ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA ELEKTROTECHNICKÁ

KATEDRA KYBERNETIKY

Modelování vlivu demografie na politické preference

Modelling the Influence of Demography on Political Preferences

Bakalářská práce

Studijní program: Otevřená informatika

Studijní obor: Informatika a počítačové vědy

Vedoucí práce: RNDr. Michal Čertický, Ph.D.

Kamil Švec

Praha, prosinec 2018

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Švec** Jméno: **Kamil** Osobní číslo: **435029**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra kybernetiky**

Studijní program: **Otevřená informatika**

Studijní obor: **Informatika a počítačové vědy**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Modelování vlivu demografie na politické preference

Název bakalářské práce anglicky:

Modelling the Influence of Demography on Political Preferences

Pokyny pro vypracování:

Cílem práce je implementace modelu volebních výsledků v ČR pomocí metod strojového učení - např. neuronových sítí. V rámci přípravy student vytvoří scripty pro automatické stahování historických výsledků voleb na úrovni volebních okrsků a geografických hranic samotných okrsků z veřejných zdrojů. Tyto data zkombinuje s demografií populace žijící na daném území. Student provede experimenty s různými typy a konfiguracemi modelu a porovná jejich přesnost.

Seznam doporučené literatury:

[1] Anderson, James A. An introduction to neural networks. MIT press, 1995
[2] Zhang, Guoqiang Peter. Neural networks for classification: a survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 30.4 (2000): 451-462
[3] Abadi, Matrin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467 (2016)

Jméno a pracoviště vedoucí(ho) bakalářské práce:

RNDr. Michal Čertický, Ph.D., katedra počítačů FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **05.01.2018** Termín odevzdání bakalářské práce: **08.01.2019**

Platnost zadání bakalářské práce: **30.09.2019**

RNDr. Michal Čertický, Ph.D.	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Ing. Pavel Ripka, CSc.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studenta

Prohlášení autora práce

„Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.“

V Praze dne ……. …….
podpis autora práce

Poděkování:

Mé poděkování patří RNDr. Michalu Čertickému, Ph.D. za odborné vedení práce a ochotu, se kterou se mi v průběhu jejího vypracování věnoval.

Abstrakt:

Tato práce se zabývá tvorbou modelu politických preferencí v České Republice na úrovni jednotlivých volebních okrsků. Zahrnuje sběr dat z veřejně dostupných zdrojů, přípravu trénovací množiny, implementaci modelů a jejich porovnání a také následnou prezentaci modelu v uživatelsky přívětivé podobě. Využita jsou data o volebních výsledcích na úrovni volebních okrsků a k nim komplementární geografická data. Modely jsou implementovány pomocí metody strojového učení - neuronových sítí. K prezentaci modelů byla vytvořena webová aplikace.

Klíčová slova:

volby, strojové učení, shapefile, neuronová síť

Abstract:

This thesis deals with the problem of creation of a model of political preferences in the Czech Republic on an electoral district level. It includes collection of data from publicly available sources, preparation of training set, implementation of models and their comparison as well as their subsequent presentation in a user-friendly way. It uses election results data on an electoral district level and to them complementary geographic data. Models are implemented using a machine learning method – neural networks. A web application was created for the presentation of these models.

Keywords:

election, machine learning, shapefile, neural network

Obsah

Seznam obrázků:

Seznam Tabulek:

1 Úvod

1.1 Popis řešeného problému

Cílem této práce je vytvoření modelu závislosti politických preferencí na demografických údajích. Tento úkol přirozeně vede na rozdělení na čtyři hlavní podproblémy: sběr dat, jejich zpracování, tvorba samotného modelu a následná uživatelsky přívětivá prezentace modelu.

V první části je potřeba shromáždit dostatečně velkou množinu demografických a politických dat. Dále je potřeba tyto data připravit pro strojové učení. V našem případě to znamená vytvořit množinu párů demografických údajů a k nim odpovídajících volebních preferencí. Poté přijde na řadu samotné učení a porovnávání konfigurací modelů. Posledním úkolem je připravení způsobu představení modelu v lidsky přívětivé podobě.

1.2 Popis navrhovaného řešení

K těmto čtyřem podproblémům práce překládá čtyři navrhovaná řešení.

Aby bylo vytvoření takového modelu možné, je potřeba shromáždit dostatečně velkou množinu trénovacích dat. Jako data o volebních preferencích použijeme výsledky voleb do Poslanecké sněmovny Parlamentu České republiky, které se konaly v pátek 20. a v sobotu 21. října 2017. K nim je zapotřebí odpovídajících demografických dat. Ty pocházejí z multi-agentního aktivitního modelu populace, publikovaného v [1]. Každá jejich položka představuje jednoho obyvatele žijícího na území určeném zeměpisnými souřadnicemi. Aby bylo dosaženo co možná nejlepší přesnosti modelu, využijí se volební data z co nejmenší zaznamenané oblasti. To znamená z jednotlivých volebních okrsků. Aby ale bylo možné přiřadit obyvatele jejich okrskům, je potřeba znát i přesné umístění a hranice každého takového okrsku. Tato geografická data jsou poskytována Českým úřadem zeměměřičským a katastrálním na jeho webových stránkách.

Bylo učiněno rozhodnutí, že volební data budou sesbírána automatickými skripty ze stránek Českého statistického úřadu (ČSÚ), ačkoliv ČSÚ tyto poskytuje i ve strojově čitelné podobě XLSX či XML souborů. Existuje pro to řada důvodů. Mají příliš složitou a nám nevyhovující strukturu a nevhodně míchají kódy městských částí a obcí, což vytváří potřebu jejich dohledávání v číselnících pro pozdější kompatibilitu s označením geografických dat o okrscích. Naopak stránky ČSÚ „volby.cz“ určené pro širokou veřejnost obsahují přesně ty informace, které jsou potřeba pro účely této práce.

Další na řadě je jejich zpracování. Tím se rozumí přiřazování okrsků jednotlivým obyvatelům (tedy jejich zeměpisným souřadnicím) pomocí PostgreSQL databáze

s geografickým rozšířením PostGIS. Následně jsou spočítány celkové demografické statistiky všech okrsků a ty jsou spojeny s volebními daty do trénovacích příkladů.

Třetím podproblémem je vytvoření samotného modelu pomocí strojového učení. Vytvoříme regresní model, který na základě vektoru demografických statistik odhadne vektor volebních výsledků. K tomu práce používá neuronové sítě implementované v jazyce Python za pomoci knihovny Keras [2] (využívající TensorFlow [3]). V této části práce porovnává nejrůznější konfigurace takového modelu. Je nutné připomenout, že náš model zanedbává volební účast.

V poslední řadě práce řeší možnosti prezentace takového modelu v lidsky srozumitelné podobě. K tomu účelu byla vytvořena webová aplikace s grafickým uživatelským rozhraním, které umožňuje uživateli zobrazit výsledky takového modelu pro libovolné okrsky zvolené na mapě České republiky. Dále umožňuje namodelování demografických dat pro vlastní okrsek. K tomuto využijeme

1.3 Struktura práce

Práce má 7 kapitol. V následující kapitole osvětluje některé teoretické koncepty týkající se neuronových sítí. Kapitola 3 se věnuje procesu získání a zpracování dat. V kapitole 4 se pojednává o implementaci jednotlivých částí práce. Ta je rozdělena na 4 části tak, jak bylo výše navrženo. V kapitole 5 se věnujeme tvorbě a srovnání jednotlivých modelů. Zbývá kapitola 6, kde ukazujeme funkcionalitu webové prezentační aplikace.

2 Teorie

2.1 Neuronové sítě

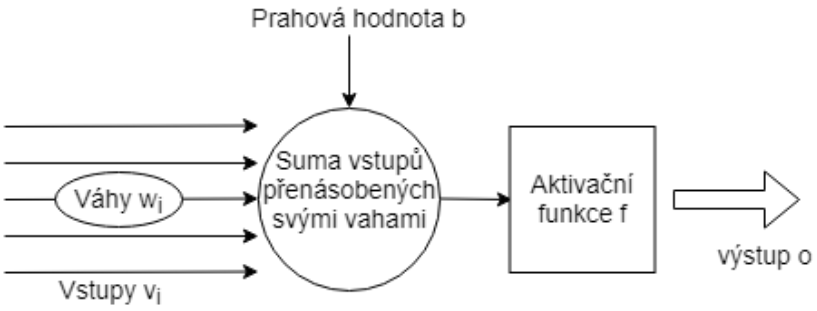
Myšlenka neuronových sítí jako výpočetního modelu je matně inspirována strukturou lidského mozku. V něm najdeme obrovské množství neuronů propojených spoji tzv. synapsemi. A pomocí těchto spojů může mozek reorganizovat strukturu svých součástí - neuronů - tak, aby odpovídala reálným problémům, se kterými se člověk setkává. Tato plasticita mozku umožňuje vyvozovat abstraktní pravidla a souvislosti. [4]

Umělé neuronové sítě v kontextu počítačových věd jsou výrazně jednodušší. Skládají se z umělých neuronů uspořádaných do vrstev. Zde je neuron reprezentován svými vstupy, tzv. prahovou hodnotou (anglicky Bias) a aktivační funkcí, která určí finální hodnotu, která se bude sítí propagovat dále. Vstupy neuronu mají navíc přiřazeny váhy. Právě tyto váhy reprezentují významnost jednotlivých spojení. Vyjádřeno matematicky:

$$o = f(\sum(w_i * v_i) + b)$$

kde f je aktivační funkce, w_i jsou váhy vstupů, v_i jsou vstupy, b je prahová hodnota a o je výstup neuronu

A ještě schematicky (schéma inspirováno [5]):



Obr. 2:1 – Schéma neuronu

Tyto neurony jsou uspořádány do minimálně dvou vrstev, vstupní a výstupní. Vrstvy mezi nimi se nazývají vrstvy skryté. Počet vrstev neuronové sítě stejně jako počet neuronů každé z nich ovlivňuje schopnost sítě vyjadřovat abstraktní schémata. Výběr počtu vrstev, stejně jako výběr počtu neuronů každé z nich, je netriviální úloha, která často vyžaduje nějaké hlubší pochopení zkoumaného problému nebo experimentování. Těmto experimentům se věnujeme v kapitole 5 této práce.

