VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ANALÝZA DAT NA SOCIÁLNÍCH SÍTÍCH S VYUŽITÍM DOLOVÁNÍ DAT

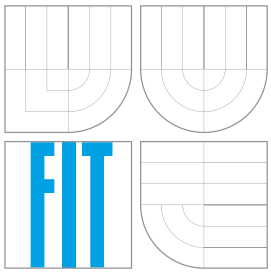
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAN SEDLÁK

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ANALÝZA DAT NA SOCIÁLNÍCH SÍTÍCH S VYUŽITÍM DOLOVÁNÍ DAT

ANALYSIS OF DATA ON SOCIAL NETWORKS BASED ON DATA MINING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAN SEDLÁK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VLADIMÍR BARTÍK, Ph.D.

BRNO 2015

Abstrakt

Tato diplomová práce se zabývá dolováním z dat v prostředí sociálních sítí. Představuje dolování z dat jako takové a jeho možné využití při dolování ze sociálních sítí. Dále se zabývá analýzou aplikačních rozhraní sociálních sítí Facebook, Twitter, Google+, LinkedIn a GitHub z pohledu dolování z dat. Představuje implementaci aplikace, která slouží pro stažení datasetu ze serveru GitHub, zabývá se experimenty se získaným datasetem a návrhem a implementací aplikace, která provádí datovou analýzu míry budoucí aktivity projektu.

Abstract

This thesis deals with data mining on social networks. It introduces data mining itself and its utilization on data analysis on social networking services. It analyses APIs of Facebook, Twitter, Google+, LinkedIn and GitHub with respect to data mining. It presents implementation of application for downloading dataset from GitHub and it deals with experiments with obtained dataset. Finally, it introduces design and implementation of application that analyses future project activity.

Klíčová slova

data mining, dolování dat, sociální sítě, GitHub, strojové učení

Keywords

data mining, social networks, GitHub, machine learning

Citace

Jan Sedlák: Analýza dat na sociálních sítích s využitím dolování dat, diplomová práce, Brno, FIT VUT v Brně, 2015

Analýza dat na sociálních sítích s využitím dolování dat

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana doktora Vladimíra Bartíka.

.....
Jan Sedlák
25. května 2015

Poděkování

Chtěl bych poděkovat panu doktoru Vladimíru Bartíkovi za poskytnutí odborné pomoci.

Obsah

Kapitola 1

Úvod

Dle dobových statistik obsahoval Internet v roce 2010 1,2 Zettabytů dat [27]. Dle dalších statistik stouplo množství dat na Internetu do roku 2013 na 1 Yottabyte, z toho 672 Exabytů bylo veřejně dostupných na více než 14,3 trilionech webových stránkách[1] [15]. Čtyři z největších internetových firem, Google, Amazon, Microsoft a Facebook souhrnně vlastnili 1 200 Petabytů dat [22]. Z infografiky firmy Domo z roku 2012 vyplývá, že každou minutu každého dne nahrají uživatelé Internetu na server Youtube dohromady 48 hodin videí, pošlou dohromady 204 166 667 e-mailů, sdílí dohromady 684 478 příspěvků na serveru Facebook a napíší dohromady 100 000 tweetů na serveru Twitter [18].

Z těchto statistik je zřejmé, že v průběhu posledních několika let se hlavní úkol těch, kteří s daty pracují, přesunul od jejich získávání k jejich třídění, čištění a zpracování. Tyto úkoly daly vzniknout novým datovým disciplínám. Zatímco prací s velkým objemem dat se zabývá obor *big data*, získávání znalostí z těchto dat se zabývá *data mining*, dolování dat, správněji dolování z dat.

Nejprve je důležité pochopit, jaký je rozdíl mezi daty a informacemi. Data jsou reprezentací informace. Bez dalších znalostí, meta-informací, je někdy těžké tyto data interpretovat a získat tím zpět informaci, která je pro nás užitečná. Řetězec `37.205.11.119` je příklad takovýchto dat. Pokud čtenář těchto dat nebude pokročilým uživatelem počítačových sítí, budou mu tyto data připadat pouze jako změť čísel. Jakmile ale tyto data interpretujeme, získáme tím informaci, jako například že 37.205.11.119 je IP adresa, že je někomu alokovaná, že se o ni stará organizace RIPE a že je přiřazena subjektu, který pochází z České Republiky.

Data mining je disciplína, která používá strojové učení, umělou inteligenci a statistiku pro automatizované rozpoznávání vzorů v datech a vytváření nových dat, která jsou člověkem snáze interpretovatelná a která určitým způsobem popisují data původní. Časté je použití dolování z dat pro získání hlubšího vhledu do dat, hledání asociací mezi ději, které data popisují a hledání závislosti mezi daty.

Sociální síť je platforma, nejčastěji webový portál, který umožňuje sociální interakci mezi lidmi. Může spojovat uživatele na základě jejich zájmu či jejich zázemí. V posledních letech, díky portálům jako je Facebook či Twitter, nabývají sociální sítě na důležitosti - již nejsou výsadou pouze technicky pokročilejších uživatelů.

Tato práce se zabývá analýzou dat ze sociálních sítí za pomocí dolování z dat. Ve druhé kapitole tohoto dokumentu je blíže představeno dolování z dat jako takové - jednotlivé fáze,

[1] „webová stránka“ zde označuje soubor s příponou `.html`. Celkový počet webových stránek ve smyslu jednoznačných doménových jmen je lehce pod jednu miliardu [17]

principy a podobně. Ve třetí kapitole jsou představeny různé sociální sítě, které mohou sloužit jako zdroje dat pro datovou analýzu. Jsou hodnoceny jejich vlastnosti z pohledu získávání dat a výsledného datasetu. Čtvrtá kapitola se zabývá návrhem aplikace, která bude provádět datovou analýzu z jednoho takového zdroje. Pátá kapitola představuje strukturu aplikace, která slouží pro získání datasetu, společně s vlastnostmi získaného datasetu. Šestá kapitola se zabývá experimentováním s různými metodami v nástroji Weka a následně za pomocí knihovny scikit-learn. Sedmá kapitola obsahuje popis implementované webové aplikace, popis použití metod strojového učení v této aplikaci a dále představuje další knihovny a nástroje, které byly při implementaci použity. V závěrečné osmé kapitole se nachází zhodnocení celé práce a návrhy na rozšíření této práce.

Této diplomové práci předcházel semestrální projekt na stejné téma. Z tohoto semestrálního projektu byly převzaty kapitoly 1 až 4, přičemž v kapitolách 1, 2 a 4 došlo k nejrůznějším změnám.

Kapitola 2

Dolování z dat

Dolování z dat, anglicky data mining, je definováno jako proces objevování vzorů v datech [28]. Tento proces musí být plně nebo aspoň částečně automatizovaný. Vzory musí být významné, což znamená že jejich znalost musí uživateli přinést určitou výhodu, nejčastěji ekonomickou. Data mining zahrnuje použití nejrůznějších technik, jakými jsou například statistika, strojové učení, použití datových skladů a získávání informací (v originále *information retrieval*) [14]. Některé publikace zařazují data mining jako jednu z fází procesu, nazývaného získávání znalostí. Těmito fázemi se zabývá zbytek této kapitoly. Při jejich popisu budeme vycházet z publikací *Data Mining: Concepts and Techniques* [14] a *Data Mining: Practical Machine Learning Tools and Techniques* [28].

2.1 Jednotlivé fáze získávání znalostí

Dle publikace *Data Mining: Concepts and Techniques* se proces získávání znalostí dělí na tyto fáze: čištění dat, integrace dat, výběr dat, transformace dat, dolování z dat, vyhodnocování vzorů a prezentace znalostí.

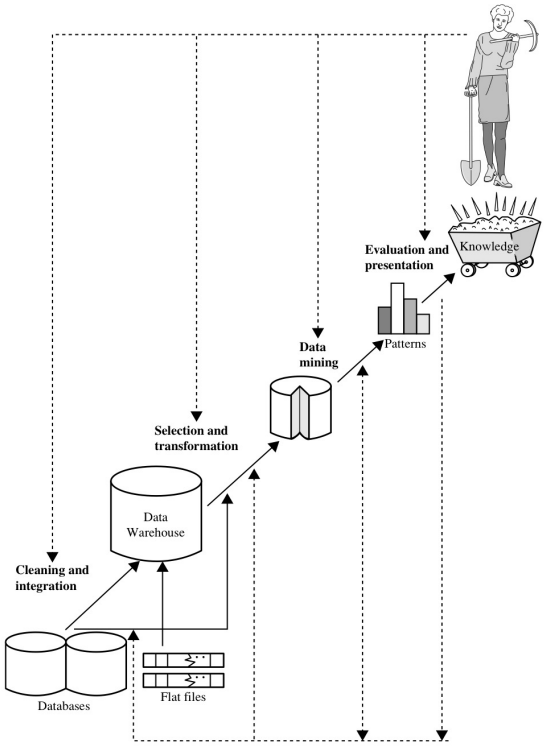
2.1.1 Předzpracování dat a integrace

Fáze předzpracování dat, která zahrnuje jejich čištění, výběr a transformaci, je důležitou součástí procesu získávání znalostí. Jelikož se jedná o data z reálného světa, jsou získaná data téměř vždy nepřesná, zašuměná. Mohou obsahovat nekonzistentní informace, chyby vytvořené při zanášení informací do databáze, problémy s rozdílnými formáty dat, neúplná data. Kvalita dat silně ovlivňuje úspěšnost dolování, je tedy užitečné zdrojová data před jejich použitím předzpracovat. Předzpracování může být ruční nebo automatizované.

Mezi úlohy předzpracování patří odstranění nedefinovaných hodnot, analýza (a případné odstranění) odlehlých hodnot, analýza rozsahů atributů, normalizace dat a dále může být vhodné atributy před samotnou aplikací strojového učení převést na jiné datové typy, příkladem je diskretizace dat. Pro redukci dimenze dat se používá analýza hlavních komponent, PCA, případně výběr podmnožiny atributů na základě informačního zisku.

Pro účely bližšího vhledu do zdrojových dat je použita statistika. Můžeme například zkoumat vlastnosti hodnot atributu pomocí průměru, mediánu, získat modus dat. Můžeme redukovat množství atributů pomocí analýzy hlavních komponent. Tyto informace nám často poskytnou nápovědu, jak k datům přistupovat ve fázi dolování.

Integrací dat myslíme spojení několika zdrojů dohromady. Integrace přináší další problémy, které musíme vyřešit ve fázi předzpracování, jako například redundanci. Hlavním



Obrázek 2.1: Fáze procesu získávání znalostí [14]

důvodem předzpracování je zjednodušení další fáze, dolování z dat.

2.1.2 Dolování z dat

Dolování z dat jakožto součást získávání znalostí zaštiťuje několik strategií, přičemž důležitým úkolem datového analytika je výběr strategie. Mezi tyto strategie patří použití statistických metod, vytváření komplexních dotazů na databázi, použití prvků vizualizace či použití strojového učení.

V odborné literatuře se definice strojového učení překrývá s definicí dolování z dat. Pro účely této práce je strojové učení postupem vytvoření modelu ze známých dat, který je následně použit pro získávání informací o jiných datech s neznámými vlastnostmi, predikci.

Členění jednotlivých metod strojového učení je závislé na charakteru dolované informace. Obvyklé dělení metod je následující:

klasifikace je použita v situaci, kdy se snažíme z dostupných dat získat hodnotu atributu, ten přitom může nabývat jedné hodnoty z výčtu. U dat, které jsou použity pro učení, je nutné, aby hodnota tohoto atributu byla předem známá (tzv. *učení s učitelem*). Mezi metody pro klasifikaci patří vytváření rozhodovacích stromů, naivní bayesův klasifikátor (v originále *naïve Bayes classifier*), neuronové sítě či SVM.

shlukování je použito, pokud předem neznáme třídu, podle které bychom chtěli data rozdělit. Algoritmy, provádějící shlukování, se snaží rozdělit data do skupin tak, aby variabilita dat v rámci skupiny byla co nejnižší, zatímco rozdíly dat mezi skupinami co nejvyšší. Díky tomuto postupu můžeme zjistit, která data mají něco společného,

aniž bychom předem věděli, do jakých skupin chceme vzorek dělit. Mezi algoritmy pro shlukování patří algoritmu DBSCAN či k-means.

asociace slouží pro vytváření asociačních pravidel. V zadaném vzorku hledá data, která se často vyskytují společně, tak zvané *frekventované množiny*. Kromě analýzy nákupního košíku se dolování asociací používá také v bioinformatice nebo v různých metodikách měření oblíbenosti (například oblíbenost webových stránek). Mezi časté metody pro dolování asociací patří například algoritmus Apriori či FP-strom.

regrese neboli regresní analýza je podobná klasifikaci tím způsobem, že se pomocí strojového učení snažíme odhadnout hodnotu atributu. Na rozdíl od klasifikace však odhadujeme hodnotu spojitou. Lze říci, že úkolem regresní analýzy je najít funkci, která popisuje vztah mezi nezávislými proměnnými (to jsou hodnoty atributů dat) a závislou proměnnou (to je hodnota, kterou se snažíme předpovědět). Existuje mnoho různých metod regresní analýzy, například lineární regrese, Lasso, SVR či aditivní regrese.

detekce odlehlých hodnot byla zmíněna již v kapitole o předzpracování dat. Mnohdy však odlehlé hodnoty nejsou chybou v datech, ale naopak cennou informací, kterou chceme získat. Detekcí anomálií v počítačových sítích jsme schopni včas odhalit hrozbu. Analýza odlehlých hodnot v bankovních převodech může ukázat na rozsáhlé daňové úniky. Pro všechny tyto úlohy jsou vhodné metody, které se používají pro detekci odlehlých hodnot, jako jsou SVM či metoda k-nejbližších sousedů (v originále *k-nearest neighbours*).

Speciálním případem strojového učení je NLP, zpracování přirozeného jazyka. Příkladem takovéto úlohy je pochopení tématu z textu či *sentiment analysis*, analýza emoční nálady textu.

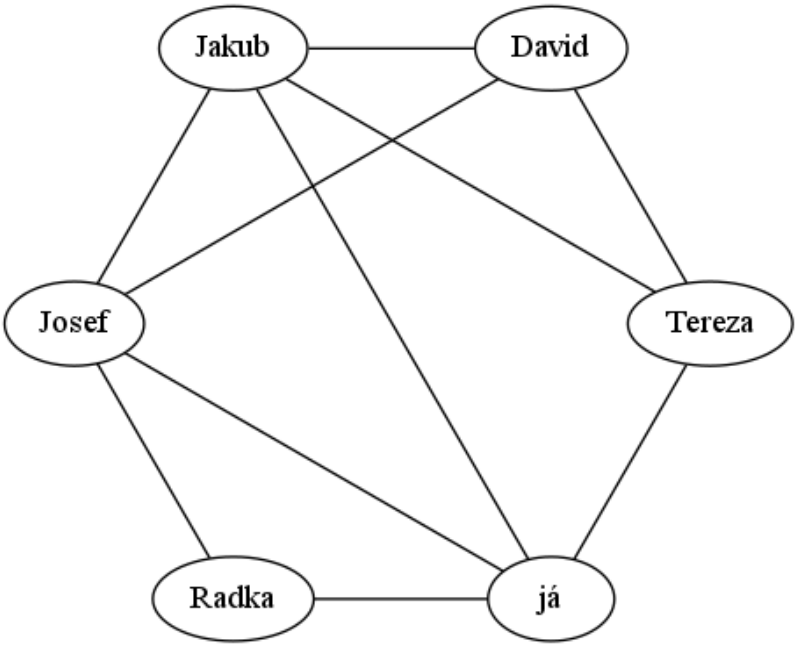
Dolování dat ze sociálních sítí

Při dolování z dat, které byly získány ze sociálních sítí, používáme podobné algoritmy jako při dolování z jakýchkoliv jiných dat. Kniha *Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More* [23] uvádí příklady různých druhů dolování v různých sociálních sítích - frekvenční analýzu tweetů na Twitteru, prozkoumávání sociálního grafu na Facebooku, shlukování spojení na LinkedIn, aplikace NLP na příspěvky Google+ a prozkoumávání grafu zájmů na serveru GitHub. Jelikož je graf zájmu velmi podobný sociálnímu grafu a samotný sociální graf popisuje datovou strukturu sociálních sítí v oboru dolování z dat, prozkoumáme pojem sociální sítě z pohledu dolování z dat trochu blíže.

sociální síť z pohledu dolování z dat je sociální struktura, reprezentovaná sociálním grafem:

$$G = (V, E) \tag{2.1}$$

kde V označuje množinu vrcholů, které reprezentují jedince, a $E \subseteq V \times V$ je množina spojení vrcholů, přičemž spojení reprezentuje přátelství, společný zájem, spolupráci atp. V určitých případech mohou hrany též nést určitou váhu, která může určovat například počet společně vypracovaných prací.



Obrázek 2.2: Příklad sociálního grafu

Nad sociálním grafem lze provádět shlukování, čímž získáme skupiny lidí s podobnými zájmy a můžeme např. lépe cílit reklamu. Častou operací je také měření podobnosti vrcholů v sociálním grafu pomocí algoritmu SimRank.

Algoritmus SimRank

Algoritmus SimRank slouží pro měření podobnosti mezi vrcholy v sociálním grafu. SimRank vrací pro zadané dva vrcholy hodnotu od 0 do 1, přičemž 1 znamená identitu (oba vrcholy jsou jeden a ten samý), 0 znamená že jsou vrcholy naprosto rozdílné. Je založen na myšlence, že dva vrcholy v grafu jsou podobné, pokud mají stejné okolí. Mějme graf z definice 2.1, dále definujme:

$$I(v) = \{u|(u,v) \in E\} \tag{2.2}$$

jako množinu vstupního sousedství jedince (v originále *individual in-neighborhood*) vrcholu v a

$$O(v) = \{w|(v,w) \in E\} \tag{2.3}$$

jako množinu výstupního sousedství jedince (v originále *individual out-neighborhood*) vrcholu v. Hodnota funkce SimRank pro vrcholy u, v je pak definována jako:

$$s(u,v) = \frac{C}{|I(u)||I(v)|} \sum_{x \in I(u)} \sum_{y \in I(v)} s(x,y) \tag{2.4}$$

kde C je konstanta od 0 do 1. Dále specifikujeme, že pokud je množina I vstupního sousedství jedince jednoho z vrcholů prázdná množina (vrchol nemá žádnou vstupní hranu), je hodnota s algoritmu SimRank pro tyto dva vrcholy nulová. Parametr C označuje míru rozpadu při šíření hodnoty podobnosti mezi vrcholy.

Jedná se o iterativní algoritmus. Hodnoty podobností vrcholů jsou přepočítávány, dokud není nalezen pevný bod funkce, dokud se výstupní hodnota funkce mění. Na počátku algoritmu je hodnota s podobnosti nastavena na 0 (pokud se jedná o různé vrcholy), nebo 1 (pokud se jedná o stejné vrcholy).

2.1.3 Interpretace výsledků a jejich prezentace

Poslední, avšak neméně důležitou částí procesu získávání znalostí je interpretace a prezentace výsledků dolování. V této fázi je důležité vyhodnotit úspěšnost použité metody, přičemž toto hodnocení je často již součástí samotného dolování. Hodnocení úspěšnosti se liší podle použitých metod. Pro metodu regresní analýzy používáme hodnotu mean squared error, MSE. Pokud hodnotíme úspěšnost klasifikace a pokud máme omezený soubor vstupních dat pro učení, provedeme tak zvanou křížovou validaci (v originále *cross-validation*), kdy rozdělíme vstupní data do k skupin a následně $k$krát provedeme „zadržení“ části dat a naučení vytvořeného modelu na zbylých $k-1$ skupinách dat. Po té použijeme zadrženou skupinu pro vyhodnocení jednoho kroku úspěšnosti - pozorujeme, pro kolik hodnot byl odhad správný a pro kolik byl chybný.

