

UNIVERZITA KONŠTANTÍNA FILOZOFA V NITRE
FAKULTA PRÍRODNÝCH VIED A INFORMATIKY

**VYUŽITIE PRVKOV UMELEJ INTELIGENCIE
PRI KLASIFIKÁCII EMOCIONÁLNEHO STAVU
NA ZÁKLADE BEHAVIORÁLNEJ
CHARAKTERISTIKY POUŽÍVATEĽA
DIPLOMOVÁ PRÁCA**

UNIVERZITA KONŠTANTÍNA FILOZOFA V NITRE
FAKULTA PRÍRODNÝCH VIED A INFORMATIKY

**VYUŽITIE PRVKOV UMELEJ INTELIGENCIE
PRI KLASIFIKÁCII EMOCIONÁLNEHO STAVU
NA ZÁKLADE BEHAVIORÁLNEJ
CHARAKTERISTIKY POUŽÍVATEĽA**
DIPLOMOVÁ PRÁCA

Študijný odbor: 18. Informatika
Študijný program: Aplikovaná informatika
Školiace pracovisko: Katedra informatiky
Školiteľ: doc. PaedDr. Martin Magdin, Ph.D.



ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Dávid Držík
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: Diplomová práca
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Využitie prvkov umelej inteligencie pri klasifikácii emocionálneho stavu na základe behaviorálnej charakteristiky používateľa

Anotácia: V súčasnom období je evidovaný zvýšený dopyt po riešeníach umelej inteligencie. Prvky umelej inteligencie je možné aplikovať v rôznych odvetviach. Mnoho popredných softvérových výrobcov či už pokusne alebo komerčne hľadá vhodné spôsoby zakomponovania AI do svojich produktov, napríklad v podobe virtuálneho asistenta alebo systému, ktorý by dokázal automaticky rozpoznávať a klasifikovať emocionálny stav používateľa. Emocionálny stav je možné rozpoznávať a klasifikovať aj pomocou behaviorálnych charakteristík, medzi ktoré patria gestá, výraz tváre ale i rýchlosť písania, počet páuz, sila stlačení kláves na klávesnici a rýchlosť pohybu...). Na základe týchto charakteristík je následne možné zvoliť prístup, ktorý je buď kategorický (nabudenie/energia) alebo dimenzionálny (valencia/pôžitok).

Cieľ diplomovej práce: Cieľom bude vývoj riešenia automatického rozpoznávania a klasifikovania emocionálneho stavu používateľa s prvkami umelej inteligencie (napríklad neurónová sieť) na základe behaviorálnych charakteristík – práca na klávesnici a práca s myšou, resp. pri tablete/smartfóne – práca s prstom po obrazovke. Výstup je očakávaný vo forme plne funkčného prototypu riešenia, ktoré musí byť overené na dostatočnej vzorke, aby bola preukázaná percentuálna úspešnosť rozpoznávania a klasifikovania navrhovaného riešenia.

Charakter práce:
- aplikačný

Požiadavky na obsah podľa charakteru práce:
- aplikačný – popis riešeného problému, návrh systému/hardvérového riešenia a pod. (modely, ..), metodika vývoja/tvorby, implementácia, popis vytvoreného riešenia, testovanie

Požiadavky na vedomosti a zručnosti študenta:

- znalosť anglického jazyka za účelom štúdia súvisiacej technickej dokumentácie
- znalosť programovacích jazykov (preferovaný je jazyk C, Java, Python)
- základná znalosť java scriptingu (client side scripting)



Univerzita Konštantína Filozofa v Nitre
Fakulta prírodných vied a informatiky

- znalosť problematiky webových služieb (pre potreby integrácie formou API)

Školiteľ: doc. PaedDr. Martin Magdin, Ph.D.

Oponent: doc. PaedDr. Jozef Kapusta, PhD.

Katedra: KI - Katedra informatiky

Dátum zadania: 19.10.2020

Dátum schválenia: 05.04.2021

RNDr. Ján Skalka, PhD., v. r.
vedúci/a katedry

POĎAKOVANIE

Chcel by som sa poďakovať môjmu školiteľovi, pánovi doc. PaedDr. Martinovi Magdinovi, Ph.D., za vecné pripomienky, cenné rady a odborné konzultácie pri písaní mojej diplomovej práce. Taktiež sa chcem poďakovať mojej bývalej triednej pani učiteľke z Gymnázia Ivana Kupca v Hlohovci, Mgr. Dane Filipkovej, za pomoc pri distribúcii aplikácií vo fáze zberu dát. A v neposlednom rade sa chcem poďakovať mojej sestre a všetkým účastníkom experimentov, bez ktorých by túto prácu nebolo možné zrealizovať.

ABSTRAKT

DRŽÍK, Dávid: Využitie prvkov umelej inteligencie pri klasifikácii emocionálneho stavu na základe behaviorálnej charakteristiky používateľa. [Diplomová práca]. Univerzita Konštantína Filozofa v Nitre. Fakulta prírodných vied a informatiky. Školiteľ: doc. PaedDr. Martin Magdin, Ph.D. Stupeň odbornej kvalifikácie: Magister odboru Aplikovaná informatika. Nitra: FPVaI, 2022. 77 s.

Táto diplomová práca sa zaoberá klasifikáciou emocionálneho stavu používateľa na základe správania sa pri práci s počítačom a so smartfónom. Hlavným cieľom je využitie prvkov umelej inteligencie na dátach získaných z experimentov v našej vlastnej desktopovej a mobilnej aplikácii. V teoretickej časti práce sa venujeme súčasnemu pohľadu na umelú inteligenciu ako aj jednotlivým metódam strojového učenia sa s učiteľom, a taktiež analyzujeme práce autorov, ktorí sa venujú klasifikácii emócií podľa vzorov správania sa. Praktická časť tejto práce pozostáva z dvoch častí. Prvou časťou je samotný vývoj a tvorba experimentov v našich aplikáciách a druhá časť zahŕňa zber, analýzu a predspracovanie dát a použitie vybraných klasifikačných algoritmov na týchto dátach v programovacom jazyku Python. Vo výsledkoch práce a diskusii porovnávame nami dosiahnuté výsledky (dosiahnuté metódou 3-násobnej krížovej validácie) s výsledkami najpodobnejších prác autorov z analýzy súčasného stavu. Najvyššiu klasifikačnú presnosť, až 87,88%, dosahujeme klasifikovaním vzrušenia (nízke, stredné, vysoké) algoritmom neurónovej siete pre druhý dátový súbor. Spomedzi výsledkov klasifikácie valencie (negatívna, neutrálna, pozitívna) najpresnejšie klasifikuje taktiež klasifikátor neurónovej siete pre prvý dátový súbor s presnosťou 85,61%.