2.1.1 Učení

Učení neuronové sítě je proces hledání vah a prahových hodnot tak, abychom pro zadaný vstup získali požadovaný výstup. V naší práci se věnujeme tzv. učení s učitelem (anglicky supervised learning). To je proces úpravy vah neuronové sítě pomocí párů vstupů a k nim odpovídajících výstupů, takzvaných příkladů. Existuje množství metod používaných pro toto učení. [6] V této práci používáme optimalizátor Adam [7], protože s ním bylo dosahováno nejlepších výsledků.

2.1.2 Aktivační funkce

Aktivační funkce určují výstup neuronu do další vrstvy sítě. Jejich výběr ovlivňuje konvergenci výpočtu učení.

ReLU

ReLU je zkratkou pro Rectified Linear Unit. Poprvé představená v [8]., je nejpoužívanější aktivační funkcí. Protože derivace (gradient) $f(x)$ ReLU je 1 pro všechna kladná x, nedochází k zeslabování gradientu jako například u aktivačních funkcí Sigmoid či Tanh. Tedy, pokud je vstup větší než 0, je zobrazen na výstupu (identita). Pokud je vstup menší nebo roven nule, je výstup nulový.

ReLU se potýká s problémem tzv. mrtvých neuronů. Může dojít k nastavení vah takovým způsobem, že třeba i pro všechny trénovací příklady bude vstup aktivační funkce záporný a výstup tedy vždy nulový. V takovém případě neuron zcela ztrácí jakoukoli rozpoznávací schopnost. Protože gradient bude nulový, nebude docházet k úpravě vah tohoto neuronu.

$$f(x) = \max(x, 0)$$

kde f je aktivační funkce ReLU

LeakyReLU

LeakyReLU pomáhá odstraňovat problém s mrtvými neurony tím, že pro záporné hodnoty dává na výstup vstup násobený velmi malou konstantou. Gradient je tedy malý, ale není nulový. Zůstává proto možnost, že se opraví váhy neuronu a znovu se obnoví jeho schopnost rozpoznávání. [9]

$$f(x) = \begin{cases} x, & pro\ x > 0 \\ \alpha x, & pro\ x \leq 0 \end{cases}$$

kde f je aktivační funkce LeakyReLU, α je konstanta [10]

ELU

Pro kladný vstup opět funguje ELU stejně jako LeakyReLU a ReLU. ELU tedy uchovává hlavní přednost ReLU – odstranění problému „mizejícího“ gradientu. Jak popisuje článek [9], ELU na rozdíl od ReLU umožňuje negativní hodnoty výstupu, což průměrnou hodnotu aktivace přibližuje nule. To podle článku vede k lepšímu a rychlejšímu učení. Aktivační funkce LeakyReLU sice umožňuje záporné hodnoty, ale nedostatečně zaručuje deaktivovaný stav odolný proti šumu. ELU ale má pro malé záporné vstupy větší záporný výstup než LeakyReLU a díky tomu je vůči šumu odolnější.

$$f(x) = \begin{cases} x, & pro\ x \geq 0 \\ \alpha(e^x - 1), & pro\ x < 0 \end{cases}$$

kde f je aktivační funkce ELU, α je konstanta

Softmax

Tato aktivační funkce má pro naše účely dvě klíčové vlastnosti. Zaprvé, její výstup je v intervalu 0 až 1. Zadruhé, součet všech výstupů je 1. Počítá se jako podíl exponenciální funkce vstupu do aktivační funkce o základu e a součtu všech takových exponenciál pro všechny vstupy.

Funkce spočítá rozdělení pravděpodobnosti pro jednotlivé kategorie (reprezentované výstupy neuronů). To odpovídá výstupu jako vektoru podílů zisků hlasů jednotlivých stran vůči celkovému počtu hlasů a tedy je to ideální pro použití s volebními výsledky. V naší síti proto využijeme Softmax jako funkci výstupní vrstvy. [11]

Její vzorec je:

$$f(x_i) = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}}$$

kde f je funkce Softmax, e je Eulerovo číslo, $\boldsymbol{x}$ je vektor vstupů do aktivačních funkcí všech neuronů, i je index neuronu od 1 do K a K je celkový počet neuronů vrstvy se Softmax funkcí

2.1.3 Ztrátová funkce

Jedna z možností, jak vytvořit model odhadující volební preference v jednotlivých volebních okrscích, je uvažovat model, jehož výstupem bude vektor volebních výsledků v určitém okrsku. Takový model je samozřejmě regresní. Je zřejmé, že pro učení takového modelu bude potřeba funkcí vyjadřujících „vzdálenost“ vektoru výsledků předpovězených modelem od vektoru skutečných výsledků.

Mean Squared Error

Mean Squared Error (MSE) je kvadratický průměr chyb (střední hodnota druhých mocnin rozdílů skutečných a předpovězených hodnot). Kvůli druhé mocnině je MSE citlivý na výstřední hodnoty (velkou váhu mají položky s velkou chybou). [12]

MSE vyjadřuje velikost chyby ve stejných jednotkách jako položky vektorů. Je nezáporný a hodnoty bližší nule znamenají menší chybu.

$$e = y - \hat{y}$$

kde e je chyba, y je skutečná hodnota pozorování a $\hat{y}$ je hodnota předpověďi

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(e_i)^2$$

kde MSE je Mean Squared Error, n je počet položek a e_i je položka vektoru chyb [13]

Cross-entropy

Ztrátová funkce Cross-entropy je vhodná ke kombinaci se Softmax funkcí, protože CE vyžaduje vstupy v intervalu 0 až 1. Ač se to na první pohled nezdá, funkce dosahuje minima právě když si vektor pozorování a vektor předpovědí odpovídají. [14]

$$CE = -\sum_i y_i \log(\hat{y}_i)$$

kde CEje Cross-entropy ztrátová funkce, y je skutečná hodnota pozorování a $\hat{y}$ je hodnota předpověďi

2.1.4 Metriky

K vyhodnocení přesnosti modelu jsou potřeba metody hodnotící kvalitu jeho předpovědí. Pro účely hodnocení modelů a jejich porovnání mezi sebou jsou v této práci použita funkce Mean Absolute Error (střední hodnota absolutní hodnoty chyby).

Mean Absolute Error

Mean Absolute Error (MAE) je střední hodnota absolutní hodnoty chyb, kde chybou se rozumí rozdíl hodnot pozorování a hodnot předpovědí. Mean Absolute Error nezahrnuje žádné váhy, chyby na všech znacích (položkách vektorů) tak ovlivňují výslednou hodnotu stejně. Kvůli absolutní hodnotě není důležité znaménko chyby, ale pouze její velikost. MAE vyjadřuje chybu modelu ve stejných jednotkách jako mají položky vektorů. Dosahuje nezáporných hodnot, přičemž nižší hodnoty znamenají menší chybu. [12]

Pro případ voleb můžeme MAE chápat jako podíl špatně přiřazených hlasů průměrně pro každou kandidující stranu.

Následuje vzorec pro chybu a Mean Absolute Error:

$$e = y - \hat{y}$$

kde e je chyba, y je skutečná hodnota pozorování a $\hat{y}$ je předpověď

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |e_i|$$

kde MAE je Mean Absolute Error, n je počet položek, e_i je položka vektoru chyb [15]

3 Data

3.1 Demografie

Údaje o demografii byly dodány vedoucím práce.

Demografické vlastnosti populace byli získány z multi-agentního aktivitního modelu Prahy a Středočeského kraje. Populace je modelována autonomními agenty, s chováním ovlivňovaným jejich sociodemografickými atributy, aktuálními potřebami a situačním kontextem. Výstup tohoto modelu byl použit i v jiných pracích, jako například [16] nebo [17].

Poskytnuté údaje sestávaly z csv souborů obsahujících výčet voličů ve Středočeském a Jihomoravském kraji a Praze. Protože každý volič má přesně určené umístění ve smyslu zeměpisných souřadnic, je možné ho přiřadit do volebního okrsku.

Níže je uveden seznam všech druhů údajů o každém voliči společně s možnými hodnotami:

Zeměpisná šířka
Zeměpisná délka
Věk
Pohlaví (muž, žena)
Vzdělání (základní, střední bez maturity, střední s maturitou, vyšší odborné, univerzitní)
Stav (rozvedený/á, ženatý/vdaná, svobodný/á, ovdovělý/á)
Ekonomická aktivita (neaktivní, student, pracující)
Počet osob žijících v domácnosti
Počet dětí žijících v domácnosti

3.2 Územní rozdělení (geografie)

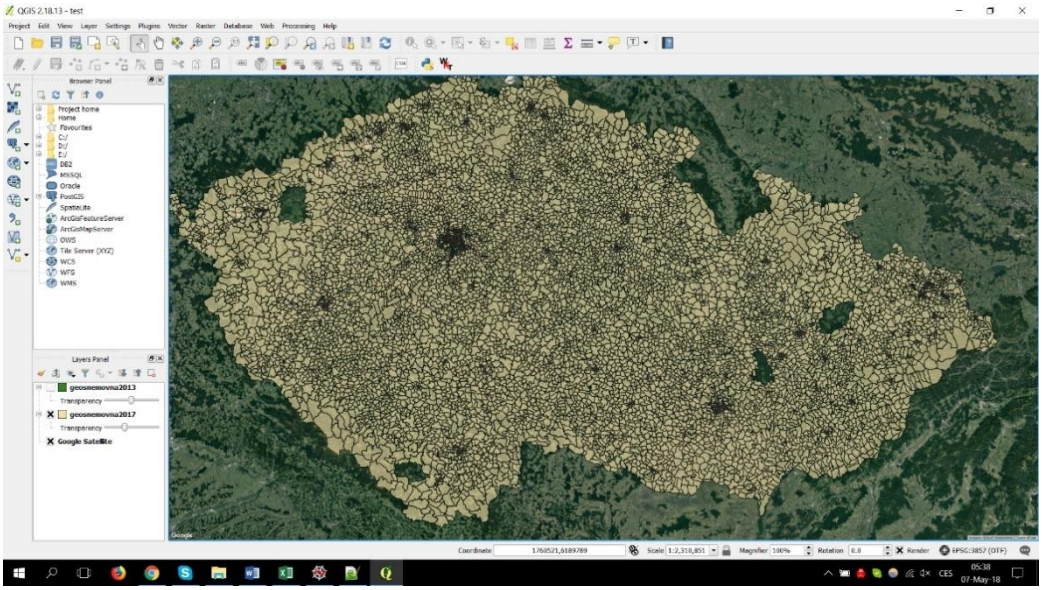
Mezikrokem ve tvorbě trénovacích příkladů je práce s geografií volebních okrsků. Volební okrsky totiž tvoří logické spojení mezi obyvateli a výsledky za okrsky. Tyto údaje získáme z otevřených zdrojů, konkrétně z webových stránek Českého úřadu zeměměřičského a katastrálního (services.cuzk.cz/shp/obec/epsg-5514). Jsou ve formátu Shapefile - trojice souborů s příponami shp, shx a dbf společně se souborem prj s údaji o použité mapové projekci. V tomto případě se jedná o projekci číslo 5514 J-JTSK/ Krovak East North, která je závazným geodetickým referečním systémem na území České republiky [18].

