ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA STAVEBNÍ

STUDIJNÍ PROGRAM GEODÉZIE A KARTOGRAFIE

OBOR GEOMATIKA

DIPLOMOVÁ PRÁCE

Využití strojového učení v oblasti geodézie a kartografie

Usage of Machine Learning in geodesy and cartography field

Vedoucí práce: Ing. Jan Pytel, Ph.D.

Katedra geomatiky

Květen 2022

Jan Kučera

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta stavební
Thákurova 7, 166 29 Praha 6

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: Kučera Jméno: Jan Osobní číslo: 477480

Zadávající katedra: 11155 - Katedra geomatiky

Studijní program: magisterský navazující na bakalářský

Studijní obor/specializace: 36T0GKGM - Geomatika

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce: Využití strojového učení v oblasti geodézie a kartografie

Název diplomové práce anglicky: Usage of Machine Learning in geodesy and cartography field

Pokyny pro vypracování:
Cílem diplomové práce se zaměřit na využití strojového učení pro oblast geodézie a kartografie. Cílem diplomové práce je v kosce popsat strojové učení (používané modely ...) a pokusit se implementovat strojové učení na praktickém využití v oblasti geodézie a kartografie.

Seznam doporučené literatury:
Kurzy ohledně Machine Learning na www.udemy.com
spark.apache.org
Building Machine Learning Powered Applications - Going from Idea to Product
Yeaser S. Abu-Mostafa - Learning From Data

Jméno vedoucího diplomové práce: Jan Pytel

Datum zadání diplomové práce: 4.2.2022

Termín odevzdání DP v IS KOS: 15. 5. 2022
Údaj uveďte v souladu s datem v časovém plánu příslušného ak. roku

[redacted]
Podpis vedoucího práce

[redacted]
Podpis vedoucího katedry

III. PŘEVZETÍ ZADÁNÍ

Beru na vědomí, že jsem povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je nutné uvést v diplomové práci a při citování postupovat v souladu s metodickou příručkou ČVUT „Jak psát vysokoškolské závěrečné práce" a metodickým pokynem ČVUT „O dodržování etických principů při přípravě vysokoškolských závěrečných prací".

8.3.2022
Datum převzetí zadání

[redacted]
Podpis studenta(ky)

Čestné prohlášení

Tímto prohlašuji, že jsem diplomovou práci na téma „Využití knihoven strojového učení v oblasti geodézie“ vypracoval samostatně.

Veškerou použitou literaturu a podkladové materiály uvádím v seznamu zdrojů.

V Praze dne …… ……

(podpis autora)

Poděkování

První a největší mé díky chci věnovat vedoucímu této práce Ing. Janu Pytlovi, Ph.D. za odborné vedení a za spoustu jiných zkušeností nabitých pří vytváření této práce jak v oboru IT, tak i v oblasti profesních a životních rad.

Též bych svá poděkování rád věnoval všem ostatním, kteří mi pomohli či byli oporou jak v této práci, tak i během celého studia. Zvláště tedy rodině, přítelkyni, vyučujícím a spolužákům.

Abstrakt

Tato práce se věnuje tématu strojového učení a jeho možného využití do jiných oblastí jako je geodézie. V první části práce je rozebrána stručná a základní teorie a v části druhé je pak část praktická zaobírající se praktickým využitím. Velkou roli zde bude hrát služba Azure od společnosti Microsoft, kterou budu využívat prostřednictvím implementace jejich nástrojových balíčků (knihoven) do programovacích prostředí jazyků Java a Python. První praktickou ukázkou bude rozpoznávání tváří a druhou vytvoření modelu detekující vlastní zvolené objekty.

Klíčová slova

Strojové učení, Hluboké učení, Regrese, Klasifikace, Klastrování, Učení asociačních pravidel, Zpětnovazební učení, Zpracování přirozeného jazyka, Neuronová síť, Scikit learn, Keras, Tensorflow, AWS, Azure, Azure Cognitive Services, Face API, Custom Vision, Eclipse, Java, Python.

Abstract

This work focuses on Machine Learning subject and its possible usage in other fields like geodesy. In the first part of this work there will be some theory explained and the second part will be about its practical usage. A big role in here plays Azure services from Microsoft, which I'll be using through implementation of its libraries into environment of programming languages Java and Python. The first usage example is face recognition and the second is about creating my own model for custom object detection.

Keywords

Machine learning, Deep learning, Regression, Classification, Clustering, Association rule learning, Reinforcement learning, Natural language processing, Neural network, Scikit learn, Keras, Tensorflow, AWS, Azure, Azure Cognitive Services, Face API, Custom Vision, Eclipse, Java, Python.

Obsah

Obsah 6

Použité zkratky 8

Úvod 9

1 Strojové učení 9

1.1 Regrese 10

1.1.1 Jednoduchá lineární regrese 11

1.1.2 Vícenásobná lineární regrese 11

1.1.3 Polynomická lineární regrese 12

1.1.4 Regrese podpůrných vektorů 12

1.1.5 Regrese rozhodovacím stromem 13

1.1.6 Regrese náhodným lesem 14

1.2 Klasifikace 15

1.2.1 Logistická regrese 15

1.2.2 K-nejbližších sousedů 16

1.2.3 Metoda podpůrných vektorů 17

1.2.4 Kernel SVM 18

1.2.5 Naivní bayesovská metoda 21

1.2.6 Klasifikace rozhodovacím stromem 22

1.2.7 Klasifikace náhodným lesem 23

1.3 Klastrování 24

1.3.1 Klastrování k-průměrů 24

1.3.2 Aglomerativní hierarchické klastrování 26

1.4 Učení asociačních pravidel 29

1.4.1 Apriori 29

1.4.2 Eclat 31

1.5 Zpětnovazební učení ... 31
1.5.1 Horní hranice spolehlivosti ... 32
1.5.2 Thompsonovo vzorkování ... 33
1.6 Zpracování Přirozeného Jazyka ... 34
1.6.1 Bag-of-words model ... 35
1.7 Hluboké učení ... 36
1.7.1 Umělé Neuronové Sítě ... 36
1.7.2 Konvoluční neuronové sítě ... 39
1.8 Evoluční algoritmy ... 40
2 Knihovny ... 41
3 Webové služby ... 42
4 Využití služby Microsoft Azure ... 42
4.1 Azure portál a vytváření prostředku kognitivních služeb ... 42
4.2 Připravení prostředí Eclipse ... 44
4.3 FaceAPI ... 44
4.3.1 Vytvoření klienta ... 44
4.3.2 Nastavování parametrů vracených službou ... 45
4.3.3 Zpracování obrázku a přistupování k výsledkům ... 46
4.3.4 Příklady výsledků FaceAPI ... 47
4.4 Zhodnocení ... 49
5 Machine Learning v geodézii ... 49
5.1 Automatické vyhledávání vlícovacích bodů ... 50
5.1.1 Klasifikace vs objektová detekce ... 51
5.1.2 Služba Azure Custom Vision pro vyhledávání VB ... 51
5.1.3 Představení výsledků služby Azure Custom Vision ... 60
5.1.4 Výsledky ... 62
Závěr ... 66

Literatura a zdroje 67

Zdroje trénovacích a testovacích obrázků 70

Seznam obrázků 73

Použité zkratky

SVM – Support Vector Machine

ML – Machine Learning (Strojové učení)

ANN – Artificial Neural Network

CNN – Convolutional Neural Network

NLP – Natural Language Processing

EA – Evolutionary Algorithms

UCB – Upper Confidence Bound

RBF – Radial Basis Function

K-NN – K-Nearest Neighbours

RTS – Robotická totální stanice

GIS – Geoinformační systém

VB – Vlícovací bod

Úvod

S postupem času se vývoj lidské technologie dostává rychle kupředu. Technologický vývoj dokonce rapidně zrychluje. Kapacity paměťových zařízení jsou teď o mnoho řádů výše než před 20 lety. To samé se dá říct o výpočetním výkonu procesorů. Nejen tyto technologické oblasti ale třeba i vývoj matematiky, jakožto prostředku k algoritmizaci úloh, a další oblasti velice přispívají sice k zanikání některých stávajících oborů, třeba z důvodu automatizace, ale dávají vzniknout úplně novým techničtějším oborům. Jedním z fenoménů moderní doby díky technologickým kapacitám je strojové učení nebo rovnou obecně umělá inteligence. Tyto pojmy sice existují už přes 60 let, ale jejich využití a rozmach je nejznačnější právě nyní. Podle mého názoru je jen otázkou času, kdy se tyto obory plně dostanou do všech jiných oborů, geodézie či kartografie nevyjímaje. První invaze strojového učení do geodézie už jsou známé jako například automaticky na operátorův cíl cílící robotické totální stanice nebo třeba GIS nástroje pro interpolace a extrapolace a další.

V této práci budou rozebírány aparáty strojového učení a ukázkou jejich možného využití. Nejprve bude pozornost věnována teoreticky hlavně na úrovni intuice různým typům algoritmů a modelů. Vysvětlení budou tedy hlavně cílena na vytvoření představy o principu fungování. Matematická stránka věci tohoto tématu není v této práci podstatná a věnována ji bude pozornost pouze s ohledem na didaktičnost vysvětlování a jeho obtížnosti. Pro účely této práce bude snaha některé pojmy překládat do češtiny. V programovací branži se ale využívá zpravidla anglického názvosloví a pojem jako Machine Learning a další se většinou nepřekládají.

Další částí bude praktická stránka věci cílící na vytváření podnětů pro aplikaci strojového učení kde se dá. Konkrétně zájmem bude rozpoznávání obrazu a detekce objektů. Tématy budou jak rozpoznávání tváří, tak i detekce vlastních objektů jako třeba vlícovací body. Práci budou doprovázet samozřejmě ukázky kódu a výsledky, které jsou samozřejmě nejnázornější.

1 Strojové učení

Rozpoznávání řeči a její převod na text, rozpoznávání obličejů na fotografiích na sociálních sítích a jejich přiřazování k příslušným uživatelským účtům, rozpoznávání pohybů s headsetem virtuální reality či snímání a rozpoznávání pohybů snímaných Kinect senzorem. To jsou jedny příklady z mnohých, ve kterých je dneska využito aparátů strojového učení a existuje jich mnohem víc a vývoj ukazuje, že budou stále hojně přibývat. Jde o součást

moderního oboru umělé inteligence, kde strojové učení hraje velice významnou roli. Je nástrojem, díky kterému je lidstvo schopné řešit rozličné problémy výpočetní technikou. Může se jednat jednak o atraktivní oblasti již zmíněné na začátku tohoto odstavce, ale taky třeba o statistické analýzy či predikce do budoucna.

Strojové učení neboli *Machine Learning* (ML) je obor založený na algoritmech, které se využíváním dat sami vyvíjí. Taková data se pak nazývají trénovací. Výsledný vytrénovaný model pak může sloužit k predikci ke klasifikaci či k nějakým odhadům nebo rozhodováním. Například zdali zákazník koupí nějaký daný produkt nebo ne a to na základě dat o něm. Třeba na základě výše jeho měsíčního platu a jeho věku. Je ale ovšem zapotřebí dostatečného množství dat a taky potřebnou korelaci dat s výsledkem. Metod strojového učení je hodně, ale v této práci se budeme zabývat regresí, klasifikací, klastrováním, učením asociačních pravidel, zpětnovazebním učením, zpracováním přirozeného jazyka, evolučními algoritmy a neuronovými sítěmi jakožto nástroji hlubokého učení. Všem metodám ale předchází část předzpracování. Je potřeba data uvést do podoby, které nebudou vadit strojovému zpracování. Data by kupříkladu neměla obsahovat prázdná pole. Tyto chybějící data je potřeba nějak doplnit. Doplňují se takové hodnoty, které by na výsledek měly mít co možná nejmenší vliv, jako třeba doplnění nulou nebo aritmetickým průměrem zbývajících dat či úplné vynechání konkrétního pozorování (pozorováním se myslí jeden řádek dat). Ale úplné vynechání může být škoda a někdy je dobré využít všechna dostupná data. Jeli ale dat dostatečné množství, tak vynechání konkrétních pozorování může být šetrnou volbou. Data také mohou obsahovat kategorie, například státní příslušnost či zdravotní pojišťovnu nebo název obchodního produktu a tak dále. V takovém případě je potřeba takové hodnoty kódovat, aby počítač mohl pracovat s čísly a ne s textem.

V rámci této kapitoly bylo čerpáno z [1].

1.1 Regrese

Regrese je studium, kde se snažíme odhadnout chování dat, která jsou spojitá. Trénovací data sice budou dána diskrétně, ale celý charakter je kontinuální (třeba v čase). Například když odhadujeme plat pracovníka, který může záviset na době jeho praxe a chceme mu jakožto zaměstnavatel poskytnout optimální platovou nabídku. V regresi se využívá velkého množství modelů. Zde si ale budeme představovat jen některé z nich.

1.1.1 Jednoduchá lineární regrese

Anglicky *Simple linear regression* je nejjednodušším příkladem regresního modelu. Je velice jednoduchý, dobře představitelný a dobře se na něm představují základní principy regrese. Jednoduchou lineární regresi popisujeme touto rovnicí:

$$y = a + bx. \tag{1.1}$$

Koeficienty a a b jsou počítačem určovány na základě trénovacích dat. Parametr x se nazývá nezávislou proměnnou a je složkou reprezentující spojitý charakter funkce (může se jednat například o čas, rychlost, množství síly, vzdálenost apod.) a parametr y se nazývá závislou proměnnou, kterou se snažíme rovnicí popsat a co nejlépe odhadnout pro hodnoty x, které už jsou mimo rozsah dat obsažených v trénovací množině. Kdyby se jednalo o čas, tak se budeme snažit třeba predikovat budoucnost chování. Tento model se v praxi moc nepoužívá, hlavně protože v praxi většinou nemáme pouze jednu nezávislou proměnnou, ale veškerá pozorování obsahují více proměnných, protože většinou z podstaty věci je výsledek ovlivňován větším množstvím vlivů. Například čas, zkušenosti, plat v předešlé práci, dosažené vzdělání atp., budou ovlivňovat platové ohodnocení člověka ucházejícího se o práci ve firmě.

V rámci této kapitoly bylo čerpáno z [1].

1.1.2 Vícenásobná lineární regrese

Neboli *Multiple linear regression* je rozšířeným modelem oproti jednoduché lineární regresi. Obsahuje totiž celý vektor nezávislých proměnných, a tedy i lépe popisuje reálné závislosti většiny problému. Rovnici takového modelu bychom pak zapsali:

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n. \tag{1.2}$$

Vektor x je vektorem nezávislých proměnných a vektor b je vektor koeficientů. Tyto koeficienty můžeme chápat jako míru vlivu příslušné nezávislé proměnné. Číslo n pak značí počet nezávislých proměnných. Využití tohoto modelu je vhodné tam, kde data mají lineární charakter.

1.1.3 Polynomická lineární regrese

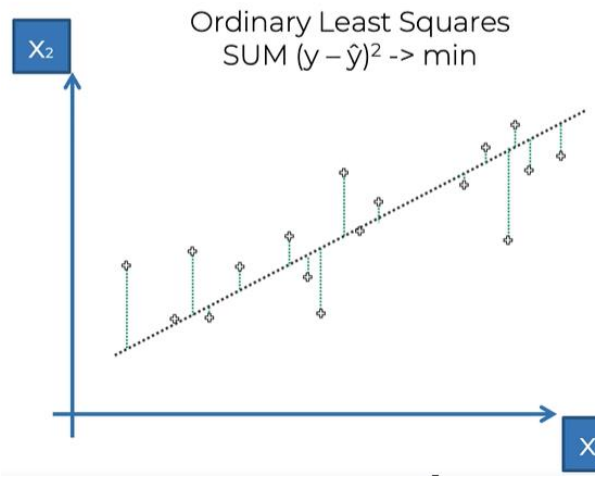
Anglicky *Polynomial linear regression* je model tvarem podobný vícenásobné lineární regresi. Jak už napovídá název, jedná se zde o polynomický tvar regresní křivky. Rovnici bychom vyjádřili takto:

$$y = b_0 x_0^0 + b_1 x_1^1 + b_2 x_2^2 + b_3 x_3^3 + \cdots + b_n x_n^n. \tag{1.3}$$

Název nám sice říká, že se jedná o lineární rovnici a my přitom zde můžeme vidět nezávislé proměnné v exponenciálním tvaru. To je proto, že název lineární se vztahuje k vektoru koeficientů *b*, které odhadujeme. Data nebudou mít vždy lineární charakter a pro přesnější odhad může být lepší tento model. Například můžeme uvažovat, že v některých zaměstnáních může doba praxe člověka ucházejícího se o pozici ve firmě exponenciálně ovlivňovat jeho výsledný plat.

Ve všech případech zmíněných lineárních regresí se vyrovnaná hodnota parametrů hledá metodou nejmenších čtverců. Ta se snaží minimalizovat celkovou sumu čtverců chyb vyrovnávající regresní křivky vzhledem k původním hodnotám. Regresní křivku si moc dobře u vícedimensionálních případů představit nelze, tak je pro jednoduchost lepší ji chápat jako vektor konkrétního řešení.

V rámci této kapitoly bylo čerpáno z [1].

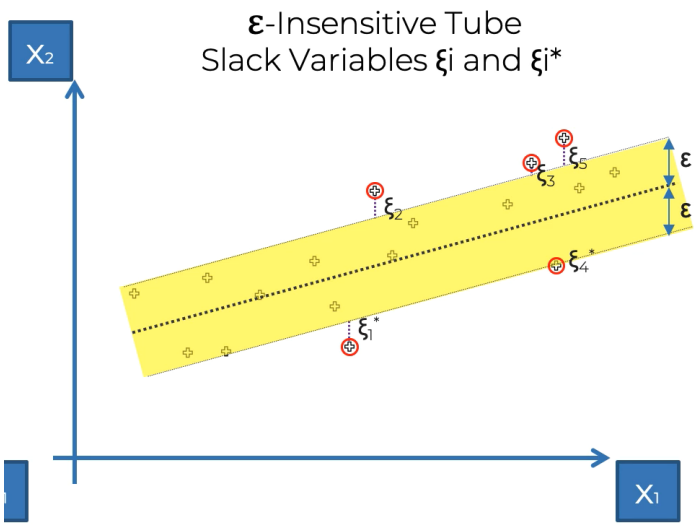


Obrázek 1 - Příklad vyrovnání metodou nejmenších čtverců [1]

1.1.4 Regrese podpůrných vektorů

Anglicky *Support vector regression* (dále SVR). Hlavní rozdíl SVR oproti lineárním regresím je vyrovnávací křivka. Zde se kromě tvaru a polohy křivky uvažuje i její obalová zóna (viz

obrázek 2). Vzorec pro tento model křivky už je o poznání složitější a uveden zde nebude. Princip je ale podobný. I zde je snaha minimalizovat sumu čtverců chyb, ale pouze těch pozorování, které jsou mimo rozsah obalové zóny. Její rozptyl charakterizuje parametr ε. (anglicky se této zóně říká „insensitive tube“) To dává této metodě vetší míru flexibility, protože v realitě jsou chyby všudypřítomné a je proto někdy vhodné disponovat určitou mírou tolerance. Výsledné křivky po takovém vyrovnání se pak mohou dost lišit od křivek lineární regrese.

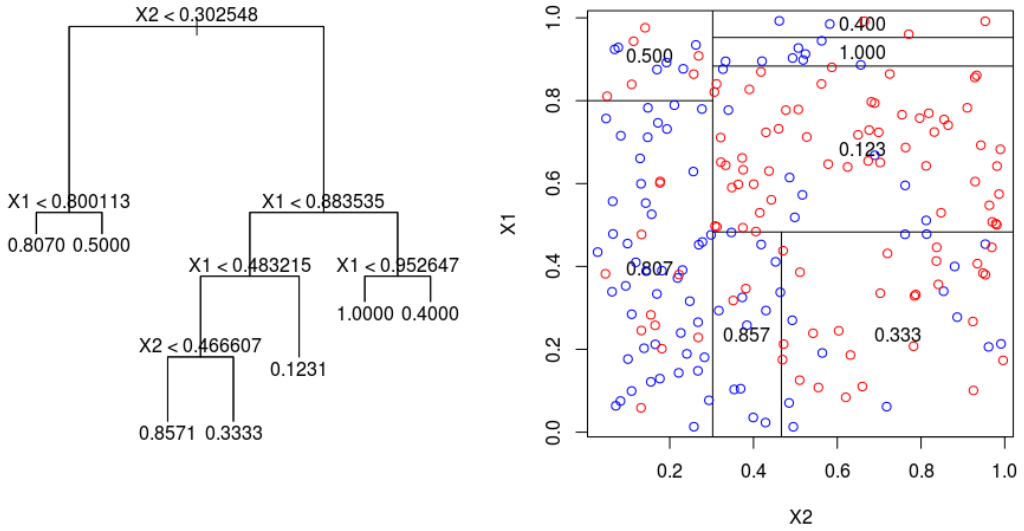


Obrázek 2 - Ukázka vyrovnávací křivky SVR [2]

V rámci této kapitoly bylo čerpáno z [1].

1.1.5 Regrese rozhodovacím stromem

Tato metoda *Decision Tree Regression* je nazvána podle stromového grafu, který ilustruje myšlenku tohoto modelu (viz obrázek 3). Jedná se o rozdělování dat na takzvané listy. Rozdělování dat do listů je problematikou, kterou se zabývá studium informační entropie (anglicky „information entropy”, popř. „information theory”). Zde se tímto zabývat nebudeme. Pro představu celé idey modelu využijme obrázku 3.



Obrázek 3 - Ukázka Decision Tree [5]

Pro lepší představu zde budou uvažovány 2 nezávislé veličiny. Závislou veličinu y si lze představit jako třetí osu vystupující z obrázku, ale to není teď podstatné. Algoritmus na základě určitých podobností data rozdělí na konečné listy („terminal leaves"). To je možné vidět na obrázku vpravo. *V* levé části se pak nachází stromový graf. Tato struktura se vytvoří vytrénováním modelu trénujícími daty. Predikce pak vypadá tak, že algoritmus podle hodnot nezávislých proměnných zařadí dané pozorování do jednoho z konečných listů (na stromovém grafu by to šlo udělat vyhodnocováním nerovností odshora dolů) a aktuálnímu pozorovaní přiřadí hodnotu průměrovanou ze všech pozorování v tomto konečném listu. Například pro pozorování, kde $x_1 = 0{,}7$ a $x_2 = 0{,}8$ by predikce byla $y = 0{,}123$. Nevýhodou tohoto modelu je, že je zapotřebí většího množství trénovacích dat. Umí si ale dobře poradit jak s lineárními, tak i s nelineárními problémy.

V rámci této kapitoly bylo čerpáno z [1].

1.1.6 Regrese náhodným lesem

Random Forest Regression. Název tohoto modelu je analogií předcházejícího modelu. Nevytváří se zde pouze jeden strom ale větší množství stromů, tedy celý les. Algoritmus je založen na tom, že se nejprve náhodně určí množství dat, nad kterými se vytvoří stromový model. Poté se určí počet stromů, které se mají vytvořit a proces se následně opakuje. Tedy se znovu určí množství dat a vytvoří se stromový model. Každý model pak predikuje vlastní hodnotu. Výsledný odhad je pak průměrem predikcí všech vytvořených rozhodujících stromů. Výsledkem je pak mnohem silnější model, který dává obyčejně lepší výsledky než předchozí

model. Nevýhodou je, že může snadno dojít k jevů zvanému „overfitting". Jedná se o přetrénování modelu, což vede k tomu, že model funguje příliš dobře na trénovacích datech, ale na jiných už ne a výsledky (predikce) jsou nepřesné.

V rámci této kapitoly bylo čerpáno z [1].

1.2 Klasifikace

Klasifikace je oblastí, kde se nesnažíme odhadovat spojité hodnoty, ale snažíme se určit diskrétní hodnoty z omezené množiny řešení. Například se snažíme-li se rozhodnout, zda zákazník koupí daný produkt nebo ne (pouze 2 možná řešení). Nebo zdali se geodetické měření dá považovat za odlehlé či nikoliv. Opět je zde velké množství algoritmů jako u regrese. Mnohé z nich pracují na stejném principu jako u jejich stejnojmenných protějšků z regresních modelů.

