Mendelova univerzita v Brně
Provozně ekonomická fakulta

Aproximace polohy mobilního zařízení pomocí známé sítě Wi-Fi přístupových bodů.

Diplomová práce

Vedoucí práce:
Ing. Jan Kolomazník, Ph.D.

Bc. Ondřej Kalman

Brno 2017

Chtěl bych poděkovat vedoucímu své práce Ing. Janu Kolomazníkovi, Ph.D. za odborné vední mé práce a za ochotu a rady, které mi při zpracování této práce poskytnul. Dále bych rád poděkoval Ing. Luboši Juránkovi, Bc. Angelice Činčárové a Lukáši Halamovi za pomoc při realizaci praktické části této práce.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Aproximace polohy mobilního zařízení pomocí známé sítě Wi-Fi přístupových bodů.**
vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*
Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.
Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně, dne 1. ledna 2017 .....

Abstract

Kalman O. *Approximation of a mobile device position using known network of Wi-Fi access-points* Diploma thesis. Brno 2017

Goal of this thesis is to create an online service, suitable for approximation of user's position inside a building via his mobile device. For this goal a known network of Wi-Fi access-points was used in combination with a machine learning algorithms. Algorithms were trained and tested on the ground floor in building Q of Mendel University in Brno. The Algorithms and the online service were tested with automated tests as well as in real environment with a mobile application, which was also created as a part of this thesis. Results have shown that reliability and accuracy of service is highly dependent on localized mobile device. Success rate floated between 43 % and 66 %, depending on a mobile device.

Keywords: localization, Android, cloud, machine learning

Abstrakt

Kalman O. *Aproximace polohy mobilního zařízení pomocí známé sítě Wi-Fi přístupových bodů.* Diplomová práce. Brno 2017

Cílem této práce je vytvořit online službu pro aproximování polohy uživatele uvnitř budovy pomocí mobilního zařízení a existující infrastruktury Wi-Fi vysílačů. Zvolený problém je vyřešen pomocí algoritmů strojového učení, které byly natrénovány pro použití v přízemním patře budovy Q Mendelovy Univerzity v Brně. Použité algoritmy i služba samotná poté byly otestovány jak automatizovanými testy, tak i pomocí demonstrační aplikace, která byla v rámci této práce rovněž vytvořena. Testy ukázaly, že přesnost a spolehlivost aproximace záleží na zařízení použitém k lokalizaci. Úspěšnost aproximace se pohybovala mezi 43 % a 66 %, v závislosti na použitém mobilním telefonu.

Klíčová slova: určování polohy, Android, internetové služby, strojové učení

Obsah

1 Úvod **7**

2 Cíl a metodika práce **8**
2.1 Cíl . . . 8
2.2 Metodika práce . . . 8

3 Teoretický základ **9**
3.1 Řešení postavená na standardech 802.11 . . . 9
Problematika . . . 9
Regresní a klasifikační algoritmy . . . 10
SVM . . . 10
Neuronové sítě . . . 11
K-NN . . . 12
C5.0 . . . 13
Bayesovská klasifikace . . . 14
Numerické algoritmy . . . 15
Kombinace numerických a regresních algoritmů a Bayesovské sítě . . . 16

4 Návrh a architektura **17**
4.1 Návrh celku . . . 17
4.2 Serverová část . . . 17
Webový servlet . . . 17
IBM Predictive Analysis (IBM Watson) . . . 18
4.3 Vývoj aplikací pro operační Systém Android . . . 18
Android SDK . . . 19
Manifest . . . 22
Uživatelské rozhraní . . . 23
Asynchronní operace . . . 24
Skenování okolních Wi-Fi sítí . . . 25

5 Vývoj a implementace lokalizační služby **26**
5.1 Sběr dat . . . 26
Sbírání dat pomocí mobilní aplikace . . . 26
Transformace dat . . . 28
5.2 Trénování algoritmů . . . 30
Vstupní uzly . . . 32
Transformační uzly . . . 32
Uzly pro zpracování výsledků a jejich analýzu . . . 34
Uzel klasifikačního algoritmu K-NN . . . 35
Uzel klasifikačního algoritmu C5.0 . . . 36
5.3 Serverová část . . . 36
JSON API . . . 37

Zpracování a vyhodnocování dotazů 38

6 Testování **40**
6.1 Laboratorní testy 40
6.2 Reálné testy 44
Metodika testování 44
Analýza dat 46
Zhodnocení testů 49

7 Závěr **51**

8 Reference **52**

1 Úvod

Tato diplomová práce se zabývá problematikou určování polohy uvnitř budov, tedy tam, kde není možno využít signál GPS. K tomuto účelu bude jako alternativní zdroj signálu využita síť vysílačů Wi-Fi sítě. Potřeba lokalizovat uživatele uvnitř budovy vzrostla s nástupem moderních mobilních telefonů, označovaných jako chytrý telefon (anglicky „SmartPhone“).

Technologie mobilních zařízení zaznamenala za posledních deset let nečekaně rychlý nárůst. Dříve velice drahá, ne příliš výkonná zařízení se přetransformovala do podoby výkonných komunikačních nástrojů cenově dostupných široké veřejnosti. Chytré telefony, tablety, lehké a snadno přenositelné počítače umožnily vznik mnoha novým službám. Řada aplikací a služeb využívá senzorů, které jsou součástí těchto zařízení. Snad nejrozšířenějším je čip pro příjem signálu GPS, ten umožňuje s přesností na několik metrů určit polohu mobilního telefonu kdekoli pod otevřeným nebem. Přístup k poloze umožňuje aplikacím nabízet uživateli relevantnější obsah (pokud je to žádoucí), nebo nabízet navigační služby pro snadnější orientaci. GPS signál je ale příliš slabý na to, aby pronikl do interiérů budov, tunelů a někdy i do husté městské zástavby. V takových případech je nutné využít jiné technologie, které dokáží zařízení lokalizovat.

Zařízení s operačním systémem android byla již dříve použita pro sběr dat společností Google. Komunita uživatelů pohybujících se ve městech sbírala o Wi-Fi vysílačích a zasazovala je do lokalizovaného kontextu pomocí GPS. Uživatelé potom mohli využívat lokalizace pomocí okolních Wi-Fi sítí. Zmiňovaná technologie se nemůže z hlediska přesnosti a spolehlivosti rovnat technologii GPS, nicméně pro spoustu aplikací může být přesnost dostatečná a rychlost přibližné lokalizace je podstatně vyšší, což může být pro mnohé služby důležitější.

2 Cíl a metodika práce

2.1 Cíl

Cílem této práce je nalézt vhodné metody pro aproximaci polohy unitř budovy s využitím Wi-Fi sítě a cloudových technologií IBM Bluemix. Tyto metody poté použít k vytvoření funkční lokalizační služby použitelné mobilními telefony. Dále je třeba vytvořit demonstrační aplikaci, na které bude možno lokalizační službu vyzkoušet a testovat.

2.2 Metodika práce

Prvním krokem je najít v konzoli IBM Bluemix služby vhodné pro provoz serverové aplikace a služby, které nám budou poskytovat aproximační metody pro naše řešení. Potom je potřeba tyto služby v konzoli spustit a propojit. V našem případě budou spuštěny následující služby:

- Aplikační server Liberty Java
- IBM Watson
- MySQL databáze

V následujícím kroku vytvoříme aplikaci pro Android, která nám bude sloužit pro sběr dat. V případě této práce bude obsahovat interaktivní mapu, rozdělenou na malé čtverce. Pomocí aplikace potom můžeme začít sbírat data v jednotlivých čtvercích na mapě.

Dále vytvoříme v programu IBM SPSS Modeler model, který bude obsahovat algoritmy pro aproximaci polohy. V případě této práce budou použity klasifikační algoritmy.

V následujícím kroku, data nasbíraná mobilní aplikací použijeme k natrénování klasifikačních algoritmů. Tento model poté nahrajeme do služby IBM Watson. Nasbíraná data také nahrajeme do MySQL databáze, kterou jsme si vytvořili na začátku. Není potřeba nahrávat všechna. Postačí jen hlavička.

Dále vytvoříme serverovou aplikaci, která bude poskytovat aplikační rozhraní pro online dotazování mobilními klienty, a která je bude překládat a přeposílat službě IBM Watson. Při překladu bude využita hlavička, kterou jsme vytvořili v MySQL databázi.

Poté do mobilní aplikace doimplementujeme funkcionalitu dotazování na server tak, aby výsledky mohly být prezentovány na zobrazené mapě.

Dalším krokem je testování funkcionality celého řešení. První budou otestovány klasifikační algoritmy pomocí dat, která byla získána pro trénink. Následně jsou provedeny testy v reálném prostředí.

Na závěr jsou zhodnoceny výsledky testování a použitelnost našeho řešení.

3 Teoretický základ

Určování polohy v zastřešených prostorách se v posledních letech stalo velkým tématem. Zásluhu na tom mají především moderní chytré telefony, ty nejen že obsahují hardware, jenž může posloužit tomuto účelu, ale především mohou spouštět aplikace, které ze znalosti polohy uživatele mohou těžit. Ačkoli se může na první pohled zdát tato problematika značně triviální, opak je pravdou. Důkazem komplexnosti tohoto problému je množství technologií a řešení, kterých se v praxi používá. Každé z řešení má své výhody i nevýhody, některá jsou si podobná a jiná jsou naopak velmi originální. Všechna řešení ovšem mají podobné limity a určení toho nejlepšího není snadným řešením. Jako u každého takového hodnocení, vše záleží na zvolených kritériích a nastavení jejich priorit.

3.1 Řešení postavená na standardech 802.11

Standardy 802.11 jsou známější pod názvem Wi-Fi. Wi-Fi je z principu komunikační technologie určená k bezdrátovému přenosu dat. Nicméně rozšířenost infrastruktury umožňuje i jiná využití. Jedním z nich je i lokalizování připojených uživatelů v prostoru. K tomuto účelu slouží především RSSI(*Received Signal Strength Indicator*), což je jeden z parametrů bezdrátového připojení, který je znám jak klientům připojeným do sítě, tak síťovým prvkům (dále v textu označovaných jako AP z anglického slova Access-Point), které signál šíří. Hodnota RSSI popisuje sílu signálu mezi AP a klientem. Způsobů, jak s hodnotami RSSI pracovat a s jejich pomocí určit přibližnou polohu klienta, je mnoho. Některé z nich budou představeny v následujících podkapitolách.

Problematika

Tato technologie využívá infrastruktury bezdrátových přístupových bodů sítě Wi-Fi nebo infrastruktury majáků využívajících technologii bluetooth, které jsou rozmístěny na předem zmapovaných místech uvnitř budovy. Mobilní telefon nebo jiné podobné zařízení v pravidelných intervalech získává informace o dostupných AP v okolí. Pro určení polohy se poté typicky používá jeden ze dvou přístupů, triangulace polohy nebo porovnání „otisků“[1] na předpřipravené mapě s otiskem získaným z lokalizovaného zařízení.

Triangulace polohy ze signálu několika dostupných přístupových bodů je ve své podstatě značně naivní metodou, protože nelze dost dobře globalizovat míru útlumu signálu procházejícího různými materiály. Nespornou výhodou potom je velice rychlá možnost nasazení, protože tato metoda nevyžaduje kromě mapy budovy s vyznačenými AP žádné jiné prerekvizity.

[1] Ostiskem (anglicky finger print) je nazýván záznam, obsahující informaci o poloze a seznam naměřených RSII z okolních přístupových bodů

Sofistikovanější metody, které pracují s mapou otisků, vyžadují její vytvoření před tím, než lze lokalizační systém nasadit. Mapa budovy se rozdělí na dlaždice konstantní velikosti a na každé dlaždici se poté změří hodnota RSSI pro dostupné přístupové body. Jak uvádí (Elnahrawy, Li a Martin, 2004), vytvoření mapy otisků pro každou dlaždici na mapě je časově náročné. Pro snížení časové zátěže můžeme využít interpolačních metod a mezi měřenými dlaždicemi hodnoty dopočítávat. Interpolovaná mapová mřížka nabízí oproti triangulaci výhodu reálných dat, jejichž obsah je ovlivněn reálnými okolními podmínkami.

V posledních letech bylo na téma lokalizace uvnitř budov zpracováno mnoho studií a byla navržena různá řešení. Většina z nich využívá druhého ze zmíněných postupů. Ve studiích byly zkoumány různé regresní algoritmy a jejich výsledky v reálném použití a také celkové limity těchto algoritmů, a to z hlediska přesnosti lokalizace a pravděpodobnosti správného určení polohy. Ve studii „The Limits of Localization Using Signal Strength: A Comparative Study“ (Elnahrawy, Li a Martin, 2004) autoři uvádí, že je možné dosáhnout mediánu chyby přibližně 3 metry a stanovení rádiusu 9 metrů s pravděpodobností výskytu uvnitř s pravděpodobností 97 %.

Úspěšnost lokalizace v reálném prostředí může být částečně ovlivněna různou citlivostí přijímačů, stejně jako různou konstrukcí telefonů, tabletů či notebooků.

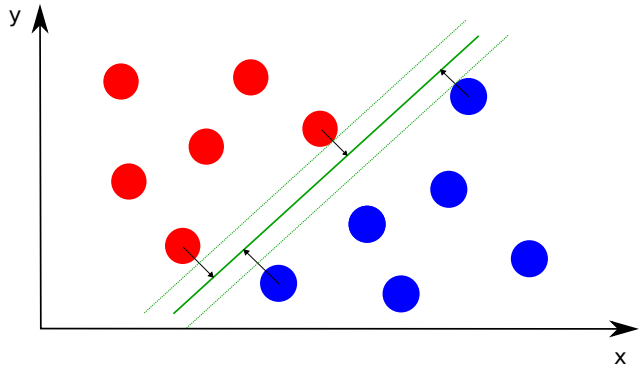
Regresní a klasifikační algoritmy

Takto jsou označovány algoritmy strojového učení, které se učí s učitelem. Při řešení problému lokalizace lze využít řadu z nich. Populární je využití algoritmu K-NN nebo Bayesovských sítí. Řešení pomocí regresních algoritmů jsou často implementována v kombinaci s interpolovanou mapovou mřížkou tak, aby vymezovaly oblast pravděpodobného výskytu. Rozdíl mezi regresním a klasifikačním algoritmem je, že regresní algoritmus pracuje se spojitou veličinou, kterou se pro jednotlivé výsledky snaží aproximovat. Naproti tomu klasifikační algoritmy se snaží rozčlenit výsledky do jedné z předem známých skupin.

SVM

Regresní algoritmus SVM (*support vector machine*) funguje na principu rozdělení prostoru pomocí nadroviny, kde na jedné straně nadroviny jsou prvky, které patří do jedné kategorie, a na druhé straně nadroviny jsou prvky, které patří do druhé kategorie. Jak uvádí (Amari a Wu, 1999), u vícedimenzionálních prostorů nemusí být konstrukce takovéto nadroviny možná, protože se data v některé z dimenzí příliš prolínají. Tento problém je řešen tzv. jádrovými funkcemi, pomocí kterých je prostor transformován tak, aby konstrukce nadroviny možná byla.