Data jsou rozdělena do zip archivů po obcích – shp soubor v každé složce pak obsahuje polygony jednoho až tisíců volebních okrsků v závislosti na velikosti obce. Ke každému tomuto

polygon soubor dodatečně nese kód obce, městské části/obvodu a kód okrsku (označované jako OBEC_KOD, MOMC_KOD, CISLO), které se shodují s kódy volebních dat.

Výměry volebních okrsků se v průběhu času pochopitelně do jisté míry mohou měnit, tyto změny pro potřeby práce zanedbáme. Posbíraná data pochází přibližně z doby voleb 2017 a mají pro naše účely více než dostatečnou přesnost. Je možné je jistě použít i pro volby předchozí, protože nedošlo k velkým změnám (největší změnou je rušení vojenských prostorů, ve kterých ale lidé nežijí), které by nebylo možné zanedbat.

Postupně všechny tyto soubory otevřeme a polygony v něm obsažené uložíme do relační databáze s geografickým rozšířením (PostgreSQL/PostGIS), kde tvoří jeden sloupec tabulky. Další sloupce pak tvoří jejich identifikační kódy. To provedeme skripty popsanými v kapitole 4.2.



Obr. 3:1 - PostgreSQL tabulka s geometriemi okrsků zobrazená v programu QGIS

3.3 Volební výsledky

Data o volebních výsledcích na úrovni okrsků z voleb do Poslanecké sněmovny roku 2017 jsme získali z webových stránek Českého statistického úřadu. Posbírali jsme výsledky za celkem 11 stran, které získali více než jedno procento hlasů v celorepublikových výsledcích. Tyto strany jsou: ODS, ČSSD, STAN, KSČM, Strana zelených, Strana svobodných občanů, Česká pirátská strana, TOP09, Ano, KDU-ČSL a SPD. Hlasy ostatních stran jsou sloučeny do jedné položky.

Na území České republiky bylo v době těchto voleb celkem 14750 okrsků. V relevantních krajích (Praha a Středočeský a Jihomoravský kraj) jich bylo celkem 4683.

3.4 Trénovací příklady

Z posbíraných dat je potřeba sestavit množinu příkladů pro trénink a testování modelu. To znamená vytvořit páry vstupních vektorů a vektorů k nim přidružených výsledků. Na základě údajů o obyvatelích jsme pro všechny okrsky, pro které máme data o alespoň jednom obyvateli, spočítali následující statistiky (znak % označuje procentuální část obyvatel okrsku pro které platí následující).

1	Počet obyvatel	**16**	% ženatých/vdaných
2	Plocha v km^2	**17**	% svobodných
3	Hustota zalidnění	**18**	% ovdovělých
4	Průměrný věk	**19**	% ekonomicky neaktivních
5	% pod 18 let	**20**	% studentů
6	% 18 až 35 let	**21**	% pracujících
7	% 35 až 55 let	**22**	% pracujících nebo studentů
8	% nad 55 let	**23**	Průměrný počet lidí žijících v domácnosti
9	Poměr pohlaví	**24**	% žijících samy
10	% se základním vzděláním	**25**	% žijících ve dvou
11	% se středním vzděláním bez maturity	**26**	% žijících ve více než dvou lidech
12	% se středním vzděláním s maturitou	**27**	Průměrný počet dětí osoby
13	% s vyšším odborným vzděláním	**28**	% bezdětných
14	% s univerzitním vzděláním	**29**	% s jedním nebo dvěma dětmi
15	% rozvedených	**30**	% s více než dvěma dětmi

Tab. 3:1 - Tabulka kategorií statistik

Těchto třicet údajů tvoří vstupní vektor do naší sítě. Pro názornost uvedeme příklad vektoru pro konkrétní okrsek. Pro okrsek číslo 24, kód obce 554782 a kód městské části nebo obvodu 539694 – jedná se o okrsek v Praze 13, Nových Butovicích:

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
437	0,11	3878,71	35,59	0,18	0,30	0,37	0,15
9	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1,01	0,12	0,17	0,33	0,01	0,24	0,47	0,37
17	**18**	**19**	**20**	**21**	**22**	**23**	**24**
0,13	0,03	0,25	0,22	0,53	0,75	3,21	0,07
25	**26**	**27**	**28**	**29**	**30**		
0,21	0,72	1,04	0,47	0,43	0,10		

Tab. 3:2 - Tabulka s příkladem hodnot vstupního vektoru

Takových okrsků máme celkem 4555. Do množiny jsme ale zařadili pouze ty okrsky, ve kterých máme údaje o více než 50 obyvatelích, aby údaje byli reprezentativní. Celková velikost množiny je potom 3971 příkladů.

Těchto 3971 příkladů normalizujeme pro lepší učení. Normalizujeme tak, aby sloupce matice příkladů (3971 řádků, 30 sloupců) měly jednotkový rozptyl a nulovou střední hodnotu.

Vektor volebních výsledků pak obsahuje údaje o výsledcích stran jako poměr počtu hlasů pro stranu vůči celkovému počtu hlasů v daném okrsku. Takový vektor, pro stejný volební okrsek jako výše, vypadá následovně:

Zbylé	**ODS**	**ČSSD**	**STAN**	**KSČM**	**Zelení**
0,0510	0,1438	0,0616	0,0496	0,0342	0,0222
Svobodní	**Piráti**	**TOP09**	**ANO**	**SPD**	
0,0393	0,1695	0,1113	0,1849	0,0821	

Tab. 3:3 - Tabulka s příkladem hodnot výstupního vektoru

4 Implementace

4.1 Sběr dat (Scrap)

Pro potřeby práce byl vytvořen skript pro sběr dat o volbách do Poslanecké sněmovny v roce 2017 z webové stránky „http://www.volby.cz“ Českého statistického úřadu. Využita je podstránka s výsledky hlasování za územní celky, která se dále větví na podstránky okresů, obcí a nakonec jejich okrsků. Skript je vytvořen v jazyce Python s využitím knihovny Beautiful Soup 4. Je upraven tak, aby ho bylo možné použít i pro volby v roce 2013.

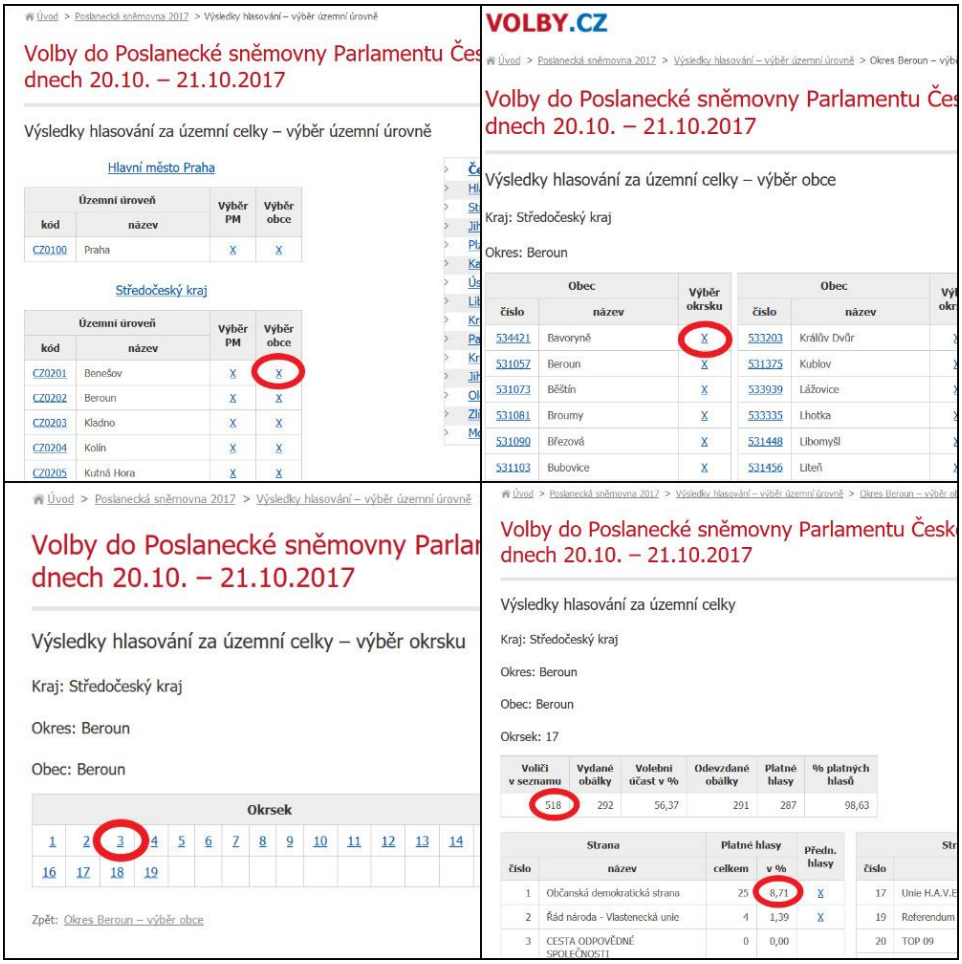
Skript tedy v každé úrovni vybere všechny relevantní odkazy a postupně je otevře. Na poslední stránce – stránce okrsku – zaznamená kódy obce, městské části či obvodu a okrsku, dále statistiky o volbě a výsledky za jednotlivé strany v absolutním počtu a v procentech. Tyto údaje uloží ve formátu JSON pro další použití.

Protože stránka volby.cz obsahuje chybu v označení obce ve statutárních městech, je využito číselníku kódů městských částí/obvodů k jejich zjištění.

