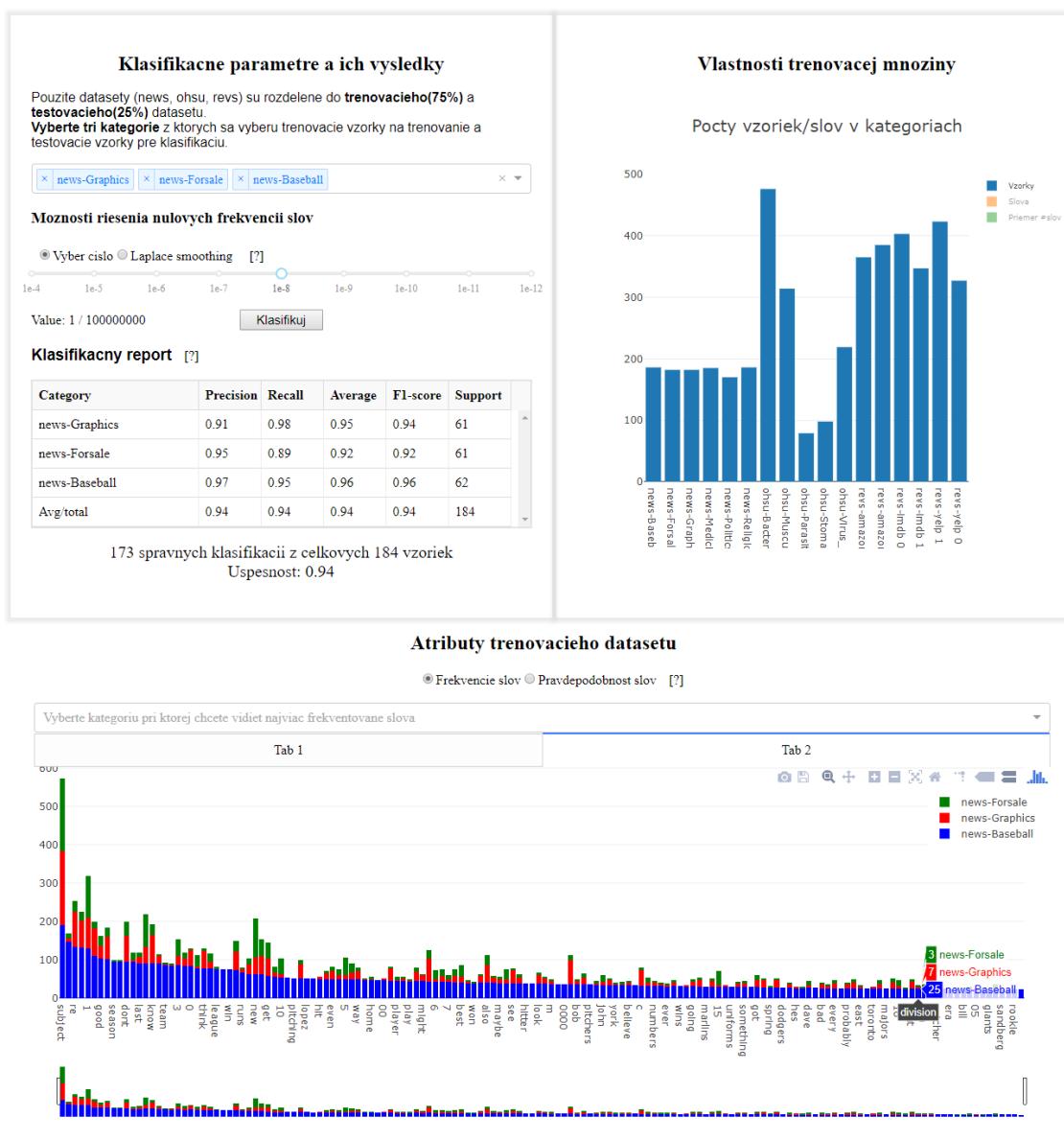
Proces klasifikacie je možné vyjednati nasledujúcu výpočtovou formulou:

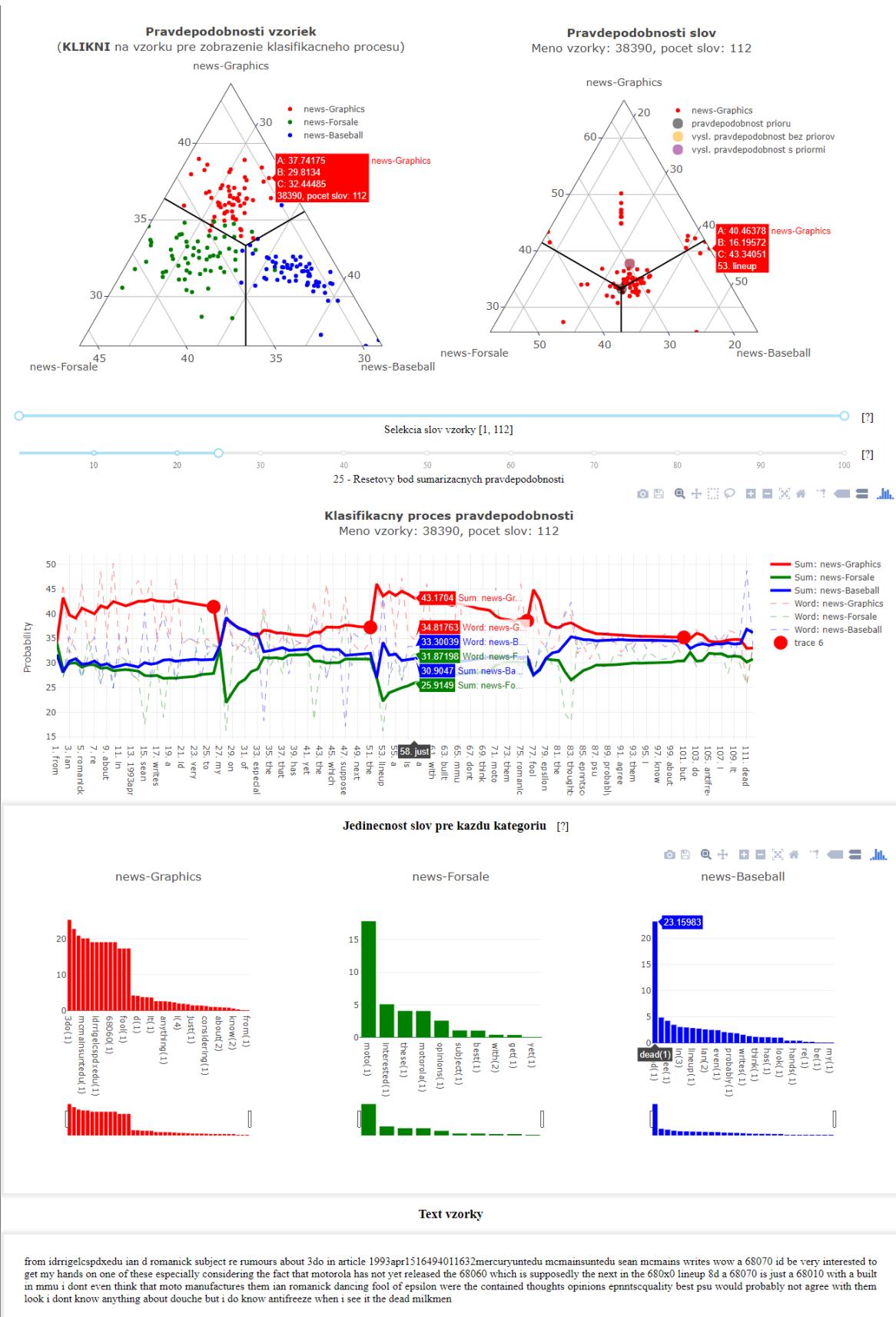
$$\hat{y} = \underset{k \in \{1, \dots, K\}}{\operatorname{argmax}} P(C_k) \prod_{i=1}^n P(w_i | C_k)$$

C_k - aktuálne spracovaná kategória
 w_i - aktuálne vypočítavane slovo actually being computed
 K - počet kategórii
 n - počet slov v neznamenej vzorke