Z předchozího odstavce tedy vyplývá, že SVM algoritmus umožňuje klasifikaci pouze do dvou tříd. Způsob jakým je algoritmus využit pro lokalizaci, je popsán v článku „Localization In Wireless Sensor Networks based on Support Vector Machines“ následovně. SVM je trénován zvlášť pro každou z os v prostoru (typicky používáme 2D prostor jakožto mapu). Algoritmus SVM poté klasifikuje, zda bod

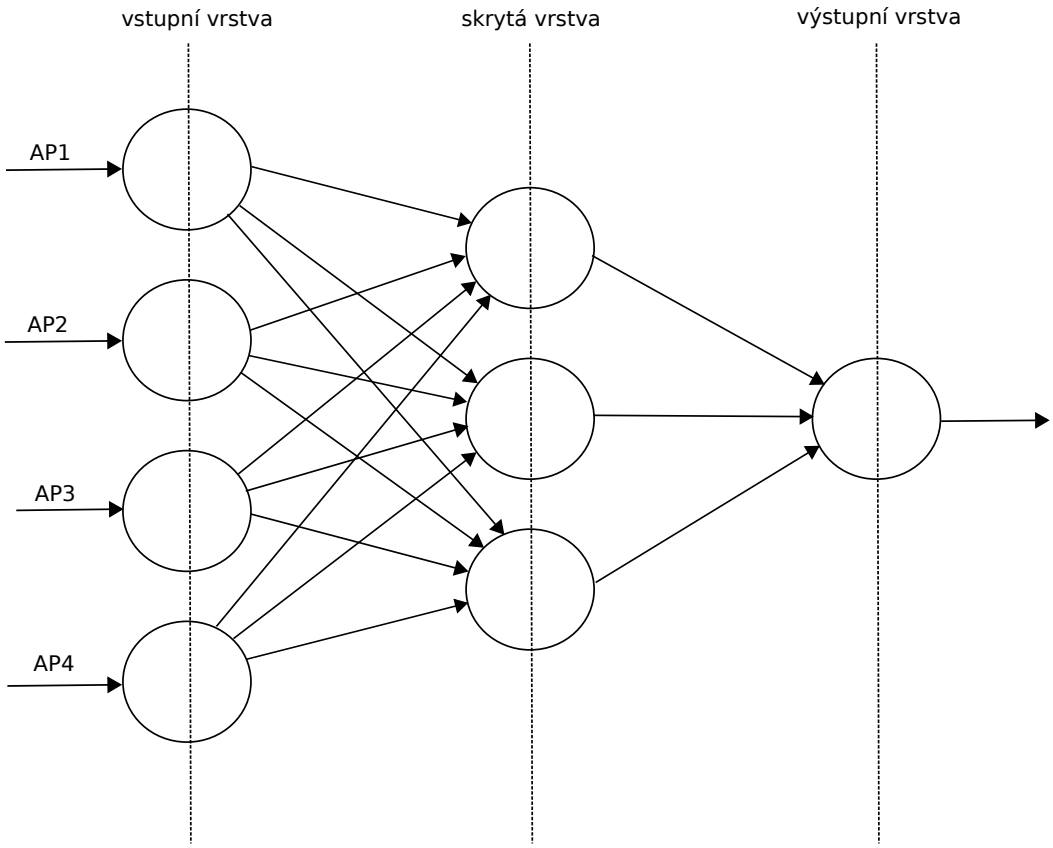


Obrázek 1: Ukázka rozdělení objektů nadrovinou algoritmem SVM

M leží na ose X vpravo od aktuálně zkoumané souřadnice x či nikoli a zároveň se zkoumá, zda bod M leží na ose Y nad aktuálně zkoumanou souřadnicí y. Na každé z os hledáme tedy právě dvě souřadnice, mezi kterými došlo ke změně ve výsledku klasifikace. Poslední známá souřadnice x která byla vyhodnocena kladně („je napravo od x“) a y souřadnice vyhodnocená kladně („je nad y“) je přiřazena bodu M jako jeho aktuální poloha. (Tran a Nguyen, 2008)

Neuronové sítě

Běžně používaným modelem neuronové sítě je takzvaný „vícevrstvý perceptron“. Základní vlastností takovéto neuronové sítě je, že neurony ve stejné vrstvě mezi sebou nejsou nijak propojeny. Propoje jsou pouze mezi neurony v různých vrstvách. Architektura je složena ze vstupní a výstupní vrstvy a libovolného počtu skrytých vrstev.



Obrázek 2: Architektura neuronové sítě vícevrstvého perceptronu

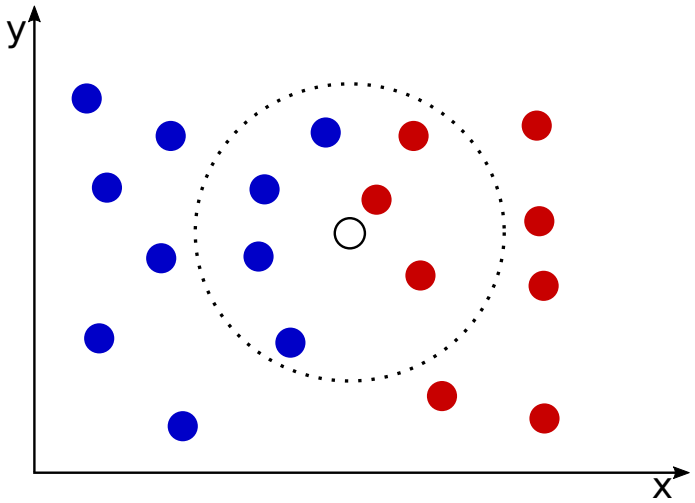
Nevýhodou použití neuronových sítí je, že vstupní vrstva očekává informaci o všech AP, které jsou v interiéru zapojeny. Klienti většinou nemají informace o všech AP, ale jen o těch, které jsou aktuálně v dosahu. Proto je možné použít modulárního přístupu. Jak uvádí (Ahmad, Gavrilov a Lee, 2011), princip modularizace spočívá ve vytvoření mnoha malých neuronových sítí pro různé kombinace dosažitelných AP. V architektuře je zahrnut jeden klasifikátor, který rozhoduje o tom, jaký modul bude použit.

K-NN

Jak uvádí (Brownlee, 2016), algoritmus K-NN (k-nearest neighbours) může pracovat jako regresní i jako klasifikační algoritmus, dle toho, jaký výsledek od něj požadujeme. Algoritmus K-NN očekává na vstupu vektor souřadnic. Každá komponenta vektoru, reprezentuje souřadnici bodu na jedné ose v prostoru. Vstupní data, tedy musí mít pokáždé stejný počet komponent, v opačném případě by nebylo možné umístit bod do prostoru. Počet komponent vektoru není nijak omezen, algoritmus K-NN je tedy schopen pracovat v n-dimenzionálním prostoru. V klasifikačním problému, jako je určení správného políčka na mapě dle síly signálu okolních Wi-Fi vysílačů, pracuje algoritmus následujícím způsobem:

1. Umístí klasifikovaný bod do prostoru, dle vektoru jeho souřadnic
2. Najde k nejbližších sousedů
3. Třída, která mezi nalezenými sousedy převažuje, je přidělena našemu klasifikovanému bodu

Důležitou konstantou je mnohokrát zmiňované k, které je i součástí jména algoritmu. Jedná se o konstantu, která určuje kolik sousedů se při klasifikaci bude hledat. Postup pro správné zvolení této konstanty prakticky neexistuje, vždy záleží na tom, kolik tříd máme předdefinováno, a jak jsou v prostoru jejich prvky rozmístěny. Většinou se chceme vyhnout volbě, kdy k=2, protože šance, že dva sousedi budou pocházet z různých tříd, je poměrně velká. Naštěstí většina moderních frameworků implementující tento algoritmus disponuje automatickým režimem, kdy je tréninková množina dat analyzována a k je určeno automaticky tak, aby algoritmus pracoval s co největší přesností, podle (Statistica, 2016) se typicky používá metoda křížové validace. Výpočet vzdálenosti od svých sousedů může být implementován různě, klasicky je to euklidovská nebo manhattanská vzdálenost.



Obrázek 3: Neobarvené kolečko uprostřed rádiusu by bylo dle algoritmu K-NN vybarveno modře

C5.0

Algoritmus C5.0 patří do kategorie klasifikačních. Jeho základem je budování takzvaného rozhodovacího stromu. Rozhodovací strom je binární strom, jehož uzly tvoří pravidla. V rozhodovacím stromu algoritmu C5.0 je pravidlo vytvořeno na základě hodnoty jednoho z atributů, atribut je zvolen tak, aby byl informační zisk po rozdělení do dvou potomků co největší. Pro správné pochopení je tedy nutné definovat dva pojmy:

1. Informační entropie, značí se písmenem H a udává míru neurčitosti v datech, tedy jak moc se jeví data jako náhodná. Je dána rozložením pravděpodobnosti P všech možných možností s z množiny S, dle následujícího vzorce 1.

$$H = -\sum_{s \in S} P(s\ ) \cdot log\,(P(s\ )) \tag{1}$$

2. Informační zisk (information gain), značí se jako IG. Vyjadřuje, jak moc se sníží míra entropie v datech, pokud je rozdělíme na základě hodnoty některého z atributů v datech. Při rozdělování se tedy snažíme rozdělit data podle atributu (a jeho hodnoty) s co největším informačním ziskem. Toto rozdělení je pak nazýváno „pravidlo“. Obecný vzorec pro výpočet informačního zisku 2 tedy vychází ze změny entropie v datech T po aplikování pravidla a.

$$IG\,(T, a\ ) = H\,(T) - H\,(T \mid a) \tag{2}$$

Pravidla jsou takto řetězena do stromu pod sebe ve formě uzlů pro všechny ostatní atributy. Po vytvoření rozhodovacího stromu je v určitých případech nutné ho prořezat, to znamená, sloučit některé podstromy dohromady. Tato úprava sice zhorší úspěšnost algoritmu pro trénovací data, ale zvýší úspěšnost pro data reálná. Prořezávání se provádí proto, aby strom lépe zachycoval obecnější vztahy v datech.

Jak uvádí (Bujlow, Riaz a Pedersen, 2012), algoritmus C5.0 disponuje oproti starší verzi C4.5 a jiným algoritmům na bázi rozhodovacích stromů ještě jednou zásadní výhodou, podporuje takzvaný „boosting“. Při použití boostingu se zanalyzují výsledky původního rozhodovacího stromu a vytipují se případy, kde dochází k nejvíce chybám. Poté je vytvořeno několik nových rozhodovacích stromů, přičemž každý z nich je vygenerován tak, aby se zlepšil v rozhodování pro jeden z vytipovaných případů, při tom se ale může zhoršit jeho úspěšnost pro jiné. Tento proces vede k tomu, že ve výsledku vedle sebe existuje několik mírně se lišících stromů. Při klasifikaci poté rozhodují všechny společně formou hlasování. Metoda boostingu může snížit chybovost algoritmu až o 25%.

Bayesovská klasifikace

Bayesovská klasifikace je založena na pravděpodobnostním modelu, ve kterém se snažíme kvantifikovat pravděpodobnost platnosti hypotéz při pozorování možných souvisejících jevů. Základem Bayesovské klasifikace je Bayesův teorém 3. P(h) je pravděpodobnost hypotézy h bez znalosti jakýchkoli návazností na možné jevy. P(D) je pravděpodobnost pozorování jevu D bez znalosti jakékoli návaznosti na hypotézu h. $P(D \mid h)$ je pravděpodobnost zpozorování jevu D v případě, že platí hypotéza h. $P(h \mid D)$ je pravděpodobnost hypotézy h při pozorování jevu D.

$$P\,(h \mid D) = \frac{P\,(D \mid h) \cdot P\,(h)}{P\,(D)} \tag{3}$$

Následující příklad bude ilustrovat jednoduchý klasifikační problém, na jehož základě dojdeme k pravděpodobnostem jednotlivých hypotéz: Mějme hypotézu h_1, která říká, že na chodník před méně jak čtvrt hodinou pršelo. Mějme také hypotézu h_2, která říká, že po chodníku jel uklízecí vůz. Řekněme, že pravděpodobnost že vyjdeme z domu na chodník po tom, co na něj právě naprší je 20 %, tedy P(h_1)=0,2. Řekněme také, že pravděpodobnost toho, že vyjdeme z domu na chodník po tom, co byl uklizen čistícím vozem, je mnohem menší, protože čistící vozy samy o sobě příliš často nejezdí, bude tedy jen 1 %, P(h_2)=0,01. Nyní je potřebovat nadefinovat množinu jevů, které bychom s našimi hypotézami mohli spojit. Takovým jevem může být třeba mokrý chodník. Nyní pro něj musíme stanovit pravděpodobnost P(D), což je pravděpodobnost, že při vstupu na chodník bude mokrý, ta bude třeba 25 %. Dále je potřeba definovat pravděpodobnosti pozorování tohoto jevu při platnosti našich hypotéz. $P(D \mid h_1)$, což je pravděpodobnost, že když před chvílí pršelo, tak bude mokrý chodník. Zvolme pravděpodobnost 90 %, protože se může stát, že chodník usušilo slunce. To stejné musíme udělat i pro čistící vůz, tedy $P(D \mid h_2)$, zde řekněme, že bude pravděpodobnost 80 %. Nyní již můžeme spočítat pravděpodobnosti našich hypotéz, výsledky jsou uvedeny v rovnicích 4 a 5.

U předchozího příkladu je důležité poznamenat, že veškeré definice pravděpodobností byly pro tento příklad vymyšlené. Při řešení reálného problému by bylo potřeba tato data nasbírat, tedy mít před domem kameru a zaznamenávat si při každé vycházce z domu jaké bylo počasí, jaký byl stav chodníku a tak dále. U ilustrovaného příkladu si také můžeme všimnout, že součet pravděpodobností se nerovná 1. Bayesovský klasifikátor totiž hodnotí každou hypotézu samostatně, bez ohledu na ostatní. Může se tak stát, že nám vyhodnotí více hypotéz, které budou mít součet pravděpodobností větší než 1. Důležité je také upozorni na fakt, že bayesovský klasifikátor nezavrhuje ani jednu z hypotéz, byť málo pravděpodobnou, je jen na nás jak s výsledky naložíme. V našem případě bychom podpořili hypotézu č.1.

$$P(h_1 \mid D) = \frac{0,9 \cdot 0,2}{0,25} = 0,72 \tag{4}$$

$$P(h_2 \mid D) = \frac{0,8 \cdot 0,01}{0,25} = 0,032 \tag{5}$$

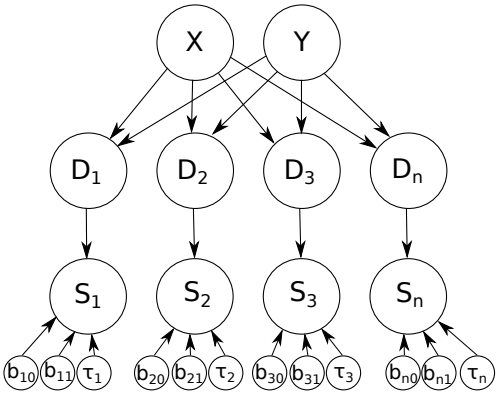
Numerické algoritmy

Numerickými algoritmy jsou označeny algoritmy, které pracují výhradně s aktuálně dostupnými daty zachycenými bezdrátovým přijímačem. Pro výpočet polohy využívají mapu přístupových bodů, hodnoty RSSI a matematický model propagace signálu. Pomocí modelu propagace signálu poté můžeme aproximovat vzdálenost zařízení od AP a využít metody triangulace při určování polohy.

Kombinace numerických a regresních algoritmů a Bayesovské sítě

Některé přístupy řešící aproximaci polohy mohou využívat obou přístupů zároveň. Regresní algoritmy nejprve vytipují několik AP, jejichž RSSI má být k lokalizaci použito a poté se pomocí numerického modelu vypočítá přibližná poloha. (Brunato a Kiss Kallo, 2002)

Kombinace matematického modelu a učení s učitelem jsou součástí řešení využívajících Bayesovských sítí. Bayesovské sítě jsou pravděpodobnostní modely, pracující s grafovou reprezentací. Jak uvádí (Brunato a Kiss Kallo, 2002), pro výpočet polohy je využíváno pravděpodobnostního modelu, který v sobě zahrnuje i nějakou formu matematického modelu propagace signálu. Regresí se poté postupně nastavují koeficienty modelu propagace tak, aby pro trénovací data vracela Bayesovská síť správné souřadnice.



Obrázek 4: Architektura Bayesovské sítě pro N AP

4 Návrh a architektura

Tato kapitola a následující podkapitoly obsahují popis implementovaného řešení, jeho návrhu a architektury. Také zde budou uvedeny ukázky řešení, a to jak formou útržků zdrojových kódů, tak obrázků.

4.1 Návrh celku

Lokalizační řešení, které je v této práci implementováno, se skládá ze dvou oddělených částí. První z nich je klientská část, která běží na koncových zařízeních. Druhá část je tzv. "backend", to je serverová část, která běží na vzdáleném serveru. Obě strany spolu komunikují přes internet pomocí jednoduchého JSON API. Při návrhu bylo myšleno na to, aby klientská zařízení nebyla zbytečně zatěžována jak z hlediska výpočetního výkonu, tak z hlediska množství přenášených dat. Proto jsou na server odesílány jen předem přefiltrované informace.

Důvody k použití výše uvedeného návrhu jsou:

1. Náročnost na výpočetní výkon a s tím spojená náročnost na baterii mobilních zařízení.
2. Snadná udržovatelnost a upravitelnost dat, podle kterých se lokalizuje.
3. Rychlost odezvy. Výkon serveru je nesrovnatelně vyšší v porovnání s telefony.
4. Možnost použití služeb jako je IBM Watson

4.2 Serverová část

Serverová část se skládá ze dvou dílčích podčástí, které jsou na sobě nezávislé a v případě nutnosti nemusí být spuštěny u jednoho poskytovatele. První z nich je servletová podčást, jež poskytuje API pro mobilní klienty, je implementována v Javě a může být spuštěna na libovolném serveru, který poskytuje Java servlet služby. Druhou je služba poskytující přístup k natrénovaným klasifikačním algoritmům, které zajišťují výpočet polohy. V případě této práce je to služba IBM Predictive Analysis.

Webový servlet

Úkolem servletu je poskytování jednotného API pro klienty, překlad jejich dotazů a jejich přeposílání službě IBM Predictive Analysis, kde je spuštěna služba, která provádí onu lokalizaci. Následně jsou data opět upravena a zmenšena a poslána klientovi zpět.

Tato architektura umožňuje kontrolu nad množstvím poslaných dotazů v čase, také není nutné mít na všech klientech uloženy tajné klíče, které jsou nutné pro komunikaci mezi IBM Predictive Analysys serverem a dotazujícím se zařízením. Pokud je na stranách klientů použité doménové jméno, na kterém se servlet nachází, je

také možné tuto část bez větších obtíží migrovat od jednoho poskytovatele k jinému. Stejně tak je možné za běhu vylepšovat či ladit servlet, případně měnit cílovou adresu serveru, na kterém běží prediktivní model. Všechny tyto modifikace mohou být prováděny za běhu s minimálním dopadem na funkčnost klientských zařízení. V neposlední řadě takové řešení umožňuje návrh vlastního API, jehož specifikace může být v této práci popsána, a které poté může být implementováno na libovolném klientském zařízení různými programátory.