1.2.1 Logistická regrese

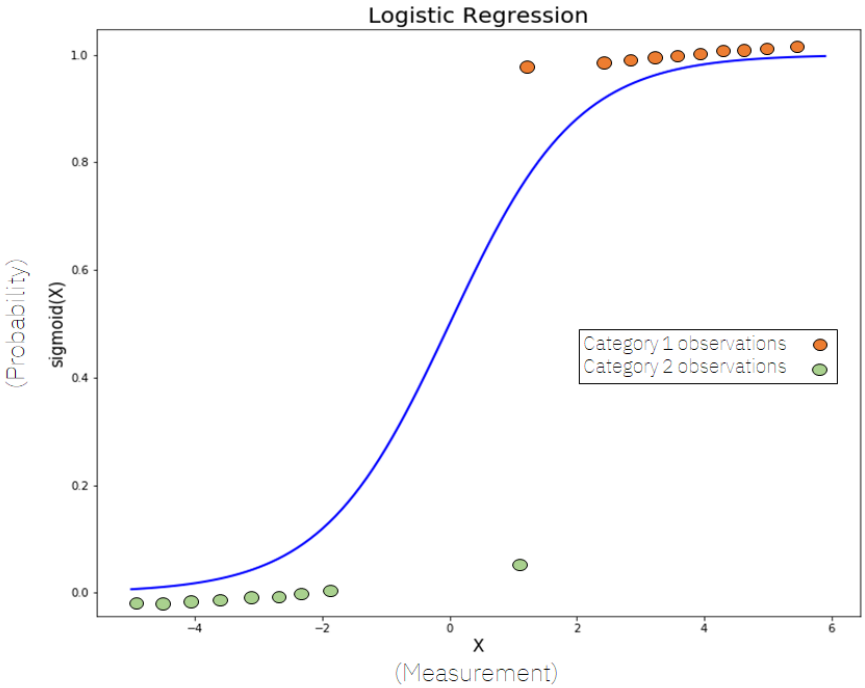
Původně *Logistic regression*, je model, který vychází z jednoduché lineární regrese. Myšlenkou je opět proložit data co možná nejlépe vystihující funkcí. Zde nás ale bude zajímat pravděpodobnost či odhad pro konkrétní rozhodnutí (kategorizaci) oproti jednoduché lineární regresi, která nám popisuje plynulé změny. Například snažíme-li se určit, zda je měření odlehlé nebo ne. Zde máme pouze 2 kategorie (stavy), které potřebujeme předpovědět a nějaké hodnoty mezi tím nás nezajímají. Třeba hodnoty 0, když měření není odlehlé a 1, když měření je odlehlé. Hodnota 0,5 tedy nemá valný smysl. Proto přichází na scénu funkce sigmoida:

$$P(y) = \frac{1}{1 + e^{-y}} \tag{1.4}$$

Jedná se o konkrétní typ logistické funkce, která nám převádí situaci na pravděpodobnost, tedy do intervalu $y \in [0,1]$. Vyjádřením parametru y ze sigmoidy a dosazením do rovnice jednoduché lineární regrese získáme funkci:

$$\ln\left(\frac{p}{1-p}\right) = b_0 + b_1 x \tag{1.5}$$

Toto je rovnice logistické regrese.



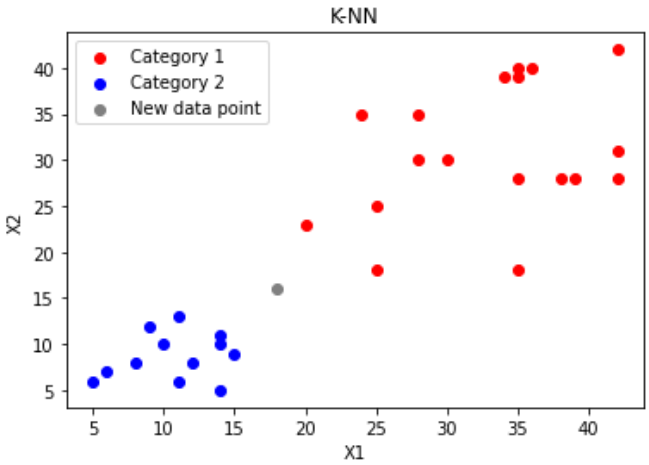
Obrázek 4 - Ukázka křivky logistické regrese [6]

Tato funkce ve výsledku odhaduje pravděpodobnost, s jakou pozorování může patřit do určité kategorie. Standardně kvůli symetrii se volí práh pravděpodobnosti 50% pro kategorizování.

V rámci této kapitoly bylo čerpáno z [1].

1.2.2 K-nejbližších sousedů

K-Nearest Neighbours (K-NN) algoritmus má mnohem jednodušší ideu než předchozí model. Spočívá v tom, že si pro pozorování, které chce klasifikovat, spočítá své sousedy a přiřadí mu hodnotu nejpočetnější kategorie z nich. Počet sousedů se počítá na základě vzdálenosti. Ta nemusí být vždy eukleidovská.



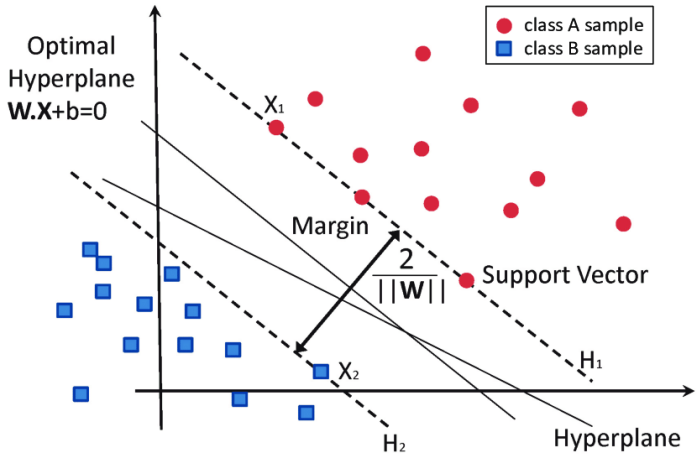
Obrázek 5 - Ukázka situace pro K-NN

Ze situace na obrázku by algoritmus odhadl, že bod (jedno pozorování) patří do kategorie 2, protože největší množství jeho sousedů pochází právě z této kategorie. Další představou aplikace tohoto algoritmu by mohla být klasifikace v GIS. Máme-li digitální mapu v podobě matice čísel, která představují označení typu půdy. Pokud by tato matice nebyla úplná, tak K-NN na základě okolí, třeba že kolem je hodně lesa, odhadne, že dané místo taktéž představuje pixel lesa. Jedná se ale celkově pouze o lineární klasifikátor.

V rámci této kapitoly bylo čerpáno z [1].

1.2.3 Metoda podpůrných vektorů

Support Vector Machine (SVM). Obdoba tu byla u regrese. I zde uvažujeme objekt s obalovou zónou. Avšak vlastnost této zóny je v tomto případě jiná. Na obrázku 5 jsme měli 2 kategorie (modrá a červená) a jednalo se o graf zobrazující dvojici nezávislých proměnných. Algoritmus se snaží najít přímku, která tato data rozděluje. Tato přímka má ale tu vlastnost, že se snaží najít maximální rozpětí obalové zóny takovým způsobem, aby neobsahovala žádné body. Takovou situaci znázorňuje obrázek 6.



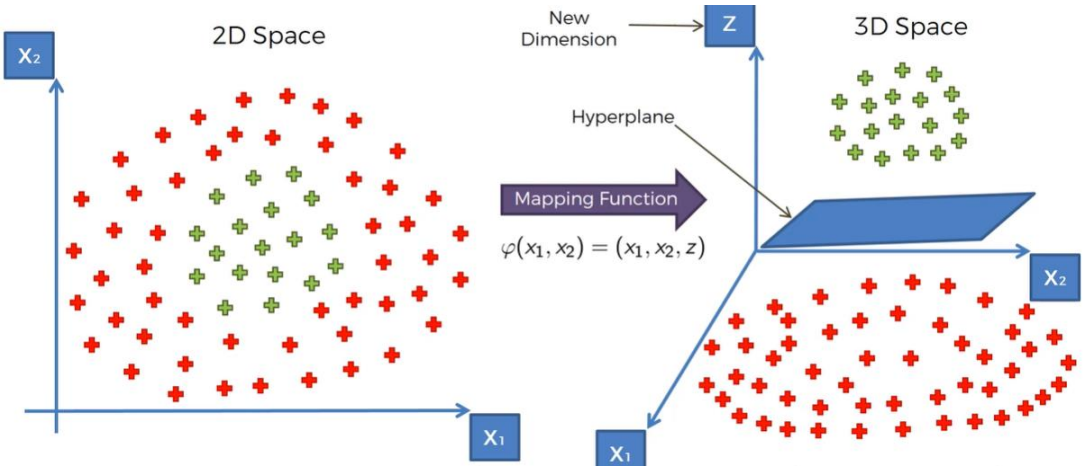
Obrázek 6 – SVM [7]

Na obrázku vidíme sice 2D situaci. Ve 3D, tedy se třemi nezávislými proměnnými, by se jednalo o rozdělující rovinu. V obecných n-dimenzionálních případech pak mluvíme o hyperrovinách („hyperplanes“). Hledaná je pak na obrázku označena jako „optimal hyperplane”. Metoda se nazývá metodou podpůrných vektorů proto, že všechny body jsou reálně vektory nezávislých veličin a ty body, které ohraničují obalovou zónu optimální hyperroviny, jsou oněmi podpůrnými vektory. Modely tohoto klasifikátoru jsou silné v tom, že jsou trénovány tak, aby rozlišovaly rozdíly nejvíce podobných pozorování různých tříd. Jednoduše řečeno, pokud bychom tedy měli trénovací data, kde každé pozorování je vektorem nezávislých proměnných charakterizující podobu objektu. Kombinace hodnot těchto proměnných pak představuje třeba vzhled rodinného domku, trafostanice či kostela. Samotný algoritmus by pak zajímaly hlavně ty případy, kde se tyto objekty vzhledově liší nejméně a mohlo by být možné si je na první pohled splést. Nevýhodou je, že aby model správně fungoval, musí být data lineárně separovatelná.

V rámci této kapitoly bylo čerpáno z [1].

1.2.4 Kernel SVM

Co teda dělat, když jsou data lineárně nerozdělitelná? Lze využít triku, kdy si pomůžeme mapováním dat do vyšší dimenze.

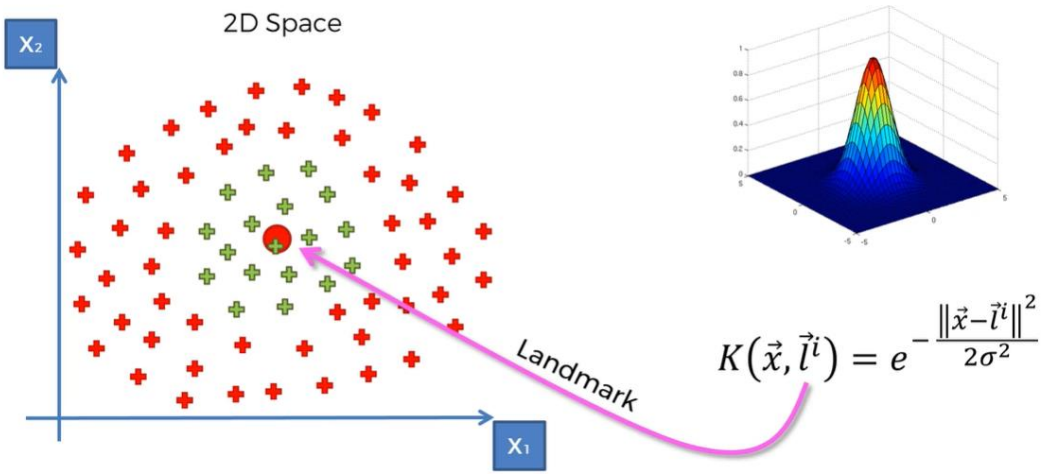


Obrázek 7 - Mapování dat do vyšší dimenze [8]

Na obrázku 7 vlevo vidíme lineárně nerozdělitelná data. Libovolná přímka nám zde stačit nebude. Vytvoříme tedy mapovací funkci projektující body na plášť o dimenzi vyššího objektu (třeba paraboloidu). Tím se nám data posunou a můžeme najít optimální hyperrovinu rozdělující data. Výsledek pak nakonec jen promítneme zpátky a tím nám vzniknou křivkou rozdělené množiny dat. Bohužel mapování do vyšších dimenzí je hlavně při větším množství dat (více nezávislých proměnných) extrémně výpočetně náročné. Trik tedy obměníme a využijeme kernel funkce. Těchto funkcí je víc. Já zde zmíním konkrétní a to Gaussovu RBF Kernel funkci:

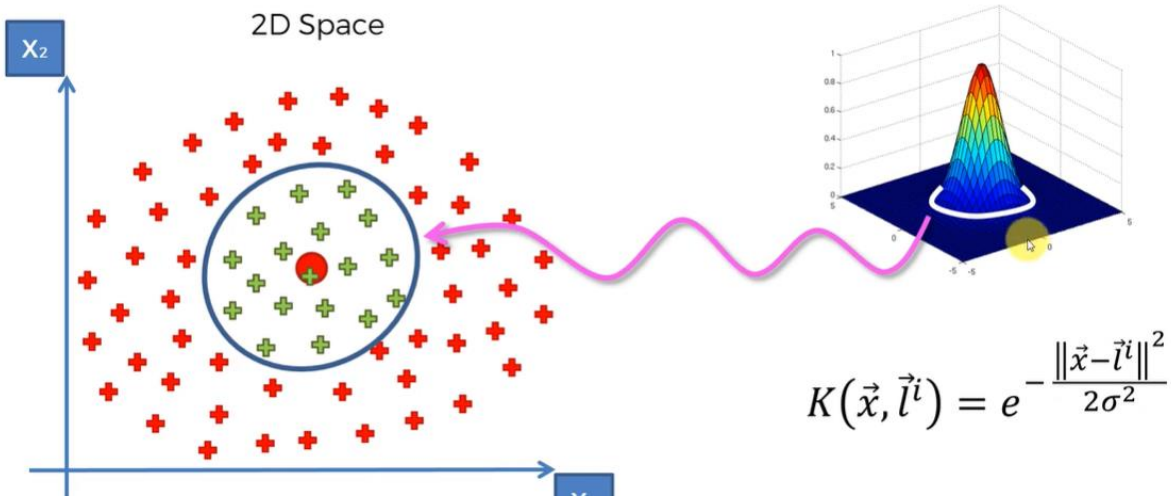
$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\left\|\vec{x}-\vec{l}^i\right\|^2}{2\sigma^2}} \tag{1.6}$$

Kde $\vec{x}$ je vektor nezávislých proměnných, $\vec{l}^i$ jsou tzv. „landmarks“ (orientační body) a σ je parametr funkce. Do dat někam umístíme náš orientační bod (může jich být více). Na umisťování těchto bodů existuje celá metodologie, ale tím se zabývat nebudeme. Tento problém bude ponechám pouze v rovině intuice.



Obrázek 8 - Landmark umístění [8]

Co funkce udělá je, že každému bodu přiřadí číslo vyjadřující vzdálenost k určenému orientačnímu bodu. Jak je vidět z funkce, tento vzdálenostní vztah není lineární a hodnoty kernel funkce blíže k orientačnímu bodu začnou prudce narůstat. Takový průběh je možné názorně vidět na obrázku 8 vpravo. Představa sice uvažuje prostor o dimenzi vyšší, ale kernel funkce žádnou vyšší dimenzi nevyužívá. Pouze počítá, zda vypočtené číslo je vetší než určitá hranice (z principu nula nebo blízká nule) a na základě vyhodnocení této nerovnosti odhadne třídu.



Obrázek 9 - Hranice Kernel SVM [8]

Zde na obrázku 9 je vidět právě taková hranice. Ještě nebyl vysvětlen význam parametru σ. Ten ovlivňuje rozpětí právě tohoto ohraničení. Tato modifikace SVM tedy umožňuje přistupovat i k nelineárním datům.

V rámci této kapitoly bylo čerpáno z [1].

1.2.5 Naivní bayesovská metoda

Anglicky *Naive Bayes* je čistě pravděpodobnostní metoda. Algoritmus spočítá pro nová data pravděpodobnost, s jakou můžou patřit do jednotlivých tříd a vybere tu největší možnou. bayesovská metoda je to proto, že využívá Bayesovy věty („Bayes theorem“):

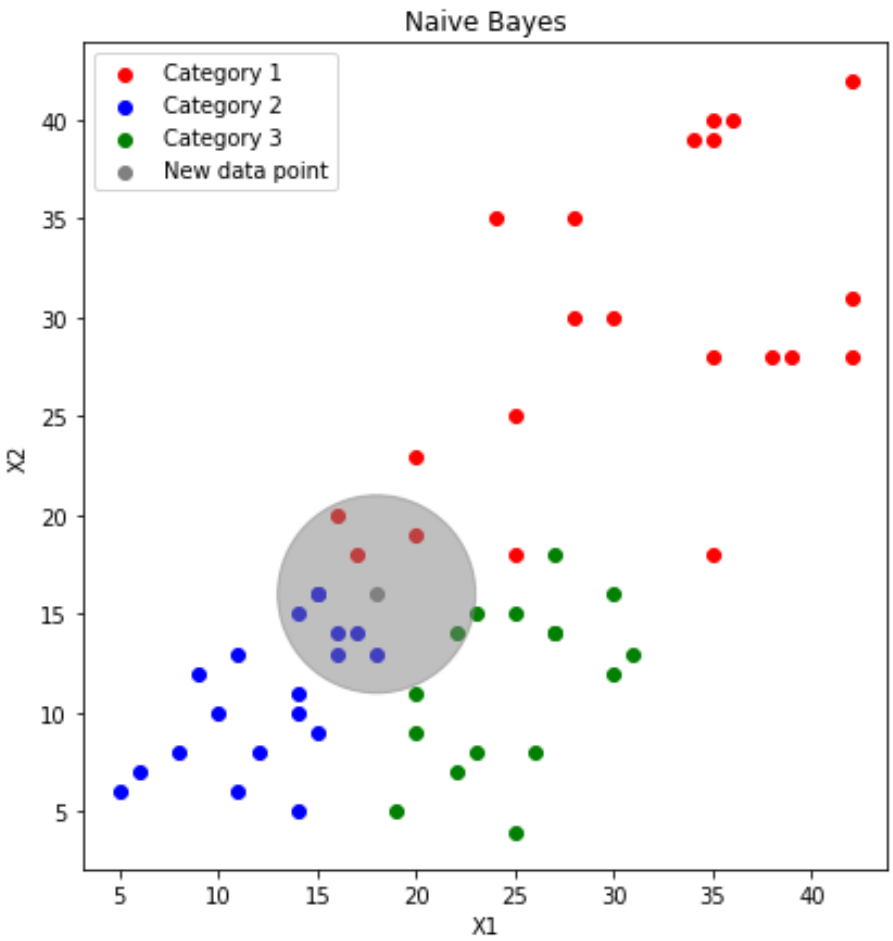
$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} \tag{1.7}$$

Zkráceně a zjednodušeně řečeno se jedná o vypočítání podmíněné pravděpodobnosti (pravděpodobnost jevu A, když platí jev B) za pomoci známé obrácené pravděpodobnosti (pravděpodobnost jevu B, když platí jev A). Metoda se nazývá naivní, protože zmíněná pravděpodobnostní formule předpokládá nezávislost jevů. V praxi toto často splněno není a mezi jevy (nezávislými proměnnými) se může objevovat určitá míra korelace, byť sebemenší. Například věk může korelovat s výší platu, protože vyšší věk může představovat více zkušeností a tím i vyšší hodnotu zaměstnance. Díky neplatnosti nezávislosti je tedy algoritmus naivní, ale přesto se používá.

Aplikace tohoto přístupu v klasifikaci možná nemusí být úplně zřejmá. Začněme s názornější rovnicí:

$$P(A|X) = \frac{P(X|A) * P(A)}{P(X)} \tag{1.8}$$

Toto značení je mírně názornější. X totiž představuje náš nový bod (nové pozorování), který budeme klasifikovat a A značí konkrétní třídu. Nemusí být jasné, co vlastně znamená $P(X)$. Tomuto členu se říká marginální pravděpodobnost („marginal likelihood“) a v tomto členu vlastně sledujeme pravděpodobnost podobných pozorování, tedy kde se hodnoty všech nezávislých proměnných od nového pozorování moc neliší (viz obrázek 10).



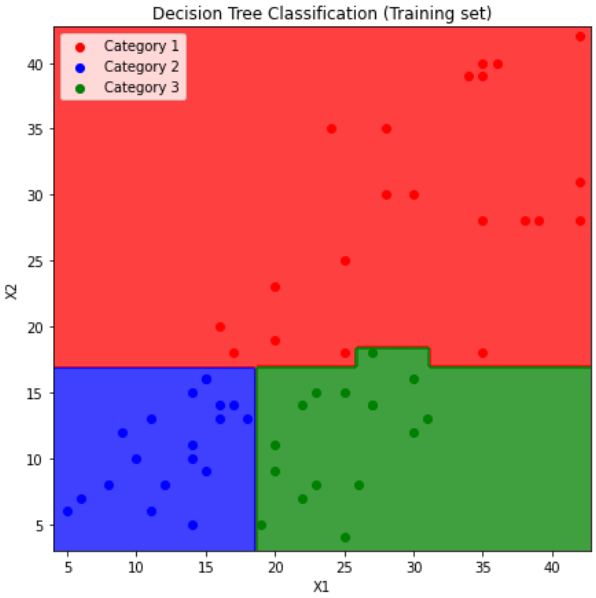
Obrázek 10 - Příklad oblasti podobných pozorování

Takže jinak řečeno podle obrázku 10, s jakou pravděpodobností bude nové pozorování v šedě označeném kruhu. Tato veličina, jak velký rozsah podobnosti uvažujeme, je jedním z parametrů tohoto algoritmu. Lze využít Gaussovy modifikace tzv. *Gaussian Naive Bayes*, který tuto oblast podobným způsobem počítá jako tomu bylo u Kernel SVM (1.6). Potom, co vytrénujeme náš naivní bayesovský model, bude algoritmus propočítávat pravděpodobnosti v každé třídě (jevem A se bude uvažovat každá třída zvlášť). Ty pak porovná a vybere maximum.

V rámci této kapitoly bylo čerpáno z [1].

1.2.6 Klasifikace rozhodovacím stromem

Princip klasifikátoru *Decision Tree Classification* je stejný jako u stejnojmenné regrese. Algoritmus rozdělí trénovací data na třídní oblasti a pro data nová určí třídu na základě těchto oblastí.

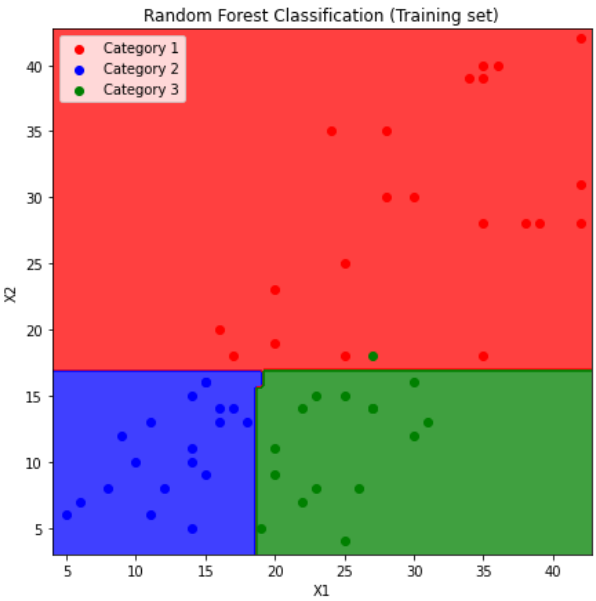


Obrázek 11 - Ukázka rozdělení oblastí pomocí rozhodovacím stromem

V rámci této kapitoly bylo čerpáno z [1].

1.2.7 Klasifikace náhodným lesem

Random Forest Classification. Opět i zde je princip stejný jako u stejnojmenné regrese. Jedná se o složení více modelů. Tomuto využití skládání více modelů pro vytvoření sofistikovanějšího modelu se říká „Ensemble Learning“. I zde se náhodně určují čísla četností bodů, ze kterých se vytvoří modely rozhodovacích stromů. Všechny tyto modely klasifikují nová data a nejčetnější odhad je pak výsledkem.



Obrázek 12 - Ukázka klasifikace náhodným lesem

Na obrázku 12 je vidět, že i když se jedná o stejná data jako v předchozí kapitole a jedná se o klasifikátory založené na algoritmech rozhodovacích stromů, tak se výsledky liší. Obecně robustnější klasifikátor náhodným lesem určil třídní oblasti hůře (chyba zelené tečky v červené oblasti). To je způsobeno hlavně malým množstvím dat. Oba klasifikátory potřebují pro lepší výsledky větší množství dat (stejně jako tomu bylo u regrese).

