VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

HODNOCENÍ UCHAZEČŮ O ZAMĚSTNÁNÍ POUŽITÍM NEUROVĚD

EVALUATION OF JOB APPLICANTS USING NEUROSCIENCE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. TOMÁŠ BANK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. IGOR SZÖKE, Ph.D.**
SUPERVISOR

BRNO 2017

Zadání diplomové práce/19493/2016/xbankt00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání diplomové práce

Řešitel: **Bank Tomáš, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Hodnocení uchazečů o zaměstnání použitím neurověd**
Evaluation of Job Applicants Using Neuroscience
Kategorie: Softwarové inženýrství

Pokyny:

1. Nastudujte a popište testy založené na neurovědě. Nastudujte a zvolte vhodný webový framework.
2. Implementujte cca 10 testů pro testování různých vlastností respondenta.
3. Sesbírejte výsledky testů v širší populaci pro získání referenčních dat.
4. Navrhněte a implementujte algoritmus, který z nasbíraných dat určí optimální pozici konkrétního jedince na základě výsledků jeho testu.
5. Zhodnoťte výsledky a navrhněte směry dalšího vývoje.
6. Vyrobte A2 plakátek a cca 30 vteřinové video prezentující výsledky vaší práce.

Literatura:

- Podle pokynů školitele

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3 ze zadání.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Szőke Igor, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 24. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato práce se zabývá možností vyhodnocení kognitivních a emocionálních vlastností člověka ve vztahu k pracovním pozicím/oborům. Podstatou je vytvoření sady testů, které bude možné použít u přijímacího řízení při rozhodování mezi kandidáty. Také navrhuje způsob klasifikace uživatele na základě absolvování sady testů, čímž se snaží poskytnout podporu při volbě vhodného zaměstnání. Práce podává stručný náhled do způsobu testování jednotlivých vlastností, popisuje návrh implementace webové aplikace, popisuje implementaci klasifikátoru založeného na neuronových sítích a také prezentuje získané výsledky.

Abstract

This thesis deals with the possibility to evaluate cognitive and emotional traits and their relations to job positions and functions. The basis is to create a set of tests which could be used during a hiring procedure while deciding among candidates. It also suggests how to classify users based on a set of tests and thereby provide support in choosing the right job. The thesis gives a brief outlook of individually tested traits, describes a proposal of a web application and its implementation, describes implementation of neural network classifier and presents obtained results.

Klíčová slova

neurovědy, web, webová aplikace, testování kandidátů, kognitivní vlastnosti, emocionální vlastnosti, change detection, stop signal, simple reaction time, flanker, tower of london, balloon analogue risk, geometric analogy, cognitve effort discounting, klasifikace, neuronové sítě, webové sokety

Keywords

neroscience, web, web aplication, candidate screening, cognitive traits, personality traits, change detection, stop signal, simple reaction time, flanker, tower of london, balloon analogue risk, geometric analogy, cognitve effort discounting, classification, neural networks, websocket

Citace

BANK, Tomáš. *Hodnocení uchazečů o zaměstnání použitím neurověd*. Brno, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Szöke Igor.

Hodnocení uchazečů o zaměstnání použitím neurověd

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Igora Szöke. Další informace mi poskytl MSc. Libor Doležel. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Bank
22. května 2017

Poděkování

Tímto bych rád poděkoval panu MSc. Liborovi Doleželovi za konzultace a poskytnuté informace ohledně neurověd a způsobů testování a také panu Ing. Igorovi Szöke, Ph.D. za vedení diplomové práce. V neposlední řadě také firmě TOPefekt s.r.o., která mi tuto práci umožnila realizovat.

Obsah

Kapitola 1

Úvod

V dnešní době, kdy je kladen velký důraz na efektivitu a výkon zaměstnanců je pro zaměstnavatele velice důležité si umět z potenciálních kandidátů vybrat toho nejvhodnějšího. Přijetí nevhodného člověka s sebou nese množství problémů. Jestliže je výběr špatně proveden, může to mít vážné dopady, jak pro společnost, tak pro samotného zaměstnance. Nábor špatného zaměstnance může vést ke snížení výkonnosti kolektivu nebo týmu, finanční ztrátě a samotný zaměstnanec může trpět syndromem vyhoření. Statistiky [10] uvádějí, že asi 20% zaměstnanců není vhodně vybráno na pracovní pozice, což má za následek nízkou výkonnost samotného zaměstnance. Toto z pohledu firmy může znamenat nemalé finanční ztráty[32, 15].

Jsou zde tudíž velké podněty proto, aby společnosti nabíraly vhodné zaměstnance a předešly tak monetárním ztrátám, a na druhou stranu zvýšily výnosnost investice spojené s náborem zaměstnanců. Tradičními vyhodnocovacími nástroji, které společnosti využívají, jsou dotazníky orální, vizuální a další testy, které pomáhají stanovit, zdali je daný kandidát vhodný na danou pracovní pozici nebo ne. Problémem těchto nástrojů však je, že výsledek dokáže být subjektivně ovlivněn samotným účastníkem testu.

Tato práce pojednává o řešení tohoto problému a to spojením neurověd a informačních technologií. Výsledná aplikace by měla pomoci společnostem ke správné identifikaci a výběru nejvhodnějšího kandidáta na danou pozici. Každý z kandidátů podstoupí sadu vizuálních úloh, založených na výzkumech z oblasti neurověd a psychologie, které vyhodnocují kognitivní a emocionální vlastnosti člověka a tedy zajišťují objektivní předvídatelnost výkonnosti uchazeče. Výsledkem vyhodnocení by měl být souhrn konkrétních unikátních vlastností uchazeče. Toto poskytne zaměstnavateli lepší a objektivnější možnost při rozhodování, kterého z kandidátů vybrat na danou pozici, jelikož aplikace eliminuje subjektivitu, jak nejvíce to lze.

Na druhou stranu aplikaci mohou využít i lidé hledající tu pravou pracovní pozici přesně podle svých schopností a vlastností. Po absolvování všech úloh aplikace na základě výsledků klasifikuje člověka do nejpravděpodobnějšího oboru či pracovní pozice (záleží na konkrétnosti dat).

Práce je realizována ve spolupráci s firmou TOPefekt s.r.o., která by se tímto zajímavým projektem ráda pokusila nabídnout zaměstnavatelům doplňující možnost ke klasickému přijímacímu řízení a tím i tento proces zefektivnit, jelikož v současné době není na trhu mnoho podobných aplikací.

Kapitola 2

Testy založené na neurovědě

Člověk při vykonávání každodenních pracovních činností využívá určité části mozku, které jsou vázány k jednotlivým mentálním procesům. Tyto procesy mohou být například plánování, změna úloh, kreativita a mnohé další. Různá odvětví neurověd jsou schopna objektivně zodpovědět, které části mozku jsou v průběhu těchto procesů aktivní, a tedy jaké jsou rozdíly mezi jednotlivci. Výsledky takovýchto zkoumání poté ukazují daleko přesněji, než ostatní nástroje, například zdali ve srovnání s jinými lidmi, jsou schopni provádět více úloh najednou.

Ve většině neurovědeckých výzkumů se při různých činnostech člověka aplikuje jeden nebo více nástrojů pro měření mozku (fMRI[1], EEG[2] a další), které se snaží zjistit, jak mozek funguje, a jak reaguje na rozdílné stimuly. Jsou zde ale i další možnosti, které se na první pohled nemusí jevit jako příbuzné těm nejznámějším z oblasti neurověd. Vycházejí z kognitivních neurověd a psychologie, zkoumají jednotlivé mentální schopnosti člověka a dají se tedy aplikovat přímo ve výběrovém řízení. Z převážné většiny se jedná o vizuální úlohy [34].

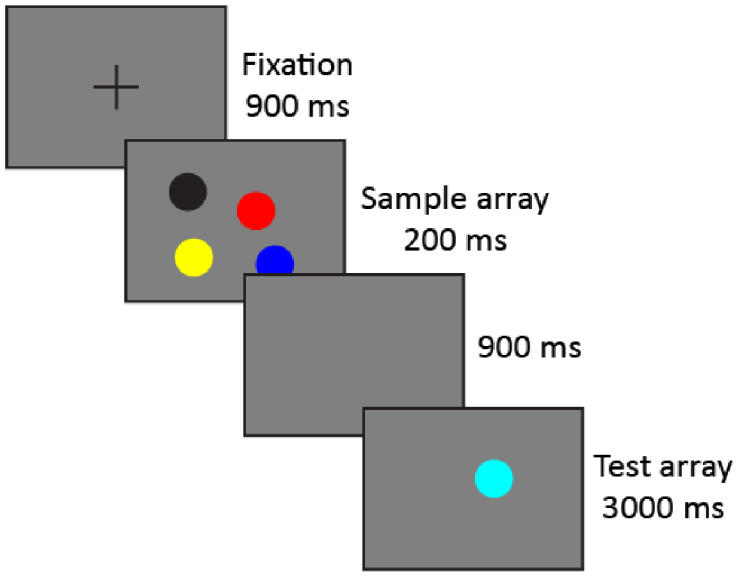
Většina těchto nástrojů dokáže objektivně a spolehlivě měřit kognitivní a emocionální vlastnosti člověka, značící jak se člověk chová v různých situacích. Tyto vlastnosti zahrnují například: exekutivní funkce, riskování, odolání pokušení, soustředění se, uvažování, analyzování a mnohé další [6].

2.1 Change detection

Tato úloha má za úkol zjistit, jak velkou má jedinec pracovní paměť, která slouží k aktivnímu udržování informace potřebné při probíhající úloze. Dále jak člověk dokáže pracovat a manipulovat s informacemi, hledat souvislosti a další. Samotná pracovní paměť je součástí exekutivních funkcí, které jsou potřebné v každodenním životě.

Test se skládá ze čtyř kroků. V prvním kroku je zobrazen pouze křížek uprostřed obrazovky, na který se uchazeč soustředí (tzv. fixation krok). V dalším kroku se krátce zobrazí sada objektů (kruhů), kde každý má jinou barvu, úkolem je zapamatovat si jednotlivé barvy. Následuje opět fixation krok ovšem bez křížku, po kterém přichází poslední a nejdůležitější krok, kde je zobrazen pouze jeden objekt (na původní pozici). Úkolem respondenta je rozhodnout, zda u zobrazeného objektu došlo ke změně barvy či nikoli.

[1] Funkční magnetická rezonance
[2] Elektroencefalogram



Obrázek 2.1: Ukázka change detection test.

Change detection měří kapacitu vizuální pracovní paměti (visual working memory - WM), která velmi silně koreluje s celkovými kognitivními schopnostmi. Předpokladem je, že člověk má v paměti K míst (WM se skládá z K slotů) a každé z nich ukládá informaci. Je-li účastníkovi zobrazeno N informací k zapamatování, pouze K z nich může být uloženo do WM. Logicky tedy pokud informace není v paměti, pak účastník hádá. Vizuální pracovní paměť lze definovat jako aktivní udržování vizuální informace, která je potřebná k pokračujícímu vývoji nějakého úkolu. Mezi jedinci se hodnota WM liší, ale bylo zjištěno, že se kapacita pohybuje kolem čísla 2–4 [28, 31].

2.2 Stop signal

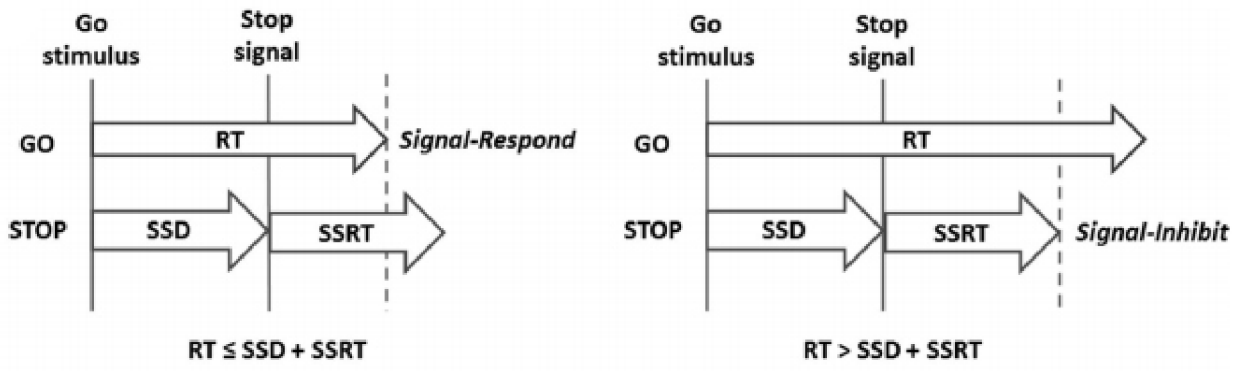
Stop signal měří inhibitory control (kontrola potlačení), značící jak se člověk dokáže ovládat a potlačit určité (třeba i nevhodné) chování, a také ovládání se po emocionální stránce. Schopnost potlačit určité chování je důležitá při vykonávání jakéhokoliv úkolu v měnícím se prostředí. Absence této schopnosti se také pojí s poruchami jako například, porucha pozornosti, obsesivně-kompulzivní porucha a další [14].

Úkolem účastníka je vybrat a vykonat reakci na jeden ze dvou zobrazených stimulů (šipka doleva nebo doprava), přičemž se snaží zareagovat co nejrychleji a nejpřesněji (*GO* signál). Příležitostně se ovšem vyskytne takzvaný *STOP* signál, který vyžaduje, aby účastník reakci potlačil.

Test je složen ze dvou opakujících se kroků. Prvním je opět fixace očí na střed obrazovky, za níž následuje zobrazení jednoho ze dvou stimulů (čtverec nebo trojúhelník), na který účastník reaguje. Náhodně se přes stimul může po určité době zobrazit *STOP* signál (červený kruh), který účastníkovi říká, aby reakci potlačil, tedy snažil se nezareagovat. Časová prodleva *SSD* (stop signal delay) mezi samotným stimulem a *STOP* signálem je různá a odvíjí se mimo jiné i od předchozích potlačení účastníka.

Obrázek 2.2: Ukázka stop signal testu.

Stejně jako se měří rychlost reakce, lze měřit i rychlost potlačení reakce. Délka potlačení reakce je však implicitní, a tudíž nejde měřit přímo. Potlačení reakce se měří pomocí *SSRT* (stop signal reaction time). *SSRT* tedy měří, jak dlouho uživateli trvá potlačit reakci, pokud je zobrazen *STOP* signál. Průběh úlohy lze znázornit jako závod mezi *GO* procesem, který je spuštěný uvedením *GO* signálu a *STOP* procesem, který je spuštěn *STOP* signálem. Pokud účastník úspěšně potlačil reakci, jedná se o *signal-inhibit*, na druhou stranu *signal-respond* značí, že účastník zareagoval, kdy neměl. Reakce je dána reakčním časem *RT* (reaction time) a je úspěšně potlačena pokud je *STOP* proces dokončen před *GO* procesem. To znamená, že *signal-inhibit* nastane, když je $RT > SSD+SSRT$. Na druhou stranu reakce je vykonána, když je *GO* proces dokončen před *STOP* procesem [24].



Obrázek 2.3: Znázornění závodu mezi GO a STOP procesy [24].

2.3 Flanker

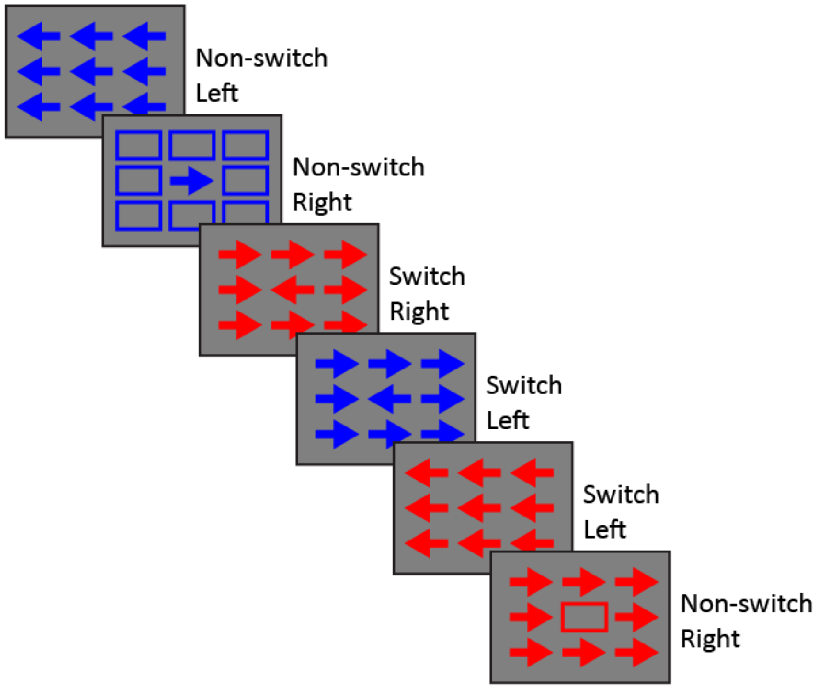
Díky této úloze je u účastníka možné měřit více vlastností najednou. Selektivní pozornost, která je spojená i s inhibitory control, multitasking (switching task) a učení se z vlastních chyb. O selektivní pozornosti lze zjednodušeně říci, že se jedná o schopnost našeho mozku filtrovat to co doopravdy vnímáme. Switching task udává, jak účastníci zvládají změnu mezi dvěma úlohami.