Nejčastěji používanou hodnotou, která udává velikost chyby při predikci spojité hodnoty, je *mean-squared error, MSE* [28]. MSE, udávající rozptyl hodnot, se spočítá jako:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2 \tag{2.5}$$

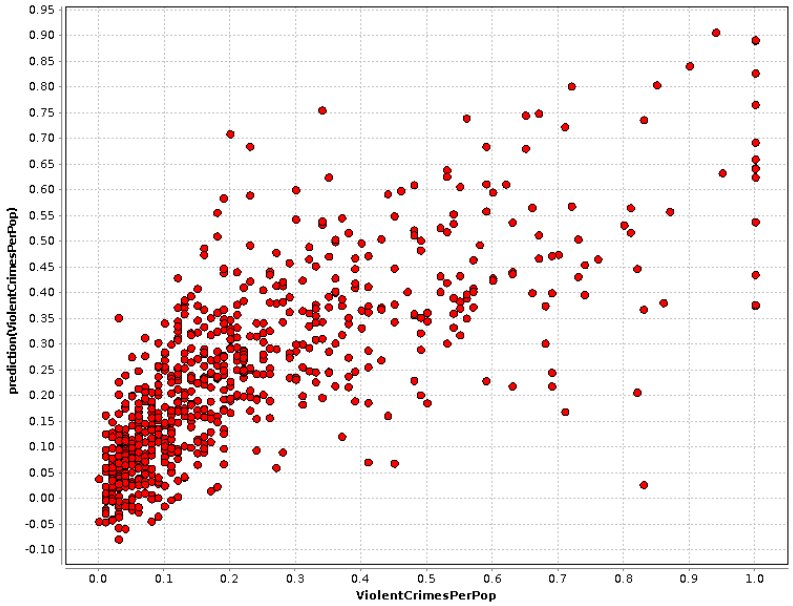
kde n je počet vzorků, $\hat{Y}_i$ je hodnota predikce výstupní hodnoty pro vzorek i a Y_i je opravdová výstupní hodnota pro vzorek i. Zjednodušeně řečeno, MSE vrací průměr mocnin všech chyb. Hodnota MSE se často převádí na *root-mean-squared error*, RMSE, které představuje odchylku hodnot a které se spočítá jako $\sqrt{MSE}$. RMSE má tu výhodu, že je ve stejných jednotkách, jako původní data.

MSE a RMSE mají tu nevýhodu, že jejich hodnota je závislá na rozmezí predikovaných hodnot. Například pokud náš model predikuje hodnoty od 0 do 100, znamená RMSE 13 výrazně lepší výsledek, než stejné RMSE při predikci hodnot od 0 do 1. Z tohoto důvodu byly zavedeny *relativní chyby*, jako je například *root relative-squared error*, RRSE. RRSE se spočítá jako:

$$RRSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y}_i)^2}} \tag{2.6}$$

kde $\bar{Y}_i$ je definováno jako průměr výstupních hodnot testovacího vektoru. Jedná se vlastně o odmocninu podílu MSE našeho modelu a MSE modelu, který pro každou hodnotu předpoví jako výstupní hodnotu průměr naučených hodnot. Často se tato hodnota násobí číslem 100 a pak je udávaná v procentech. Například $RRSE = 100\%$ znamená, že je náš prediktor stejně úspěšný, jak kdybychom pro každý vstupní vektor vrátili průměr výstupních hodnot, které byly použity pro naučení. Hodnotu RRSE se budeme snažit dostat pod 100 % (aby námi použitý model vůbec dával smysl) a obecně platí, že menší RRSE znamená vyšší přesnost.

Poslední metodou, kterou budeme používat pro odhad chyb je Pearsonův korelační koeficient ρ. Korelační koeficient se počítá pro dva stejně dlouhé vektory hodnot a udává



Obrázek 2.3: Vizualizace úspěšnosti regresní analýzy pomocí rozptylového grafu

míru lineární závislosti hodnot prvního vektoru na hodnotách druhého vektoru. $\rho = -1$ znamená, že jsou data silně záporně korelovaná (s rostoucími hodnotami prvního vektoru klesají hodnoty druhého vektoru), $\rho = 0$ znamená, že data nejsou vůbec korelovaná a nakonec $\rho = 1$ znamená, že jsou data silně pozitivně korelovaná (s rostoucí hodnotou v prvním vektoru roste hodnota v druhém vektoru).

Jednotlivé metody se také liší mírou, do které jsme schopni z vytvořeného naučeného modelu získat pravidla pro predikci čitelná uživateli. Zatímco po vytvoření rozhodovacích stromů jsou pravidla jednoduše prezentovatelná, při použití neuronových sítí je vytvoření čitelných a jednoduchých pravidel téměř nemožné.

Poslední částí získávání znalostí je prezentace výsledků uživateli. Pro tento účel je vhodné použít prvky vizualizace - barevně označit významné hodnoty, zvýraznit odchylky. Existuje mnoho různých způsobů vizualizace dat, výběr vhodného grafu je důležitou součástí prezentace dat. Příklad použití rozptylového grafu pro vizualizaci úspěšnosti je na obrázku 2.3.

Kapitola 3

Sociální sítě

E-mail (1971), bulletin board systémy (1978), chat (1980), MUD (1980), internetová fóra (1984), IRC (1988), rozmach on-line her (devadesátá léta), WEB 2.0 (od 1999) a obsah generovaný uživatelem, virtuální reality (například Second Life, 2003), poslední dobou sociální sítě jako je Myspace (2003), Facebook (2004) a Twitter (2006). Již od prvopočátku se lidé snaží najít další a další způsoby, jak z Internetu vytvořit sociální platformu, kam by mohli zaznamenávat všechny své životní události, vkládat svůj vlastní obsah a ten pak sdílet se svými přáteli i celým světem. Tento fakt dělá ze sociálních sítí, fenoménu posledních let, jeden z nejplodnějších zdrojů dat pro datovou analýzu. Typické je také vytváření sociálních sítí pro specializovanější zájmové skupiny jako je například Pinterest, Instagram, Last.fm či GitHub.

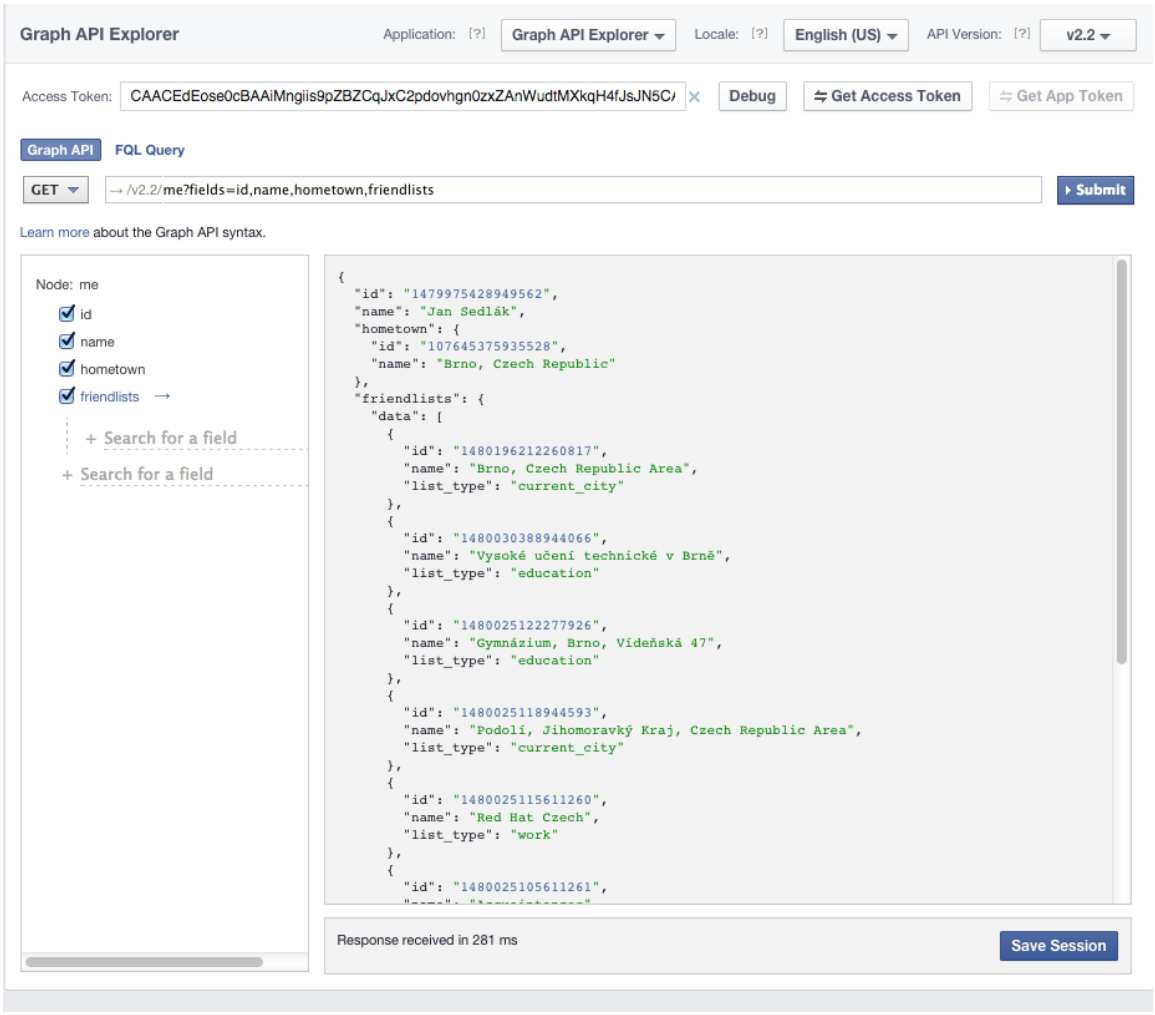
V této kapitole je představeno pět sociálních sítí, které poskytují vhodná data pro dolování spolu se zhodnocením těchto sítí jako zdrojů dat. Pro analýzu programového rozhraní sociálních sítí byla použita kniha *Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More* [23].

3.1 Facebook

Facebook je, spolu s 1,35 miliardou aktivních uživatelů [11], jednou z nejpopulárnějších a nejznámějších sociálních sítí. Jedná se o univerzální sociální síť, která uživatelům umožňuje spojit se se svými přáteli a se svou rodinou, navzájem sdílet příspěvky, fotografie, videa, psát si privátní zprávy (IM), plánovat události či sledovat veřejné příspěvky známých osobností. Pomocí systému „Like“ (česky „Líbí se mi“) uživatelé sdílí svoje zájmy s ostatními. Facebook také umožňuje sdružování do zájmových skupin.

Z pohledu množství dat je Facebook ideálním zdrojem pro data mining. Jedná se o síť, kterou aktivně využívá více než jedna sedmina světové populace. Facebook může být zdrojem multimediální dat jako jsou fotky či videa, textových dat pro zpracování přirozeného jazyka a *sentiment analysis* či pro vytváření sociálního grafu. Díky *Facebook platform* má téměř každá internetová stránka svůj profil na Facebooku spolu s informacemi o počtu sdílení či komentářů k této stránce. Velké množství pozic, které Facebook nabízí v oblasti datové analytiky [10, 19] prozrazuje, že Facebook klade na dolování z dat velký důraz.

Hlavním programovým rozhraním je *Graph API*. Jedná se o REST API, které umožňuje procházet sociálním grafem účtu. Výhodou je jeho jednoduchá použitelnost, vysoká granularita nastavení oprávnění či nástroj Graph API Explorer, na obrázku 3.1, který umožňuje jednoduše prozkoumávat API Facebooku.



Obrázek 3.1: Facebook Graph API Explorer

Možností pro dolování je několik. Sociální graf, vytvoření nad daty z Facebooku, by umožňoval vytváření návrhů hudebních skupin, webových stránek či nových přátel na základě dat získaných od přátel uživatele. Podobné informace by šly získat pomocí shlukové analýzy. Zpracování přirozeného jazyka z textových příspěvků by posloužilo například pro automatickou detekci sociálních vztahů. Nakonec, z fotografií a videí by na základě datové analýzy mohlo být užitečné získat polohu pořízení či počet a identitu osob. Je však také nutno říci, že většinou této funkcionality již Facebook disponuje.

Nevýhody

Hlavní nevýhodou a také důvodem, proč ve skutečnosti není téměř možné Facebook pro dolování dat použít, jsou změny v bezpečnosti v API verze 2.0 [20]. Ve stručnosti, uživatel Facebooku musí ručně odsouhlasit že daná aplikace může získat jeho informace a to včetně jména a příjmení. V praxi to pak znamená, že aplikace třetích stran nemůže získat ani seznam přátel uživatele, pod kterým je spuštěna. Každý uživatel, který by se měl objevit v tomto seznamu, by totiž musel ručně tuto aplikaci autorizovat. To stejné by potom platilo u seznamu přátel tohoto uživatele. Je proto téměř nemožné vytvářet sociální graf, jelikož

by každý účastník, který by se v tomto grafu objevil, musel nejprve odsouhlasit přístup aplikaci, která by sociální graf vytvářela. Tato změna má sice pozitivní dopad na soukromí uživatelů, ale silně znevýhodňuje aplikace třetích stran z pohledu dolování z dat.

3.2 Twitter

Twitter je sociální síť pro zveřejňování kratších zpráv o délce maximálně 140 znaků. Zprávy jsou veřejné, což Twitter dělá vhodným zdrojem dat pro data mining. Uživatel může „sledovat“ jiné uživatele - na rozdíl od Facebooku bývá tato sociální interakce jednostranná. Omezení na délku zprávy má za následek stručnost příspěvků a také relativní jednoduchost jejich vytváření. Dá se proto očekávat, že uživatel Twitteru bude psát příspěvky na Twitter častěji, než by psal podobné příspěvky na jiné druhy webových stránek (na blog apod.). Dle výroční zprávy tweetuje 284 milionů uživatelů [26], mezi nimi i významné osobnosti jako je Barack Obama či papež František. Dalším rysem příspěvků na Twitteru jsou tzv. *hashtagy*. Hashtag je slovo ve formátu `#slovo`, kde `slovo` je jednoslovné vyjádření tématu, kterým se příspěvek zabývá. Jedná se de facto o druh meta-informace o příspěvku. Autor pomocí hashtagu může vyjádřit téma příspěvku, jeho náladu či událost, ke které se příspěvek vztahuje. Čtenář pak může vyhledat všechny příspěvky se stejným hashtagem a zjistit tak názory lidí na dané téma.

Mezi problémy, řešené pomocí dolování dat nad daty z Twitteru patří například automatické navrhování hashtagů k příspěvku či analýza tzv. *influencerů*, čili uživatelů, kteří svými příspěvky ovlivňují velké množství lidí. Analýza sledovaných uživatelů Twitter účtu je zobrazena na obrázku 3.2.

Nevýhody

Nevýhodou dolování z Twitteru oproti dolování ze sítě Facebook je menší počet uživatelů. Zvláště v České Republice se jedná o nepříliš rozšířenou sociální síť, kterou používají většinou technicky vzdělanější uživatelé. Lokální příspěvky mají tedy málo rozmanitou zdrojovou i cílovou skupinu. Při výběru zdroje dat musíme také zohlednit, že byl Twitter zvolen jako zdroj dat již ve dvou předchozích diplomových pracích.

3.3 LinkedIn

LinkedIn je úzce zaměřenou sociální sítí. Jedná se o interaktivní životopis doplněný o sociální interakce ve formě pracovních spojení. Uživatel sítě může na svůj profil umístit pracovní zkušenosti, informace o účasti na různých projektech, svých dovednostech a svém vzdělání. Jiní uživatelé pak mohou tyto informace potvrzovat. Jedná se tedy o sociální síť pro zaměstnance a zaměstnavatele, pro profesní networking. Profil na LinkedIn má přibližně 300 milionů uživatelů [21].

Úzké zaměření této sociální sítě také specifikuje možnosti dolování z dat. Nabízí se vytváření sociálního grafu a pak analýza influencerů, hledání vhodných zaměstnanců na zadanou pozici, hledání vhodné náhrady za konkrétního zaměstnance.

Nevýhody

Hlavní nevýhodou datové analýzy pomocí programového rozhraní na LinkedIn jsou omezení, která jsou na API kladena. LinkedIn povolí aplikacím třetí stran přístup pouze ke spojením

Obrázek 3.2: Statistiky z Twitter účtu

prvního stupně. To znamená, že program může pomocí API získat seznam přímých spojení uživatele ale už ne spojení těchto uživatelů, což značně komplikuje vytváření sociálního grafu.

Dalším omezením je omezení hledání. Používat API pro vyhledávání na LinkedIn mohou pouze autorizované aplikace. Autorizace vyžaduje podání oficiální žádost, kontrolu autorizované aplikace a zakazuje využívat API pro účely dolování z dat.

3.4 Google+

Google+ je, jak je z názvu zřejmé, sociální síť od firmy Google. Jedná se o univerzální sociální síť, ve které uživatel sdílí příspěvky, fotky, videa či odkazy se svými přáteli. Přátele může přiřazovat do kruhů a příspěvky pak sdílet s konkrétním kruhem, což zvyšuje soukromí. Charakteristika sítě Google+ je podobná charakteristice sítě Facebook. Určitou výhodou by mohlo být propojení Google+ s ostatními službami firmy Google jako je Google Picasa, Google Hangouts nebo Youtube. Stejně tak možnosti dolování z dat jsou obdobné možnostem u sítě Facebook.

Nevýhody

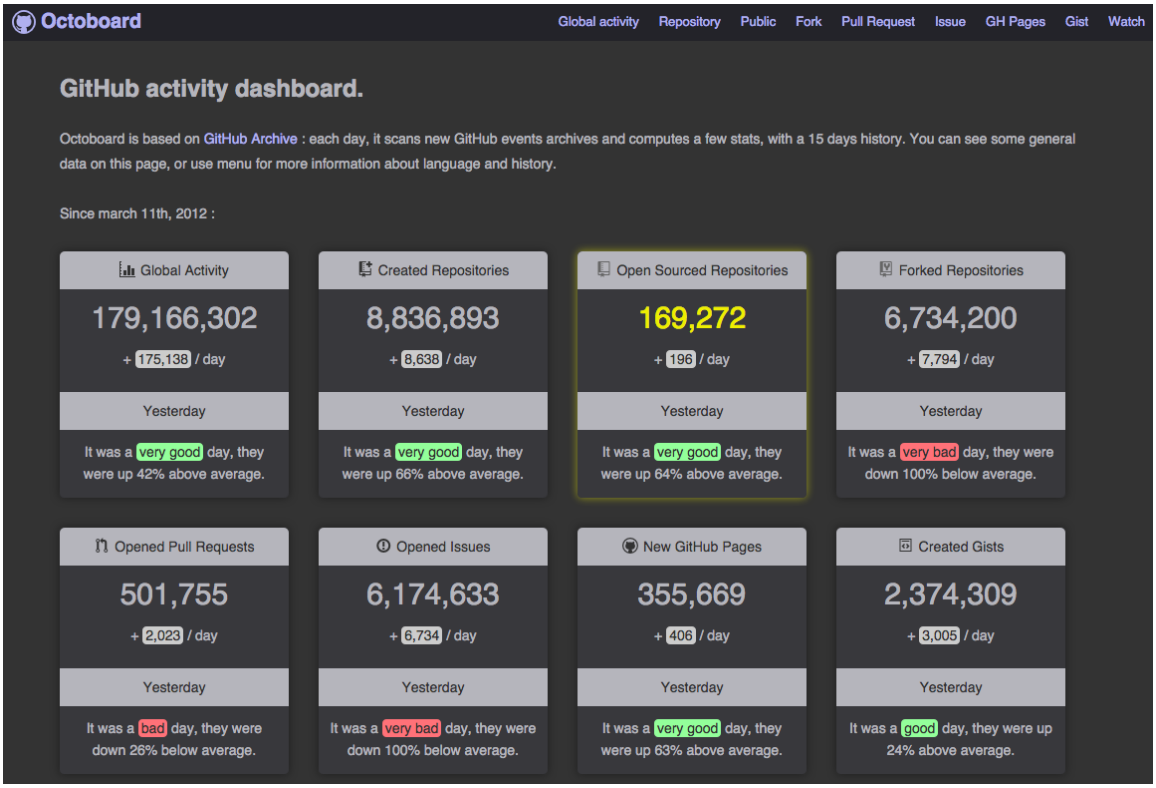
Jednou z hlavních nevýhod sítě Google+ je omezení počtu dotazů na programové rozhraní. Určité omezení mají i všechny ostatní sítě, ale Google+ je nejpřísnější - povoluje pouze 10 000 dotazů za den, přičemž oficiální dokumentace upozorňuje, že není garantována ani tato kvóta. Další nevýhodou je nemožnost pomocí API zjistit který uživatel patří do kterého kruhu - pouze zda-li ho osoba má nebo nemá v přátelích. Tyto vlastnosti odsuzují použití API pouze pro účely přístupu k síti Google+ ale ne pro účely dolování z dat.

Kontroverzní otázkou je také počet uživatelů této sítě. Oficiální zdroje uvádí 540 milionů aktivních uživatelů v roce 2013 [13], ale do tohoto počtu jsou zahrnuti také uživatelé, kteří například komentují na serveru Youtube. Objevují se názory, že skutečný počet uživatelů této sítě může být nižší [7].

3.5 GitHub

GitHub, jenž nepatří mezi typické sociální sítě, poskytuje svým uživatelům repozitáře pro systém správy zdrojového kódu Git. Git umožňuje správu zdrojových kódů projektů, jejich jednoduchou revizi, řešení konfliktů při souběžné práci na stejném kódu a uchovává veškeré změny, které byly vytvořeny v průběhu vývoje. Služba GitHub pak nad tento systém přidává sociální vrstvu - komentáře nad změnami zdrojového kódu, možnost diskuze nad chybami, Wiki stránky, označování populárních repozitářů hvězdou (obdoba „Like“ z Facebooku), možnost „větvení“ (v originále *fork*) projektu (vytvoření kopie repozitáře, ke kterému má přístup ten uživatel, který větvení provedl) a následné vytváření žádosti o zavedení změn zpět do původního projektu.