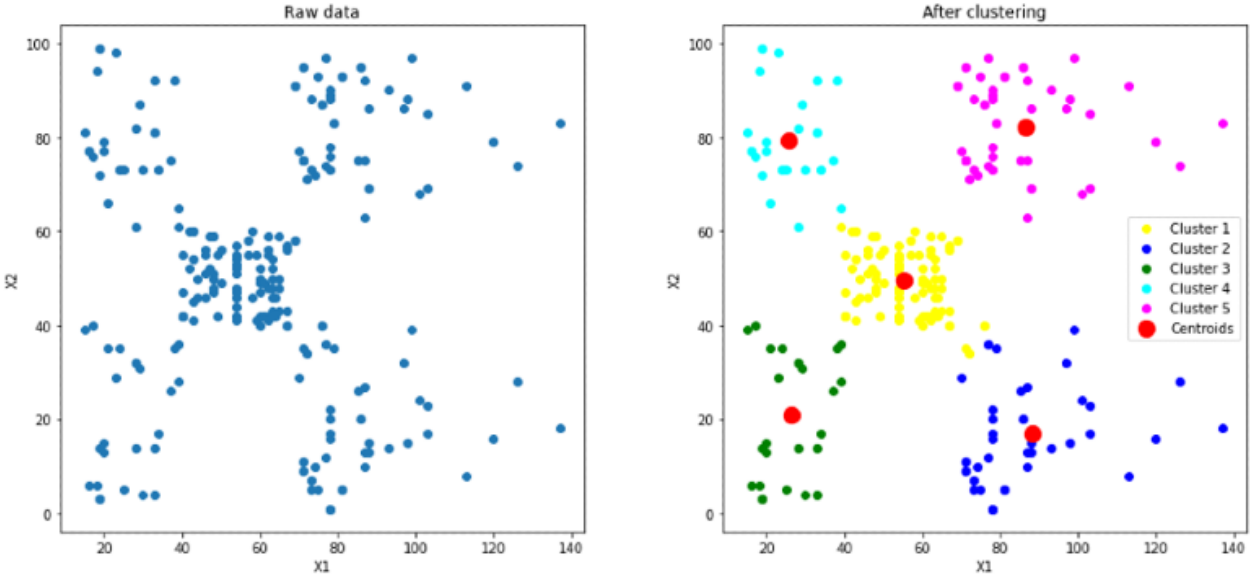
V rámci této kapitoly bylo čerpáno z [1].

1.3 Klastrování

Dalším nástrojem strojového učení je *Clustering* čili klastrování nebo shlukování. Nejedná se nyní o žádné modely sloužící k predikcím. Zde je účelem zkoumání dat, nacházet v datech nějaké vzory či závislosti (shluky). V této kapitole budou představeny 2 metody: klastrování K-průměrů a hierarchické klastrování. Výsledky tohoto procesu mohou být využity například pro trénování klasifikátorů.

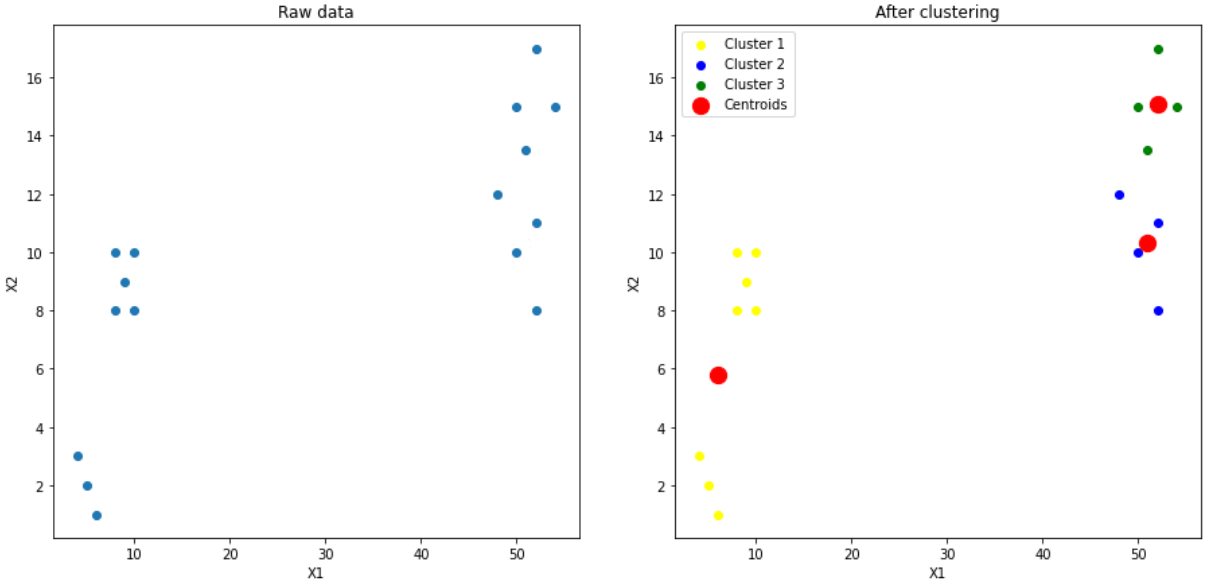
1.3.1 Klastrování k-průměrů

Anglicky *K-Means Clustering* je metoda založená na shlukování dat kolem těžišť. Jedná se o iterační metodu, kde se na začátku trénování vytvoří tolik těžišť kolik klastrů míníme vytvořit. Každý bod se přisoudí nebližšímu těžišti, tím se vytvoří klastry první iterace. Následně se v každém klastru přepočítají těžiště. Díky tomu se těžiště posunou více do středů jednotlivých klastrů. Poté opět probíhá přiřazování každého bodu k nově vytvořeným těžištím. Pokud nějaký bod změní klastr, tak iterování probíhá dál.



Obrázek 13 - Ukázka K-Means klastrování

Obrázek 13 ukazuje výsledek vpravo oproti vstupu vlevo. V grafu výsledku jsou zobrazeny pro názornost i jejich výsledná těžiště, ke kterým byly klastry přiřazeny. Těžiště ale nejsou reálnými daty nýbrž pouhým pomocným produktem algoritmu. Celý postup tohoto typu klastrování je ale bohužel závislé na počátečním určení těžišť. Špatným počátečním určením může nastat situace zvaná jako „Random initialization trap“ neboli past náhodné inicializace. To znamená, že iterační postup dokonverguje k řešení, zastaví se, ale shluky jsou určeny špatně (viz obrázek 14).



Obrázek 14 - Ukázka pasti náhodné inicializace

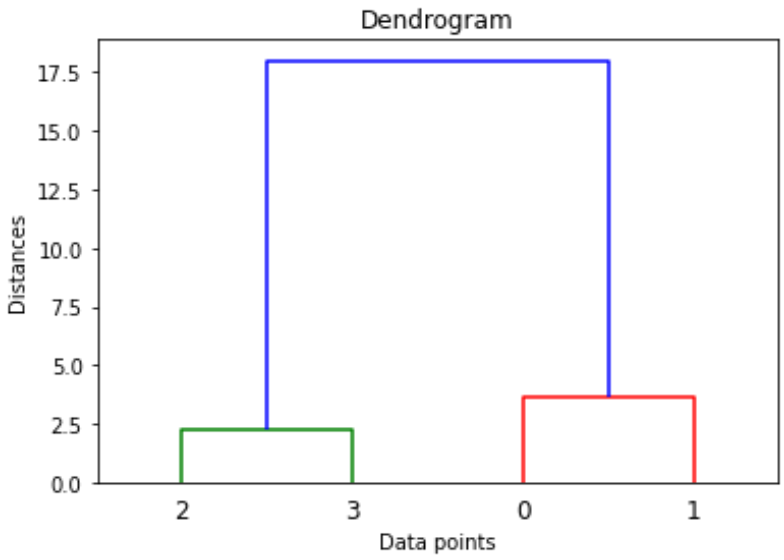
To je vyřešeno implementací modifikace algoritmu zvané *K-Means++*, která těžiště neurčuje náhodně, ale sofistikovaným způsobem je co nejvhodněji vybírá.

V rámci této kapitoly bylo čerpáno z [1].

1.3.2 Aglomerativní hierarchické klastrování

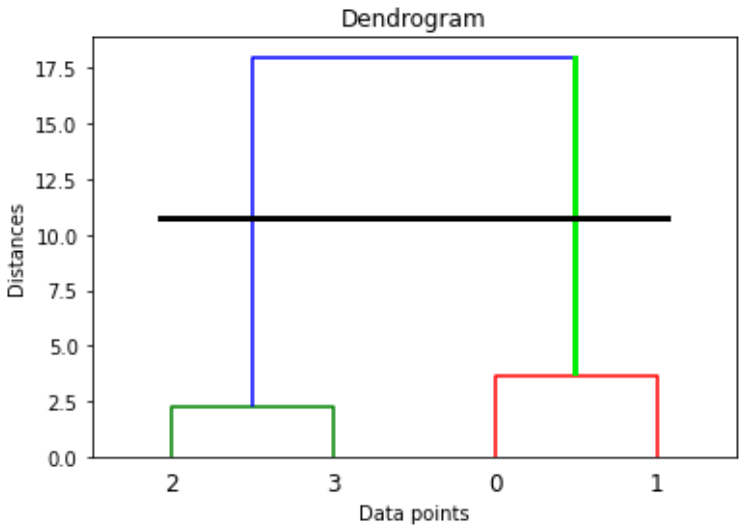
Existují 2 typy hierarchických klastrovacích algoritmů: aglomerativní („agglomerative"), rozdělující („divisive"). Aglomerativní přístup využívá bottom-top postup, tedy postupné shlukování klastrů do větších celků. Rozdělující typ na to jde opačně. Pozornost bude věnována prvnímu zmíněnému.

Agglomerative Hierarchical Clustering čili aglomerativní hierarchické klastrování na začátku všechna data zvolí jako klastry. Poté postupně sdružuje nebližší dvojice do klastrů nových až zbude jen jeden, který obsahuje všechna data. Je samozřejmě nutné určit jaký typ vzdálenosti se bude počítat. Celý proces vizualizuje tzv. dendrogram. To je graf zobrazující postupné změny celého procesu.



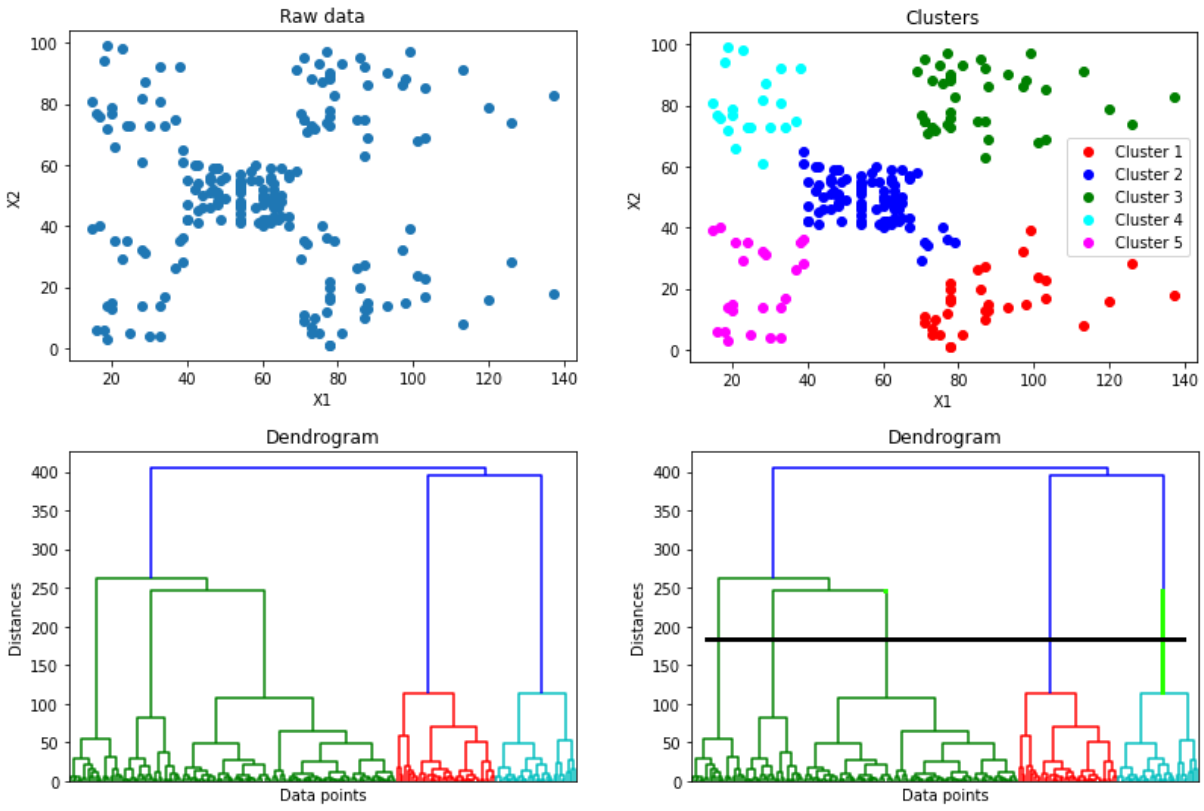
Obrázek 15 - Ukázka dendrogramu

Bude dobré vysvětlit, jak dendrogram funguje. Na obrázku 15 je jednoduchý příklad. Na ose x jsou rozloženy všechny body (pozorování), osa y pak přísluší velikosti vzdáleností. Jednotlivé „vidlice" zobrazují spojení klastrů. Výška vidlice znázorňuje vzdálenost těchto dvou spojených klastrů. Teď ale nastává otázka, jak určit správné množství klastrů? Klastry by měly být množiny bodů zjevně rozlišitelné. To bychom vizualizací poznali tak, že shluky budou znatelně daleko od sebe (jak nám to intuice napovídá). Tedy nás zajímá délka vidlic. Jeden ze standardních přístupů, jak určit počet klastrů z dendrogramu je, že se najde nejdelší vidlice nebo její část taková, která nekříží žádnou pomyslně protaženou horizontální linii zobrazující spojení dvou klastrů. Na obrázku 15 by to byla pravá noha modré vidlice. Přes celý graf pak protáhneme horizontální linii procházející středem této nalezené spojnice. Počet překřížených noh vidlic této vytvořené horizontální linie nám pak říká počet výsledných klastrů.



Obrázek 16 - Ukázka určení počtu klastrů z dendrogramu

Zde na obrázku 16 je světle zeleně označena nejdelší spojnice klastrů, která neprotíná žádnou horizontální spojnici. Jejím středem s pak vedla černá horizontální linie, ta kříží 2 linie, takže v tomto případě je počet nejvhodnějších klastrů 2.



Obrázek 17 - Ukázka případu s 5 klastry

Na obrázku 17 můžeme vidět rozsáhlejší případ. V dendrogramu byla nalezena nejdelší vzdálenost (světle zeleně) nepřekračující protažené horizontální spojnice klastrů, potom byla

opět vedena (černá) horizontální linie přes celý graf. Tato linie kříží celkově 5 linií. Z tohoto odhadu lze poté klastrovacímu algoritmu předat zjištěný parametr počtů klastrů. Výsledek je pak na obrázku možný vidět vpravo nahoře.

V rámci této kapitoly bylo čerpáno z [1].

1.4 Učení asociačních pravidel

Anglicky *Association Rule Learning*. Tento nástroj se v datech snaží hledat vazby. Pravidlem se tu myslí nějaká akce a tato akce může implikovat jinou akci. Například bychom si mohli představit filmový doporučující algoritmus. Platformy jako YouTube, Netflix či reklamy poskytované na webových stránkách, všechny využívají algoritmy asociačních pravidel za účelem jejich efektivní volby a účinného cílení na vhodné zákazníky. Kupříkladu když někdo zhlédne film *Ex Machina* (mysteriózní sci-fi s umělou inteligencí), je zde slušná šance, že by se mohl rád podívat i na film *I, Robot* (opět mysteriózní sci-fi s umělou inteligencí). Nebo koupí-li si zákazník laptop nějaké značky, je zde šance, že by si mohl chtít koupit i příslušenství či jiné produkty stejné značky a podle toho mu budou prezentovány na stránkách příslušné reklamy. Je zřejmé, že tento nástroj je velice hojně využívaný v obchodním a zábavním průmyslu. Lidé jsou totiž dobré generátory těchto asociačních pravidel.

1.4.1 Apriori

Základní tři složky apriori algoritmu jsou: *support*, *confidence* a *lift*. Každá složka napočítává statistickou pravděpodobnost určité vlastnosti jevu, u kterého zjišťujeme asociační pravidlo. Vysvětlování bude lepší na konkrétním příkladu.

Představme si velkoobchod. Nás budou zajímat položky nákupů všech zákazníků. Řekněme, že jsme tyto data sledovali po dobu jednoho dne. V obchodě se vystřídalo 1000 zákazníků a každý si domů odnášel libovolný nákup obsahující nenulový počet položek. Nástrojem učení se asociačních pravidel budeme sledovat, které potraviny jsou příčinou koupě jiných potravin. Máme tedy k dispozici data seznamů surovin, které zákazníci nakupovali. Vezměme si třeba položku pivo ***P***. *Support* složka by pak vypadala takto:

$$support(\boldsymbol{P}) = \frac{počet\ nákupů\ obsahující\ \boldsymbol{P}}{celkový\ počet\ nákupů}. \tag{1.9}$$

Dostáváme tedy pravděpodobnost, s jakou se položka piva může objevit v nákupech zákazníků. Následovat budeme složkou *confidence*. Zde přichází na scénu potenciální asociační pravidlo. Budeme tedy k pivu pozorovat i položku buřtů ***B***:

$$confidence(\boldsymbol{P} \rightarrow \boldsymbol{B}) = \frac{počet\ nákupů\ obsahující\ \boldsymbol{P}\ a\ \boldsymbol{B}}{celkový\ počet\ nákupů\ obsahující\ \boldsymbol{P}}. \tag{1.10}$$

Tímto tedy sledujeme pravděpodobnost, s jakou se mezi nákupy s pivem můžou objevovat buřty. Poslední zmíněnou částí algoritmu je složka *lift*:

$$lift(\boldsymbol{P} \rightarrow \boldsymbol{B}) = \frac{confidence(\boldsymbol{P} \rightarrow \boldsymbol{B})}{support(\boldsymbol{B})}. \tag{1.11}$$

Zde tedy porovnáváme pravděpodobnost buřtů mezi nákupy s pivem a pravděpodobnost buřtů mezi všemi nákupy. Zde je totiž snaha uvažovat korelaci jevů a tímto poměrem zjistíme, jestli se náš odhad zlepšil nebo ne. To je taky důvod názvu této metody, protože používáme nějakou apriorní informaci (zde, že kombinace piva a buřtů je oblíbená). Tento konkrétní poměr by znamenal koeficient změny pravděpodobnosti, kde při sledování nákupů buřtů místo všech nákupů budeme uvažovat jen ty, které obsahují piva, zdali zde není větší poměrová šance, nějakého pravděpodobnostního zlepšení (*lift* = zvednutí). To je hlavní složka algoritmu, kterou budeme pozorovat.

Aby ale algoritmus nepočítal všechny možné kombinace, musí se zde zavést podmínky. Nemusí se jednat totiž o asociační pravidlo 2 položek, ale za jevy ***P*** a ***B*** můžeme uvažovat celé množiny položek. Závislost třeba 8 položek implikující koupi 2 dalších položek (třeba nákupy podle veřejně oblíbených receptů), kde nejen množství ale i variabilita surovin opět přispívá k složitosti. Všechno to by mohlo pro velká data být výpočetně obrovsky složité. Na začátku je tedy třeba určit mezní hodnotu pro *support* a pro *confidence*. Budeme uvažovat pouze podmnožinu pravidel vyšší než určená mez *support*. Ta se může určit buď empiricky nebo zkoušením. Z této podmnožiny budeme uvažovat pravidla přesahující mez *confidence*. Výsledná podmnožina pravidel je pak seřazena sestupně podle *lift*. Výsledkem je pak sestupný seznam pravidel podle síly asociace.

V rámci této kapitoly bylo čerpáno z [1].

1.4.2 Eclat

Eclat algoritmus je základnější verze apriori algoritmu. Může dávat rozdílné výsledky. Zde se totiž uvažuje pouze položka *support*. I zde se určí mez, nad níž se budou uvažovat pravidla a celá množina se pak podle této složky *support* seřadí sestupně. Zopakujme, že za jev ***P*** (viz (1.9)), v této složce zkoumaný, můžeme uvažovat celou množinu objektů a aby bylo možné pomocí Eclat zkoumat asociační pravidla, je nutné, aby zde bylo 2 a více objektů. Jinak bychom počítali pouze pravděpodobnost výskytu jednoho objektu.

V rámci této kapitoly bylo čerpáno z [1].

1.5 Zpětnovazební učení

Anglicky *Reinforcement Learning*. Někdy může být potřeba výsledků analýzy vyžadována ihned během získávání dat a ne až po jejich nasbírání a následném analytickém postprocessingu. Tato potřeba může plynout z požadavku ekonomiky prostředků nebo z efektivity dosahování výsledků či úspory času nebo jenom z toho, že chceme, aby se algoritmus naučil sám pracovat s problémem. Je důležité si zde ale definovat, co je správně a co správně není, aby algoritmus věděl, na co má cílit a z čeho se učit. Tyto stavy můžeme jednoduše označit binárně: 1 – úspěch, 0 - neúspěch.

Dobrým příkladem je problém zvaný *The Multi-Armed Bandit Problem.* Problém se zabývá hracími automaty („slot machines“), které jsou jedním z nejrychlejších způsobů, jak přijít o peníze. Dříve tyto automaty měly ze strany páčku, za kterou hráč musel zatáhnout, pokud chtěl zahájit hru. Z toho důvodu se jim říká jednoruký bandita, protože páka představovala ruku a automat banditu, který měl jediný úkol, a to obrat hráče. Máme-li více automatů, jedná o mnohorukého banditu. Existují 2 stavy, hráč buď vyhraje (výsledek 1) nebo nevyhraje nic (výsledek 0). Předpokládejme, že automaty se nechovají stejně, že šance na výhru u nich se liší. Každý automat bude tedy mít své různé pravděpodobnostní rozdělení, a tedy má smysl uvažovat, že některý z automatů bude výdělečnější než ostatní. A přesně o to nám teď půjde. Do automatů budeme házet peníze, takže každý pokus nás bude něco stát. Nejsme žádní miliardáři a je tedy pro nás důležité zjistit, který automat je nejvýhodnější co nejdříve. Budeme se zabývat 2 metodami, které k řešení přistupují různým způsobem.

V rámci této kapitoly bylo čerpáno z [1].

1.5.1 Horní hranice spolehlivosti

Anglicky *Upper Confidence Bound* (UCB). Tato metoda na začátku samozřejmě uvažuje stejná rozdělení u všech automatů, která se pak posléze budou měnit. Tato rozdělení jsou charakterizována svojí střední hodnotou a intervalem spolehlivosti, tedy obecně shrnuto přibližnou potenciální akumulací výtěžku každého z automatů. Jen pro poznámku, interval spolehlivosti je rozsah hodnot, kde s určitou pravděpodobností očekáváme hodnoty a tato pravděpodobnost se určí hladinou spolehlivosti (často 95%). Hodnotami zde rozumíme výše odměn každého z automatů. My ale budeme uvažovat pouze interval spolehlivosti, tedy budeme odhadovat rozsah hodnot, které budeme získávat. Postupným testováním se bude odhadovaný interval spolehlivosti zužovat v závislosti na tom, jestli dostaneme výhru nebo ne. Jelikož se na začátku nic neví, tak jsou tyto intervaly rozsáhlé a jednoduše se zkusí všechny automaty pro první pozorování změn. Teď nastává otázka, co po prvním ozkoušení všech automatů dál a možností je víc. Můžeme se rozhodovat třeba podle výše střední hodnoty, ale my se budeme řídit jménem algoritmu - budou nás zajímat hodnoty horních hranic intervalů spolehlivosti automatů a budeme vždycky jako další volbu volit ten automat, který má tuto hranici nejvýše a to je vlastně celé. Snažíme se defacto uhádnout pravděpodobnostní rozdělení.