Účastníkovi je zobrazena série šipek (modrých nebo červených), podle nichž se snaží reagovat (stisknout šipku doleva či doprava) co nejrychleji a nejpřesněji. Pro modré šipky je klíčový směr, který udává šipka ve středu. Pro červené je to právě naopak, tedy správný směr ukazují všechny okolní šipky (okolo středové). Stimuly (skupina šipek) se dělí na tři typy.

- Congruent - všechny šipky ve skupině jsou ve stejném směru
- Incongruent - středová šipka je vždy oproti ostatním otočena
- Neutral - zobrazeny jsou jen šipky, které jsou pro danou barvu podstatné, ostatní jsou nahrazeny obdélníky

Důležité je také dělení na *non-switch* a *switch*, které udává, jestli nastala změna v barvě oproti předchozímu stimulu.

Opět jsou mezi jednotlivými kroky se stimuly zobrazeny kroky s fixačním křížem na středu obrazovky, aby pomohly udržet účastníkův pohled na správné pozici. Každý stimul je zobrazen maximálně na 1500 milisekund nebo do doby než respondent zareaguje.



Obrázek 2.4: Ukázka testu flanker s popisky typů a očekávaných reakcí.

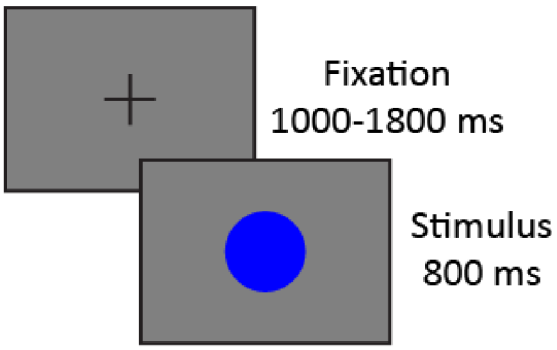
Při měření selektivní pozornosti předpokládáme, že *incongruent* stimuly způsobují interferenci, která vede k pomalejším a méně přesným reakcím, a zároveň zohledňujeme jen ty,

které jsou *non-switch* [19]. Pokud změříme reakční časy zvlášť pro *switch* a *non-switch* a následně je porovnáme, můžeme určit, jak účastníci zvládají změnu mezi stimuly. Při tomto přechodu obvykle dochází k podstatnému zpomalení reakcí a větším chybám, z tohoto důvodu provádíme tohle měření [30].

2.4 Simple reaction time

Tato úloha měří rychlost zpracování informace, která udává jak rychle a efektivně dokáže člověk zpracovat a vykonat kognitivní operace, a také určuje míru pozornosti. Je dokázáno, že lidé s lepšími výsledky inteligenčních testů mají obvykle rychlejší a stabilnější rychlosti reakcí [17].

Podstatou je tedy snaha účastníka reagovat na stimul co možná nejrychleji, reakce by měla proběhnout ukazováčkem dominantní ruky. V této úloze je stimulem modrý kruh. Opět se zde střídají dva kroky, kde nejdříve přichází fixace (jako v předchozích úlohách) a poté stimul. Pokud účastník reaguje před nastoupením stimulu, je krok s fixací zopakován, abychom se vyhnuli situaci, kdy respondent jen kliká rychle za sebou. Důležité je, že doba před nastoupením stimulu je vždy různá a účastník to tedy nemůže odhadnout.



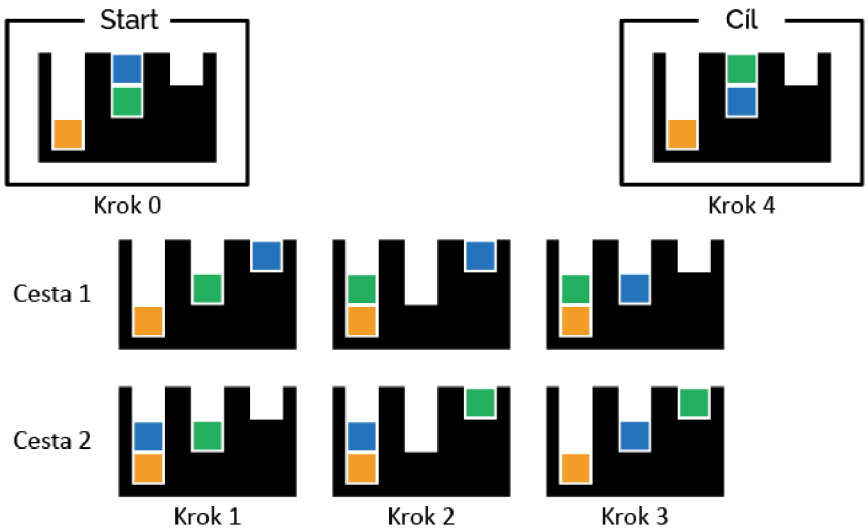
Obrázek 2.5: Ukázka testu simple reaction time.

2.5 Tower of London

Tower of London měří u respondentů schopnost plánování, která udává jak rychle a efektivně dokáže člověk plánovat, přičemž je zapojena i pracovní paměť. Ve spojení s touto úlohou je dokázáno, že dospělí lidé trpící nějakou z neurobiologických poruch dosahují o něco horších výkonů [18].

Úloha vychází ze známé hádanky Hanoiské věže. Máme k dispozici tři zásobníky, které obsahují objekty (čtverce). Úkolem uchazeče je přemístit objekty v zásobnících tak, aby se dostal ze startovního uspořádání do cílového a to v co nejmenším počtu kroků a nejkratším čase. Přičemž přemístit, lze vždy jen jeden objekt, který musí být na vrcholku zásobníku. Tato úloha lze vždy vyřešit v určitém minimálním počtu kroků a obvykle má alespoň dvě možná řešení [33].

Pro měření strategie plánování při řešení problému slouží takzvaný ”initial thinking time” (ITT), který udává dobu uplynulou od zobrazení problému po vykonání prvního kroku, tedy dobu po kterou člověk přemýšlí. K tomuto se váže i doba řešení zaznamenaná od prvního kroku (celková doba - ITT).



Obrázek 2.6: Ukázka Tower of London.

Dále se také měří účinnost řešení, tedy kolik kroků účastník udělá při řešení úlohy. Schopnost a účinnost plánování určuje počet kroků navíc od ideálního řešení. Platí, že čím méně kroků, tím lepší je respondent v plánování.

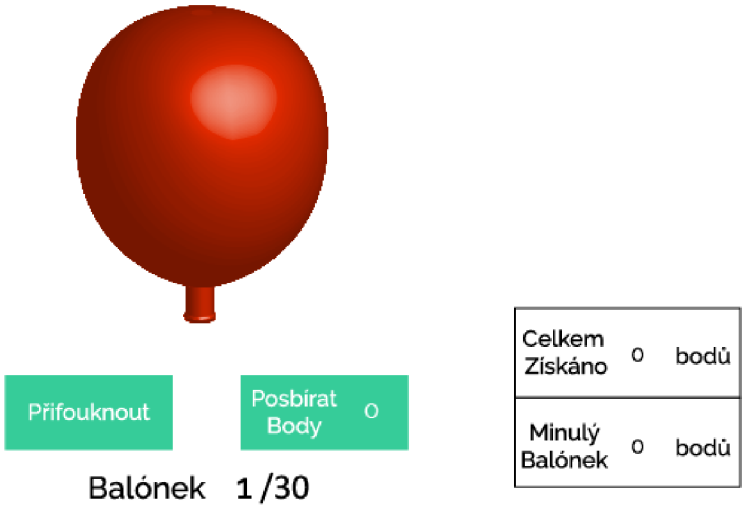
2.6 Balloon analogue risk

Tato úloha se zabývá rizikovým chováním respondentů. Toto představuje chování jenž zahrnuje určité nebezpečí nebo riziko škody jím způsobené, ale také poskytuje příležitost získat nějakou odměnu [11].

Účastníkovi je zobrazen balónek, který má za úkol nafukovat tlačítkem ”přifouknout”. S každým přifouknutím obdrží jeden bod do dočasného banku, body z dočasného banku může převést tlačítkem ”posbírat” do trvalého banku, čímž se zároveň posune na další balónek. Pokud balónek při nafukování praskne účastník ztrácí body z dočasného banku a posouvá se dále. Balónky mají tři různé barvy, pro které platí různé pravděpodobnosti prasknutí.

- Červený - nafukuje se nejrychleji
- Žlutý
- Modrý - nafukuje se nejpomaleji (má nejmenší pravděpodobnost prasknutí)

S každým přifouknutím se pravděpodobnost prasknutí balónku zvyšuje. Úkolem uchazeče je odhadnout riziko a pokusit se nasbírat do trvalého banku co největší počet bodů.



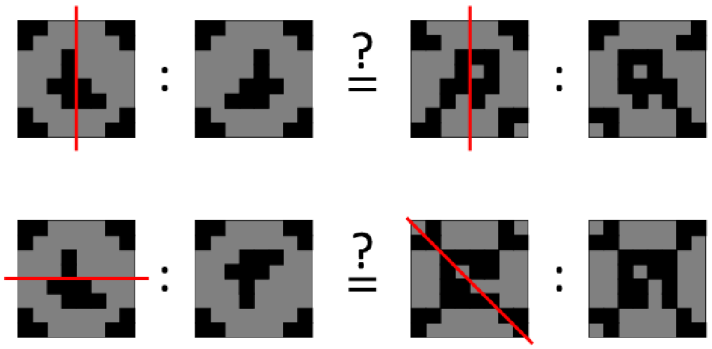
Obrázek 2.7: Ukázka úlohy Balloon analogue risk.

Počet vybuchlých balónků s počtem úspěšných nafouknutí (nafouknutí při kterém balónek nepraskl) jsou ukazatele při vyhodnocení riskování uchazeče. Celkový dosažený počet bodů udává, jak moc je člověk efektivní při rizikovém rozhodování.

2.7 Geometric analogy

Geometric analogy měří schopnost fluidního uvažování. Fluidní uvažování představuje, jak člověk dokáže uvažovat jak po induktivní, tak deduktivní stránce, řešit problémy v neobvyklých situacích bez ohledu na předešle získaných znalostech. Výzkumy ukazují, že lidé s vyšší fluidní inteligencí dokáží tuto úlohu vyřešit rychleji a to i s větší přesností [8].

Účastníkovi jsou zobrazeny dva páry vzorů (zdrojový a cílový). V páru je vždy jeden vzor výchozí, vůči kterému je druhý zrcadlen (podle vertikální osy, horizontální osy nebo podle jedné z diagonál). Podstatou této úlohy je rozhodnout, zda má cílový pár analogické zrcadlení s párem zdrojovým. Tudíž musí respondent rozhodnout zda $A : A' = B : B'$ a to za pouhých 15 sekund.



Obrázek 2.8: V prvním případě jsou oba páry zrcadleny vertikálně - analogicky. Ve druhém je zdrojový pár zrcadlen podle horizontální osy a cílový podle diagonály.

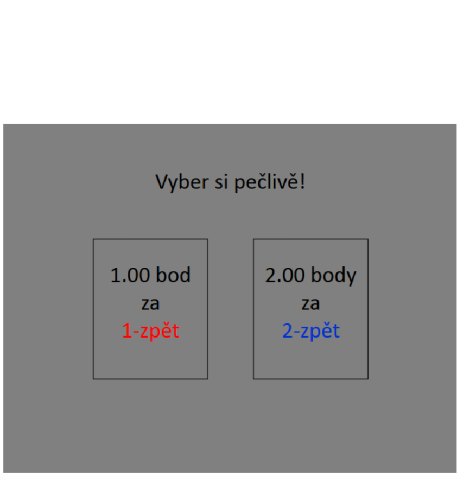
Diagonální zrcadlení je obtížnější, a proto při výpočtu odlišujeme dvě kategorie zrcadlení. Do první spadá vertikální i horizontální a do druhé diagonální zrcadlení. Na základě

přesnosti odpovědí uchazečů lze změřit fluidní uvažování. K měření rychlosti uvažování stačí měřit dobu reakce respondentů [16].

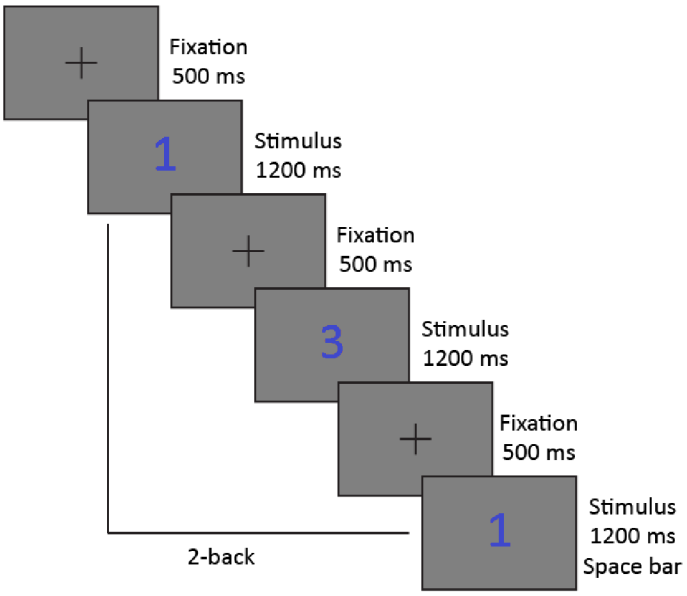
2.8 Cognitive effort discounting

Tato úloha se zabývá kognitivním úsilím uchazeče, tedy jak moc je uchazeč ochotný vynaložit úsilí. Zároveň s tím je testována i jeho pracovní paměť. Kognitivní úsilí je velice důležité pro fungování v moderní společnosti, jelikož je nezbytné pro komplexní plánování a rozhodování [5].

Cílem je zjistit, zda si člověk vybírá spíše lehčí úkol nebo těžší. Uchazeči je promítnuta série číslic (jedno číslo po druhém) a jeho úkolem je detekovat, zda se aktuální číslo shoduje s číslem o několik míst zpět (o 1–4 místa zpět). Vždy má na výběr mezi *1-zpět* a *N-zpět* (2, 3, 4), přičemž jednodušší volba *1-zpět* má proměnlivé bodové ohodnocení začínající na 1.00 bodu a *N-zpět* je hodnoceno fixními 2.00 body. Rozhodnutí *1-zpět vs. N-zpět* je pro každé N v úloze třikrát, což nám dává devět rozhodnutí uchazeče. Pokud si účastník zvolí těžší obtížnost ($N > 1$), pak se na příští volbě, mezi stejnými obtížnostmi, zvýší počet bodů o polovinu a naopak (pro snazší volbu se o polovinu sníží) [7].



(a) Výběr obtížnosti.



(b) Ukázka 2-zpět úlohy.

Obrázek 2.9: Příklad průběhu úlohy.

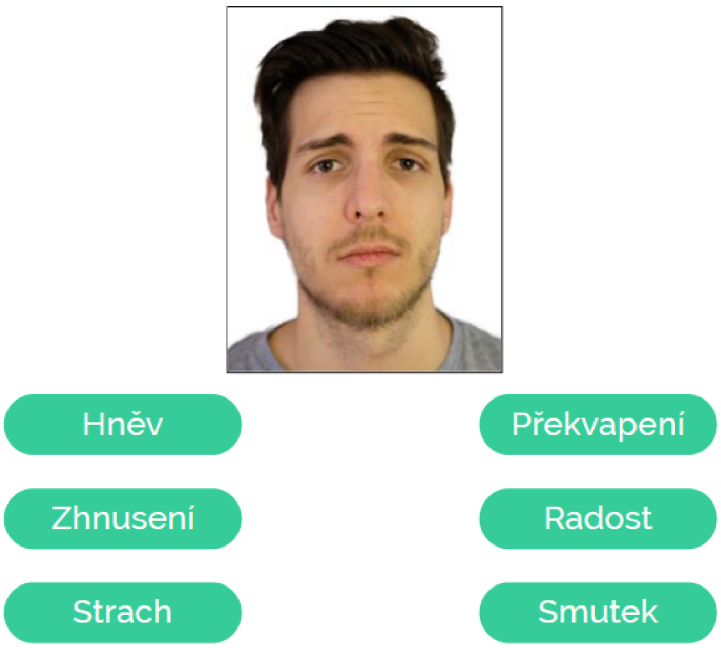
Tudíž tato úloha (výběr mezi lehčí a těžší volbou) kvantifikuje úsilí jako náklad, neboli jak moc je uchazeč ochoten zapojit se do více náročného úkolu. Pokud bychom ponechali odměny vždy fixní, můžeme zkoumat tzv. body lhostejnosti mezi jednotlivými rozhodnutími. Vedle toho je možné stanovit i vnitřní motivaci uchazeče k zapojení se v namáhavé činnosti [7].