Implementace klasifikačních algoritmů vyžaduje, aby byl vektor vstupních dat vždy stejný, z tohoto důvodu musí být dotazy na vzdálený server upravovány do podoby, která odpovídá podobě, na níž byl algoritmus v minulosti trénován. Servlet tedy musí mít přístup k databázi dat, na kterých byl prováděn trénink a automaticky analyzovat podobu natrénovaného vektoru a transformovat příchozí data, do použitelného vstupu.

IBM Predictive Analysis (IBM Watson)

IBM Predictive Analysis je služba poskytovaná firmou IBM v rámci konzole IBM Bluemix. Tato služba je součástí programu Watson, což je soubor funkcionalit a služeb zaměřených na strojové učení. Abychom mohli zpracovávat data o okolních Wi-Fi sítích, je potřeba vytvořit prediktivní model, který na to bude připravený.

Vytváření prediktivního modelu probíhá v programu IBM SPSS Modeler, který nejenže umožňuje vytvářet prediktivní modely a následně i trénovat v něm zabudované algoritmy strojového učení, ale umožňuje i nad těmito natrénovanými algoritmy provádět dotazy. Základem takového modelu je definice vstupních dat, tedy to, jak vstupní data vypadají a jaká je jejich sémantika (datový typ), poté se data přivedou na vstup natrénovaných algoritmů, ty provedou klasifikaci a na výstupu uvedou výsledek a odhad s jakou jistotou je výsledek správně. Model v IBM SPSS modeleru umožňuje sesbírat výsledky z více klasifikátorů a udělat z nich jeden co možná nejlepší. To se může hodit především v případech, kdy jeden z algoritmů selže a neposkytne žádný relevantní výstup.

4.3 Vývoj aplikací pro operační Systém Android

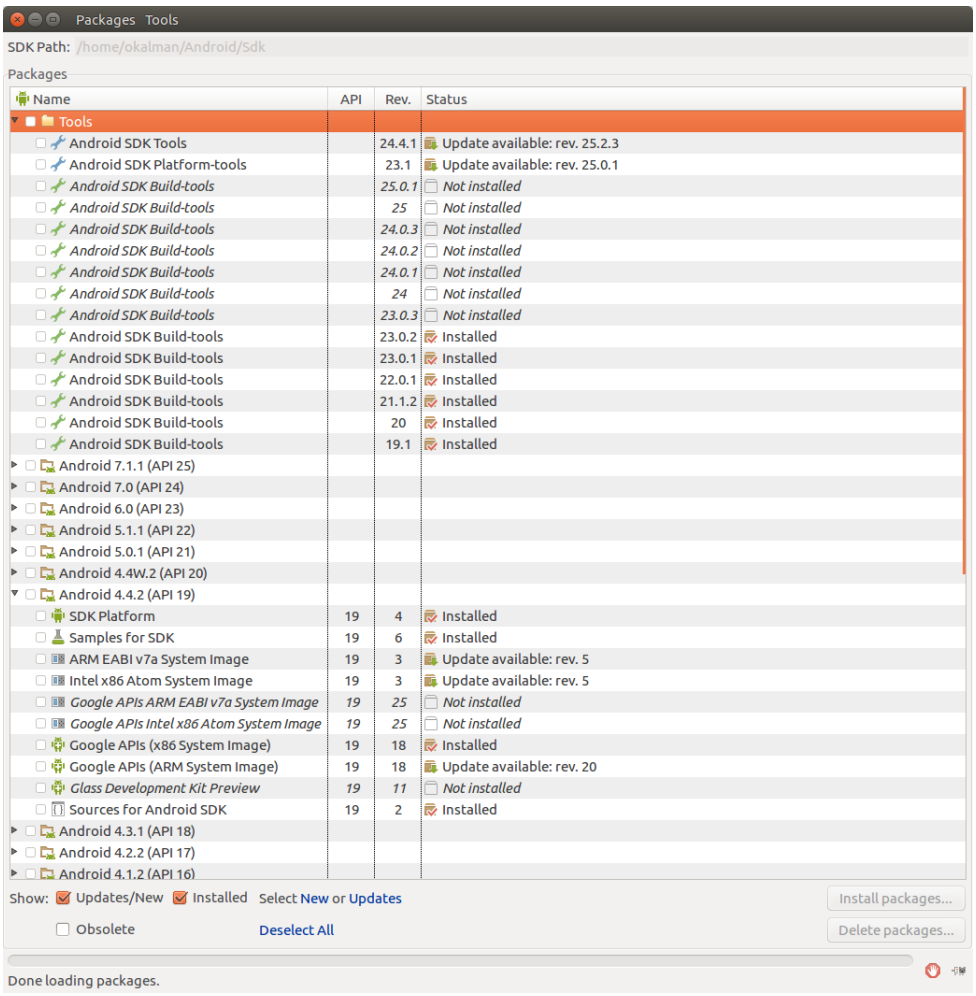
Ukázková aplikace byla vytvořena pro operační systém Android. Aplikace je naprogramovaná pomocí Android SDK (Software Development Kit), což je sada nástrojů v kombinaci s knihovnami a vývojovým prostředím, které vytvořila přímo společnost Google. Programovacím jazykem pro Android SDK je Java ve verzi 6. V následujících odstavcích se pokusím zjednodušeně nastínit, jaké důležité komponenty Android SDK nabízí a jaké běžně používané konstrukce a postupy jsem při programování aplikace použil.

Android SDK

Současné vývojové prostředí pro operační systém Android (dále jen Android nebo os. Android) se nazývá Android Studio, je založeno na open-source vývojovém prostředí IntelliJ Idea a integruje i dříve samostatné komponenty z Android SDK. V rámci této podkapitoly budou popsány 3 základní a dříve i do značné míry oddělené komponenty Android SDK, které jsou nyní v rámci Android Studia integrovány. Těmito komponentami jsou:

- SDK Manager (stále existuje i jako samostatná aplikace)
- Android Virtual Device Manager
- Integrované vývojové prostředí

SDK Manager je komponenta, která se jako první přihlásí o slovo po nainstalování a spuštění Android Studia. Je to správce knihoven a nástrojů, které umožňují využívat služeb Androidu, nespravují se zde zdroje externí, tedy knihovny, kterými si chce programátor nějakým způsobem zjednodušit práci, či využít existující volně dostupné řešení některých problémů. Jak uvádí (Ujbányai, 2012), při vytváření aplikace se musí programátor rozhodnout, jakou nejnižší verzi operačního systému bude používat, ta se volí pomocí takzvaného "API Level"(verze API), což je číslo v současné době od 7 do 25 přičemž každému odpovídá jedna z verzí operačního systému Android od verze 2.1 do 7.1.1. Starší verze než 2.1 již nejsou v rámci Android SDK podporovány. Programátor si tedy v SDKMmanageru může spravovat knihovny pro vývoj na libovolné Verzi API, pokud chce, může mít stažené knihovny i pro více verzí API. V rámci SDK Manageru se také dají spravovat speciální knihovny a nástroje, například knihovna pro zpětnou kompatibilitu nebo knihovny, které aplikacím budou umožňovat využívat některé typicky předinstalované (avšak nepovinné) součásti operačního systému. Jedná se především o integrované služby Googlu.



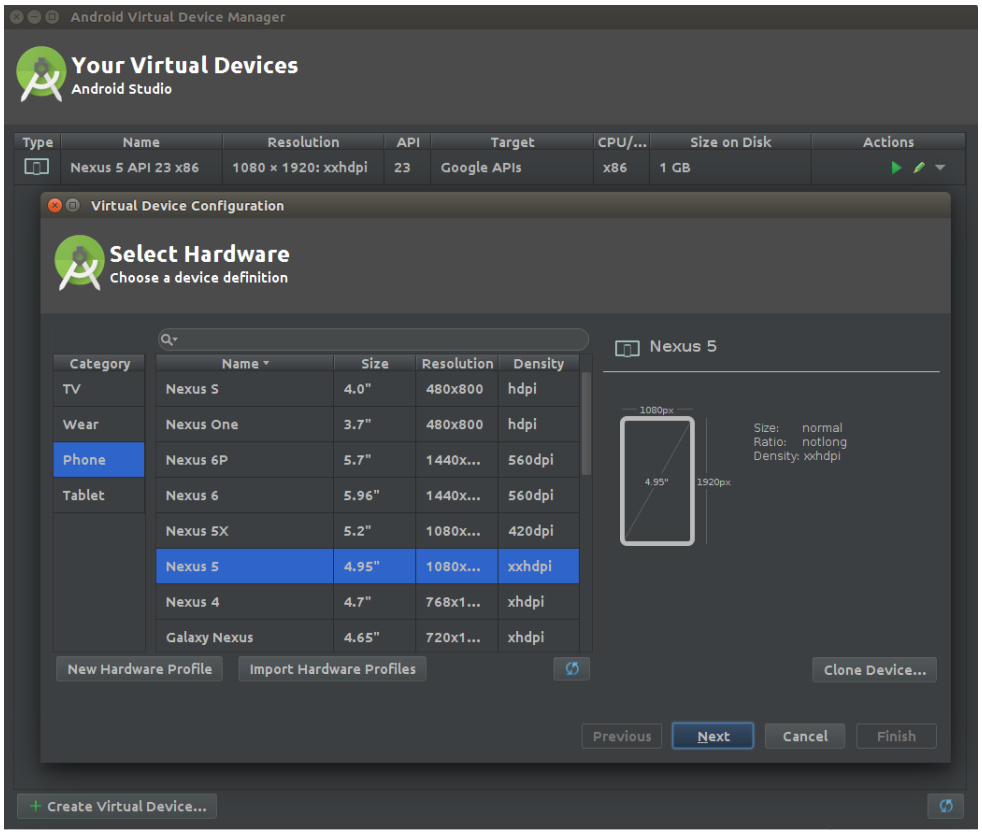
Obrázek 5: Samostatná aplikace SDK Manager

Jak je patrné z obrázku 6, pro vývoj jednoduché aplikace není nutné instalovat všechno a už vůbec není nutné instalovat knihovny pro podporu všech API. Zde je nainstalována podpora pro API verze 19. To znamená, že aplikace poběží na operačním systému ve verzi 4.4.2 a vyšší, což stačí pro drtivou většinu současných telefonů. Podle statistik společnosti Alphabet.Inc[2] mělo k 5. 12. 2016 84.7 % zařízení se systémem Android verzi 4.4.2 nebo novější (Android Developers, 2016).

Další důležitou komponentou SDK a to hlavně z hlediska testování je Android Virtual Device Manager. Tento nástroj nám umožňuje spravovat virtuální zařízení. Jedná se o virtuální mobilní telefony, tablety, hodinky a další produkty, na které se operační systém android běžně instaluje. Pomocí tohoto nástroje si můžeme například vytvořit virtuální mobilní telefon, zvolit si verzi operačního systému, velikost

[2]V srpnu 2015 došlo k reorganizaci struktur společnosti Google.Inc. Všechny divize Googlu nesouvisející s Androidem a webovými projekty jako je internetový vyhledávač, youtube a pod. byly od společnosti Google odděleny jako samostatné firmy. Všechny takto nově vzniklé firmy včetně původního Googlu se poté staly součástí nově vzniklého holdingu Alphabet.Inc

a rozlišení displeje, zda má telefon hardwarová navigační tlačítka, nebo zda se mají používat virtuální na displeji. Tento nástroj je vhodný k tomu, aby vývojáři zkontrolovali, zda aplikace na různě velkých telefonech a různých rozlišeních vypadá tak jak má, nicméně některé funkce (jako třeba Wi-Fi) jsou na emulátoru nedostupné. Stejně tak se na emulátoru nedá ladit výkon, protože virtuální zařízení jsou dost pomalá.

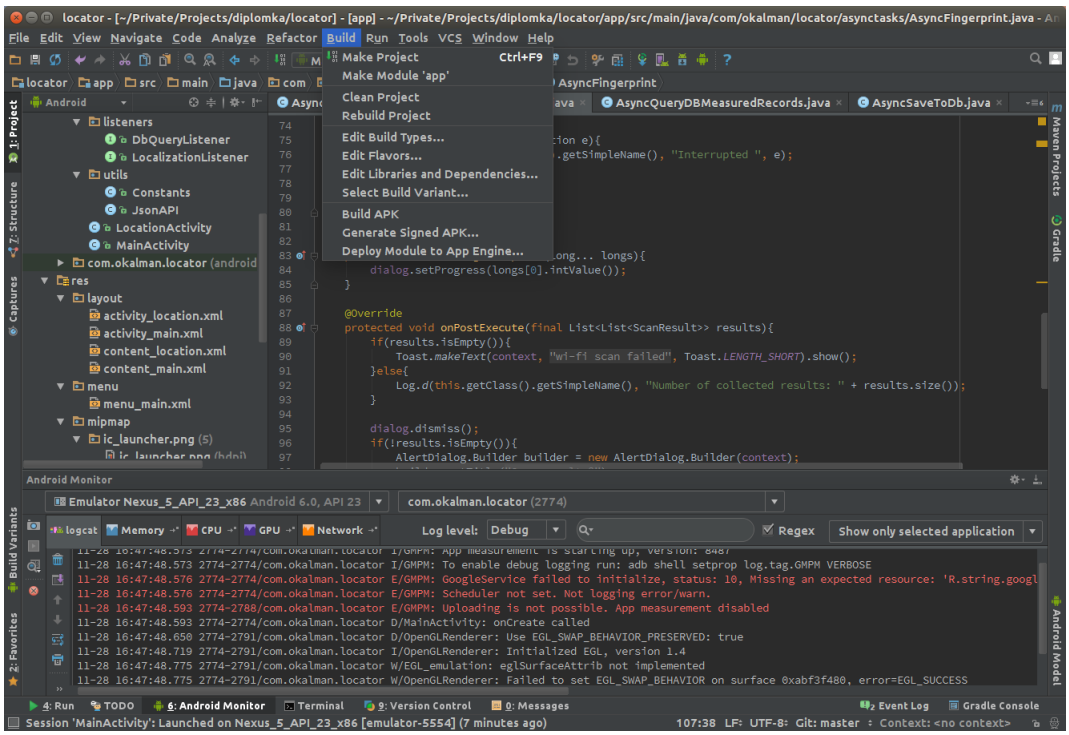


Obrázek 6: Integrovaný AVD Manager ukazuje jedno dostupné virtuální zařízení a průvodce k vytvoření dalšího

V AVD Manageru můžeme mít vytvořeno prakticky neomezené množství virtuálních zařízení, limitováni jsme pouze velikostí pevného disku (každý virtuální stroj si nějaký prostor alokuje). Zároveň můžeme mít spuštěno několik z těchto virtuálních strojů, opět jsme zde limitování pouze hardwarem našeho počítače. Při testování aplikace nám dá vždy Android studio na výběr z virtuálních strojů, na kterých aplikace může běžet (splňují minimální verzi API), a to jak z těch, co už jsou spuštěné, tak i z těch, co máme vytvořené, ale ještě neběží.

Poslední a největší komponentou Android SDK je samotné Integrované vývojové prostředí Android Studio. To dohromady integruje obě dříve popisované komponenty, přidává pokročilý editor zdrojového kódu, designer uživatelského rozhraní

a další funkce. Editor kódu disponuje inteligentním vyhledáváním importů, inteligentním doplňováním kódu a podporuje široké možnosti refactoringu kódu.



Obrázek 7: Integrovaný vývojové prostředí android studio

Další užitečnou funkcí Android Studia je integrovaný debugger, který se dokáže připojit na připojená zařízení, ať už se jedná o fyzický telefon nebo jeho virtuální ekvivalent vytvořený přes AVD Manager. Na obrázku 7 jsou ukázány i funkce pro kompilování a sestavování aplikací, je možné si zde vygenerovat podepisovací RSA certifikáty, pokud bychom měli v úmyslu aplikaci prodávat v oficiálním obchodu s aplikacemi Google Play. Ve spodní části je pak ukázaný připojený logger. Do něj můžeme pomocí Android.util.Log třídy posílat textová data z telefonu. Vedle logování je k dispozici také monitor obsazení operační paměti, vytížení procesoru a grafického čipu a monitor síťového provozu na telefonu.

Manifest

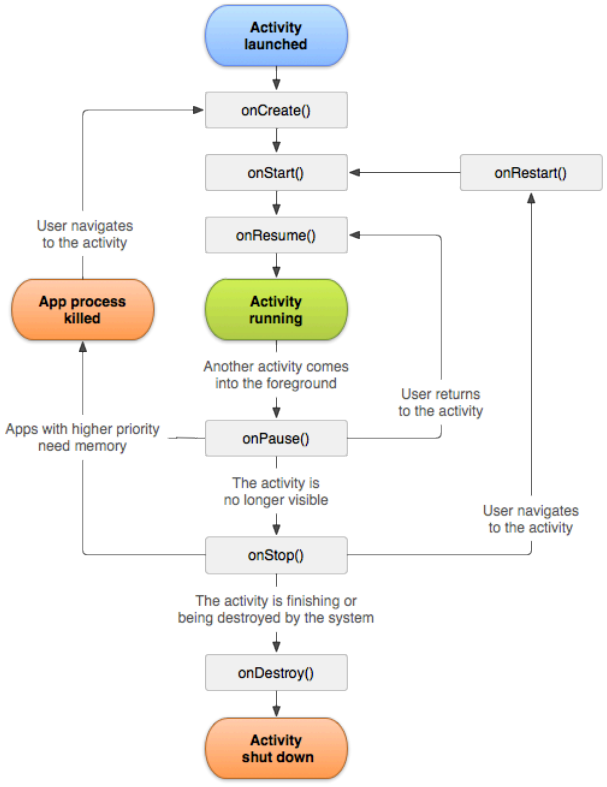
Vývojové prostředí, které bylo popsáno výše, nám usnadňuje tvorbu aplikací. V následujících odstavcích si tedy krátce popíšeme některé z konstrukcí a důležitých komponent aplikace.