Teď k tomu, jak se tyto výpočty vlastně provádí. Začněme se stěžejní výpočetní částí:

$$\bar{r}_i(n) = \frac{R_i(n)}{N_i(n)} \tag{1.12}$$

n značí počet kol a i je index automatu. Jedním kolem nazýváme jeden hráčův pokus, tedy zatáhnutí za páku jednoho z automatů. $R_i(n)$ je suma odměn (suma všech 1) aktuálního automatu i až do aktuálního kola n. $N_i(n)$ je počet výběrů automatu i, tedy kolikrát do aktuálního kola hráč zatáhnul páku tohoto automatu. $\bar{r}_i(n)$ pak značí průměrnou výhru automatu i. Když máme střední hodnotu, tak pak následuje přepočítání intervalu spolehlivosti.

$$\Delta_i(n) = \sqrt{\frac{3}{2}\frac{log(n)}{N_i(n)}} \tag{1.13}$$

Interval spolehlivosti zde definujeme jako $[\bar{r}_i(n) - \Delta_i(n),\ \bar{r}_i(n) + \Delta_i(n)]$. Zde tedy počítáme hranice intervalu spolehlivosti. Jak už bylo řečeno, je snaha vybírat pro další pokus automat s nevyšší horní hranicí. Na začátku se tedy inicializují dostatečně vysoké hodnoty, aby každý automat měl nárok na jeden pokus a poté se hranice přepočítávají:

$$UCB = \bar{r}_i(n) + \Delta_i(n) \tag{1.14}$$

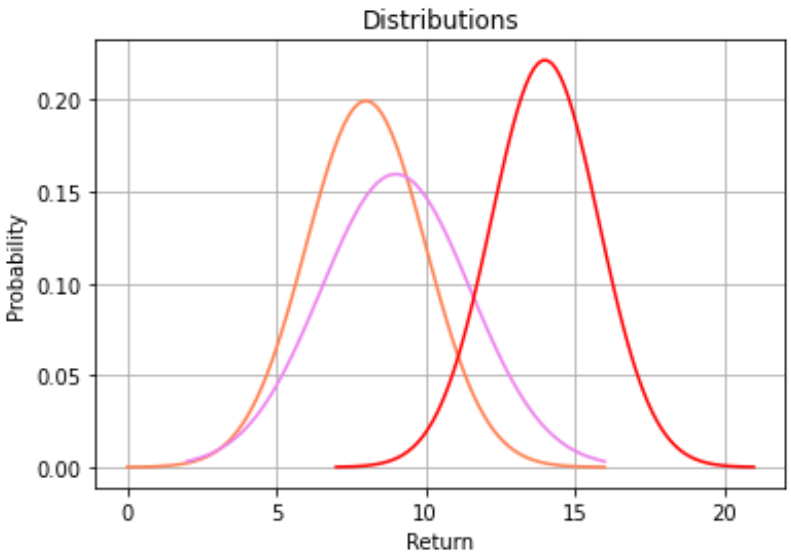
Na konci budou dle výše horních hranic všechna rozdělení setříděná a algoritmus bude vždy vybírat tu nejvyšší.

V rámci této kapitoly bylo čerpáno z [1].

1.5.2 Thompsonovo vzorkování

Thompson Sampling algoritmus k řešení nepřistupuje deterministicky, jako tomu bylo v případě UCB, kde šlo o určování rozdělení, ale probabilisticky. Tedy na základě principu pravděpodobnosti budeme volit naše rozhodnutí. Thompsonovým algoritmem budeme z dat vytvářet a upravovat rozdělení, což je rozdíl od UCB, kde se tohle nedělo. Na základě parametrů těchto rozdělení v každém z nich budeme generovat naše předpovědi a podle nich se rozhodovat. Samozřejmě je na nás, které rozdělení budeme uvažovat a měli bychom ho zvolit moudře, protože se podle toho bude řídit kvalita výsledku.

V prvním kroku algoritmus potřebuje pro první odhad parametrů rozdělení počáteční vzorky. Využijeme opět příkladu mnohorukého bandity. Na začátku tedy zkusíme několik pokusů na každém z automatů. Podle toho, kolikrát jsme vyhráli a nevyhráli, určíme první odhad rozdělení. Z těchto rozdělení necháme vygenerovat náhodnou hodnotu (výši zisku). Jako by to asi i udělal každý i algoritmus vybere automat, jehož rozdělení vygenerovalo nejvyšší odhad a toto je zásadní rozdíl a jiný přístup tohoto algoritmu než u UCB. Po tomto výběru tedy algoritmus zatáhnu za páku příslušného automatu a čeká co dostane. Adekvátně k této hodnotě se upraví rozdělení toho automatu a celý proces se opakuje. Tímto způsobem se můžou jednotlivá rozdělení dost změnit a statisticky bude převládat automat s nejvýhodnějším rozdělením.



Obrázek 18 - Příklad 3 možných rozdělení

Na obrázku 18 je možné vidět situaci se 3 automaty, kde se jedná o 3 různá (Gaussova) rozdělení, z nichž se budou náhodně generovat odhady pro další postup. Samozřejmě jde pouze o ilustraci a reálné problémy mohou být složitější a nemusí vykazovat normální rozdělení. Jelikož se jedná o probabilistický model, bude dávat pokaždé trochu jiné výsledky. Ovšem, bude-li dostatek dat, výsledek to neovlivní. Bude-li ale dat málo, tak zde i v případě UCB nemusí mít model dost času na naučení a výsledky nemusí být žádoucí.

V rámci této kapitoly bylo čerpáno z [1].

1.6 Zpracování Přirozeného Jazyka

Natural Language Processing (NLP) je nástrojem zabývající se celkově komunikací člověka s počítači. Může se jednat o problematiku převodu řeči na text, otázek a odpovědí či chatbotů (textová komunikace člověka s robotem), sumarizace textů, hledání důležitých pasáží v textu, měření úrovně sentimentu, interpretace textů apod. Existuje mnoho modelů a některé využívají vlastnosti hlubokého učení. Například *If/Else Rules* (chatbot), *Audio frequency components analysis* (rozpoznávání řeči), *CNN for text recognition* (klasifikace) a aktuálně nesilnějším modelem je *Seq2seq* model. Zde se budeme zabývat jenom jedním modelem nazývajícím se *Bag-of-words* model.

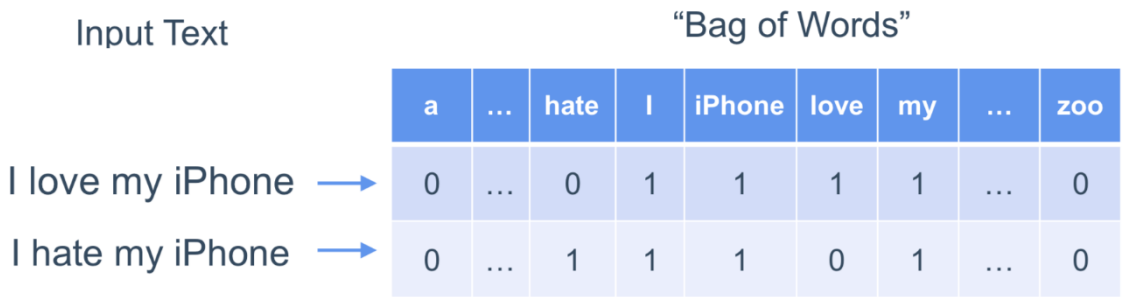
V rámci této kapitoly bylo čerpáno z [1].

1.6.1 Bag-of-words model

Mohli bychom říct model „Pytlík slov“. Toto pojmenování skutečně naznačuje princip fungování. Model, který trénujeme, je jakýmsi „pytlem“ obsahujícím hromadu slov, s kterými je schopný pracovat. Samozřejmě jsou ve skutečnosti uložené v nějaké struktuře. Využívá teorie, že průměrný člověk aktivně využívá v životě cca 20000 slov, přičemž normální konverzace dokážou být v 95% pokryty zásobou asi pouze 3000 slov.

Tento model slouží pro klasifikační typy problémů. Například chceme-li zjistit, jestli je zpráva pozitivní či negativní, jestli se jedná o zprávu odpovídací „ano“ nebo „ne“ na nějakou otázku a tak podobně. Velké využití může mít třeba pro hodnocení recenzí.

Například má-li obchod svoji webovou stránku, kde jejich zákazníci mohou psát recenze. Bylo by náročné všechny recenze procházet. Může jich být hodně, mohou přibývat každý den. Může tedy být nasnadě využít tento model, aby pozitivnost či negativnost těchto recenzí zhodnotil sám ve zlomku času. Ovšem je potřeba ho nejdříve vytrénovat daty, kde jsou ohodnocení recenzí známa.



Obrázek 19 - Ilustrace představy Bag-of-words modelu [11]

Bag-of-words model tedy disponuje vektorem známých slov, která mu na začátku dáme. Každá jeho souřadnice přísluší konkrétnímu slovu. Všechny recenze též představují vektory slov a tento model je vidí jako vektor nul s řídkou populací nenulových čísel představující počet slov na takových souřadnicích, na kterých je tento model má naučené. Tato čísla označují počet slov v dané recenzi. Představu je možné vidět na obrázku 19, kde každá věta (vlevo) je převedená na vektor histogramu slov (vpravo). Vektory mají pevnou strukturu, která se nemění. Je dobré do modelu nezahrnovat slova, která nám v klasifikaci nepomohou (tzv. „stopwords“). Například slovo „auto“ nám neřekne o pozitivitě žádnou informaci. Ale slova

jako „dobrý“, „otřesný“, „nebyl“, „je“, apod. už význam mají. Model pak jen pracuje s kombinacemi čísel, představující věty a trénováním se naučí odhadovat, které vektory by mohli naznačovat dobrou nebo špatnou recenzi. Výsledná klasifikace využívající tento model pak může být provedena vložením do modelu logistické regrese (NLP) anebo do modelu neuronové sítě (Deep NLP – DNLP).

V rámci této kapitoly bylo čerpáno z [1], [9] a [10].

1.7 Hluboké učení

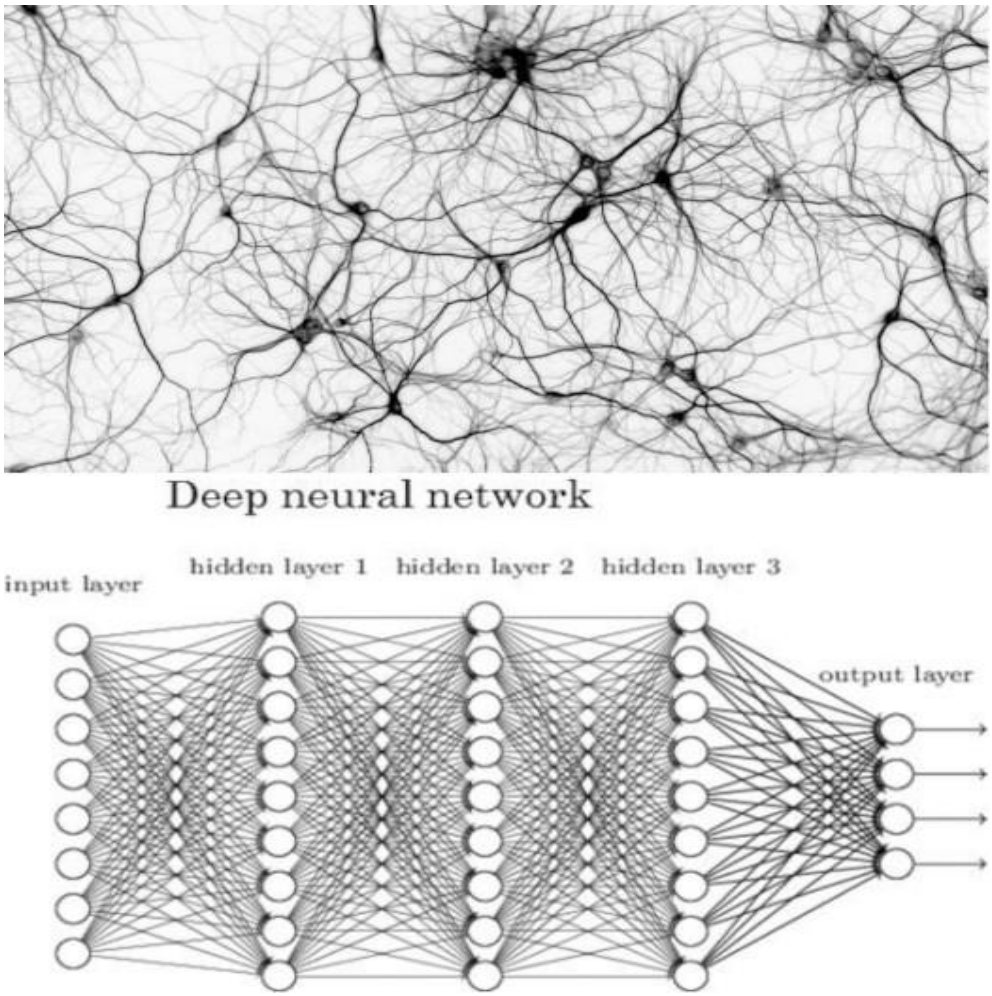
Neboli *Deep learning* je jedním z nejsilnějších nástrojů strojového učení. Jedná se podoblast klasického strojového učení. Vyznačuje se tím, s jakými daty je pracováno a jakým způsobem jsou modely trénovány. Klasické strojové učení pracuje se strukturovanými daty (třeba CSV tabulky). Tato data na počátku strukturována být nemusí, ale předzpracováním se pak vždycky uvedou do nějaké strukturované podoby. Hluboké učení dokáže pracovat i s nestrukturovanými daty jako třeba texty, obrázky, dokumenty. Druhým rozdílem je trénování. Hluboké učení využívá modelu tzv. neuronových sítí. Ty dokážou automatizovaně pracovat s parametry a sami mohou učinit extrakci nedůležitých či vytvořit jejich kombinace pro sofistikovanější korelace s výsledkem.

Myšlenka vytvoření neuronových sítí je tu cca od padesátých let minulého století. Plán sestrojit počítačový model pracující jako lidský mozek tehdy byl úspěšně vytvořen. Avšak byly zde technologické limity jednak výpočetní síly a taky paměťových kapacit tehdejších počítačů. Nebylo tak možné využít skutečný potenciál neuronových sítí jako prostředek hlubokého učení. Postupem času byl extrémní vývoj technologií příčinou, proč se teď neuronové sítě dostávají do trendu. Počítače dnes už disponují mnohem výkonnějšími procesory a paměťovými zařízeními rovněž. Hluboké učení je tedy dnes více než možné.

V rámci této kapitoly bylo čerpáno z [1], [12].

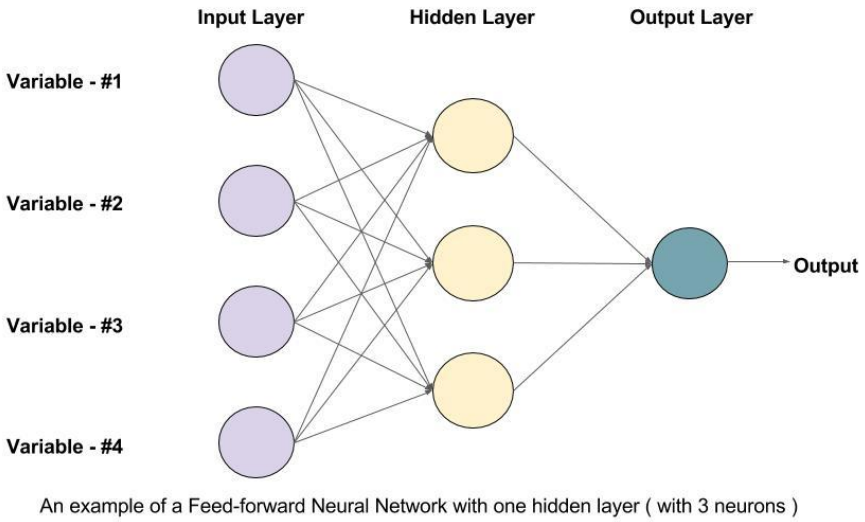
1.7.1 Umělé Neuronové Sítě

Anglicky *Artificial Neural Networks* (ANN). Název skutečně stojí za myšlenkou převedení principu fungování lidského mozku do počítače. Jako i v mozku, tak i v ANN jsou informace předávány základními jednotkami a to neurony spojenými synapsemi.



Obrázek 20 - Porovnání lidské neuronové sítě (nahoře) s ANN (dole) [13], [14]

Na obrázku 20 je možné vidět ilustrační porovnání skutečné a umělé neuronové sítě. Ukázány jsou zde už trochu rozsáhlejší sítě, takže si pro představení ukažme jednodušší model sítě.



Obrázek 21 - Příklad neuronové sítě s 1 skrytou vrstvou se 3 neurony (žlutě) [16]

Základní strukturou ANN jsou 3 složky: vstupní vrstva (*input layer*), skryté vrstvy (*hidden layer*) a výstupní vrstva (*output layer*). Ve vstupní vrstvě se nacházejí všechny vstupní parametry. Výstupní vrstva disponuje výsledky ať klasifikace či regrese. Skryté vrstvy (ano, může jich být víc) obsahují ony neurony předávající informace. Neurony jsou spojené synapsemi se vstupní i výstupní vrstvou (popř. s neurony v dalších skrytých vrstvách). Tyto synapse chápejme jako váhy udávající sílu vlivu hodnoty z neuronu, z kterého vychází. Na obrázku jsou sice neurony plně propojené se všemi proměnnými. To ale tak být nemusí. Síť si automaticky vytvoří hodnotu vah a může se stát, že některé synapse nebudou mít význam. Tyto váhy jsou zásadní a představují, jak se vlastně celá síť učí. Síť synapsí tedy přímo reprezentuje zkušenosti celé sítě. Každý neuron pak představuje funkci

$$y = \varphi\left(\sum_{i=1}^{n} \omega_i x_i + b\right), \tag{1.15}$$

která určí výstupní hodnotu y ze vstupních hodnot x_i a vah ω_i do toho neuronu vstupující. Parametr b určuje lineární posun a φ je tzv. aktivační funkce, která škáluje celkovou výstupní hodnotu y podle potřeby. Díky tomu může pak říct, jestli je neuron aktivovaný nebo v pasivním stavu (hodnota bude škálována k nule). Tyto aktivační funkce jsou různé a záleží na uživateli, jaké zvolí v závislosti na jeho potřebách nebo na potřebách řešené úlohy. Známé aktivační funkce jsou například skoková přenosová funkce (*Threshold function*), sigmoidální přenosová funkce (*Sigmoid function*) nebo funkce *Rectifier Linear Unit* (ReLU).

Teď, když už máme představu, jak neuronová síť vypadá nastává otázka, jak se učí? Jak už bylo zmíněno, tak synapse reprezentují zkušenosti (váhy) sítě. Tyto hodnoty se tedy učením musí měnit. Trénováním sítě se tedy hodnoty těchto vah postupně nastavují. Síť totiž na začátku ve výstupní vrstvě vrátí výstupní hodnotu. Ta se od skutečné hodnoty většinou bude lišit, tak se porovná prostřednictvím tzv. chybové (ztrátové) funkce (*Cost function*). Výstup této funkce nám jednoduše říká, jak velkou chybu jsme udělali v našem odhadu. Cílem je tedy hodnotu této chybové funkce minimalizovat. Chybových funkcí existuje víc a opět je předmětem výběru uživatele. Pokud máme skutečnou hodnotu k porovnávání s odhadem, tak se jedná o případ tzv. učení s učitelem. Jsou případy, kdy vektor závislých proměnných není dostupný, to pak jde o tzv. učení bez učitele. Proces, kdy se ze vstupních hodnot určuje odhadovaná hodnota, se nazývá dopředná propagace a proces, kdy se na základě chybové funkce upravují hodnoty vah, se nazývá zpětná propagace. Zpětná propagace síti předává zkušenosti prostřednictvím propagačních algoritmů.

V rámci této kapitoly bylo čerpáno z [1], [13], [14], [15], [16].

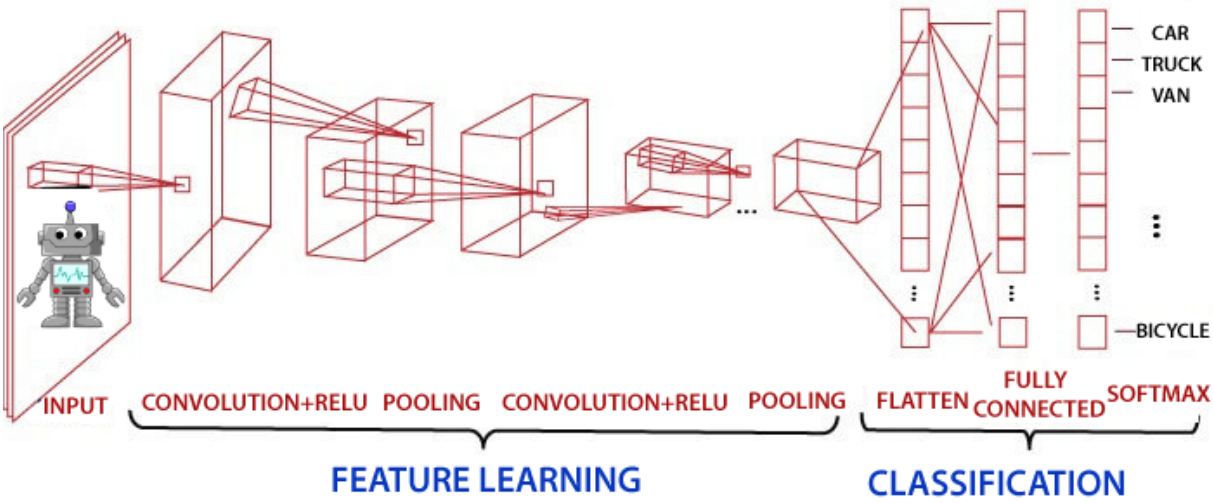
1.7.2 Konvoluční neuronové sítě

Convolutional Neural Networks (CNN) jsou sítě, které se využívají hlavně v oblasti strojového rozpoznávání obrazů. Hlavním rozdílem oproti ANN jsou 3 obrazy předzpracovávající složky algoritmu: konvoluce (*convolution*), sdružování (*pooling*) a zploštění (*flattening*). Výsledky těchto procesů je pak vektor nezávislých proměnných vstupující do ANN.

Konvoluce je proces, při kterém se na obrazová data jako matice čísel (hodnoty pixelů) aplikuje soubor tzv. příznakových detektorů (*feature detectors*). Jsou to matice čísel, které se porovnávají s obrazem pro zachycení důležitých prvků (tvary, vertikální hrany, barevné rozdíly apod.). Výstupem každého detektoru je příznaková mapa a výstupem celé konvoluce je pak konvoluční vrstva, což je soubor příznakových map.

Sdružování je opět aplikace matice čísel, jejíž výsledkem je poolingová vrstva, což je soubor sdružených příznakových map. Důvodů tohoto procesu je hned několik. Jednak je účelem tzv. prostorová invariance. Snažíme-li se totiž poznat na obrázku třeba opici, může tato opice být různě veliká, různě natočená, různě zdeformovaná, různě osvětlená či na různých pozicích v obrazu. Nejen toto poolingové masky (matice) řeší, ale také zabraňují overfittingu a redukují velikost obrázku. Záleží na velikosti masky a podle ní bude i výsledná vrstva více či méně zjednodušená.

Zploštění je proces převedení sdružených příznakových map na vektory čísel, jež budou následně vstupem do ANN jakožto vektory nezávislých proměnných.



Obrázek 22 - Znázornění postupu CNN [17]

Na obrázku 22 je celý postup CNN znázorněn. Jak je možné vidět, tak konvoluce a sdružování se mohou opakovat. Vpravo na obrázku je pak znázorněna neuronová síť klasifikující zpracovávaný obraz.

V rámci této kapitoly bylo čerpáno z [1], [17].

1.8 Evoluční algoritmy

Evolutionary Algorithms. Myšlenka se opírá o princip evoluce v přírodě. Evoluční algoritmy jsou typem optimalizační algoritmů. Celý proces pracuje s množinou řešení. Tu se snaží postupně vylepšovat, až dojde k optimalizovaným hodnotám. To řeší na základě tzv. fitness funkce, která každému řešení přiřadí číslo udávající kvalitu řešení. Množinu řešení nazýváme populace, každé partikulární řešení je nazýváno jedincem obsahující chromozom (či genom) a každý chromozom představuje vektor řešení v podobě pole čísel. Struktura běhu algoritmu se skládá z několika základních kroků: **inicializace**, **selekce**, **genetické operátory** a **ukončení**, přičemž kroky selekce a genetických operátorů se opakují a každá jejich iterace se nazývá generací.

Je potřeba algoritmus na začátku nějak začít, a proto se při **inicializaci** vygeneruje náhodná populace. Tato populace by měla splňovat dané podmínky úlohy a může být vytvořena na základě nějakého prvotního odhadu řešení.

Selekce pak za využití fitness funkce vybere nejvhodnější jedince, kteří budou posléze podrobeni genetickým operátorům. Je důležité, jaká fitness funkce se vytvoří, protože její hodnocení jedinců bude mít zásadní vliv na celkový výsledek úlohy. Vytvoření této funkce nemusí být vůbec snadnou záležitostí.

Nyní nastává chvíle, kdy dochází k evoluci. Na vybraných jedincích budou aplikovány **genetické operátory** upravující jejich chromozomy. Tyto operátory jsou nejčastěji mutace (*mutation*) a křížení (*crossover*). Křížení 2 a více jedinců způsobí, že si navzájem vymění jejich části chromozomů, tím dojde ke kombinaci jejich řešení. Samo křížení by do řešení nepřineslo evoluci a po určitém počtu iterací by nedocházelo k žádným novým změnám, a tedy žádnému dalšímu vývoji. Proto na scénu přichází mutace náhodně mutující jedince prostřednictvím náhodné změny v chromozomu. Náhodné změny jsou obecně velice silný heuristický nástroj k nacházení řešení. Toho optimalizace evolučních algoritmů využívá.