2.9 Emotion recognition

Jak název napovídá tato úloha zkoumá schopnost rozpoznání emocí obličeje. Schopnost přesně a efektivně rozpoznat výrazy obličeje je zásadní už při typických mezilidských in-

terakcích a podporuje pro-sociální chování jednotlivců. Je známo, že deficity rozpoznávání výrazů mohou být součástí různých poruch jako je například schizofrenie [29].

Princip této úlohy je velice jednoduchý, uchazeči je na krátký okamžik prezentován obličej s jednou ze základních emocí (strach, hněv, znechucení, překvapení, radost a smutek). Po zobrazení obličeje má uchazeč cca 10 vteřin na rozhodnutí, o kterou z emocí se jedná.



Obrázek 2.10: Ukázka úlohy rozpoznání emocí.

2.10 Prisonner's dilemma with risk

Tato úloha je známá zejména z teorie her a označuje situaci, kdy dva individuální jedinci mají na výběr spolupracovat či nespolupracovat. Úloha tedy měří míru kooperace a důvěry ve spoluhráče. Míra důvěry jež máme v ostatní lidi je velice důležitá i v každodenním životě, jelikož přímo ovlivňuje rozhodnutí, která děláme a současně s tím také naši ochotu spolupracovat [13].

Úloha implementuje tzv. iterativní verzi tohoto problému a je pojatá formou hry, ve které je účastník spojen s druhým hráčem. Hra má 12 kol, v každém z nich mají oba hráči k dispozici 10 mincí z nichž určitou část (1–10 mincí) musí svěřit spoluhráči. Jakmile si oba hráči navzájem svěří mince nastává pro každého z nich rozhodnutí, zda svěřené mince vrátit nebo si je ponechat (spolupracovat či nespolupracovat), přičemž vrácené mince budou zdvojnásobeny. Hlavním cílem účastníka je maximalizovat celkový zisk mincí. Účastníkovi je úloha předkládána jako hra dvou hráčů, ve skutečnosti je však druhý hráč pouze simulován. Rozhodování ze strany simulovaného hráče je prováděno na základě kombinace strategií TFT (tit for tat) a TFTT (tit for two tats). Ty říkají:

- TFT - pokud uživatel nespolupracuje, pak mu to oplatím
- TFTT - pokud nespolupracuje dvakrát za sebou, pak mu to oplatím

Přidělování mincí závisí na počtu mincí svěřených účastníkem a jeho spolupráci/nespolupráci.

Podle počtu svěřených mincí a podle míry spolupráce účastníka s druhým hráčem je možné změřit míru důvěry i kooperace. Z analýzy vývoje přidělovaných mincí lze odvodit jak je uchazeč ochoten riskovat při budování důvěry i jak se učí z interakce s ostatními. Stejně tak z analýzy všech rozhodnutí uživatele je možné odhadnout, zda je spíše sobečtějším nebo týmovým hráčem.

2.11 Shrnutí

Úlohy popsané v předchozích podkapitolách jsou založené na neuropsychologickém/neurovědeckém výzkumu. Vizuální formou testují člověka a ve většině případů zaznamenávají jeho reakce, reakční čas i přesnost odpovědí. Ze všech těchto zaznamenaných hodnot je možné vypočítat specifické metriky. Na základě toho lze následně určit různé kognitivní a emocionální vlastnosti testovaného člověka. Test složený z těchto úloh dokáže podat objektivní přehled o konkrétních vlastnostech uchazeče a tím zefektivnit přijímání nových zaměstnanců.

Každá z úloh měří různé vlastnosti, většina úloh pouze jednu, ale některé i více. Na druhou stranu určité vlastnosti jako například kreativita a schopnost učení jsou průnikem několika metrik napříč úlohami. Všechny úlohy dohromady tedy utváří jakýsi celkový obrázek o uchazeči.

Ve většině úloh je podstatným prvkem rychlost zobrazení objektů (stimulů). Stimuly musí být na obrazovce zobrazeny na přesný časový úsek (typicky v řádech stovek milisekund). Je potřeba tedy zajistit stejný průběh úloh (z pohledu délky zobrazení) pro všechny účastníky. Pokud by docházelo k latenci při zobrazování stimulů, měření tím budou zkreslena.

Při získávání referenčních dat vyplnilo všech 10 úloh pouze 58 uživatelů (z celkem 220 zaregistrovaných), proto budeme dále pracovat pouze s výsledky 8 úloh, které absolvovalo 108 lidí. Z nich ovšem použijeme pouze čtyři nejpočetnější skupiny: IT (22), administrativa (18), finance (16) a management (9), ostatní byli roztroušeni přes různé obory, vice viz 6.

Kapitola 3

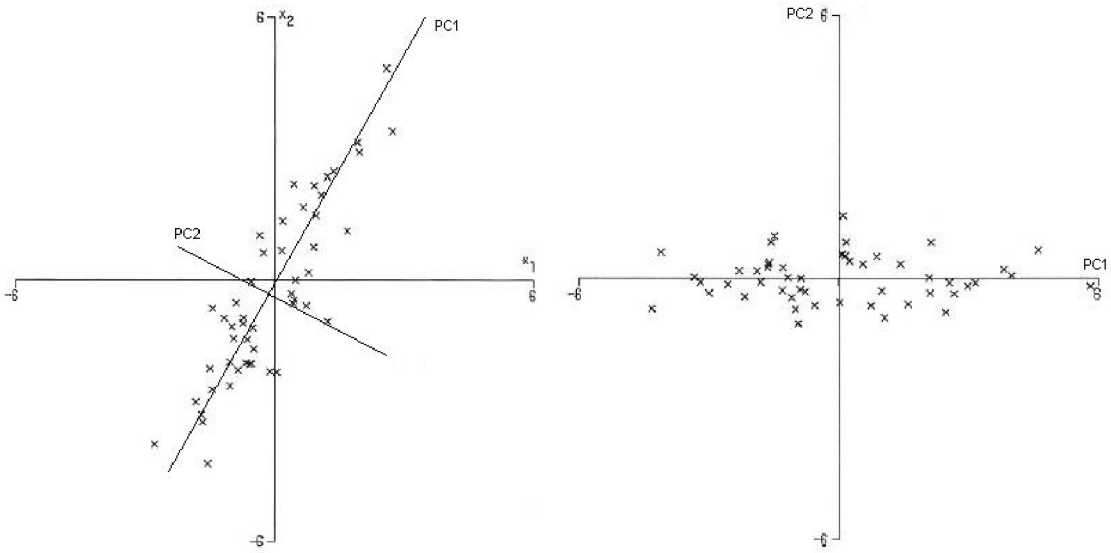
Určení optimální pracovní pozice

Na základě výsledků úloh popsaných v předchozí kapitole by se měla aplikace, kromě vyhodnocení jednotlivých vlastností, pokusit zařadit konkrétního jedince na optimální pracovní pozici. Abychom mohli tohoto dosáhnout, je potřeba získat větší množství referenčních dat od lidí již pracujících v různých pracovních oborech či přímo na konkrétních pozicích. Idea je v tomto případě taková, že uživatel po absolvování testu získá přehled pro něj ideálních pracovních pozic či oborů (záleží na konkrétnosti referenčních dat) s procentuálním hodnocením.

K tomuto účelu lze využít metody strojového učení. Jelikož máme k dispozici referenční data, která poslouží pro učení, jedná se o tzv. supervised learning neboli učení s učitelem. Cílem je zařadit uživatele na základě jeho výsledků do jedné z K tříd, případně do N z K tříd, jedná se tedy o úlohu klasifikace. Před samotnou klasifikací bude vhodné provést redukci dimensionality, a to především za účelem lepšího zobrazení vstupních dat, z toho důvodu jsou v následující podkapitole představeny metody *PCA* a *LDA*.

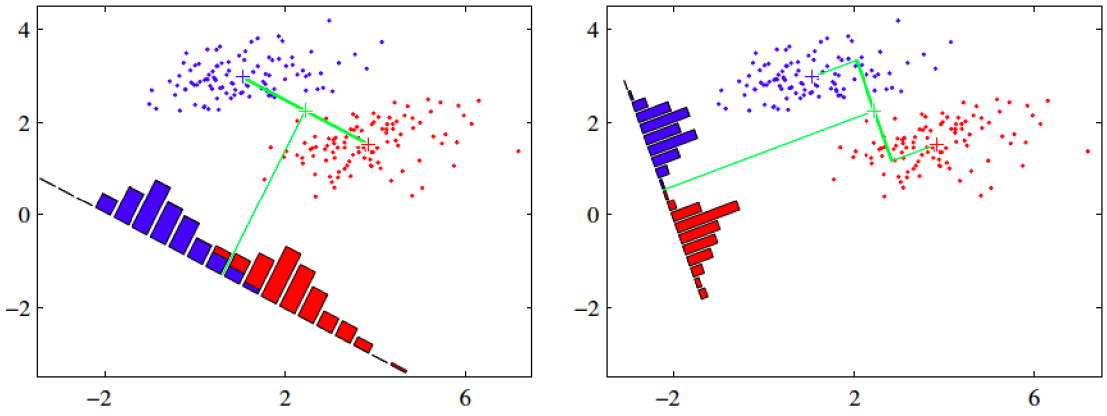
3.1 Redukce dimensionality

Jednou z nejpoužívanějších metod je *analýza hlavních komponent* (Principal Component Analysis - PCA). Tato technika slouží také k identifikování vzorů v datech, přičemž se snaží o vyjádření dat se zvýrazněním podobností a odlišností v nich. Cílem *PCA* je zjednodušení popisu lineárně závislých tj. korelovaných znaků, tedy transformace dat z původních proměnných x_i do menšího počtu latentních proměnných y_j, $j = 1, ..., m$. [20, 21]. Latentní proměnné (hlavní komponenty) jsou lineární kombinace původních proměnných. První hlavní komponenta y_1 popisuje největší část proměnlivosti původních dat, y_2 zase největší část proměnlivosti neobsažené v y_1 atd. První hlavní komponenta je tedy takovou lineární kombinací vstupních proměnných, která zahrnuje největší proměnlivost mezi všemi lineárními kombinacemi. Počet hlavních komponent (m) je roven počtu objektů nebo počtu proměnných (tomu menšímu z nich). Pro redukci na k dimenzí vezmeme pouze prvních k hlavních komponent [21].



Obrázek 3.1: Ukázka zobrazení dat s nalezenými hlavnimi komponentami (vlevo) a transformovaná data podle dvou hlavních komponent (vpravo) [20].

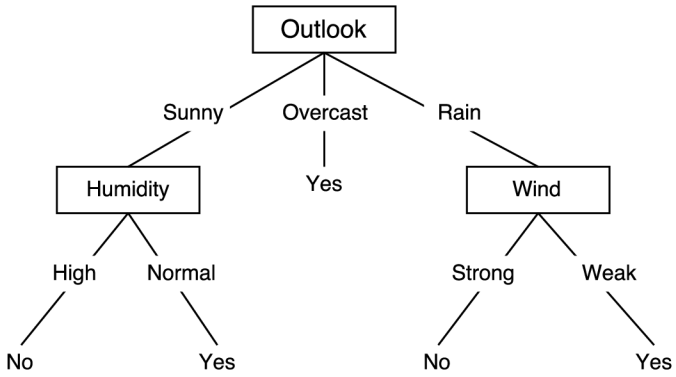
Obdobnou metodou jako je *PCA* je i *lineární diskriminační analýza* (Linear Discriminant Analysis - LDA). Ta se na rozdíl od *PCA* v datech snaží najít takový směr, ve kterém jsou obsažné tříd maximálně odděleny a zároveň snížit rozptyl v každé z tříd. Tato metoda je opět založena na hledání lineární kombinace proměnných, která nejlépe odděluje třídy jednu od druhé.



Obrázek 3.2: Ukázka zobrazení dvou tříd spolu s histogramy pro danou projekci. Vlevo vidíme překrytí obou tříd, kdežto vpravo pěkné oddělení tříd po aplikaci *LDA* [12]

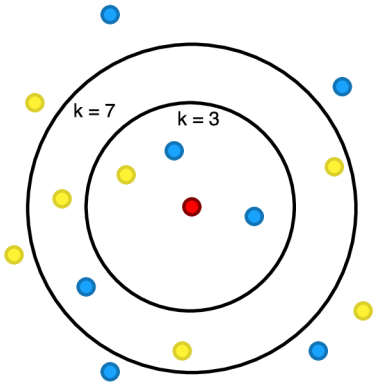
3.2 Klasifikační metody

Ke klasifikaci existuje mnoho metod, jednou z nich jsou například *Rozhodovací stromy*. Rozhodovací strom tvoří sada hierarchicky uspořádaných rozhodovacích pravidel, kde každý uzel stromu testuje určitý atribut vstupu, přičemž každá větev uzlu odpovídá jedné z možných hodnot atributu. Proces klasifikace vstupního objektu probíhá průchodem stromu od kořene k listu. Klíčovou otázkou algoritmu je, jak vybrat vhodný atribut pro větvení stromu [23].



Obrázek 3.3: Jednoduchý příklad rozhodovacího stromu, rozhodnutí zda jít hrát tenis [23].

Další možností je metoda *k-nejbližších sousedů*. Zde jsou prvky chápány jako body v n-rozměrném prostoru atributů. Nejbližší sousedé prvku jsou nalezeni pomocí vzdálenosti mezi prvky (např. Eukleidovská či Hammingova vzdálenost). Při klasifikaci nového prvku je nalezeno právě K prvků, jejichž vzdálenost je nejmenší k právě klasifikovanému. Prvek je zařazen do třídy jež má u těchto K vybraných nejčastější zastoupení [23].



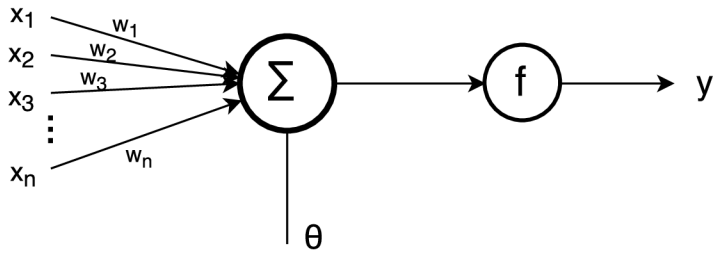
Obrázek 3.4: Demonstrace k-NN pro $k = 3$ bude vstup klasifikován jako modrý a pro $k = 7$ jako žlutý.

Pro realizaci v aplikaci však byla vybrána metoda umělých neuronových sítí. Neuronové sítě mají obecně výbornou schopnost generalizace a nemají problém s vysoko dimenzionálními daty (vstupem našeho klasifikátoru budou výstupy ze všech úloh). Za umělou neuronovou síť se obecně považuje taková struktura pro distribuované paralelní zpracování dat, která se skládá z jistého (obvykle velmi vysokého) počtu vzájemně propojených výkonných prvků. Každý výkonný prvek transformuje vstupní data na výstupní podle jisté přenosové funkce a přitom též může uplatnit obsah své lokální paměti [22].

Výkonným prvkem se zde rozumí jeden neuron, jenž zpracovává hodnoty na svém vstupu podle následujícího vztahu.

$$y = f(\sum_{i=1}^{N} w_i x_i + \Theta), \tag{3.1}$$

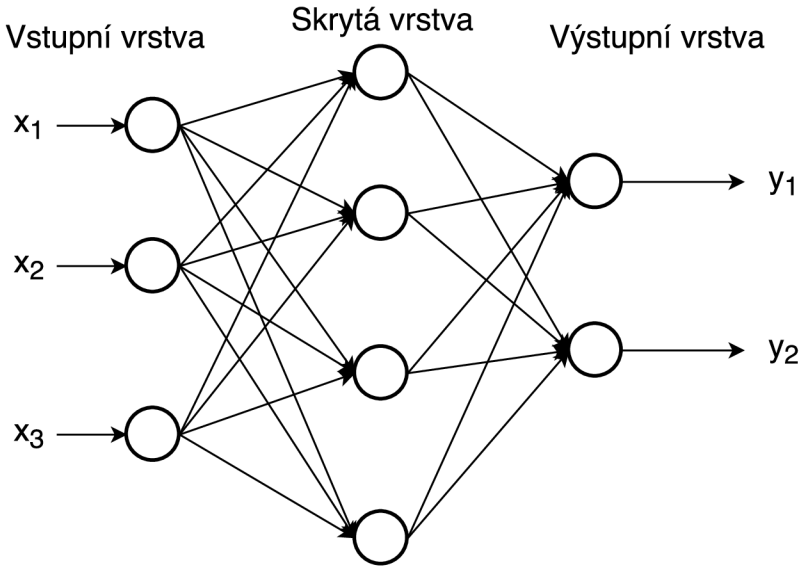
kde f je přenosová funkce neuronu, x_i jsou vstupy neuronu, w_i jsou váhy vstupů a Θ je práh neuronu. Suma vážených vstupů s přičteným prahem (výraz v závorce) se též označuje jako vnitřní potenciál neuronu. Aktivační (přenosová) funkce neuronu převádí vnitřní potenciál neuronu do definovaného oboru výstupních hodnot. Nejčastější aktivační funkce jsou lineární funkce, sigmoida, skoková funkce, hyperbolická tangenta a mnoho dalších.