Kľúčové slová: Umelá inteligencia. Strojové učenie. Emócia. Klasifikácia. Klávesnica. Myš. Smartfón.

ABSTRACT

DRŽÍK, Dávid: The use of elements of artificial intelligence in the classification of emotional state based on the behavioral characteristics of the user. [Diploma Thesis]. Constantine the Philosopher University in Nitra. Faculty of Natural Sciences and Informatics. Supervisor: doc. PaedDr. Martin Magdin, Ph.D. Degree of Qualification: Master of Applied Informatics. Nitra: FNSaI, 2022. 77 p.

This diploma thesis deals with the classification of the user's emotional state based on the behavior when working with a computer and a smartphone. The main goal is to use elements of artificial intelligence on data obtained from experiments in our own desktop and mobile application. In the theoretical part of the work we deal with the current view of artificial intelligence, as well as individual methods of supervised machine learning and also analyze the work of authors who deal with the classification of emotions according to patterns of behavior. The practical part of this work consists of two parts. The first part is the development and creation of experiments in our applications and the second part includes the collection, analysis and preprocessing of data and the use of selected classification algorithms on this data in the Python programming language. In the results of the work and discussion, we compare the results achieved by us (achieved by the method of 3-fold cross-validation) with the results of the most similar works of the authors from the analysis of the current state. The highest classification accuracy, up to 87.88%, is achieved by classifying arousal (low, medium, high) with the neural network algorithm for the second dataset. Among the results of the valence classification (negative, neutral, positive), the neural network classifier for the first dataset with the accuracy of 85.61% also classifies the most accurately.

Keywords: Artificial intelligence. Machine learning. Emotion. Classification. Keyboard. Mouse. Smartphone.

OBSAH

Úvod	9
1 Analýza súčasného stavu.....	11
1.1 Súčasný pohľad na umelú inteligenciu	11
1.2 Analýza prác autorov venujúcich sa klasifikácii emócií zo vzorov správania	23
2 Ciele diplomovej práce	29
3 Tvorba aplikácií a zber dát.....	30
3.1 Úprava existujúceho kódu	30
3.2 Webový server.....	31
3.3 Databáza.....	35
3.4 Sebahodnotenie v aplikáciách	36
3.5 Tvorba experimentov v desktopovej aplikácii	38
3.5.1 Experiment 1 – Prepis vopred definovaného textu.....	39
3.5.2 Experiment 2 – Prepis textu zvukovej nahrávky	40
3.5.3 Experiment 3 – V správnom poradí	41
3.5.4 Experiment 4 – Hlavalam 8 štvorcov.....	42
3.6 Tvorba experimentov v mobilnej aplikácii	44
3.6.1 Experiment 5 – Prepis vopred definovaného textu.....	45
3.6.2 Experiment 6 – Hlavalam 8 štvorcov.....	47
4 Analýza dát a klasifikácia emócií používateľa.....	49
4.1 Zber dát	49
4.2 Analýza dát.....	49
4.3 Predspracovanie dát.....	51
4.4 Použitie algoritmov strojového učenia	54
4.4.1 K-Nearest Neighbors	54
4.4.2 Logistická regresia.....	57
4.4.3 Naive Bayes.....	59
4.4.4 Neurónová sieť	60
4.4.5 Random Forest.....	64
4.4.6 Support Vector Machine	66
5 Výsledky práce a diskusia	68
Záver	71
Zoznam bibliografických odkazov	72
Zoznam príloh	77

ÚVOD

V súčasnom období už existuje viacero prístupov, ako klasifikovať a predikovať emocionálny stav človeka. Medzi tieto prístupy patrí napríklad klasifikácia na základe fyziologických alebo behaviorálnych prejavov ľudí. Všetky dnešné riešenia predikcie emócií už využívajú prvky umelej inteligencie prostredníctvom natrénovaných modelov strojového učenia sa. Cieľom tejto diplomovej práce bude klasifikovať emocionálny stav používateľa pomocou jeho behaviorálnych prejavov pri práci s počítačom a smartfónom, s využitím prvkov umelej inteligencie.

V prvej kapitole sa budeme teoreticky venovať hlavne súčasnému pohľadu na umelú inteligenciu a jej jednotlivým oblastiam. Pri oblasti strojového učenia sa pozrieme na najpoužívanéjšie algoritmy strojového učenia sa s učiteľom. Okrem toho analyzujeme a popíšeme aj práce autorov, ktorí využili prvky umelej inteligencie pri klasifikácii alebo predikcii emócií používateľov na základe správania sa popri práci s klávesnicou, myšou počítača a smartfónom.

V tretej kapitole, v našej aplikačnej časti, budeme popisovať úpravu už existujúceho zdrojového kódu desktopovej aplikácie našej predchádzajúcej práce (Držík, 2020). V ďalšej časti si rozoberieme význam tvorby používateľského profilu a následnú komunikáciu aplikácií s MySQL databázou prostredníctvom PHP skriptov uložených na vzdialenom webovom serveri. Takisto si popíšeme aj návrh E-R diagramu MySQL databázy spolu so všetkými charakteristikami správania sa, ktoré budeme po vykonaní experimentov ukladať. Najdôležitejšou časťou tejto kapitoly je tvorba jednotlivých experimentov v našich aplikáciách. Pre desktopovú aplikáciu popíšeme vytvorenie dvoch experimentov zameraných na prepis vopred definovaného textu pomocou klávesnice počítača a dvoch experimentov, ktorých úlohou je poklikať objekty za účelom splnenia daných experimentov pomocou myši počítača. Popíšeme aj 2 experimenty vytvorené v mobilnej aplikácii, kde úlohou prvého je prepísať text pomocou nami vytvorenej softvérovej klávesnice a úlohou druhého je pomocou prsta potúkať dané objekty na obrazovke. Na dokončenie experimentov je vyhradený presne definovaný čas. Pred aj po absolvovaní experimentu budú používatelia hodnotiť svoj emocionálny stav na Russelovom kruhovom modeli na dimenziách valencie a vzrušenia.

Vo štvrtej kapitole sa budeme na začiatku venovať fázam zberu, analýze a predspracovaniu dát v programovacom jazyku Python. Predspracovanie bude

pozostávať zo zmeny hodnotenia emócií používateľov z podoby reálnych čísel do podoby troch kategórií emocionálnych stavov pre valenciu aj pre vzrušenie, čím vzniknú dve cieľové premenné. Druhou časťou predspracovania bude zoškálované vstupné a cieľové dáta rozdeliť na tréningové a testovacie množiny. Ďalšou časťou v tejto kapitole bude nasadiť vybrané klasifikačné algoritmy na našich 6 dátových súborov. Pri každom algoritme popíšeme nielen, ako funguje, ale popíšeme aj štruktúru každého jedného dátového súboru. Pre všetky vybrané algoritmy si vyhodnotíme presnosť klasifikácie modelov metódou stratifikovanej 3-násobnej krížovej validácie prostredníctvom metrík accuracy (presnosť) a F1-score pre všetky dátové súbory.