GitHub se za poslední roky stal jednou z nejvýznamnějších platforem pro sdílení zdrojového kódu. Zdrojové kódy svých projektů na něm zveřejnily velké firmy jako například Google [4], Facebook [3], Microsoft [5] či státní organizace Spojených států amerických [6]. O tom, že uživatelé přistupují ke GitHubu jako k sociální síti, svědčí jeho zařazení do publikace *Mining the Social Web* nakladatelství O'Reilly [23]. Dolováním z dat z GitHubu jako sociální sítě se také zabývá několik vědeckých prací, ze kterých tato práce vychází [29, 25, 9].



Obrázek 3.3: Octoboard, služba dolující data z GitHubu

Programové rozhraní služby GitHub je přístupnější dolování než API jiných služeb. Služba povoluje 5 000 dotazů za hodinu a samotný GitHub pak od zveřejnění API každoročně pořádá soutěž v data miningu z dat GitHubu [1]. Dva z vítězných projektů jsou zobrazeny na obrázcích 3.3 a 3.4. Dolování dat ze služby GitHub bylo také tématem soutěže, která proběhla na konferenci *The 11th Working Conference on Mining Software Repositories* v Hyderabadu v Indii [2].

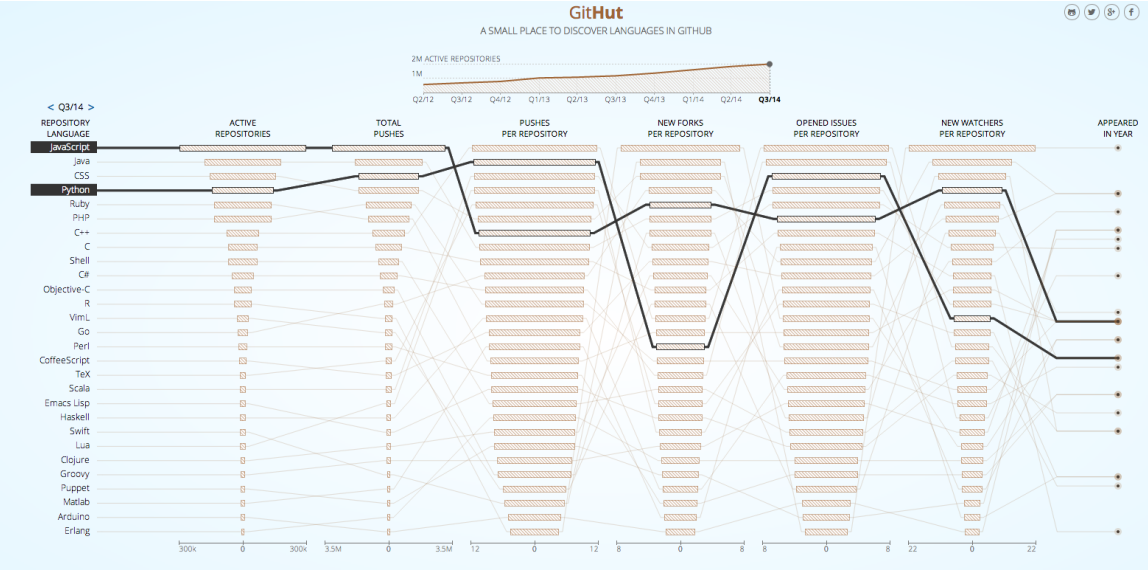
Existuje několik možných témat při dolování dat z GitHubu. Mezi tato témata patří například analýza vlivu způsobu řízení projektu na úspěch projektu [29, 9, 24] či vytváření sociálního grafu [25].

Nevýhody

Mezi nevýhody použití sítě GitHub pro dolování z dat patří velké množství „mrtvých“ repozitářů (takových, které uživatel vytvořil, nahrál do nich zdrojové kódy a následně je opustil) a také možnost získat informace pouze o veřejných repozitářích (počet neveřejných repozitářů na síti GitHub je neznámý). GitHub je také s 8 miliony uživatelů a 18 miliony repozitářů [16] nejmenší z diskutovaných sociálních sítí.

3.6 Společné rysy

Překvapujícím zjištěním byla míra, s jakou většina sociálních sítí střeží data svých uživatelů. Staré verze API však napovídají, že k tomuto uzavření došlo až v posledních letech, zřejmě zároveň se stoupající mírou negativní publicity, kterou sociální sítě získávají ve spojení



Obrázek 3.4: Služba GitHut pro své statistiky používá dataset projektu GitHub Archive

s ochranou osobních údajů. Výjimku tvoří pouze služby GitHub a Twitter, protože pro obě služby je veřejná dostupnost dat klíčová.

Všechny zkoumané služby využívají REST API, což je princip návrhu API, podle kterého klient pro komunikaci používá HTTP či HTTPS, přičemž pro výběr dat se používá URL. Odpověď serveru je nejčastěji ve formátu JSON (JavaScript Object Notation). Pro zabezpečení využívaly zkoumané sociální sítě systém OAuth verze 1.0 či 2.0.

Kapitola 4

Návrh aplikace pro dolování z dat ze sociální sítě GitHub

Na základě výsledků z předcházející kapitoly bylo vybráno, že výsledná aplikace bude provádět datovou analýzu na datech, získaných ze sociální sítě GitHub. Častým problémem při výběru programu či vhodné knihovny programovacího jazyka je otázka budoucí aktivity tohoto projektu. Pokud je v projektu použita knihovna, která již nebude nadále vyvíjena, mohou časem nastat problémy. Nově nalezené chyby v projektu už původní autor neopraví. Při změně API služeb či jiných knihoven, které ten daný projekt používá nedojde k aktualizaci jejich použití, což může mít za následek znefunkčnění celého projektu. Analýza projektu z pohledu aktivity je pak logickým požadavkem.

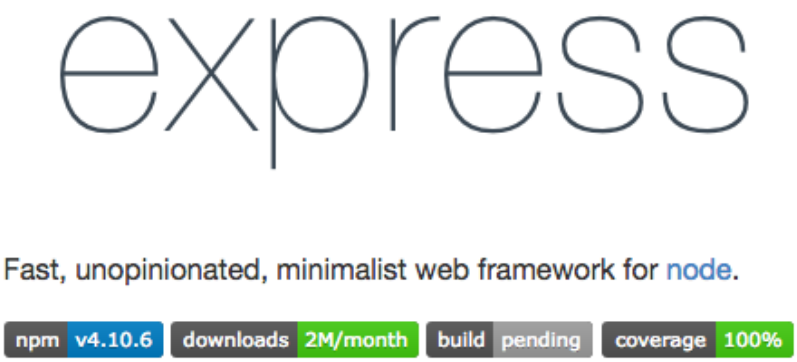
Pro datovou analýzu aktivity projektu je možné použít několik různých zdrojů pro dolování. Jelikož součástí stránky projektu na serveru GitHub je také repozitář zdrojových kódů, je možné analyzovat četnost změn kódu, dále poměr vyřešených chyb ku nevyřešeným chybám, sledovat střední dobu, která uplyne od nahlášení chyby k jejímu opravení, sledovat aktivitu autorů projektu také v jiných projektech, sledovat popularitu repozitáře a podobně.

4.1 Princip fungování a použité technologie

Výsledná aplikace bude webový portál, který po spárování s projektem na serveru GitHub zobrazí různé statistiky týkající se aktivit v jednotlivých rysech vývoje projektu (změny zdrojového kódu, řešení nahlášených chyb) a také sumární hodnotu budoucí aktivity projektu. Jednou ze služeb, kterou bude portál poskytovat, bude odznak, vyjadřující aktivitu projektu. Příklad takových odznaků je na obrázku 4.1.

Aplikace bude implementována v programovacím jazyce Python, pro vytvoření webové služby bude použit framework Django. Dále bude použita knihovna scikit-learn[1], obsahující funkce pro práci s daty a strojové učení. Pro prvotní vytvoření modelu bude potřeba využít rozsáhlého předpřipraveného datasetu, v průběhu života pak bude aplikace používat API serveru GitHub.

[1] `http://scikit-learn.org/`



Obrázek 4.1: Příklad stránky projektu spolu s různými „odznaky“

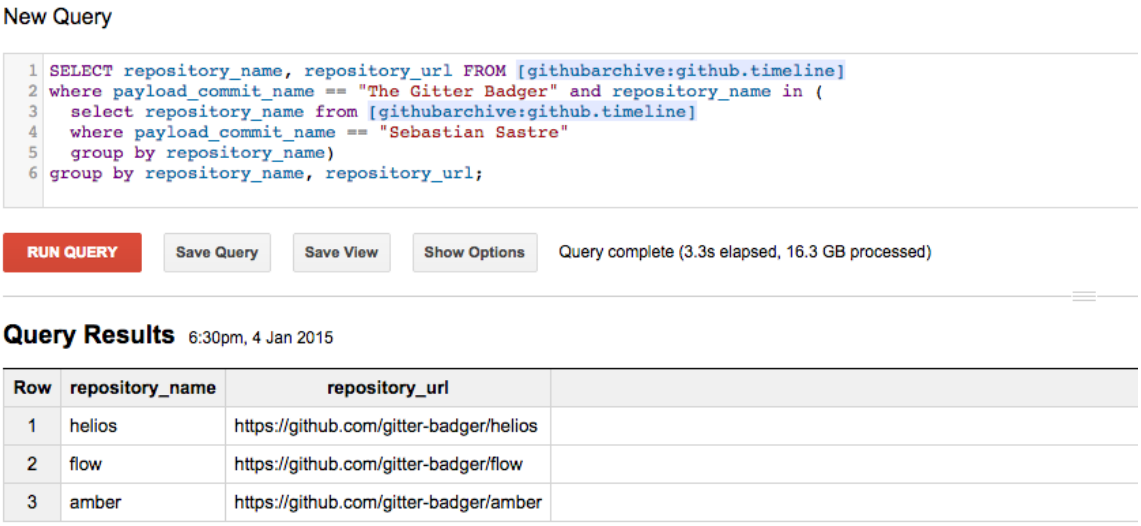
4.2 Zdroje dat pro dolování

Zdrojů pro vytvoření datasetu je několik. Kromě samotného API GitHubu existuje také projekt GitHub Archive[2], který každou hodinu stahuje souhrnná data z GitHubu a poskytuje je přes jednoduché API ve formě komprimovaného JSON. Stáhnutí všech dat z konkrétní hodiny konkrétního dne lze pak jednoduchým dotazem na URL, například `http://data.githubarchive.org/2015-01-01-15.json.gz`. Podobná data, avšak pomocí torrentů, poskytuje projekt GHTorrent[3]. Výhodou těchto projektů je jednoduché získání velkého objemu dat. Nevýhodou je pak fakt, že tyto datasety neobsahují obraz kompletní databáze GitHubu, pouze jeho *Timeline*. Timeline obsahuje sled událostí, které se dějí na serverech GitHubu spolu s inkrementálními informacemi o repozitářích. Jinými slovy, pokud bychom chtěli zjistit stav konkrétního projektu, musíme najít chvíli, kdy v tomto projektu došlo k poslední události (poslední commit či komentář) a součástí této události jsou pak nejnovější údaje o našem repozitáři. Další nevýhodou je objem dat - JSON popisující timeline z jedné hodiny jednoho dne má zhruba 30 MB. Z tohoto důvodu by bylo určitě nutné vybrat pouze menší vzorek dat z tohoto datasetu.

Dalším problémem je změna použití API GitHubu, která proběhla k 1. 1. 2015. Timeline API bylo nahrazeno Events API, které je sice stručnější, ale jednotlivé události neobsahují tolik užitečných dat. Příklad reprezentace jedné události z Events API je v příloze A.2. Příklad reprezentace jiné události z Timeline API je v příloze A.1. Pro vytvoření modelu však budou potřeba kompletní data jednoho repozitáře. K těmto datům bude zřejmě potřeba použít jak data z GitHub Archive, tak přímo API GitHubu. Pro získání dat bude zřejmě nutné použít crawler, který rekurzivně stáhne veškerá data ke konkrétnímu repozitáři.

Alternativním řešením by bylo použití služby Google BigQuery, která již data z GitHub Archive poskytuje [12]. Tato služba umožňuje nahrání velkého objemu dat a poskytuje jazyk (podobný jazyku SQL), kterým je možné se na tato data dotazovat. Příklad použití je na obrázku 4.2. Problémem však je, že se jedná o placenou službu, přičemž částka není fixní, ale mění se od počtu dotazů a množství zpracovaných dat. Nevhodnou formulací dotazů by pak mohla cena za BigQuery dosáhnout astronomických částek.

[2] `http://www.githubarchive.org/`
[3] `http://ghtorrent.org/`



Obrázek 4.2: Použití Google BigQuery

4.3 API sítě GitHub

Síť GitHub poskytuje, kromě samotných webových stránek, také REST API, které vrací hodnoty ve formátu JSON. Omezení na počty požadavků nejsou nijak přísné, anonymní uživatel může zaslat 60 požadavků za hodinu, ale přihlášený uživatel může zaslat až 5 000 požadavků za hodinu. Ačkoliv poskytuje API po přihlášení také možnosti pro vytváření projektů pomocí HTTP metody POST, v této práci toho nebude potřeba využít. Taková část API, která vrací seznam položek (například zaslaných změn, nahlášených chyb apod.) tento seznam stránkuje, a to po 30 položkách na stránku. To znamená, že pokud repozitář obsahuje 90 zaslaných změn, jsou k jejich stáhnutí potřeba tři požadavky GET. Následuje příklad dotazu, na který API vrátí seznam prvních třiceti změn repozitáře:

```
GET /repos/octocat/Hello-World/commits
```

Z dat, která poskytuje GitHub přes své API, mohou být užitečné následující:

events poskytují seznam obecných, veřejných událostí - založení projektu, vytvoření požadavku o zaslání změn apod. Jedná se o sumarizaci všech ostatních, veřejně dostupných informací. Dataset projektu GitHub Archive obsahuje souhrn těchto dat.

starring and watching obsahuje informace o tom, kteří uživatelé označili jaký projekt hvězdou.

commits obsahují informace o jednotlivých zaslaných změnách. Nevýhodou je, že se jednoduchým způsobem nedá zjistit, kolik změn repozitář obsahuje.

issues obsahují seznam nahlášených chyb včetně komentářů k těmto chybám.

pull requests obsahují seznam požadavků o začlenění změn včetně komentářů k těmto požadavkům.

collaborators obsahuje seznam lidí, kteří mají právo pro zápis do repozitáře.

comments obsahuje seznam komentářů ke změnám.

forks obsahuje seznam a data větvení daného repozitáře.

statistics obsahuje seznam statistik o repozitáři. Protože vytváření statistik trvá dlouhou dobu, GitHub je neposkytuje okamžitě ke všem svým repozitářům. Použití těchto statistik ale zřejmě nebude nutné, protože je můžeme zjistit z ostatních dat sami.

Použití REST API je jednoduché (stačí libovolná knihovna pro HTTP), pro využití GitHub API však pro programovací jazyk Python existují také pomocné knihovny. Do výsledného projektu byla vybrána knihovna *githubpy*, která jednotlivé možnosti URL API poskytuje jako metody objektu a výsledky automaticky překládá z JSONu do objektů v Pythonu. Výše zmíněný dotaz pak v kódu Pythonu vypadá jako:

```
commits = gh.repos("octocat")("Hello-World").commits().get()
```

Kapitola 5

Charakteristika datasetu

Pro správný odhad míry budoucí aktivity potřebujeme dataset - soubor příkladů projektů spolu s jejich odpovídajícími hodnotami budoucích aktivit. Pro účely získání datasetu bylo nakonec vybráno přímo API GitHubu. Zajímají nás kompletní historie konkrétních repozitářů, což znesnadňuje použít služby, které poskytují agregovaná data z GitHub Archive. Další nevýhodou je, že se v těchto datech nevyskytují informace o repozitářích, které byly nejprve neveřejné a které byly otevřeny až později. Naštěstí jsou limity API natolik vysoké, že nás nebude přímé použití tohoto API při stahování dat omezovat.

Nejprve je nutné vytvořit program, který tento dataset vytvoří. Tento program bude komunikovat s API GitHubu, ze kterého bude získávat statistiky o náhodně zvolených repozitářích - frekvenci zasílání změn, frekvenci oprav chyb apod. Nakonec spočítá budoucí aktivitu takového projektu. Pro vytvoření přesného modelu potřebujeme velké množství záznamů. Jejich získávání bude trvat pravděpodobně velice dlouhou dobu (v řádech jednotek dní), je proto nutné věnovat zvýšenou pozornost návrhu této části programu.

Kvalita datasetu výrazně ovlivňuje úspěšnost modelu, který se z tohoto datasetu naučí. V případě, že si vytváříme dataset sami, je užitečné použít nástroj na jednoduché a rychlé experimentování s datasetem. K tomuto účelu byl zvolen nástroj Weka[1]. Dle výsledků experimentování s programem Weka pak vybereme metody, které budou použity v knihovně scikit-learn ve výsledném projektu. Tento přístup nám zajistí flexibilitu při získávání datasetu jako i robustnost při implementaci projektu.

5.1 Získání a popis položek datasetu

Za účelem získání datasetu byl vytvořen program *ghaminer*. Jedná se o *crawler*, čili program, který automatizovaně a systematicky prozkoumává web (v tomto případě však prozkoumává JSON API). Program ghaminer generuje náhodná ID, získává informace o repozitářích s danými ID a výsledek zapisuje do souboru ve formátu CSV, který bude sloužit jako dataset. Seznam atributů, které program získává o jednotlivých repozitářích, je v příloze B.

Informace, zda daný repozitář vznikl rozvětvením jiného repozitáře, může být velice cenná - většina větvení totiž vzniká za účelem vytvoření požadavku o začlenění a jakmile jsou tyto začleněny, autor již dále žádné změny do repozitáře nezasílá. Také rozdíl či poměr počtu dní od vzniku repozitáře a od posledního zaslání změny by mohl ukazovat na budoucí aktivitu. Nevýhodou by však mohly být různé zvyklosti frekvence zasílání změn do repozitáře - pokud existuje projekt, kam autoři zasílají změnu jednou za půl roku už

[1] `http://www.cs.waikato.ac.nz/ml/weka/`

posledních několik let, může informace, že od posledního zaslání změny uběhly tři měsíce, ukazovat na aktivní projekt. Pokud však existuje projekt, kam autoři zasílali desítky změn denně, může stejná informace znamenat, že autoři již projekt opustili.

Kvůli těmto případům může být důležité zahrnout do získaných dat informace o průměrných frekvencích, přičemž tyto frekvence se počítají na nejrůznějších atributech za poslední týden, měsíc, půlrok, rok a celkovou dobu života repozitáře. Jedná se o denormalizaci, která je provedena za účelem vystihnutí dynamičnosti vývoje aktivity projektu. Zatímco v původní databázi na serveru GitHub jsou tyto informace uloženy například jako relace mezi zasláním změny a repozitářem, pro naše účely je nutné vytvořit z těchto relací další atributy záznamu. Abychom redukovali počet těchto atributů, program ghaminer ukládá právě pouze charakteristiky (počet, frekvence). Frekvence zasílání změn je definována jako:

$$f_{comm}(t) = \frac{C_{comm}(t)}{(t - t_0)} \tag{5.1}$$

kde $C_{comm}(t)$ je počet zaslaných změn od data prvního zaslání změny do data t a t_0 je datum prvního zaslání změny. Jednotka této veličiny je $změny/den$.

Služba GitHub nabízí projektům systém pro hlášení chyb uživateli. Kdokoliv, kdo objeví chybu v daném projektu, může chybu v tomto systému nahlásit. Autoři pak mohou nad touto chybou vést diskuze a jakmile je chyba opravená, označí ji za vyřízenou. Frekvence nahlášených chyb může ukazovat na zájem uživatelů o tento projekt - čím víc uživatelů, tím větší šance je, že budou chyby v kódu nalezeny a opraveny. Důležitou informací je také procento nahlášených chyb, které se povedlo opravit. Průměrná doba od nahlášení chyby po její opravení ukazuje na zájem autorů starat se o tento projekt.