Obrázek 3.5: Činnost jednoho neuronu.

Při učení s učitelem se neuronová síť učí srovnáváním aktuálního výstupu s výstupem požadovaným a nastavováním vah tak, aby se dosáhlo co nejmenšího rozdílu mezi skutečným a požadovaným výstupem [22]. Různých typů umělých neuronových sítí je mnoho (například Perceptron, Kohonenova síť, Hopfieldova síť, ART síť atd.). V této práci však budou uvažovány pouze vícevrstvé sítě.

Jak název napovídá vícevrstvé sítě se skládají z vrstev neuronů, vrstvy jsou mezi sebou propojeny tak, že neuron z jedné vrstvy je propojen se všemi neurony vrstvy sousední, přitom v rámci jedné vrstvy mezi neurony nejsou žádné vazby.



Obrázek 3.6: Vícevrstvá neuronová síť s jednou skrytou vrstvou.

Počet neuronů vstupní vrstvy typicky odpovídá počtu vstupů, podobně počet neuronů výstupní vrstvy je ovlivněn kódováním výstupu (například při klasifikaci může odpovídat počtu tříd, do kterých klasifikujeme). Mezi vstupní a výstupní vrstvou může být obecně několik skrytých vrstev. Je důležité vhodně zvolit počet neuronů ve skrytých vrstvách, při nízkém počtu neuronů nemusí mít neuronová síť kapacitu k naučení daného problému, naopak při použití příliš mnoha neuronů může dojít k přeučení (overfitting).

Učení neuronové sítě lze formálněji popsat následujícím způsobem. Uvažujme neuronovou síť s n vstupy a m výstupy, po této síti požadujeme realizaci zobrazení ϕ z množiny vstupních vektorů $X \subset R^n$ do množiny výstupních vektorů $Y \subset R^m$. Aproximaci tohoto zobrazení neuronová síť provede funkcí.

$$\vec{y} = f(\vec{x}, \vec{w}, \vec{\Theta}), \tag{3.2}$$

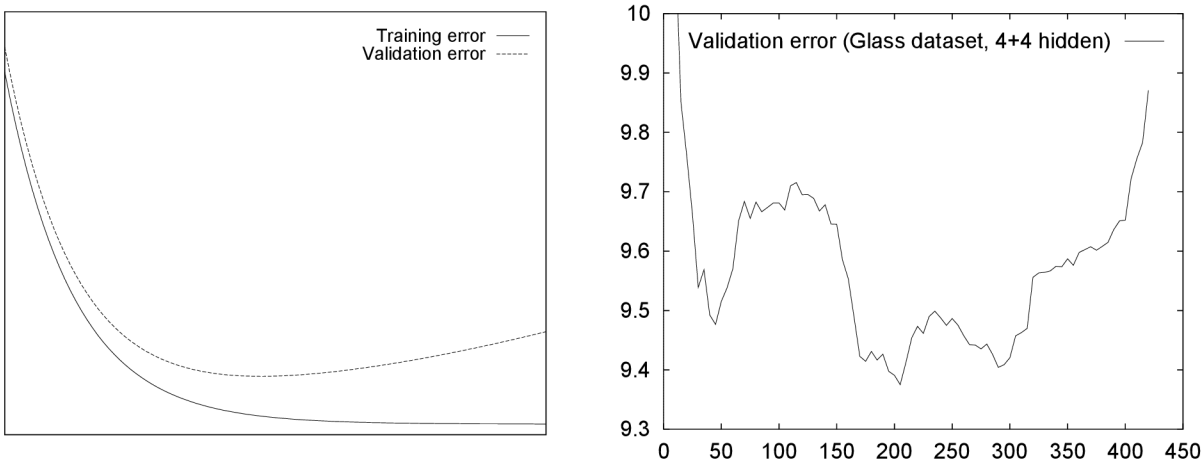
kde $\vec{y}$ je výstupní vektor, $\vec{x}$ je vstupní vektor, $\vec{w}$ je vektor všech vah sítě, $\vec{\Theta}$ je vektor všech prahů a f je aktivační funkce. Učící algoritmus nalezne pro požadované zobrazení $\phi : X \to Y$ takové parametry $\vec{w}$ a $\vec{\Theta}$ funkce f, že tato funkce je právě aproximací tohoto zobrazení [22].

Nejčastějším algoritmem používaným v oblasti neuronových síti pro klasifikaci a učení neuronové sítě je algoritmus zpětného šíření chyby (*Back-propagation*). Algoritmus jako první inicializuje váhy sítě na náhodné hodnoty a poté je vypočtena odezva sítě pro jeden vstup tréninkové množiny. Ta se porovná s požadovaným výstupem a na základě rozdílu mezi vypočtenou a požadovanou hodnotou je určena parciální chyba sítě (tedy chyba pro konkrétní vzor). Celková chyba sítě je pak dána součtem parciálních chyb. Získaná parciální chyba se násobí tzv. *learning-rate* (rychlost učení) a je propagována zpětně sítí od výstupu ke vstupům. Při zpětném šíření chyby jsou upravovány jednotlivé váhové koeficienty takovým způsobem, aby při dalším výpočtu odezvy sítě byla chyba menší. Tento proces se opakuje dokud je celková chyba sítě větší než zadané kritérium [22]. Princip tohoto algoritmu tedy spočívá v minimalizaci chybové funkce. K tomu se používá *gradientní metoda*, která se snaží snížit hodnotu chybové funkce modifikací váhových koeficientů a prahů. Určí se tedy o kolik se změní chybová funkce, pokud změníme určitou váhu. Tuto funkci si lze představit jako zakřivenou plochu v hyperprostoru, okamžitá hodnota vah a prahů je pak bodem na této ploše.

Jak již bylo řečeno k přeučení neuronové sítě může dojít nevhodným zvolením topologie sítě (příliš mnoho neuronů ve skrytých vrstvách). Pokud chceme problému přeučení předejít a dosáhnout optimální schopnosti generalizace, je možné použít trénování s dřívějším zastavením. Základní myšlenka je následující.

1. Rozdělit trénovací data na trénovací množinu a validační množinu (například 10% z trénovacích dat).
2. Učení sítě provádět pouze na trénovací množině a jednou za čas vyhodnocovat chybu na validační množině (například každou pátou epochu).
3. Zastavit trénování, jakmile chyba na validační množině začne stoupat.

Pro praktické použití tento zjednodušený přístup však není dostatečný, jelikož průběh chyby nemá pouze klesající charakter, ale typicky obsahuje lokální minima. Cílem této práce však není detailní popis těchto přístupů více viz [25].

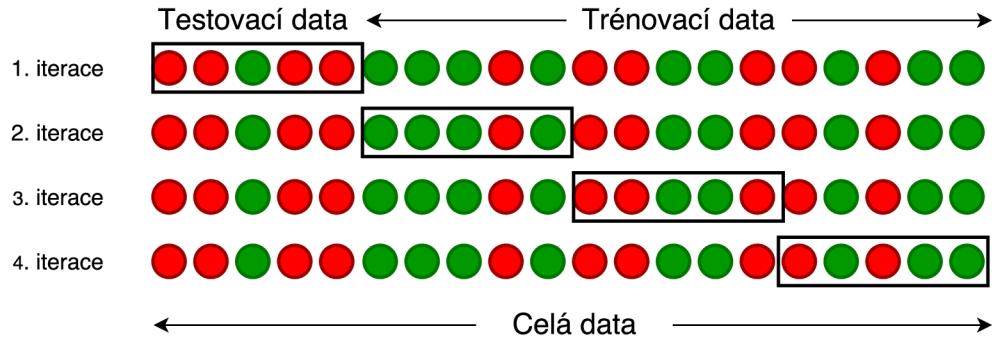


Obrázek 3.7: Vlevo ideální křivka chyby vpravo reálná [25].

3.3 Přesnost klasifikačních modelů

Míra přesnosti klasifikačního modelu vyjadřuje jeho schopnost klasifikovat neznámá data. Určení přesnosti je nutné provádět na neviděných datech (takových jenž nebyly použity pro trénování), opačný případ by vedl k chybným výsledkům. Vyhodnocení je pak založeno na testování naučených znalostí na datech, pro které máme možnost porovnat, zda se nalezené výsledky shodují s realitou. Někdy je ovšem dat málo a další data pro testování nejsme schopni získat, jindy může být obtížné vhodně rozdělit data na trénovací a testovací. Existuje však několik způsobů, které umožní řešit různé způsoby výběru trénovacích a testovacích dat. V následujících odstavcích je popsána základní myšlenka jen pár z nich.

Často využívanou metodou je metoda *k-fold cross-validation*, při které jsou data náhodně rozdělena do k (například 10) disjunktních podmnožin (tzv. folds) o přibližně stejné velikosti. Vyhodnocení probíhá v k iteracích, kdy je brána vždy pouze jedna část pro testování a zbylé pro trénování modelu. Pro celkovou přesnost se výsledky jednotlivých iterací zprůměrují. Existuje také mírná modifikace této metody tzv. *stratified cross-validation*, která data dělí do podmnožin tak, aby každá podmnožina měla přibližně stejnou distribuci tříd.



Obrázek 3.8: Demonstrace *k-fold* pro $k = 4$.

Variantou *k-fold cross-validation* je i metoda *leave-one-out*. Ta vezme z dat pouze jeden vzorek pro testování a zbylá data se použijí na trénování. Toto je opakováno tolikrát, kolik vzorků máme. Jedná se tedy prakticky o *n-fold cross-validation*. Pro velký počet vzorků (například desetitisíce) však n běhů algoritmu není zrovna efektivní, existují ovšem přístupy, které toto řeší [9].

Další metodou je metoda *bootstrap*, která vybírá vzorky pro učení náhodně. Na rozdíl od předchozích metod, zde se tentýž vzorek může pro učení vybrat více než jednou. Máme-li datovou množinu o velikost n, provede se náhodný výběr n-krát, abychom získali n trénovacích vzorků. Pravděpodobnost výběru vzorku je $\frac{1}{n}$, tedy pravděpodobnost, že vzorek vybrán nebude je $1 - \frac{1}{n}$. Pokud provedeme výběr n-krát bude pravděpodobnost, že vzorek vybrán nebude:

$$\lim_{n\to\infty} \left(1 - \frac{1}{n}\right)^n = e^{-1} = 0.368 \tag{3.3}$$

Vybere se tedy asi 63.2% dat pro učení a zbytek bude pro testování [9].

Jelikož vstupních dat pro učení je málo, je pro nás nejzajímavější metoda *k-fold*. Ta se tedy využije při testování modelu a vyhodnocení jeho stability viz. kapitola zhodnocení dosažených výsledků 7.

Kapitola 4

Návrh aplikace

Cílem je vytvořit webovou aplikaci nezávislou na platformě klienta či webovém prohlížeči tak, aby byla aplikace funkční bez potřeby instalace dalších závislostí. Funkčnost na mobilních zařízeních není prioritou, jelikož pro objektivní vyhodnocení úloh je požadováno, aby každý z kandidátů měl alespoň přibližně stejné podmínky (velikost zobrazení, klávesnice atd.), je to ovšem jeden z možných směrů dalšího vývoje. Jedním z požadavků je, aby aplikace brala v úvahu i různá postižení (například barvoslepost), kterými může být uchazeč omezen a nabízet odpovídající alternativy, které zajistí stejné podmínky všem účastníkům.

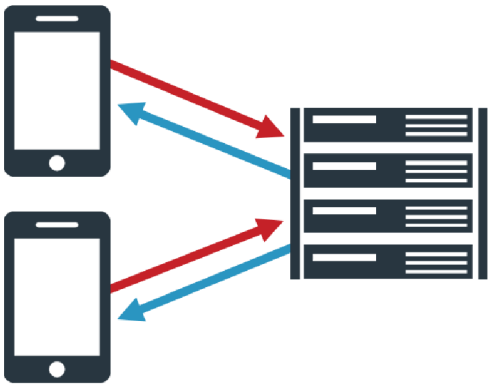
4.1 Architektura

Typickou architekturou pro webové aplikace je architektura klient–server. Webový server na základě požadavků klienta generuje webové stránky. Uživatelský prohlížeč internetu jakožto klient pak pouze zobrazuje obsah generovaný serverem a zajišťuje interakci s uživatelem. Komunikaci typicky zajišťuje protokol HTTP, který funguje na principu dotaz–odpověď, což lze vidět na obrázku 4.1. Naše aplikace však není obyčejnou webovou aplikací. Webový server budou doplňovat další dva servery, jeden pro testování a druhý pro klasifikaci uživatele na pracovní pozici.

HTTP server

Webový server bude implementován v jazyce *PHP* s využitím frameworku *NETTE*. Framework byl vybrán díky své spolehlivosti, rychlosti a pohodlnému vývoji. Výhodou je i fakt, že dokumentace je v češtině a i jeho komunita je poměrně rozsáhlá [2]. Tento server bude zajišťovat veškeré úkony webové aplikace od registrace a následné autentizace klienta až po postupné procházení klienta jednotlivými úlohami testu. V administrační části pak umožní nastavení parametrů jednotlivých úloh i parametrů potřebných k vyhodnocení výsledků.

Každá z úloh sestává z určitého počtu kroků. Kroky ve většině úloh obsahují náhodně vybraný a umístěný obsah (relevantní k úloze), který je obvykle závislý na obsahu předchozích/následujících kroků. Vygenerování takové posloupnosti typicky není triviální a u složitějších úloh může trvat i několik sekund. Z toho důvodu jsou tyto ”scénáře” pro úlohy generovány offline a uloženy do databáze.

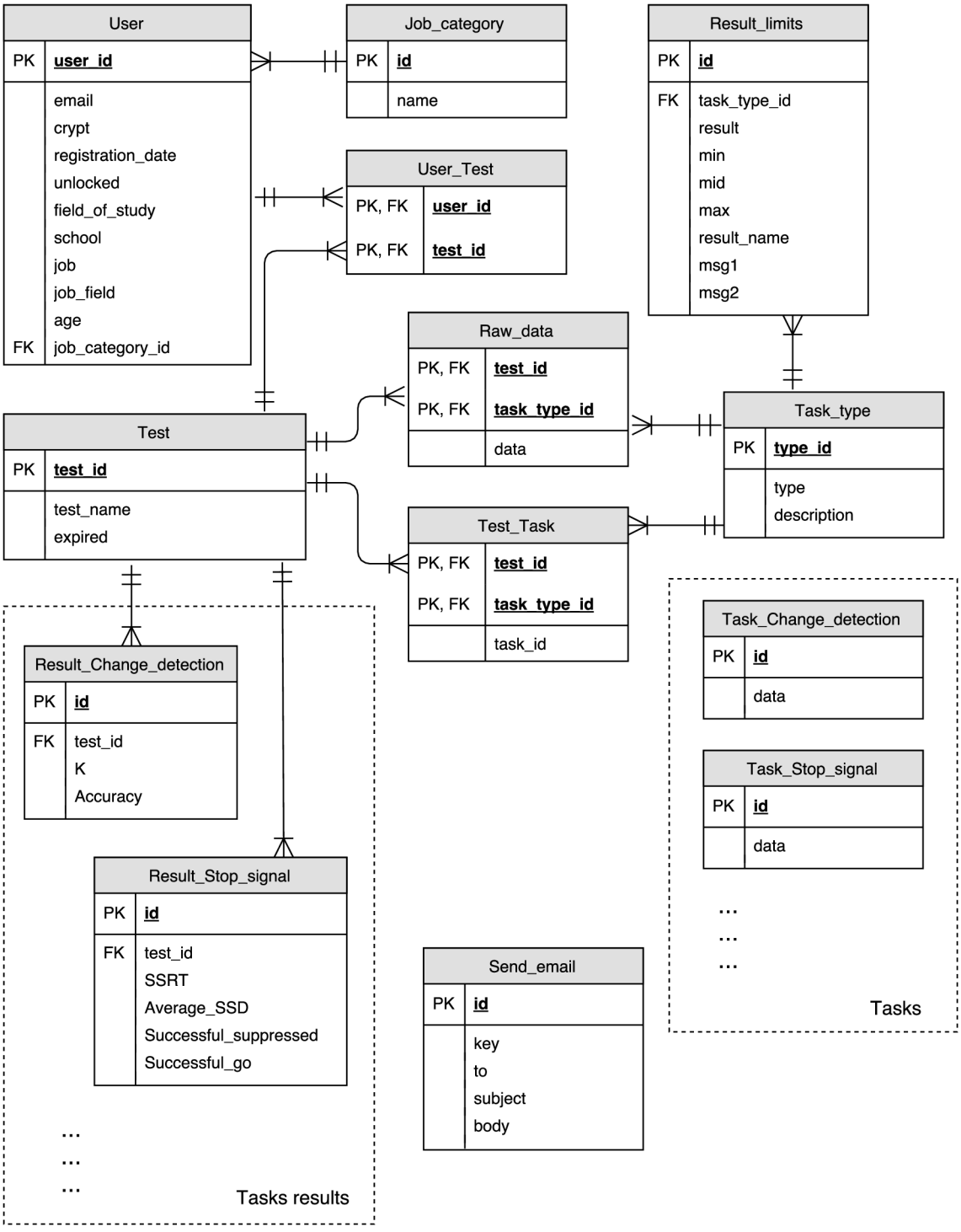


Obrázek 4.1: Request-respond komunikace protokolu HTTP [26].