Zhrnutie celého procesu: Hľadame argmax najpravdepodobnejšej kategórii (y^*) do ktorej može patrniť neznáma vzorka. Pre každu z K možnych kategórii sa vypočíta pravdepodobnosť ktorá pozostáva z vynásobenia týchto dvoch pravdepodobností:

- **prior** $P(C_k)$ - pravdepodobnosť vybrať vzorku s kategóriou C_k zo všetkých vzoriek v trenovacej mnozine (#vzoriek v C_k kategórii / #vzoriek vo všetkých kategóriach)
- **likelihood** $\prod_i P(w_i | C_k)$ - pravdepodobnosť pozostavajúca z vynásobení pravdepodobnosti medzi všetkymi n slovami vyskytujúcimi sa vo vzorke ktorá sa predikuje. Tieto pravdepodobnosti sú vypočítané týmto vzťahom: (#slov w_i v kategórii C_k) / (celkový # slov v kategórii C_k)
 - pri likelihoode može dojsť k prípadu slova s nulovou početnosťou, čo spôsobí celkovú nulovú pravdepodobnosť likelihoodu. Preto je potrebné bud nahrať tiež pravdepodobnosť racionálom cílom alebo pridať jeden výskyt pre všetky existujúce slova (Laplace smoothing).





C.2.1 Dash príručka

Naivný Bayesovský klasifikátor

Je algoritmus strojového učenia využívajúci Bayesové pravidlo založené na teórii pravdepodobnosti pre klasifikovanie kategórie vzorky (kde napr. vzorka je textový dokument a kategóriou môže byť šport či umenie). Na reprezentáciu textových dát pri učení sa klasifikácie je používaný model bag-of-words, ktorý spočíva v spočítavaní frekvencií každého slova v trénovacích vzorkách pre kategóriu do ktorej spadajú. Majme napríklad 5 trénovacích vzoriek s dvoma rôznymi ohodnoteniami (kategóriami) produktu Good a Bad v nasledujúcej tabuľke:

Text vzorky	Kategória (review)
excellent headset	Good
mic doesn't work	Bad
love this headset	Good
bad quality mic	Bad
great mic	Good

Reprezentácia textu vzoriek pre každú kategóriu sa cez model bag-of-words dá vyjadriť nasledujúcou tabuľkou, kde atribúty sú slová a ich hodnoty sú frekvencie v dvoch rôznych kategóriách.

Kategória	excellent	headset	mic	doesn't	work	love	this	bad	quality	great
Good	1	2	1	0	0	1	1	0	0	1
Bad	0	0	2	1	1	0	0	1	1	0

Spočítavanie frekvencií slov je takmer celý proces učenia sa a prostredníctvom tejto tabuľky máme najväčší nástroj pre vypočítavanie pravdepodobností pre každú kategóriu do ktorej môže neznáma vzorka patriť. Na predikovanie kategórie neznámej vzorky jednoducho vybereme kategóriu s najväčšou vypočítanou pravdepodobnosťou.

Výber kategórií

Na to, aby sme videli ako presne funguje celý Naivný Bayesovský klasifikátor, vo webovej aplikácii máme v sekcii “klasifikačné parametre a ich výsledky” vybraté tieto 3 kategórie:

- ohsu-Bacterial_Infections_and_Mycoses
- ohsu-Stomatognathic_Diseases
- news-Medicine

Výber vzorky

Klikneme na Klasifikuj a nižšie v trojuholníkovom grafe, predstavujúci pravdepodobnosti niekoľkých klasifikovaných neznámych vzoriek je vidieť jednu zle klasifikovanú vzorku ktorá patrí do Bacterial_infections. Keď na ňu klikneme, zobrazia sa nám ďalšie grafy prostredníctvom ktorých môžeme vidieť výpočtové procesy vzorky.

Trojuholníkový graf napravo obsahuje slová kliknutej vzorky a ukazuje ich vypočítané pravdepodobnosti spočívajúce ku ktorým kategóriám najviac korešpondujú.