Manifest je textový soubor psaný ve formátu XML, který je obsažen v každé aplikaci. Jak uvádí (Hashimi a Komatineni, 2009), v tomto souboru jsou obsaženy základní informace o aplikacia jejím obsahu, které mohou být analyzovány ještě před

tím, než ji uživatel nainstaluje. Pro samotné uživatele je asi nejdůležitější sekce ”povolení” (anglicky permissions). V této sekci programátor definuje, k jakým funkcím telefonu bude chtít aplikace přistupovat. Operační systém nebo Google Play Store (oficiální kanál, přes který se aplikace nakupují a stahují) poté na základě těchto informací uživateli oznámí, co aplikace bude smět dělat. Uživatel toto oznámení musí odsouhlasit, než bude aplikace nainstalována. Tento mechanizmus je jedním ze základních bezpečnostních prvků, na kterém stojí a padá zabezpečení dat v telefonu/tabletu.

Uživatelské rozhraní

Základním kamenem uživatelského rozhraní je Aktivita (třída Activity) a její odvozeniny. Jak uvádí (Grant, 2012), na Aktivitě se vykreslují základní prvky uživatelského rozhraní, mimo jiné má také implementovaný takzvaný životní cyklus, což je sada metod, které jsou automaticky volány v průběhu života Aktivity.



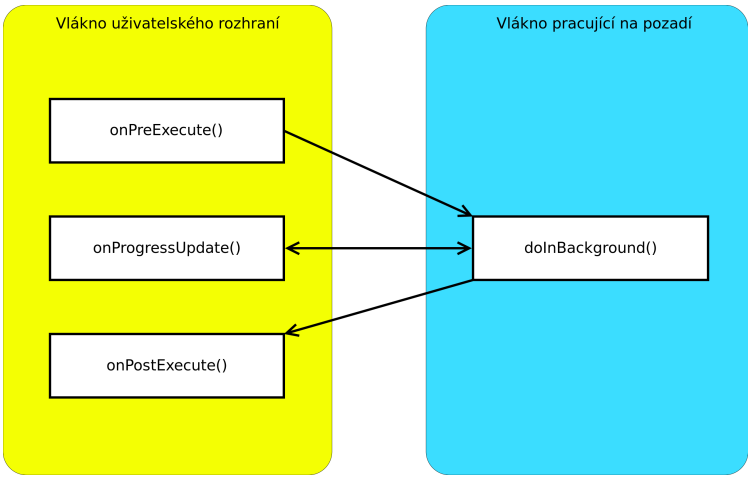
Obrázek 8: životní cyklus Aktivity (zdroj: https://developer.android.com, licence:CC2)

Na obrázku 8 je zobrazen životní cyklus aktivity, respektive metody, které může vývojář implementovat, a které boudou v průběhu života Aktivity volány. V implementaci ukázkové aplikace v rámci této práce je v Aktivitě dostupné menu a největší část plochy poté zabírá mapa budovy. Ta je zobrazena pomocí knihovny Phowto-

View, kterou naimplementoval Chris Banes[3] a je šířena pod Apache 2.0 licencí. Tato knihovna umožňuje zobrazovat obrázky, využívat na nich gesta a zaznamenávat souřadnice dotyků prstů.

Asynchronní operace

Blokující volání je operace, kdy program čeká na dokončení výpočtu nějaké funkce. Takové operace jsou při programování zcela běžné, i obyčejný součet dvou čísel je blokující operací, problém nastává, pokud jsou vykonávány ve vlákně uživatelského rozhraní a trvají příliš dlouhou dobu, nebo je jich příliš mnoho. V takovém případě se uživatelské rozhraní dostane do stavu, kdy nereaguje na podněty uživatele. Operační systémy obvykle tento problém detekují a nabídnou uživateli násilné ukončení této aplikace nebo programu. Jak uvádí (Murphy, 2008), Android se chová stejně a pokud se aplikace stane na dobu alespoň 5 sekund neresponzivní, doporučí ji uživateli ukončit. Části kódu, jejichž vykonání by mohlo trvat příliš dlouhou dobu, se proto doporučuje spouštět v samostatném vlákně a do vlákna s uživatelským rozhraním předávat data pomocí zpětného volání (tzv. callbacku).



Obrázek 9: Ukázka workflow při použití *AsyncTasku* pro asynchronní operace

Zpracování velkého množství dat, přístup k online zdrojům, sbírání dat a podobné operace přesně zapadají do kategorie těch, které by mohly způsobit zamrznutí uživatelského rozhraní. Slovo ”mohly” je zmíněno záměrně, protože ne vždy se například dotaz na databázi musí protáhnout do té doby, aby si toho uživatel všiml. Potíž je v tom, že programátor nikdy nemůže vědět, jak rychlá tato odezva bude. Obecně platí, že veškeré operace pracující se síťovými nebo diskovými zdroji (platí i pro SD karty), by měly být vykonávány na pozadí. Pro tyto účely je v Android SDK připravena speciální třída *AsyncTask*. Ta má za úkol programátorům zjednodušit používání více vláken. Obsahuje metody, které jsou vykonávány na uživatelském vlákně

[3]https://github.com/chrisbanes/PhotoView

i metodu vykonávanou v samostatném vlákně, navíc se mezi nimi dají jednoduše předávat data. Sada poskytnutých metod umožňuje připravit uživatelské rozhraní na výkon asynchronní operace *AsyncTasku*, průběžné aktualizování uživatelského rozhraní, například informování o průběhu asynchronní operace *AsyncTasku* a nakonec aktualizovat uživatelské rozhraní po ukončení asynchronní operace *AsyncTasku*, a to ať už byla práce na pozadí přerušena, nebo doběhla korektně do konce.

Skenování okolních Wi-Fi sítí

Android poskytuje API, přes které můžeme zaznamenávat informace o okolních bezdrátových sítích. Abychom se ale mohli k těmto datům dostat, musíme v Manifestu deklarovat, že naše aplikace bude k Wi-Fi sítím přistupovat a co víc, aplikace také musí požádat o povolení lokalizovat telefon s vysokou přesností, protože programátoři Googlu jsou si vědomi, že tato data mohou být použita k určení polohy telefonu s přesností na několik metrů.

Třída, která umožňuje tyto informace získávat, se nazývá *WifiManager*, nabízí však i další funkcionalitu, jako třeba zjištění stavu Wi-Fi karty (jestli máme na telefonu/tabletu Wi-Fi zapnutou), zda telefon podporuje 5GHz pásmo a podobně. Metoda *getScanResults()* poskytuje seznam okolních Wi-Fi vysílačů, nicméně může se stát, že pokud ji zavoláme během krátké chvíle vícekrát, vrátí nám identické výsledky. Z tohoto důvodu je dobré si kontrolovat, zda byla od poslední žádosti data aktualizována, to může být realizováno pomocí porovnání časových razítek, která jsou u každého záznamu uvedena.

5 Vývoj a implementace lokalizační služby

Při vývoji lokalizační služby bylo vycházeno z návrhu popsaného v kapitole 4. Při realizaci bylo naprogramováno několik aplikací

1. Ukázková aplikace - pro demonstraci funckionality
2. Webový servlet - prostředník mezi IBM Watson a ukázkovou aplikací
3. Pomocná transformační aplikace - pro zpracování sesbíraných dat určených pro trénování a testování

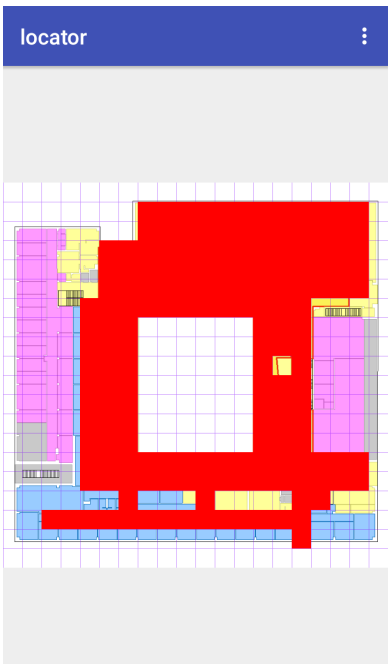
V této kapitole budou podrobně představeny jednotlivé implementační části včetně jejich ukázek. Budou zde objasněny implementační postupy a odůvodněny některé zajímavé nebo problém řešící metody, které byly použity.

5.1 Sběr dat

Pro trénování modelu a případné praktické ukázky lokalizace byla v rámci práce vytvořena mobilní aplikace pro os. Android. Jako mapový podklad slouží této aplikaci web http://indica.mendelu.cz/inidica , ze kterého je mapa při spuštění aplikace stažena. V rámci této práce jsem pracoval s přízemním podlažím budovy Q Provozně Ekonomické Fakulty Mendelovy univerzity v Brně. Aplikace umí pracovat i s dalšími poschodími této budovy. Podpora ostatních budov nebyla implementována, nicméně z technického hlediska je možné uplatnit metody lokalizace popisované v této práci i kdekoli jinde. Jak už bylo zmíněno, aplikace má dvě základní funkcionality, každá z nich bude popsána v následujících podsekcích samostatně. Při sbírání dat byly brány v potaze AP šířící síť eduroam, a to z důvodu možného zanesení šumu do dat mobilními hotspoty, které si návštěvníci budovy mohou dle libosti vytvářet svými mobilními telefony.

Sbírání dat pomocí mobilní aplikace

Základním předpokladem pro sbírání dat a vytváření tzv. otisků je rozdělení mapy na políčka, která budou dále nedělitelná a budou představovat pozici na mapě. Mapový podklad byl pokryt mřížkou 20x20 čtverců, velikost každého čtverce je přibližně 3x3 metry. Každému čtverci bylo z levého horního rohu mapy přiděleno unikátní číslo (ID) z intervalu 0-399. V těchto čtvercích jsou potom oskenovány okolní přístupové body, ty jsou potom jako jeden záznam uloženy, včetně čísla políčka do databáze. Algoritmy strojového učení vyžadují značné množství trénovacích dat, proto je skenování prováděno vícekrát. Aplikace sbírá data pro každé políčko přesně 25s, za tuto dobu se obvykle povede získat 25-35 unikátních záznamů. Tato část je do jisté míry ovlivněna použitým hardwarem (telefonem), respektive ovladačem bezdrátového modulu, který výrobce telefonu dodal.



Obrázek 10: Ukázka aplikace při trénování, červeně vybarvená políčka jsou již oskenována

Aplikace umí nasbíraná data vyexportovat z databáze do CSV souboru, ten je poté uložen do interní paměti telefonu a může být dále požit buďto přímo pro trénování algoritmů, nebo naimportován do jiné databáze. Akce jako je export dat, nebo samotné skenování mohou zabrat poměrně dost času, proto je uživatel informován o průběhu těchto akcí. Export databáze použité při trénování algoritmů v této práci zabere i několik minut.

Záznam	
•x	float
◦y	float
◦floor	int
◦tile	int
◦MAC_0	String
◦level_0	int
◦MAC_1	String
◦level_1	int
◦MAC_2	String
◦level_2	int
◦MAC_3	String
◦level_3	int
◦MAC_4	String
◦level_4	int
◦MAC_5	String
◦level_5	int
◦MAC_6	String
◦level_6	int
◦MAC_7	String
◦level_7	int

Obrázek 11: Takto vypadá jeden záznam při sbírání Wi-Fi sítí

Transformace dat

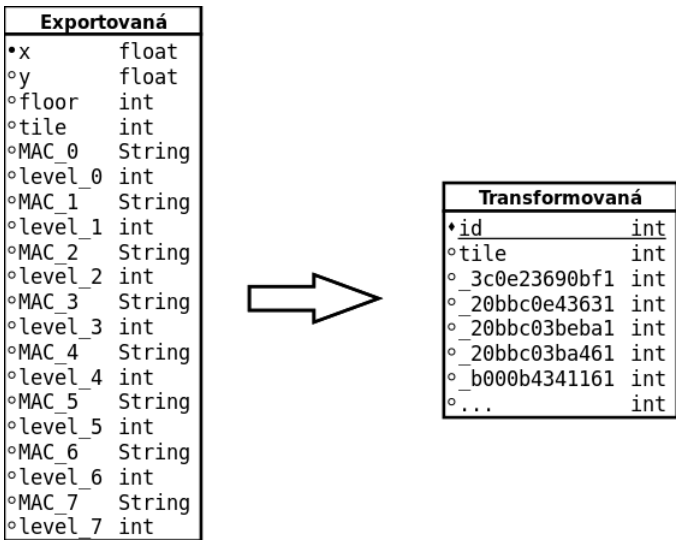
Formát dat, tak jak jsou sesbírána mobilním telefonem, není vhodný pro přímé použití žádným z algoritmům strojového učení. Pro tento účel byl vytvořen malý javovský program, který přečte data z exportovaného souboru a přetransformuje je do správného formátu. Postup transformace je následný:

1. Projdeme exportovaný soubor a vytvoříme množinu všech MAC adres, na které jsme narazili, na obrázku 13 jsou to hodnoty políček MAC_x.
2. Vytvoříme v databázi tabulku, jejíž sloupce budou odpovídat všem prvkům z množiny MAC adres + jeden sloupec pro číslo políčka a jeden pro ID. Defaultní hodnota pro sloupce s názvy MAC adres bude -200.
3. Projdeme soubor znovu a pro každý záznam v souboru vytvoříme řádek v databázi. Vyplníme pouze sloupce, které v našem záznamu mají odpovídající MAC_x a to příslušnou hodnotou level_x.

Při vytváření tabulky můžeme narazit na problém, že jména MAC adres nemohou být použita jako jméno sloupce a to z několika důvodů:

1. dvojtečka není povolený znak v názvu sloupce: ‚usíme tedy odstranit z MAC adres všechny dvojtečky.
2. název sloupce nemůže začínat více než dvěma čísly: na tento problém narazíme po odstranění dvojteček, přidáme tedy nějaký validní znak na začátek. V našem případě to bylo podtržítko.

Po provedených úpravách by neměl být problém vytvořit tabulku v databázi s příslušnými daty. Takto upravená data v databázi mohou být následně použita pro trénování algoritmů.



Obrázek 12: Ukázka transformace v praxi

Z výše uvedeného postupu je patrné, že velikost databáze se při sbírání dat rozšiřuje po obou osách, tady jak počtem řádků, tak i počtem sloupců a časem může narůst do značných rozměrů. To je jeden z argumentů, proč se tato část práce neprovádí přímo na zařízení, které data nasbíralo. Dalším argumentem může být to, že pro sběr dat můžeme paralelně využít několik různých zařízení a jejich vyexportovaná data poté sloučit do jednoho souboru a až poté transformaci provést. Pro účely transformace se používá lokální databáze a to především z výkonnostních důvodů. Na obrázku 12 si ještě můžeme všimnout, že jsme při transformaci vynechali některé sloupce, které v původních datech byly. Jedná se o sloupce x, y a floor. Tyto sloupce totiž nejsou pro trénování algoritmů relevantní. Za zmínku určitě stojí i jedna obrovská nevýhoda celého tohoto řešení, tou je nemožnost trénovat algoritmy inkrementálně, jelikož se vstupní vektor (hlavička tabulky) při postupném sbírání dat stále mění.

id	tile	_3c0e23690b ▲	_b000b420c451	_20bbc0e43631	_20bbc03beba1	_20bbc03ba461	_b000b4341161	_1c1d86306391	_5ca48a686e41	_5ca
36	124	-200	-200	-200	-86	-73	-200	-200	-78	-200
37	124	-200	-200	-200	-82	-73	-200	-200	-77	-200
46	124	-200	-200	-200	-85	-68	-200	-200	-74	-200
47	124	-200	-200	-200	-85	-67	-200	-200	-74	-200
48	124	-200	-200	-200	-85	-66	-200	-200	-75	-200
49	124	-200	-200	-200	-85	-67	-200	-200	-75	-200
50	124	-200	-200	-200	-85	-68	-200	-200	-74	-200
51	124	-200	-200	-200	-200	-68	-200	-200	-77	-200
52	124	-200	-200	-200	-200	-68	-200	-200	-75	-200
53	124	-200	-200	-200	-200	-68	-200	-200	-75	-200

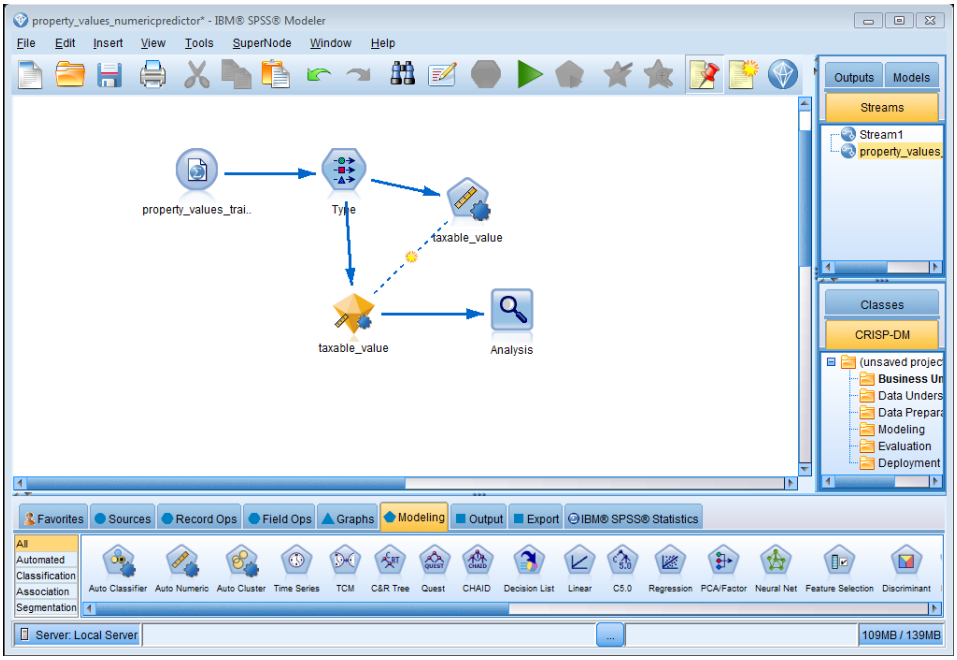
Obrázek 13: Ukázka části SQL tabulky po provedené transformaci

Transformační program dělá ještě jednu důležitou věc, a tou je vytvoření tabulky v databázi používané servletem, tedy databázi hostované na vzdáleném serveru. Tabulka vypadá úplně stejně jako ta, co je vytvořena v lokální databázi, jen se do ní neukládají žádná data. Podstatná je pouze hlavička.