Databáze

Aplikace bude postavená nad relační databází *MariaDB*, jenž vychází z databázového systému *MySQL* a je publikována pod open-source licencí. Oproti původnímu *MySQL* má *MariaDB* údajně lepší výkon [1]. Na obrázku 4.2 lze vidět návrh schématu databáze.

Pro předgenerované úlohy jsou zde tabulky s prefixem *Task_* následovaným názvem úlohy (ve schématu níže jsou uvedeny pouze dvě tabulky z důvodu úspory prostoru). Stejně tak tabulky pro uchování výsledků mají vlastní prefix, následovaný názvem úlohy, zde každá z tabulek obsahuje sloupce pro uložení výsledků specifických pro úlohu. Tabulka *User* obsahuje informace o testovaném uživateli, jako je jeho pracovní pozice a oblast, ve které pracuje. Podle těchto dvou údajů je uživateli přiřazena kategorie. Úlohy jsou seskupeny do testu, který je následně přiřazen uživateli. V tabulce *Task_type* jsou názvy jednotlivých úloh, které slouží mimo jiné i k identifikaci tabulek. Toto je využito například v tabulce *Test_task*, která spojuje test s konkrétními úlohami. *Test_id* zde říká, do kterého testu úloha patří, *task_type_id* ukazuje na název úlohy, podle kterého je identifikována jedna z tabulek s prefixem *Task_*, ve které pak *task_id* označuje konkrétní řádek. Parametry pro výpočet procentuálních výsledků jsou v tabulce *Result_limits*. Pro možné pozdější přepočítání jsou uložena i surová data v tabulce *Raw_data*.



Obrázek 4.2: Schéma databáze.

Websoketový server

Vedle webového serveru bude aplikace využívat i technologii webových soketů. Webové sokety fungují na principu klasických soketů, tedy udržují otevřené spojení mezi klientem a serverem, což je oproti komunikaci webového serveru s klientem pomocí protokolu HTTP

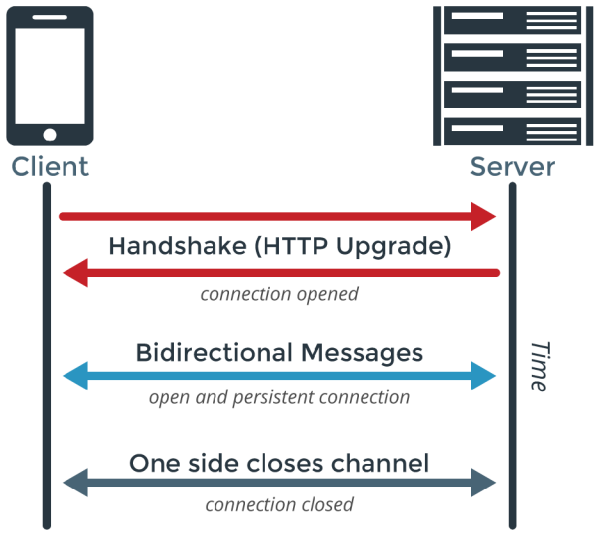
velkou výhodou. Data jsou ovšem přenášena ve formě otevřeného textu. Z důvodu nebezpečí odchycení takovéto komunikace je tedy vhodné použít tzv. zabezpečené webové sokety, které komunikují přes protokol SSL/TSL, přenášený obsah je tedy šifrován. Webové sokety neřeší autentizaci či autorizaci, proto je nutné navrhnout vlastní postup. V naší aplikaci bude proces autentizace následující.

1. Před otevřením soketu kontaktuje klient webový server s požadavkem na autorizační token.

2. Server vygeneruje token na základě uživatelského ID či jiného identifikátoru, IP adresy a dalších informací, odešle token klientovi a zároveň ho ukládá do databáze. Token má platnost pár desítek minut.

3. Klient otevře spojení přes web soket a zasílá svůj token jakožto inicializaci komunikace.

4. Server zkontroluje token a jeho platnost, ověří zda spojení přišlo ze stejné IP adresy, i to že nebyl token použit vícekrát. Na základě toho verifikuje spojení.



Obrázek 4.3: Komunikace pomocí webových soketů [27].

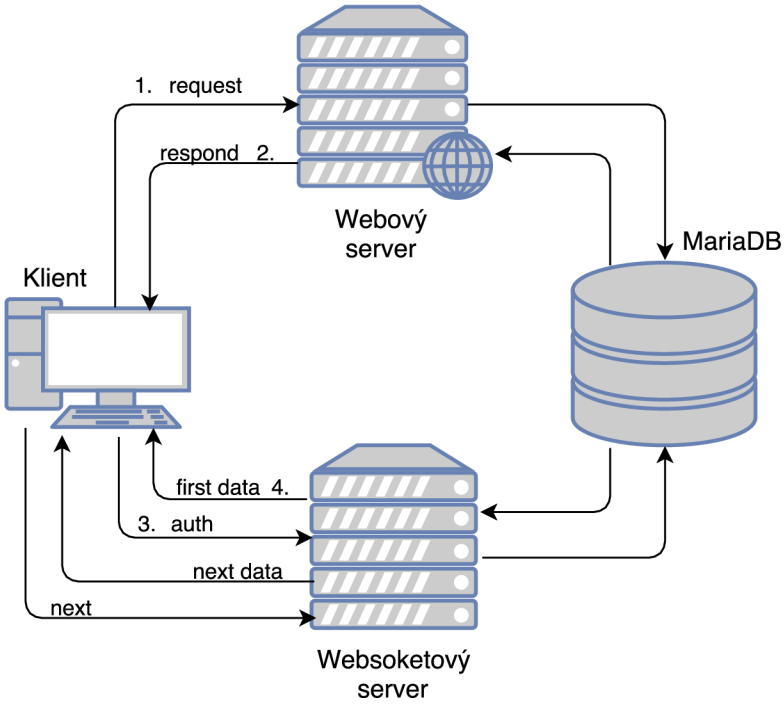
Server obsluhující požadavky přes tyto sokety bude konkurentní a v aplikaci oddělen od klasického webového serveru. Toto zajistí jak větší přehlednost aplikace, tak rychlejší obsluhu takovýchto požadavků. Nízká latence těchto přenosů je klíčová, jelikož se tímto komunikačním kanálem budou přenášet části testů k vykreslení u klienta (viz obrázek níže). Každý krok každé úlohy má striktní časové ohraničení (typicky v řádech stovek milisekund), které musí být dodrženo, aby se daly výsledky považovat za objektivní.

Implementace tohoto serveru proběhne v programovacím jazyce *GO* vyvinutým společností Google. *GO* nebo také *GoLang* je staticky typovaný kompilovaný jazyk, ovšem programování v něm často připomíná dynamicky typované jazyky. Výhodami jsou například snadná a efektivní práce s více vlákny, garbage collector či opravdu jednoduchá instalace a import balíčků [4].

Klient

Veškeré objekty úlohy budou vykresleny do vektorového grafického formátu *SVG*. Ten je založen na značkovacím jazyce *XML*, a proto je možné ho snadno použít v *HTML* dokumentu. Přístup k *SVG* zajistí *DOM* (Document Object Model), díky kterému lze klientským javascriptem provádět modifikace, i přidávat akce pro interakci s uživatelem. K tomuto účelu bude použita javascriptová knihovna *Snap* [3], jež poskytuje rozhraní pro práci s *SVG* na straně klienta. Většina objektů tedy bude do *SVG* přímo nakreslena. Díky tomu lze jednoduše pracovat nezávisle s jednotlivými zobrazenými prvky.

V průběhu testování uchazeče jsou zde zaznamenávány jeho reakce, reakční časy a vůbec všechny potřebné informace nutné k vyhodnocení úlohy, jak bylo popsáno v kapitole 2. Toto jsou výše zmíněná surová data, která se také ukládají.



Obrázek 4.4: Schéma návrhu aplikace pro testování.

Obrázek 4.4 zobrazuje jednoduché schéma aplikace. Klient jako první posílá klasický HTTP požadavek, který webový server vyhodnotí a vrátí odpověď. Jakmile uživatel zahájí úlohu, klient kontaktuje s autentizačními údaji uživatele druhý server. Ten načítá z databáze veškerá data spojená s úlohou a pošle klientovi jejich první část. V průběhu úlohy klient požaduje po serveru další a další části, které aktuálně potřebuje. Oba servery v případě potřeby přistupují k databázi.

Klasifikační server

V neposlední řadě aplikace obsahuje server obstarávající klasifikaci uživatelů. Ten nebude součástí základní aplikace pro testování (proto není naznačen ve schématu výše), bude jakýmsi rozšířením funkcionality. Implementován bude v programovacím jazyce *Python* s využitím knihovny *Theano* a její nadstavby knihovny *Keras*. Pro komunikaci zde bude implementováno jednoduché aplikační rozhraní (API), přes které server dostane odpovídající vstupní data a vrátí výsledek.

Pro samotnou klasifikaci bude implementovaná vícevrstvá neuronová síť. Pro začátek bude síť obsahovat pouze jednu skrytou vrstvu. Síť se případně rozšíří, jakmile bude k dispozici větší množství trénovacích dat pro experimenty. Vstupem budou výsledky všech úloh, což je pro jednoho uživatele asi 45 hodnot. Jde tedy o 45-dimenzionální vstupní data. Jelikož mají data v dimenzích poměrně velké rozsahy, bude vhodné každou dimenzi před použitím normalizovat. K normalizaci použijeme metodu Z-Score, která normalizuje na distribuci se střední hodnotou nula a standardní odchylkou jedna.

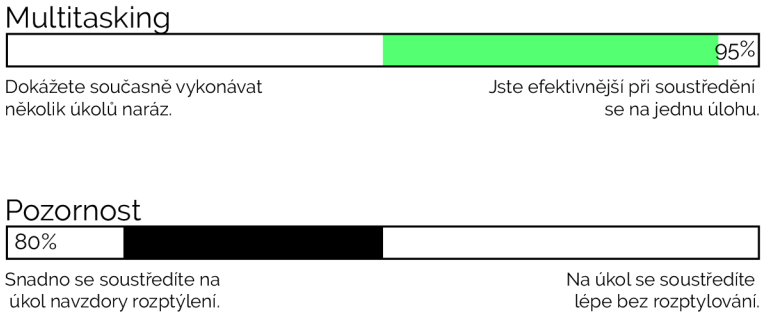
Místo použití výsledků všech testů k naučení jediného modelu, je také možný přístup vytvořit model pro každou z úloh zvlášť (tzn. 10 modelů). Klasifikaci provádět nad každou z úloh. Výsledky pro každou třídu jednoduše průměrovat nebo přidat "zastřešující" neuronovou síť, která by měla na vstupu výstupy všech 10 modelů.

4.2 Metriky

Každá úloha má za úkol vyhodnotit jiné vlastnosti respondentů. Na vlastnost lze nahlížet jako na omezený prostor výsledků, který definuje metriku určující úspěšnost v dané vlastnosti. Na základě reakcí lze spočítat metriky specifické pro úlohu. Každá z vypočítaných metrik poslouží k zobrazení procentuálního hodnocení jednotlivých vlastností. Některé vlastnosti vycházejí z výsledků více než jedné metriky (například kreativita).

Po dokončení úlohy zašle klient na websoketový server veškerá data změřená během úlohy. Na serveru proběhne validace přijatých dat a následné vypočítání všech metrik. Data jsou zasílána ve formátu *JSON* a jsou považována za validní, pokud je jich dostatek (uživatel musí typicky reagovat na více než polovinu stimulů) a v případě reakčních časů nesmí obsahovat víc než 15% příliš malých hodnot (například menších než 150ms). Toto by značilo příliš rychlé reakce uživatele a s největší pravděpodobností pokus od uživatele test ošálit.

Osa zobrazení procent se dělí na dvě části 0%–50% a 50%–100%, každá z částí vypovídá o vlastnosti respondenta něco jiného (obvykle levá část osy značí horší variantu). Uživatel ovšem vidí procenta zobrazená na ose 100%—50%—100% viz. obrázek níže. Aby bylo možné procentuální hodnocení vypočíst, je nutné omezit výsledky metrik a stanovit střed. Pro každou metriku jsou tedy stanovené hranice pro výpočet těchto procent ve tvaru: minimální hodnota, střed, maximální hodnota. Středová hranice je potom přesně 50% na zobrazené ose. Většina hranic vyplývá z citovaných výzkumů, některé jsou však experimentálně určeny přímo z dat.



Obrázek 4.5: Ukázka zobrazení výsledků.

Při rozdělení do pracovních oborů je vhodné mít pro různé pracovní obory různé hranice tak, aby výsledky co nejlépe reflektovaly vlastnosti uživatele v tomto oboru. Abychom tohoto dosáhli, je potřeba zaměřit se na získaná data a podle nich vhodným způsobem hranice nastavit. Jedním z návrhů, jak optimalizovat hranice, je vzít například 15% nejhorších výsledků, z nich určit hodnotu mediánu a tu prohlásit za novou minimální hranici. Stejným způsobem pak určit i maximum a střed. Před tímto procesem bude potřeba data očistit od odlehlých hodnot, které by mohly vznikat například vzdálením uživatele, přičemž měření času pořád běží.

Kapitola 5

Implementace

Podle návrhu prezentovaného v předešlé kapitole je aplikace rozdělena na dvě serverové části a část klientskou. Jeden ze serverů představuje klasický webový server využívající *PHP* a druhý napsaný v programovacím jazyce *Go*, který se stará o veškerá data spojená s úlohami testu. Komunikaci mezi klientem a *Go* serverem zajišťují webové sokety. Oba servery přistupují k databázi, kde jsou uložena data potřebná k testům. Vlastní aplikaci pro testování doplňuje další server, který implementuje klasifikaci uživatelů do oborů/pracovních pozic. Tato kapitola si dává za cíl, dát uživateli představu o struktuře a implementaci aplikace.

5.1 Webový Server

Implementace využívá moderní webový framework *NETTE*. Ten je postavený na architektuře MVC (Model-View-Controller), která odděluje kód obsluhy od kódu aplikační logiky a kódu grafického uživatelského rozhraní zobrazujícího data. Tento server zajišťuje interakci s uživatelem a probíhá zde také předgenerování jednotlivých úloh do databáze kvůli snížení latence při požadavku na úlohu mezi klientem a serverem.

Majoritní funkčnost aplikace řeší tři kontrolery. Třída `HomepagePresenter`, která obsluhuje požadavky na hlavní stránku (v tuto chvíli pouze přesměrování na vstupní formulář). Třída `TestingPresenter` zpracovává všechno ohledně testování. Po odeslání vstupního formuláře zavolá model `TestModel`, který se postará o vytvoření uživatelova testu. Test se vytvoří tak, že se každá úloha vybere náhodně z databáze předgenerovaných (v *SQL* klauzuli *ORDER BY* je použita funkce *RAND()*). Také se generuje token pro uživatele, který mu umožní kdykoliv se ke svému testu vrátit, aby test nutně nemusel absolvovat na jedno sezení (odkaz je uživateli odeslán na email). Při přístupu uživatele ke svému testu, pak token ověří a načte do *SESSIONS* potřebné informace k testu. Třída `ResultsPresenter` zajistí díky modelu `ResultsModel` výpočet procentuálního hodnocení vlastností z výsledků (jak bylo popsáno v 4.2).

Každý kontroler spolupracuje s mnoha třídami modelu, především pro přístup k databázi a k manipulaci s daty. Každá z úloh má svůj model, který ji generuje. Výchozí třídou pro generování úloh je `GenerateTask`, která obsahuje metody a konstanty společné pro všechny úlohy, a ze které dědí jednotlivé třídy pro generování úloh. Vygenerované úlohy jsou uloženy ve formátu *JSON* do databáze do tabulky odpovídající svému názvu.

5.2 Klient

Klient zajišťuje zobrazení aplikace uchazeči. Po načtení klasické webové stránky obsahující informace o průběhu testu má uživatel možnost kliknout na konkrétní úlohu, kterou chce vykonat. Veškeré akce spojené s testováním zajišťuje klientský javascript, využívající knihovnu *jQuery*, pro práci s *SVG* je použita knihovna *Snap* [3]. Při kliku na tlačítko, které spouští úlohu je pomocí web soketu kontaktován *Go* server, který poskytuje data pro jednotlivé kroky každé úlohy (co, kam a na jak dlouho má být zobrazeno). Nejdůležitější funkcí klienta je v průběhu testu sbírat data od uživatele a následně je odesílat na server.