Nižšie od trojuholníkových grafov je graf vyjadrujúci priebeh výpočtov pravdepodobností po slovách v neznámej vzorke. Čiarkované čiary predstavujú pravdepodobnosti samostatných slov, ktoré je možné vidieť v pravom trojuholníkovom grafe. Plná čiara vyjadruje summarizujúcu pravdepodobnosť ktorá bere do úvahy všetky predošlé slová v jednom segmente slov. Segmenty sú oddelované tučným bodom, ktorý farbou vyjadruje kategóriu ku ktorej by bol príslušný segment slov klasifikovaný. Po každom tomto bode sa tieto summarizujúce pravdepodobnosti resetujú (pre lepšiu vizualizáciu výpočtov cez celú vzorku). Dĺžku segmentov je možné zmeniť prostredníctvom slideru vysšie – “Resetovy bod summarizacnych pravdepodobností”.

Slová

Nastavme ho na 35. Uvidíme na začiatku druhého segmentu slov ako pravdepodobnosťne vedie kategória, ktorá spôsobila chybnú klasifikáciu (Stomatognathic_Diseases) druhého segmentu slov. Najväčší chybný vplyv mal v tomto segmente slová tumors (39. slovo), bony (47), tumor (49/56) dissection (66). Tieto slová je možné vidieť aj v grafe uvedenom nižšie, ktorý hovorí o ich jedinečnosti cez výpočet percentuálneho rozdielu medzi najpravdepodobnejším a druhým

najpravdepodobnejším slovom medzi kategóriami. Pozrime sa na slovo tumor (49) bližšie cez atribúty trénovacej množiny, ktorý je vyjadrený modelom bag-of-words.

Atribúty trénovacieho datasetu

Graf týchto atribútov je zobrazený vyššie (atributy trenovacieho datasetu). Pre zobrazenie najfrekventovanejších používaných slov kategórie spôsobujúcu chybu klikneme na dropdown pre výber tejto kategórie (Stomatognathic_Diseases) a nájdeme slovo tumor napr. cez ctrl+f. Je vidieť, že frekvencia tohto slova je veľmi malá a pomerový rozdiel s ostatnými kategóriami nieje tak veľký, no napriek tomu toto slovo malo zásadný vplyv na pravdepodobnostný výpočet. (Tento príklad nieje veľmi šťastný, o niečo neskôr spomeniem slovo na ktorom je vidieť krajšie vidieť frekvencie spolu aj s pomerom)

V tomto grafe taktiež môžeme vidieť výsledné pravdepodobnosti slov cez možnosť "Pravdepodobnosť slov". Cez tieto dve možnosti je vidieť, ako pravdepodobnosť nezávisí len od samotných frekvencií slov pretože pomer hodnôt medzi kategóriami sa zásadne mení.

Vlastnosti trénovacej množiny

Na to, aby sme mohli počítať pravdepodobnosť slova je treba zohľadniť, koľko slov sa vyskytuje vo všetkých vzorkách jednej kategórie. Ak by sme mali napr. 10 slov v jednej kategórii a 3 z nich by boli „zdravie“, je omnoho pravdepodobnejšie, že slovo zdravie bude viac korešpondovať s touto kategóriou ako s kategóriou ktorá obsahuje 7 takýchto slov s celkovým počtom slov až 300.

V našom prípade rozdiel v počtoch slov medzi kategóriami (Stomatognathic a Bacterial) je cca 7-násobný. Toto je možné spozorovať pri najvyššie zobrazenom grafe (Vlastnosti trenovacej množiny). Defaultne sú nastavené počty trénovacích vzoriek pre každú kategóriu, pomocou interaktívnej legendy však môžeme zakliknúť "Slova" pre zobrazenie počtov slov v každej kategórii.

Ak sa napr. pozrime na 4. alebo 5. slovo (treatment a disease), v grafe „atributy trenovacieho datasetu“, môžeme vidieť, že kategória Bacterial ich má najväčší počet, ak sa však pozrime na ich výsledné pravdepodobnosti, kategória Stomatognathic má prekvapivo pravdepodobnejší výskyt tohto slova.

Pravdepodobnosť slova neznámej vzorky sa teda počíta počtom výskytov (z trénovacích vzoriek) určitého slova v kategórii vydeleným celkovým počtom slov (z trénovacích vzoriek) v kategórii. Tieto vypočítané pravdepodobnosti sú násobené každým ďalším slovom vo vzorke a tvoria výslednú pravdepodobnosť - **Likelihood**.