V piatej kapitole, vo výsledkoch práce a diskusii, si vyhodnotíme najvyššie dosiahnuté presnosti klasifikácie vo všetkých dátových súboroch a dosiahnuté výsledky porovnáme s výsledkami nám najpodobnejších prác autorov, ktoré uvádzame v analýze súčasného stavu.

1 ANALÝZA SÚČASNÉHO STAVU

Prvky umelej inteligencie sa v dnešnej dobe používajú v rôznych odvetviach, najmä však v medicíne na diagnostiku chorôb, v automobilovom priemysle na automatické riadenie áut, vo webovom vyhľadávaní, v preklade textu, ale aj v online nakupovaní a v rôznych reklamných odporúčaníach (Europa.eu, 2020). V nasledujúcich podkapitolách si predstavíme súčasný pohľad na umelú inteligenciu ako aj strojové učenie, a taktiež analyzujeme práce iných autorov, ktorí sa snažili klasifikovať emócie pomocou správania sa pri práci s počítačom a so smartfónom.

1.1 SÚČASNÝ POHLED NA UMELÚ INTELIGENCIU

Umelá inteligencia je vedný odbor z odvetvia informatiky, ktorý sa snaží simulovať inteligenciu človeka v stroji, aby následne tieto stroje mohli vykonávať komplexné úlohy, ktoré si zvyčajne vyžadujú ľudskú inteligenciu. Systémy umelej inteligencie sú poháňané algoritmami, ktoré používajú techniky strojového učenia (machine learning), hlbokého učenia (deep learning) a/alebo rozhodovacie pravidlá (decision rules). Algoritmy strojového učenia dodávajú dáta do systémov umelej inteligencie pomocou štatistických techník, ktoré umožňujú systémom umelej inteligencie učiť sa. Vďaka strojovému učeniu sa systémy umelej inteligencie postupne zdokonaľujú v úlohách bez toho, aby na to museli byť špeciálne naprogramované (Escott, 2017; Koďousková, 2021).

Umelú inteligenciu je možné klasifikovať ľubovoľným počtom spôsobov, existujú však dva hlavné typy klasifikácie. Prvým typom je rozdelenie umelej inteligencie podľa schopnosti na *slabú*, *silnú* a *super silnú* umelú inteligenciu.

Slabá umelá inteligencia, nazývaná aj obmedzená umelá inteligencia, je navrhnutá tak, aby vykonávala jedinečné úlohy so zameraním na cieľ. Radí sa sem napríklad rozpoznávanie tváre, reči (virtuálni asistenti), emócií, riadenie áut alebo vyhľadávanie na internete. Je veľmi inteligentná, ale iba pri plnení konkrétnych úloh, na ktoré je naprogramovaná. Aj keď sa stroje s takouto inteligenciou môžu zdať inteligentné, pracujú pod úzkym súborom obmedzení a táto inteligencia nenapodobňuje ani nereplikuje ľudskú inteligenciu, iba simuluje ľudské správanie na základe úzkeho súboru dát (Escott, 2017; Chakraborty, 2021).

Silná umelá inteligencia, často označovaná aj ako hlboká či všeobecná umelá inteligencia, je koncept stroja so všeobecnou inteligenciou, ktorý napodobňuje ľudskú

inteligenciu a ľudské správanie, so schopnosťou učiť sa a pomocou svojej inteligencie je schopný vyriešiť akýkoľvek typ problému. Vedci a tvorcovia umelej inteligencie ešte nedokázali vytvoriť silnú umelú inteligenciu. Aby sa tak stalo, museli by nájsť spôsob, ako dať strojom vedomie a naprogramovať im celý súbor kognitívnych schopností. Stroje by okrem toho mali získavať poznatky a znalosti z prostredia, ktoré by využili v širšom spektre rôznych problémov. Vzhľadom na to, že dnes máme nedostatok komplexných znalostí o funkčnosti ľudského mozgu sa vedci aspoň snažia replikovať základné funkcie zraku a pohybu (Escott, 2017; Chakraborty, 2021; Marr, 2020).

Super silná umelá inteligencia je len hypotetická umelá inteligencia, ktorá nielen napodobňuje a chápe ľudskú inteligenciu a správanie, ale aj stroje si uvedomujú sami seba a prekonávajú kapacitu ľudskej inteligencie a schopností (Escott, 2017).

Druhým typom rozdelenia umelej inteligencie je podľa funkčnosti. Rozlišujeme tu až 4 druhy:

- **reaktívny stroj** je jedným zo základných typov umelej inteligencie a ako z jej názvu plynie, reaguje na existujúce podmienky, avšak nedokáže sa učiť, pretože nemá pamäť. Funguje tak, ako bola naprogramovaná, s predvídateľným výstupom na základe vstupu, ktorý prijíma. Príkladom takéhoto typu umelej inteligencie je šachový turnaj medzi superpočítačom *Deep Blue* od spoločnosti IBM, ktorý v 90. rokoch porazil Garryho Kasparova,
- **obmedzená pamäť** – Systémy umelej inteligencie s obmedzenou pamäťou môžu využiť skúsenosti z minulosti na informovanie o budúcich rozhodnutiach. Pomocou strojového učenia sú modely týchto systémov automaticky trénované na základne dát a následne aktualizované podľa správania sa daného modelu. Umelá inteligencia s obmedzenou pamäťou môže vykonávať komplexné klasifikačné a predikčné úlohy. Tento typ sa dnes veľmi často používa a naďalej sa zdokonaľuje,
- **teória mysle** – Stroj s týmto typom umelej inteligencie by mal byť schopný porozumieť emóciám, myšlienkam, očakávaniam ľudí a byť schopný sociálne komunikovať. Aj keď teória mysle umelej inteligencie nie je úplne dosiahnutá, humanoidná robotka *Sophia*, s podobizňou človeka bola schopná „cítiť“ emócie a aj primerane na ne reagovať,
- **sebavedomie** – Posledným typom umelej inteligencie je stroj s vlastným vedomím. Stroje si nebudú uvedomovať len emócie a duševné stavy ostatných,

ale aj svoje vlastné. Keď sa dosiahne sebavedomá umelá inteligencia, ktorá má vedomie na úrovni ľudí a rovná sa ľudskej inteligencii s rovnakými potrebami, túžbami a emóciami, bude to veľmi dôležitý míľnik v tejto oblasti (Marr, 2021).