Selekce, mutace a křížení se opakují tak dlouho, dokud algoritmus nepozná, že má být konec. Obyčejně to pozná dvěma způsoby. **Ukončení** nastane buď tehdy, když se nalezne nejzpůsobilejší (*fittest*) jedinec anebo po dokončení určitého počtu generací.

Použití těchto algoritmů je mnoho. Ve velké míře se jedná o učení chování umělé inteligence ve videohrách a mnohé další.

V rámci této kapitoly bylo čerpáno z [18], [19], [20].

2 Knihovny

Dnes už se většinou nepředpokládá, že by si programátor kód pro strojové učení napsal od základů sám. Existuje poměrně hodně knihoven obsahující právě tyto nástroje volně k použití. Zde budou zmíněny některé nejznámější.

Scikit learn je jednou z nejsilnějších a nerobustnějších knihoven programovacího jazyka Python. Byla vytvořena převážně v Python na knihovnách NumPy, SciPy a Matplotlib. Obsahuje řadu regresních, klasifikačních a klastrovacích algoritmů. Všechny algoritmy, které byly prezentovány v této práci (kapitoly 1.1, 1.2 a 1.3) tato knihovna obsahuje. Nabízí i předzpracovávající nástroje, složené metody (např. účinnou *Gradient Boosting* metodu či jiné) a mnoho další. Knihovna včetně API je dostupná ze zdroje [3].

Tensorflow je knihovna vytvořená společností Google, jejíž síla spočívá v modelech hlubokého učení. Využití má však napříč celým spektrem strojového učení. Využití má hlavně v jazyku Python, ale též v Javascript, C++ a v jazyku Java. Dostupnost na [22].

Keras je knihovna vyvinutá na knihovně Tensorflow. Jedná se o její high-level programovací verzi. Představuje API pro Tensorflow k ještě rychlejší a jednodušší práci s testováním neuronových sítí. Využít ji lze v jazyce Python. Dostupnost na [24].

V rámci této kapitoly bylo čerpáno z [21], [23], [25] a [26].

3 Webové služby

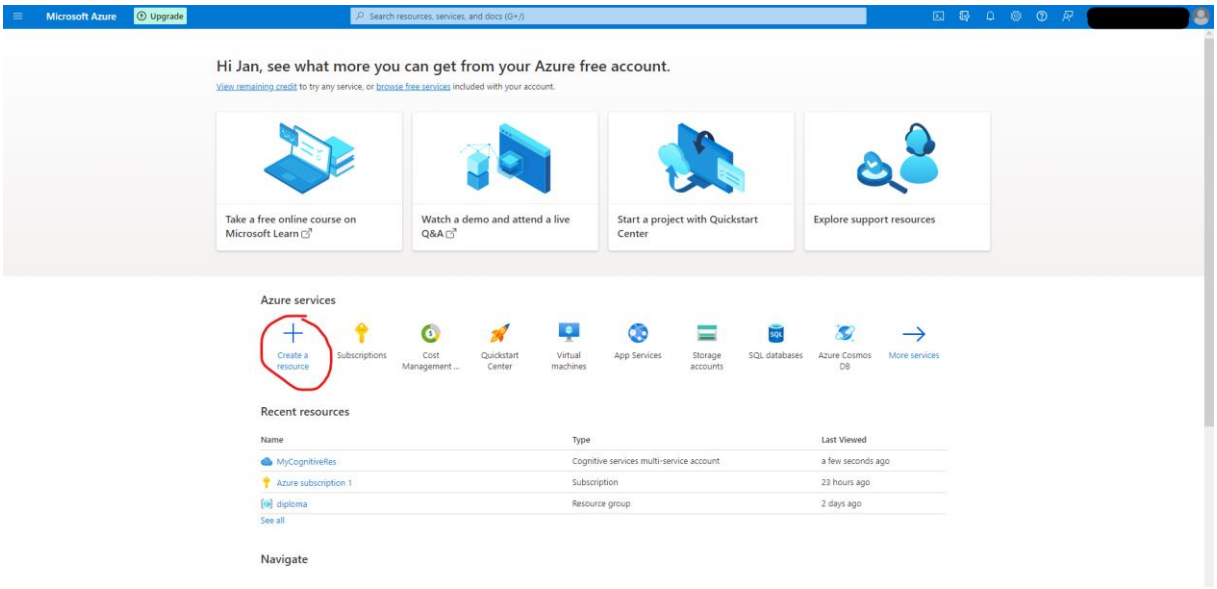
Webové služby jsou síťově využitelné prostředky pro širokou veřejnost uživatelů. Nabízí rentabilní servery s dostupnými úložišti, výpočetní sílou či rozličné algoritmické nástroje. V oblasti obrazových klasifikací či analýz sentimentu textů mohou ML servery poskytovat pro implementace už vytvořené a vytrénované modely, což může být obrovskou výhodou a ušetření práce i prostředků. Mezi 2 nejrozšířenější webové služby (aspoň pro oblast AI a ML) patří Amazon Web Service (AWS) [29] od společnosti Amazon a Azure od Microsoftu [28].

4 Využití služby Microsoft Azure

Dnes už asi skoro každý člověk přišel do styku s nějakým produktem od Microsoft. Mohl to být operační systém Windows, mohly to být editační nástroje v podobě Word nebo Excel, či po rozšíření nákazy COVID-19 zpopularizovaný Microsoft Teams pro online komunikaci, ať už v podobě videokonferencí nebo jinak. Zde se zaměříme na produkt Microsoft Azure, jehož využití je pro téma této práce atraktivní.

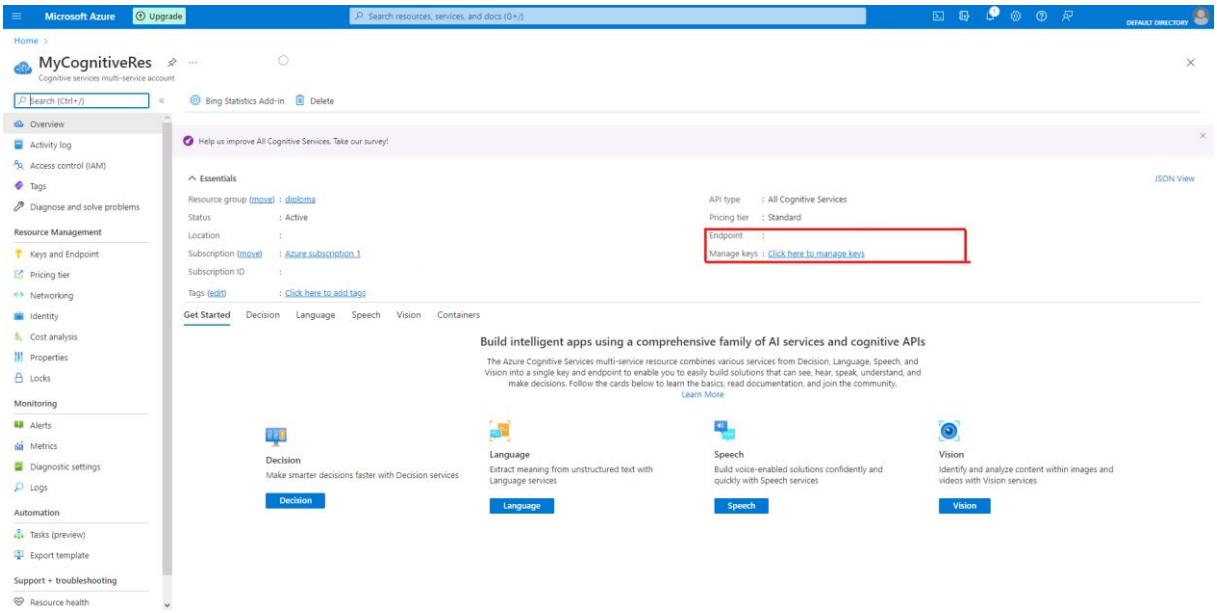
4.1 Azure portál a vytváření prostředku kognitivních služeb

Azure je cloudová platforma obsahující více než 200 produktů pro vytváření a udržování aplikací či serverů. Jsou zde nástroje pro úpravu a správu webů, pro datové analýzy a mnoho dalšího. Pro účely této práce nás bude zajímat produkt *Azure Cognitive Services* tedy kognitivní služby sloužící pro práci s řečí, jazyky, textem, viděním či počítačovým rozhodováním. Pro využívání Azure je potřeba založit si účet (*Azure subscription*). Po jeho založení je pak potřeba pro přístup ke konkrétní službě vytvořit prostředek (*resource*).



Obrázek 23 - Ukázka přihlášení v portálu Azure a vytváření prostředku [31]

Poté bude následovat série formulářů k vyplnění, kde je potřeba uvést konkrétně, k jaké službě chce uživatel přístup a další věci. Po jeho vyplnění je pak uživateli zprostředkován přístup ke službě prostřednictvím klíče a koncového bodu (*endpoint*). Pomocí těchto dvou údajů je pak možné se ke službě připojit. Pro zjištění těchto informací je potřeba rozbalit menu v levém horním rohu úvodní obrazovky portálu (viz obrázek 23), vybrat záložku "všechny prostředky" a poté vybrat konkrétní prostředek.



Obrázek 24 - Ukázka prostředí s informacemi o prostředku (údaje jsou skryty) [31]

Na obrázku 24 je červeně označeno, odkud se dají informace získat. Teď, když máme k dispozici nezbytné údaje k vytváření klientů z knihoven této služby, je potřeba nejprve si knihovny do programovacího prostředí nainstalovat.

4.2 Připravení prostředí Eclipse

Pro ukázku kognitivních služeb bude využíváno jazyka Java a vývojového prostředí Eclipse. Než bude vůbec možné vytvořit klienta služby pro připojení je nutné nejprve si nainstalovat Java Development Kit verze 8 a vyšší. Poté je potřeba naistalovat FaceAPI knihovnu a knihovny, na kterých je FaceAPI závislé (formáty JAR). Všechny tyto knihovny jsou dostupné z repozitáře na Github ze zdroje [32]. Tyto knihovny je pak potřeba přidat do domovského adresáře projektu do nově vytvořené složky pojmenované “lib”. Jednotlivé požadavky jsou pohromadě vypsány v repozitáři na Github. Po úspěšném nainstalování knihovny FaceAPI a jejích závislostí můžeme přistoupit k jejímu představení a ukázce.

4.3 FaceAPI

FaceAPI je jednou z nabízených implementací obrazového rozpoznávání kognitivních služeb. Zde jde o detekci obličejů na snímcích. Knihovna umí detekovat obličeje a jejich pozici, rozpoznávat na nich emoce a části obličeje (oči, nos apod.), rozlišovat různé skupiny obličejů na obrázku (třeba na nějaké koláži, kde se tváře opakují), identifikovat naučené tváře nebo zjišťovat, zda dva obličeje přísluší jedné osobě. Je ovšem potřeba, aby tváře byly dostatečně veliké, dostatečně barevně rozpoznatelné a ne v extrémní pozici jako například z profilu. Tvář z profilu (z boku) by nemusel rozeznat. Bude zde představena implementace této služby projektu prostředí Eclipse v jazyce Java počínajíc vytvořením klienta, pokračujíc nastavováním volitelných parametrů vrácených službou a samotnou detekcí společně s manipulací s vrácenými výsledky a na konec budou uvedeny příklady procesovaných obrázků.

4.3.1 Vytvoření klienta

Po samozřejmém importu knihoven je tedy na začátku nutné nejprve vytvořit klienta faceAPI. Je to krok, kde dochází k autentizaci uživatele. Budou potřeba údaje, které jsme dostali při vytvoření prostředku této služby. Půjde o klíč a region, pro který jsme prostředek vytvořili. Ukázky importu knihoven pro službu Azure a autentizaci jsou zde:

```
import com.microsoft.azure.cognitiveservices.vision.faceapi.*;
import com.microsoft.azure.cognitiveservices.vision.faceapi.models.*;
```

Obrázek 25 - Import knihoven služeb Azure

```
//Autentizace
final String KEY = "                                    ";
final AzureRegions REGION = AzureRegions.WESTEUROPE;

//Vytváření face klienta
FaceAPI client = FaceAPIManager.authenticate(REGION, KEY);
```

Obrázek 26 - Autentizace a vytvoření klienta služby Azure

Po úspěšném vytvoření klienta nyní lze využívat možnosti, které nám knihovna FaceAPI nabízí. Můžeme tedy přistoupit k detekci tváří. Než to ale bude učiněno, je potřeba si nejprve sestavit objekt volitelných parametrů, kterému je nutno říct, které parametry chceme vracet, protože ve výchozím stavu vrací pouze pozici tváří na obrázku.

4.3.2 Nastavování parametrů vracených službou

Funkce pro detekci obličejů (bude zmíněno později) ve výchozím nastavení pouze detekuje pozici obličejů (hodnoty okrajů ohraničujících obdélníků). Pro vrácení více informací je potřeba funkci pro detekci předat celý objekt představující vstupní parametry.

```
//Vytváření objektu pro volitelné výstupy pro nastavování požadovaných výstupů
DetectWithStreamOptionalParameter optionalParams = new DetectWithStreamOptionalParameter();

//Nastavení vracení id tváří
optionalParams.withReturnFaceId(true);

//Vytváření seznamu atributů pro vrácení (věk, emoce, pohlaví, úsměv, brýle)
List<FaceAttributeType> attrList = Arrays.asList(FaceAttributeType.AGE,
                                                 FaceAttributeType.EMOTION,
                                                 FaceAttributeType.GENDER,
                                                 FaceAttributeType.SMILE,
                                                 FaceAttributeType.GLASSES);
//Nastavení vracení seznamu atributů
optionalParams.withReturnFaceAttributes(attrList);

//Nastavení vracení obličejových bodů
optionalParams.withReturnFaceLandmarks(true);
```

Obrázek 27 - Vytvoření a nastavení objektu volitelných parametrů

Na začátku byl tedy nejprve vytvořen objekt *DetectWithStreamOptionalParameter*. Tomuto objektu byly poté postupně předány požadavky na vrácení id tváří, věku, emocí, pohlaví, úsměvů, jestli tváře mají brýle a další. Nyní, když je objekt s volitelnými parametry sestaven, můžeme pokračovat dále a zavolat metodu pro detekci tváří, které tento objekt předáme jako parametr.

4.3.3 Zpracování obrázku a přistupování k výsledkům

Pro získání výsledků je potřeba zavolat službu, která vstupní obrázky zpracuje a vyhodnotí.

```
//Použití Azure služby pro detekování tváří na obrázku (vrátí objekt DetectedFace)
List<DetectedFace> facesList = client.faces().detectWithStream(byteDatafile, streamOptionalParams);
List<DetectedFace> facesListUrl = client.faces().detectWithUrl(URL, urlOptionalParams);
```

Obrázek 28 - Volání služby Azure faceAPI pro zpracování obrázku (soubor x URL)

Na obrázku 28 jsou uvedeny dvě možnosti zpracování. Obrázek se dá předat buď URL adresou nebo souborem ve formátu bytového pole. Je nutné si povšimnout, že vstupní parametr funkcí obsahující objekt volitelných parametrů je nutné vytvořit příslušně pro volající funkci. Pokud zpracováváme obrázek pomocí URL adresy, je nutné mu předat URL-objekt pro volitelné parametry. Vrácen bude seznam detekovaných tváří ve formě seznamu objektů typu *DetectedFace*. Poté stačí už jen tento seznam procházet a metodami volat kýžené výsledky.

```
//Vytváření seznamu id tváří, objektu atributů a obdelníků pozic obličejů
List<UUID> faceUuids = new ArrayList<>();
FaceAttributes faceAttr = new FaceAttributes();
FaceRectangle faceRect = new FaceRectangle();

//Vypisování informací a kreslení obdélníků kolem detekovaných tváří (každé je přiřazeno jedno faceId)
for (DetectedFace face : facesList) {
    //Vypsání id tváře
    faceUuids.add(face.faceId());
    System.out.println(face.faceId());

    //Vytvoření objektu obličejových atributů a obdélníků z objektu face (což jsou detekované tváře)
    faceAttr = face.faceAttributes();
    faceRect = face.faceRectangle();

    //Vypisování informací
    System.out.println("Age: " + faceAttr.age());
    System.out.println("Gender: " + faceAttr.gender());
    System.out.println("===========Face emotions==========");
    System.out.println("Smile: " + faceAttr.smile());
    System.out.println("Happiness: " + faceAttr.emotion().happiness());
    System.out.println("Anger: " + faceAttr.emotion().anger());
    System.out.println("Neutral: " + faceAttr.emotion().neutral());
    System.out.println("Sadness: " + faceAttr.emotion().sadness());
    System.out.println("Surprise: " + faceAttr.emotion().surprise());
    System.out.println("Fear: " + faceAttr.emotion().fear());
    System.out.println("Disgust: " + faceAttr.emotion().disgust());
    System.out.println("Contempt: " + faceAttr.emotion().contempt());
    System.out.println("===========Face accessories==========");
    System.out.println("Accessories: " + faceAttr.accessories());
    System.out.println("Glasses: " + faceAttr.glasses());
    System.out.println("===========Face rectangle==========");
    System.out.println("Height: " + faceRect.height());
    System.out.println("Width: " + faceRect.width());
    System.out.println("Left: " + faceRect.left());
    System.out.println("Top: " + faceRect.top());

    //Vykreslení obdélníku kolem detekované tváře
    g2d.drawRect(faceRect.left(), faceRect.top(), faceRect.width(), faceRect.height());
}
```

Obrázek 29 - Přistupování k výsledkům a jejich zobrazování

Jak je na obrázku 29 vidět, výsledky jsou obsaženy v zásadě ve 3 objektech: faceUuids (seznam id), *faceAttr* (objekt atributů), *faceRect* (objekt pozice tváře). Jak už bylo zmíněno, služba dokáže odhadovat emoce. Ty jsou obsažené právě v objektu atributů. Na konci obrázku je pak využito objektu *Graphics2D* pro vykreslení ohraničujícího obdélníku kolem tváře. Obrázek s obdélníky je pak na konci jednoduše uložen do nového souboru.

```
//Uložení vstupního obrázku s obdélníky kolem tváří
File outputfile = new File("saved.jpg");
ImageIO.write(img, "JPG", outputfile);
```

Obrázek 30 - Uložení obrázku do nového souboru

Pro podrobnější informace o dalších možnostech této knihovny FaceAPI je doporučeno navštívit její dokumentaci (zdroj [4]). Nyní budou představeny výsledky této služby. Budou to jak označené tváře, tak i ukázka dalších možností, které jsou službou nabízeny.

4.3.4 Příklady výsledků FaceAPI

Samozřejmě zde uvedu názorné ukázky využití těchto služeb. Pro příklad zde uvedu fotografie z mých cest a výsledky Azure služby s tím, jak si poradila s detekcí mých tváří.

Obrázek 31 - Ukázka detekce fotografií s jedním obličejem

Na obrázku 31 je vytvořena koláž výstupů. Jedná se tedy o postupné procesování fotografií. Se všemi čtyřmi fotografiemi si služba poradila dobře. Dokázala i obtížnější příklad, kde jsem na beachvolejbalovém kurtu. Zde jsem trochu vzdálen a můj obličej je trochu z boku. Pro představu, jakým způsobem vrací služba další informace, bude ukázán příklad výstupu z kódu této práce.

```
Age: 27.0
Gender: male
===========Face emotions==========
Smile: 0.96
Happiness: 0.96
Anger: 0.0
Neutral: 0.034
Sadness: 0.004
Surprise: 0.0
Fear: 0.0
Disgust: 0.0
Contempt: 0.002
===========Face accessories=======
Accessories: null
Glasses: noGlasses
===========Face rectangle=========
Height: 444
Width: 444
Left: 176
Top: 346
```

```
Age: 26.0
Gender: female
===========Face emotions==========
Smile: 0.908
Happiness: 0.908
Anger: 0.006
Neutral: 0.07
Sadness: 0.002
Surprise: 0.0
Fear: 0.0
Disgust: 0.006
Contempt: 0.008
===========Face accessories=======
Accessories: null
Glasses: noGlasses
===========Face rectangle=========
Height: 235
Width: 235
Left: 89
Top: 469
```

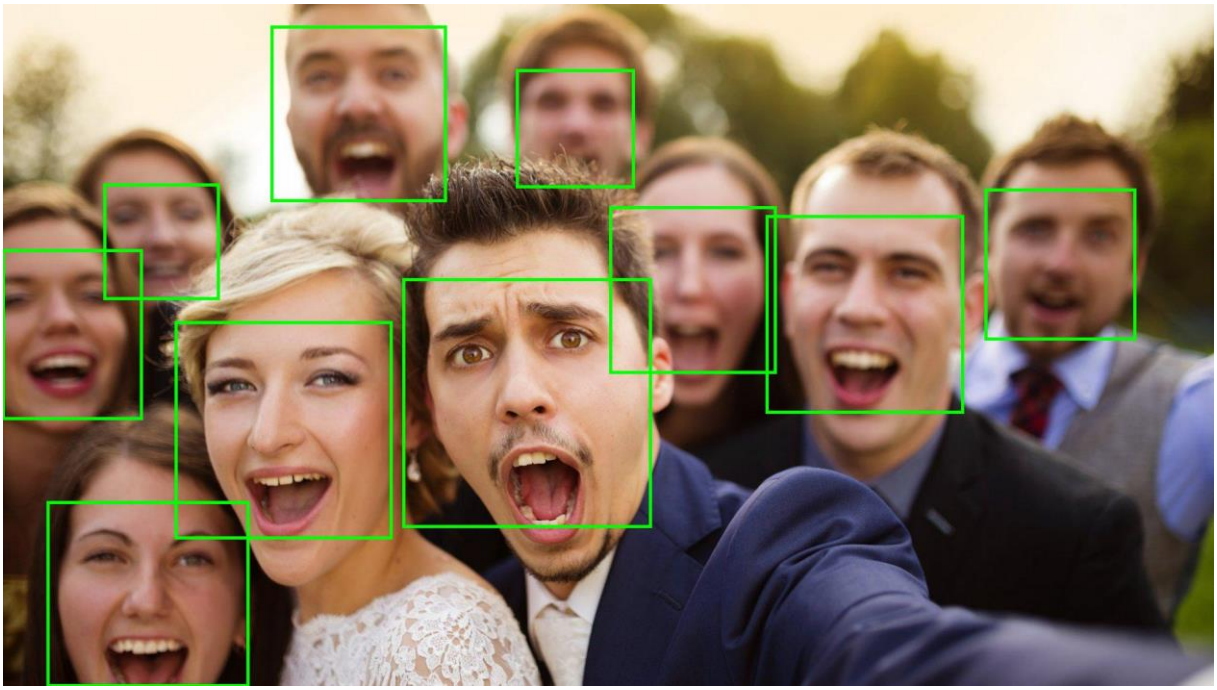
```
Age: 26.0
Gender: male
===========Face emotions==========
Smile: 0.845
Happiness: 0.845
Anger: 0.003
Neutral: 0.078
Sadness: 0.004
Surprise: 0.068
Fear: 0.001
Disgust: 0.001
Contempt: 0.001
===========Face accessories=======
Accessories: null
Glasses: noGlasses
===========Face rectangle=========
Height: 73
Width: 73
Left: 611
Top: 283
```

```
Age: 29.0
Gender: male
===========Face emotions==========
Smile: 0.787
Happiness: 0.787
Anger: 0.0
Neutral: 0.205
Sadness: 0.0
Surprise: 0.0
Fear: 0.0
Disgust: 0.0
Contempt: 0.008
===========Face accessories=======
Accessories: null
Glasses: noGlasses
===========Face rectangle=========
Height: 43
Width: 43
Left: 246
Top: 394
```

Obrázek 32 - Ukázka výstupů dalších informací o fotkách

Jak už bylo zmíněno, Azure dokáže detekovat i emoce. Každé emoci přiřadí číslo pravděpodobnosti, s jakou si myslí, že daná emoce přísluší obličeji. Na obrázku 31 každé tváři přiřadil emoci radost. Odhadnuté parametry nemusí být zcela přesné. Třeba odhad mého věku se lehce liší (cca o 5 let). Je ale možné, že postupem času budou modely sofistikovanější a trénovanější, a tedy velice přesné.

Na fotografiích, kde je pouze jeden člověk, služba fungovala dobře, ale jak si služba poradí, když by na jednom obrázku bylo více tváří?



Obrázek 33 - Ukázka případu s více tvářemi - zdroj procesovaného obrázku [33]

S více tvářemi neměla služba žádný problém. Dokonce i tváře v pozadí, které jsou lehce rozmazané rozpoznala dobře.