Klasifikačný proces vzorky obsahuje ešte jedno vynásobenie s tzv. **Priorom**. Tým, že sa jedná len o jedno vynásobenie navyše, pri vzorkách s dlhým textom mávajú tieto priory veľmi slabý vplyv.

Tak ako sme brali do úvahy celkový počet slov v kategóriach, môžeme brať do úvahy aj počty vzoriek v každej kategórii. Ak máme kategóriu so zriedkavejšími výskytmi vzoriek, chceli by sme, aby bola o niečo menej pravdepodobná vo výslednej klasifikácii.

Pravdepodobnosť Prioru dosiahneme počtom vzoriek v kategórii vydeleným celkovým počtom vzoriek vo všetkých kategóriách. Čím majú kategórie podobnejšie počty vzoriek, tým budú mať menší vplyv na výsledok. Ak však tam aj je veľký rozdiel, pri klasifikácii dlhých textov to bude mať tak či tak slabý vplyv na výsledok.

Vizualizácia vplyvu priorov

Vplyv je možné vidieť v grafe pravdepodobností slov. Kedže máme zakliknutú vzorku s veľkým počtom slov, tento vplyv priorov je veľmi slabý a nie je vidieť odlišnosť vo výsledných pravdepodobnostiach, ktoré sú označené oranžovou a fialovou farbou. Na to, aby sme videli lepšie vplyv priorov, zredukujme text označenej vzorky prostredníctvom slideru "Selekcia slov vzorky" nižšie. Stiahnime bod konca na začiatok pre výber napr. len prvých 2 slov. Uvidíme, že výsledné pravdepodobnosti bez prioru a s priorom sa odlišujú omnoho viac. Každým pridaným slovom môžeme sledovať, ako sa tieto výsledné pravdepodobnosti približujú k sebe.

Bayesové pravidlo

Proces výpočtu neznámej vzorky pre určenú kategóriu je inšpirovaný už spomínaným Bayesovým pravidlom, kde B je neznáma vzorka a A je kategória:

$$\overbrace{P(A | B)}^{\text{posterior}} = \frac{\overbrace{P(B | A) \times P(A)}^{\text{likelihood}}}{\overbrace{P(B)}^{\text{prior}}}$$

- Presnejšie vyjadrenie výpočtov pravdepodobností je vyjadrený vo webovej aplikácii.
- Tento algoritmus je možné modifikovať tak, aby vedel pracovať aj s reálnymi číslami v atribútoch (Gaussian Naive Bayes).
- Hlavná nevýhoda algoritmu je predpoklad nezávislosti slov - keďže počítame pravdepodobnosti každého slova zvlášť, vytvára sa predpoklad, že slová niesú medzi sebou v žiadnom vzťahu.
- Výhodou algoritmu je rýchlosť trénovalia a klasifikácie, vystačí si s malým množstvom trénovacích dát a poradí si aj s veľkým počtom atribútov tak, ako sme to aj videli pri našej textovej klasifikácii.
- Pri vypočítavaní likelihoodu sa môže stať, že niektorá kategória neobsahuje v trénovacej množine žiadne slovo ktoré sa vypočítava v neznámej vzorke a spôsobí nulovú celkovú pravdepodobnosť likelihoodu. Preto sa používa Laplace smoothing – pridanie jedného výskytu slova do všetkých atribútov kategórie.

Po aplikovaní Laplace smoothingu bude náš model spomenutý na začiatku obsahovať v každej kategórii jedno slovo navyše.

Kategória	excellent	headset	mic	doesnt	work	love	this	bad	quality	great	počet vzoriek
Positive	1+1	2+1	1+1	0+1	0+1	1+1	1+1	0+1	0+1	1+1	3
Negative	0+1	0+1	2+1	1+1	1+1	0+1	0+1	1+1	1+1	0+1	2

Celý priebeh výpočtu pravdepodobností oboch kategórií na základe nášho modelu pri vzorke *excellent headset mic quality* bude vyzeráť takto:

$$P(\text{Good} | \text{excellent headset mic quality}) = \frac{1+1}{7+10} \times \frac{2+1}{7+10} \times \frac{1+1}{7+10} \times \frac{0+1}{7+10} \times \frac{3}{5} = 0.000517$$

$$P(\text{Bad} | \text{excellent headset mic quality}) = \frac{0+1}{6+10} \times \frac{0+1}{6+10} \times \frac{2+1}{6+10} \times \frac{1+1}{6+10} \times \frac{2}{5} = 0.000293$$

Výslednou predikciou bude kategória Good s najvyššou vypočítanou pravdepodobnosťou.