Umelá inteligencia je ako veda delená do ďalších siedmich skupín. Medzi tieto skupiny patrí strojové učenie (machine learning), spracovanie prirodzeného jazyka (natural language processing), expertné systémy, strojové (počítačové) videnie, rozpoznávanie reči, plánovanie a robotika (Kumar, 2018). Skupina strojového učenia nás bude zaujímať najviac, pretože tam sa orientuje téma našej práce najviac. Najskôr si však v krátkosti predstavíme ostatné skupiny umelej inteligencie.

Spracovanie prirodzeného jazyka sa zaoberá interakciami medzi počítačom a prírodnými jazykmi a delí sa na dve podkategórie: *porozumenie prirodzenému jazyku* a *generovanie prirodzeného jazyka*. Kombinuje sa tu strojové učenie a lingvistika. Spracovanie prirodzeného jazyka sa dnes používa najmä v prekladačoch, v chatbotoch, vo virtuálnych asistentoch a úzko súvisí aj s **rozpoznávaním reči** (Bukovčanová, 2021).

Expertný systém je počítačový systém, ktorý v presne vymedzenej oblasti problému umožní dospieť k rozhodnutiu, k akému by pravdepodobne dospel aj odborník na danú oblasť (expert). Expertné systémy sú navrhnuté tak, aby riešili zložité problémy uvažovaním prostredníctvom súborov znalostí, reprezentovaných hlavne pravidlami. Expertný systém je rozdelený do dvoch subsystémov: *riadiaci mechanizmus* a *báza znalostí*. Báza znalostí predstavuje hlavne fakty a pravidlá. Riadiaci mechanizmus aplikuje pravidlá na známe skutočnosti, aby vyvodil nové skutočnosti a môže tiež zahŕňať aj vysvetľujúce schopnosti, prečo tak systém rozhodol (Zwass, 2015).

Strojové videnie umožňuje počítačom a systémom získavať zmysluplné informácie z digitálnych obrázkov, videí a iných vizuálnych vstupov. Pomocou strojového a hlbokého učenia vieme z daných multimediálnych vstupov rozpoznávať tváre ľudí, objekty, dokonca aj emócie či vek človeka. Je však potrebné zozbierať dostatočne veľa vzoriek, aby sa to model naučil rozpoznávať (IBM.com, 2020).

Úlohou **plánovania** v umelej inteligencii je výber komplexnej postupnosti činností systému, pre dosiahnutie svojho cieľa, čo najvyhovujúcejším spôsobom. Keď je toto plánovanie úspešné, umožňuje automatizáciu úloh. V dynamických prostrediach, napríklad v reálnom svete, s neustálymi zmenami sa vyžaduje časté doladovanie

pomocou metódy pokus a omyl. Obyčajne plánovanie vykonávajú inteligentní agenti, autonómne roboty a autonómne autá (Adler, 2019).

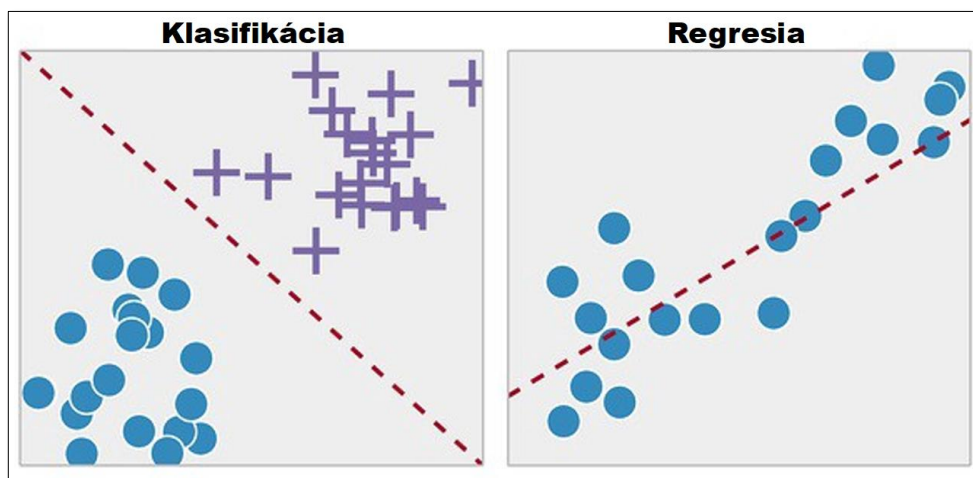
Robotika je odvetvie umelej inteligencie, ktoré navrhuje, vyvíja a nasadzuje roboty v hardvérovej alebo softvérovej podobe. Roboty môžu byť vybavené senzormi, ktoré sú ekvivalentom ľudských zmyslov. Niektoré sú dokonca schopné jednoduchého rozhodovania a súčasný výskum robotiky sa zameriava na vývoj robotov so stupňom sebestačnosti, ktorý umožní neštruktúrovanú mobilitu (Adler, 2019).

Strojové učenie je podoblasťou umelej inteligencie, ktorá poskytuje systémom schopnosť automaticky sa učiť bez toho, aby boli na to explicitne naprogramované. Algoritmy strojového učenia zostroja model na základe vstupných historických dát z dátového súboru. Následne sa na týchto historických dátach snažia nájsť podobnosti, vzorce alebo sa len naučia využiť výsledky na ich zovšeobecnenie. Algoritmy strojového učenia sú navrhnuté tak, aby sa už naučený model stále automaticky zlepšoval pomocou nových dát. Dobre naučený model je neskôr schopný sám predikovať výsledný stav nových nezatriedených dát (Exper.ai, 2020). Strojové učenie sa spravidla rozdeľuje na *učenie s učiteľom*, *učenie bez učiteľa* a *posilňované učenie*.

Učenie s učiteľom je typ strojového učenia, ktorý je používaný v prípade, že sa v dátovom súbore nachádzajú vstupné charakteristiky (x) a k nim zodpovedajúce výstupné cieľové (y) charakteristiky. Sú tu teda k dispozícii príklady správneho správania a správnej zmeny vstupných charakteristík na výstupnú cieľovú charakteristiku. Priebeh učenia sa spočíva v tom, že sa využíva tréningová časť dátového súboru (tvorí zvyčajne 70 až 80% dátového súboru, zvyšok tvorí testovacia časť), na ktorej sa algoritmus v priebehu času postupne učí porovnávaním svojho výstupu so správnymi výstupmi a meria svoju presnosť pomocou stratovej funkcie a upravuje ju, pokiaľ chyba nie je dostatočne malá. Učenie s učiteľom možno zaradiť do dvoch kategórií algoritmov (Ongsulee, 2017; Expert.ai, 2020):

- **klasifikácia** – Pri klasifikačných úlohách musí program strojového učenia vyvodiť záver z pozorovaných hodnôt a určiť, do ktorej kategórie patria nové pozorovania. Cieľová charakteristika musí byť preto kategorická. Najbežnejšie používané klasifikačné algoritmy sú *Rozhodovacie stromy*, *K-nearest neighbors*, *Naive Bayes*, *Support Vector Machine (SVM)* a *Logistická regresia*,
- **regresia** – Pri regresných úlohách musí program strojového učenia odhadnúť vzťahy medzi charakteristikami. Regresná analýza sa zameriava na jednu závislú

premennú (cieľová premenná je konkrétna hodnota) a sériu ďalších meniacich sa premenných, čo je obzvlášť dôležité pre predikciu. Najobľúbenejšie regresné algoritmy sú *jednoduchá a viacnásobná lineárna regresia* (Kumar, 2018).