5.2 Trénování algoritmů

Dalším krokem po sesbírání trénovacích dat je jejich použití. V programu IBM SPSS Modeler tedy vytvoříme prediktivní model, který bude obsahovat kromě samotných klasifikačních algoritmů ještě řadu dalších komponent, které nám umožní pracovat se vstupními daty a analyzovat výstupy. Všechny podstatné komponenty budou v této podkapitole vysvětleny. Pro začátek je nutné sjednotit terminologii, která bude pro popis v této podkapitole používána.

- Prediktivní model: projekt vytvořený v SPSS modeleru pro účely lokalizace
- Uzel: jeden z modrých objektů na pracovní ploše modeleru viz obrázek 14
- Záznam: jeden řádek v tabulce vstupních dat
- Atribut: jeden sloupec v tabulce vstupních dat



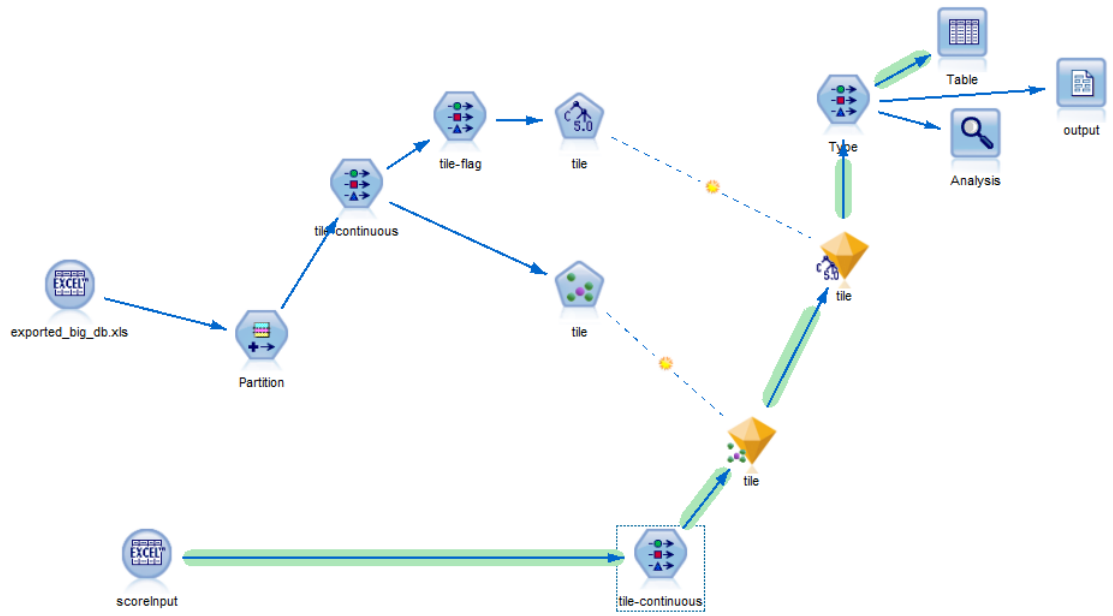
Obrázek 14: Ukázka prostředí IBM SPSS Modeler 17

Při tvorbě modelu využíváme uzly obsažené ve spodní liště Modeleru, do modelu je přidáváme obyčejným přetažením myši a mají světle modrou barvu, uzly jsou setříděny do několika záložek podle jejich použití. Na obrázku 14 je ukázán

obsah záložky s analytickými uzly pro vyhodnocování výsledků. Po přetažení uzlu na pracovní plochu jej většinou můžeme dále konfigurovat. Nabídka nastavení se samozřejmě u každého uzlu liší. Následně můžeme uzly propojit do větších celků. Propojením se vytvoří datový spoj, kterým budou předávána data a to ve směru šipky. Typický model má následující strukturu:

- Uzel definující vstupní data, například soubor, nebo připojení do databáze
- Uzly modifikující data
- Uzly definující sémantiku dat
- Uzly představující algoritmy (klasifikační, regresní a další)
- Uzly pro zpracování a anylýzu výsledků

Po natrénování algoritmu se na pracovní plochu automaticky přidá nový objekt, narozdíl od ostatních uzlů je vybarven žlutě. Jedná se o natrénovanou instanci algoritmu, která je schopná zpracovávat vstupy a poskytovat relevantní výstup. Na tento objekt poté můžeme napojit uzly pro analýzu výsledků, nebo jinou instanci natrénovaného algoritmu a udělat tak složitější vyhodnocovací řetězec.



Obrázek 15: Výsledný prediktivní model používaný při lokalizaci

Prediktivní model pro aproximaci polohy je zobrazen na obrázku 15. Skládá se ze dvou vstupních uzlů, šesti transformačních uzlů, dvou klasifikačních algoritmů a třech uzlů pro práci s výsledky.

Vstupní uzly

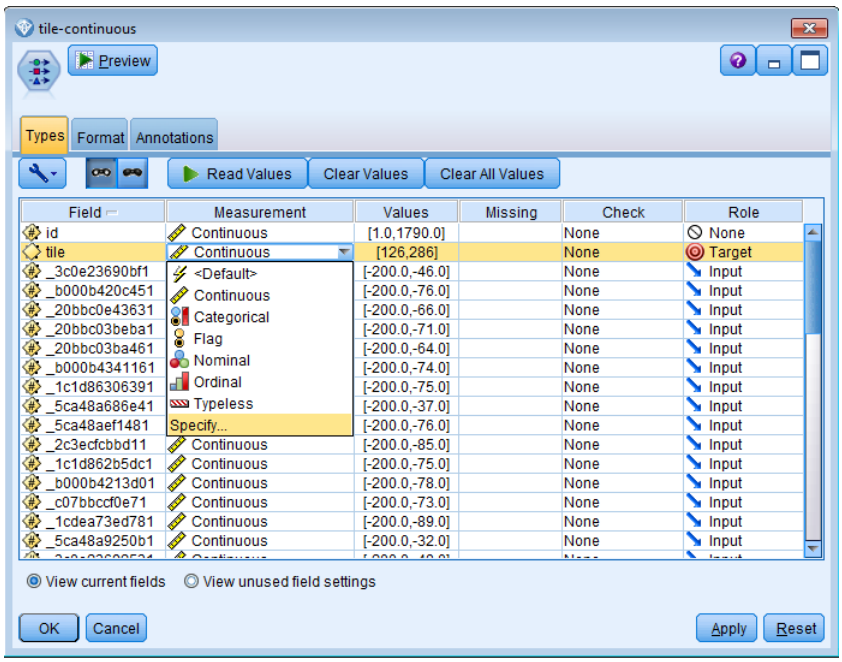
Vstupní uzly mají na obrázku 15 kruhový tvar s nápisem „EXCEL“. Jsou to uzly, kterým můžeme nastavit vstupní soubor ve formátu xls nebo xlsx, vytvořený v programech MS Excel, LibreOffice Calc a dalších. Jako jediný se ukázal dostatečně spolehlivý při zpracování většího množství dat. Typické nastavení v tomto uzlu spočívá v zaškrtnutí políčka dle toho, jestli se na prvním řádku nachází záhlaví tabulky. Dále je zde možné základní filtrování atributů tím, že u některých můžeme nastavit jejich vynechání. Pokud u některého z atributů toto nastavíme, po připojení vstupního uzlu na další uzel nebude tento atribut dat předán z výstupu na vstup. Zajímavějším z hlediska nastavování by byl uzel, který umožňuje zpracování CSV souborů, zde je totiž typicky možno nastavit nejen existenci záhlaví, ale také oddělovače a uvozování textových řetězců. Naneštěstí tento uzel nedokázal pracovat s tak velkým množstvím dat, která byla nasbírána.

Transformační uzly

Transformačních uzlů je v prediktivním modelu 6 a mají tvar osmiúhelníku, z toho 5 je stejného typu s názvem „Type“, jsou to uzly se třemi šipkami. Tento typ uzlu má primárně tři funkcionality

1. Nastavovat a měnit datové typy atributů.
2. Označovat atributy, které mají klasifikační algoritmy brát jako svůj vstup.
3. Označit atribut, který bude klasifikačními algoritmy brán jako výstup (budou se podle něj učit).

Použití transformačního uzlu „Type“ se hodí tam, kde potřebujeme změnit sémantiku dat. Například algoritmus C5.0 vyžaduje, aby byla výstupní proměnná nominální hodnota, tedy aby byla chápána pouze jako příznak označující nějakou množinu dat. Naproti tomu algoritmus K-NN vyžaduje, aby byla výstupní hodnota chápána jako reálné číslo. Pro každý z těchto algoritmů tedy můžeme nastavit vlastní sémantiku dat tak, aby se algoritmy byly schopny natrénovat.



Obrázek 16: Nastavení uzlu „Type“

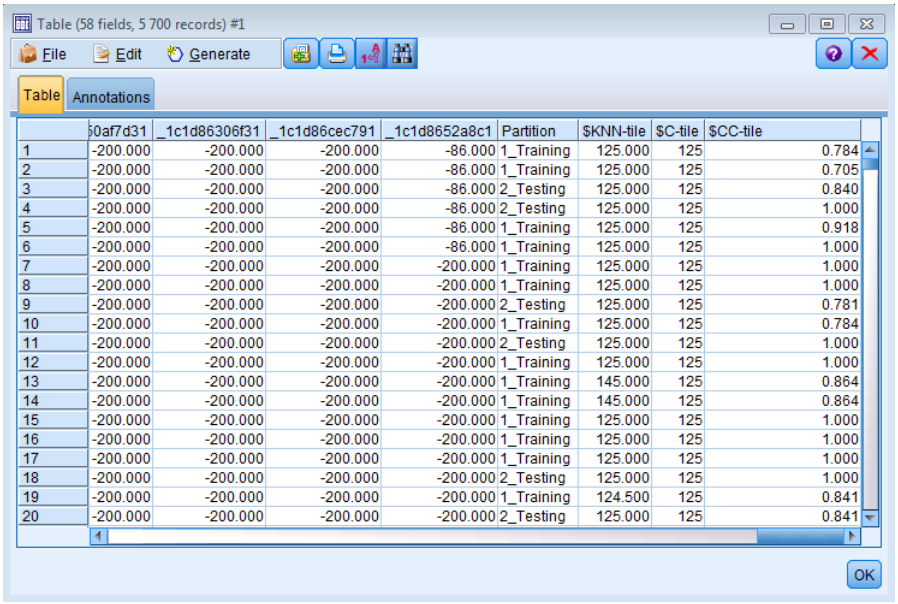
Datových typů, které můžeme pro atribut zvolit, je celá řada, na obrázku 16 si můžeme všimnout, že se nabízí i volba vlastnoručně definovaného datového typu. To se hodí především při analyzování dat. Pro klasifikační algoritmy je vhodné použít ty předdefinované. Také zde jsou dva další sloupce „Missing“ a „Check“, První z nich slouží ke zpracování záznamů, u kterých daná hodnota chybí. Takovéto záznamy můžeme buď vyřadit, nebo jim nějakou hodnotu přiřadit (například průměrnou hodnotu a podobně). Druhý, sloupec „Check“, slouží k ověření, že hodnoty daného atributu odpovídají u všech záznamů nastavenému datovému typu, tato funkce je užitečná při trénování algoritmů, protože na chybu v datech přijdeme mnohem rychleji a ne až po několika minutách trénování.

Dalším transformačním uzlem je uzel „Partition“, na obrázku má identický popisek. Tento uzel je užitečný především pro testování algoritmů, dají se zde rozdělit data na dvě disjunktní množiny v libovolném poměru a poté jednu z nich použít pro trénování algoritmů a druhou pro testování. V rámci tohoto uzlu se dá nastavit i pevný seed, tedy hodnota, ze které bude vycházet kongruentní generátor náhodných čísel při vyváření obou množin. Tím si můžeme zajistit to, že se například testovací sada nemění a to i když si zmenšíme velikost trénovací sady (při zachování velikosti testovací sady). Můžeme tedy například vzít 30 % všech dat a používat je jako testovací, zatímco zbylých 70 % trénovacích dat můžeme ořezávat a zjišťovat tak jak se mění úspěšnost algoritmů v závislosti na velikosti testovací množiny. Alternativně můžou být data rozdělena na tři díly, třetí část pak slouží pro validaci.

Uzly pro zpracování výsledků a jejich analýzu

V prediktivním modelu lokalizátoru zobrazeného na obrázku 15 máme tři uzly, které nějakým způsobem pracují s výstupem z klasifikačních algoritmů.

1. Uzel „Table“ slouží pro jednoduché zobrazení vstupních a zároveň i výstupních dat
2. Uzel „Flat file“ (na obrázku pojmenovaný jako „output“) slouží pro exportování vstupních a výstupních dat do CSV souboru
3. Uzel „Analysis“ se používá k analyzování výsledků, a to především z hlediska úspěšnosti klasifikátorů.



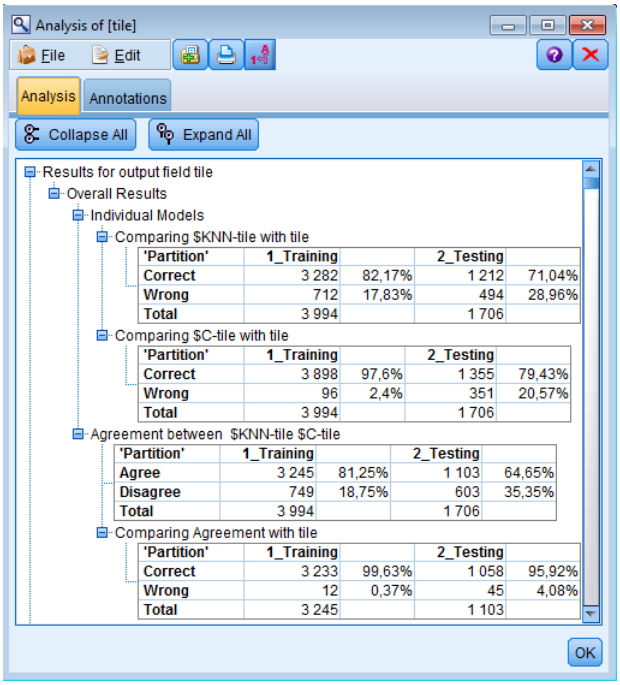
	0af7d31	_1c1d86306f31	_1c1d86cec791	_1c1d8652a8c1	Partition	$KNN-tile	$C-tile	$CC-tile
1	-200.000	-200.000	-200.000	-86.000	1_Training	125.000	125	0.784
2	-200.000	-200.000	-200.000	-86.000	1_Training	125.000	125	0.705
3	-200.000	-200.000	-200.000	-86.000	2_Testing	125.000	125	0.840
4	-200.000	-200.000	-200.000	-86.000	2_Testing	125.000	125	1.000
5	-200.000	-200.000	-200.000	-86.000	1_Training	125.000	125	0.918
6	-200.000	-200.000	-200.000	-86.000	1_Training	125.000	125	1.000
7	-200.000	-200.000	-200.000	-200.000	1_Training	125.000	125	1.000
8	-200.000	-200.000	-200.000	-200.000	1_Training	125.000	125	1.000
9	-200.000	-200.000	-200.000	-200.000	2_Testing	125.000	125	0.781
10	-200.000	-200.000	-200.000	-200.000	1_Training	125.000	125	0.784
11	-200.000	-200.000	-200.000	-200.000	2_Testing	125.000	125	1.000
12	-200.000	-200.000	-200.000	-200.000	1_Training	125.000	125	1.000
13	-200.000	-200.000	-200.000	-200.000	1_Training	145.000	125	0.864
14	-200.000	-200.000	-200.000	-200.000	1_Training	145.000	125	0.864
15	-200.000	-200.000	-200.000	-200.000	1_Training	125.000	125	1.000
16	-200.000	-200.000	-200.000	-200.000	1_Training	125.000	125	1.000
17	-200.000	-200.000	-200.000	-200.000	1_Training	125.000	125	1.000
18	-200.000	-200.000	-200.000	-200.000	2_Testing	125.000	125	1.000
19	-200.000	-200.000	-200.000	-200.000	1_Training	124.500	125	0.841
20	-200.000	-200.000	-200.000	-200.000	2_Testing	125.000	125	0.841

Obrázek 17: Takto jsou reprezentovány výsledky klasifikací v tabulce

Transformační uzly a klasifikační algoritmy mohou do záznamů vygenerovat nový atribut. Výsledek je možné si prohlédnout pomocí uzlu „Table“. V případě popisovaného prediktivního modelu byly při testování a trénování vygenerovány 4 nové atributy viz obrázek 17, jedná se o poslední 4 atributy napravo. Z pohledu aproximace polohy jsou nejzajímavější atributy „$KNN-tile“ a „$C-tile“ zde se totiž jedná o výstupy samotných klasifikačních algoritmů. Záznamy, které mají atribut „Partition“ nastavený na hodnotu „2_Testing“, nebyly použity při trénování algoritmů a jsou to tedy nezávislé výsledky klasifikace.