Zde je příklad jednoho takového JSON souboru pro ukázku formátování:


```
{"properties": {
        "xobec": 554821,
        "xokrsek": 1,
        "xmc": 546135,
        "statistics": {
                "VOLICI": 5801,
                "VYDANE": 696,
                "ODEVZDANE": 696,
                "PLATNE": 694,
                "UCAST": "12.00",
                "PLATNYCH": "99.71"
        },
        "result": {
                "OTHER": [32, "4.57"],
                "ODS": [55, "7.92"],
                "CSSD": [45, "6.48"],
                "STAN": [10, "1.44"],
                "KSCM": [66, "9.51"],
                "ZELENI": [10, "1.44"],
                "SVOBODNI": [8, "1.15"],
                "PIRATI": [76, "10.95"],
                "TOP09": [18, "2.59"],
                "ANO": [208, "29.97"],
                "KDU-CSL": [12, "1.72"],
                "SPD": [154, "22.19"]
        }
}}
```

Následující čtyři obrázky zobrazují rozvržení podstránek s vyznačenými relevantními odkazy a informacemi. Postupuje se od krajů České republiky, přes stránku s výčtem obcí, následně stránku s výčtem okrsků až k výsledkům v okrsku. Ty jsou zaznamenány.



Obr. 4:1 - Výsledky hlasování za územní celky - Česká republika, Okres, Obec, Okrsek

4.2 Databáze

Další částí je zpracování všech posbíraných dat. To znamená rozřazení všech obyvatel do příslušných okrsků na základě jejich souřadnic a spočítání statistik pro každý tento okrsek. K tomuto účelu byla využita PostgreSQL/PostGIS databáze. Data byla do databáze importována Python skripty pomocí knihovny psycopg2. Zdrojové kódy jsou uvedeny v příloze ve složce import.

Byly vytvořeny následující tabulky:

geo – Tabulka obsahuje geometrie volebního okrsku jako multipolygon spolu s jeho identifikačními kódy. Byla vytvořena skriptem „import/import_shapefiles.py“ uvedeným v příloze.

demo_brno, demo_prague – Tabulky demografických údajů o jednotlivých obyvatelích s identifikačními údaji okrsku. Vytvořeny skriptem „import/import_demograph.py“.

stat_brno, stat_prague – Tabulky s demografickými statistikami pro jednotlivé okrsky.Vytvořeny skriptem „import/import_stSatistics.py“.

snemovna2017 – Tabulka s volebními údaji pro jednotlivé okrsky. Vytvořena skriptem „import/import_election_data.py“ ze JSON souborů s výsledky.

model_cat, model_mse – Tabulky dvou nejlepších modelů s výsledky na celé datové množině. Podrobněji vysvětleno v kapitole 5.

4.3 Neuronové sítě

Pro učení neuronových sítí využíváme knihovnu Keras [2] s backendem TensorFlow [3].

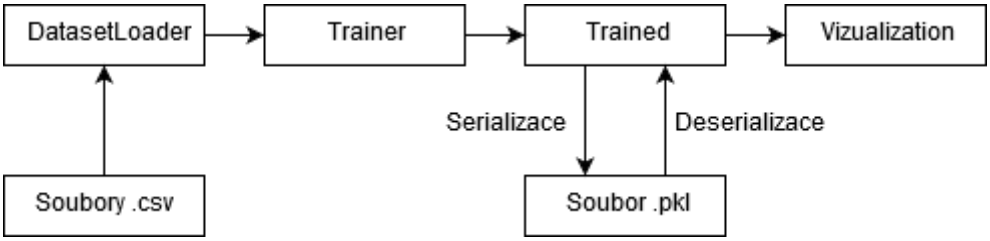
Pro načítání dat z csv souborů používáme třídu DatasetLoader. Pro usnadnění testování různých konfigurací neuronových sítí byly vytvořeny třídy Trainer, Trained a Visualization.

Třída Trainer obsahuje metody sloužící k samotnému trénování jí poskytnutého modelu. Přesněji, jeho instance si sama vytváří podle poskytnutého předpisu. Ten je předán jako funkce, která vrací novou instanci. Trénovací metody této třídy vrací objekt třídy Trained.

Ten obsahuje natrénovaný model, informace o něm a také informace o průběhu učení, především vývoj ztrátové funkce na trénovací i validační sadě. Obsahuje také metody pro serializaci a deserializaci do souboru.

Třída Visualization využívá knihovny Matplotlib [19] ke kreslení grafů z natrénovaných modelů. Přijímá přímo objekty třídy Trained a zobrazuje průběhy učení. Grafy v této práci jsou generovány právě pomocí metod této třídy.

Dá se tedy říci, že tyto třídy budou obvykle použity postupně, tak jak jsem je uvedl. Pro znázornění přidávám grafiku tohoto procesu. Celý tento proces je zvnějšku ovládán skripty, například skriptem mcrossval.py, který provede učení s cross-validací nad jedním poskytnutým modelem. Je možné volat ho opakovaně s různými modely, jak to dělá skript mass_crossval.py.



Obr. 4:2 - Schéma průběhu

Navíc jsou přidány skripty:

training_full.py – Provede finální natrénování vybraných modelů na celé množině dat.

graph_models.py – Demonstruje použití vizualizačních metod. Pomocí něho byly generovány grafy v této práci.

normalize_dataset.py – Použit k normalizaci datové množiny.

Kódy učení sítí jsou uvedeny ve složce „keras“ v příloze.

4.4 Webová aplikace

Pro prezentační účely naučených modelů byla vytvořena webová aplikace. Ta bude plnit dvě různé úlohy. Zaprvé umožní zobrazení výsledků voleb pro jeden okrsek či libovolný výběr více okrsků (např. část města či okresu). Zadruhé umožní namodelování vlastního okrsku zvolením příslušných demografických dat a následné zobrazení predikce volebního výsledku pro zadaná data.

Oba tyto úkoly by měly být splněny v uživatelsky přívětivé podobě. To znamená vytvoření grafického uživatelského prostředí. Přirozeně nejlepším způsobem zobrazení okrsků je jejich vykreslení do mapy České republiky. Pro lepší orientaci v mapě budou vykresleny pouze jejich hranice. Mapa by měla umožnit výběr okrsku kliknutím i také nějakou formu hromadného výběru. Jako velmi flexibilní se jeví výběr okrsků kreslením libovolného polygonu do mapy.

Pro splnění druhého úkolu, tedy vytvoření vlastních demografických hodnot okrsku nestačí pouze množství vstupních polí. Některé jejich hodnoty jsou spolu totiž provázané a tyto logické vazby je třeba dodržet, ideálně je i vhodně graficky reprezentovat.

Kódy jsou uvedeny v příloze ve složce „web-react“.

4.4.1 Technologie

Ačkoliv se může zdát, že se jedná o malou úlohu, bude potřeba značně široká paleta nástrojů. Použijeme dvě sady nástrojů (tzv. solution stack), jednu optimalizovanou na vývoj a druhou produkční. Vývojová sada bude lokální, produkční použijeme pro verzi přístupnou na webu.

Je třeba také vyřešit otázku hostingu. Rozhodli jsme se využít Google Cloud Platform jako IaaS (Infrastructure as a Service). Tam si pro účely našeho produkčního serveru vytvoříme virtuální stroj s operačním systémem Linux Ubuntu. V poznámce na konci této práce je uvedena IP adresa, na které aplikace poběží.

Z technického pohledu bude potřeba zajistit tři odlišné požadavky:

1) – Poskytování samotné webové stránky a databáze volebních výsledků

2) – Poskytování geografických údajů tj. hranic okrsků a dodatečných služeb pro výběr okrsků

3) – Poskytování předpovědí modelů neuronových sítí pro demografické údaje zadané uživatelem.

Pro samotnou stránku použijeme obvyklý server Apache, doplněný již zmiňovanou PostgreSQL databází a PHP skripty. Aplikace bude vytvořena v javascriptovém UI frameworku React (jako tzv. Single Page Application), poskytována bude staticky (samotný web se nebude měnit v závislosti na kontextu a požadavcích klienta) a renderování bude také na straně klienta. To nevyužívá plné možnosti tohoto frameworku, ale pro naše účely to postačí. Pro tvorbu usnadnění tvorby uživatelského rozhraní je vhodné využít jednu z řady knihoven dostupných pro React, abychom nemuseli vytvářet grafiku vlastnoručně pomocí CSS. V našem případě využijeme především knihovnu Semantic UI v jejím React portu.

Využit bude Javascript nejméně ve verzi ES6 (ECMAScript 2015), to proto, že pro React je vhodné použití syntaxe tříd a tzv. Fat Arrow funkcí zachovávajících kontext a také tzv. Spread operátoru.

Demografické údaje z PostgreSQL databáze budeme poskytovat PHP skripty. Ty budou poskytovat jakési API pro přístup. To není zcela moderní řešení takového úkolu, ale je funkční, rychlé a snadné na vývoj.

Pro geografickou část naší aplikace použijeme servlet kontejner Apache Tomcat, do kterého nainstalujeme Geoserver jako servlet. Ten si bude data sbírat z PostgreSQL databáze. Naše aplikace se pak na Geoserver bude dotazovat pomocí tzv. Web Feature Service (WFS), standardu žádostí o geografická data založené na XML formátu.

Bylo by samozřejmě možné data poskytovat přímo Apache serverem například přes již zmíněné PHP rozhraní, protože geografické rozšíření PostGIS (jako součást PostgreSQL databáze) obsahuje všechny potřebné metody pro výběr všech okrsků v zadaném polygonu.