Obrázok 1 Klasifikácia a regresia¹

Učenie bez učiteľa, tiež nazývané aj učenie bez dozoru, pracuje na rozdiel od učenia s učiteľom iba s dátovým súborom, ktorý obsahuje len vstupné charakteristiky (x) a neobsahuje žiadne výstupné cieľové (y) charakteristiky. Pretože tu nie sú cieľové premenné, učenie bez učiteľa prebieha na vstupných dátach, na ktorých nachádza štruktúru napríklad zoskupovaním. Algoritmus strojového učenia sa pokúša tieto údaje nejakým spôsobom usporiadať, aby v nich našiel ich vnútornú štruktúru. To môže znamenať zoskupenie údajov do klastrov alebo ich usporiadanie spôsobom, ktorý vyzerá organizovanejšie. Učenie bez učiteľa môžeme ďalej rozdeliť na dva typy:

- **klastrovanie** je metóda zoskupovania objektov do klastrov (zhlukov) tak, že objekty podobné s inými objektami sú v jednej skupine a majú malú alebo žiadnu podobnosť s objektami inej skupiny. Pomocou klastrovej analýzy sa zisťujú spoločné znaky medzi dátovými objektami a kategorizujú sa podľa prítomnosti a neprítomnosti týchto spoločných znakov. Najznámejší a najpoužívanější algoritmus pre uplatnenie zhľukovania je *K-means*.
- **asociácia** alebo asociačné pravidlo sa používa na nájdenie vzťahov medzi premennými vo veľkej databáze. Určuje množinu položiek, ktoré sa vyskytujú spoločne v množine údajov. Typickým príkladom asociačného pravidla je analýza nákupného košíka na trhu (Mishra, 2017; Ongsulee, 2017).

¹ Zdroj Obrázka 1: <https://blog.evolvemachinelearners.com/what-is-machine-learning/>

Kombináciou dvoch predošlých metód strojového učenia vzniká **čiasťočné učenie s učiteľom**. Na natréňovanie sa používa malé množstvo vstupných dát spolu s cieľovou premennou a veľké množstvo dát bez tejto cieľovej premennej, čím sa môže výrazne zlepšiť presnosť učenia (Expert.ai, 2020; Mishra, 2017; Ongsulee, 2017).

Posilňované učenie je učenie modelov strojového učenia, pri ktorom sa prijímajú rôzne rozhodnutia na maximalizáciu odmeny v konkrétnej situácii. Agent sa učí dosiahnuť cieľ v neistom, potenciálne komplexnom prostredí. Používa sa na nájdenie najlepšieho možného správania alebo cesty, ktorou by sa mal v konkrétnej situácii uberať. Pracuje s metódou pokus a omyl, aby našiel správne riešenie problému. Umelá inteligencia dostáva na záver riešenia buď odmeny alebo tresty za činnosti, ktoré vykonáva s cieľom maximalizovať odmenu. Posilňované učenie sa líši od učenia s učiteľom v tom, že pri učení neexistujú žiadne výstupné cieľové premenné, ale agent rozhodne, čo má robiť, aby splnil danú úlohu. Najznámejším príkladom algoritmu pre posilňované učenie je *Q-learning* (Osiński, et al., 2018; Mishra, 2017; Expert.ai, 2020).

Strojové učenie, ktoré je podmnožinou samotnej umelej inteligencie, samo obsahuje ešte jednu dôležitú podmnožinu, ktorou je **hlboké učenie** (deep learning). Definícia hlbokého učenia je, že poskytuje schopnosť stroju vykonávať úlohy podobné človeku bez jeho účasti so schopnosťou napodobňovať činnosť ľudského mozgu (aj keď zďaleka nezodpovedá jeho schopnostiam). Hlboké učenie je implementované prostredníctvom architektúry neurónových sietí, preto majú tieto siete názov **hlboké neurónové siete**. Hlboké učenie sa tiež rozdeľuje tak ako strojové učenia na *učenie s učiteľom*, *učenie bez učiteľa* a *posilňované učenie* (Brownlee, 2019; Ongsulee 2017). V nasledujúcich častiach si popíšeme najpoužívanéjšie metódy strojového a hlbokého učenia s učiteľom.

Logistická regresia

Logistická regresia je mimoriadne flexibilnou metódou pre klasifikačné problémy. Modely logistickej regresie môžu byť použité pre **binárnu** (kategorická premenná má len dva možné stavy), **multinominálnu** (kategorická premenná môže mať 3 a viac možných stavov) aj **ordinálnu** klasifikáciu. Rieši mnohé problémy, ktoré lineárna regresia nevyrieši, pretože namiesto predpovedania číselnej hodnoty, predpovedá diskretnú hodnotu. Ako klasifikačné problémy, pre ktoré je logistická

regresia vhodná, je často uvádzaná zdravotná diagnostika, kde sa klasifikuje, či daná osoba má alebo nemá konkrétne ochorenie (Seufert, 2014).

Logistická regresia, podobne ako lineárna regresia, vytvára model vzťahu medzi viacerými premennými a je pomenovaná podľa logistickej funkcie, nazývanej tiež *sigmoidálna funkcia*. Je to krivka v tvare písmena S, ktorá dokáže akékoľvek číslo s reálnou hodnotou znormalizovať na hodnotu v intervale 0 a 1. Logistická regresia používa ako reprezentáciu rovnicu (Vzorec 1), veľmi podobnú lineárnej regresii. Vstupné hodnoty (x) sa lineárne kombinujú pomocou váh alebo hodnôt koeficientov na predpovedanie výstupnej hodnoty (y). V rovnici je y predikovaný výstup, β_0 je bias a β_1 až β_n sú koeficienty pre jednu vstupnú hodnotu (x_1 až x_n). Každý stĺpec vo vstupných dátach má priradený koeficient β (konštantná skutočná hodnota), ktorý sa natrénuje z tréningových dát (Seufert, 2014; Polamuri, 2017).

$$y = \frac{e^{(\beta_0 + \beta_1 x_1 + \dots + \beta_n x_n)}}{1 + e^{(\beta_0 + \beta_1 x_1 + \dots + \beta_n x_n)}}$$

Vzorec 1 Rovnica logistickej regresie

Naive Bayes

Naive Bayes klasifikátor je kolekciou pravdepodobnostných klasifikačných algoritmov strojového učenia založených na Bayesovej vete. Nejde tu len o jediný algoritmus, ale o skupinu algoritmov, kde všetky zdieľajú spoločný princíp, že každá premenná je nezávislá a navzájom neinteraguje, takže každá nezávislá premenná rovnako prispieva k pravdepodobnosti, že vzorka patrí do konkrétnej triedy. Naive Bayes je ľahko implementovateľný a výpočtovo rýchly a dobre funguje vo veľkých množinách údajov s vysokou dimenzionalitou. Spracováva tréningovú časť dátového súboru výpočtom pravdepodobností tried $P(y_i)$ a podmienených pravdepodobností, ktoré definujú frekvenciu každej hodnoty premennej pre danú triedu delenú frekvenciou inštancií s touto hodnotou triedy (Gandhi, 2018).