Pro tvorbu uživatelského rozhraní využívá aplikace šablonovací systém *Latte*. Ten mimo jiné pomocí vhodného escapování chrání proti zranitelnostem typu *XSS*. Každá úloha má tedy vlastní šablonu, která načítá stejnojmennou javascriptovou třídou (např. `ChangeDetection`). Tyto třídy implementují zobrazení konkrétních objektů úlohy, obohacují *SVG* o prvky interakce a také ukládají uživatelské reakce a data potřebná k vyhodnocení úlohy. Všechny tyto třídy dědí z `CreateTask`. Tato třída implementuje otevření webového soketu, komunikaci se serverem a další společné části. Postupně po jednotlivých krocích poptává data po serveru, obdržená data načítá z *JSONu* a pro každý vykreslovaný objekt vytvoří instanci `DrawingObject`, pro přehlednou reprezentaci vykreslovaného objektu (jsou zde uloženy informace, jako typ objektu, pozice, barva atd.). Třída `Timer` implementuje jednoduchý časovač, který po uplynutí specifikované doby zavolá definovaný *callback*. Tímto způsobem funguje načítání dalších kroků úlohy, jakmile jsou všechny objekty vykresleny časovač se spustí a jako *callback* je předaná metoda pro načtení dalšího kroku. V `CreateTask` je také načtení instrukcí před každou úlohou, instrukce jsou v *SVG* formátu tzn., že po vložení do stránky je potřeba tlačítkům přidat odpovídající *callbacky*. Jakmile je úloha u konce (klient obdrží prázdná data), odešle serveru uložené hodnoty a čeká na odpověď. Server data validuje a posílá klientovi informaci, zda jsou data v pořádku. V případě chyby, pak i chybovou hlášku pro uživatele.

5.3 Websoketový server

Druhá serverová část je implementována v poměrně novém programovacím jazyce *Go*, úkolem je dodávat data pro jednotlivé kroky každé úlohy. Také je zodpovědný za vyhodnocování a ukládání výsledků. Domluva obou aktérů pak spočívá ve výměně dat na základě jednoduchého textového protokolu.

Jazyk *Golang* neobsahuje třídy, ale dovoluje definovat struktury (podobně jako v *C*), na které je možné navazovat metody, což ve výsledku objektovost dobře nahrazuje. Implementace konkurentního serveru je rozdělena na dvě části. První část je struktura `Server`, která obsahuje adresu serveru, seznam všech klientů, kanály pro koordinaci mezi vlákny a metody pro obsluhu příchozích požadavků. Nejpodstatnější je metoda `Listen()`, ve které server čeká na příchozí nová spojení. Druhá část je struktura `Client`, která obsahuje ukazatel na otevřené spojení a data klienta. Jakmile server obdrží nové spojení, vytvoří novou instanci `Client`, přidá ji do seznamu klientů a v novém vlákně spustí naslouchání pro tohoto konkrétního klienta. Od této chvíle se o klienta stará struktura `Client`. Ta má pro naslouchání metody dvě. `ListenRead()` čte data ze soketu a volá funkce pro jejich další zpracování. `ListenWrite()` data do soketu zapíše jakmile jsou připravena. Veškerá komunikace (mezi těmito dvěma metodami i mezi strukturami `Client` a `Server`) je řízena pomocí kanálů [4],

které toto velice zjednodušují. Pokud se klient odpojí je opět nastaven odpovídající kanál, díky kterému server klienta smaže.

Při inicializaci komunikace server ověřuje token uživatele, pokud je v pořádku, pak se data úlohy načtou z databáze a jejich první část se posílá klientovi. Ten si později může vyžádat právě následující část dat nebo i část posunutou o N kroků dále. V případě úlohy, která předgenerovaná v databázi není (pouze *Prisonner's dilemma*) server pouze ověří token uživatele a čeká na výsledky. Implementace vyhodnocení výsledků je v balíčku `Evaluation` pro přehlednost rozdělena do souborů podle názvů úloh. Implementace vyhodnocení vychází především z citovaných výzkumů.

Aplikace je logicky strukturovaná do balíčků. Název každého balíčku odpovídá činnosti, kterou implementuje. Pro práci s databází je tu `Database`, pro obsluhu soketů `Websocket`. V `DataModel` jsou definovány datové struktury a potřebné konstanty. V neposlední řadě vyhodnocení úloh je v `Evaluation`.

5.4 Klasifikace

Klasifikační část je momentálně implementována bez navrhovaného API a funguje zatím pouze nad vstupními soubory ve formátu *csv*. Implementace je realizovaná v jazyce *Python* s knihovnami *Theano* a *Keras*. Tento program sestává z hlavního souboru *main* a třídy `Classifier`. Ta obsahuje vytvoření neuronové sítě (klasifikačního modelu), učení modelu, vyhodnocení úspěšnosti modelu a vlastní klasifikaci.

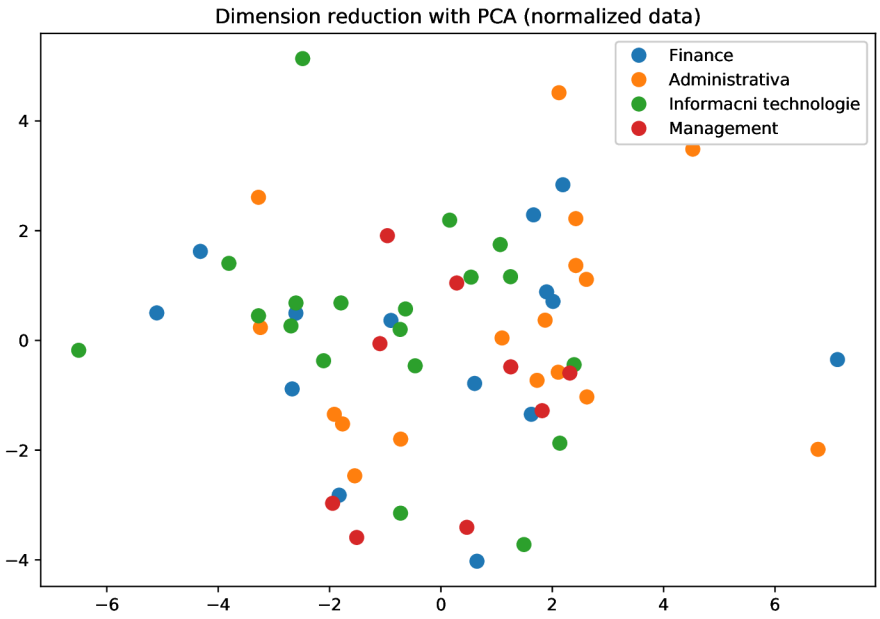
Při spuštění programu je možné zadat několik parametrů popsaných dále. Vytvoření modelu je implementováno dvěma způsoby. Jením z nich je síť pouze se vstupní a výstupní vrstvou a druhý neuronová síť s jednou skrytou vrstvou. Toto rozlišuje vstupní parametr `--hidden`, který podmiňuje použití modelu se skrytou vrstvou (jako výchozí je použita síť bez skryté vrstvy). U obou modelů je aktivační funkce ve výstupní vrstvě sítě funkce *Softmax*, která zařídí pravděpodobnosti na výstupu. Aktivační funkce ve skryté vrstvě je funkce *Rectifier*.

Další vstupní parametr je `-e` či `--evaluate`, který říká, že místo samotné klasifikace nad modelem bude vyhodnocena úspěšnost modelu, `--save_model` a `--load_model` určují, zda se natrénovaný model pro klasifikaci uloží/načte do/z souboru, aby nebylo nutné učení sítě pořád opakovat. Zadáním `-m` nebo `--mode` je možné specifikovat mód klasifikace. Ten určuje, zda vzít výsledky ze všech úloh najednou a vytvořit pouze jeden model, na kterém klasifikovat, anebo pro každou z úloh vytvořit model zvlášť (celkem 10 modelů) a výsledky klasifikace poté zprůměrovat. Pro kontrolní výpisy během učení sítě stačí zadat `-v` nebo `--verbose`, pro výpis nápovědy `-h` či `--help`.

V konstruktoru třídy `Classifier` jsou načteny data pro učení modelu ze souboru, předaného parametrem (výchozí soubor je *data/results.csv*). Pokud je parametr `normalize` nastaven na *True* (defaulní hodnota), pak zde proběhne i normalizace dat, tak jak bylo řečeno v návrhu. Pokud je program spuštěn s úmyslem klasifikace, pak je nejprve nutné spustit učení neuronové sítě (metoda `train()` této třídy). Ta vezme načtená data a použije je pro učení sítě. Učení využívá algoritmus *Back-propagation* s optimalizační metodou *Gradient descent*. Pokud byl program spuštěn s parametrem `--load_model`, pak se model pouze načte ze souboru. Poté je v metodě `predict()` provedena samotná klasifikace. Té jsou předaná data ke klasifikaci a parametr určující, zda mají být data před predikcí normalizována. Metoda vrací pole pravděpodobností příslušnosti vstupních dat k jednotlivým třídám.

Pro vyhodnocení úspěšnosti modelu (metoda `evaluateModel()`) je použita metoda *k-fold cross-validation* s hodnotou $k = 10$, která byla experimentálně určena. Vyhodnocení funguje přesně tak, jak je popsáno v návrhu. Vstupní množina je rozdělena na 10 částí, kdy v každé z 10 iterací je použita jedna část na testování a zbylé na trénování modelu. Výsledkem je teoretická přesnost modelu. V případě módu, kdy se vytváří model pro každou úlohu zvlášť se výsledky opět průměrují.

Pro experimenty byla použita metoda PCA (Principal Component Analysis), která slouží k redukci dimenzionality. Přesnost modelu byla po redukci na pouze 2-dimenzionální data totožná s přesností bez redukce. Následující graf zobrazuje data po redukci do dvou dimenzí. Dále byl v rámci experimentů model ověřen na syntetických datech, které obsahovaly dvě naprosto oddělené třídy ve 3-dimenzionálním prostoru. Úspěšnost vyhodnocení se držela na 100%.



Obrázek 5.1: Normalizovaná vstupní data po redukci do dvou dimenzí.

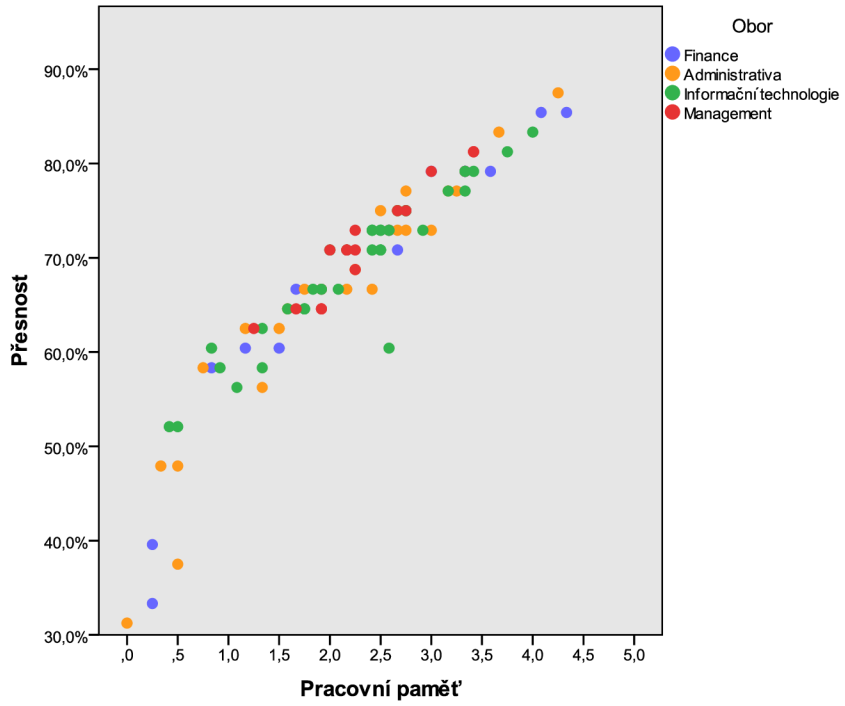
Kapitola 6

Sesbírané výsledky

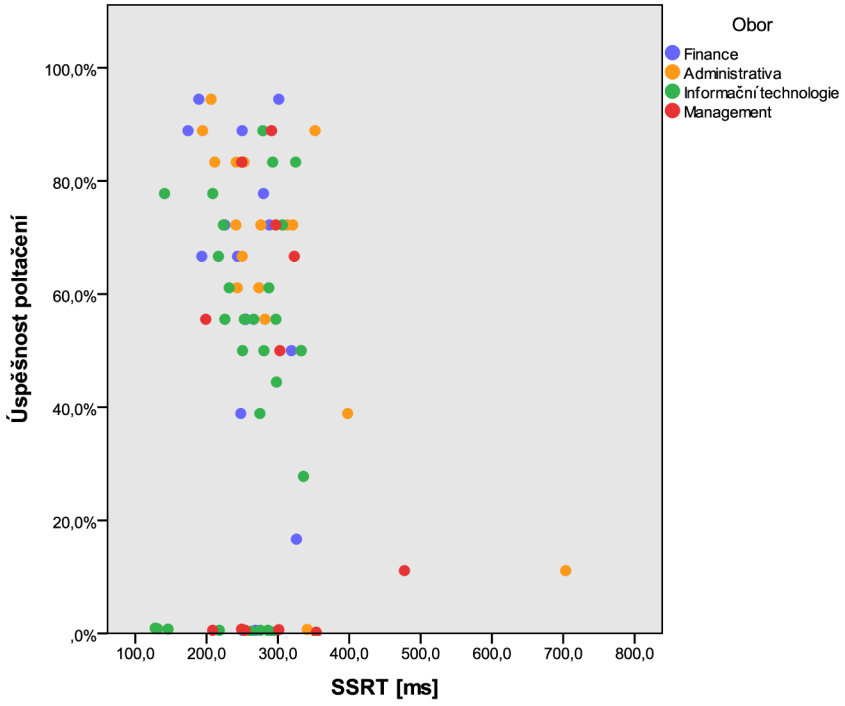
Pro sbírání dat bylo osloveno několik spřátelených firem, jejichž zaměstnanci test absolvovali, oslovení dalších lidí probíhalo prostřednictvím sociálních sítí. Před samotným testem, každý z účastníků vyplnil svou pracovní pozici a obor ve kterém pracuje (v případě studentů studijní obor a název školy). Z celkového množství zhruba 220 přihlášených lidí celým testem (všemi 10 úlohami) prošlo pouze 58 uživatelů, 8 z 10 úloh dokončilo 108 lidí. Je vidět, že aktivních uživatelů je opravdu málo. Proto budeme dále pracovat s výsledky pouze osmi úloh, kterých je zhruba polovina z celku. Největší podíl, z těch kteří prošli úspěšně alespoň osmi úlohami, byli lidé z oboru informačních technologií (22), dále administrativní pracovníci (18), finance a účetnictví (16) a management (9). Zbytek byl roztroušen přes všechny různé obory obsahující pouze jednoho či dva uživatele.

6.1 Grafické znázornění

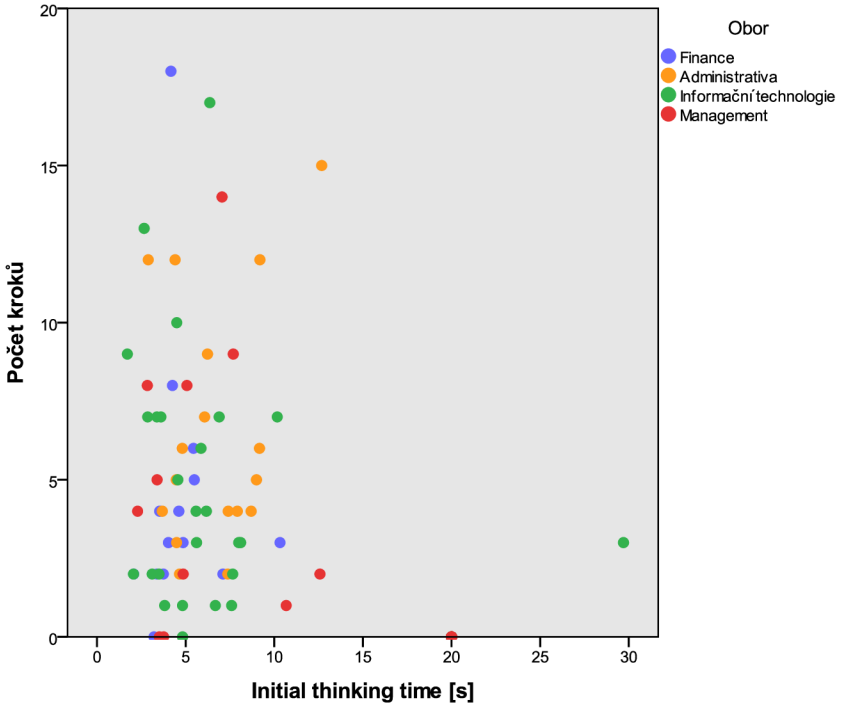
Pro lepší znázornění nasbíraných dat obsahuje tato podkapitola některé související metriky vynesené do bodového grafu. Záměrem je ověření předpokladu, že výsledky odrážející vlastnosti pracujících ve stejných oborech, budou tvořit shluky. Jak je vidět na grafech níže, výsledky metrik jednotlivých oborů jsou promíchané a shluky není příliš vidět. V ideálním případě by měly být jednotlivé třídy patrné i na grafu po aplikaci PCA 5.1. Což bude částečně způsobeno tím, že nebereme konkrétní pracovní pozice, k tomu by ovšem bylo zapotřebí velké množství dat.