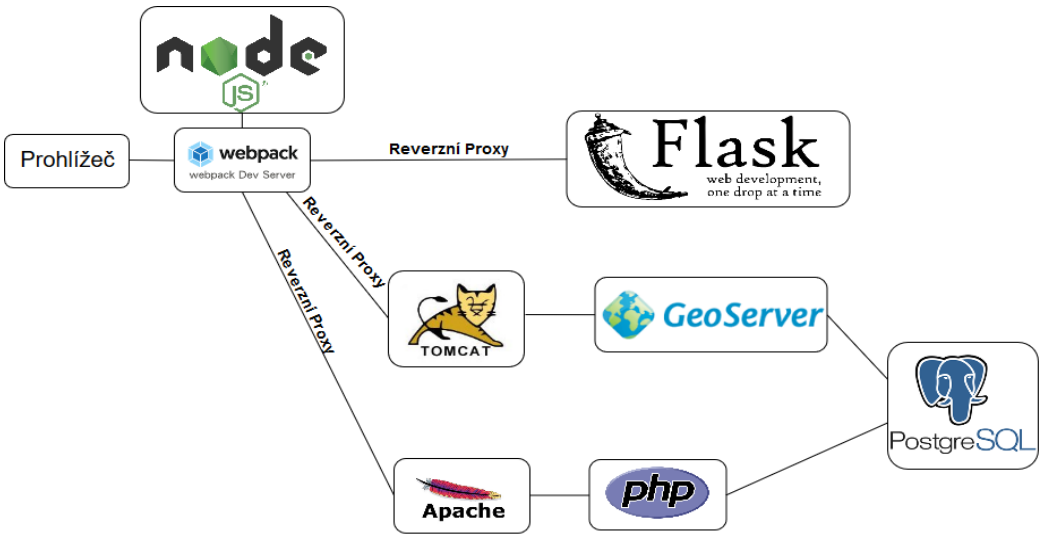
Geoserver je ale pro tyto úkoly optimalizovaný (např. si data ukládá do cache) a výkon námi napsaného rozhraní by byl téměř s jistotou v porovnání horší. Kromě toho není nastavení Geoserveru nijak obzvlášť obtížné.

Část této práce týkající se neuronových sítí ale máme vytvořenou v jazyce Python, konkrétně v knihovnách Keras s Tensorflow backendem. Rozhodli jsme se využít Python frameworku Flask pro vytvoření API k poskytování metod práce s modelem. Python aplikace ve Flasku si tedy při spuštění načte Keras a natrénované modely, které bude posléze používat k předpovědím.

Vývoj

Pro vývoj zvolíme trochu jiný postup. Pro zjednodušení nastavení React aplikace využijeme manažer balíčků (package manager) Yarn. Jeho prostřednictvím nainstalujeme Create-React-App, nástroj pro rychlé sestavení všeho potřebného pro React projekt. Create-React-App využívá NodeJS a Webpack-Dev-Server. To umožňuje automaticky sestavit projekt a zobrazit ho v prohlížeči po každém uložení kódu, což je velice pohodlné a především to vývoj významně urychlí.

Na následujícím obrázku vidíme celé vývojové řešení.



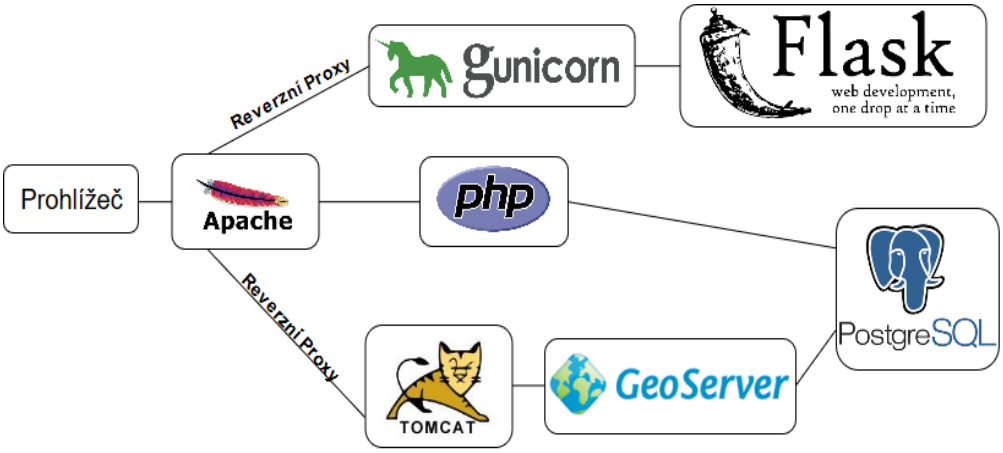
Obr. 4:3 – Vývojový stack

Produkce

Pro vývoj nám postačí server, který je poskytován přímo jako součást Flask frameworku. Ten ale není vhodný pro případnou vyšší zátěž. Pro produkci proto použijeme WSGI server Gunicorn (ten se obvykle s Flaskem kombinuje), který se postará o vrstvu mezi http požadavky a logikou ve Flasku. Požadavky na něj pak budeme přesměrovávat z Apache serveru pomocí reverzní proxy.

V produkci nám také odpadne potřeba používat Webpack-Dev-Server a s ním i NodeJS. React aplikace bude poskytována přímo prostřednictvím Apache, takže se složitost zapojení o něco sníží. Apache poběží na portu 80 a přes proxy bude předávat požadavky na porty 8000 pro Gunicorn a port 8080 pro Tomcat.

Prohlédněme si produkční řešení:



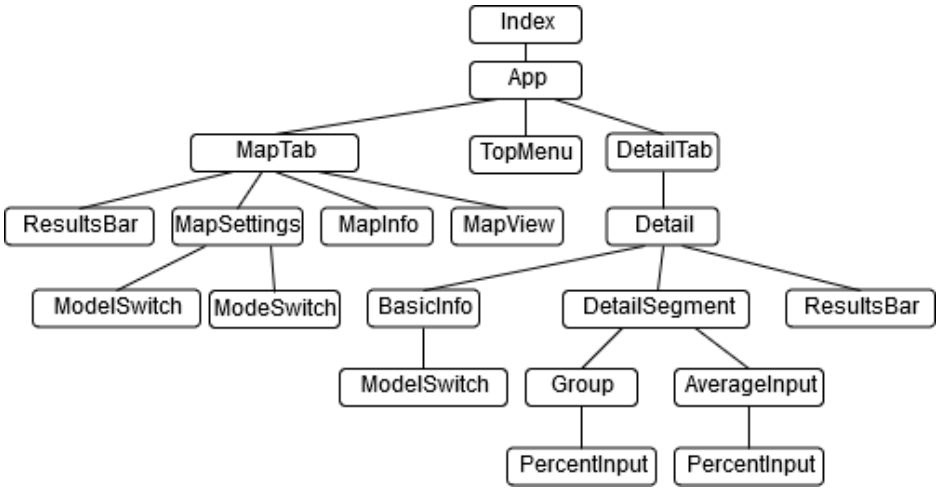
Obr. 4:4 – Produkční Stack

4.4.2 Klient

Klientská část je vytvořena v jazyce Javascript. V již zmíněném Reactu byl pro grafické rozhraní použit především framework Semantic UI, částečně také Reactstrap. Kromě toho také React Router pro URL navigaci a Nivo pro vykreslování sloupcových grafů s volebními výsledky.

Mimo React byla použita Javascriptová knihovna OpenLayers4 k vykreslení mapy okrsků. Právě ta komunikuje prostřednictvím zmíněné Web Feature Service s Geoserverem. Použitím moderního Fetch API pro AJAX požadavky jsme se vyhnuli knihovně jQuery.

Schéma React komponent



Obr. 4:5 - Schéma React komponent

Aplikace je logicky rozčleněna na dvě části. Stránku s mapou okrsků (MapTab) a stránku s detailem okrsku (DetailTab). Mezi nimi je možné přepínat v Menu (TopMenu) umístěném v listě v horní části.

O správu mapy vytvořené v OpenLayers se primárně stará dvojice objektů Geoloader a Interaction. Ty jsou napojeny v React komponentách MapTab (Dataloader) a MapView (Geoloader a Interaction)

Interaction

Objekt Interaction zpracovává interakci uživatele s vektorovou vrstvou mapy, sloužící k výběru okrsků. Ta probíhá dvěma možnými způsoby: kliknutím na požadované okrsky pro jejich přidání (nebo odebrání) anebo postupným naklikáním bodů vymezujících rohy mnohoúhelníku pro přidání všech okrsků v něm obsažených. Tyto dvě možnosti jsou označeny jako Click a Polygon.

Geoloader

Objekt Loader slouží k načítání vektorových dat do mapy pomocí služby Web Feature Service (WFS) poskytované GeoServerem. Má dvě funkce. Zaprvé načítá data o okrscích pro aktuálně zobrazenou oblast mapy. To znamená, že při posunutí mapy pošle nový požadavek na získání okrsků. Zadruhé získává identifikace všech okrsků v uživatelem nakresleném polygonu.

GeoServer získává údaje o geometriích okrsků z PostgreSQL/PostGIS databáze z view „publish“, které obsahuje data o okrscích s populací alespoň padesáti lidí. Aplikace je pak poskytuje pomocí služby Web Feature Service (WFS) ve verzi 1.0.0, která na dotaz aplikace

(z modulu Geoloader) odešle geometrie zakódované společně s identifikačními údaji okrsku ve formátu GML3. K tomu je využit WFS dotaz GetFeature

Protože aplikace umožňuje nakreslení libovolného polygonu do mapy a přidání všech okrsků v něm zcela obsažených do aktuálního výběru, je potřeba mít možnost filtrování pomocí vektorových dat. Geometrie polygonu je vepsána do URL dotazu podle specifikace OGC (Open Geospatial Consortium) Filter Encoding [20] [21]. Konkrétně je použita funkce Within. GeoServer pak v odpovědi vrátí dohodnutá ID čísla okrsků, které se zcela nachází v této oblasti.

Příklad takového požadavku:

```
http://IP_ADRESA/geoserver/wfs?service=WFS&version=1.0.0&
request=GetFeature&typeName=okrsek:publish&outputFormat=gml3&
srsName=EPSG:3857&propertyName=id&filter=
<Filter xmlns:gml="http://www.opengis.net/gml">
<Within>
<PropertyName>geom</PropertyName>
<gml:Polygon gml:id="polygon" srsName="EPSG:3857" srsDimension="2">
        <gml:outerBoundaryIs>
                <gml:LinearRing>
                        <gml:coordinates>49.15     15.12     49.14     15.12     49.15
15.13</gml:coordinates>
                </gml:LinearRing>
        </gml:outerBoundaryIs>
</gml:Polygon>
</Within>
</Filter>
```

Dataloader

Objekt slouží k načítání volebních dat pro zvolené okrsky. Obsahuje metody k provádění dotazů na PHP skripty vracející údaje o požadovaných okrscích. Data oběma směry jsou předávána ve formátu JSON. Jsou odeslány kódy obce, městské části a okrsku pro přidané/odebrané okrsky, PHP skript podle nich vyhledá výsledky v databázi a odešle.

4.4.3 API

Jak již bylo popsáno výše, náš web používá tři API (dvě z nich jsou námi napsané v jazycích PHP a Python). Vstupy i výstupy jsou ve formátu JSON.