C.3 Dotazníky

C.3.1 Strojové učenie s Naivným Bayesovským klasifikátorom a SVM

Tento dotazník sa používal pre tutoriál v Jupyteri, boli v ňom prispunuté otázky z dotazníka Naivného Bayesovského klasifikátora

Strojové učenie s Naivným Bayesovským klasifikátorom a Support Vector Machines

Zadajte Vami vymyslené ID s dĺžkou aspoň 5 znakov

* Povinné

ID riešiteľa: *

Vaša odpoveď

Čo viete o strojovom učení? *

- Neviem čo to je
- Počul/a som o tom
- Viem o čom to je
- Niečo som sa už o tom naučil/a
- Viem ako funguje algoritmus Naivného Bayesovského klasifikátora

Čo je to machine learning model?

- Reprezentácia výberu algoritmu s nastavenými atribútmi
- Reprezentácia výberu algoritmu spolu s poskytnutou trénovacou množinou a nastavenými atribútmi
- Natreňovaný algoritmus s nastavenými atribútmi a hyperparametrami
- Natreňovaný algoritmus s nastavenými atribútmi

Pri binárne klasifikačných algoritnoch sa používajú niektoré prístupy pri klasifikovaní väčšieho počtu kategórií, ktorý prístup je vhodné použiť ak chceme mať čo najpresnejšie výsledky?

- One-vs-One
- One-vs-All

Na čo slúžia podporné vektory

- Na nájdenie oddelovacej čiary s najširšou cestou pri ktorej sú správne kategoricky oddelené všetky trénovacie dátá
- Na nájdenie oddelovacej čiary s najširšou cestou na ktorej nikdy nie je trénovacia vzorka
- Na vytvorenie klasifikačného modelu
- Na nájdenie oddelovacej čiary s najširšou cestou pri ktorej niesú nutne všetky trénovacie dátá správne kategoricky oddelené
- Na nájdenie akejkoľvek čiary správne oddelujúcu kategórie všetkých trénovacích dát

Kedy sa zmení SVM klasifikačný model pri natrénovaní novej vzorky?

- Ak sa vzorka nachádza na ceste
- Zmení sa vždy
- Ak sa vzorka nenachádza na ceste ani na okrajoch cesty
- Ak sa vzorka nachádza na okraji cesty

Kedy nastáva preučenie pri SVM?

- Keď cesty majú malú šírku a všetky trénovacie dátá sú správne oddelené
- Keď nie je nastavený regularizačný parameter pri jednoducho oddeliteľných trénovacích dát
- Výsledky testovacích dát použitých na trénovanie sú výrazne lepšie ako u neznámych dát
- Použitím privysokého pomeru trénovacích dát nad testovacími

Aký tvar by mal SVM oddelovač pri jednorozmerných dátach?

- Bod
- Čiara
- Plocha
- Iné

C.3.2 Naivný Bayesovský klasifikátor

Naivný Bayesovský klasifikátor

Zadajte Vami vymyslené ID s dĺžkou aspoň 5 znakov

* Povinné

ID riešiteľa *

Vaša odpoveď

Čo viete o strojovom učení? *

- Neviem čo to je
- Počul/a som o tom
- Viem o čom to je
- Niečo som sa už o tom naučil/a
- Viem ako funguje algoritmus Naivného Bayesovského klasifikátora

Ktoré z tvrdení je pravdivé o modeli bag-of-words?

- Spočítava frekvencie slov pre každú kategóriu
- Umožňuje hľadať vzťahy medzi slovami
- Spočítava frekvencie slov a vzoriek pre každú kategóriu
- Používa sa na reprezentáciu textu

Na ktorú časť Bayesového pravidla je potrebné použiť predpoklad nezávislostí slov?