Požadavky o začlenění jsou způsob, jak mohou jiní uživatelé, kteří nejsou majitelé projektu, do tohoto projektu přispívat. Uživatel, který chce zaslat svoji změnu do nějakého repozitáře, nejdříve vytvoří větvení tohoto repozitáře, v tomto větvení vytvoří svoje změny a následně požádá o začlenění jeho změn zpět do původního repozitáře. Tím se vytvoří tzv. *pull request*, česky „požadavek o začlenění“. Původní autoři pak mohou tento požadavek zkontrolovat, diskutovat o něm s autorem tohoto požadavku a případně tento požadavek začlenit. Aktivní a oblíbené projekty budou mít velký počet požadavků o začlenění - uživatelé budou chtít rozšířit funkcionalitu, popřípadě opravit chyby projektu, který používají. Aktivita autorů pak bude zřejmá z počtu vyřízených požadavků, frekvence komentářů k těmto požadavkům a průměrné doby nutné k vyřízení požadavku.

API služby GitHub také poskytuje informace o všech událostech projektu. Souhrn těchto událostí je zdrojem dat pro GitHub Archive. Do datasetu můžeme tato data zahrnout, mají však dvě nevýhody. První je, že se v tomto přehledu zobrazují události, které se staly až v době, kdy byl projekt otevřen veřejnosti. Druhou nevýhodou je, že dle dokumentace API poskytuje GitHub pouze nejnovějších 300 událostí.

Další cennou informací, která nám může pomoci v odhadnutí budoucí aktivity projektu, je charakter týmu autorů, který na projektu pracuje. Tyto informace se ghaminer snaží zachytit atributy `contrib`. Tyto atributy udávají počet lidí, kteří seřazení sestupně podle počtu celkových změn zaslali 75 %, 50 % a 25 % všech změn a to opět za poslední týden, měsíc, půlrok, rok a celkovou dobu. Například hodnota 1 těchto atributů znamená, že je u tohoto projektu jeden člověk, který je autorem drtivé většiny všech zaslaných změn. Naopak vyšší číslo znamená, že existuje víc autorů projektu, kteří si mezi sebou práci rovnoměrně dělí. Změna této hodnoty pak může napovídat, zda projekt neopouští jeho autoři, případně zda nepřibyli nějací autoři noví. Součástí skupiny těchto atributů jsou také dva počty autorů, jednak těch, kteří u svých změn mají správně nastavené informace

o GitHub účtu (a jsou tak jednoznačně identifikovatelní) a také těch, kteří toto nastavené nemají (jejich počet je pak méně přesný). Další velice užitečnou informací by byly frekvence událostí autorů tohoto projektu, avšak na tyto data klade API GitHubu takové omezení, že jejich využití není možné.

Poslední skupinou dat jsou informace o větvení repozitáře. Větší počet větvení může ukazovat na oblíbenější projekt, může však také znamenat, že se projekt již nevyvíjí a že je lidmi větven, aby mohli pokračovat na jeho vývoji.

5.1.1 Definice budoucí aktivity projektu

Protože implementovaná metoda bude odhadovat budoucí aktivitu projektu, musíme pro vytvoření tohoto modelu použít učení s učitelem. Pro tento způsob strojového učení však potřebujeme, aby vstupní dataset obsahoval již označená data. Metoda strojového učení se pak podle tohoto označení naučí, jaké hodnoty atributů vedou k jakým hodnotám budoucí aktivity.

Označení záznamů v datasetu vyžaduje, abychom přesně definovali pojem „budoucí aktivita projektu“ v kontextu této práce. K výpočtu této hodnoty pak použijeme repozitáře, do kterých již další dobu nebyla zaslána žádná změna. Tyto repozitáře prohlásíme za neaktivní. Následně vygenerujeme náhodné datum mezi první zaslanou změnou a poslední zaslanou změnou repozitáře, spočítáme popsané atributy pouze do této chvíle (budeme ignorovat všechny zaslání změn, hlášení chyb atd. s novějším datem) a pak použijeme hodnoty atributů od tohoto data do konce pro spočítání budoucí aktivity. Tímto způsobem je simulováno to, že odhadujeme budoucí aktivitu v čase života repozitáře a zároveň máme možnost zjistit, jaký bude skutečný vývoj od této chvíle.

Přesných definic budoucí aktivity může být několik. První způsob, který bychom mohli použít, je definovat projekt jako aktivní právě tehdy, když do něj v budoucnu bude zaslána alespoň jedna změna. Tato definice budoucí aktivity je intuitivní - pokud autoři nezašlou do projektu už ani jednu změnu, můžeme prohlásit projekt za neaktivní. Pokud do něj budou autoři zasílat změny i v budoucnu, prohlásíme projekt za aktivní.

Tato definice aktivního projektu má z pohledu strojového učení jednu nevýhodu. Pokud budeme zjišťovat aktivitu projektu v poslední den jeho existence (pokud do něj bude například zítra zaslána poslední změna), je takovýto repozitář dle naší definice aktivní. Pokud budeme zjišťovat aktivitu projektu o jeden den později, je dle naší definice projektem neaktivním. Je předpoklad, že metoda, která se bude učit dle daného datasetu, bude mít problém projekty správně klasifikovat, protože hodnoty atributů (frekvence zasílání změn atp.) budou u obou případů téměř totožné. Z toho vyplývá, že „budoucí aktivita“ projektu musí být definována jako spojitá hodnota (např. číslo od 0 do 100). V tomto případě bude znamenat drobná změna atributů také drobnou změnu odhadované budoucí aktivity. V takové situaci, která byla popsána výše, bude rozdíl jednoho dne znamenat změnu aktivity například 0,25 → 0 místo původní změny `aktivní` → `neaktivní`. Z toho dále vyplývá, že bude nutné použít metody pro regresi místo metod pro klasifikaci.

Pro potřeby této práce nakonec vzniklo těchto definic několik a na základě jejich vlastností z nich bude jedna vybrána. Míru budoucí aktivity projektu můžeme definovat jako **procento změn, které do repozitáře ještě zbývá zaslat**. V tomto případě značí hodnota 100 projekt, který právě vznikl, hodnota 0 značí projekt, který již nebude dále vyvíjen a hodnota 50 značí projekt, který je přesně v polovině doby jeho života. Taková definice má tu výhodu, že výsledná hodnota patří do intervalu $< 0, 100 >$.

Další možností je definovat budoucí aktivitu jako **frekvenci změn v příštím týdnu,**

měsíci, půl roce či roce. Takto definovaná míra budoucí aktivity má tu výhodu, že eliminuje problém s repozitáři s celkovým malým počtem změn. Nevýhodou však je, že výsledek může nabývat hodnot z polouzavřeného intervalu. Pokud nám tedy model sdělí, že frekvence změn v budoucím roce je 0,5, nevíme, zda se v kontextu daného projektu jedná o velkou či malou aktivitu.

Tento problém se snaží vyřešit poslední definice míry budoucí aktivity. Ta je podle ní definována jako **poměr budoucí frekvence zasílání změn ku dosavadní frekvenci zasílání změn**. Pokud definujeme míru budoucí aktivity takto, bude hodnota 1 znamenat, že projekt zůstane stejně aktivní jako doposud, hodnota > 1 bude znamenat, že se aktivita zvýší, hodnota < 1 bude znamenat, že se aktivita sníží a hodnota 0 bude znamenat, že projekt již nebude nadále vyvíjen. Problém s touto metodou je však ten, že frekvenci musíme počítat za určité časové období. Zatímco pro výpočet dosavadní frekvence změn jsou obě hranice intervalu definovány, pro výpočet budoucí frekvence máme interval pouze polouzavřený. Druhou hranici intervalu určíme tedy jako délku intervalu od počátku projektu do času, ve kterém aktivitu počítáme. Takovýto výpočet má následující vlastnosti:

1. pokud je místo výpočtu blíže počátku projektu než jeho konci, neleží místo konce projektu ve výpočtu budoucí frekvence a výsledná frekvence je pak opravdová frekvence změn v tomto období,

2. pokud je místo výpočtu blíže konci projektu než jeho počátku, leží místo konce projektu ve výpočtu budoucí frekvence a proto se frekvence změn snižuje s blížícím se koncem projektu.

V rámci této práce byly analyzovány tři typy z projektů, které se na serveru GitHub vyskytují. Jsou to:

- školní projekt, do kterého bylo zasláno pouze několik změn, frekvence zasílání změn vyvrcholila těsně před odevzdáním a po datu odevzdání na něm již autor nepracoval,

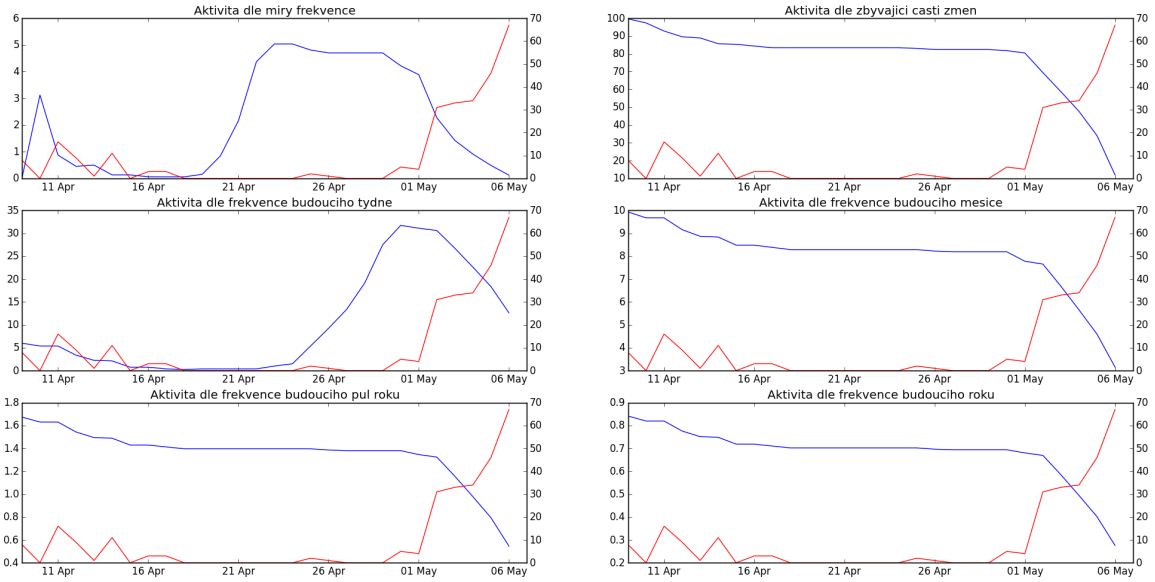
- v historii velmi aktivní (avšak nyní již dále nevyvíjený) komunitní projekt, do kterého bylo zaslána spousta změn v časovém rozpětí několika let a nakonec

- komunitní projekt, který po několika letech nízké aktivity začal být opět aktivní a je aktivní až doposud.

Grafy změn a budoucích aktivit prvního projektu[2] jsou na obrázku 5.1. Červenou barvou je zde zobrazen denní počet změn (de facto frekvence změn), modrou barvou odpovídající míra budoucí aktivity (jedná se pouze o ilustrační grafy - nezáleží na absolutních hodnotách funkcí, ale na tvaru jejich grafů). Grafy funkcí míry budoucí aktivity dle poměru frekvence a frekvence budoucího týdne jsou podobné a oba dobře vystihují budoucí aktivitu - frekvence budoucího týdne je v tomto případě zřejmě přesnější, protože na její hodnotu nemá vliv pár zaslaných změn na začátku projektu. Grafy budoucích frekvencí zbývajících časových intervalů jsou podobné grafu funkce zbývajících procent. Ačkoliv z nich není přímo patrné, jak přesně se bude aktivita vyvíjet, dá se podle těchto grafů stále vyčíst, kolik práce na projektu ještě zbývá.

Grafy těchto funkcí projektu[3], který byl dříve velice aktivní, jsou na obrázku 5.2. Aktivita dle poměru frekvencí zde není tolik přesná. Sice ukáže nárůst aktivity během druhého

[2] `https://github.com/garretraziel/icp`
[3] `https://github.com/sgala/gajim`



Obrázek 5.1: Grafy změn a aktivit školního projektu

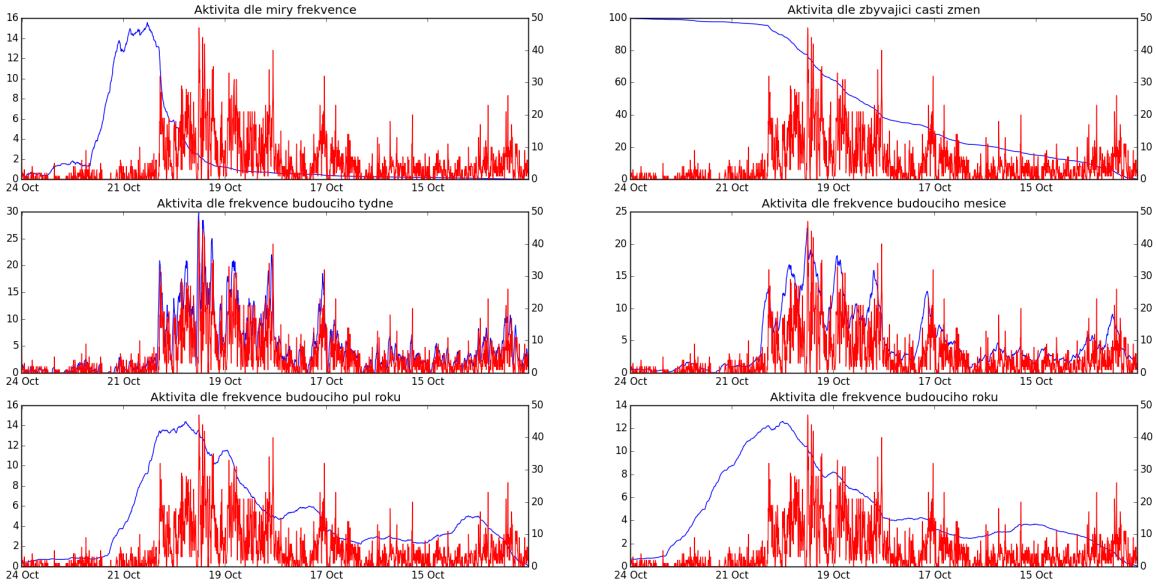
roku trvání projektu, náhle však klesne k nízkým číslům. Je to z toho důvodu, že zatímco na začátku vzroste aktivita téměř šestnáctkrát, dále se aktivita nezmění, čímž klesne graf funkce na hodnotu 1. Grafy funkcí dle frekvence v budoucím týdnu a měsíci jsou velmi podobné a jejich změna kopíruje změnu počtu zaslání. V tomto případě míru budoucí aktivity vyjadřují nejpřesněji frekvence budoucího půlroku a roku.

Jako poslední jsou na obrázku 5.3 grafy funkcí budoucích aktivit v případě stále vyvíjeného projektu[4]. V tomto případě je zřejmá nevýhoda určování budoucí aktivity pomocí poměru frekvencí změn. Čím déle je totiž projekt vyvíjen, tím delší doba musí uplynout od jeho konce. Například, pokud byl projekt vyvíjen deset let, můžeme podle charakteru funkce počítat míru budoucí aktivity až deset let od jeho posledního zaslání změny. V tomto případě je však projekt stále vyvíjen a pro to vrátila funkce hodnotu -1. V praxi by to zkomplikovalo získávání hodnot pro dataset a snížilo počet projektů, které můžeme do datasetu zařadit - dobu, která uplynula od poslední změny v repozitáři totiž můžeme zjistit až po náročném získání změn repozitáře. Tento problém nastává i u všech ostatních funkcí - tam je ale stáří, jaké projekt musí mít, pevně dané. Tento problém se vyskytuje i u funkce, která určuje aktivitu podle zbývajícího procenta změn, ačkoliv to není z tohoto grafu zcela zřejmé. Pro projekt, který může být stále velmi aktivní, vrátí hodnotu 0, protože zbývající změny budou do projektu zaslány v budoucnosti. I v tomto případě je tedy nutné určit minimální dobu, která musí uplynout od poslední změny v repozitáři.

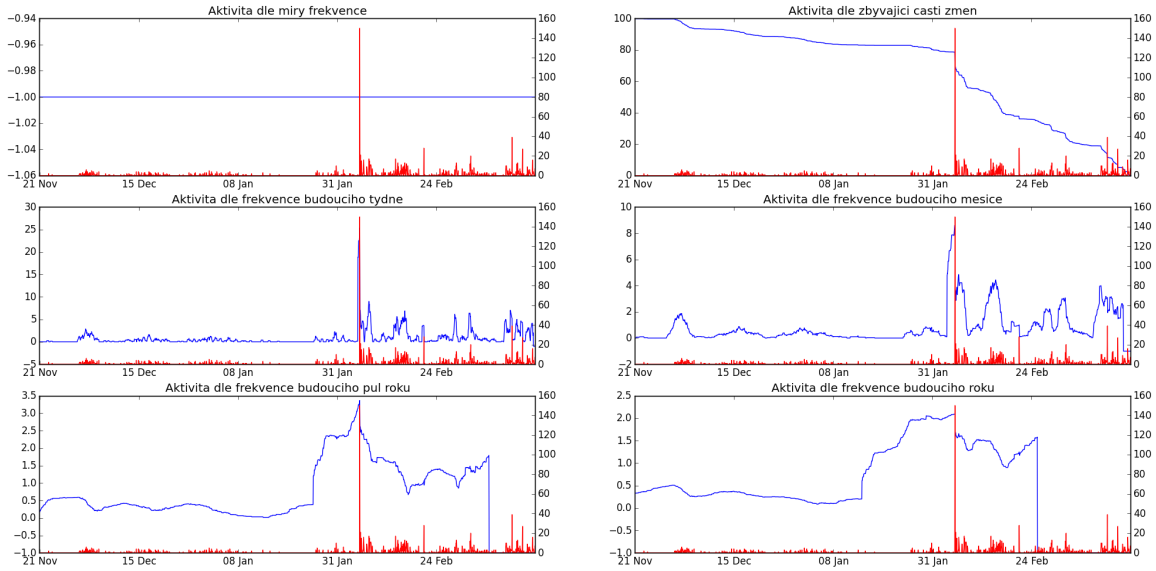
5.2 Vlastnosti získaného datasetu

Při prvním spuštění získal program ghaminer 10 981 záznamů za tři dny. Tento dataset byl následně analyzován v programu Weka a v prostředí IPython za pomocí nástrojů ze sady SciPy. Jednotlivé vlastnosti jsou popsány níže. Z atributů, které mají sloužit jako míra budoucí aktivity, je v tomto seznamu (až na jednu výjimku) zahrnut pouze atribut zbývajících procent změn `percentage_remains`. Toto zjednodušení bude zdůvodněno v následující

[4] `https://github.com/os-autoinst/os-autoinst`



Obrázek 5.2: Grafy změn a aktivit v minulosti aktivního projektu

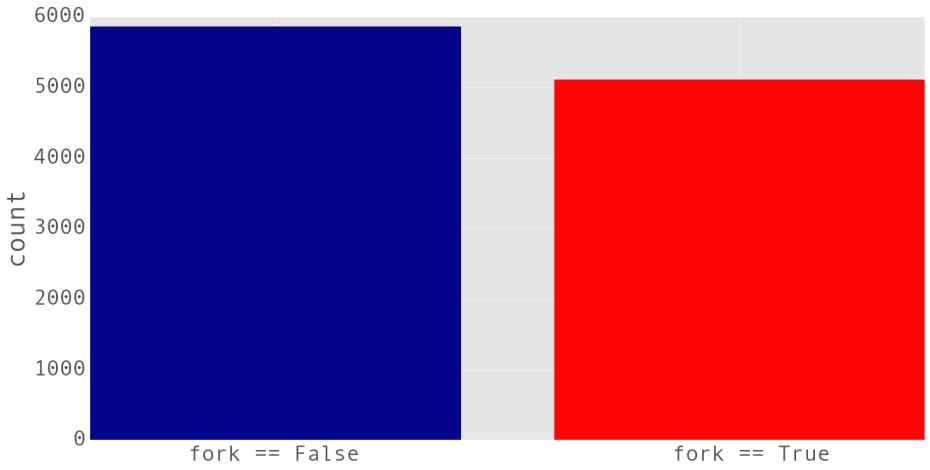


Obrázek 5.3: Grafy změn a aktivit stále aktivního projektu

kapitole.