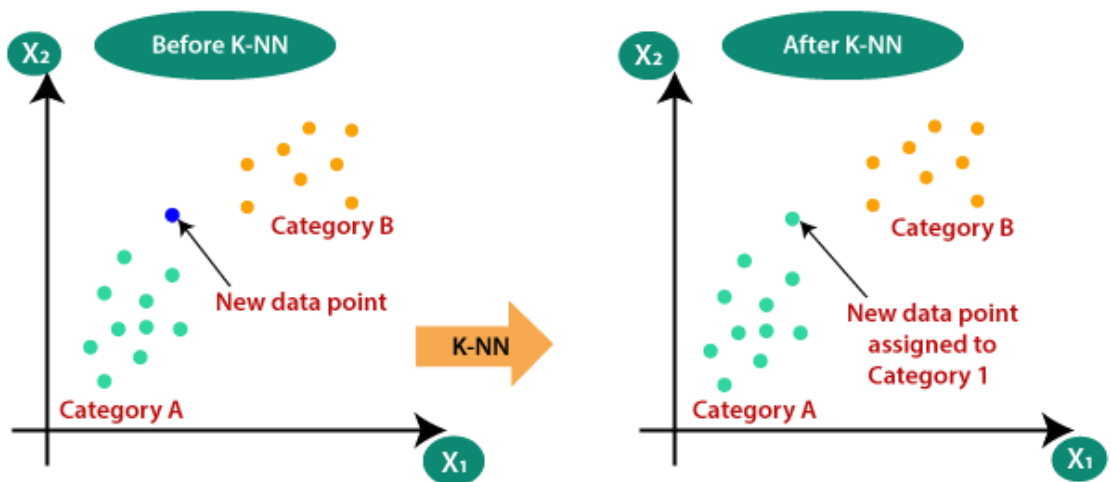
$$y = \operatorname{argmax}_y P(y) \prod_{i=1}^n P(x_i|y)$$

Vzorec 2 Triedna pravdepodobnosť Naive Bayes klasifikátora

Za účelom vytvorenia klasifikačného modelu nájdeme pravdepodobnosť danej sady vstupných premenných (x_i) pre všetky možné hodnoty cieľovej premennej (y) a vyberieme výstup s maximálnou pravdepodobnosťou. Matematicky je to vyjadrené pomocou Vzorca 2 (Gandhi, 2018).

K-nearest neighbors

K-nearest neighbors (K-NN) je typ algoritmu strojového učenia, ktorý rieši regresné, ale najmä klasifikačné problémy pri učení s učiteľom. Prostredníctvom tohto algoritmu sa dá predikovať správna trieda pre testovacie dáta vypočítaním vzdialenosti medzi testovacími dátami a všetkými tréningovými dátami. Ďalej sa vyberie K počet dát, ktorý je blízky testovacím dátam. Algoritmus potom počíta pravdepodobnosť, že testovacie dáta patria triedam tréningových dát a vyberie sa trieda s najvyššou pravdepodobnosťou. Pre vypočítanie vzdialenosti medzi bodmi sa najčastejšie používa Euklidovská vzdialenosť, ale je možné použiť aj Manhattanskú a Hammingovu vzdialenosť. V nasledujúcom diagrame je ilustrovaná predikcia nového bodu, ktorý je pridelený do kategórie A (Christopher, 2021).



Obrázok 2 Diagram predikcie prostredníctvom algoritmu K-NN²

Support Vector Machine

Cieľom algoritmu Support Vector Machine je vytvoriť najlepšiu hranicu prostredníctvom čiary (vektora), ktorá môže rozdeľovať n-rozmerný priestor do dvoch tried, aby sa neskôr pri predikcii mohol nový dátový bod ľahko zaradiť do správnej kategórie. Táto najlepšia hranica sa nazýva aj *nadrovina (hyperplane)* a pri jej vytváraní pomáhajú extrémne body. Tento algoritmus taktiež patrí medzi klasifikačné algoritmy strojového učenia pri učení s učiteľom. Body najbližšie k nadrovine sú známe ako *podporné body vektora*, pretože iba tieto dva body prispievajú k výsledku algoritmu a

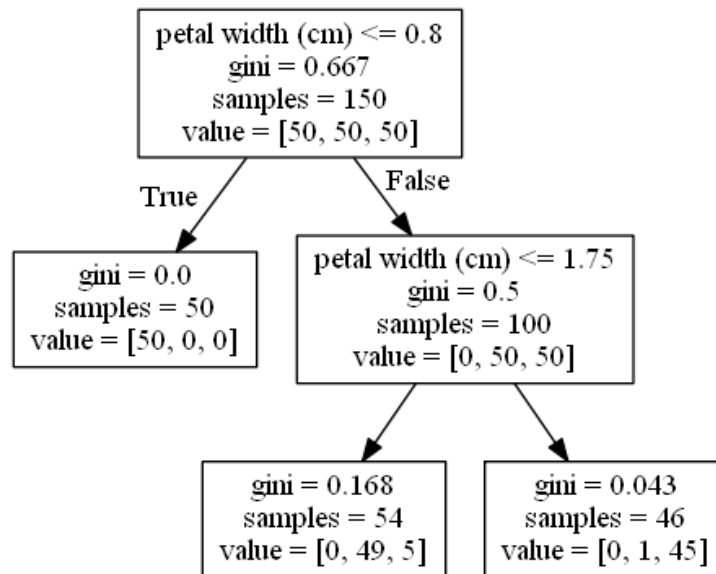
² Zdroj Obrázka 2: https://miro.medium.com/max/656/0*OltO4Txr-D0lPWNL.png

ostatné body nie. Ak bod nie je podporným vektorom, jeho odstránenie nemá na model žiadny vplyv. Rozmer nadroviny závisí od počtu vstupných premenných. Ak je počet vstupných premenných 2, potom je hranica len čiara. Ak je počet vstupných premenných 3, potom sa z hranice stane dvojrozmerná rovina (Baeldung.com, 2020).