Python API

URL	HTTP	Vstup	Výstup
/model/cat/	GET		popis modelu cat
/model/mse/	GET		popis modelu mse
/model/	POST	{'modelID':'', 'vector':''}	{'prediction':'', 'parties':''}
Přijímá typ modelu a vektor demografických údajů, vrací pole stran a vektor volební předpovědi			

Tab. 4:1 - Python API

PHP API

URL	HTTP	Vstup	Výstup
/api/result.php	POST	{ID}	Volební výsledky pro okrsky
Přijímá JSON soubor s identifikačními údaji okrsků a vrací jim příslušné skutečné volební výsledky			
/api/prediction.php	POST	Typ modelu a ID okrsků	Předpovědi modelu pro okrsky
Přijímá JSON soubor s typem modelu a identifikačními údaji okrsků a vrací jim příslušné předpovědi tohoto modelu			

Tab. 4:2 - PHP API

5 Porovnání modelů

V této části porovnáme několik konfigurací neuronových sítí.

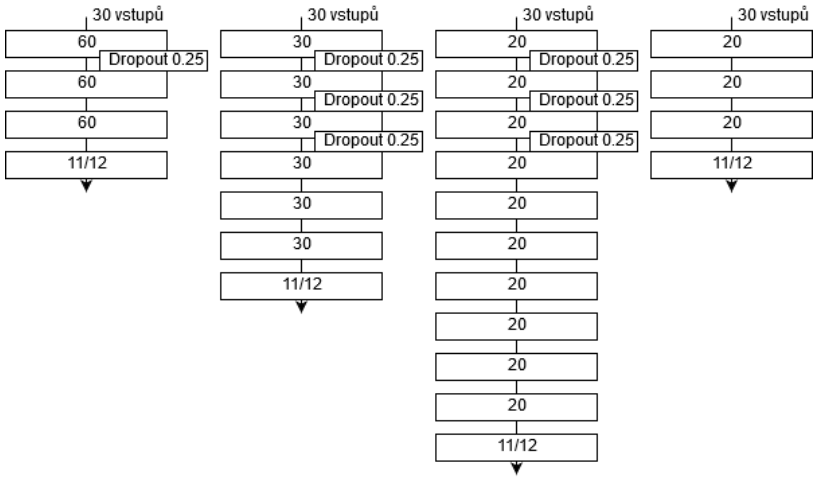
Struktura

Navrhli jsme několik různých struktur neuronové sítě (lišící se počtem vrstev a neuronů), se kterými jsme experimentovali. Vyzkoušeli jsme dvě různé ztrátové funkce.

Pro ztrátovou funkci s Mean Squared Error bude mít výstupní vrstva jedenáct neuronů, jednu stranu vynecháme. MSE nezaručuje, že součet výsledného vektoru pravděpodobností bude do jedničky. Proto je praktické vyřadit jednu politickou stranu z vektoru a výsledný součet pak dopočítat. Přirozeně se nabízí použít stranu s nejlepším volebním výsledkem. Nejméně totiž hrozí, že se předpověď netrefí a součet už jedenácti stran bude vyšší než jedna.Protože chceme měřit i hodnotu MAE, budeme ji z tohoto důvodu muset počítat sami odděleně a nespoléhat se na implementaci v Keras. Jako aktivační funkci výstupní funkci použijeme lineární funkci, tedy pouze vrátíme vstup této vrstvy.

Kromě MSE vyzkoušíme ztrátovou funkci Cross-entropy. V Keras je naimplementována pod názvem Categorical Crossentropy. U ní je naopak vhodné použít vektor všech dvanácti stran. To proto, že výsledný vektor se bude vždy sčítat do jedné. Jako funkci výstupní vrstvy použijeme funkci Softmax.

Vyzkoušeli jsme tři aktivační funkce skrytých vrstev sítě: ReLU, LeakyReLU a ELU. Rovněž jsme experimentovali s použitím metody Dropout. Navrženy byly čtyři různé struktury neuronové sítě.



Obr. 5:1 - Struktury neuronové sítě popořadě 01,02,03 a 04

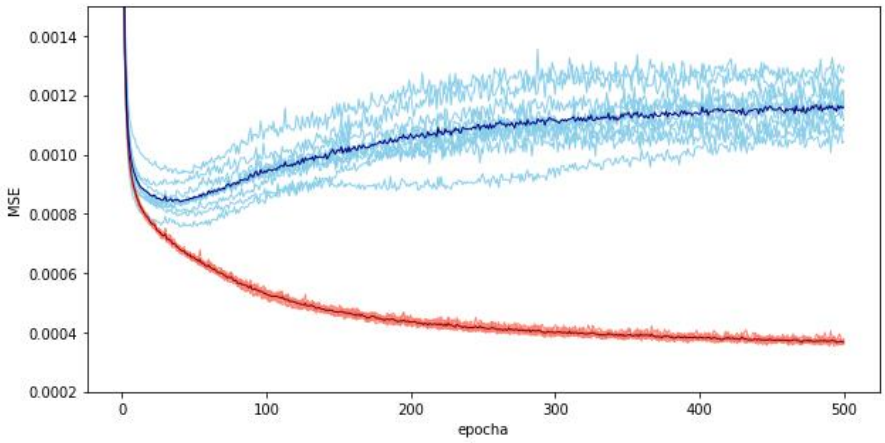
Struktury 01,02 a 03 jsme testovali se třemi aktivačními funkcemi skrytých vrstev a se zapojením Dropout a bez něj. To je 18 konfigurací pro každou ztrátovou funkci. K tomu jsme testovali strukturu 04. Tu pouze bez Dropoutu, tedy 6 konfigurací. Celkem jsme tedy vyzkoušeli 42 konfigurací. Všechny jsme trénovali po 500 epoch a po každé epoše změřili výsledky na trénovací a validační sadě. Ty, u kterých bylo minimum na validační sadě dosaženo v posledních z těchto 500 epoch, jsme trénovali dalších 500 epoch.

Bylo potřeba zvolit koeficienty pro aktivační funkce LeakyReLU a ELU. Zvolili jsme běžně používané hodnoty 0,1 pro LeakyReLU a 1 pro ELU. Více jsme s nimi neexperimentovali. Podobně velikost dávky (batch size) jsme zvolili 64.

Hledání nejlepší konfigurace

Pro porovnání modelů použijeme tzv. 10-fold cross-validaci. Příklady nejprve zamícháme a poté desetkrát rozdělíme na trénovací (90%) a validační část (10%). Trénovat budeme 500 epoch, případně více, pokud to bude nezbytné. Po každé epoše vyhodnotíme model zvlášť na trénovací a validační sadě. Zajímá nás nejnižší dosažená hodnota ztrátové funkce na validačních datech, tedy průměr ze všech deseti instancí modelu v dané epoše.

V následujícím obrázku je zobrazen takový vývoj (pro model02, bez Dropoutu) hodnoty ztrátové funkce MSE. Modrou barvou jsou vyznačeny výsledky na validační sadě a červenou na testovací. Tmavá barva je průměr zmíněných deseti hodnot (ty jsou vyznačeny světle).

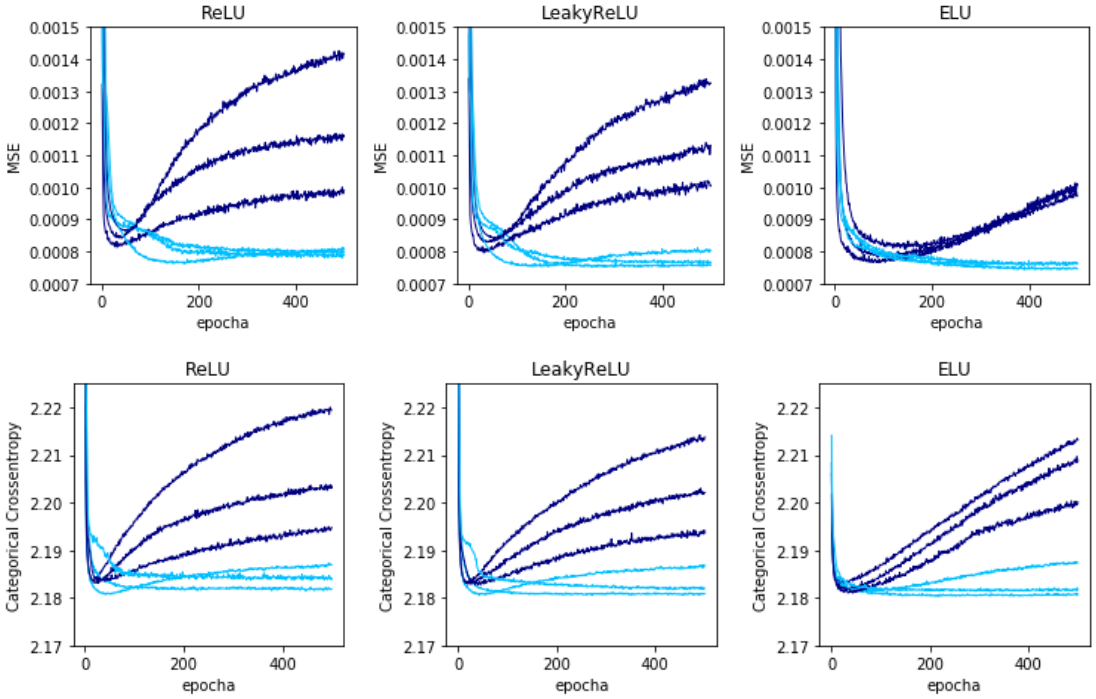


Obr. 5:2 - Vývoj ztrátové funkce na trénovacích a validačních datech

Zde vidíme, že po určitém množství epoch se kvalita modelu na validačních datech již začíná zhoršovat – patrně dochází k přeučení. To je ostatně logika za naším rozhodnutím použít Dropout. Prozkoumáme tedy jeho vliv. Podíváme se na to, jak pro každou aktivační funkci ovlivňuje zapojení Dropoutu vývoj této křivky.

Na následujícím obrázku vidíme vývoj ztrátové funkce na validačních datech (průměr chyby z 10-fold cross-validace). Každá křivka reprezentuje vývoj pro danou aktivační funkci na

jedné ze struktur 01,02 a 03. Modely s využitím Dropoutu jsou obarveny světle modrou barvou. To v porovnání s modely bez jeho zapojení – ty budou tmavě modrou. Vidíme, že Dropout pomáhá předejít přeučení a v našem případě pomáhá dosáhnout nižší hodnoty ztrátové funkce. Ve všech případech jsme s ním dosáhli lepších výsledků.