4.4 Zhodnocení

Toto byla ukázka jedné z kognitivních služeb Azure. Jednalo se už o vytrénovaný model sloužící výhradně k rozpoznávání tváří. Výhodou už vytrénovaného modelu od Azure (či odjinud) může být jeho detekční síla. Společnost Microsoft díky svým projektům a popularitě má k dispozici veliké množství trénovacích dat, což je zásadní pro detekční schopnosti modelů. Nyní se budeme zabývat využitím ML (kognitivních služeb) specificky pro obor geodézie. Zůstaneme u obrazového rozpoznávání. Nebude se už ale jednat o předtrénovaný model, ale o vlastní, určený k detekci jiných objektů. Bude tedy představeno i trénování predikčního modelu a bude i ukázán vliv množství trénovacích dat na detekční schopnost.

V rámci této kapitoly bylo čerpáno z [4], [27], [29], [30].

5 Machine Learning v geodézii

Se vrůstajícím vývojem a využitím moderních technologií se časem můžeme setkávat se strojovým učením v celé oblasti geodézie, kartografie a geoinformatiky.

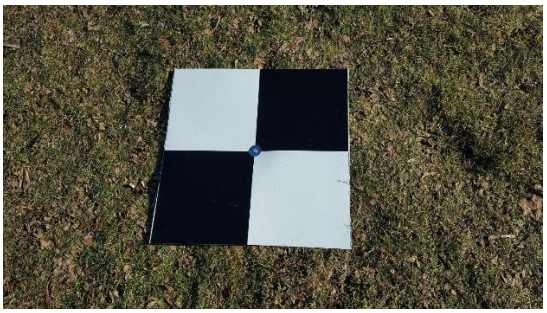
Například při měření totální stanicí je možné využít tzv. robotické totální stanice (RTS), tedy stanici s dálkovým ovládáním, kde je zapotřebí pouze jednoho operátora. Některé tyto stanice jsou opatřeny ATR technologií. ATR neboli *Automatic Target Recognitio*n je algoritmy hlubokého učení poháněná technologie pomáhající se zpracováváním obrazu, která se u RTS používá například pro automatické sledování operátorova odrazného hranolu. Příkladem jsou totální stanice iCON iCR70 či iCON iCR80 od společnosti Leica.

Další příklad využití ML je využití pro klasifikaci půdy v GIS. Tímto se např. vytváří vrstvy land-cover z leteckých snímků (či družicových). Obecně v GIS by se dalo uvažovat nepřeberné množství využití jako: hledání vzorů v mapě (křižovatky, sídliště, parkoviště s více než 50 místy apod.), hledání objektů (domy, bazény, červená auta, řeky aj.), hledání populárních míst (třeba velké množství lidí na jednom místě), odhadování bonity půdy, interpolace a extrapolace jevů a další. Mnohé ze zmíněných už je dávno nástroji v GIS softwarech implementováno.

V rámci této kapitoly bylo čerpáno z [37], [38] a [39].

5.1 Automatické vyhledávání vlícovacích bodů

V rámci této práce zabrousíme do oblasti fotogrammetrie. Jedná se o problematiku automatického vyhledávání vlícovacích bodů (VB, anglicky *Ground control points)*. Cíl bude tedy jasný; k dispozici budeme mít snímek a automatizovaně v něm budeme chtít označit oblast s černobílým pozemním terčem. Vzor takového terče je zde:



Obrázek 34 - Ukázka podoby pozemního vlícovacího bodu [34]

Budou nás tedy zajímat tyto čtvercové černobílé terče. V praxi se ale používají pro signalizaci VB i jiné terče nebo i symboly vyrobené na zemi či leccos jiného jednoznačně identifikovatelného (třeba rohy nějakých objektů). Na základě dosavadní teorie bude tedy úkolem opět vytvořit model, vytrénovat ho skupinou fotografií a poté využít tohoto modelu pro určení polohy těchto terčů z nových fotografií.

5.1.1 Klasifikace vs objektová detekce

V prvé řadě by bylo dobré objasnit tyto dva pojmy, protože se jedná o dvě rozdílné úlohy. Při klasifikaci bereme vstupní obrázek jako celek a výstupní hodnota v podobě třídy opět přísluší celému obrázku. Tedy kdybychom zde obrázky klasifikovali, zjišťovali bychom pouze zda se na obrázku někde alespoň jeden terč nachází. To ale pro náš účel nestačí.

Druhou úlohou je detekce objektů. Ta místo s celým obrázkem pracuje s regiony obrázku, které obsahují tzv. tag. Při procesu trénování se jedná o pojmenování druhu uživatelem určené oblasti. Tagy označují konkrétní objekty na obrázku a pomocí nich dochází k trénování poznávání těchto objektů. Je tedy nutné pro úspěšný model tyto tagy vytvořit. Výsledný model pak v testovacím obrázku určuje regiony, které označuje naučenými tagy.

Teď, když byly pojmy vysvětleny, lze s čistým štítem mluvit o detekci a její následující praktičtější ukázce. Pro tuto práci se bude jednat o detekci vlícovacích bodů.

5.1.2 Služba Azure Custom Vision pro vyhledávání VB

V kapitole 4.3 jsme si ukazovali službu Azure Face. Nyní budeme využívat podobnou službu, leč nebude tak konkrétní. Služba *Custom Vision* se nezabývá specifickými objekty, jakými jsou například tváře. Zabývá se rozpoznáváním obecných objektů na obrázcích a umožňuje uživateli vytvořit si a trénovat vlastní modely.

Kvůli potížím s připojením se na službu Custom Vision z prostředí jazyka Java bude teď využito jazyka Python verze 3.9 pro implementaci této služby v prostředí Spyder. Začneme s instalací a importem knihoven této služby. Pokračovat budeme vytvořením klienta a projektu. Následovat bude označování obrázků, jejich nahrávání do projektu a poté trénování. Nakonec bude ukázáno detekování na testovacích obrázcích a volitelné mazání projektu. Po této kapitole budou samozřejmě následovat praktické ukázky detekce na testovacích obrázcích pro vizualizaci kvality predikcí vytvořeného modelu.

5.1.2.1 Instalace a import knihoven

Abychom mohli používat nástroje knihovny, je nejprve nutné je nainstalovat. Z důvodu využívání jiného jazyka oproti předchozí službě to zde bude uvedeno znovu. Knihovna pro službu Custom Vision vyžaduje spoustu knihoven, na kterých je závislá. Manažerem v Pythonu pro tyto závislosti je příkaz *pip:*

```
pip install azure-cognitiveservices-vision-customvision
```

Tímto příkazem v PowerShellu nebo v konzoli se stáhne a nainstaluje celý balíček Custom Vison SDK. Poté lze úspěšně naimportovat knihovny služby Azure:

```
from azure.cognitiveservices.vision.customvision.training import CustomVisionTrainingClient
from azure.cognitiveservices.vision.customvision.prediction import CustomVisionPredictionClient
from azure.cognitiveservices.vision.customvision.training.models import ImageFileCreateBatch, ImageFileCreateEntry, Region, CustomVisionErrorException
from msrest.authentication import ApiKeyCredentials

from PIL import Image, ImageDraw

import os, time
```

Obrázek 35 - Import nástrojů z knihoven Custom Vision

Po úspěšné instalaci a importu knihoven lze nyní vytvořit klienta služby pro přístup k jejím možnostem.

5.1.2.2 Vytvoření klienta

Pro využívaní služby je nutné vytvořit objekt klienta, který opět slouží jako ověření oprávnění uživatele k využívání této služby. Zde je potřeba zdůraznit, že si uživatel musí vytvořit prostředek speciálně pro využití služby Custom Vision. Klíč ze služby Cognitive Services je využitelný pouze pro trénování, nikoliv pak pro predikci. Během zakládání prostředku (postup analogicky viz 4.1) je nutné specifikovat jeho využití jak k trénování, tak i k predikci. Jsou zde totiž potřeba klíče a endpointy zvlášť pro vytvoření objektu trenéra, tak i objektu predikátora, oba jakožto samostatné klienty. Pokud uživatel nebude disponovat těmito údaji a nevloží je, nebude schopen se ověřit a služba vrátí chybu autorizace.

```
# Autentizacni udaje
ENDPOINT = "[redacted]"
ENDPOINT_prediction = "[redacted]"
training_key = "[redacted]"
prediction_key = "[redacted]"
prediction_resource_id = "[redacted]"

# Vytvoreni objektu API klice
credentials = ApiKeyCredentials(in_headers={"Training-key": training_key})
prediction_credentials = ApiKeyCredentials(in_headers={"Prediction-key": prediction_key})

# Vytvoreni klienta pro trenovani a klienta pro predikci
trainer = CustomVisionTrainingClient(ENDPOINT, credentials)
predictor = CustomVisionPredictionClient(ENDPOINT_prediction, prediction_credentials)
```

Obrázek 36 - Vytvoření klientů Custom Vision pro trénování a predikci

Když je klient vytvořen, lze využívat nabízených prostředků této služby. Začneme vytvořením projektu, který vlastně představuje celý náš model, který budeme posléze i trénovat.

5.1.2.3 Vytvoření projektu

Všechny následující akce od této kapitoly dál se dají udělat ručně pomocí prostředí na adrese: https://www.customvision.ai. Zde budou všechny kroky prezentovány Python kódem.

Nyní je potřeba vytvořit projekt. Protože výchozí typ úlohy je klasifikace, je potřeba při vytváření projektu předat id objektu domény (*Domain*) s typem *Object Detection*. Zde je to provedeno prohledáváním přes všechny možné domény projektu.

```
# Hledani domeny detekce objektu
obj_detection_domain = next(domain for domain in trainer.get_domains() if domain.type == "ObjectDetection" and domain.name == "General")

# Vytvoreni noveho projektu
project = trainer.create_project("Detekce Vlicovacich bodu", domain_id=obj_detection_domain.id)
```

Obrázek 37 - Vytváření projektu Custom Vision

Po vytvoření projektu jakožto modelu je potřeba ho vytrénovat. Abychom to mohli udělat, je nejprve nutné nahrát trénovací obrázky, jejichž vlícovací body je nutné ručně označit.

5.1.2.4 Označování trénovacích obrázků a jejich nahrávání do projektu

Prostředek označování obrázků je tag, který nyní vytvoříme.

```
# Vytvoreni noveho tagu v projektu
GCP_tag = trainer.create_tag(project.id, "Vlicovaci bod")
```

Obrázek 38 - Vytvoření tagu

Označování tagu na obrázcích prostřednictvím kódu je trochu nešikovné. Je totiž potřeba znát souřadnice objektů v obrázcích a ty předat při vytváření objektů regionů. Nejlepší to je udělat ručně z prostředí vytvořeným Microsoftem (https://www.customvision.ai). Pro kompletní účinnost kódu bude uvedena implementace, kde je potřeba předat souřadnice všech označení hodnotou.

```
# Slovnik obrazku: kazdy klic je shodny se jmenem obrazku, kazda hodnota obsahuje seznam ctveric *
# *souradnic [levy okraj, horni okraj, sirka, vyska] obdelnika oznacujiciho vlicovaci bod
GCP_image_regions = {
"GCP_1":  [[0.14503284,  0.07174524,  0.7266537,    0.9117647   ]],
"GCP_2":  [[0.259174,    0.21579088,  0.39958668,   0.5830561   ]],
"GCP_3":  [[0.4051589,   0.5577953,   0.1525077,    0.20222926  ]],
"GCP_4":  [[0.1749643,   0.168353,    0.6236728,    0.4250071   ]],
"GCP_5":  [[0.45738587,  0.4009313,   0.15729335,   0.1432907   ]],
"GCP_6":  [[0.14570466,  0.11371032,  0.6513959,    0.8467      ]],
"GCP_7":  [[0.4568229,   0.48594114,  0.045,        0.060606062 ]],
"GCP_8":  [[0.46495378,  0.4849375,   0.04049296,   0.052       ]],
"GCP_9":  [[0.049993366, 0.11754427,  0.8749579,    0.8600839   ]],
"GCP_10": [[0.4125665,   0.37098667,  0.13846079,   0.21241716  ]],
"GCP_11": [[0.45096353,  0.34615546,  0.12469438,   0.21374446  ]],
"GCP_12": [],
"GCP_13": [[0.324375,    0.44319332,  0.104,        0.11336032  ],
           [0.496375,    0.42295042,  0.16,         0.17004049  ]],
"GCP_14": [[0.32353967,  0.42653275,  0.10844681,   0.12630177  ],
           [0.48578364,  0.4129492,   0.17856663,   0.20225006  ]],
"GCP_15": [[0.55687696,  0.1843774,   0.44311303,   0.5493654   ]],
"GCP_16": [[0.48013023,  0.46104896,  0.05550146,   0.08171206  ]],
"GCP_17": [],
"GCP_18": [[0.5730013,   0.7730819,   0.033930898,  0.08742398  ],
           [0.21775149,  0.5509693,   0.11306365,   0.18931943  ],
           [0.0697169,   0.44019714,  0.035198748,  0.08557364  ],
           [0.50516534,  0.13441047,  0.03368354,   0.08529471  ],
           [0.30070913,  0.14986068,  0.034056425,  0.08406618  ],
           [0.7274811,   0.24656048,  0.03247595,   0.08466795  ],
           [.88988155,   0.6034442,   0.032914042,  0.0861761   ]],
```

Obrázek 39 - Ukázka slovníku se souřadnicemi pro regiony (jenom část)

Zde je ukázka užitého slovníku. Klíče představují název trénovacího obrázku bez přípony a hodnoty přísluší souřadnicím pro tag. Objekty na obrázku označujeme prostřednictvím objektu region (*Region)*. Konstruktor tohoto objektu vyžaduje jednak id tagu, jednak souřadnice oblasti. Oblasti jsou definované hodnotami souřadnic levého a horního okraje, šířky a výšky obdélníku. Jednotky souřadnic nejsou v pixelech, nýbrž vyjadřují poměrovou část obrázku. Vypočítali bychom je jako:

$$poměrová\ souřadnice\ x\ bodu = \frac{souřadnice\ x\ bodu\ v\ pixelech}{celková\ šířka\ obrázku\ v\ pixelech}, \tag{5.1}$$

resp.

$$poměrová\ souřadnice\ y\ bodu = \frac{souřadnice\ y\ bodu\ v\ pixelech}{celková\ výška\ obrázku\ v\ pixelech}. \tag{5.2}$$

Nyní bude uveden trik získání těchto souřadnic, protože ruční napočítávání je značně zdlouhavé a je zde spousta prostoru pro chyby. Spočívá v dávkování fotek po 1. Bylo využito internetového prostředí této služby. Zde se totiž dají obrázky nahlédnout a myší snadno vytvořit obdélníky kolem vlícovacích bodů a označit je tagem. Do projektu je nejprve potřeba nahrát trénovací obrázky (s náhodně zvolenými regiony, třeba pouze hodnota souřadnic 0,1).

```python
# Inicializace pomocnych promennych
IMAGE_COUNTER_THRESHOLD = 64

tagged_images_with_regions = []
batches = []
regions = []
batch_image_names = []
image_counter = 0
images_quantity = 0
i = 0

# Prochazeni slovniku obrazku, vytvoření dávek potřebné k odeslání otagovaných obrazku do projektu
for file_name in GCP_image_regions.keys():
    list_xywh = GCP_image_regions[file_name]
    i =+1

    # Pokud obrazek obsahuje aspon jednu oblast
    if len(list_xywh) > 0:

        # Prochazeni jednotlivych oblasti kazdeho obrazku a vytvoreni seznamu objektu regionu (oblasti pro tagy)
        for xywh in list_xywh:

            # Pokud oblast ma vsechny 4 parametry: levy okraj, horni okraj, sirka, vyska
            if len(xywh) == 4:
                regions.append(Region(tag_id=GCP_tag.id, left=xywh[0],top=xywh[1],width=xywh[2],height=xywh[3]))

            else:
                print("{}: xywh musi obsahovat prave 4 hodnoty".format(file_name))


        # Ulozi obrazek s regiony do seznamu oznacenych fotek
        with open(os.path.join (base_image_location, "GCPoints_trenovani", file_name + ".jpg"), mode="rb") as image_contents:
            tagged_images_with_regions.append(ImageFileCreateEntry(name=file_name, contents=image_contents.read(), regions=regions))
            regions = []
            image_counter += 1
            images_quantity += 1
            batch_image_names.append(file_name)

            # Pokud je v seznamu oznacenych obrazku vice nez 64 fotek, vytvori davku a ulozi ji do seznamu davek
            # (varka nemuze obsahovat vice nez 64 fotek)
            if image_counter >= IMAGE_COUNTER_THRESHOLD:
                image_counter = 0
                batches.append(ImageFileCreateBatch(images=tagged_images_with_regions))
                tagged_images_with_regions = []

    # Kdyz obrazek nema zadny region, vypis info
    else:
        print("{} nema zadny tag region".format(file_name))

# Ze zbylych obrazku udela davku a ulozi ji do seznamu davek
if tagged_images_with_regions != []:
    batches.append(ImageFileCreateBatch(images=tagged_images_with_regions))
```

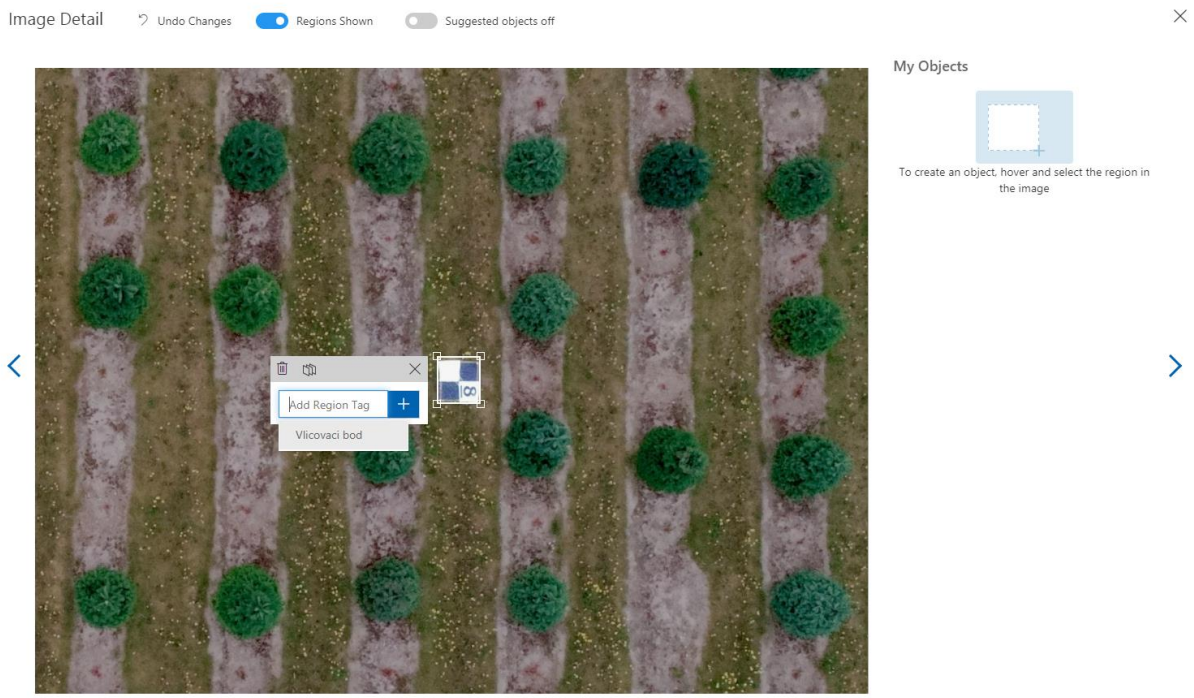
Obrázek 40 - Vytváření dávek pro posílání otagovaných fotek do projektu

Zde je uveden cyklus, kde ze slovníku uvedeného na obrázku 39 je procházena každá hodnota. Jelikož obrázky jsou přesně označeny tagy, může se nyní jednat pouze o náhodné hodnoty souřadnic. Pro každý klíč (trénovací obrázek) je nejprve přistoupeno ke všem souřadnicím. Jedná se o 2D pole, kde každý klíč může obsahovat celý vektor souřadnic pro tagy. Je tedy vytvořen seznam všech regionů pro aktuální obrázek a z něho následně společně s obrázkem vytvořen objekt *ImageFileCreateEntry*. Ten slouží jako sdružení regionů s obrázkem do jednoho objektu a též jako parametr konstruktoru objektu dávky *ImageFileCreateBatch*. Dávky jsou pak zapsány do seznamu dávek. Každá dávka může obsahovat maximálně 64 fotografií. Zde tuto kapacitu diriguje konstanta kódu *IMAGE_COUNTER_THRESHOLD*. Nyní vytvořené dávky (seznam *batches*) nahrajeme do projektu.

```
# Pokud jsou davky po 1 obrazku (slouzi pro kontrolu info a ziskani souradnic naklikanych regionu v projektu z https://www.customvision.ai)
if IMAGE_COUNTER_THRESHOLD == 1:
    images_dict = {}

    # Projde seznam davek a kazdou posle
    for i,batch in enumerate(batches):

        # Zkus v projektu vytvorit objekt Image (1 fotku)
        try:
            upload_result = trainer.create_images_from_files(project.id, batch)

        # Po nezdaru vytvoreni Image vypis info
        except CustomVisionErrorException as e:
            print("Fotka {} neni validni: {}".format(batch_image_names[i], e))

        # Projdi obrazky, ktere jsme se pokousili nahrat
        for image in upload_result.images:
            # Status nahrani fotky
            print("{}: {}".format(batch_image_names[i], image.status))

            # Pokud se obrazek nahral zdarile
            if (image.image != None):
                images_dict[image.image.id] = batch_image_names[i]

    # Ziskani otagovanych obrazku z projektu
    tagged_images = trainer.get_tagged_images(project_id=project.id, order_by='Newest')

    # Vypisovani souradnic regionu kazdeho otagovaneho obrazku
    for image in tagged_images:
        regions = image.regions
        for region in regions:
            print('"{}": [[{},    {},    {},    {}   ]],'.format(images_dict[image.id], region.left, region.top, region.width, region.height))

# Pokud davky obsahuji vice nez 1 obrazek
else:
    for i,batch in enumerate(batches):
        print("Posilam {}. davku fotek".format(i+1))
        try:
            upload_result = trainer.create_images_from_files(project.id, batch)
        except CustomVisionErrorException:
            print("No valid image files")
        # Upozorneni, pokud nejaka fotka vykazuje jakykoliv problem
        if not upload_result.is_batch_successful:
            print("{}. davka: NEUSPECH".format(i))
        else:
            print("{}. davka: OK".format(i))
```

Obrázek 41 - Nahrávání obrázků do projektu

V kódu na obrázku 41 oba cykly rozdělené podmínkou dělají defacto to samé – pošlou dávku a vypíšou její stav. Podmínka je zde důležitá, hlídá hodnotu *IMAGE_COUNTER_THRESHOLD* představující počet obrázků v každé dávce, kterou je v této fázi doporučeno mít nastavenou na hodnotu 1. Budou se tedy do projektu nahrávat obrázky po 1. Tím se dá sledovat status nahrání každého obrázku zvlášť, a protože se třeba z důvodu špatného formátu obrázky vždycky nahrát nemusí, tímto způsobem se dá zjistit, o který chybový obrázek se jedná. Celé je to zajištěno vytvořením slovníku *images_dict*, kde klíči jsou id každého obrázku a hodnotami jsou jejich názvy. Nahráním se totiž vytvoří v projektu objekty Image, které název obrázků neobsahují, ale pouze jejich id (a další atributy). Dávky jsou sice do objektu ukládány popořadě, ale jejich nahrávání do projektu už je náhodné. Po nahrání obrázků nastává ruční otagování obrázků.



Obrázek 42 - Ukázka označování obrázků z internetového prostředí Custom Vision [https://www.customvision.ai]

Po dokončeném otagování je nutné celý proces nahrávání obrázků zopakovat. Nahrajeme obrázky znovu s náhodnými regiony (třeba se souřadnicemi 0,1; 0,1; 0,1; 0,1) do stejného projektu. Je potřeba si na začátku uložit id projektu a id tagu a nikoli vytvořit nové.

```
project = trainer.get_project("vlozit id projektu")
GCP_tag = trainer.get_tag(project_id = project.id, tag_id = "vlozit id tagu")
```

Obrázek 43 - Přistoupení k existujícímu projektu a tagu

V cyklu podmíněném dávkováním po 1 (viz obrázek 41) pak metodou *get_tagged_images* a využitím slovníku *images_dict* jsou souřadnice regionů vypisovány.

```
# Ziskani otagovanych obrazku z projektu
tagged_images = trainer.get_tagged_images(project_id=project.id, order_by='Newest')

# Vypisovani souradnic regionu kazdeho otagovaneho obrazku
for image in tagged_images:
    regions = image.regions
    for region in regions:
        print('"{}": [[{},    {},    {},    {}   ]],'.format(images_dict[image.id], region.left, region.top, region.width, region.height))
```

Obrázek 44 - Vypisování regionů

Vše je pak jen záležitostí kopírování do slovníku *GCP_image_regions* (viz obrázek 39). Ovšem je nutno vymazat uměle vytvořené regiony a dbát na strukturu tohoto slovníku. Nyní, když máme přesné souřadnice regionů, můžeme proces znovu opakovat. Pro rychlejší běh celého procesu můžeme nastavit dávkování na maximální službou povolenou hodnotu 64. Je lepší vytvořit nový projekt a předchozí smazat, kvůli uměle znovu vytvořeným regionům.