- Likelihood
- Prior

Ktoré z uvedených tvrdení o Naivnom Bayesovskom klasifikátore je pravda?

- Používa atribúty, ktoré považuje za nezávislé
- Nie je možné ho modifikovať tak, aby vedel predikovať číselné hodnoty
- Je pomerne úspešný pri vysoko dimenzionálnych dátach (s veľkým počtom atribútov)
- Je pomerne neúspešný pri malom množstve trénovacích dát

Čo je Likelihood v kontexte Naivného Bayesovského klasifikátora?

- Výpočet pravdepodobnosti výskytu slova neznámej vzorky pre určitú kategóriu
- Výpočet pravdepodobností výskytu slova neznámej vzorky pre všetky kategórie
- Výpočet pravdepodobnosti výskytu všetkých slov neznámej vzorky pre určitú kategóriu
- Výpočet pravdepodobností výskytu všetkých slov neznámej vzorky pre všetky kategórie

Ak máme kategóriu s najfrekventovanejším slovom, znamená to, že má najvyššie vypočítanú pravdepodobnosť zo všetkých kategórií vo výpočte Likelihoodu?

- Áno
- Nie
- Nedá sa jednoznačne odpovedať

Čo je Prior v kontexte Naivného Bayesovského klasifikátora?

- Pravdepodobnosti výberu trénovacej vzorky s určitou kategóriou pre všetky kategórie
- Pravdepodobnosť výberu trénovacej vzorky s určitou kategóriou
- Pravdepodobnosť výberu testovacej vzorky so správnou kategóriou
- Pravdepodobnosť výberu kategórie zo všetkých kategórií

Ktoré z uvedených tvrdení o Prioroch je pravda?

- Rôznorodosť počtov vzoriek v kategóriách trénovacej množiny je najvplyvnejším faktorom na klasifikáciu
- Počet výpočtov sa rovná počtu možných kategórií na klasifikovanie
- Čím viac atribútov sa používa pri klasifikácii, tým sa zmenšuje jeho vplyv
- Čím väčší počet možných kategórií pre klasifikáciu, tým má väčší vplyv

Z nasledujúceho obrázka je pri klasifikácii potrebné vypočítavať $P(B)$ pri klasifikácii? (pravdepodobnosť výberu určitého slova vo vzorke)

$$P(A | B) = \frac{P(B | A) \times P(A)}{P(B)}$$

- Áno
- Nie

Kedy nie je potrebné použiť Laplace smoothing?

- Keď všetky kategórie v trénovacej množine majú rovnaký počet slov
- Keď všetky kategórie v trénovacej množine majú počty rôznych použitých slov rovnaké
- Keď všetky kategórie v trénovacej množine majú nenulové hodnoty atribútov
- Keď všetky kategórie v trénovacej množine majú rovnaké atribúty

Čo sa vypočítava prostredníctvom jedného výpočtu Bayesového pravidla?

- Pravdepodobnosť neznámej vzorky patriacej do určitej kategórie
- Pravdepodobnosti neznámej vzorky patriacej do každej kategórie
- Pravdepodobnosti všetkých neznámych vzoriek patriacich do každej kategórie
- Pravdepodobnosť všetkých neznámych vzoriek patriacich do určitej kategórie
- Posterior

Aké sú dôsledky krátkych textov, ktoré sa idú klasifikovať?

- Vplyv Likelihoodu sa zmenšuje, vplyv Prioru sa zvyšuje
- Vplyv Likelihoodu sa zvyšuje, vplyv Prioru sa zvyšuje
- Vplyv Likelihoodu sa zmenšuje, vplyv Prioru sa zmenšuje
- vplyv Likelihoodu sa zvyšuje, vplyv Prioru sa zmenšuje

D Obsah elektronického média

/implementacia

- implementácia jupyter notebooku a dashu spolu s používanými modulmi

/implementacia/datasets

- prednačítané datasety pomocou pickle súborov a jeden predspracovaný dataset

/implementacia/images

- obrázok použitý v Dashi

/dokument

- práca vo formáte pdf