Obr. 5:3 - Porovnání výsledků pro modely 01,02,03 s a bez zapojení Dropoutu

Tyto modely se při velkém počtu skrytých neuronů (a vrstev) nepřeučují pouze v případě, že je zapojen dropout. To může znamenat, že tyto modely jsou pro náš problém příliš komplexní nebo že k němu nemáme dostatečné množství příkladů.

K výraznému přeučování nedochází pouze u nejmenšího modelu (04). To může být způsobené právě tím, že jeho rozlišovací schopnost je srovnatelná s komplexitou problému.

5.1 Výsledky

Podívejme se nyní na dosažená minima ztrátové funkce pro všech 42 konfigurací. Zajímá nás samotná hodnota a také to, po jaké epoše k ní došlo. V tabulce je zvýrazněna konfigurace s nejnižší hodnotou ztrátové funkce pro obě tyto funkce.

Strukt.	Drop.	Act	Epocha	MSE	RMSE	Strukt.	Drop.	Act	Epocha	CAT
mse01	ne	RELU	52	0.00086353	0.02939	cat01	ne	RELU	15	2.183677
mse01	ne	Leaky	61	0.00084071	0.02899	cat01	ne	Leaky	23	2.183269
mse01	ne	ELU	145	0.00080705	0.02841	cat01	ne	ELU	21	2.183071
mse01	ano	RELU	147	0.00076344	0.02763	cat01	ano	RELU	42	2.180682
mse01	ano	Leaky	165	0.00075336	0.02745	cat01	ano	Leaky	55	2.180677
mse01	ano	ELU	429	0.00075649	0.02750	cat01	ano	ELU	71	2.181258
mse02	ne	RELU	35	0.00083903	0.02897	cat02	ne	RELU	25	2.183031
mse02	ne	Leaky	44	0.00082432	0.02871	cat02	ne	Leaky	24	2.182935
mse02	ne	ELU	88	0.00078121	0.02795	cat02	ne	ELU	42	2.181795
mse02	ano	RELU	394	0.00078320	0.02799	cat02	ano	RELU	394	2.181579
mse02	ano	Leaky	265	0.00075217	0.02743	cat02	ano	Leaky	209	2.180571
mse02	ano	ELU	390	0.00074386	0.02727	cat02	ano	ELU	217	2.180268
mse03	ne	RELU	36	0.00081554	0.02856	cat03	ne	RELU	35	2.18346
mse03	ne	Leaky	32	0.00079129	0.02813	cat03	ne	Leaky	46	2.182503
mse03	ne	ELU	93	0.00076526	0.02766	cat03	ne	ELU	44	2.180981
mse03	ano	RELU	418	0.00078130	0.02795	cat03	ano	RELU	340	2.183565
mse03	ano	Leaky	418	0.00076268	0.02762	cat03	ano	Leaky	447	2.181729
mse03	ano	ELU	458	0.00075825	0.02754	cat03	ano	ELU	293	2.18136
mse04	ne	RELU	175	0.00081218	0.02850	cat04	ne	RELU	51	2.182509
mse04	ne	Leaky	194	0.00078774	0.02807	cat04	ne	Leaky	51	2.182172
mse04	ne	ELU	266	0.00076796	0.02771	cat04	ne	ELU	87	2.181368

Tab. 5:1 - Nejlepší výsledky všech konfigurací modelů

Máme tedy dva výsledné modely. Ty natrénujeme nyní již na celé množině dat, poté vygenerujeme jejich „předpověď“ (na trénovací množině dat) a nahrajeme do databáze pro zobrazení ve webové aplikaci. Stejně tak do složky s python API uložíme samotné modely, aby je bylo možné z webu použít k dalším předpovědím.

V průběhu tréninku jsme navíc měřili i hodnotu Mean Absolute Error, aby existovala alespoň velmi přibližná možnost srovnání mezi oběma funkcemi. Ta je mírně lepší pro ztrátovou funkci Cross Entropy. V kontextu vektoru volebních výsledků jde MAE vlastně chápat jako procentuální vyjádření podílu hlasů, které byli v daném okrsku přiřazeny nesprávné straně v průměru pro jednu stranu. To znamená, že celkový procentuální podíl nesprávně přiřazených hlasů se rovná hodnotě MAE (násobené stem) násobené počtem stran. Pro námi naměřenou hodnotu 0,02 to znamená 24 % špatně přiřazených hlasů.

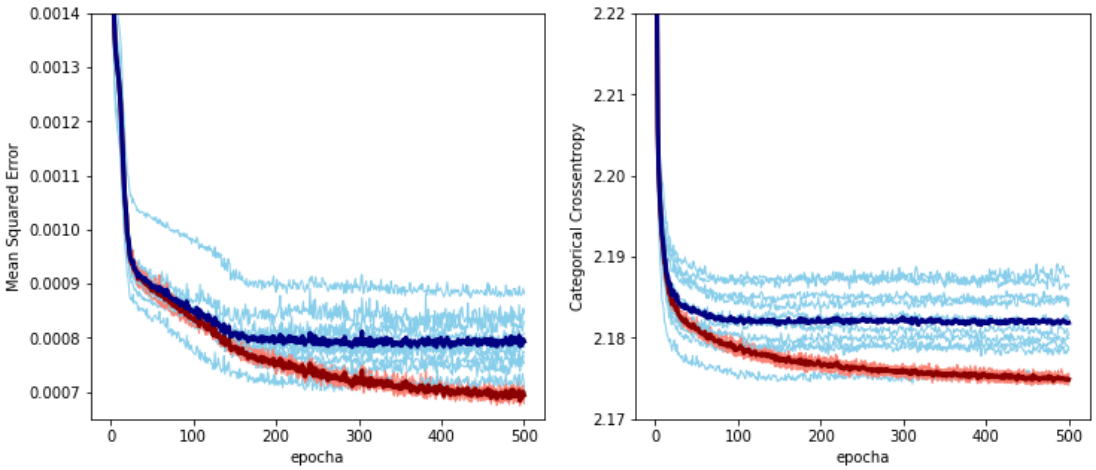
Také stojí za to srovnat výsledky se střední hodnotou volebního výsledku v naší množině dat a také skutečným celorepublikovým výsledkem voleb, abychom věděli, jestli se náš model

vůbec dokáže smysluplně učit oproti situaci, kdybychom jako předpověď vždy použili stejný takový vektor. Vidíme, že náš model se skutečně učit dokáže.

Struktura	Dropout	Aktivace	Epocha	Ztrát. Funkce	MAE
mse02	Ano	ELU	390	0.000744	0.02003
cat02	Ano	ELU	217	2.18027	0.01986
Střední hodnota volebního výsledku datové sady					0.02830
Skutečný celorepublikový výsledek voleb					0.02937

Tab. 5:2 - Porovnání nejlepších konfigurací

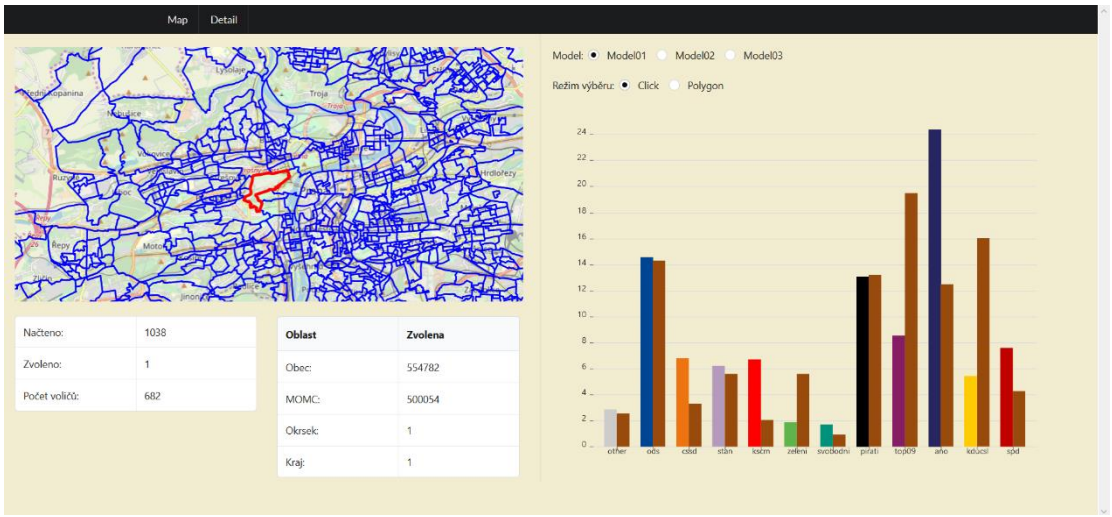
Zde vidíme průběh cross-validace těchto dvou nejlepších modelů.



Obr. 5:4 - Vývoj hodnoty ztrátové funkce pro nejlepší modely při cross-validaci

6 Webové rozhraní

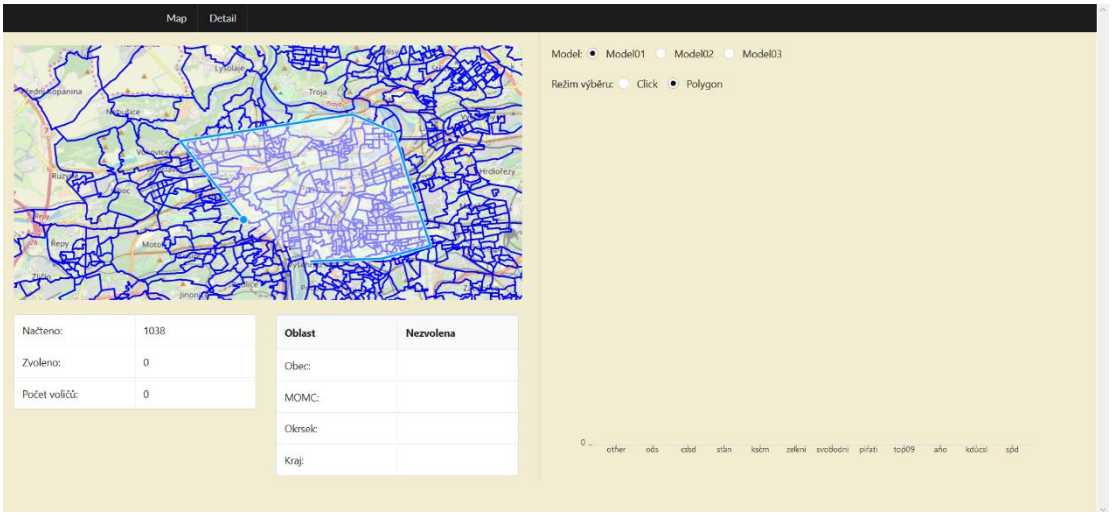
K zobrazení výsledků bylo vytvořeno webové rozhraní. To umožňuje pro vybrané volební okrsky zobrazit volební výsledky a výsledky voleb a modelu. Skládá se ze dvou částí: Map – výběr okrsků a jejich zobrazení - a Detail – tvorba demografických údajů a získaní předpovědi. Mezi těmito se volí v menu v horní liště.