Proces má kvůli nutnosti opakování značné manipulační nedostatky. Možnosti jeho vylepšení budou diskutovány v závěru této práce. Nyní, po dokončeném nahrávání trénovacích obrázků do projektu, nastává trénování projektu.

5.1.2.5 Trénování

Jsou-li fotografie do projektu nahrané s tagy označenými regiony, můžeme přistoupit k trénovací fázi. Tu stačí pouze zavolat metodou klienta *train_project*. Tím vytvoříme objekt iterace (*iteration*).

```
# Trenovani
iteration = trainer.train_project(project.id)
while (iteration.status != "Completed"):
    iteration = trainer.get_iteration(project.id, iteration.id)
    print ("Trenovaci status: " + iteration.status)
    time.sleep(1)
```

Obrázek 45 - Trénování projektu

Protože trénování nějakou dobu potrvá, je zde ještě cyklus pravidelně vypisující po 1 sekundě stav trénování, dokud není dokončené. Až tomu tak bude, je potřeba celou tuto iteraci do projektu publikovat.

```
# Jmeno pro publikovanou iteraci
publish_iteration_name = "Detekce vlicovacich bodu"

# Iterace dotrenovana. Publikovani iterace do projektu
trainer.publish_iteration(project.id, iteration.id, publish_iteration_name, prediction_resource_id)
```

Obrázek 46 - Publikování iterace

Publikaci je potřeba přiřadit nějaký název. Každé trénování je iterací a model můžeme iterovat vícekrát. Ovšem záleží na službě Azure, kolik jich nám dovolí. Placené verze služby budou nabízet větší množství iterací. Trialová verze jich nabízí maximálně 10 na každý projekt.

Až bude projekt vytrénován (publikovanou iterací), je možné ho využít k detekci, jež nyní následuje.

5.1.2.6 Detekce

Nyní máme v určité míře model vytrénovaný a můžeme ho tedy konečně využít pro detekci objektů na jiných obrázcích.

```
# Otevreni obrazku pro detekci VB a vraceni vysledku detekce
file_pred_name = "GCP_4"
file_pred_path = os.path.join (base_image_location, "GCPoints_detekce", file_pred_name + ".jpg")
with open(file_pred_path, mode="rb") as test_data:
    results = predictor.detect_image(project.id, publish_iteration_name, test_data)
```

Obrázek 47 - Detekce objektů

Stačí pouze otevřít obrázek a předat ho metodě *detect_image*. Zde je ale důležité vyzdvihnout, že se jedná o metodu klienta predikce nikoliv trénování. Poté můžeme výsledky predikce procházet. Zde byla zvolena jak cesta vypisování, tak i cesta vykreslování regionů do obrázku a jeho následné uložení.

```
# Vytvoreni kresliciho objektu pro ulozeni obdelniku kolem detekovanych vlicovacich bodu
image = Image.open(file_pred_path)
width, height = image.size
draw = ImageDraw.Draw(image)
color = "yellow"
```

Obrázek 48 - Vytvoření objektu kreslení a sejmutí rozměrů testovacího obrázku

Pro vykreslování regionů je třeba vytvořit objekt kreslení. Též je nutno kvůli poměrovým souřadnicím, které klient predikce jako výsledky vrací, si uložit hodnoty rozměru obrázku, abych posléze mohl regiony vykreslit v přepočtených pixelových souřadnicích (viz rovnice (5.1) a (5.2)).

```
# Zobrazeni vysledku a vykreslovani regionu do obrazku na zaklade zvoleneho prahu [%]
PREDICTION_THRESHOLD = 50
for prediction in results.predictions:
    pred_prob = prediction.probability * 100
    pred_x = prediction.bounding_box.left
    pred_y = prediction.bounding_box.top
    pred_w = prediction.bounding_box.width
    pred_h = prediction.bounding_box.height

    result_format = ": {0:.2f}% bbox.left = {1:.2f}, bbox.top = {2:.2f}, bbox.width = {3:.2f}, bbox.height = {4:.2f}"
    result_probab_region = result_format.format(pred_prob, pred_x, pred_y, pred_w, pred_h)

    print("\t" + prediction.tag_name + result_probab_region)

    if pred_prob >= PREDICTION_THRESHOLD:
        draw.rectangle((pred_x*width, pred_y*height, (pred_x + pred_w)*width, (pred_y+pred_h)*height), outline=color, width=3)
```

Obrázek 49 - Vypisování výsledků a vykreslování detekovaných regionů do výstupního obrázku

Na obrázku 49 je cyklus procházející všechny výsledky predikce k danému testovacímu obrázku. Predikce klienta vrací jak souřadnice odhadovaného regionu, tak i pravděpodobnost jeho předpokládané správnosti. Výsledků vrací predikátor hodně. Spousta z nich nebude vůbec hledaný objekt obsahovat. Je potřeba tyto odhady na základě vrácené pravděpodobnosti filtrovat. Budou se zde totiž nacházet i výsledky s velice malou pravděpodobností, ty pravděpodobně nebudou detekcí ničeho. Nabízí se docela standardní přístup 50% prahu, který je možný vidět i v uvedeném obrázku 49. Poté stačí výsledek vykreslení už jen uložit.

```
# Ulozeni obrazku s detekovanym vlicovacim bodem
image.save(os.path.join (base_image_location, "GCPoints_detekce", file_pred_name + "_detekce.jpg"))
```

Obrázek 50 - Ukládání obrázku s vykreslenými detekcemi

Projekt se samozřejmě dá využívat nadále k dalším detekcím. Může se ale stát, že bude potřeba projekt smazat.

5.1.2.7 Smazání projektu

Pro případ potřeby bude uvedeno, jak se takový projekt z rozhraní Python ruší. Nelze ho rovnou smazat, pokud je zde nějaká publikovaná iterace. Je nutné tyto iterace nejprve odpublikovat a následně je možné projekt smazat.

```
# Odpublikovani iteraci
trainer.unpublish_iteration(project.id, iteration.id)

# Smazani projektu
trainer.delete_project(project.id)
```

Obrázek 51 - Odpublikování iterací a smazání projektu

Teď, když byla implementace představena, můžou být uvedeny příklady výsledků pro prezentování síly predikce modelu.

5.1.3 Představení výsledků služby Azure Custom Vision

Než budou ukázány testovací obrázky, bude dobré představit jaké výsledky služba vrací. Metoda *detect_image* (viz obrázek 47) vrací objekt třídy *ImagePrediction*. Ten obsahuje informace o jednotlivém obrázku, a to id této predikce, id projektu, id iterace, která byla použita pro detekci (je totiž možné využít, kterékoliv publikované iterace v daném projektu), čas, kdy byla predikce vytvořena, a nakonec to hlavní – seznam objektů *Prediction*. Tyto objekty představují jednotlivé detekované oblasti na obrázku. Atributy jsou pravděpodobnost, s jakou může jít o označený tag, id tagu, jméno tagu, *BoundingBox* objekt, který obsahuje souřadnice této detekované oblasti a typ tagu (existují 3 typy, použit je typ *regular*, ale to není podstatné). Pro účel této práce bude důležitý pouze atribut ohraničujícího obdélníku a pravděpodobnost. Objekt *BoundingBox* obsahuje pouze 4 atributy a to

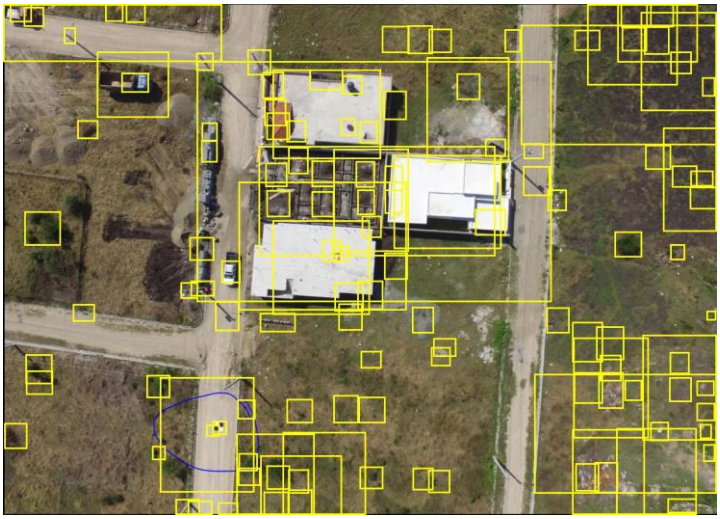
souřadnice: levý okraj, horní kraj, šířka, výška. Příklad výstupu sestrojeného kódu pro je následovný:

```
Vlicovaci bod: 60.61% bbox.left = 0.17, bbox.top = 0.13, bbox.width = 0.04, bbox.height = 0.04
Vlicovaci bod: 57.31% bbox.left = 0.29, bbox.top = 0.82, bbox.width = 0.02, bbox.height = 0.03
Vlicovaci bod: 46.80% bbox.left = 0.31, bbox.top = 0.50, bbox.width = 0.03, bbox.height = 0.04
Vlicovaci bod: 33.28% bbox.left = 0.48, bbox.top = 0.14, bbox.width = 0.03, bbox.height = 0.04
Vlicovaci bod: 25.82% bbox.left = 0.24, bbox.top = 0.97, bbox.width = 0.03, bbox.height = 0.03
Vlicovaci bod: 17.48% bbox.left = 0.68, bbox.top = 0.26, bbox.width = 0.03, bbox.height = 0.04
Vlicovaci bod: 12.23% bbox.left = 0.21, bbox.top = 0.87, bbox.width = 0.02, bbox.height = 0.03
Vlicovaci bod: 10.15% bbox.left = 0.33, bbox.top = 0.35, bbox.width = 0.24, bbox.height = 0.25
Vlicovaci bod: 9.37% bbox.left = 0.99, bbox.top = 0.60, bbox.width = 0.01, bbox.height = 0.02
Vlicovaci bod: 9.17% bbox.left = 0.46, bbox.top = 0.47, bbox.width = 0.02, bbox.height = 0.03
Vlicovaci bod: 7.58% bbox.left = 0.47, bbox.top = 0.22, bbox.width = 0.02, bbox.height = 0.03
Vlicovaci bod: 7.55% bbox.left = 0.36, bbox.top = 0.60, bbox.width = 0.05, bbox.height = 0.04
Vlicovaci bod: 6.20% bbox.left = 0.45, bbox.top = 0.46, bbox.width = 0.03, bbox.height = 0.04
Vlicovaci bod: 5.87% bbox.left = 0.26, bbox.top = 0.97, bbox.width = 0.03, bbox.height = 0.03
Vlicovaci bod: 4.86% bbox.left = 0.01, bbox.top = 0.00, bbox.width = 0.03, bbox.height = 0.04
Vlicovaci bod: 4.62% bbox.left = 0.47, bbox.top = 0.59, bbox.width = 0.03, bbox.height = 0.05
Vlicovaci bod: 4.24% bbox.left = 0.50, bbox.top = 0.41, bbox.width = 0.03, bbox.height = 0.04
Vlicovaci bod: 3.83% bbox.left = 0.57, bbox.top = 0.59, bbox.width = 0.03, bbox.height = 0.05
Vlicovaci bod: 3.68% bbox.left = 0.43, bbox.top = 0.37, bbox.width = 0.03, bbox.height = 0.05
Vlicovaci bod: 3.63% bbox.left = 0.83, bbox.top = 0.72, bbox.width = 0.04, bbox.height = 0.07
Vlicovaci bod: 3.32% bbox.left = 0.49, bbox.top = 0.31, bbox.width = 0.03, bbox.height = 0.05
Vlicovaci bod: 3.20% bbox.left = 0.27, bbox.top = 0.11, bbox.width = 0.50, bbox.height = 0.47
Vlicovaci bod: 3.19% bbox.left = 0.38, bbox.top = 0.13, bbox.width = 0.15, bbox.height = 0.17
Vlicovaci bod: 3.01% bbox.left = 0.26, bbox.top = 0.54, bbox.width = 0.03, bbox.height = 0.03
Vlicovaci bod: 2.95% bbox.left = 0.25, bbox.top = 0.54, bbox.width = 0.02, bbox.height = 0.03
Vlicovaci bod: 2.84% bbox.left = 0.28, bbox.top = 0.27, bbox.width = 0.02, bbox.height = 0.04
Vlicovaci bod: 2.60% bbox.left = 0.50, bbox.top = 0.36, bbox.width = 0.02, bbox.height = 0.05
Vlicovaci bod: 2.60% bbox.left = 0.40, bbox.top = 0.23, bbox.width = 0.03, bbox.height = 0.04
Vlicovaci bod: 2.39% bbox.left = 0.03, bbox.top = 0.01, bbox.width = 0.03, bbox.height = 0.04
Vlicovaci bod: 2.38% bbox.left = 0.13, bbox.top = 0.09, bbox.width = 0.10, bbox.height = 0.13
Vlicovaci bod: 2.29% bbox.left = 0.09, bbox.top = 0.05, bbox.width = 0.02, bbox.height = 0.03
Vlicovaci bod: 2.24% bbox.left = 0.84, bbox.top = 0.69, bbox.width = 0.03, bbox.height = 0.05
Vlicovaci bod: 2.24% bbox.left = 0.37, bbox.top = 0.90, bbox.width = 0.03, bbox.height = 0.05
Vlicovaci bod: 1.99% bbox.left = 0.37, bbox.top = 0.36, bbox.width = 0.03, bbox.height = 0.05
Vlicovaci bod: 1.98% bbox.left = 0.37, bbox.top = 0.19, bbox.width = 0.03, bbox.height = 0.05
Vlicovaci bod: 1.94% bbox.left = 0.53, bbox.top = 0.17, bbox.width = 0.03, bbox.height = 0.06
Vlicovaci bod: 1.91% bbox.left = 0.36, bbox.top = 0.28, bbox.width = 0.33, bbox.height = 0.22
Vlicovaci bod: 1.90% bbox.left = 0.34, bbox.top = 0.09, bbox.width = 0.03, bbox.height = 0.05
Vlicovaci bod: 1.76% bbox.left = 0.80, bbox.top = 0.83, bbox.width = 0.10, bbox.height = 0.17
Vlicovaci bod: 1.74% bbox.left = 0.97, bbox.top = 0.74, bbox.width = 0.03, bbox.height = 0.03
Vlicovaci bod: 1.68% bbox.left = 0.96, bbox.top = 0.33, bbox.width = 0.03, bbox.height = 0.03
Vlicovaci bod: 1.67% bbox.left = 0.86, bbox.top = 0.44, bbox.width = 0.04, bbox.height = 0.05
Vlicovaci bod: 1.67% bbox.left = 0.33, bbox.top = 0.91, bbox.width = 0.03, bbox.height = 0.04
Vlicovaci bod: 1.62% bbox.left = 0.83, bbox.top = 0.90, bbox.width = 0.03, bbox.height = 0.06
Vlicovaci bod: 1.61% bbox.left = 0.40, bbox.top = 0.28, bbox.width = 0.03, bbox.height = 0.05
```

Obrázek 52 - Ukázka výstupu kódu pro službu Custom Vision

Pracováno bylo tedy pouze se souřadnicemi detekce a její pravděpodobností. Jak je možné vidět na obrázku, služba detekuje na obrázku větší množství objektů. Zde jsou seřazeny podle pravděpodobnosti sestupně. Výsledky s největší pravděpodobností by zřejmě mohly obsahovat hledaný terč vlícovacího bodu. U obrázku 49 byl uveden 50% práh, který tyto

výsledky filtruje. Pokud by toto filtrování výsledků nebylo učiněno, vypadaly by výsledky následovně:



Obrázek 53 - Ukázku výstupu s nulovým prahem

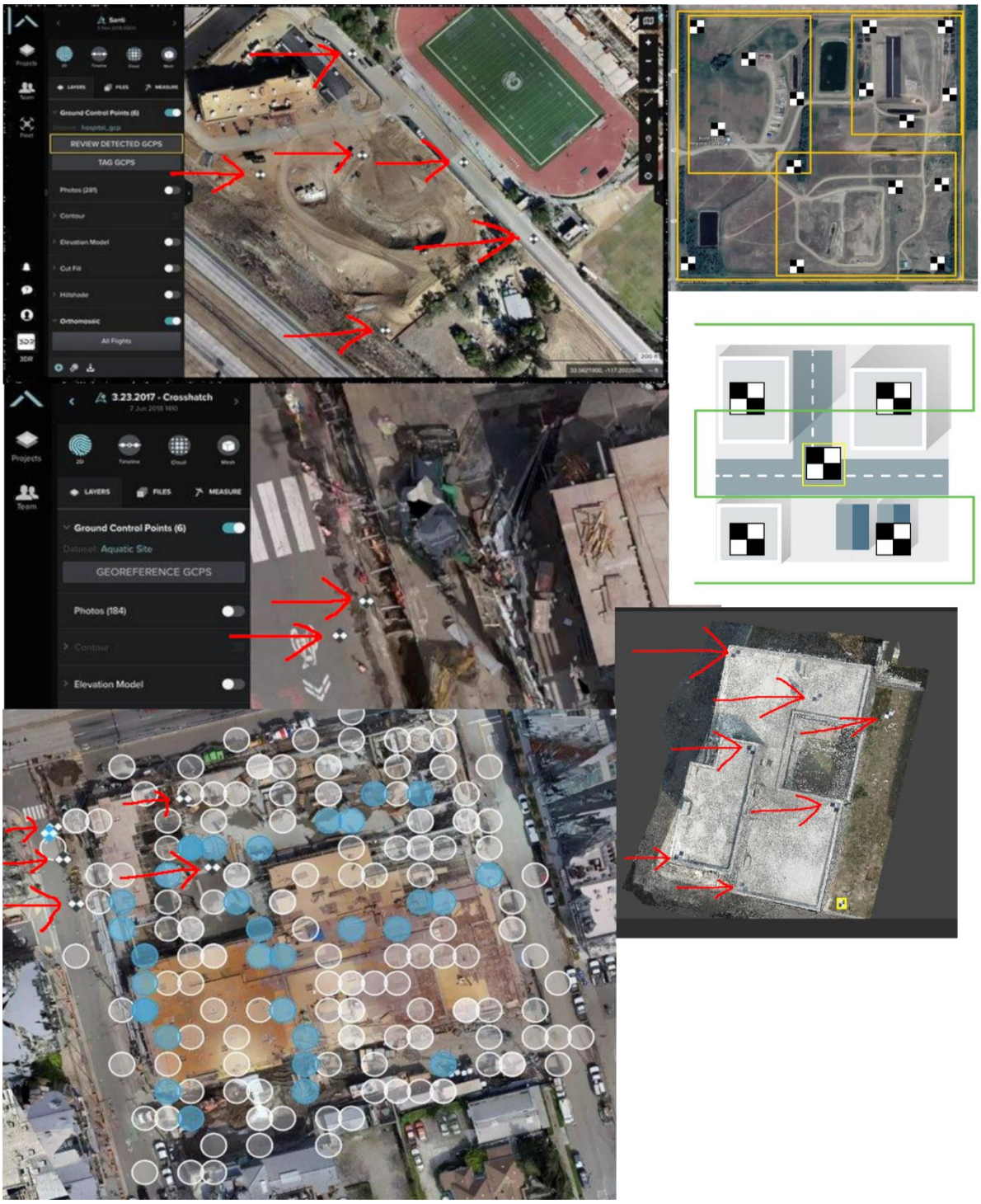
Je zřejmé, že takový výsledek není žádoucí a potvrzuje důležitost role prahu. Dále tedy bude v ukázkách výsledků na testovacích obrázcích pracováno s kritériem alespoň 50% pravděpodobnosti.

5.1.4 Výsledky

Nyní budou uvedeny výsledky 18 testovacích obrázků sestavených do 3 koláží. Všechny terče, které nebudou na obrázcích úspěšně označeny a zároveň nebudou jasně viditelné, budou ukázány ručně vytvořenou červenou šipkou.

Obrázek 54 - Výsledky Custom Vision (koláž)

Obrázek 55 - Výsledky Custom Vision (koláž)



Obrázek 56 - Výsledky Custom Vision (koláž)

Je zjevné, že model není ani zdaleka 100% účinný. Na obrázku 56 vpravo nahoře se nachází větší žluté obdélníky. Zde se nejedná o oblasti detekce, ale součást původního testovacího obrázku (model zde nedetekoval nic). Diskusi o kvalitě výsledků uvedu v nadcházejícím závěru práce.

V rámci této kapitoly 5 bylo čerpáno z [34], [35], [36].

Závěr

V této práci bylo osvětleno téma strojového učení. S tímto oborem se student při studiu téměř nesetká, pouze okrajově v rámci oboru vyrovnávacího počtu či pravděpodobnosti a statistiky. Proto zde v první části práce byla rozebírána jednotlivá podtémata jako regrese, klasifikace, klastrování, učení asociačních pravidel, zpětnovazební učení, zpracování přirozeného jazyka, evoluční algoritmy a hluboké učení, vše jakožto nástroje či úlohy strojového učení. V každém podtématu bylo vyzdvihnuto několik algoritmů řešící onu úlohu. Důraz byl hlavně na získaní intuice a povrchového chápání.

V druhé části byly uvedeny 2 ukázky využití webových služeb Azure, jejichž kódy jsou přílohou této práce. Tou první byla služba Face sloužící k rozpoznávání lidských tváří a jejich analýze. Jednalo se o využití už předtrénovaného modelu z prostředí jazyka Java. Využito bylo Azure SDK (sada vývojových nástrojů) a to Face API.

Druhou ukázkou 2. části byla služba Custom Vision. Z důvodu komplikací s vytvořením klienta služby nebylo v této ukázce využíváno stejného prostředí jazyka Java jako v ukázce první, ale prostředí jazyka Python. Knihovnou byla sada nástrojů Custom Vision. Zde byl vytvořen zcela nový a vlastní model určený k detekování vlastních objektů, kterými byly pozemní terče vlícovacích bodů pro snímkování z dronu či letadel. Model byl trénován na cca 100 obrázcích a výsledek výkonu modelu byl testován na 18 testovacích obrázcích. Kód této ukázky je velice hrubý. Je zde hodně prostoru pro vylepšení zamýšlené funkcionality. Jednak by objekt nesoucí souřadnice regionů nemusel být součástí kódu, ale mohl by být importován z externího textového souboru. Také by se mohlo lépe zautomatizovat získávání souřadnic regionů. Potřeba opětovného spouštění kódu se změnami parametrů je kostrbatá a pro neorientovaného člověka matoucí. Obrázky by se mohly nahrávat bez nutného vložení „umělých“ regionů (libovolné souřadnice), což implikuje nutnost úpravy následně vrácené dávky souřadnic regionů a také by se celý kód mohl rozdělit na více samostatných Python projektů (např. zvlášť vložení obrázků, trénování a detekce) pro lepší celkovou manipulaci se službou.

Co se týče výsledků testovacích obrázků, ty byly záměrně rozděleny do 3 skupin (koláží). První skupina (obrázek 54) vykazuje oproti zbývajícím 2 nejlepší výsledky a chyby jsou zde minimální. Druhou skupinou (obrázek 55) jsou snímky z dronů, které už tolik úspěšné nejsou. Terče jsou detekovány ojediněle. Poslední koláží (obrázek 56) je soubor graficky upravených snímků či zcela graficky vytvořených obrázků. Porovnáním všech výsledků lze říct, že model

umí na snímcích lepé rozpoznávat bližší terče nebo pokud je jednodušší terén (například pouze zelená louka). Jde-li o graficky vytvořené obrázky nebo o velice malé štítky či barevně složitý terén, začne mít model s detekcí problémy. Trénovací množina čítající cca 100 obrázků je nedostačující. Pro sofistikovanější model bude potřeba mnohem početnějšího vzorku. Záleží také o jaké trénovací obrázky se jedná. V této práci se pracovalo s velice různými obrázky (různá vzdálenost terčů od objektivu, různá rozlišení atd.). Spousta z nich, ale obsahovala terče velice zblízka. Pokud bychom chtěli dobrou detekční schopnost pro snímky z dronů či letadel, je dobré model natrénovat právě na těchto snímcích.

Snahou této práce byla hlavně ukázka těchto služeb a předání podnětů k dalším možným aplikacím strojového učení do oblasti geodézie, kartografie či geoinformatiky. Praktická část s detekcí vlícovacích bodů ukázala, že využití ML rozhodně smysl má a že s dostatkem času a množstvím dat se jedná o silný dobře využitelný nástroj, který značně zrychlí a zjednoduší lidskou práci.