Rozhodovacie stromy

Rozhodovacie stromy patria medzi najpoužívanejšie metódy strojového učenia, ktoré sa uplatňujú pri klasifikačných a regresných problémoch. Rozhodovací strom je hierarchický, viacstupňový rozhodovací systém, ktorý obsahuje súbor riadiacich príkazov na podporu rozhodovania. Rozhodovací strom je tvorený z uzlov a obsahuje 2 hlavné typy uzlov. Prvým typom je **vnútorný (rozhodovací) uzol** a predstavuje rozhodovanie podľa jedného atribútu údajov, ktorý spomedzi ostatných atribútov najlepšie rozdeľuje dáta podľa cieľovej premennej. Najvrchnejšiemu uzlu sa hovorí aj koreňový uzol. Všetky vnútorné uzly môžu ďalej obsahovať ďalšie vnútorné uzly, a/alebo obsahujú aj listy, ktoré sú druhým typom uzla. **Listový uzol** je konečný uzol stromu, ktorý obsahuje len výstupnú cieľovú premennú. Ak sa pomocou daných rozhodnutí dostaneme až do konkrétneho listu, obsah listu (cieľová premenná) je výsledkom algoritmu. Je potrebné zabezpečiť, aby strom nebol ani príliš veľký, ani príliš malý. V prípade veľkých stromov je potrebné vykonávať prerezávanie stromu (Navlani, 2018).

Rozhodovacie stromy sú vytvárané rekurzívnym rozdelením tréningových dát podľa atribútov, ktoré dáta najlepšie rozdeľujú. To sa dosiahne vyhodnotením určitých metrík, ako je *Giniho index nečistoty* alebo *Entropia* pre kategorické rozhodovacie stromy a *zvyšková* alebo *priemerná kvadratická chyba* pre regresné stromy. Cieľom všetkých týchto metrík je docielenie správneho rozdelenia cieľovej premennej tak, aby hodnoty cieľovej premennej boli čo najviac homogénne vo vnútri uzla a medzi uzlami by mali byť najrozdielnejšie. To je možné vidieť aj na nasledujúcom Obrázku 3, že po prvom rozdelení je Gini index 0,0, čím sa jednoznačne oddelila jedna klasifikačná skupina od zvyšných dvoch, ktoré majú Gini index 0,5 a po rozdelení má druhá skupina 0,168 a tretia 0,043 (Navlani, 2018).



Obrázok 3 Rozhodovací strom pre Iris dataset³

Najčastejšie používaným algoritmom, ktorý využíva princíp rozhodovacích stromov, je algoritmus ID3, poprípade jeho rozšírená verzia C4.5. Tento algoritmus používa, už vyššie spomínanú, metriku Entropia. Hlavnou výhodou rozhodovacích stromov je **ľahká interpretácia výsledkov**. Zatiaľ čo ostatné modely strojového učenia sú takmer ako čierne skrinky, rozhodovacie stromy poskytujú grafický spôsob, ako porozumieť tomu, čo algoritmus robí. A na rozdiel od iných metód, nepožadujú na natrénovanie modelu veľké množstvo dát. Nevýhodou rozhodovacích stromov je, že sú náchylné na **preučenie sa** a sú citlivé na extrémne hodnoty (Navlani, 2018).

Jeden rozhodovací strom má sám osebe porovnateľnú presnosť v porovnaní s inými algoritmami strojového učenia. Jeden strom však zvyčajne nevytvára najlepšie predpovede, ale stromová štruktúra áno. V algoritmoch, ktoré kombinujú viacero stromov, ako napríklad **Random Forest**, má model oveľa lepší výkon v porovnaní s jediným rozhodovacím stromom (Navlani, 2018).

Neurónové siete

Umelé neurónové siete sú inšpirované skutočnými biologickými neurónovými sieťami v mozgu človeka. Umelé neurónové siete sa skladajú z rôznych vrstiev (vstupná, skrytá, výstupná) prepojených umelých neurónov poháňaných aktivačnými funkciami, ktoré dodávajú určitý výstup na základe vstupu. Každé spojenie medzi

³ Zdroj Obrázka 3: https://miro.medium.com/max/491/1*4ztnUngyDvkAd4dcR8H6dA.png

umelými neurónmi, podobne ako synapsie v mozgu človeka, môže prenášať signál do iných neurónov. Jednotlivé spojenia majú buď excitačný (zosilňujúci) alebo inhibičný (zoslabujúci) charakter, ktorý je určený na základe ich číselnej hodnoty váhy (sily prepojenia) (Great Learning, 2020).

Každý neurón dostane na vstup vynásobené hodnoty vstupov (x_i) a váh (w_i), ktoré sa potom sčítajú aj spolu s prahovou hodnotou neurónu, nazývaný aj ako bias (b). Výsledná hodnota sa potom preniesie do vhodnej aktivačnej funkcie, ktorá rozhodne o konečnej hodnote, ktorá má byť výstupom neurónu. Podľa povahy vstupných hodnôt sú k dispozícii rôzne aktivačné funkcie. Akonáhle je známy výstup z výstupnej vrstvy neurónovej siete, vypočíta sa strata (porovnanie požadovaného výstupu s dosiahnutým výstupom) a vykoná sa *metóda spätného šírenia chyby* (backpropagation), kde sa váhy jednotlivých neurónov upravujú tak, aby bola strata minimálna (Great Learning, 2020).

$$y = \varphi(w_1x_1 + \dots + w_nx_n - b) = \varphi\left(\sum_{i=1}^n w_ix_i - b\right)$$

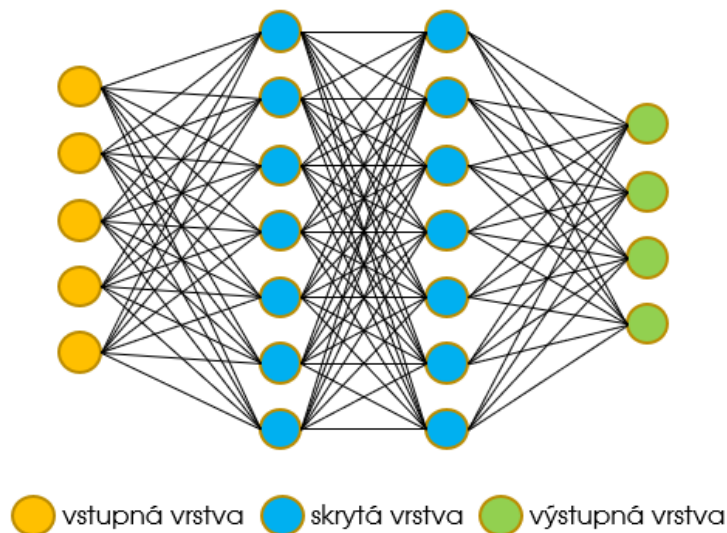
Vzorec 3 Výstup umelého neurónu s aktivačnou funkciou

Existuje veľa typov neurónových sietí, väčšina z nich sa radí medzi hlboké neurónové siete. Jedným z najjednoduchších modelov neurónu je model **Perceptrónu**, ktorý navrhli Minský a Papert, a je to najmenšia jednotka neurónovej siete, ktorá robí výpočty vstupov s váhami, čím vo výsledku rozdeľuje vstupný priestor na dve polovice v spolupráci s aktivačnou funkciou. Môže byť použitý teda iba na implementáciu lineárne oddeliteľných funkcií ako sú napríklad Booleovské funkcie *AND*, *OR*, *NOT*, ale nie *XOR*, pretože to nie je lineárne oddeliteľná funkcia (Great Learning, 2020).