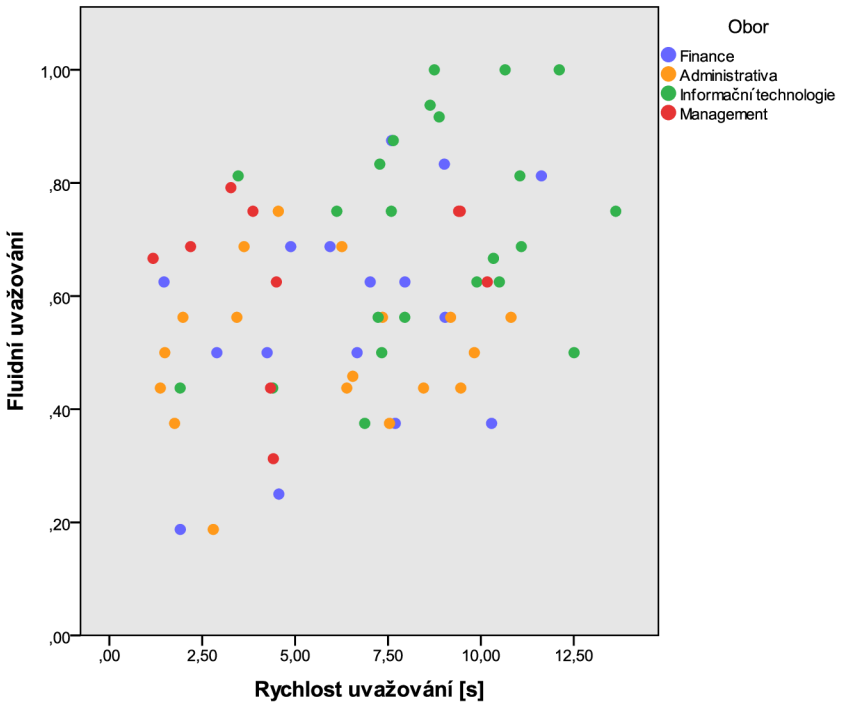
Obrázek 6.1: Závislost mezi přesností odpovědí a velikostí pracovní paměti v úloze Change detection. Z grafu je patrné, že se zvětšující se pracovní pamětí se zvyšuje přesnost výsledku.



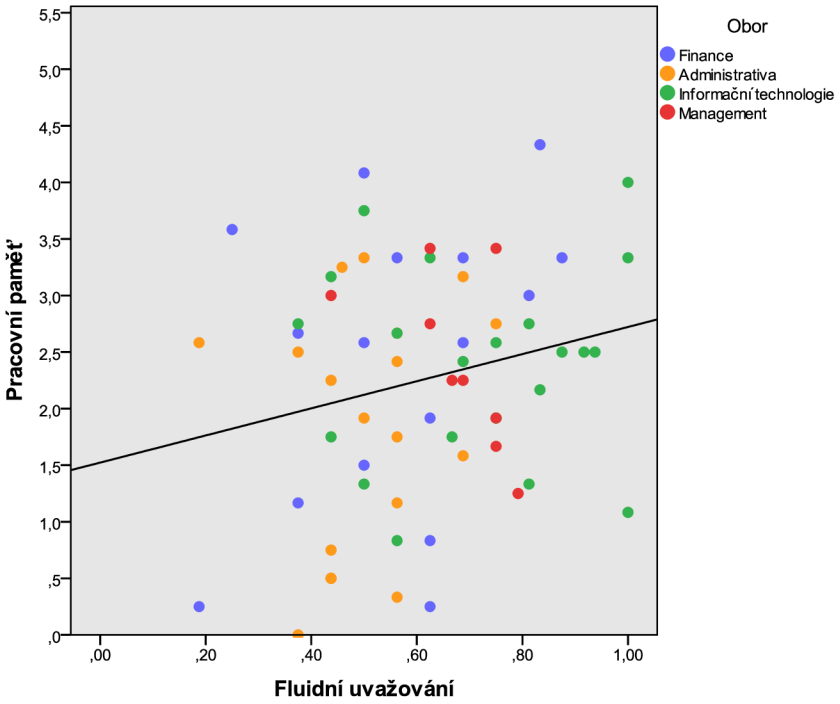
Obrázek 6.2: Stop signal – závislost mezi počtem úspěšně potlačených a délkou potlačení reakce. Lze pozorovat, že ti kteří dokázali úspěně potlačit reakci měli logicky i nižší SSRT.



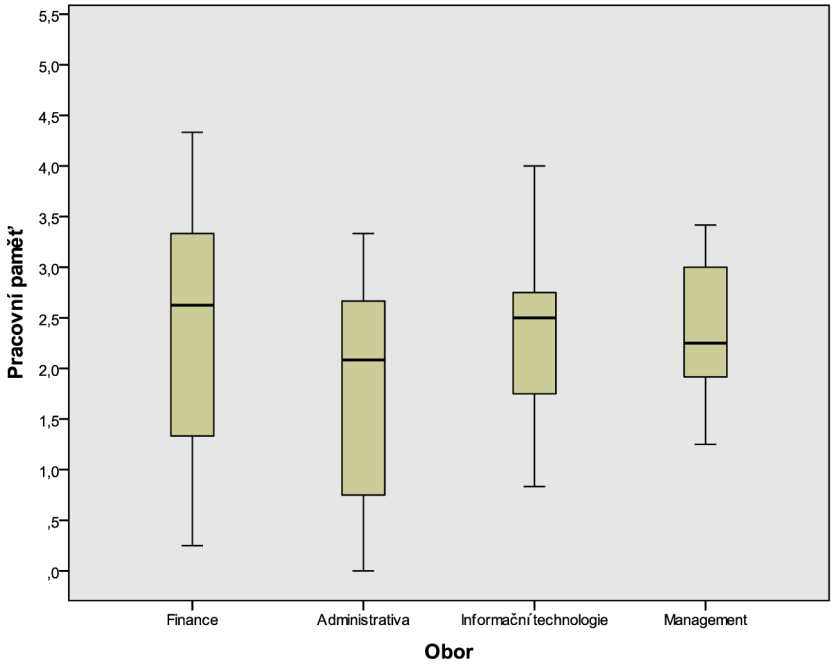
Obrázek 6.3: Zobrazení počtu kroků spolu s časem prvního kroku (ITT) pro úlohu Tower of London.



Obrázek 6.4: Metriky úlohy Geometric analogy – fluidní uvažování ve spojitosti s rychlostí uvažování.

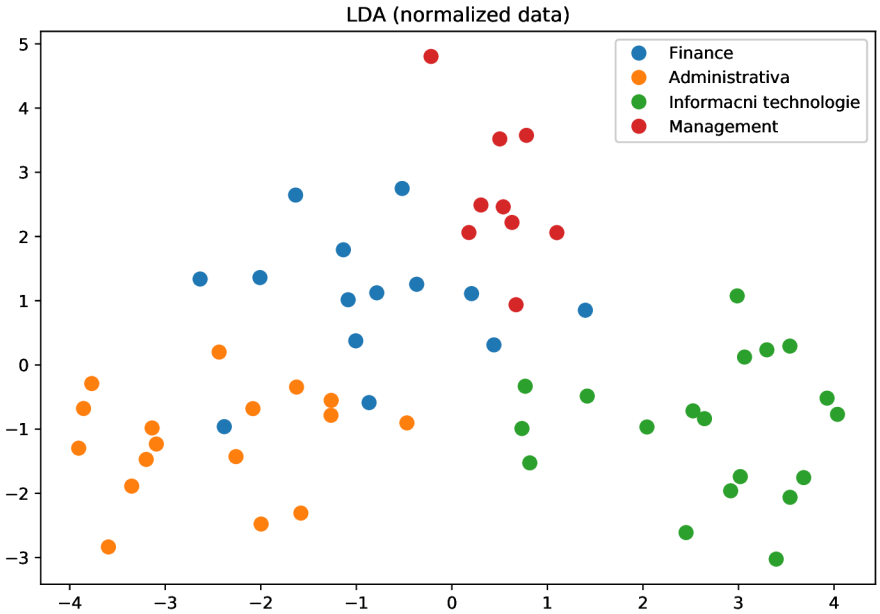


Obrázek 6.5: Spojením fluidního uvažování s velikostí pracovní paměti dostáváme schopnost učení. Ze spojnice trendu lze vidět, že čím větší pracovní pamět tím lepší fluidní uvažování.



Obrázek 6.6: Porovnání dosažené pracovní paměti pomocí krabicového grafu.

Rozdíly mezi obory nejsou z grafů příliš patrné a to ani po aplikaci metody *PCA*. *PCA* ovšem nebere jednotlivé třídy v úvahu, proto se nabízí zkusit ještě metodu *LDA* (linear discriminant analysis), ta také provede redukci dimenzí, ale s ohledem na obsažené třídy.



Obrázek 6.7: Data po aplikaci metody *LDA*.

Obrázek 6.7 už vypadá o něco lépe, dokonce jsou zde třídy vidět na první pohled. Ačkoli vzdálenosti ve shluku jsou poměrně velké a jde vidět, že jsou třídy velice blízko sebe. Obor *Finance* dokonce částečně zasahuje i do ostatních.

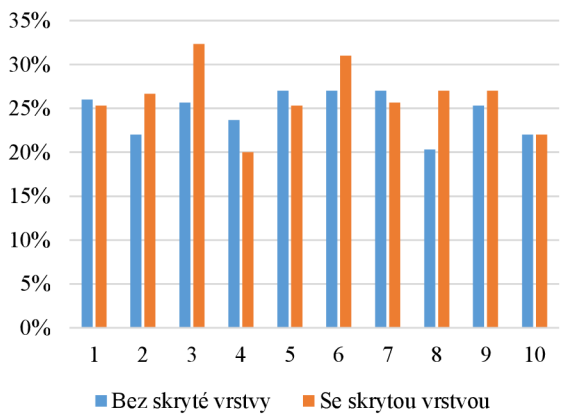
Kapitola 7

Zhodnocení dosažených výsledků

Pro učení neuronové sítě byla použita data pouze čtyřech oborů, které měly největší zastoupení (informační technologie, administrativa, finance a management). To je dohromady asi 65 hodnot, což je velice málo. Byly otestovány oba přístupy navržené v 4.1. Natrénování jediného modelu na výsledcích všech úloh a natrénování modelu pro každou z úloh zvlášť. Druhý způsob nám dá představu o tom, jak se výsledky jednotlivých úloh liší napříč obory. Také byla porovnána přesnost klasifikace neuronové sítě se skrytou vrstvou a bez ní.

Pro testování modelů byla použita metoda *k-fold*, která data vždy dělila na stejných 10 částí. Váhy neuronové sítě byly pro každý test náhodně inicializovány. Každý test byl proveden 10x, z čehož byla vypočítána průměrná hodnota a standardní odchylka. Následující tabulky a grafy prezentují výsledky těchto testů.

Test	Bez skryté vrstvy	Se skrytou vrstvou
1.	26,00%	25,33%
2.	22,00%	26,67%
3.	25,67%	32,33%
4.	23,67%	20,00%
5.	27,00%	25,33%
6.	27,00%	31,00%
7.	27,00%	25,67%
8.	20,33%	27,00%
9.	25,33%	27,00%
10.	22,00%	22,00%
Průměr	**24,60%**	**26,23%**
Std. Dev.	2,21%	3,30%



Tabulka 7.1: Porovnání přesnosti klasifikace jednoho modelu se skrytou vrstvou a bez ní.

Z této tabulky lze vidět, že oba modely jsou srovnatelné, a že neuronová síť se skrytou vrstvou má jen nepatrně větší průměrnou přesnost. Pro lepší přehlednost jsou hodnoty vyneseny do grafu, který toto potvrzuje. Přesnost 26% je však pro čtyři třídy velice špatná.

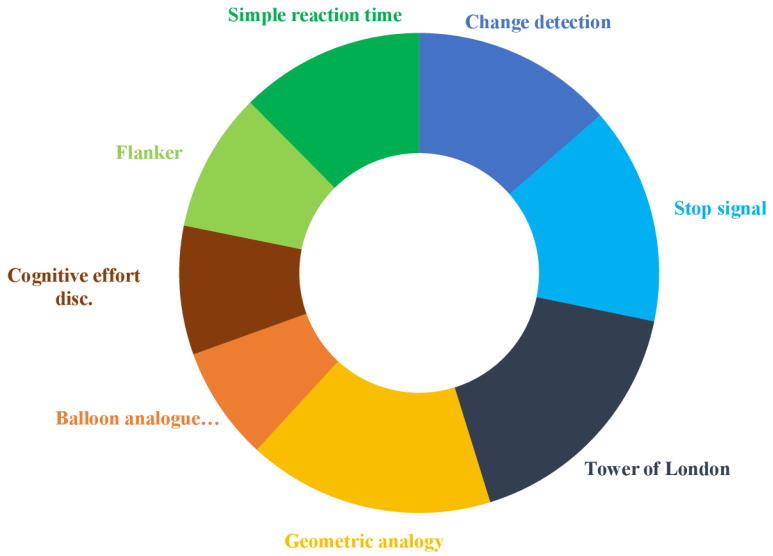
Úloha	Test										Průměr	Std. Dev.
	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.		
Change detection	31,1%	34,6%	30,1%	29,9%	28,9%	27,8%	34,6%	33,5%	33,3%	32,4%	31,61%	2,28%
Stop signal	37,9%	32,4%	32,1%	32,4%	30,7%	33,8%	27,9%	30,5%	30,7%	33,8%	32,21%	2,52%
Tower of London	36,4%	34,0%	38,9%	37,6%	37,6%	38,9%	37,8%	40,0%	36,5%	36,5%	37,43%	1,59%
Geometric analogy	39,5%	39,8%	39,5%	34,8%	37,6%	38,3%	39,3%	36,0%	41,2%	38,1%	38,40%	1,81%
Balloon analogue risk	24,8%	24,3%	32,4%	29,5%	24,1%	24,1%	26,4%	21,2%	19,5%	25,7%	25,19%	3,52%
Cognitive effort disc.	18,3%	20,0%	16,7%	23,3%	20,0%	20,0%	10,0%	13,3%	16,7%	18,3%	17,67%	3,59%
Flanker	24,5%	20,0%	23,1%	18,8%	18,8%	23,1%	23,1%	17,4%	22,9%	23,1%	21,48%	2,34%
Simple reaction time	29,1%	26,4%	23,6%	30,5%	26,4%	30,5%	20,5%	36,3%	35,0%	26,4%	28,48%	4,60%
Průměr	30,20%	28,93%	29,55%	29,61%	28,02%	29,56%	27,45%	28,51%	29,48%	29,30%	**29,06%**	0,80%

Tabulka 7.2: Tabulka přesností 10-ti testů pro model bez skryté vrstvy.

Úloha	Test										Průměr	Std. Dev.
	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.		
Change detection	35,6%	34,4%	33,5%	26,7%	33,3%	30,0%	36,8%	33,3%	33,5%	27,5%	32,46%	3,17%
Stop signal	35,2%	37,6%	28,8%	35,2%	31,9%	34,8%	40,7%	35,0%	33,8%	36,4%	34,95%	3,02%
Tower of London	40,0%	39,0%	38,6%	40,0%	42,2%	41,3%	43,8%	41,0%	39,0%	40,3%	40,51%	1,51%
Geometric analogy	41,2%	38,1%	31,7%	37,9%	41,4%	34,5%	42,9%	42,9%	42,9%	43,1%	39,65%	3,79%
Balloon analogue risk	24,5%	11,4%	26,2%	18,1%	16,4%	19,3%	21,2%	14,8%	16,4%	14,8%	18,31%	4,35%
Cognitive effort disc.	17,7%	20,0%	20,0%	19,3%	21,7%	21,7%	21,7%	17,7%	24,0%	23,3%	20,70%	2,05%
Flanker	26,0%	21,7%	18,8%	26,2%	20,2%	24,5%	20,5%	24,8%	20,2%	21,7%	22,45%	2,53%
Simple reaction time	27,7%	30,4%	33,2%	30,5%	29,3%	30,5%	23,6%	27,7%	30,4%	33,2%	29,64%	2,70%
Průměr	30,98%	29,08%	28,85%	29,24%	29,56%	29,57%	31,38%	29,63%	30,03%	30,04%	**29,83%**	0,76%

Tabulka 7.3: Tabulka přesností 10-ti testů pro model se skrytou vrstvou.