Posledním uzlem ze jmenované trojice je uzel „Analysis“ ten automaticky porovnává výsledky predikcí z trénovacích i testovacích dat. Máme tedy okamžitý přehled o úspešnosti algoritmů. V tabulce je zde rozepsána jak procentuální úspěšnost, tak i celkový souhrn počtu záznamů jež byly klasifikovány úspěšně a neúspěšně. V ana-

lytickém uzlu jsou také podrobněji rozepsány i úspěšnosti predikcí pro jednotlivé klasifikační třídy.



Obrázek 18: Analýza výsledků

Uzel klasifikačního algoritmu K-NN

Uzly klasifikačních algoritmů představují jakousi šablonu, kde můžeme nastavovat parametry klasifikačních algoritmů tak, aby dosahovaly co nejlepších výsledků. Seznam nastavitelných parametrů se stejně jako u většiny jiných uzlů liší. Vzhledem k tomu, že klasifikační algoritmy představují jakési jádro celého lokalizačního řešení, budou oba popsány zvlášť. Jako první tedy začneme s algoritmem K-NN. Základní principy a charakteristiky tohoto algoritmu jsou popsány v kapitole 3.1. Zde bude především popsáno, jak byl algoritmus K-NN nastaven pro řešení lokalizačního problému.

Algoritmus K-NN má nastavenu automatickou volbu čísla k viz 3.1. Hodnota je zvolena na základě křížové validace (anglicky Cross-Validation). Při ní jsou trénovací data náhodně rozdělena na 10 podmnožin, algoritmus se poté trénuje vždy na devíti z nich a vynechaná podmnožina je použita pro otestování úspěšnosti. Tento postup je proveden desetkrát pro každé z možných k tak, aby byla každá z množin použita pro testování. V našem případě je velikost k omezena intervalem $< 3; 7 >$. Je tedy provedeno $5 \cdot 10$ validačních cyklů. Poté je zvoleno k s nejmenší chybovostí během křížové validace. Jak uvádí dokumentace (IBM, 2016), dalším parametrem, který může mít na úspěšnost klasifikačního algoritmu K-NN vliv, je způsob, jakým se počítá vzdálenost klasifikovaného záznamu od ostatních. V našem případě je zvolena

možnost „City Block metric“, častěji také známá jako Manhattanská vzdálenost. Posledním významným parametrem, je způsob jakým je vypočítána cílová hodnota (třída, do níž má záznam patřit). Jedním ze způsobů je, že se vypočítá průměr z cílových hodnot k nejbližších sousedů. Tento způsob není pro klasifikaci příliš vhodný, protože takto může vzniknout hodnota, která neodpovídá žádné ze známých tříd a to třeba i po zaokrouhlení. V našem případě je tedy zvolena možnost výběru dle mediánu. Vybere se tedy medián z cílových hodnot k nejbližších sousedů.

Uzel klasifikačního algoritmu C5.0

Druhým použitým algoritmem je Algoritmus C5.0, ten je z pohledu nastavení podstatně jednodušší. V základu jsou k dispozici pouze 2 parametry ovlivňující jeho výkonnost. Prvním z nich je použití boostingu a s tím související hodnota „Number of trials“, ta udává počet rozhodovacích stromů, jež se budou na klasifikaci podílet. V rámci této práce byly zvoleny 4. Větší počet rozhodovacích stromů často vede k lepším výsledkům a menší chybovosti, nicméně 32bitová trial verze Modeleru měla při větším počtu stromů problém s alokováním většího množství paměti než 1,5 GB. Byla tedy zvolena maximální možná hodnota. Druhým parametrem je předpokládaná míra šumu v datech udávaná v procentech „Expected noise“. Zde byla ponechána původní hodnota, protože se ukázala jako efektivní. Tento parametr má vliv na to, jak moc budou výsledné stromy prořezávány. Metoda křížové validace je v tomto případě určena pouze pro odhad přesnosti algoritmu, pokud nedisponujeme dostatečně velkou množinou dat, abychom ji mohli rozdělit na trénovací a testovací část.

5.3 Serverová část

Dotazy na určení polohy jsou zpracovávány na vzdáleném serveru. Aplikace mohou se servletem komunikovat pomocí JSON aplikačního rozhraní (dále jen API), jehož forma bude popsána dále v této podkapitole.

Pro implementaci byla použita technologie servletů. Servlet je malý program napsaný v jazyce Java spuštěný na webovém serveru. Jeho úkolem je zpracovávat HTTP/HTTPS požadavky a odpovídat na ně. Servlety samotné mohou být použity ke generování webových stránek, protože do HTTP odpovědi může být vložen klasický HTML kód. Tato praxe však není příliš běžná, protože programatické tvoření HTML kódu přímo v Javě není příliš pohodlné. Jak uvadí (Hunter a Crawford, 2001), častěji je technologie servletů používána v kombinaci s JSP (JavaServer Pages). Dalším případem užití je implementace webových služeb (web servicies), to jsou služby komunikující pomocí HTTP protokolu, jež primárně nemají za úkol poskytovat uživatelské rozhraní. Místo toho komunikují s klienty pomocí definovaného API a poskytují jim data. Takováto webová služba byla použita i v případě této práce.

JSON API

Pro komunikaci se serverem je použit objekt, který je knihovnou Jackson[4], poskytovanou pod Apache 2.0 licencí, automaticky serializován do JSON formátu. Nejjednodušším způsobem jak API popsat a představit je ukázka:


```
{"data":[
[   {"id":0,"level":-61,"mac":"6c:19:8f:c8:75:60",
    "record":null,"ssid":"eduroam"},
    {"id":0,"level":-63,"mac":"48:5b:39:cd:14:f2",
    "record":null,"ssid":"eduroam"},
    {"id":0,"level":-65,"mac":"18:a6:f7:8d:32:16",
    "record":null,"ssid":"eduroam"},
    {"id":0,"level":-76,"mac":"72:54:d2:89:56:9d",
    "record":null,"ssid":"eduroam"},
    {"id":0,"level":-77,"mac":"70:54:d2:89:56:9b",
    "record":null,"ssid":"eduroam"},
    {"id":0,"level":-71,"mac":"00:24:8c:90:09:57",
    "record":null,"ssid":"eduroam"}],
[   {"id":0,"level":-61,"mac":"6c:19:8f:c8:75:60",
    "record":null,"ssid":"eduroam"},
    {"id":0,"level":-59,"mac":"48:5b:39:cd:14:f2",
    "record":null,"ssid":"eduroam"},
    {"id":0,"level":-65,"mac":"18:a6:f7:8d:32:16",
    "record":null,"ssid":"eduroam"},
    {"id":0,"level":-72,"mac":"72:54:d2:89:56:9d",
    "record":null,"ssid":"eduroam"},
    {"id":0,"level":-73,"mac":"70:54:d2:89:56:9b",
    "record":null,"ssid":"eduroam"},
    {"id":0,"level":-71,"mac":"00:24:8c:90:09:57",
    "record":null,"ssid":"eduroam"}]
],
"key":"=$rQXue!hMjyJX54",
"response":""
}
```


V ukázce si můžeme všimnout, že API používá tři objekty

- data
- key
- response

[4] http://wiki.fasterxml.com/JacksonHome

Objekt „data“ v sobě nese seznam polí, kde každé pole představuje jeden dotaz na Wi-Fi kartu, která poskytne seznam okolních AP a sílu jejich signálu. Takových polí může být v objektu neomezené množství. Na serveru jsou tato data přefiltrována a transformována a nakonec odeslána prediktivnímu modelu jako jeden větší dotaz. Tímto řešením se šetří náklady, protože metrika pro výpočet ceny za hostované cloudové služby často zahrnuje počet obsloužených dotazů.

Objekt „key“ je tajný klíč, jenž je při zpracování dotazu ověřen servletem. Toto jednoduché opatření umožňuje omezit funkcionalitu jinak veřejné služby pouze pro schválené aplikace. Pokud by na servlet mohl posílat dotazy kdokoli, snadno by se mohlo stát, že ať už záměrně nebo nevědomky, způsobí nečekaný nárůst poplatků za služby IBM Bluemix. V našem případě by spíš mohl vyčerpat limit poskytovaný zdarma.

Objekt „response“ je používaný až při odpovědi ze serveru. Pro dotazování a odpovídání se tedy používá jeden stejný JSON objekt. Stejně jako není pro dotaz povinnost uvádět „response“, v odpovědi můžou analogicky chybět položky „data“ a „key“, protože jsou bezpředmětné.

Zpracování a vyhodnocování dotazů

JSON objekt uvnitř HTTP dotazu (ať už se jedná o GET nebo POST) je zpracován opět pomocí knihovny Jackson a transformován na objekt jazyka Java. Poté jsou nad ním provedené některé transformační operace popsané v kapitole 5.1. Prvním krokem je odstranění všech záznamů o AP, které vysílají jiné SSID nežli „eduroam“. Poté následuje odstranění dvojteček z MAC adres a přidání podtržítka na začátek. Následně pomocí dotazu na databázi zjistíme, jak má vypadat hlavička dotazu pro prediktivní model. Hlavička, jak je uvedeno v kapitole 5.1, musí být stejná jako u tréninkových dat. Poté je sestaven JSON objekt, jehož obsah je kompatibilní se vstupem prediktivního modelu.

```
{"header":["id", "tile","_3c0e23690bf1","_b000b420c451",
      "_20bbc0e43631", "_20bbc03beba1", "..."],
"data":[[-200, -200, -88, -90, -200, -75, "..."],[-200,
       -200, -88, -70, -85, -200, "..."]],
"tablename":"scoreInput"
}
```

Na ukázce výše je zkrácená verze JSON objektu posílaného prediktivnímu modelu. Hodnota -200 je standardní hodnota pro všechny atributy, u nichž reálnou hodnotu neznáme. Stejná hodnota byla použita i při trénování algoritmů. Odpovědí je podobný JSON objekt jako jsme poslali na vstup, v jeho hlavičce jsou ale navíc sloupce, reprezentující výstup klasifikačních algoritmů a v polích obsahující data jsou příslušné hodnoty výstupů.

Pokud server obdrží od klienta více dat k vyhodnocení, předpokládá, že byly nasbírány na jednom místě. Výsledky klasifikátorů jsou použity k hlasování. Při hlasování mají oba klasifikační algoritmy stejnou váhu hlasu. V případě stejného

počtu hlasů u některých z výsledků je zvolen ten, jehož první výskyt byl nalezen první

6 Testování

Testování funkčnosti lokalizačního řešení bylo prováděno ve dvou krocích. V takzvaných laboratorních testech a potom v reálném provozu, kdy byla nahrána testovací aplikace do několika mobilních telefonů a uživatelé zkoušeli zaznamenávat úspěšnost lokalizace v přízemí budovy Q Provozně Ekonomické Fakulty na Mendelově Univerzitě v Brně. V následujících podkapitolách budou shrnuty výsledky získané při testování.

6.1 Laboratorní testy

Laboratorní testy se od reálných liší tím, že používají data primárně určena k trénování klasifikačních algoritmů. Data jsou dle potřeby rozdělena na dvě disjunktní množiny, většinou různé velikosti a následně je jedna z množin použita pro trénink klasifikačních algoritmů a druhá pro jejich testování. Testuje se tedy na reálných datech, které klasifikační algoritmy neznají. Data byla do množin rozdělována pomocí lineárního kongruentního generátoru náhodných čísel, předem daný byl pouze poměr, v jakém mají být data rozdělena, seed byl volen náhodně. Celkem bylo zvoleno šest testovacích sad. Testovací sada seskupuje testy podle poměru množství testovacích a trénovacích dat. V rámci každé testovací sady bylo provedeno deset iterací s různým seedem kongruentního generátoru tak, aby byly trénovací a testovací množiny různé v každé z iterací. Použití generátoru náhodných čísel pro rozdělení množin má za následek to, že se jejich velikost v různých iteracích může mírně lišit, proto budou v následujících tabulkách uváděny průměrné velkosti trénovacích a testovacích množin. V rámci testování jsem se pokud možno snažil zachovat konstantní velikost trénovací množiny a to 30% z celkového množství dat, jediná výjimka byla provedena u trénovací množiny, jež obsahovala 80% dat, zde obsahovala testovací množina jen zbylých 20%.

Výše popsaný způsob testování má oproti reálným testům značnou výhodu v tom, že časová složitost testů je mnohem menší. Vše totiž probíhá naprosto automatizovaně stejně jako trénink algoritmů. Stovky až tisíce testovacích příkladů jsou otestovány během několika minut. Velké množství testovacích dat a snadná manipulace s nimi, tedy umožňuje testovat prediktivní model podrobně. Další výhodou tohoto způsobu testování je naprosto přesné porovnání výsledků. Testovací data totiž nesou informaci o tom, na jakém políčku mapy byla pořízena společně a v jeden čas s trénovacími daty. Tato vlastnost může být z opačného hlediska považována za nevýhodu, protože při reálném použití uživatel nestojí na přesně stejném místě, na kterém byla testovací data pořízena. Jiné pravděpodobně bude i postavení uživatele či jeho otočení. Reálné výsledky tedy mohou být ovlivněny spoustou dalších proměnných, které toto testování nebere v úvahu. Druhou nevýhodou je, že model není nikdy natrénován stejným způsobem jako při reálném nasazení. Trénovací množina je totiž ochuzena o testovací část. Tento neduh je do značné míry elimino-

ván iterativním testováním s různě rozdělenými daty do testovacích a trénovacích množin.

Hodnota výsledků tedy spočívá v trochu jiné rovině. Ukazuje schopnost algoritmů pracovat se vstupním vektorem a objevovat souvislosti v trénovací množině dat.

Před tím, než budou uvedeny výsledky testování, se patří uvést informace o nasbíraných datech, se kterými se provádělo trénování i testování. Záznamů je příliš mnoho, než aby bylo možné je v rámci práce prezentovat. V tabulce 1 je tedy alespoň malý souhrn.

Tabulka 1: Informace o sesbíraných datech

počet záznamů	počet měřených políček	poč. záznamů/políčko	poč. AP/políčko
5700	168	33,9	7,53

Celkově bylo pro trénování klasifikačních algoritmů sesbíráno 5700 unikátních záznamů, toto číslo není nijak zaokrouhleno a je přesné. Ze 400 možných políček na mapě, byl trénink proveden na 168 z nich. Pokrytí přízemního podlaží je ukázáno na obrázku 10. Byly využity pouze veřejné prostory přízemního podlaží a to tak aby sbírání dat jakkoli nenarušovalo provoz školy. Na jednom políčku bylo za 25 sekund měření sesbíráno průměrně 33,9 unikátních záznamů. V každém z těchto záznamů bylo průměrně zaznamenáno 7,53 vysílačů vysílajících síť „eduroam“, přičemž maximální počet zaznamenávaných AP byl omezen na 8.