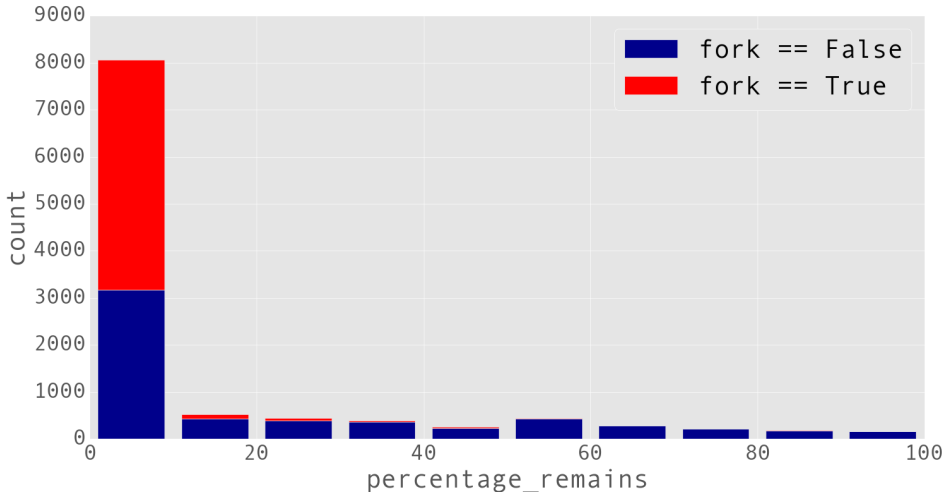
Analýzou byly zjištěny následující vlastnosti:

- Zhruba 47 % repozitářů tvoří repozitáře, které vznikly rozvětvením. To znamená, že 53 % repozitářů je původních.



Obrázek 5.4: Počet repozitářů podle příznaku větvení

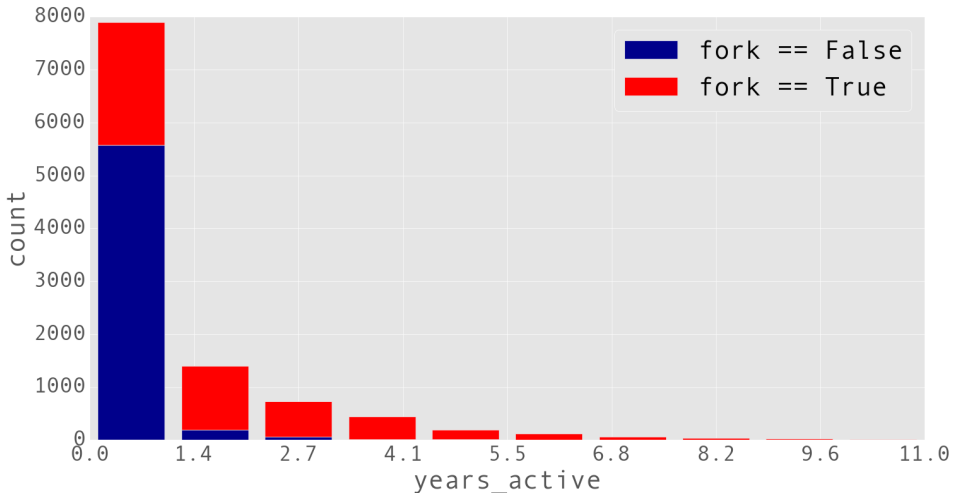
- Jak bylo předpokládáno, pokud je repozitář označený jako větvení, má projekt výrazně menší pravděpodobnost, že bude v budoucnu vyvíjen, jak je zřejmé z histogramu na následujícím obrázku. Medián zbývajících procent u repozitářů, které nejsou rozvětveními, je 5 %, průměr 21 %. Medián zbývajících procent u repozitářů, které vznikly větvením, je 0 %, průměr 1,5 %.



Obrázek 5.5: Procento zbývajících změn k zaslání podle příznaku větvení

- Nejstarší projekty z datasetu byly staré až 45 let. Jedná o repozitáře projektů, které vznikly dlouho před vznikem samotného GitHubu. Překvapivě, starší projekty jsou

většinou původně větveními jiných projektů, jak je vidět na následujícím histogramu.

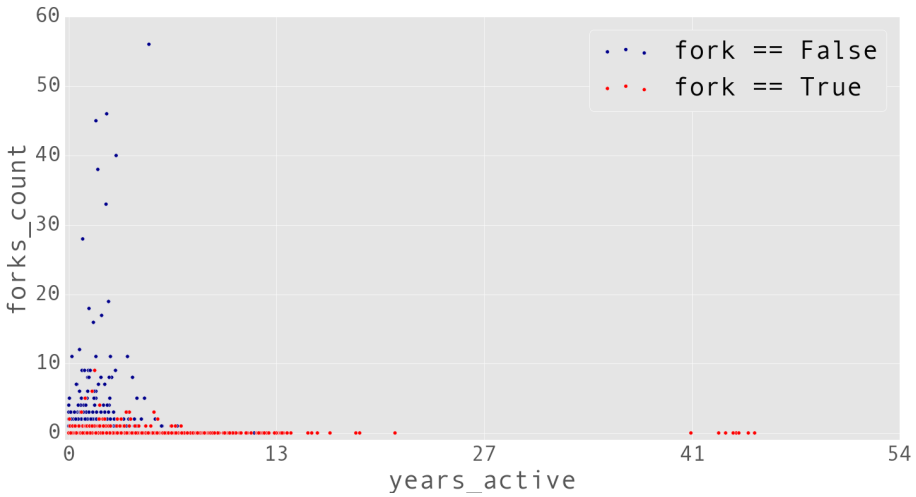


Obrázek 5.6: Stáří repozitářů v rocích podle příznaku větvení

Vysvětlení může být jednoduché - stáří projektu je udávané dle data prvního zaslání změny a ne data vytvoření větvení. Pokud někdo vytvoří větvení starého repozitáře, ghaminer toto větvení označí za stejně staré, jako původní repozitář. Projekty, které jsou po několika letech stále aktivní, mají vyšší pravděpodobnost, že je lidé budou větvit. Tím vzniká velká spousta starých repozitářů, které jsou větveními - větším množstvím rozvětvení jednoho původního repozitáře.

Po tomto zjištění bylo potřeba udělat úpravu v programu ghaminer. Je jistě logické, aby byla aktivita repozitářů, které vznikly větvením, posuzována až od tohoto větvení.

- Následující graf ukazuje závislost počtu větvení repozitáře na stáří projektů a na příznaku, zda samotný repozitář je větvením.

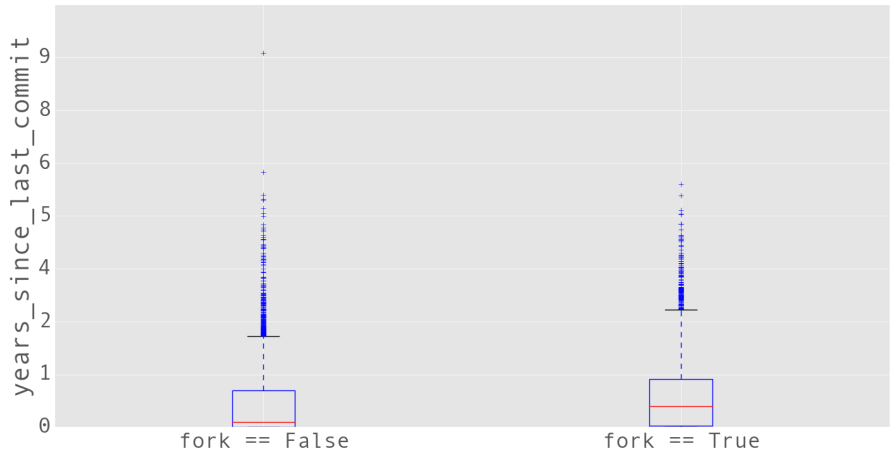


Obrázek 5.7: Závislost počtu větvení na stáří repozitáře a příznaku větvení

Z grafu vyplývá, že nejčastěji jsou větveny repozitáře staré od jednoho roku do pěti let,

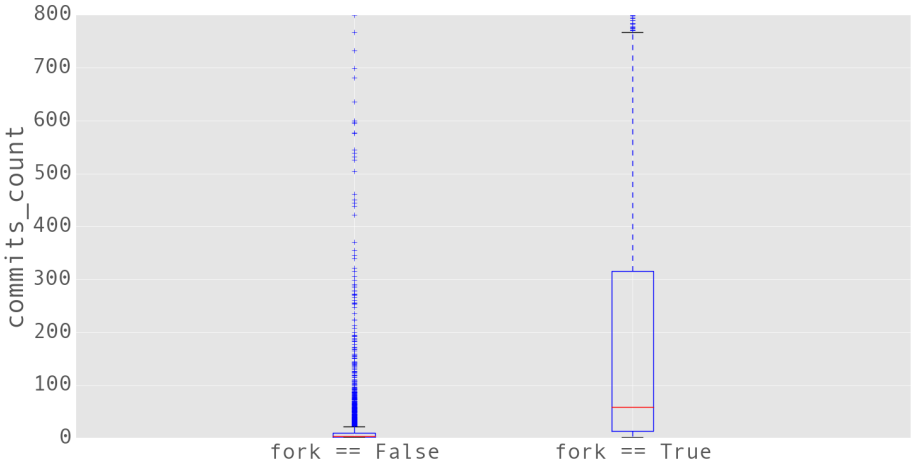
přičemž samy nebyly vytvořené větvením. Důvod většího stáří větvených repozitářů byl již diskutován.

- Na následujícím krabicovém grafu je zřejmé, že u repozitářů, které vznikly větvením, uběhla delší doba od posledního zaslání změny (medián 47 dní oproti mediánu 196 dní), což potvrzuje domněnku, že většina větvených repozitářů vzniká za účelem jednorázového požadavku o začlenění změn.



Obrázek 5.8: Krabicový graf počtu let od posledního zaslání změny

- Následující krabicový graf ukazuje závislost počtu změn v repozitáři na příznaku větvení. Ukazuje, že medián počtu změn v repozitářích, které nevznikly větvením je menší, než ten stejný údaj u repozitářů, které větvením vznikly (4 změny oproti 59 změnám). Výsledný graf je na ose y záměrně oříznut hodnotou 800.

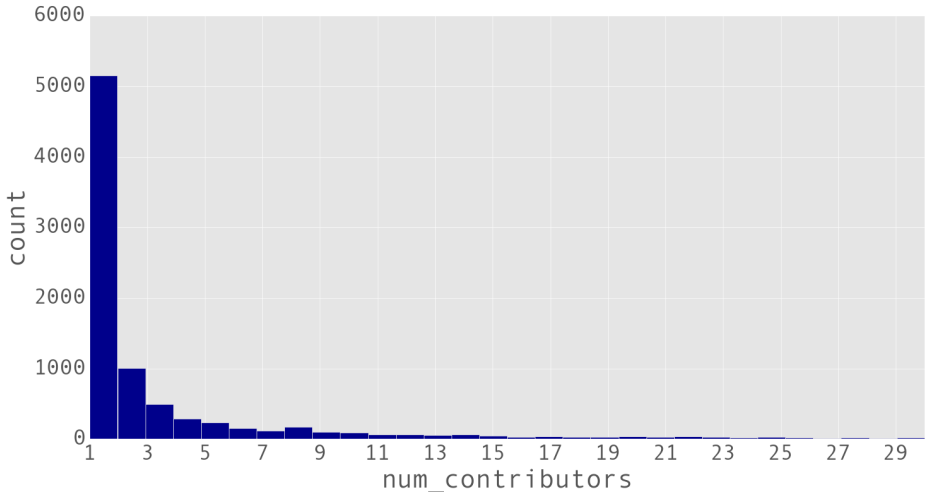


Obrázek 5.9: Krabicový graf počtu změn v repozitáři

- Procento zbývajících změn je slabě záporně korelované k počtu dnů, které uplynuly od

poslední zaslání změny s hodnotou Pearsonova korelačního koeficientu $\rho = -0{,}2677$.

- Dle očekávání, frekvence zasílání změn je mezi sebou silně korelovaná, přičemž čím blíže si jsou intervaly délkou, tím vyšší je korelace. Například korelace mezi frekvencí zasílání změn v posledním týdnu a posledním měsíci je $\rho = 0{,}94$, v posledním týdnu a posledním půl roce je $\rho = 0{,}882$, v posledním týdnu a posledním roce je $\rho = 0{,}85$ a v posledním týdnu a za celou dobu existence je $\rho = 0{,}779$.
- Mezi frekvencí změn v posledním týdnu a frekvencí změn v příštím půl roce je nejvyšší korelace ze všech korelací mezi minulými a budoucími frekvencemi změn a to s hodnotou $\rho = 0{,}3$.
- V celém datasetu je pouze 226 projektů, které mají nenulový počet nahlášených chyb, 147 repozitářů, které mají nenulový počet komentářů, 189 repozitářů, do kterých byla zaslána aspoň jedna žádost o začlenění a jen 419 repozitářů bylo aspoň jednou rozvětveno. V celém datasetu se také nachází pouze tři záznamy, které nemají ani jednu chybějící položku.
- Na následujícím grafu jsou zobrazeny počty lidí, kteří zaslali změnu do jednoho repozitáře. Do více jak poloviny projektů v získaném datasetu přispívá pouze jeden člověk.



Obrázek 5.10: Počty lidí, kteří přispívají do jednoho repozitáře

- Zatímco v mnoha úlohách pro dolování z dat je dataset součástí zadání úlohy (a hodnoty pro predikci jsou již k datům správně přiřazeny), program na získání datasetu je zde součástí projektu a jako takový může také obsahovat chyby. Vytvoření špatného datasetu bude mít za následek vytvoření nekvalitního modelu pro predikci, ačkoliv metody pro ohodnocení modelu budou vracet vysoké hodnoty přesnosti. Důvodem je, že vytvořený model sice bude správně popisovat vstupní dataset, ale ten bude špatně popisovat data reálného světa.

Kapitola 6

Experimenty se získaným datasetem

Pro experimentování se získaným datasetem byl použit nástroj Weka, úspěšné metody byly následně vyzkoušeny v interaktivním interpretru IPython jazyka Python za pomocí knihoven scikit-learn pro strojové učení, SciPy pro vědeckotechnické výpočty, NumPy a pandas pro práci s maticemi a matplotlib pro vytváření grafů.

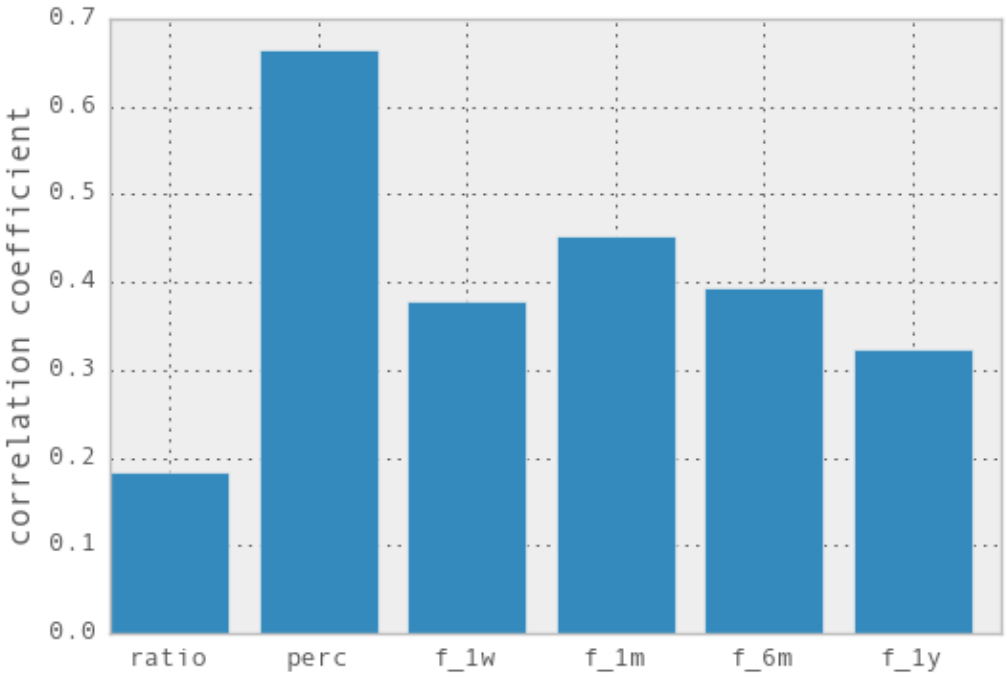
Jak bude popsáno dále, pro účely této práce musely být nakonec získány datasety tři. V průběhu experimentování s datasetem byly totiž zjištěny chyby v programu, které zapříčinily, že dataset nesprávně reprezentoval existující data a ačkoliv bylo s níže popsanými metodami dosaženo vysoké přesnosti, predikce na příkladech z reálného světa byly často nelogické.

Prvním problémem byl fakt, že náhodné datum, které bylo použito pro spočítání statistik a budoucí aktivity, bylo u repozitářů, které vznikly jako větvení, vygenerováno v libovolném bodě doby života repozitáře a ne až po vytvoření větvení. Z toho důvodu se metody strojového učení nenaučily pravidla, které byly spojeny s vlastnostmi větvených repozitářů. Spočítané budoucí aktivity totiž neodpovídaly aktivitě po vytvoření větvení.

Dále bylo zjištěno, že v původním datasetu schází atribut `days_since_last_commit`. Náhodné datum, které bylo použito pro spočítání statistiky a budoucí aktivity, bylo vždy vygenerováno mezi začátkem a koncem práce na kódu v repozitáři. V průběhu učení tedy scházely příklady repozitářů, do kterých už několik let nebyla zaslána žádná změna a které by z našeho pohledu byly již mrtvé. Ačkoliv bylo s tímto datasetem dosaženo vysoké přesnosti v rámci křížové validace, když pak měl na tomto datasetu naučený model strojového učení predikovat budoucí aktivitu repozitáře, do kterého byla před pěti lety zaslána pouze jedna změna a od té doby byl projekt neaktivní, predikoval tento model budoucí aktivitu 97 %, což je s nejvyšší pravděpodobností chybný odhad. Důvod byl ten, že se tento model v rámci učení nesetkal s příkladem repozitáře, který byl již dlouhou dobu neaktivní a nevěděl proto, jak se bude aktivita vyvíjet v tomto případě.

V průběhu experimentování se také ukázalo, že metody strojového učení jsou s velkým odstupem nejúspěšnější při predikci budoucí aktivity založené na procentu zbývajících změn. Z toho důvodu byly zbývající způsoby pro definici budoucí aktivity zanedbány. Na obrázku 6.1 je porovnání přesnosti predikce jednotlivých atributů pomocí metody random forest[1] (rozložení úspěšnosti ostatních metod bylo podobné). Pro porovnání přesnosti byl

[1] Hodnota korelačního koeficientu ρ neodpovídá přesně hodnotě prezentované dále v tomto textu, protože tyto hodnoty byly naměřeny na starším datasetu.



Obrázek 6.1: Porovnání přesnosti předpovědi jednotlivých způsobů určení budoucí aktivity: *ratio* - na základě poměru minulé a budoucí frekvence, *perc* - na základě procent zbývajících změn, *f_1w* - na základě frekvence změn budoucího týdne, *f_1m* - na základě frekvence změn budoucího měsíce, *f_6m* - na základě frekvence změn budoucího půlroku, *f_1y* - na základě frekvence změn budoucího roku

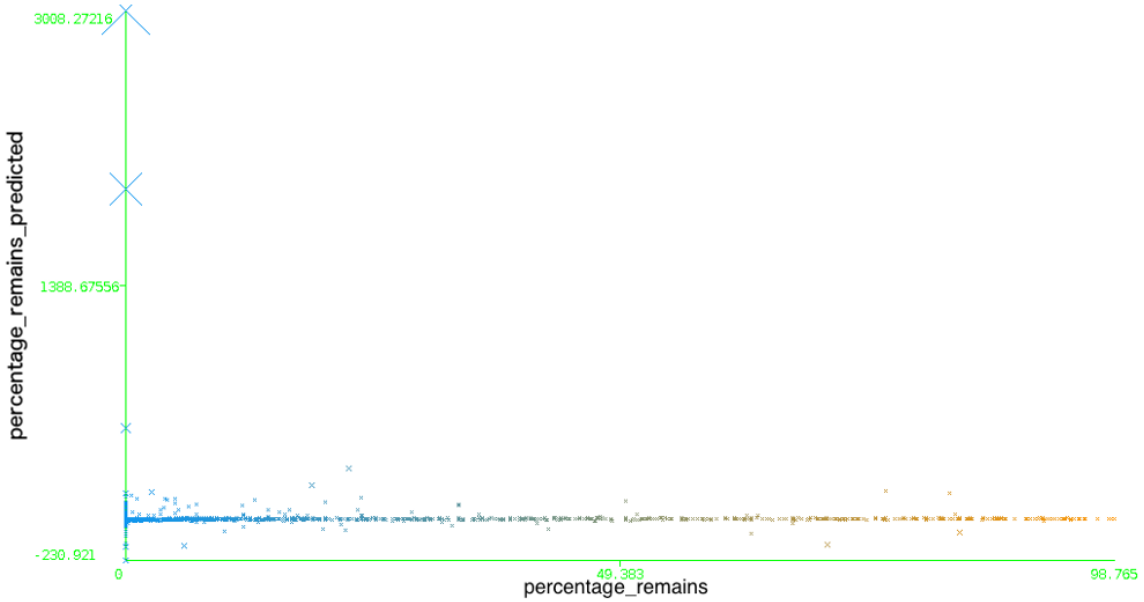
použit korelační koeficient ρ mezi předpovězenými a opravdovými hodnotami (vyšší hodnota ρ znamená lepší výsledek).