Literatura a zdroje

[1] EREMENKO, Kirill; DE PONTEVES, Hadelin (2022). *Machine Learning A-Z™: Hands-On Python & R In Data Science* [online kurz]. UDEMY [cit. 5.4.2022]. Dostupný z: https://www.udemy.com/course/machinelearning/

[2] EREMENKO, Kirill (2022). *SVR Intuition* [lekce online kurzu]. Kirill, E.; Hadelin, de P. (2022); *Machine Learning A-Z™: Hands-On Python & R In Data Science* [online kurz]. UDEMY [cit. 5.4.2022]. Dostupný z: https://www.udemy.com/course/machinelearning/learn/lecture/19505880#learning-tools

[3] *Scikit Learn* [online]. Dostupný z: https://scikit-learn.org/stable/index.html

[4] *Face API* [online]. Microsoft [cit. 3.5.2022]. Dostupné z: https://docs.microsoft.com/en-us/java/api/overview/azure/cognitiveservices/client/faceapi?view=azure-java-legacy

[5] Georgios, D. *Decision Tree Regressor explained in depth* [online]. *GDCoder* [cit. 5.4.2022]. Dostupný z: https://gdcoder.com/content/images/2019/05/Screen-Shot-2019-05-17-at-00.09.26.png

[6] Amal. *Logistic Regression Made Simple* [online]. Medium [cit. 7.4.2022]. Dostupné z: https://amalaj7.medium.com/logistic-regression-eb2903251107

[7] GARCÍA-GONZALO, Esperanza a kol. *Hard-Rock Stability Analysis for Span Design in Entry-Type Excavations with Learning Classifiers - Scientific Figure* [online]. ResearchGate [cit. 7.4.2022]. Dostupné z: https://www.researchgate.net/figure/Classification-of-data-by-support-vector-machine-SVM_fig8_304611323

[8] SUVIGYA, Saxena. *The Gaussian RBF Kernel in Non Linear* [online]. Medium [cit. 7.4.2022]. Dostupné z: https://medium.com/@suvigya2001/the-gaussian-rbf-kernel-in-non-linear-svm-2fb1c822aae0

[9] *How Many Words Do I Need To Know*? The 95/5 Rule In Language Learning, Part 2/2 [online]. lingHOLIC [cit. 18.4.2022]. Dostupné z: https://www.lingholic.com/how-many-words-do-i-need-to-know-the-955-rule-in-language-learning-part-2/

[10] Wil. *How Many Words Are in the English Language* [online]. English Live [cit. 18.4.2022]. Dostupné z: https://englishlive.ef.com/blog/language-lab/many-words-english-language/#:~:text=If%20we%20want%20to%20talk,How%20about%20you%2C%20though%3F

[11] SHUKLA, Lavanya. *Natural Language Processing* [online]. Weights & Biases [cit. 18.4.2022]. Dostupné z: https://wandb.ai/lavanyashukla/vega-plots/reports/Natural-Language-Processing--Vmlldzo2Nzk2Ng

[12] D. FOOTE, Keith. *A Brief History of Deep Learning* [online]. Dataversity [cit. 19.4.2022]. Dostupné z: https://www.dataversity.net/brief-history-deep-learning/

[13] HEMSOTH, Nicole. *DEEP LEARNING PIONEER PUSHING GPU NEURAL NETWORK LIMITS* [online]. The Next Platform [cit. 19.4.2022]. Dostupné z: https://www.nextplatform.com/2015/05/11/deep-learning-pioneer-pushing-gpu-neural-network-limits/

[14] KYRYKOVYCH, Anastasia. *Deep Neural Networks* [online]. KDnuggets [cit. 19.4.2022]. Dostupné z: https://www.kdnuggets.com/2020/02/deep-neural-networks.html

[15] KUČERA, Jiří. *Strojové učení pro řízení simulovanýchvozidel* [online]. Praha, 2020 [cit. 19.4.2022]. Diplomová práce. Univerzita Karlova, Matematicko-fyzikální fakulta. Mgr. Jakub Gemrot, Ph.D. Dostupné z: https://dspace.cuni.cz/bitstream/handle/20.500.11956/120970/120371184.pdf?sequence=1&isAllowed=y

[16] GUPTA, Vikas (2017). *Understanding Feedforward Neural Networks* [online]. LearnOpenCV [cit. 19.4.2022]. Dostupné z: https://learnopencv.com/understanding-feedforward-neural-networks/

[17] *Introduction of Convolutional Neural Network in TensorFlow* [online]. Javatpoint [cit. 19.4.2022]. Dostupné z: https://www.javatpoint.com/convolutional-neural-network-in-tensorflow

[18] KUČERA, Jiří. *Učení plánovacích modelů*. Praha, 2017 [cit. 20.4.2022]. Bakalářská práce. Univerzita Karlova, Matematicko-fyzikální fakulta. prof. RNDr. Roman Barták, Ph.D.

[19] SONI, Devin. *Introduction to Evolutionary Algorithms* [online]. Towards Data Science [cit. 20.4.2022]. Dostupné z: https://towardsdatascience.com/introduction-to-evolutionary-algorithms-a8594b484ac

[20] *Evolutionary Algorithm* [online]. Science Direct [cit. 20.4.2022]. Dostupné z: https://www.sciencedirect.com/topics/biochemistry-genetics-and-molecular-biology/evolutionary-algorithm

[21] *Scikit Learn - Introduction* [online]. Tutorialspoint [cit. 20.4.2022]. Dostupné z: https://www.tutorialspoint.com/scikit_learn/scikit_learn_introduction.htm#:~:text=Scikit-learn%20(Sklearn)%20is,a%20consistence%20interface%20in%20Python.

[22] *Tensorflow* [online]. Dostupné z: https://www.tensorflow.org

[23] *Basics of machine learning with TensorFlow* [online]. Tensorflow [cit. 20.4.2022]. Dostupné z: https://www.tensorflow.org/resources/learn-ml/basics-of-machine-learning

[24] *Keras* [online]. Dostupné z: https://keras.io

[25] *About Keras* [online]. Keras [cit. 20.4.2022]. Dostupné z: https://keras.io/about/

[26] *Scikit-learn vs. TensorFlow vs. PyTorch vs. Keras* [online]. Ritza Articles [cit. 20.4.2022]. Dostupné z: https://ritza.co/articles/scikit-learn-vs-tensorflow-vs-pytorch-vs-keras/

[27] *What are web services in Machine Learning Server?* [online]. Microsoft [cit. 21.4.2022]. Dostupné z: https://docs.microsoft.com/en-us/machine-learning-server/operationalize/concept-what-are-web-services

[28] Azure [online]. Dostupné z: https://azure.microsoft.com/en-us/

[29] *Machine Learning on AWS* [online]. AWS [cit. 21.4.2022]. Dostupné z: https://aws.amazon.com/machine-learning/

[30] *What is Azure?* [online]. Microsoft Azure [cit. 25.4.2022]. Dostupné z: https://azure.microsoft.com/en-us/overview/what-is-azure/

[31] *Home – Microsoft Azure* [sejmutí obrazovky online]. Microsoft Azure [cit. 25.4.2022]. Dostupné z: https://portal.azure.com/#home

[32] *Azure-Samples/cognitive-services-quickstart-code* [online]. Github [cit. 25.4.2022]. Dostupné z: https://github.com/Azure-Samples/cognitive-services-quickstart-code/tree/master/java/Face

[33] HALL, John (2018). *You Are the Average of the 10 People You Spend Time With: Here Are 5 Tips to Curate Great People* [online]. Inc. [cit. 3.5.2022]. Dostupné z: https://www.incimages.com/uploaded_files/image/1920x1080/getty_517194189_373099.jpg

[34] FARLEY, Patrick a kol (2021). *Quickstart: Create an image classification project with the Custom Vision client library or REST API* [online]. Microsoft. [cit. 9.5.2022]. Dostupné z: https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/quickstarts/image-classification?tabs=visual-studio&pivots=programming-language-python

[35] *cognitive-services-quickstart-code* [online]. Github [cit. 9.5.2022]. Dostupné z: https://github.com/Azure-Samples/cognitive-services-quickstart-code/blob/ed46e9ce325455daeb5fbe7c2656d4f77192b99f/python/CustomVision/ObjectDetection/CustomVisionQuickstart.py

[36] *customvision Package* [online]. Microsoft. [cit. 9.5.2022]. Dostupné z: https://docs.microsoft.com/en-us/python/api/azure-cognitiveservices-vision-customvision/azure.cognitiveservices.vision.customvision?view=azure-python

[37] *Operational AutomaticTarget Recognition (ATR) Whitepaper* [online]. Seebyte [cit. 7.5.2022]. Dostupné z: https://www.seebyte.com/media/1455/operational-atr-whitepaper.pdf

[38] *Leica iCON – Robotic Total Stations for Construction* [online]. Leica Geosystems [cit. 7.5.2022]. https://leica-geosystems.com/products/construction-tps-and-gnss/robotic-total-stations

[39] SINGH, Rohit (2019). *Where Deep Learning Meets GIS* [online]. Esri. [cit. 3.5.2022]. Dostupné z: https://www.esri.com/about/newsroom/arcwatch/where-deep-learning-meets-gis/#:~:text=Machine%20learning%20has%20been%20a,create%20land-cover%20classification%20layers.

Zdroje trénovacích a testovacích obrázků

[40] BOFFERDING, Alyce (2021*). New in Skyward Mapping & Modeling: Ground Control Points (GCPs)*. Skyward. [cit. 10.5.2022]. Dostupné z: https://skyward.io/new-in-skyward-mapping-modeling-ground-control-points-gcps/

[41] IR. Tools (2021). *Ground Control Point Markers* [online]. Army Technology [cit. 10.5.2022]. Dostupné z: https://www.army-technology.com/products/ground-control-point-markers/

[42] *Ground Control Points. The Cornerstone Of Accuracy* [online]. PIX-PRO. [cit. 10.5.2022]. Dostupné z: https://www.pix-pro.com/ground-control-points-accuracy/

[43] Willis Worxs. *UAV Ground Control Points (GCPs)/Aerial Targets for Aerial Mapping & Surveying (6 Pack) with Center Passthrough* [online]. Amazon. [cit. 10.5.2022]. Dostupné z: https://www.amazon.com/Control-Targets-Mapping-Surveying-Passthrough/dp/B07G8W26CP

[44] *How To: Use ground control points in Site Scan Manager for ArcGIS* [online]. Esri. [cit. 10.5.2022]. Dostupné z: https://support.esri.com/en/technical-article/000023040

[45] *Show us your Ground Control Point targets - Share your experience* [online]. Emlid. [cit. 10.5.2022]. Dostupné z: https://community.emlid.com/t/show-us-your-ground-control-point-targets-share-your-experience/13039

[46] Damian Wierzbicki, Marcin Nienaltowski. (2019)*. Accuracy Analysis of a 3D Model of Excavation, Created from Images Acquired with an Action Camera from Low Altitudes* [online]. Research Gate.

[cit. 10.5.2022]. Dostupné z: https://www.researchgate.net/publication/331093994_Accuracy_Analysis_of_a_3D_Model_of_Excavation_Created_from_Images_Acquired_with_an_Action_Camera_from_Low_Altitudes

[47] Larry Fox. (2017). *A COST EFFECTIVE METHOD FOR CAPTURING HIGH ACCURACY GROUND CONTROL* [online]. Bad Elf. [cit. 10.5.2022]. Dostupné z: https://bad-elf.com/blogs/bad-elf/high-accuracy-ground-control-for-aerial-imagery-capture

[48] PeteM. (2017). *Where to buy GCP plastic target?* [online]. Phantom Pilots. [cit. 10.5.2022]. Dostupné z: https://phantompilots.com/threads/where-to-buy-gcp-plastic-target.108809/page-2

[49] *Ground Control Points as a Service* [online]. Civil Tracker. [cit. 10.5.2022]. Dostupné z: https://civiltracker.xyz/ground-control-points-as-a-service/

[50] *Signalizační terč 300 x 300 mm, sada 10ks* [online]. Bohemia G Shop. [cit. 10.5.2022]. Dostupné z: https://bohemiagshop.cz/geodeticke-terce-pro-drony-reflexni-terciky-stitky-terce-pro-fotogrammetrii/927-signalizacni-terc-300-x-300-mm-sada-10ks.html

[51] maxdarrow01. *Ground Control Points: A Complete Guide* [online]. Flykit Blog. [cit. 10.5.2022]. Dostupné z: https://blog.flykit.app/ground-control-points-a-complete-guide/

[52] Mike Ball (2016). *Propeller Aero Announces Smart Drone Survey Ground Control Points. [online]. Unmanned Systems Technology*. [cit. 10.5.2022]. Dostupné z: https://www.unmannedsystemstechnology.com/2016/08/propeller-aero-announces-smart-drone-survey-ground-control-points/

[53] Willis Worxs. *XL UAV Ground Control Points (GCPs)/Aerial Targets for Aerial Mapping & Surveying (6 Pack) with Center Eyelet (1 m x 1 m) | Drone GCP* [online]. Amazon. [cit. 10.5.2022]. Dostupné z: https://www.amazon.co.uk/Ground-Control-Targets-Mapping-Surveying/dp/B08DW5MTZ1

[54] Aerotas. *UAV Ground Control Points (GCPs)/Disposable Aerial Targets for Aerial Mapping & Drone Surveying (100 Pack)* [online]. Amazon. [cit. 10.5.2022]. Dostupné z: https://www.amazon.com/Control-Disposable-Targets-Mapping-Surveying/dp/B08HJQM5YN

[55] (2019). *Evolution of Aerotas Ground Control Points (GCPs)* [online]. Aerotas. [cit. 10.5.2022]. Dostupné z: https://www.aerotas.com/blog/evolution-of-aerotas-gcps

[56] *GROUND CONTROL POINTS (GCPS)* [online]. Aerotas. [cit. 10.5.2022]. Dostupné z: https://www.aerotas.com/store/aerotas-ground-control-points

[57] Enviroq Asset Management. *UAV Drone SUSA Ground Control Points (4 Pack) GCP (60 cm x 60 cm) Aerial Target Marker Boards for Aerial Surveying and Mapping Check Points CP Rigid & Weatherproof* [online]. Amazon. [cit. 10.5.2022]. Dostupné z: https://www.amazon.co.uk/Ground-Control-Surveying-Mapping-Weatherproof-Black/dp/B08MRRS94S

[58] (2020). *The future of accuracy: marking GCPs automatically* [online]. Pix4D. [cit. 10.5.2022]. Dostupné z: https://www.pix4d.com/blog/automatic-ground-control-points

[59] Ty_Brady (2017). *How To Make Ground Control Points* [online]. Pix4D. [cit. 10.5.2022]. Dostupné z: https://community.pix4d.com/t/how-to-make-ground-control-points/3805

[60] *Creating Quality GCPs for Mapping Contour Lines* [online]. Ground Control Points. [cit. 10.5.2022]. Dostupné z: https://www.groundcontrolpoints.com/mapping-contour-lines-using-gcps

[61] *Ground Control Points (GCPs) Best Practices* [online]. Rockrobotic. [cit. 10.5.2022]. Dostupné z: https://learn.rockrobotic.com/gcp-best-practices

[62] *How to Place Ground Control Points: Tutorial by Emlid* [online]. Emlid. [cit. 10.5.2022]. Dostupné z: https://community.emlid.com/t/how-to-place-ground-control-points-tutorial-by-emlid/20892/6

[63] *DIY Ground Control Points* [online]. Open Drone Map. [cit. 10.5.2022]. Dostupné z: https://community.opendronemap.org/t/diy-ground-control-points/10421/7

[64] Hoodman. *Hoodman HGCP15 UAV Ground Control Point Kit 5 Pack Aerial Targets Photogrammetry Mapping Surveying* [online]. Amazon. [cit. 10.5.2022]. Dostupné z: https://www.amazon.com/Hoodman-Control-Targets-Photogrammetry-Surveying/dp/B07Y1D2ZG7?th=1

[65] *24" GCP 6-pack Pro kit – With Carry Bag and Stakes – Ground Control Point* [online]. Madison Area Drone Service. [cit. 10.5.2022]. Dostupné z: https://widroneservice.com/product/24-gcp-6-pack-pro-kit-with-carry-bag-and-stakes-ground-control-point/

[66] (2021*) Ground control points: How many do you need (and when are checkpoints enough)?* [online]. Wingtra. [cit. 10.5.2022]. Dostupné z: https://wingtra.com/ground-control-points-how-many-do-you-need-and-when-are-checkpoints-enough/

[67] Joe Burgett. (2020). *Hack for Cheap Ground Control Points* [online]. Clemson Drone. [cit. 10.5.2022]. Dostupné z: https://www.clemsondrone.com/post/hack-for-cheap-ground-control-points

[68] Dan Wawrzyn. (2022). *How to Optimize Your Ground Control Point Placement* [online]. Propeller. [cit. 10.5.2022]. Dostupné z: https://www.propelleraero.com/blog/how-to-optimize-your-ground-control-point-placement/

[69] *Photo set manager* [online]. Pixpro. [cit. 10.5.2022]. Dostupné z: https://help.pix-pro.com/advanced-features-and-workflows/photo-set-manager.html

[70] Paolo Corradeghini. (2017). *TARGET ARTIFICIALI PER IL RILIEVO AEROFOTOGRAMMETRICO*. [online]. 3DMetrica. [cit. 10.5.2022]. Dostupné z: https://3dmetrica.it/target-artificiali-rilievo-aerofotogrammetrico/

Seznam obrázků

OBRÁZEK 1 - PŘÍKLAD VYROVNÁNÍ METODOU NEJMENŠÍCH ČTVERCŮ [1] ... 12
OBRÁZEK 2 - UKÁZKA VYROVNÁVACÍ KŘIVKY SVR [2] ... 13
OBRÁZEK 3 - UKÁZKA DECISION TREE [5] ... 14
OBRÁZEK 4 - UKÁZKA KŘIVKY LOGISTICKÉ REGRESE [6] ... 16
OBRÁZEK 5 - UKÁZKA SITUACE PRO K-NN ... 17
OBRÁZEK 6 – SVM [7] ... 18
OBRÁZEK 7 - MAPOVÁNÍ DAT DO VYŠŠÍ DIMENZE [8] ... 19
OBRÁZEK 8 - LANDMARK UMÍSTĚNÍ [8] ... 20
OBRÁZEK 9 - HRANICE KERNEL SVM [8] ... 20
OBRÁZEK 10 - PŘÍKLAD OBLASTI PODOBNÝCH POZOROVÁNÍ ... 22
OBRÁZEK 11 - UKÁZKA ROZDĚLENÍ OBLASTÍ POMOCÍ ROZHODOVACÍM STROMEM ... 23
OBRÁZEK 12 - UKÁZKA KLASIFIKACE NÁHODNÝM LESEM ... 23
OBRÁZEK 13 - UKÁZKA K-MEANS KLASTROVÁNÍ ... 25
OBRÁZEK 14 - UKÁZKA PASTI NÁHODNÉ INICIALIZACE ... 26
OBRÁZEK 15 - UKÁZKA DENDROGRAMU ... 27
OBRÁZEK 16 - UKÁZKA URČENÍ POČTU KLASTRŮ Z DENDROGRAMU ... 28
OBRÁZEK 17 - UKÁZKA PŘÍPADU S 5 KLASTRY ... 28
OBRÁZEK 18 - PŘÍKLAD 3 MOŽNÝCH ROZDĚLENÍ ... 34
OBRÁZEK 19 - ILUSTRACE PŘEDSTAVY BAG-OF-WORDS MODELU [11] ... 35
OBRÁZEK 20 - POROVNÁNÍ LIDSKÉ NEURONOVÉ SÍTĚ (NAHOŘE) S ANN (DOLE) [13], [14] ... 37
OBRÁZEK 21 - PŘÍKLAD NEURONOVÉ SÍTĚ S 1 SKRYTOU VRSTVOU SE 3 NEURONY (ŽLUTĚ) [16] ... 38
OBRÁZEK 22 - ZNÁZORNĚNÍ POSTUPU CNN [17] ... 40
OBRÁZEK 23 - UKÁZKA PŘIHLÁŠENÍ V PORTÁLU AZURE A VYTVÁŘENÍ PROSTŘEDKU [31] ... 43
OBRÁZEK 24 - UKÁZKA PROSTŘEDÍ S INFORMACEMI O PROSTŘEDKU (ÚDAJE JSOU SKRYTY) [31] ... 43
OBRÁZEK 25 - IMPORT KNIHOVEN SLUŽEB AZURE ... 45
OBRÁZEK 26 - AUTENTIZACE A VYTVOŘENÍ KLIENTA SLUŽBY AZURE ... 45
OBRÁZEK 27 - VYTVOŘENÍ A NASTAVENÍ OBJEKTU VOLITELNÝCH PARAMETRŮ ... 45
OBRÁZEK 28 - VOLÁNÍ SLUŽBY AZURE FACEAPI PRO ZPRACOVÁNÍ OBRÁZKU (SOUBOR X URL) ... 46
OBRÁZEK 29 - PŘISTUPOVÁNÍ K VÝSLEDKŮM A JEJICH ZOBRAZOVÁNÍ ... 46
OBRÁZEK 30 - ULOŽENÍ OBRÁZKU DO NOVÉHO SOUBORU ... 47
OBRÁZEK 31 - UKÁZKA DETEKCE FOTOGRAFIÍ S JEDNÍM OBLIČEJEM ... 47
OBRÁZEK 32 - UKÁZKA VÝSTUPŮ DALŠÍCH INFORMACÍ O FOTKÁCH ... 48
OBRÁZEK 33 - UKÁZKA PŘÍPADU S VÍCE TVÁŘEMI - ZDROJ PROCESOVANÉHO OBRÁZKU [33] ... 49
OBRÁZEK 34 - UKÁZKA PODOBY POZEMNÍHO VLÍCOVACÍHO BODU [34] ... 50
OBRÁZEK 35 - IMPORT NÁSTROJŮ Z KNIHOVEN CUSTOM VISION ... 52
OBRÁZEK 36 - VYTVOŘENÍ KLIENTŮ CUSTOM VISION PRO TRÉNOVÁNÍ A PREDIKCI ... 52

OBRÁZEK 37 - VYTVÁŘENÍ PROJEKTU CUSTOM VISION 53
OBRÁZEK 38 - VYTVOŘENÍ TAGU 53
OBRÁZEK 39 - UKÁZKA SLOVNÍKU SE SOUŘADNICEMI PRO REGIONY (JENOM ČÁST) 53
OBRÁZEK 40 - VYTVÁŘENÍ DÁVEK PRO POSÍLÁNÍ OTAGOVANÝCH FOTEK DO PROJEKTU 55
OBRÁZEK 41 - NAHRÁVÁNÍ OBRÁZKŮ DO PROJEKTU 56
OBRÁZEK 42 - UKÁZKA OZNAČOVÁNÍ OBRÁZKŮ Z INTERNETOVÉHO PROSTŘEDÍ CUSTOM VISION [HTTPS://WWW.CUSTOMVISION.AI] 57
OBRÁZEK 43 - PŘISTOUPENÍ K EXISTUJÍCÍMU PROJEKTU A TAGU 57
OBRÁZEK 44 - VYPISOVÁNÍ REGIONŮ 57
OBRÁZEK 45 - TRÉNOVÁNÍ PROJEKTU 58
OBRÁZEK 46 - PUBLIKOVÁNÍ ITERACE 58
OBRÁZEK 47 - DETEKCE OBJEKTŮ 59
OBRÁZEK 48 - VYTVOŘENÍ OBJEKTU KRESLENÍ A SEJMUTÍ ROZMĚRŮ TESTOVACÍHO OBRÁZKU 59
OBRÁZEK 49 - VYPISOVÁNÍ VÝSLEDKŮ A VYKRESLOVÁNÍ DETEKOVANÝCH REGIONŮ DO VÝSTUPNÍHO OBRÁZKU 59
OBRÁZEK 50 - UKLÁDÁNÍ OBRÁZKU S VYKRESLENÝMI DETEKCEMI 60
OBRÁZEK 51 - ODPUBLIKOVÁNÍ ITERACÍ A SMAZÁNÍ PROJEKTU 60
OBRÁZEK 52 - UKÁZKA VÝSTUPU KÓDU PRO SLUŽBU CUSTOM VISION 61
OBRÁZEK 53 - UKÁZKU VÝSTUPU S NULOVÝM PRAHEM 62
OBRÁZEK 54 - VÝSLEDKY CUSTOM VISION (KOLÁŽ) 63
OBRÁZEK 55 - VÝSLEDKY CUSTOM VISION (KOLÁŽ) 64
OBRÁZEK 56 - VÝSLEDKY CUSTOM VISION (KOLÁŽ) 65