Tabulka 2: Testovací data

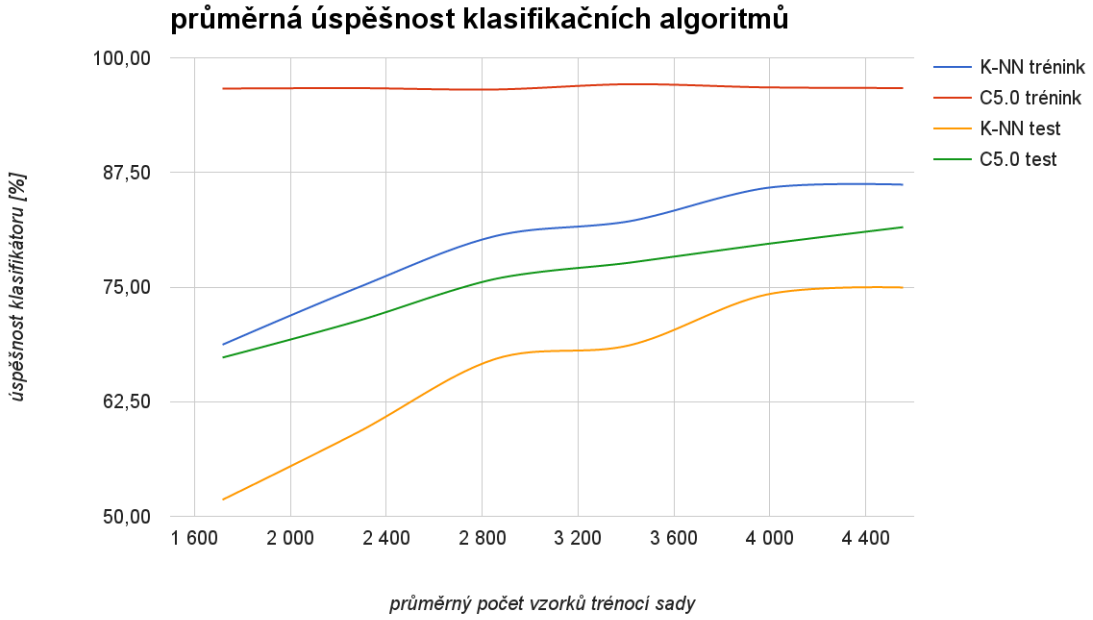
tr. sada	tst. sada	poměr tr/tst [%/%]	sm. odch. tr. [vz.]	sm. odch. tst [vz.]
4 556	1 144	80/20	33,8	33,8
3 986	1 714	70/30	30,9	30,9
3 415	1 713	60/30	35,8	49,8
2 853	1 701	50/30	40,1	24,5
2 284	1 716	40/30	33,5	28,5
1 717	1 707	30/30	36,8	49,5

V tabulce 2 jsou obsaženy údaje o tom, jak byla data v jednotlivých testovacích sadách rozdělena. První dva sloupce z leva obsahují počet vzorků trénovacích a testovacích dat pro jednotlivé sady. Další sloupec obsahuje procentuální rozdělení, kterého mělo být dosaženo, a zbylé dva sloupce obsahují směrodatnou odchylku, které se při rozdělování dopustil generátor náhodných čísel. Směrodatná odchylka je vztažena k počtu vybraných vzorků.

Tabulka 3: Výsledky

poměr tr/tst [%/%]	K-NN tr. [%]	K-NN tst. [%]	C5.0 tr. [%]	C5.0 tst. [%]
80/20	86,17	74,95	96,70	81,53
70/30	85,812	74,181	96,775	79,698
60/30	82,181	68,653	97,127	77,659
50/30	80,552	67,146	96,546	75,881
40/30	74,973	59,197	96,694	71,331
30/30	68,719	51,809	96,649	67,313

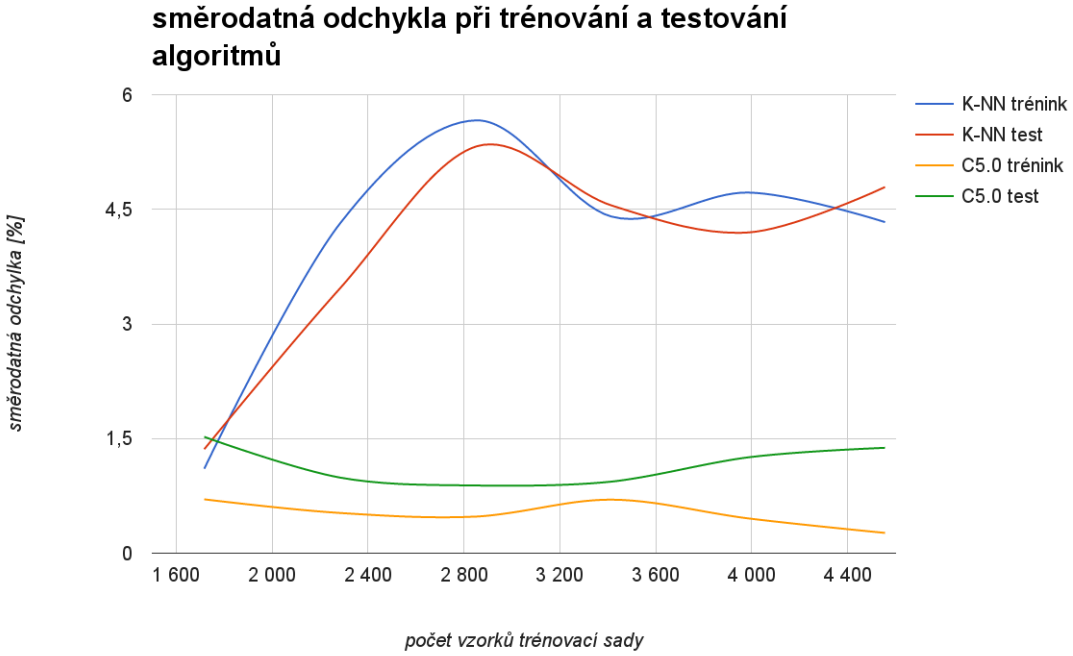
Tabulka 3 obsahuje výsledky testování pro jednotlivé testovací sady. Ty jsou odlišeny procentuálním poměrem trénovacích a testovacích dat. Přesnější údaje o testovacích sadách obsahuje tabulka 2. Další sloupce obsahují průměrnou procentuální úspěšnost obou algoritmů, a to jak na trénovací množině dat, tak i na testovací.



Obrázek 19: Graf výsledků

Na obrázku 19 je graf s výsledky testování a trénování algoritmů. Je jasně vidět, že úspěšnost správné predikce je závislá na velikosti trénovací množiny. Úspěšnost algoritmů při testech na trénovacích množině dat je značně větší než na testovací množině. Zejména u algoritmu C5.0 je znatelně větší rozdíl při trénovaní na menší množině dat, než u algoritmu K-NN. To je ale způsobeno vysokou přesností algoritmu C5.0 testovaného na tréninkových datech. Z grafu je patrné, že úspěšnost

algoritmu C5.0 se při testování na trénovacích datech prakticky nemění. Algoritmus K-NN má úspěšnost při tréninku mnohem pozvolnější a s přibývajícími daty se zlepšuje jeho úspěšnost jak na trénovací, tak i na testovací množině dat. Zvláštním úkazem je srovnání křivky růstu úspěšnosti při použití 50 a 60 procent dostupných tréninkových dat. Zde se totitž neprojevilo žádné zlepšení úspěšnosti klasifikátoru.



Obrázek 20: Graf směrodatné odchylky

Směrodatná odchylka udává, jak moc se při jednotlivých iteracích v testovacích sadách výsledky lišily. Zajímavé je, že ačkoli úspěšnost trénovacích algoritmů se s přibývajícími trénovacími daty zlepšuje, směrodatná odchylka má tendenci ustálit se na nějaké úrovni a kolem ní se držet. Zajímavé také je, že směrodatná odchylka se u testování na trénovacích a testovacích datech příliš neliší. Z grafu na obrázku 20 je patrné, že algoritmus K-NN má se spolehlivostí predikcí o poznání větší problém než algoritmus C5.0. Směrodatná odchylka začíná poměrně nízko, protože s menším počtem dat v trénovací množině byly jeho výsledky konzistentnější, přesto že spěšnost byla celkově nižší. S přibývajícími daty se úspěšnost algoritmu značně zlepšila, nicméně se u něj začaly projevovat větší výkyvy. Úspěšnost algoritmu C5.0 je napříč iteracemi v testovacích sadách konstantní. Celkově dopadl v laboratorních testech lépe algoritmus C5.0, jenž měl vyšší úspěšnost predikcí s nižší odchylkou. Nicméně oba algoritmy měly dostatečně dobré výsledky pro to, aby se daly nasadit v rámci prediktivního modelu pro reálné využití. S osmdesáti procenty celkových dat měl algoritmus K-NN úspěšnost 76,2% a algoritmus C5.0 81,5%. U algoritmů ukazuje

křivka v grafu rostoucí trend úspěšnosti s přibývajícími trénovacími daty. Lze tedy předpokládat, že po přidání zbylých dvaceti procent dat, se úspěšnost algoritmů ještě zlepší.

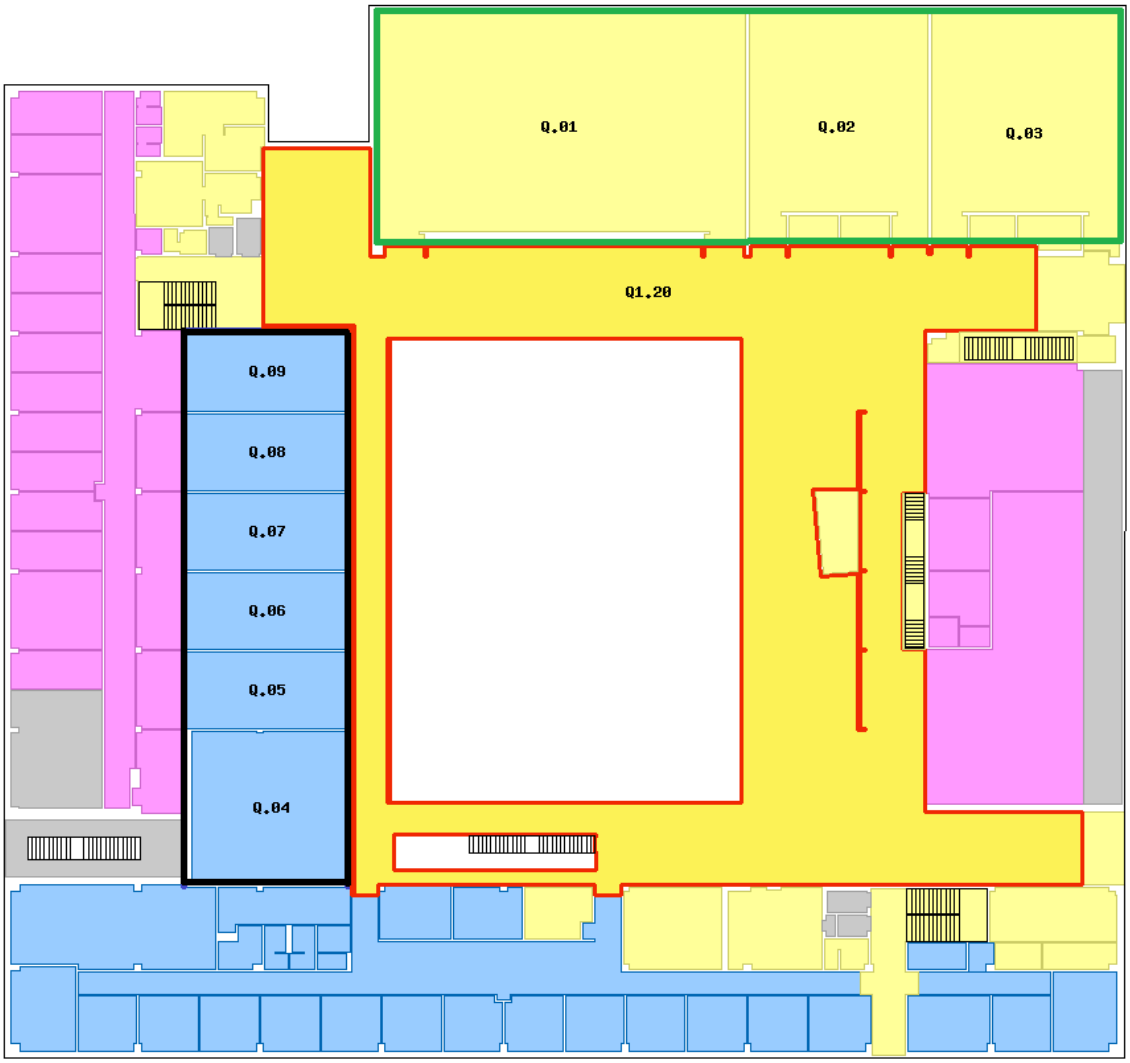
6.2 Reálné testy

Reálné testy byly prováděny se třemi mobilními telefony v podvečerních hodinách. Provoz v budově, tedy nebyl příliš hustý, přesto se však po přízemním podlaží pohybovali lidé. Čas testování byl z volen tak, aby sbírání dat mohlo proběhnout i v místnostech, kde přes den běžně probíhá výuka.

Metodika testování

Testeři dostali za úkol pohybovat se po přízemí budovy Q a náhodně zkoušet, zda je testovací aplikace schopna zobrazit jejich aktuální polohu. Během testování se pohybovali v oblastech vyznačených na obrázku 21. Po každém pokusu o určení polohy zaznamenali do záznamového archu, zda byla jejich poloha určena správně či nikoli. U negativních výsledků měli také za úkol označit ty, které byly naprosto špatně.

Vyhodnocování výsledků ze strany testerů bylo do značné míry subjektivní. Instrukce při testování byly takové, že pokud aplikace označí polohu testera na mapě s přesností přibližně na jeden okolní čtverec a zároveň ho umístí do správné místnosti, je výsledek označen za správný. U špatných výsledků byli testeři instruování, aby označili pokusy, u nichž lokalizace naprosto selhala, tedy ty pokusy, kde se lokalizace netrefila ani do správného kvadrantu budovy.



Obrázek 21: Mapa přízemí budovy Q s vyznačenými oblastmi

Na obrázku 21 jsou barevně obtaženy tři oblasti, v nichž bylo prováděno testování. Jsou to místnosti, kde probíhají počítačová cvičení, velké přednáškové haly a chodba (aula) přízemního podlaží.

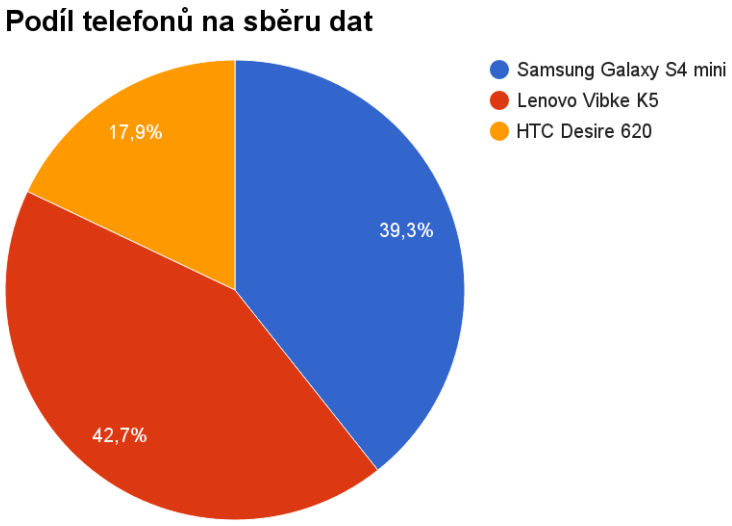
- černá - malé místnosti
- červená - chodba
- zelená - velké místnosti

Analýza dat

Jako první je nutné i v tomto případě uvést souhrnné informace o testovacích datech. Tentokrát bylo testovacích dat podstatně méně než v případě laboratorních testů, ale i přesto je tabulka příliš velká na to, aby zde byla celá uvedena. Při analýze je potřeba brát v úvahu, že na různých telefonech bylo nasbíráno různé množství dat. Následkem je, že u celkových průměrů, kdy výsledek nemá být závislý na testovaných telefonech, mohou být data zkreslena, protože množství nasbíraných dat na jednom telefonu převáží ostatní. Tento neduh může být eliminován tím, že nejprve budou spočítány průměry dle jednotlivých telefonů a až ty budou zprůměrovány do celku. Tento postup bude zajišťovat to, že data z každého telefonu budou mít stejnou váhu. Nevýhodou tohoto postupu je však to, že i malé množství dat může výrazně ovlivnit výsledky, protože jejich váha bude představovat jednu třetinu celku. Tam, kde bude takovéto zkreslení hrozit, budou použity oba postupy a uvedeny dvoje výsledky. Na tyto případy pak bude v textu upozorněno.

Tabulka 4: Přehled provedených testů

	chodba	malá místnost	velká místnost	celkem
počet testů	58	24	35	117



Obrázek 22: Graf podílu jednotlivých telefonů na testování

Přesto že se na testování podílelo několik lidí, počet provedených testů se zdaleka neblíží laboratorním testům. I z menšího množství dat se však dají vyčíst zajímavé informace.