6.1 Experimenty v prostředí Weka

Nástroj Weka, jmenovitě jeho část nazvaná Explorer, slouží pro jednoduché a rychlé experimentování. Z tohoto důvodu provádí spoustu operací (jako například nahrazení chybějících hodnot) automaticky. Uživatel má nad touto funkcionalitou menší kontrolu, ale získané výsledky nám stačí pro získání přibližného přehledu o úspěšnosti. Uživatel může vizualizovat vzájemné vztahy jednotlivých veličin či provést predikci. Weka automaticky provádí křížovou validaci a vrací míru chyby při použití dané metody. Následuje seznam metod, které byly na datasetu vyzkoušeny. Teoretická část k této kapitole byla čerpána z knihy *Data Mining: Practical machine learning tools and techniques* [28].

6.1.1 Lineární regrese

Lineární regrese je základní metodou, používanou pro predikci numerické hodnoty. Předpokládá, že výstupní hodnota je lineární kombinací vstupních hodnot a váhy jednotlivých atributů hledá pomocí metody nejmenších čtverců. Zjednodušeně řečeno, metoda lineární regrese předpokládá, že hodnoty vstupního vektoru tvoří přímku (rovinu, hyperrovinu...)



Obrázek 6.2: Vizualizace chyb modelu lineární regrese

a hledá koeficienty této přímky tak, aby minimalizovala chybu. Pokud jsou však ve vstupních datech nelinearity, lineární regrese je nedokáže popsat. Lineární regrese má následující úspěšnost:

korelační koeficient ρ	0,057
RMSE	40,5416
RRSE	237,9393 %

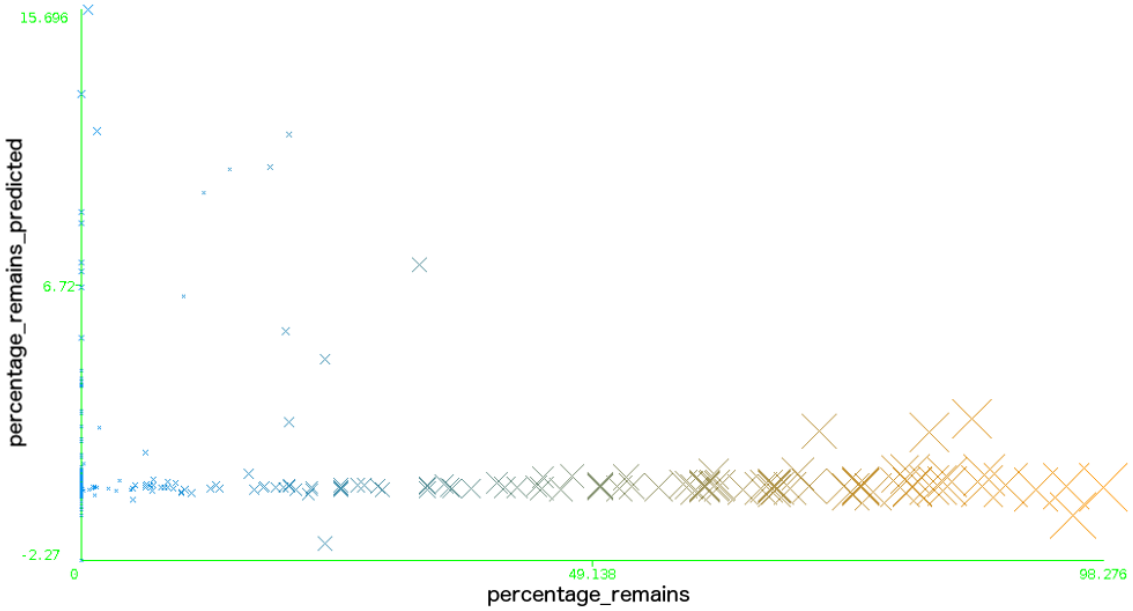
Jak vyplývá z tabulky, není lineární regrese v tomto případě příliš úspěšná. Na obrázku 6.2 je vizualizace chyb. Na ose x je opravdová hodnota budoucí aktivity, na ose y je předpovězená. Křížek v grafu značí chybný odhad hodnoty a jeho velikost udává velikost chyby. V grafu je vidět, že pro určité hodnoty vrací lineární regrese naprosto nesmyslné hodnoty. Pro projekt, u kterého je zbývající počet procent 0, vrátí lineární regrese hodnotu 3008. Pokud tyto hodnoty považujeme za hodnoty odlehlé a z datasetu je odstraníme, získáme ve skutečnosti výrazně horší výsledky, než v předešlém případě:

korelační koeficient ρ	0,03
RMSE	138,5941
RRSE	812,8241 %

Tento fakt nám dokazuje, že lineární regrese není pro náš dataset vhodná, protože vykazuje známky chaotického chování.

6.1.2 Neuronová síť

Následně vyzkoušíme neuronovou síť, která je dostupná v nástroji Weka. Jedná se o dopřednou neuronovou síť, jejíž učení probíhá metodou *backpropagation*. Jelikož by učení sítě na tolika vstupech a tolika vzorcích trvalo neúnosně dlouhou dobu, byla nejprve provedena



Obrázek 6.3: Vizualizace chyb modelu SVM

redukce dimenzionality pomocí PCA. Z neznámého důvodu však PCA v prostředí Weka vytvořila dataset, který měl o 27 atributů více. Dataset byl proto pouze redukován na prvních 1000 položek. Výsledky křížové validace neuronové sítě jsou následující:

korelační koeficient ρ	0,0348
RMSE	65,9388
RRSE	357,8486 %

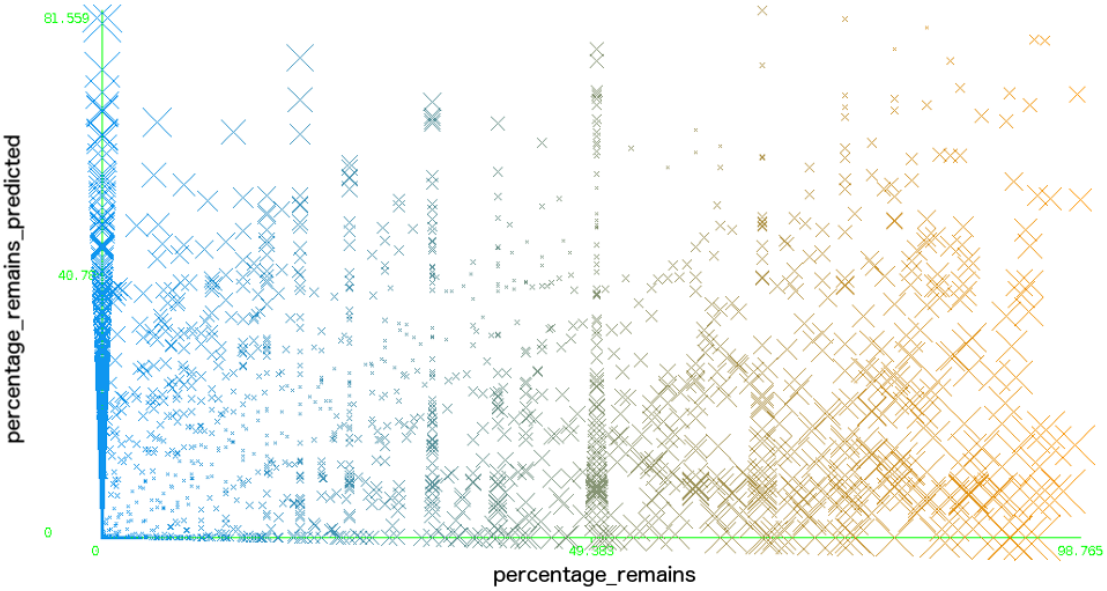
Neuronová síť je tedy na našem datasetu podobně neúspěšná, jako byla lineární regrese.

6.1.3 SVM

Při použití metody *support vector machines* (pro regresi v prostředí Weka nazvané *SMOreg*) můžeme dále zvolit, jaký *kernel* se má použít - lineární, polynomiální, radiální. SVM s polynomiálním kernelem, které je ze všech tří možností nejúspěšnější, vrací tyto hodnoty chyb:

korelační koeficient ρ	0,044
RMSE	19,4285
RRSE	105,4382 %

Ač jsou tyto hodnoty lepší než v předešlých dvou případech, z grafu chyb zjistíme, že předpovědi lineární regrese nejsou z našeho pohledu nijak užitečné. Jak je vidět na obrázku 6.3, model vytvořený metodou SVM predikuje pro všechny hodnoty číslo menší jak 15,7, přičemž pro většinu hodnot predikuje budoucí aktivitu 0, nezávisle na opravdové hodnotě budoucí aktivity.



Obrázek 6.4: Vizualizace chyb modelu kNN

6.1.4 Metoda k-nejbližších sousedů

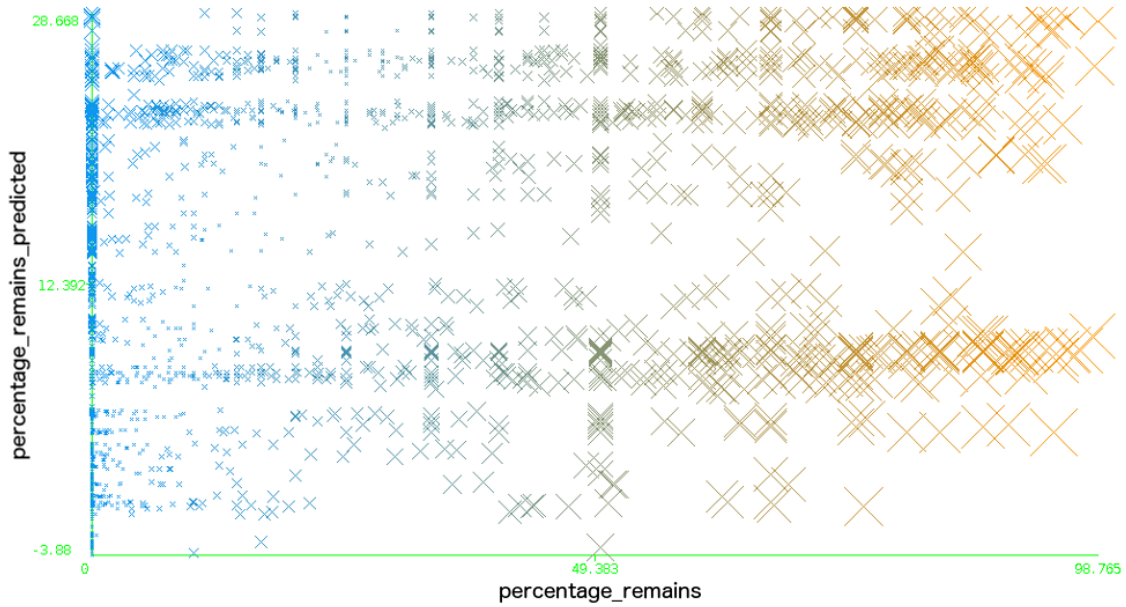
Metoda k-nejbližších sousedů (v originále *k-nearest neighbours, kNN*) patří mezi velmi jednoduché metody. V nástroji Weka je zařazen do skupiny tzv. *líných klasifikátorů*, což znamená, že ve fázi učení neprovádí žádné výpočty - veškeré výpočty se provádí až při samotné predikci. Níže jsou uvedeny hodnoty chyb při $k = 5$:

korelační koeficient ρ	0,5282
RMSE	14,693
RRSE	86,1911 %

Z těchto hodnot vyplývá, že je metoda kNN výrazně úspěšnější, než metody předchozí. Mezi předpovězenými a opravdovými hodnotami je střední korelace a jak ukazuje RRSE, metoda kNN je konečně úspěšnější, než jednoduchý prediktor průměru. Na obrázku 6.4 je zobrazen graf chyb. Ačkoliv se na první pohled může zdát, že tento graf obsahuje více chyb než grafy předchozí, opak je pravdou. Obor hodnot metody kNN je totiž roven výstupním hodnotám datasetu, tudíž nedostáváme žádné extrémní hodnoty a tak může být graf chyb zobrazen celý, ve správném měřítku.

6.1.5 Aditivní regrese

Aditivní regrese patří mezi meta-algoritmy strojového učení. Predikuje na vstupním datasetu opakovaně algoritmem *decision stump* tak, že hodnoty, které byly v předchozí iteraci predikované špatně, jsou následně použity jako vstup učení pro další iteraci. Známějším meta-algoritmem strojového učení je AdaBoost, ten však dokáže v prostředí Weka pouze klasifikovat. Stejně jak kNN přináší aditivní regrese lepší výsledky, než lineární regrese či neuronová síť. Velikost chyb aditivní regrese je následující:



Obrázek 6.5: Vizualizace chyb aditivní regrese

korelační koeficient ρ	0,4825
RMSE	14,9332
RRSE	87,5998 %

Problém této metody se však ukáže vzápětí v grafu chyb na obrázku 6.5. Lineární regrese má totiž, podobně jak SVM, tendenci predikovat nízké hodnoty.

6.1.6 Diskretizace a J48

Nástroj Weka obsahuje množství metod, které slouží pro klasifikaci. Abychom těchto metod mohli využít, nabízí Weka meta-algoritmus „regrese diskretizací“, která na daném datasetu nejprve provede diskretizaci (převede spojité hodnoty na diskrétní) a následně použije nějakou z metod klasifikace. Jako klasifikační algoritmus použijeme algoritmus *J48*. Algoritmus J48 je mírnou úpravou známějšího algoritmu C4.5. Velikosti chyb této metody jsou následující:

korelační koeficient ρ	0,5241
RMSE	14,6382
RRSE	85,8697 %

Metoda J48 je nejúspěšnější z klasifikačních metod (například metoda *naïve Bayes* měla korelační koeficient pouze 0,0632).

6.1.7 M5Rules

Metoda M5Rules patří mezi metody, které na základě vstupního datasetu vytvářejí seznam pravidel pro výstupní hodnotu. Příklad jednoho takového pravidla, které v průběhu učení vytvořila metoda M5Rules, je následující:

```
Rule: 6
IF
commits_count > 1.5
commits_f_6m > 0.04
THEN

percentage_remains =
-0.5066 * commits_count
+ 10.8349 * commits_f_1m
- 21.6487 * commits_f_6m
+ 9.8374 * commits_f_all
+ 0.001 * days_active
+ 51.0854 [201/99.291%]
```

Velikosti chyb metody M5Rules jsou tyto:

korelační koeficient ρ	0,5592
RMSE	14,1598
RRSE	83,0634 %

Z vizualizace jejích chyb na obrázku 6.6 zjistíme, že kromě pár odlehlých hodnot je hlavní problém s touto metodou množství projektů, pro které M5Rules předpověděla nulovou budoucí aktivitu. Toto je zřejmě způsobeno množstvím repozitářů v datasetu, které skutečně nulovou budoucí aktivitu mají. Pokud tento dataset redukujeme na repozitáře s nenulovou budoucí aktivitou (kterých je 1624), vypadá úspěšnost této metody takto:

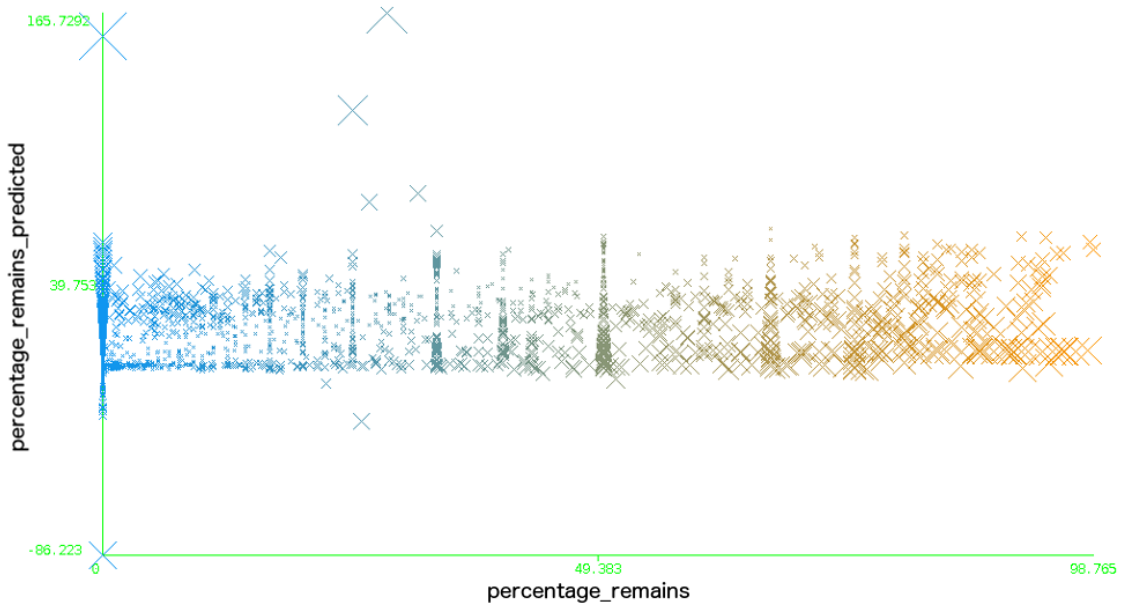
korelační koeficient ρ	0,5698
RMSE	18,1949
RRSE	82,3508 %

Zajímavé je, že ačkoliv má RMSE vyšší hodnotu než v předchozím případě, hodnota chyby RRSE nepatrně klesla. Pokud blíže prozkoumáme dataset zjistíme, že v předchozím případě byl průměr výstupní hodnoty `percentage_remains` 5,78, zatímco v redukovaném datasetu je její průměr 35,597. Protože původní dataset obsahoval velkou spoustu repozitářů s nulovou budoucí aktivitou, byl odhad hodnoty 5,78 často blízko opravdové hodnotě. V případě redukovaného datasetu je průměr častěji nepatrně dále od opravdové hodnoty. Znamená to, že redukovaný dataset má rovnoměrnější rozdělení. Ve výsledné aplikaci můžeme použít tento redukovaný dataset. Musíme ale dát pozor na to, že bude možná nepřesně reprezentovat rozložení budoucí aktivity projektů z reálného světa.

6.1.8 Random forest

Do další skupiny klasifikátorů patří metody, které pro reprezentaci pravidel používají rozhodovací strom. Ačkoliv metoda random forest patří mezi meta-algoritmy, v prostředí Weka je řazena do skupiny rozhodovacích stromů, protože výsledný model je z rozhodovacích stromů sestaven. Velikosti chyb metody random forest je následující:

korelační koeficient ρ	0,5198
RMSE	14,6891
RRSE	86,1682 %



Obrázek 6.6: Vizualizace chyb metody M5Rules

Vizualizace chyb metody random forest na obrázku 6.7 neukazuje na žádný zásadní problém s použitím této metody. Z grafu pouze vyplývá, že má metoda random forest tendenci podceňovat budoucí aktivitu projektů.

6.2 Experimenty v prostředí IPython

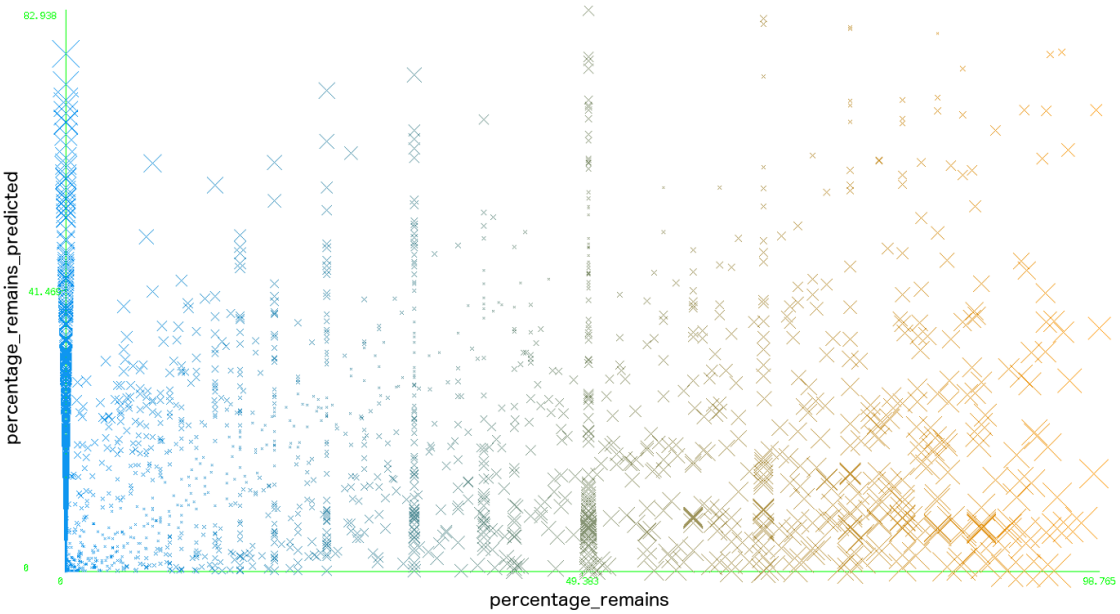
Experimenty v prostředí Weka sloužily pro vytvoření představy o teoretických možnostech nejrůznějších metod strojového učení na našem datasetu, ve výsledné aplikaci bude použita knihovna scikit-learn. Scikit-learn je rozšířením oblíbené knihovny SciPy, k vědeckotechnickým výpočtům přidává metody strojového učení. Díky jednoduchému API je použita v mnoha aplikacích, jako například Evernote či Spotify [8]. V této části ji, společně s knihovnami pandas a matplotlib, použijeme pro výběr metody, která bude nakonec přítomna ve výsledné aplikaci.