Obr. 6:1 - Webové uživatelské rozhraní

V části Map dojde po načtení stránky v levé části k zobrazení mapy České republiky jako podkladu, nad kterým se zobrazí okrsky, respektive jejich hranice. Ty jsou vyznačeny tenkými modrými čarami.

Zvolit okrsek lze kliknutím, přidat další okrsky do výběru je potom možné kliknutím s podržením klávesy SHIFT. Alternativní způsob výběru je výběr nakreslením polygonu do mapy – přidány budou všechny okrsky v něm zcela obsažené. Po zvolení režimu výběru Polygon v pravé části je možné klikáním do mapy přidávat hrany polygonu a následně ho dvojklikem uzavřít. Po jeho uzavření se okrsky přidají do výběru. Celkové výsledky ve výběru jsou poté uvedeny ve sloupcovém grafu v pravé části.



Obr. 6:2 - Výběr volebních okrsků pomocí polygonu

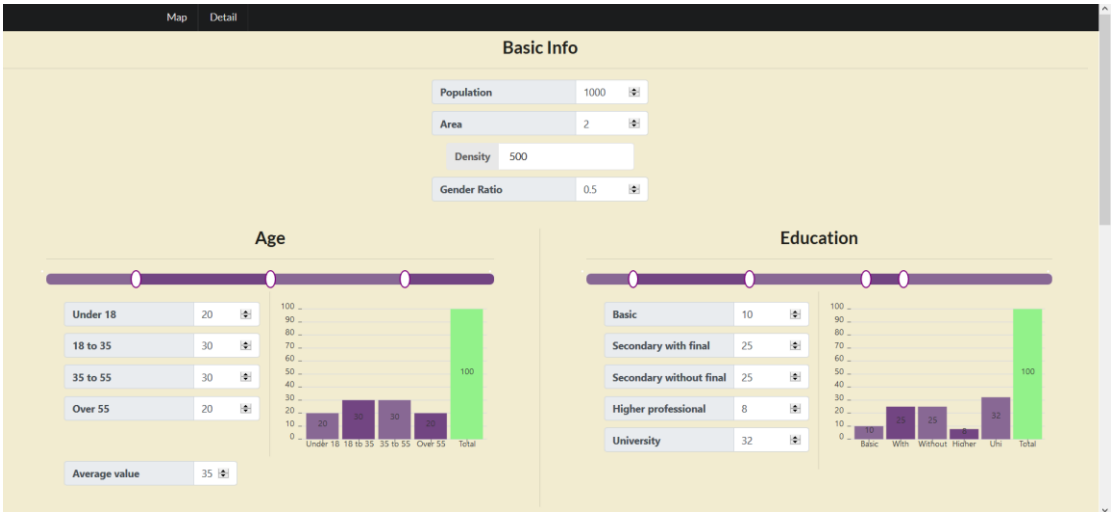
Sloupcový graf zobrazuje skutečné výsledky (barevně) a výsledky modelu (nebarevně). Model je možné vyměnit přepínačem nad grafem. To vyvolá načtení nových dat. V dolní levé části jsou zobrazeny informace o výběru a počtu právě načtených okrsků.

Obr. 6:3 - Výběr volebních okrsků pomocí polygonu - výsledek

Další částí je část pojmenovaná Detail. V této části je obsaženo rozhraní pro vytvoření vektoru demografických dat. Ty jsou podle souvislostí seskupeny do témat, protože spolu souvisejí a existují mezi nimi vztahy. Jako příklad uveďme procentuální rozdělení populace do skupin podle věku. Všechny skupiny musí pochopitelně dát dohromady celou populaci v daném okrsku.

Toto uspořádání podle témat lze vidět v dolní části obrázku níže. Je možné zadat vstupy přímo jako čísla nebo je vyjádřit jako rozdělení proporcí na čáře. Aby bylo zobrazení názornější, je přiložen i sloupcový graf těchto údajů.

Po zadání všech údajů je možné získat předpověď stisknutím tlačítka „Predict“ ve spodní části stránky. Po odpovědi serveru jsou výsledky zobrazeny opět ve sloupcovém grafu.



Obr. 6:4 - Rozhraní pro zadávání demografických údajů

7 Závěr

Sestavili jsme model závislosti politických preferencí na demografických datech. K tomuto jsme využili neuronové sítě. Otestovali jsme množství konfigurací modelů neuronových sítí a vybrali dva nejlepší.

Pro lepší představení těchto modelů jsme vytvořili webovou aplikaci. Ta umožňuje prohlížet přesnost jednotlivých modelů v libovolně zvolených oblastech Středočeského a Jihomoravského kraje a Prahy. Kromě toho je v ní také možné si vytvořit vlastní demografická data.

8 Zdroje

[1] M. Čertický, J. Drchal, M. Cuchý a M. Jakob, „Fully Agent-based Simulation Model of Multimodal mobility in European Cities," *Models and Technologies for Intelligent Transportation Systems (MT-ITS),* 2015.

[2] Chollet, Francois, et al., *Keras,* https://keras.io, 2015.

[3] Abadi, Martin; et al., *TensorFlow: Large-scale machine learning on heterogeneous distributed systems,* 2016.

[4] S. Haykin, v *Neural Networks, A Comprehensive Foundation* , Pearson, 1999, pp. 23-25.

[5] J. A. Anderson, „The Generic Neural Network Neuron," v *An Introduction to Neural Networks*, MIT press, 1995, pp. 54-55.

[6] Chollet, Francois, et al., „Keras.io," 2015. [Online]. Available: http://keras.io/optimizers/#adam.

[7] D. P. Kingma a J. Ba, „Adam: A Method for Stochastic Optimization," *arXiv:1412.6980,* 2015.

[8] P. Ramachandran, B. Zoph a Q. V. Le, „Searching for Activation Functions," *ICLR 2018 Workshop Submission,* č. 405, p. 1, 2018.

[9] D.-A. Clevert, T. Unterthiner a S. Hochreiter, „Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," *arXiv:1511.07289v5,* 22. 2. 2016.

[10] B. Xu, N. Wang, T. Chen a M. Li, „Empirical Evaluation of Rectified Activations in Comvolution Network," *arXiv:1505.00853v2,* 27. 11. 2015.

[11] R. A. Dunne a N. A. Campbell, „On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function," v *Proc. 8th Aust. Conf. on the Neural Networks*, Melbourne, 1997.

[12] C. J. Willmott a K. Matsuura, „Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," *Clim Res,* pp. 79-82, 19 Prosinec 2005.

[13] C. Sammut a G. Webb, „Mean Squared Error," v *Encyclopedia of Machine Learning*, Boston, MA, Springer, 2011.

[14] M. Nielsen, „Neural Networks and Deep Learning," [Online]. Available: http://neuralnetworksanddeeplearning.com/chap3.html#introducing_the_cross-entropy_cost_function. [Přístup získán 2018].

[15] C. Sammut and G. I. Webb, "Mean Absolute Error," in *Encyclopedia of Machine Learning*, Boston, MA, Springer, 2011.

[16] L. F. Sobková a M. Čertický, „Urban Mobility and Influence Factors - Case Study Prague," *WIT Transactions on The Built Environment, Vol. 176 (ISSN: 1743-3509, DOI: 10.2495/UT170181),* pp. 207-217, 2017.

[17] D. Fiedler, M. Čáp a M. Čertický, „Impact of Mobility-on-Demand on Traffic Congestion: Simulation-based Study," *20th IEEE International Conference on Intelligent Transportation Systems (ITSC 2017),* 2017.

[18] „Souřadnicové referenční systémy," Český úřad zeměměřičský a katastrální, 5 5 2018. [Online]. Available: http://geoportal.cuzk.cz/(S(splivzkhiqoopfxrejvce30b))/Default.aspx?lng=CZ&mode=TextMeta&side=sit.trans&text=souradsystemy.

[19] J. D. Hunter, „Matplotlib: A 2D graphics environment," *Computing In Science & Engineering,* sv. 9, č. 3, pp. 90-95, 2007.

[20] T. O. G. C. (OGC), „OGC Filter Encoding," [Online]. Available: https://www.opengeospatial.org/standards/filter. [Přístup získán 10 Říjen 2018].

[21] O. S. G. Foundation, „Geoserver Filter Encoding," [Online]. Available: https://docs.geoserver.org/latest/en/user/filter/filter_reference.html. [Přístup získán 10 Říjen 2018].

[22] P. Ramachandran, B. Zoph a Q. V. Le, „Searching for Activation Functions," *ICLR 2018 Workshop Submission,* p. 1, 2018.

Příloha

IP adresa webové aplikace je 35.204.227.8, protokol HTTP, port 80. Je doporučeno použití moderních webových prohlížečů.

Přiložené CD obsahuje:

- Soubor pdf s touto prací
- Složku scrap se zdrojovými kódy skriptů pro získání dat
- Složku import se zdrojovými kódy skriptů pro nahrání dat do databáze
- Složku keras se zdrojovými kódy implementace modelu a uloženými průbehy cross-validace (jako pickle objektu Trained)
- Složku web-react se zdrojovými kódy webové aplikace (je potřeba stáhnout balíčky příkazem yarn install, k čemuž je nutné mít nainstalovaný yarn, aplikace poté spusťte příkazem yarn start, AJAX požadavky pochopitelně budou bez odpovědí.)
- Složku api s PHP skripty
- Složku flask-app se zdrojovými kódy Python API (ke spuštění postačí příkaz python app.py v příkazové řádce)
- Složku setup, která obsahuje textové soubory, ve kterých jsem se pokusil zachytit celou instalaci serveru na operačním systému Ubuntu 18.04 na Google Cloud Platform. Soubory obsahují posloupnost příkazů a obsah některých souborů nebo nastavení.