Tabulka 5: Výsledky vypočítané z průměrů jednotlivých telefonů

	chodba	malá místnost	velká místnost	celkem
úspěšnost lokalizace [%]	56,3	55,7	57,3	56,7

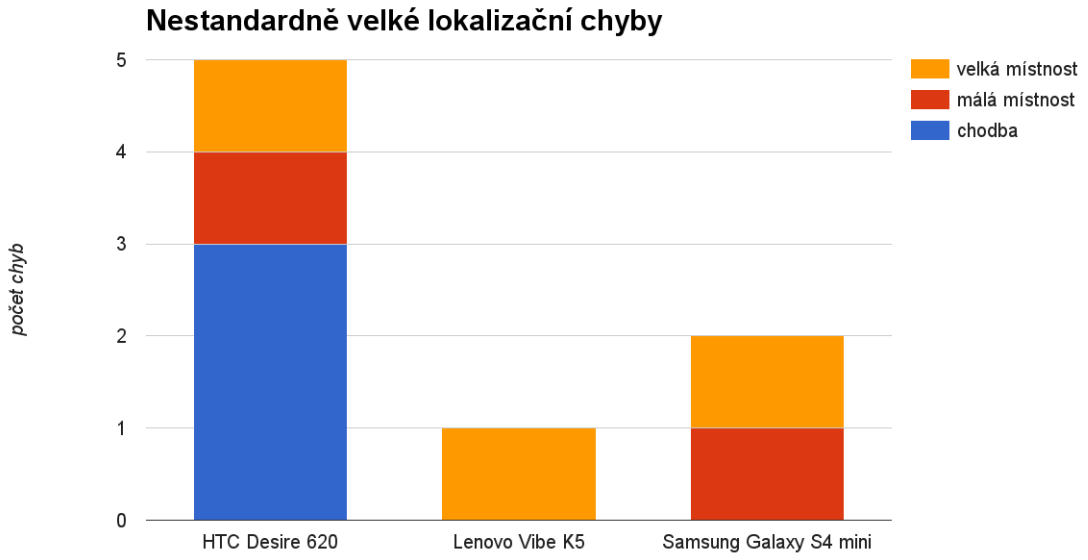
V tabulce 5 se nachází celkové výsledky v závislosti na místnosti, ve níž byly testy prováděny. Uveden je i celkový výsledek. Výsledky neberou v potaz nižší podíl dat pocházejících z telefonu HTC Desire 620, především u jednotlivých místností je dat příliš málo na to, aby mohla být považována za reprezentativní. V tabulce 6 se nachází výsledky, které byly spočítány přímo, bez předchozího průměrování dle telefonů. V tomto případě mají tedy výsledky z telefonu HTC Desire menší váhu, protože od něj pochází méně dat. Jak si můžeme všimnout, tak v takovém případě mírně stoupla celková úspěšnost lokalizace.

Tabulka 6: Výsledky vypočítané z čistých dat bez ohledu na telefon

	chodba	malá místnost	velká místnost	celkem
úspěšnost lokalizace [%]	59,6	54,2	62,9	59,8

Testování odhalilo, že úspěšnost lokalizace je do jisté míry závislá na tom v jakém typu místnosti se tester nachází. Testování v malých místnostech dosahuje v obou případech nejhorších výsledků. To může být do značné míry způsobeno metodikou měření, kdy tester může tolerovat nepřesnost z hlediska políčka na mapě, nicméně musí být správně určena místnost. Dalším faktorem může být to, že signál se do místností šíří skrze stěny a na malém prostoru se více projeví jeho odrazy. Naproti tomu lokalizace ve velkých otevřených prostorách je úspěšnější. Signál je v těchto prostorách méně ovlivněn okolními stěnami i odrazem. Také metodika testování z principu více vyhovuje testování ve větších místnostech.

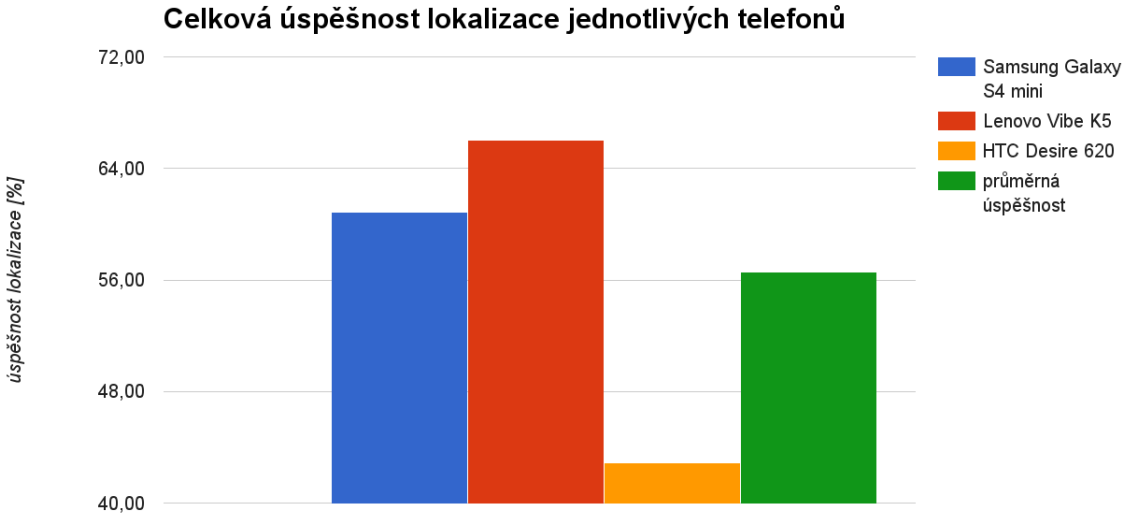
Metodika testování obsahuje i instrukce zaznamenávat opravdu velké nepřesnosti při lokalizaci. Celkem bylo zaznamenáno 9 takových případů, což tvoří celkem 7,7% testů.



Obrázek 23: Graf velkých nepřesností při lokalizaci rozdělené dle telefonů a typu místnosti

Graf na obrázku 23 ukazuje, na kterých telefonech a v jakých místnostech byly tyto chyby zaznamenány. Jak je vidět, tak víc jak polovina případů byla zaznamenána na telefonu HTC Desire 620.

V předchozích testech je patrné, že některé telefony si při testování počínají lépe jak jiné. Porovnat tedy úspěšnost lokalizace jednotlivých mobilních telefonů je důležité pro zhodnocení celkové funkčnosti lokalizační služby. V grafu na obrázku 24 je znázorněna celková úspěšnost lokalizace pro jednotlivé mobilní telefony. Nejúspěšněji byl lokalizován telefon Lenovo, tento telefon byl zároveň v minulosti použit pro sběr trénovacích dat. O pouhých 6% hůře dopadl Samsung Galaxy S4 mini, ten se na sbírání trénovacích dat nikdy nepodílel. Telefon od HTC se evidentně pro využití lokalizační služby příliš nehodí, u něj úspěšnost nedosáhla ani 43%.

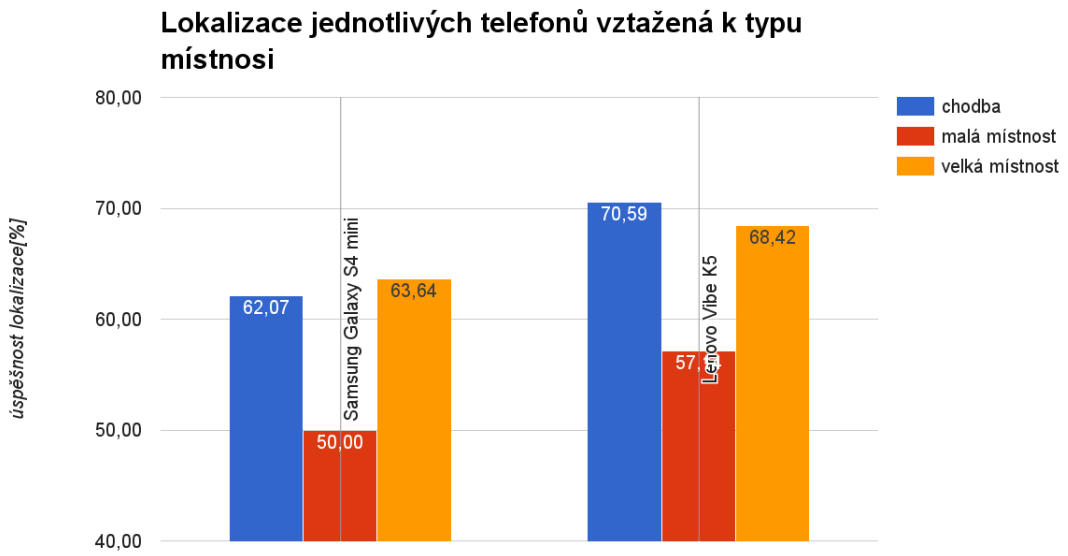


Obrázek 24: Graf úspěšnosti lokalizace při testování dle jednotlivých mobilních telefonů

Posledním srovnáním je porovnání úspěšnosti lokalizace jednotlivých telefonů v závislosti na místnosti, v níž probíhal test. Z toho srovnání byl vyřazen telefon HTC Desire 620, protože u něj byla množina dat příliš malá na to, aby mohla být rozdělena na tři menší podmnožiny. Z grafu na obrázku 25 můžeme vyčíst, že u obou telefonů byla úspěšnost v otevřených prostorách znatelně vyšší než v menších místnostech. Mezi oběma velkoprostorovými oblastmi nebyl v rámci testování na jednom telefonu příliš velký rozdíl. U obou se projevil pokles úspěšnosti lokalizace v malých místnostech o zhruba 12%.

Zhodnocení testů

Jak se ukázalo v testech, po nasazení v reálných podmínkách úspěšnost lokalizace klesla. Pokud by se úspěšnost pohybovala stabilně okolo 70%, mohlo by být řešení použito v kombinaci s jinými projekty. Při testování se však projevily neduhy, jež mohou takovéto použití značně zkomplikovat. Především se jedná o různou kvalitu Wi-Fi přijímačů uvnitř mobilních telefonů. U telefonu HTC Desire 620 byla úspěšnost testů znatelně nižší, než u Samsungu Galaxy S4 mini nebo Lenova K5. Vyloučit můžeme chybu testera, protože postup testování byl překontrolován a nevybočoval ze standardu jakým byly testy prováděny na ostatních dvou přístrojích. Druhou komplikací je nestabilita implementací ovladačů Wi-Fi karet v operačním systému Android. Na čtvrtém telefonu, se kterým původně mělo být taktéž testováno (jednalo se o Samsung Galaxy J5), se nepodařilo sesbírat data, která by mohla být odeslána



Obrázek 25: Graf úspěšnosti lokalizace Samsungu Galaxy S4 mini a Lenova Vibe K5 v závislosti na místnosti

na server. Problém je způsoben chybnou implementací třídy *WifiManager* na tomto zařízení, protože její metoda *getScanResults()* neposkytuje průběžně aktualizované výsledky. K nasazení a použití by tedy bylo potřeba vytvořit seznam kompatibilních telefonů. Na telefonech Samsung Galaxy S4 a Lenovo Vibe K5 lokalizační služba fungovala v rámci možností dobře, se solidními výsledky a to především v otevřených prostorách.

7 Závěr

Úkolem tohoto projektu bylo vytvořit službu, kterou by mohly využívat mobilní přístroje k určení polohy uvnitř budovy a potom ji na ukázkové aplikaci otestovat.

Pro aproximaci polohy byly použity klasifikační algoritmy K-NN a C5.0. O jejich implementaci se postarala společnost IBM, od které zároveň pochází software, jenž byl při vytváření lokalizační služby použit. IBM SPSS Modeler je software zaměřený na dolování dat a poskytuje přístup ke zmiňovaným algoritmům. V tomto programu byl vytvořen model s natrénovanými algoritmy. Formát tohoto modelu je kompatibilní se cloudovou službou IBM Predictive Analysis, do které byl tento model nahrán. Služba IBM Predictive Analysis s nahraným modelem poté představovala výpočetní jednotku lokalizační služby.

Demonstrační aplikace byla vytvořena pro operační systém Android, aplikace byla použita jak ke sběru dat, tak i k testování lokalizační služby.

Během testování se ukázalo, že ačkoli byly algoritmy natrénovány na poměrně velkém vzorku dat (5700 záznamů), úspěšné určení polohy v reálném prostředí může být problém. Kvalita lokalizační služby totiž pro různá mobilní zařízení kolísá. Některé mobilní telefony s lokalizační službou fungují dobře, jiné méně. Samsung Galaxy S4 Mini a Lenovo Vibe K5 zaznamenaly při testování služby 61 a 66 procent úspěšnosti. Naproti tomu na telefonu HTC Desire 620 byla úspěšnost lokalizace při testech necelých 43%. Řešení problému rozdílnosti kvalit Wi-Fi karet v telefonech není jednoduché, protože problém se nachází na straně koncového zařízení (telefonu). V tomto případě by nepomohlo ani trénování modelu na datech pocházejících z více zařízení. Při takovém postupu by data z jednoho telefonu znamenala šum v datech při určování polohy telefonu jiného. Řešením by mohlo být vytvoření více samostatných modelů, kde by každý byl natrénován na datech z jiného telefonu. Při prvním spuštění aplikace využívající lokalizační službu by uživatel musel ručně nakalibrovat aplikaci tak, aby její dotazy směřovaly na co nejpodobnější lokalizační model. Z testování také vyplynulo, že použití lokalizační služby v menší prostorách, snižuje kvalitu výsledků. Služba se tedy více hodí pro nasazení do otevřených hal, aul a chodeb. Pokud porovnáme výsledky s rešeršovanými texty, úspěšnost lokalizace implementované v rámci této práce je nižší. V tomto případě je potřeba brát v úvahu, že většina prací používá ke sběru dat a testování jedno a to samé zařízení, přičemž se většinou jedná o notebook s podstatně kvalitnějším příjmem signálu, než jakým disponují mobilní telefony.

Celkově se dá říci, že vytvoření lokalizační služby bylo úspěšné, v praxi je ale potřeba počítat s limitacemi, které toto řešení budou vždy provázet. K minimalizaci technických problémů by mohla být použita kombinace více podobných služeb, používající různé technologie Wi-Fi, Bluetooth, Piko buňky s 3G a LTE signálem a tak podobně.

8 Reference

AHMAD, Uzair; GAVRILOV, Andrey; LEE, Sungyoung. *In-building Localization using Neural Networks*. In: 2006 IEEE International Conference on Engineering of Intelligent Systems. IEEE. p. 1-6..

ALTINI, Marco, et al. *Bluetooth indoor localization with multiple neural networks*. In: Wireless Pervasive Computing (ISWPC), 2010 5th IEEE International Symposium on. IEEE, 2010. p. 295-300. .

AMARI, Shun-ichi; WU, Si. *Improving support vector machine classifiers by modifying kernel functions*. Neural Networks, 1999, 12.6: 783-789..

Android Developers [online]. Mountain View (Kalifornie): Alpahabet [cit. 2016-12-27]. Dostupné z: https://developer.android.com/about/dashboards/index.html.

BRUNATO, Mauro; BATTITI, Roberto. *Statistical learning theory for location fingerprinting in wireless LANs*. Computer Networks, 2005, 47.6: 825-845. .

BRUNATO, Mauro; KISS KALLO, Csaba. *Transparent location fingerprinting for wireless services*. 2002. .

BUJLOW, Tomasz; RIAZ, Tahir; PEDERSEN, Jens Myrup. *A method for classification of network traffic based on C5. 0 Machine Learning Algorithm*. In: Computing, Networking and Communications (ICNC), 2012 International Conference on. IEEE, 2012. p. 237-241..

ELNAHRAWY, Eiman; LI, Xiaoyan; MARTIN, Richard P. *The limits of localization using signal strength: A comparative study*. In: Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on. IEEE, 2004. p. 406-414. .

GRANT, Allen. *Beginning Android 4*. New York: Apress, 2012. ISBN 978-1-4302-3984-0..

HASHIMI, Sayed Y.; KOMATINENI, Satya. *Pro Android*. Berkely: Apress, 2009. ISBN 978-1-4302-1596-7.

HUNTER, Jason; CRAWFORD, William. *Java servlet programming*. " O'Reilly Media, Inc.", 2001..

IBM. *IBM SPSS Modeler 17 Modeling Nodes*. SPSS Modeler 17.0 Documentation. [online]. Armonk (New York)[cit. 2016-12-29]. Dostupné z: goo.gl/nlSiw4.

K-Nearest Neighbors [online]. Oklahoma: Statistica [cit. 2016-12-27]. Dostupné z: http://www.statsoft.com/Textbook/k-Nearest-Neighbors.

K-Nearest Neighbors for Machine Learning [online]. Brownlee, 2016 [cit. 2016-12-27]. Dostupné z: http://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/.

KOLYAIE, Samira; YAGHOOTI, Marjan; MAJIDI, Gilda. *Analysis and simulation of wireless signal propagation applying geostatistical interpolation techniques.* Archiwum Fotogrametrii, Kartografii i Teledetekcji, 2011, 22. .

MURPHY, Mark L. *The busy coder's guide to Android development.* United States: CommonsWare, 2008..

TRAN, Duc A.; NGUYEN, Thinh. *Localization in wireless sensor networks based on support vector machines.* IEEE Transactions on Parallel and Distributed Systems, 2008, 19.7: 981-994. .

UJBÁNYAI, Miroslav. *Programujeme pro android.* Praha: Grada Publishing as, 2012. ISBN 978-80-247-3995-3..