6.2.1 Předzpracování dat a pomocné metody

Dataset je potřeba předzpracovat ze dvou důvodů. Knihovna scikit-learn:

1. neumí pracovat s chybějícími hodnotami a
2. neumí pracovat s nenumerickým typem atributu.

Chybějící hodnoty jsou v našem datasetu reprezentovány znakem „?“ a obsahují ho ty záznamy, u kterých nebylo možné určit konkrétní hodnotu a zároveň by zde hodnota 0 měla jiný význam. Příkladem je průměrná doba pro vyřízení žádostí o začlenění, pokud v zadaném časovém období nebyla vytvořena žádná žádost o začlenění. Pro vyřešení problému chybějících hodnot použijeme objekt `sklearn.preprocessing.Imputer`, který dokáže nahradit chybějící hodnotu průměrem, mediánem či modem.



Obrázek 6.7: Vizualizace chyb metody random forest

Kategorický atribut obsahuje náš dataset naštěstí pouze jeden, příznak větvení. Použijeme objekt `sklearn.feature_extraction.DictVectorizer`, který funguje tak, že každou položku výčtu (kategorii) umístí jako samostatný atribut a položkám, u kterých patřil atribut do této kategorie, jej nastaví na hodnotu 1. V našem případě pouze transformuje hodnoty `True` na hodnoty 1 a hodnoty `False` na hodnoty 0.

Abychom mohli rychle experimentovat s metodami strojového učení, které scikit-learn poskytuje, použijeme `sklearn.pipeline.Pipeline`. Ta nám umožní zřetězit několik funkcí za sebe (doplnění chybějících hodnot, převod na numerické argumenty, popřípadě PCA, regresi) a na datasetu je spouštět pouze jedním příkazem.

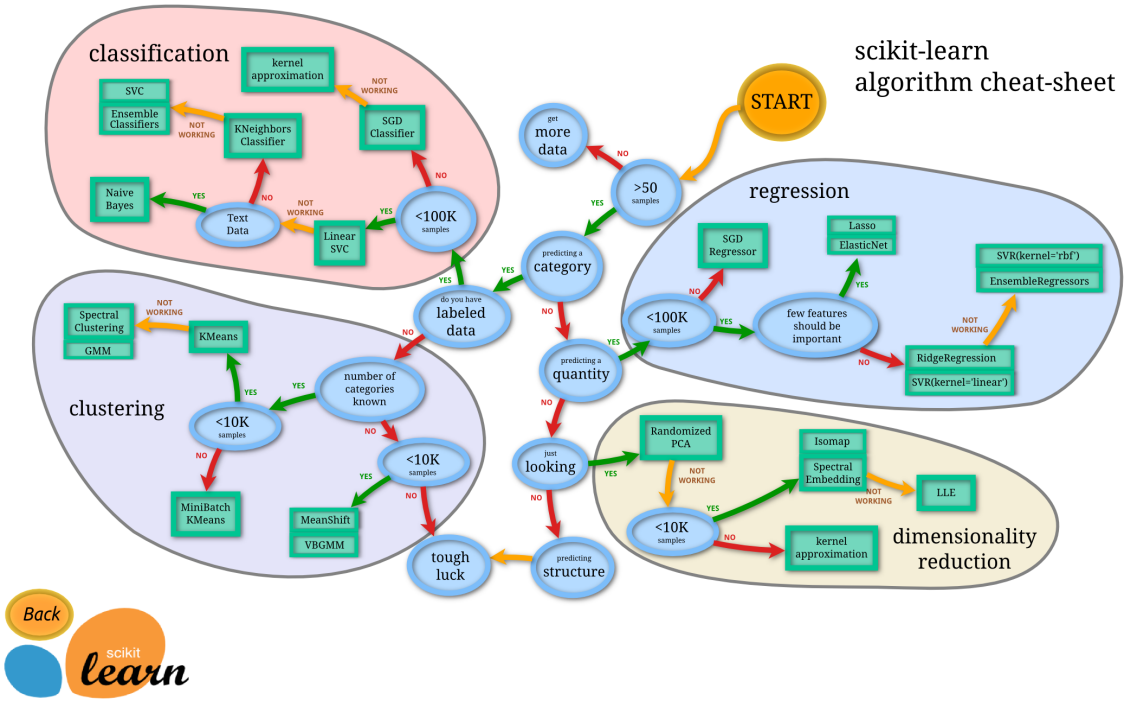
Pro hodnocení úspěšnosti metod použijeme metodu pro křížovou validaci `sklearn.cross_validation.cross_val_score`. Tato metoda nabízí několik možností hodnocení chyby, pro regresi nás bude zajímat MSE. Na RMSE jej následně převedeme pomocí:

```
numpy.sqrt(scores*-1).mean()
```

Zvláštností je, že MSE, které vrací funkce `cross_val_score`, má zápornou hodnotu. Je to z toho důvodu, že se scikit-learn snaží mít jednotné API pro všechny metody a proto jsou například hodnotící funkce psány tak, že vyšší hodnota znamená lepší skóre. Pro výpočet Pearsonova korelačního koeficientu bude použita funkce `scipy.stats.pearsonr`. Knihovna scikit-learn bohužel nenabízí hodnotící funkci RRSE. Dle výsledků z prostředí Weka však můžeme odhadnout, že RMSE prediktoru průměrem je přibližně 17,04 a toho pak využít při přibližném odhadu RRSE.

Kromě metod představených v nástroji Weka nabízí knihovna scikit-learn také další metody. Obecně se zaměříme na metody, doporučené pro regresi dokumentací knihovny scikit-learn. Scikit-learn také nabízí užitečný vývojový diagram pro určení, které metody bychom měli vyzkoušet. Tento diagram je na obrázku 6.8.

Posledním objektem, který budeme využívat při experimentování s knihovnou scikit-learn, je `sklearn.grid_search.GridSearchCV`. Mnoho metod přebírá při inicializaci další



Obrázek 6.8: Vývojový diagram metod knihovny scikit-learn

argumenty, vztahující se k vlastnostem těchto metod. `GridSearchCV` nabízí způsob, jak automaticky dojít k nejlepším hodnotám těchto argumentů pomocí prohledávání prostoru přípustných hodnot těchto metod.

6.2.2 Lasso

Lasso je jeden z jednodušších lineárních modelů, doporučených knihovnou scikit-learn. Jedná se o upravený algoritmus metody nejmenších čtverců, k němuž přidává hodnotu L_1 regularizace pro zabránění přeučení. Regularizátory L_1 a L_2 jsou funkce, při učení přičítané jako chyby predikce, které záměrně penalizují extrémní hodnoty. L_2 na rozdíl od L_1 více penalizuje řídké modely.

Metoda Lasso přidává k metodě nejmenších čtverců regularizátor L_1. Váha regularizátoru L_1, α, je parametrem metody Lasso. Pro $\alpha = 0$ je tedy metoda ekvivalentní metodě nejmenších čtverců. Jelikož učení pomocí všech atributů trvá velice dlouho, použijeme nejprve PCA s procentem rozptylu 95 %. Protože neznáme vhodnou hodnotu proměnné α, můžeme použít metodu `LassoLarsCV`, která provede nalezení nejvhodnější hodnoty α za nás. Při použití této metody získáme následující přesnost:

korelační koeficient ρ	0,195
RMSE	16,68
RRSE	97,88 %

Zajímavé je, že ačkoliv má metoda celkem nízkou chybu RMSE, má zároveň celkem nízkou hodnotu korelačního koeficientu.

6.2.3 Elastic net a ridge regression

Metoda elastic net je podobná metodě Lasso tím, že jako základ používá regularizovanou metodu nejmenších čtverců. Rozdíl je v tom, že k hodnotě chyby v metodě nejmenších čtverců nepřičítá pouze regularizátor L_1, ale také L_2. Pro správné nastavení vah obou regularizátorů použijeme metodu `ElasticNetCV`. Chyba metody elastic net je následující:

korelační koeficient ρ	0,195
RMSE	16,675
RRSE	97,85 %

Metoda elastic net se přesností od metody Lasso nijak výrazně neliší.

Ridge regression je poslední z metod lineární regrese, která byla testována. Podobně jako předchozí dvě metody, i ridge regression je založena na upravené metodě nejmenších čtverců. Není pro to překvapením, že velmi podobná je také její přesnost.

6.2.4 Gradient tree boosting

Meta-algoritmus gradient tree boosting provádí gradientní sestup nad rozhodovacími stromy. Protože si metody vytvářející pravidla a rozhodovací stromy vedly dobře při testování v prostředí Weka, můžeme předpokládat podobnou úspěšnost i u této metody. Míry chyb této metody jsou následující:

korelační koeficient ρ	0,5775
RMSE	13,88
RRSE	81,4 %

Metoda gradient tree boosting je nejúspěšnější z testovaných metod a proto bude použita při implementaci výsledné aplikace. Při použití PCA stoupne RMSE na hodnotu 14,944. Výsledný naučený model bude v aplikaci serializován, tudíž bude k procesu učení docházet velmi zřídka. Z toho důvodu nečiní delší doba učení takový problém a PCA nebude použito.

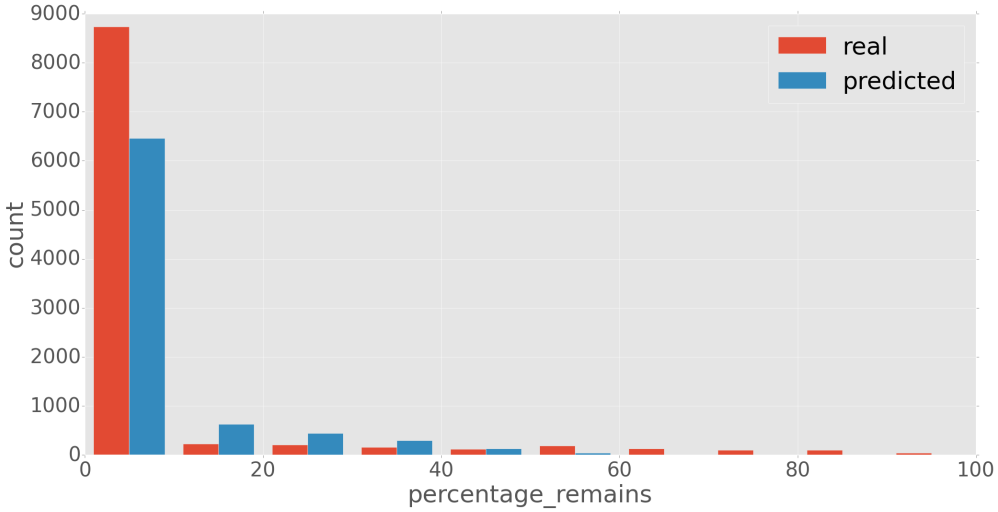
6.3 Další úpravy datasetu

Dále můžeme experimentovat s úpravami samotného datasetu.

Po testování metody gradient tree boosting na příkladech z reálného světa bylo zjištěno, že problémem pro přesnost mohou být takové repozitáře, do kterých bude za dobu jejich života zasláno málo změn. Pokud například bude existovat repozitář, do kterého byly zaslány dohromady čtyři změny v rozmezí čtyř dnů s frekvencí jedné změny denně (a od té chvíle byl již projekt neaktivní), po prvním dni je míra budoucí aktivity 75 %, po druhém dni 50 %, po třetím dni 25 % a po posledním dni 0%. Jedná se o skokové změny, přičemž ostatní sledované atributy (denní frekvence změn atp.) zůstaly téměř stejné.

Díky knihovně NumPy můžeme dataset jednoduše upravit tak, aby budoucí aktivitu všech projektů, do kterých bude dohromady zasláno méně jak 10 změn, nastavil na 0 následujícím kódem:

```
csv[100*csv['commits_count'] / (100 - csv['percentage_remains']) < 10] = 0
```



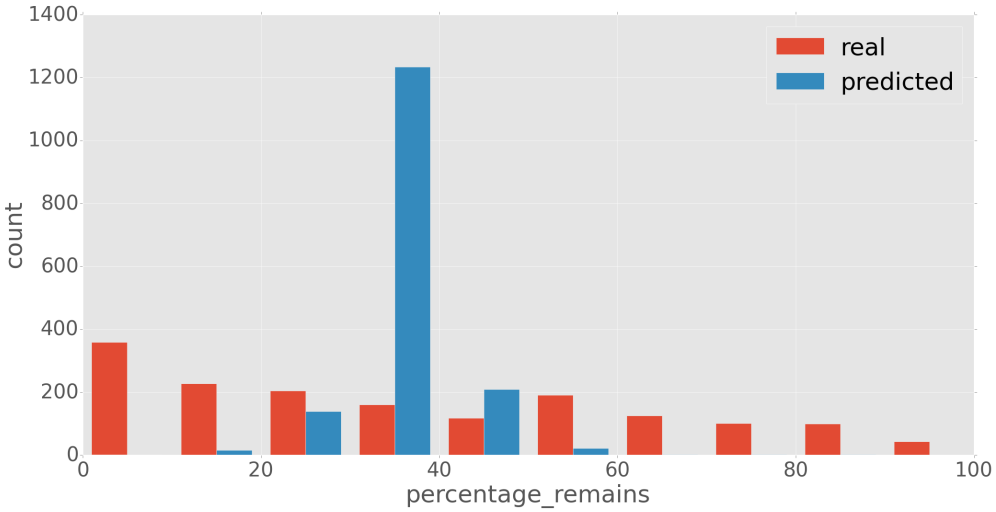
Obrázek 6.9: Histogram, porovnávající opravdovou a předpovězenou budoucí aktivitu na původním datasetu

Gradient tree boosting má na takto upraveném datasetu RMSE 7,76. Zlepšení RMSE se však dalo očekávat. Dataset totiž obsahuje více položek, jejichž budoucí aktivita je 0. V takovém případě by si lépe vedl i prediktor dle průměru.

Ve výsledném datasetu tato úprava není zahrnuta. Při testování na několika známých projektech vracel na tomto datasetu naučený prediktor překvapivě nízké míry budoucí aktivity. Například pro známý webový framework Flask[2] vracel míru budoucí aktivity 10 %, zatímco na původním datasetu naučený prediktor vracel 30 %. Ačkoliv nemůžeme vědět, který z těchto údajů je doopravdy správný, 10 % budoucí aktivity se po přezkoumání repozitáře zdá být málo. Stejně pesimistický byl i u ostatních projektů s na první pohled stále fungující a aktivní komunitou.

Další možnost úpravy datasetu je již zmíněné odstranění repozitářů s nulovou budoucí aktivitou. Jak však vyplývá z porovnání histogramu predikce na obrázku 6.9 a 6.10, není pomocí tohoto datasetu naučený prediktor příliš úspěšný.

[2] `https://github.com/mitsuhiko/flask`



Obrázek 6.10: Histogram, porovnávající opravdovou a předpovězenou budoucí aktivitu na datasetu bez položek s nulovou budoucí aktivitou

Kapitola 7

Popis fungování implementované aplikace

V další fázi této práce byla za pomoci webového frameworku Django implementována webová aplikace, která používá metodu gradient tree boosting a dataset, popsaný v kapitole 5, pro předpověď budoucí aktivity projektů. Součástí je také vizualizace nejrůznějších statistik o zadaném repozitáři pomocí JavaScriptové knihovny D3.js. Detaily webové aplikace, nazvané GitHub Activity Mining Project, zkráceně GHAM Project, se zabývá tato kapitola. V příloze C je stručný postup instalace této aplikace. Webový server s instancí této aplikace je spuštěn na adrese `https://ghamproject.com`.

7.1 Princip fungování

Získávání znalostí je zakončeno prezentací výsledků uživateli. V průběhu této práce byl získán dataset a také byla vybrána metoda strojového učení, která odhaduje míru budoucí aktivity s co nejmenší chybou. Nechceme však, aby výsledkem práce byla pouze uživateli nic neříkající hodnota RMSE. Z pohledu výsledků je nejdůležitější částí aplikace, která používá vybranou metodu strojového učení jako součást většího celku. Tato aplikace uživateli umožňuje rozhodnout, zda je vhodné použít určitý projekt, umístěný na serveru GitHub, s ohledem na budoucí vývoj tohoto projektu.

Po přihlášení se uživateli zobrazí seznam všech repozitářů, o kterých si vyžádal statistiky, přičemž uživatel může za pomocí jednoduchého formuláře získat statistiky o dalších repozitářích. Kvůli nutnosti stránkování API GitHubu je získávání statistik velice pomalé a musí probíhat asynchronně. Jakmile uživatel zadá požadavek o stáhnutí statistik, je přesměrován zpět na seznam svých repozitářů a zároveň se vytvoří vlákno, které stahování provádí. Jakmile dojde ke stáhnutí všech potřebných dat, klientská část aplikace tuto informaci získá pomocí periodicky zasílaného AJAX dotazu.

Protože stahování statistik trvá opravdu velice dlouhou dobu (3 000 požadavků u středně velkého projektu[1]), ve výsledné aplikaci je zahrnuto několik optimalizací. Jednou z nich je takový návrh uložení dat, že pokud si jiný uživatel vyžádá statistiky o repozitáři, o kterém si již jiný uživatel statistky vyžádal, nestahují se statistiky znovu - uživateli se pouze povolí přístup k již uloženým datům.

Pokud by tato aplikace měla být spolehlivá v produkčním nasazení, bylo by nutné použít

[1] `https://github.com/strongloop/express`

nějakou z forem distribuované fronty úloh, jako je například Celery[2]. Kromě ochrany proti zahlcení systému vlákny (ke kterému by v navrženém systému mohlo jednoduše dojít) by také umožňoval periodické aktualizování dat (použitý webový framework tuto funkcionalitu neposkytuje).

Po stáhnutí statistik může uživatel přistoupit na detail projektu. Zde jsou zobrazeny, kromě základních informací, také grafy změn, požadavků o zaslání změn, chyb a větvení, dále grafy počtu uzavřených chyb a požadavků o začlenění změn, průměrné doby pro jejich vyřízení a také predikce hodnoty budoucí aktivity. Pro snazší orientaci ve významu je míra budoucí aktivity rozdělena do čtyř skupin:

1. pokud je míra budoucí aktivity 100 % až 66 %, zobrazí se modrý odznak „At the Beginning“,
2. pokud je míra budoucí aktivity 34 % až 65 %, zobrazí se zelený znak „Active“,
3. pokud je míra budoucí aktivity 6% až 33 %, zobrazí se šedý znak „Declining“,
4. pokud je míra budoucí aktivity pod 6 %, zobrazí se červený znak „Probably dead“.

Tento odznak je také dostupný bez přihlášení, daný projekt ho pak může použít na své domovské stránce nebo ve svém README na GitHub stránce.

Jedním z problémů, které bylo nutné vyřešit v průběhu implementace, bylo přihlašování a přístup k API GitHubu. Protože by bylo krajně nepraktické, aby se k dotazování na API používal pouze jeden účet, je přihlašování na GHAM Project řízeno pomocí OAuth autentizace skrze GitHub. To znamená, že uživatel musí pouze povolit aplikaci přístup na jeho účet na GitHubu a tím se mu automaticky vytvoří účet na GHAM Project. Následně se jeho OAuth token uloží do databáze a je použit při všech úkonech, které uživatel s GHAM Project provádí.

Z principu fungování aplikace metody strojového učení probíhá fáze učení jednou (při spuštění aplikace), zatímco fáze predikce vícekrát (při každém dotazu na detail projektu). Aby se však zkrátil potřebný čas pro spuštění aplikace (hlavně kvůli vývoji), je naučený model serializován do souboru a při příštím spuštění je z tohoto souboru znovu načten.