Tabulky 7.2 a 7.3 zachycují přesnosti modelů pro každou z úloh odděleně. Tabulky sestávají z 10-ti testů, kdy každý test obsahuje přesnost modelu pro konkrétní úlohu. Průměrná přesnost zde dosahuje zhruba 30% a opět je na tom nepatrně lépe síť se skrytou vrstvou. Povšimněme si zde směrodatné odchylky, která je v obou případech do 1% což značí poměrně ustálený výsledek. Zároveň z těchto tabulek vyplývá, že pro úlohy *Tower of London* a *Geometric analogy* dosahuje klasifikace nejvyšší přesnosti ($\sim$ 40%), na druhou stranu *Balloon analogue risk* a *Cognitive effort discounting* celkový průměr dost kazí.



Obrázek 7.1: Výsečový graf zobrazující podíl přesností úloh na klasifikaci.

Mimo testování přesnosti klasifikačních modelů metodou *k-fold* byla (spíše pro zajímavost) otestována samotná klasifikace. Ze vstupních dat bylo náhodně vybráno 5 jedinců, z nichž byli dva z IT, dva z financí a jeden z administrativy. Klasifikace byla pro každého z nich spuštěna 10x (tzn. 50 testů). Testovány byly pouze modely se skrytou vrstvou. Níže vidíme matice záměn. U obou vidíme, že správně bylo klasifikováno 15 případů, což z celkového počtu 50 dává přesnost 30%. Tento výsledek víceméně odpovídá očekávání podle metody *k-fold*.

		Predikce			
		Finance	Administ.	IT	Management
Očekávání	Finance	4	5	7	4
	Administ.	4	3	1	2
	IT	3	8	8	1
	Management	0	0	0	0

		Predikce			
		Finance	Administ.	IT	Management
Očekávání	Finance	3	2	7	8
	Administ.	2	2	4	2
	IT	3	2	10	5
	Management	0	0	0	0

Tabulka 7.4: Matice záměn (confusion matrix) pro jeden model (vlevo) vs. pro klasifikování úloh jednu po druhé.

Z obou matic lze vyčíst, že nejúspěšnější klasifikace byla pro třídu informačních technologií, kdežto nejhůře klasifikovanou třídou byly finance. Pokud se vrátíme k obrázku 6.7, je to celkem pochopitelné.

7.1 Shrnutí

Z výše uvedených přesností vyplývá, že klasifikační model dosahuje přesnosti zhruba 30%. Vidíme, že nejlepší schopnost klasifikovat mají nejspíše úlohy *Tower of London* (*ToL*) a *Geometric analogy*, naopak nejhůře je na tom *Cognitive effort discounting*. Myslím si, že toto je způsobeno tím, že úloha je velice obtížná a většina lidí ji asi chtěla mít rychle za sebou a nevěnovala ji dostatek času. Naopak právě *ToL* se řadí k jednodušším a zábavnějším úlohám, u kterých se většina lidí snaží.

Jelikož provádíme klasifikaci do čtyřech tříd, je při náhodné klasifikaci pravděpodobnost správné třídy $\frac{1}{4} = 25\%$. Náš klasifikátor dosahuje nejlépe asi 30% úspěšnosti, což je velice nízká spolehlivost. Testy zhruba potvrzují, co bylo patrné ze zobrazených dat po aplikaci metody *LDA*, že rozdělení do tříd v datech je, ale třídy se velice špatně oddělují. Ovšem 65 vzorků je velice málo pro získání dostatečně oddělených tříd.

Kapitola 8

Závěr

V rámci této diplomové práce byla úspěšně vytvořena webová aplikace pro vyhodnocení kognitivních i emocionálních vlastností člověka. Implementováno bylo 10 vizuálních testů, které jsou inspirovány výzkumy z oblasti neurovědy. Po absolvování celé sady testů dostává uživatel souhrn vyhodnocených konkrétních vlastností. V této chvíli aplikace není určená pro mobilní zařízení, je ovšem téměř jisté, že dříve či později se bude vývoj ubírat i tímto směrem.

Pro získání referenčních dat bylo osloveno několik menších firem, jejichž zaměstnanci test vyplnili. Prostřednictvím sociálních sítí byli osloveni další lidé ochotni se testu zúčastnit. Celkově se do aplikace přihlásilo 220 lidí. Celý test však udělalo pouze 58 z nich. Zhruba polovina z celkového počtu však dokončila 8 z 10 úloh. Toto je způsobeno pozdější doimplementací posledních dvou úloh.

Mimo samotné aplikace testující vlastnosti uživatele byl implementován klasifikátor, který dokáže zařadit člověka do oboru či přímo na pracovní pozici a to na základě jeho výsledků z testování. Ke klasifikaci existuje celá řada metod a přístupů, některé z nich byly popsány v 3. Při návrhu aplikace byly pro realizaci klasifikátoru vybrány umělé neuronové sítě, jejichž koncept byl inspirován a vytvořen na základě analogie s biologickými neurony nervového systému živočichů. Jako základní byla implementována neuronová síť pouze se vstupní a výstupní vrstvou (tzn. lineární regrese), k porovnání pak také síť s jedinou skrytou vrstvou. Navrženy a implementovány byly také dva přístupy ke klasifikaci jako takové. Prvním z nich je vytvoření jediného modelu ze všech výsledků všech úloh a klasifikování na tomto modelu. Druhým je provádět klasifikaci na každé z úloh zvlášť. Největší úspěšnosti bylo dosaženo s druhým přístupem a to s neuronovou sítí se skrytou vrstvou. Tato úspěšnost se pohybovala kolem 30%, podrobnější výsledky a porovnání nabízí kapitola 7. Přesnost klasifikace 30% je velice málo, pro reálné použití není dostačující. Podle mého názoru je tato nízká přesnost způsobena především malým vzorkem lidí. Pro dosažení lepších výsledků by bylo nutné mít v každé z tříd například alespoň 100 vzorků. Dalším faktorem by mohla být nedostatečná motivovanost uživatelů se při testu dostatečně snažit. Test je totiž poměrně dlouhý a vyčerpávající, z tohoto důvodu si myslím, že někteří uživatelé jen prošli testem bez jakékoliv snahy.

Před použitím aplikace v přijímacím řízení by bylo vhodné upravit GUI, které není dotaženo do detailu, avšak se základní funkčností na to aplikace již potenciál má. Zaměstnavatel má možnost vidět přehled vlastností uchazeče, což může sloužit jako podpora k rozhodnutí o přijetí. Pro spolehlivě funkční klasifikátor je však nutné získat o dost větší vzorek dat.

Literatura

[1] MariaDB versus MySQL - Features.
URL https://mariadb.com/kb/en/mariadb/mariadb-vs-mysql-features/

[2] *Nette Framework*.
URL https://nette.org/cs/

[3] *Snap.svg The JavaScript SVG library for the modern web*. [Online; navštíveno 20.12.2016].
URL http://snapsvg.io/

[4] *The Go Programming Language*.
URL https://golang.org

[5] Adam, C.; Andrew, W.; Deanna, B.: Negative symptoms are associated with an increased subjective cost of cognitive effort. *Journal of Abnormal Psychology*, 2016, doi:10.1037/abn0000153.
URL http://dx.doi.org/10.1037/abn0000153

[6] Adele, D.: Executive Functions. *Annual Review of Psychology*, 2013, doi:10.1146/annurev-psych-113011-143750.
URL http://dx.doi.org/10.1146/annurev-psych-113011-143750

[7] Andrew, W.; Daria, K.; Todd S., B.: What Is the Subjective Cost of Cognitive Effort? Load, Trait, and Aging Effects Revealed by Economic Preference. *PLoS ONE*, 2013, doi:10.1371/journal.pone.0068210.
URL http://dx.doi.org/10.1371/journal.pone.0068210

[8] Annika, D.; Isabell, W.; Elke, v. d. M.: The role of fluid intelligence and learning in analogical reasoning. *Neurobiology of Learning and Memory*, 2016, doi:10.1016/j.nlm.2016.07.019.
URL http://dx.doi.org/10.1016/j.nlm.2016.07.019

[9] Berka, P.: *Dobývání znalostí z databází*. 2011.
URL http://sorry.vse.cz/~berka/docs/izi456/

[10] Blogs, C.: One in Five Hires Are “Bad” Hires. 2013.
URL https://www.cebglobal.com/blogs/one-in-five-hires-are-bad-hires-2/

[11] C. W., L.; Jennifer P., R.; Christopher W., K.; aj.: Evaluation of a behavioral measure of risk taking. *Journal of Experimental Psychology: Applied*, 2002, doi:10.1037//1076-898X.8.2.75.
URL http://dx.doi.org/10.1037//1076-898X.8.2.75

[12] Cheng, L.; Bingyu, W.: Fisher Linear Discriminant Analysis. 2014.
URL http://www.ccs.neu.edu/home/vip/teach/MLcourse/5_features_dimensions/lecture_notes/LDA/LDA.pdf

[13] Erin, P.; Jane, W.; James C., B. J.: Human-robot trust and cooperation through a game theoretic framework. Association for the Advancement of Artificial Intelligence, 2016.
URL http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12478

[14] Frederick, V.; Gordon D., L.: Models of response inhibition in the stop-signal and stop-change paradigms. *Neuroscience*, 2009, doi:10.1016/j.neubiorev.2008.08.014.
URL http://dx.doi.org/10.1016/j.neubiorev.2008.08.014

[15] Harvard Business Review: How to Use Psychometric Testing in Hiring. https://hbr.org/2013/09/how-to-use-psychometric-testin/, 2013 [cit. 2016-12-25].

[16] Isabell, W.; Hauke R., H.; Franziska, P.; aj.: Cerebral correlates of analogical processing and their modulation by training. *NeuroImage*, 2009, doi:10.1016/j.neuroimage.2009.06.025.
URL http://dx.doi.org/10.1016/j.neuroimage.2009.06.025

[17] Jack, N.; David, L.; Ian J., D.: Reaction time and intelligence. *Intelligence*, 2013, doi:10.1016/j.intell.2013.08.002.
URL http://dx.doi.org/10.1016/j.intell.2013.08.002

[18] Jeremy R., S.; Cynthia A., R.; Christine L., C.: Concurrent Validity of the Tower Tasks as Measures of Executive Function in Adults. *Applied Neuropsychology*, 2009, doi:10.1080/09084280802644243.
URL http://dx.doi.org/10.1080/09084280802644243

[19] Kimberly A., S.-R.; Eva, O.; Molly Stewart, L.; aj.: Enhancing cognitive and social–emotional development through a simple-to-administer mindfulness-based school program for elementary school children. *Developmental Psychology*, 2015, doi:10.1037/a0038454.
URL http://dx.doi.org/10.1037/a0038454

[20] Lejla, B.; Jip, H.; Jasper, W.: Getting More from PCA: First Results of Using Principal Component Analysis for Extensive Power Analysis. *Springer-Verlag Berlin Heidelberg*, 2012.
URL https://www.riscure.com/benzine/documents/ctrsa12.pdf

[21] M., M.: *Určení struktury a vazeb v proměnných a objektech*. 2016.
URL https://meloun.upce.cz/docs/research/chemometrics/methodology/4cmetody.pdf

[22] Miroslav, S.; Marcel, J.: *Neuronové sítě a neuropočítače*. ČVUT, 1996, ISBN 80-01-01455-X.

[23] Mitchell, T. M.: *Machine Learning*. McGraw-Hill, 1997, ISBN 0070428077.

[24] Motonori, Y.; Gordon D., L.; Patrick G., B.: Stopping while going! Response inhibition does not suffer dual-task interference. *Journal of Experimental Psychology:*

Human Perception and Performance, 2012, doi:10.1037/a0023918.
URL http://dx.doi.org/10.1037/a0023918

[25] Prechelt, L.: Automatic early stopping using cross validation. *Neural Networks*, 1998, doi:10.1016/S0893-6080(98)00010-0.
URL http://linkinghub.elsevier.com/retrieve/pii/S0893608098000100

[26] PubNub: *HTTP Request and Response.*
URL http:
//www.girliemac.com/presentation-slides/PubNub/images/pubnub/req-res.png

[27] PubNub: *Websockets diagram.*
URL
https://www.pubnub.com/wp-content/uploads/2014/09/WebSockets-Diagram.png

[28] Richard D., M.: A Bayesian hierarchical model for the measurement of working memory capacity. *Journal of Mathematical Psychology*, 2011, doi:10.1016/j.jmp.2010.08.008.
URL http://dx.doi.org/10.1016/j.jmp.2010.08.008

[29] Romina, P.; Kirsty B., O.; Joshua M., D.; aj.: New Tests to Measure Individual Differences in Matching and Labelling Facial Expressions of Emotion, and Their Association with Ability to Recognise Vocal Emotions and Facial Identity. *PLoS ONE*, 2013, doi:10.1371/journal.pone.0068126.
URL http://dx.doi.org/10.1371/journal.pone.0068126

[30] Stephen, M.: Task switching. *Trends in Cognitive Sciences*, 2003, doi:10.1016/S1364-6613(03)00028-7.
URL http://dx.doi.org/10.1016/S1364-6613(03)00028-7

[31] Steven J., L.; Edward K., V.: Visual working memory capacity. *Trends in Cognitive Sciences*, 2013, doi:10.1016/j.tics.2013.06.006.
URL http://dx.doi.org/10.1016/j.tics.2013.06.006

[32] UNSW Australia: Why Employers Use Psychometric Assessments. `https://student.unsw.edu.au/why-employers-use-psychometric-assessments`, 2013 [cit. 2016-12-25].

[33] W. Keith, B.; Dana L., B.: The Tower of London Spatial Problem-Solving Task. *Journal of Clinical and Experimental Neuropsychology (Neuropsychology, Development and Cognition: Section A)*, 2002, doi:10.1076/jcen.24.5.586.1006.
URL http://dx.doi.org/10.1076/jcen.24.5.586.1006

[34] William J., B.; Stefan, V.; M.K., W.: Leveraging neuroscience for smarter approaches to workplace intelligence. *Human Resource Management Review*, 2015, doi:10.1016/j.hrmr.2014.09.008.
URL http://dx.doi.org/10.1016/j.hrmr.2014.09.008

Přílohy

Příloha A

Manuál

Pro spuštění klasifikátoru je nutné mít nainstalovaný *Python* ve verzi 2.7, knihovnu *Theano* ve verzi 0.9.0 a *Keras* ve verzi 2.0.3. Data k vyhodnocení jsou uloženy v `data/`, odtud se čte soubor `results.csv` obsahující výsledky všech úloh, nebo jsou z adresáře `data/tasks/` načteny postupně výsledky jednotlivých úloh (podle módu klasifikace). Soubor `test_input` obsahuje 5 náhodně vybraných jedinců pro testování predikce.

Parametry programu

- `-h`, `--help` – nápověda
- `-m 'm'`, `--mode 'm'` – mód klasifikace:
  - `'one'` – vezme postupně všechny úlohy (klasifikuje pro každou z úloh)
  - `'all'` – všechny úlohy se klasifikují zároveň (1 model) - výchozí možnost
- `--hidden` – vzít model se skrytou vrstvou
- `-e`, `--evaluate` – vyhodnocení přesnosti klasifikátoru (pokud není zadán, predikujeme)
- `--save_model 'file'` – po natrénování uloží model do `'model/file'` ve formátu `h5`
- `--load_model 'file'` – místo trénování pouze načte model z `'model/file'`
- `-v`, `--verbose` – kontrolní výpisy

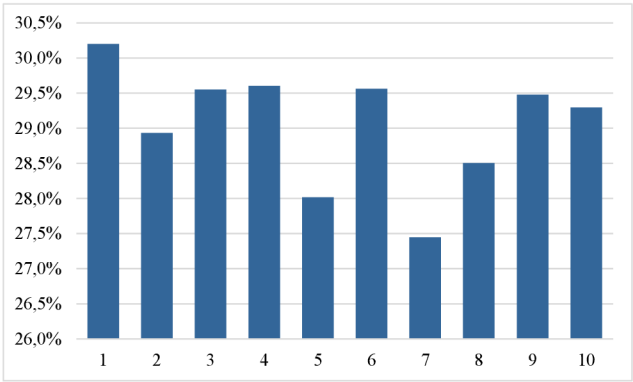
Ukázka spuštění programu: `python main.py -h`

Aplikace je aktuálně dostupná na této adrese:
http://193.85.191.56/?action=testInput&presenter=Testing

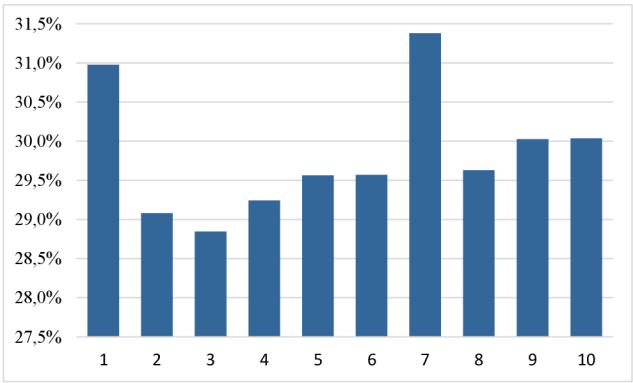
Příloha B

Grafy z výsledků klasifikátoru

Pár dalších doplňujících grafů kapitoly 7.



Obrázek B.1: Úspěšnosti jednotlivých testů pro 7.2



Obrázek B.2: Úspěšnosti jednotlivých testů pro 7.3