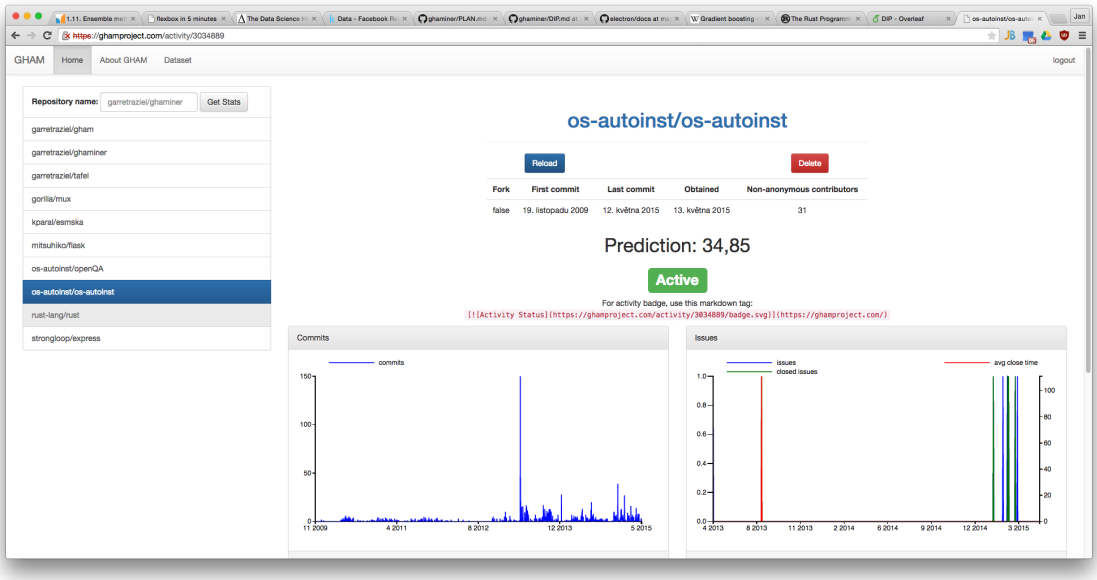
Příklad stránky detailu projektu *Express* je na obrázku 7.1.

7.2 Návrh databáze

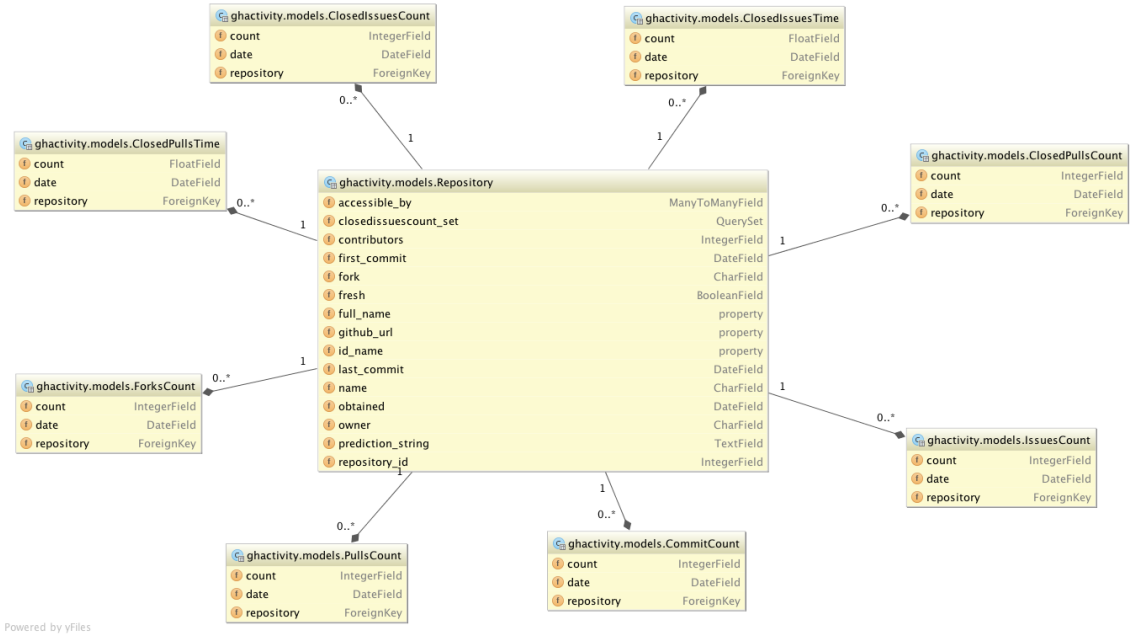
Framework Django je založen na architektuře MVC. Data aplikace jsou reprezentována pomocí objektů v Pythonu, aby je pak mohl ORM transformovat do tabulek relační databáze. Problém nastal ve chvíli, kdy nastala potřeba uložit surová data o změnách atp. Řešením je vytvořit tabulku, která jako svoje atributy obsahuje datum, ke kterému se hodnota vztahuje, samotnou hodnotu a cizí klíč jako odkaz na projekt. ER diagram tohoto modelu je na obrázku 7.2.

Tato reprezentace není ideální - u větších projektů narůstá čas, potřebný pro uložení hodnot do databáze a jejich získání. Alternativou by byla serializace hodnot do atributu typu *text* či použití vhodnějšího typu databáze, například tzv. *time series database, TSDB*.

[2] `http://www.celeryproject.org/`



Obrázek 7.1: Příklad stránky detailu projektu Express



Obrázek 7.2: ER diagram projektu

Kapitola 8

Závěr

V této práci byla představena disciplína dolování z dat včetně metod zkoumání úspěšnosti predikce, dále bylo prozkoumáno několik sociálních sítí z pohledu použití těchto sítí jako zdrojů dat pro datovou analýzu. Jako nejvhodnější byla vybrána síť GitHub, která přidává sociální vrstvu nad vývoj projektů a hostování repozitářů. Byla navrhnuta aplikace, která za pomocí dolování z dat provádí analýzu budoucí aktivity projektu. Také byly prozkoumány varianty dat dostupných z GitHubu a ve stručnosti bylo přiblíženo API služby GitHub.

Následně byl získán dataset pro metody strojového učení s učitelem, představen nástroj, který sloužil k samotnému stažení datasetu, a byly diskutovány různé způsoby určení aktivity projektu. Vlastnosti těchto definic aktivity byly představeny na třech různých příkladech s různým charakterem vývoje. Bylo vysloven důvod, proč je vhodné, aby míra aktivity byla vyjádřena spojitou veličinou.

Vlastnosti datasetu byly analyzovány v programu Weka a za pomocí knihoven z projektu SciPy. Tyto vlastnosti byly dále vizualizovány a byl diskutován předpokládaný dopad těchto vlastností na výsledný model pro predikci.

Další důležitou částí byly experimenty se získaným datasetem. Ty probíhaly nejdříve v prostředí Weka a následně za pomocí knihovny scikit-learn v prostředí IPython. Dle charakteristik byla vybrána jako nejvhodnější metoda gradient tree boosting. U všech ostatních metod byly řešeny vlastnosti jejich predikcí. Také byly představeny různé možnosti úpravy datasetu spolu s vlastnostmi, které by měl na takto upravených datasetech naučený model.

Metoda gradient tree boosting byla použita jako nástroj strojového učení ve výsledné webové aplikaci. Tato aplikace, kromě samotné hodnoty predikce budoucí aktivity zadaného projektu, zobrazuje statistiky zaslaných změn, vytvořených žádostí o začlenění, nahlášených chyb a větvení daného repozitáře. Hodnota aktivity je dále rozdělena do čtyř kategorií pro snazší orientaci. Aplikace také ke každému projektu poskytuje veřejně dostupný odznak, který je grafickou reprezentací této kategorie. Byl představen způsob přihlašování uživatelů do této aplikace přes servery služby GitHub.

První možností rozšíření této práce je využití i jiných služeb, jako je Bitbucket, CodePlex, GitLab, Launchpad či SourceForge a to jak pro kontrolu aktivity, tak pro zdrojový dataset. V rámci dalšího rozšíření by bylo vhodné zvolit jiné způsoby definice budoucí aktivity a zaměřit se více na to, jak taková definice popisuje realitu a jak jednoduše interpretovatelné jsou její hodnoty.

Další možností rozšíření této práce by bylo vytváření sociálního grafu a využití popsaného algoritmu SimRank, popřípadě jiného algoritmu pro sociální graf.

Pokud by byl zvolen jiný způsob definice budoucí aktivity projektu, bylo by vhodné blíže se zaměřit na metody pro klasifikaci.

Dále by bylo vhodné nahradit tvoření vláken programem pro distribuovanou frontou úloh.

Mezi projekty, které mají návaznost na obsah této práce, patří například projekt GitHut, který se soustředí na měření počtu aktivních repozitářů v jednotlivých jazycích v průběhu času, nebo projekt issuestats.com, který pro vybrané projekty sbírá statistiky o nahlášených chybách.

Literatura

[1] Third Annual GitHub Data Challenge. https://github.com/blog/1864-third-annual-github-data-challenge, 2014-08-22 [cit. 2015-01-03].

[2] Mining Challenge. http://2014.msrconf.org/challenge.php, 2014 [cit. 2015-01-03].

[3] Facebook. https://github.com/facebook, 2015 [cit. 2015-01-03].

[4] Google. https://github.com/google, 2015 [cit. 2015-01-03].

[5] Microsoft. https://github.com/Microsoft, 2015 [cit. 2015-01-03].

[6] @unitedstates. https://github.com/unitedstates, 2015 [cit. 2015-01-03].

[7] Borison, R.: Google+ Is Still Struggling Three Years Later. http://www.businessinsider.com/google-plus-three-years-later-2014-6, 2014-06-28 [cit. 2015-01-02].

[8] scikit-learn developers: Who is using scikit-learn? http://scikit-learn.org/stable/testimonials/testimonials.html, 2014 [cit. 2015-05-09].

[9] Ell, J.: Identifying failure inducing developer pairs within developer networks. In *Proceedings of the 2013 International Conference on Software Engineering*, IEEE Press, 2013, s. 1471–1473.

[10] Facebook: Careers at Facebook: Data and Analytics. https://www.facebook.com/careers/teams/data, 2015 [cit. 2015-01-02].

[11] Facebook: Company Info. http://newsroom.fb.com/company-info/, 2015 [cit. 2015-01-02].

[12] Grigorik, I.: Using Google BigQuery to learn from GitHub data. http://googledevelopers.blogspot.cz/2012/05/using-google-bigquery-to-learn-from.html, 2012-05-03 [cit. 2015-01-04].

[13] Gundotra, V.: Google+ Hangouts and Photos: save some time, share your story. http://googleblog.blogspot.cz/2013/10/google-hangouts-and-photos-save-some.html, 2013-10-29 [cit. 2015-01-02].

[14] Han, J.; Kamber, M.; Pei, J.: *Data Mining: Concepts and Techniques: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems, Elsevier Science, 2011, ISBN 9780123814807.

[15] Hunt, F.: Total number of Websites & Size of the Internet as of 2013. http://www.factshunt.com/2014/01/total-number-of-websites-size-of.html, 2014 [cit. 2015-01-03].

[16] inc., G.: Press. https://github.com/about/press, 2015 [cit. 2015-01-03].

[17] internetlivestats.com: Total number of Websites. http://www.internetlivestats.com/total-number-of-websites/, 2015 [cit. 2015-04-03].

[18] James, J.: How Much Data is Created Every Minute? http://www.domo.com/blog/2012/06/how-much-data-is-created-every-minute, 2012-06-08 [cit. 2015-01-03].

[19] Kaggle: Facebook Recruiting III - Keyword Extraction. https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction, 2013-07-30 [cit. 2015-01-02].

[20] Kyriakou, A.: Facebook's Graph API v2.0 kills data mining. https://github.com/ptwobrussell/Mining-the-Social-Web-2nd-Edition/issues/205, 2014-06-24 [cit. 2015-01-02].

[21] LinkedIn: O nás. https://www.linkedin.com/about-us, 2014 [cit. 2015-01-02].

[22] Mitchell, G.: HOW MANY TERABYTES OF DATA ARE ON THE INTERNET? http://www.sciencefocus.com/qa/how-many-terabytes-data-are-internet, 2013-01-23 [cit. 2015-01-03].

[23] Russell, M. A.: *Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More*. O'Reilly Media, Inc., 2013.

[24] Subramaniam, C.; Sen, R.; Nelson, M. L.: Determinants of open source software project success: A longitudinal study. *Decision Support Systems*, ročník 46, č. 2, 2009: s. 576–585.

[25] Thung, F.; Bissyandé, T. F.; Lo, D.; aj.: Network structure of social coding in GitHub. In *Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on*, IEEE, 2013, s. 323–326.

[26] Twitter: Twitter Reports Third Quarter 2014 Results. https://investor.twitterinc.com/releasedetail.cfm?ReleaseID=878170, 2014-10-27 [cit. 2015-01-02].

[27] Vellante, D.: Information Explosion & Cloud Storage. http://wikibon.org/blog/cloud-storage/, 2010-06-21 [cit. 2015-01-03].

[28] Witten, I. H.; Frank, E.: *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[29] Yoshikawa, Y.; Iwata, T.; Sawada, H.: Collaboration on Social Media: Analyzing Successful Projects on Social Coding. *arXiv preprint arXiv:1408.6012*, 2014.

Příloha A

Příklad dat z timeline Githubu

A.1 Příklad dat z Timeline API


```
{
   "created_at":"2014-05-09T15:05:24-07:00",
   "payload":{
      "shas":[
         [
            "d2e27915ff6964058bd27dc99f55a40759769a35",
            "rwatsondc@users.noreply.github.com",
            "Create DemoBranchSet.py",
            "rwatsondc",
            true
         ]
      ],
      "size":1,
      "ref":"refs/heads/master",
      "head":"d2e27915ff6964058bd27dc99f55a40759769a35"
   },
   "public":true,
   "type":"PushEvent",
   "url":"https://github.com/rwatsondc/GIS/compare/31f4eb6dc1...915ff",
   "actor":"rwatsondc",
   "actor_attributes":{
      "login":"rwatsondc",
      "type":"User",
      "gravatar_id":"151c1fdbc49af4019093df3fabdad9d5"
   },
   "repository":{
      "id":19626813,
      "name":"GIS",
      "url":"https://github.com/rwatsondc/GIS",
      "description":"Geometry, GIS, and related tools",
      "watchers":0,
      "stargazers":0,
      "forks":0,
```

```
      "fork":false,
      "size":0,
      "owner":"rwatsondc",
      "private":false,
      "open_issues":0,
      "has_issues":true,
      "has_downloads":true,
      "has_wiki":true,
      "language":"Python",
      "created_at":"2014-05-09T14:52:34-07:00",
      "pushed_at":"2014-05-09T15:05:24-07:00",
      "master_branch":"master"
   }
}
```


A.2 Příklad dat z Events API


```
{
    "id": "2489651063",
    "type":"PushEvent",
    "actor": {
        "id": 4319954,
        "login": "hermanwahyudi",
        "gravatar_id": "",
        "url": "https://api.github.com/users/hermanwahyudi",
        "avatar_url": "https://avatars.githubusercontent.com/u/4319954?"},
    "repo": {
        "id": 27826205,
        "name": "hermanwahyudi/selenium",
        "url": "https://api.github.com/repos/hermanwahyudi/selenium"},
    "payload": {
        "push_id": 536863976,
        "size": 1,
        "distinct_size": 0,
        "ref": "refs/heads/master",
        "head": "1b58dd4c4e14ea9cf5212b981774bd448a266c3c",
        "before": "20b10e3a605bd177efff62f1130943774ac07bf3",
        "commits": [{
            "sha": "1b58dd4c4e14ea9cf5212b981774bd448a266c3c",
            "author": {
                "email": "herman.wahyudi02@gmail.com",
                "name": "Herman"},
            "message": "Update README.md",
            "distinct": false,
            "url": "https://api.github.com/repos/..."}]},
    "public": true,
    "created_at": "2015-01-01T15:00:03Z"
}
```

Příloha B

Seznam položek získaného datasetu a jejich popis

`fork` je informace, zda se jedná o fork nějakého jiného repozitáře, případně jméno tohoto repozitáře,

`days_since_begin` udává počet dnů, které uplynuly od data prvního zaslání změny,

`days_since_last_commit` udává počet dnů, které uplynuly od data posledního zaslání změny,

`commits_count` udává celkový počet změn v tomto repozitáři,

varianty `commits_f` udávají denní frekvenci změn za poslední týden, měsíc, půlrok, rok a celou dobu od začátku vzniku repozitáře,

`issues_count` udává celkový počet nahlášených chyb v tomto repozitáři,

varianty `issues_f` udávají denní frekvenci hlášených chyb za poslední týden, měsíc, půlrok, rok a celou dobu od začátku vzniku repozitáře,

`closed_issues_count` udává celkový počet zavřených (vyřešených) chyb,

varianty `closed_issues_f` udávají denní frekvenci zavřených (vyřešených) chyb za poslední týden, měsíc, půlrok, rok a celou dobu od začátku vzniku repozitáře,

varianty `closed_issues_time` udávají průměrnou dobu od nahlášení chyby do zavření chyby v dnech za poslední týden, měsíc, půlrok, rok a celou dobu od začátku vzniku repozitáře,

`comments_count` udává celkový počet komentářů k nahlášeným chybám,

varianty `comments_f` udávají denní frekvenci komentářů k chybám za poslední týden, měsíc, půlrok, rok a celou dobu od začátku vzniku repozitáře,

`pulls_count` udává celkový počet požadavků o začlenění (žádostí o začlenění kódu),

varianty `pulls_f` udávají denní frekvenci požadavků o začlenění za poslední týden, měsíc, půlrok, rok a celou dobu od začátku vzniku repozitáře,

`closed_pulls_count` udává celkový počet začleněných požadavků o začlenění,

varianty `closed_pulls_f` udávají denní frekvenci začleněných požadavků o začlenění za poslední týden, měsíc, půlrok, rok a celou dobu od začátku vzniku repozitáře,

varianty `closed_pulls_time` udávají průměrnou dobu od vytvoření požadavku o začlenění do jeho začlenění v dnech za poslední týden, měsíc, půlrok, rok a celou dobu od začátku vzniku repozitáře,

`pulls_comments_count` udává celkový počet komentářů k požadavkům o začlenění,

varianty `pulls_comments_f` udávají denní frekvenci komentářů k požadavkům o začlenění za poslední týden, měsíc, půlrok, rok a celou dobu od začátku vzniku repozitáře,

`events_count` udává celkový počet událostí, které se udály u zadaného repozitáře,

varianty `events_f` udávají denní frekvence všech událostí repozitáře za poslední týden, měsíc, půlrok, rok a celou dobu od začátku vzniku repozitáře,

`contrib_count` udává celkový počet autorů - lidí, kteří mají alespoň jednu změnu v historii tohoto repozitáře a kteří jsou zároveň jednoznačně identifikovatelní (mají správně nastavené informace o GitHub účtu),

`contrib_others` udává celkový počet ostatních autorů (těch, kteří mají nesprávně nastavené informace o GitHub účtu v jimi zaslaných změnách),

varianty `contrib_p` udávají počet lidí, kteří jsou autory alespoň 25 %, 50 % a 75 % změn a to za poslední týden, měsíc, půlrok, rok a celou dobu od začátku vzniku repozitáře,

`ccoments_count` udává celkový počet komentářů k zaslaným změnám,

varianty `ccomments_f` udávají denní frekvenci komentářů k zaslaným změnám za poslední týden, měsíc, půlrok, rok a celou dobu od začátku vzniku repozitáře,

`forks_count` udává celkový počet forků repozitáře a

varianty `forks_f` udávají denní frekvenci vzniku forků repozitáře za poslední týden, měsíc, půlrok, rok a celou dobu od začátku vzniku repozitáře.

Příloha C

Postup zprovoznění implementované webové aplikace

Pro zprovoznění výsledné webové aplikace je vhodné použít operační systém unixového typu (například Linux). Hlavním požadavkem je interpreter jazyka Python (dostupný při použití balíčkovacího systému či na stránkách `https://www.python.org/`). Aplikace je psána pro Python verze 2. Volitelným, avšak vhodným nástrojem je program virtualenv (`https://virtualenv.pypa.io/en/latest/`) pro lokální správu knihoven jazyka Python. Postup zprovoznění je následující:

1. Vytvořte složku se soubory pro virtualenv:

   ```
   virtualenv ~/gham_venv
   ```

2. Aktivujte virtualenv:

   ```
   source ~/gham_venv/bin/activate
   ```

3. Nainstalujte závislosti programu GHAM Project. Ze složky s projektem spusťte:

   ```
   pip install -r requirements.txt
   ```

4. Vytvořte databázi:

   ```
   python manage.py makemigrations
   python manage.py migrate
   ```

5. Na serveru GitHub na adrese `https://github.com/settings/developers` přidejte novou aplikaci. Hodnotu `Client ID` zkopírujte do souboru `gham/settings.py` do proměnné `SOCIAL_AUTH_GITHUB_KEY` a `Client Secret` do proměnné `SOCIAL_AUTH_GITHUB_SECRET`.

6. Spusťte program pomocí:

   ```
   python manage.py runserver
   ```

7. Server bude běžet na adrese `http://localhost:8000`.