Dopredné neurónové siete sú najjednoduchšou formou neurónových sietí, kde vstupné dáta cestujú smerom dopredu a vystupujú cez výstupné neuróny. Všetky váhy v tejto sieti sú statické. Môžu, ale aj nemusia obsahovať skryté vrstvy. Na základe toho môžu byť ďalej klasifikované ako jednovrstvové alebo viacvrstvové dopredné neurónové siete (Great Learning, 2020).

Viacvrstvový perceptrón je viacvrstvová dopredná neurónová sieť a obsahuje minimálne vstupnú, skrytú a výstupnú vrstvu. Ak obsahuje viac ako jednu skrytú vrstvu, radí sa táto sieť medzi siete s hlbokým učením. Viacvrstvový perceptrón využíva učiacu sa metódu spätného šírenia chyby (backpropagation), ktorou mení hodnoty jednotlivých váh v každom kroku učenia sa s cieľom zminimalizovať veľkosť chyby porovnávaním predikovaných výstupov s reálnymi cieľovými výstupmi. Používajú sa tu nelineárne

aktivačné funkcie, najpoužívanejšie sú funkcie *Sigmoid*, *Hyperbolický Tangens*, *ReLU* a funkcia *SoftMax*, ktorá sa používa len na výstupnej vrstve. Tento druh neurónovej siete sa často používa pri klasifikačných alebo regresných problémoch na dátovom súbore v tabuľovom formáte (Great Learning, 2020).



Obrázok 4 Viacvrstvový perceptrón⁴

Rekurentná neurónová sieť bola navrhnutá na riešenie problémov predikcie sekvencií. Funguje tak, že výstup z vrstvy je znova privádzaný späť na vstup, a tak pomáha predpovedať výsledok danej vrstvy. Prvou vrstvou je typicky dopredná neurónová sieť a za ňou je vrstva rekurentných neurónových sietí, kde výstupy, ktoré mala v predchádzajúcom časovom kroku, si ukladá pre budúce použitie. Ak je predpoveď nesprávna, pomocou spätného šírenia chyby sa chybovosť minimalizuje. Tento typ neurónovej siete je používaný hlavne na preklad a spracovanie textu, ale aj na kontrolu gramatiky a transformáciu textu na reč (Great Learning, 2020).

Konvolučná neurónová sieť obsahuje namiesto štandardného dvojrozmerného poľa, trojrozmerné usporiadanie neurónov. Prvá vrstva sa nazýva konvolučná vrstva. Každý neurón v konvolučnej vrstve spracováva informácie len z malej časti zorného poľa. Pre dokončenie úplného spracovania obrazu musí sieť po častiach porozumieť daným obrázkom. Spracovanie zahŕňa konverziu obrazu z RGB modelu do odtieňov sivej. Zmeny v pixeloch pomôžu zaradiť obrázky do rôznych kategórií. Táto sieť obsahuje jednu alebo viac konvolučných vrstiev s dopredným spracovaním, po ktorých nasleduje združovanie výsledkov, kde výstup konvolučnej vrstvy prechádza do plne

⁴ Zdroj Obrázka 4: <https://umelainteligencia.sk/wp-content/uploads/2019/11/NN-uvod-siet.png>

prepojenej neurónovej sieti na klasifikáciu obrázkov, ktorá využíva dopredné spracovanie aj metódu spätného šírenia chyby. Konvolučné neurónové siete vykazujú veľmi účinné výsledky v rozpoznávaní obrázkov a videa (Great Learning, 2020).

1.2 ANALÝZA PRÁC AUTOROV VENUJÚCICH SA KLASIFIKÁCIÍ EMÓCIÍ ZO VZOROV SPRÁVANIA

V našej predošlej práci (Držík, 2020) sme analyzovali práce autorov, ktorí sa snažili buď hľadať súvislosti medzi charakteristikami správania sa pri práci s počítačom a medzi emóciami, alebo sa snažili jednotlivé charakteristiky rovno kategorizovať do skupín, ktoré tvorili jednotlivé emócie. V tejto podkapitole analyzujeme najzaujímavejšie práce klasifikácie emócií na základe správania sa pri práci s počítačom, ale aj pri práci so smartfónom s využitím prvkov umelej inteligencie.

Nahin (2014) sa s jeho kolektívom snažil identifikovať emócie používateľov len prostredníctvom klávesnice počítača a to kombináciou dynamiky stláčania klávesov a analýzou textu. Ďalej uvádza, že tento kombinovaný prístup ešte nebol použitý v žiadnej z prác a potenciálne by mohol zvýšiť presnosť klasifikovaných emócií. Proces zberu dát spočíval v dvoch úlohách. Prvou úlohou bolo prepísať vopred zadaný text aspoň jedenkrát za deň po dobu niekoľkých dní a následne mali používatelia ohodnotiť svoj emocionálny stav zakliknutím jedného zo siedmich konkrétnych stavov (radosť, strach, hnev, smútok, znechutenie, hanba, vina). Táto úloha bola naprogramovaná v jazyku Java a ukladala všetky stlačené klávesy s ich časom stlačenia a uvoľnenia do logovacieho súboru. Druhá úloha bola naprogramovaná v jazyku C#, bežala na pozadí a cieľom programu bolo zaznamenávať všetky interakcie s klávesnicou. Tento program zaznamenával všetky charakteristiky ako aj v prvej úlohe, ale navyše ukladal aj text, ktorý používateľ napísal. Každých 30 minút bol používateľ vyzývaný, aby ohodnotil svoj emocionálny stav. Celkovo sa tohto experimentu zúčastnilo 25 osôb a experiment trval celkovo 7 týždňov. Po zbere dát zo stlačení a uvoľnení klávesov extrahoval až 19 atribútov, medzi ktoré patria napríklad rýchlosť písania, doba držania klávesu, doba medzi stlačeniami a podobne. Na ďalšiu analýzu atribútov, použil nástroj Weka, ktorý mu z 19 atribútov vybral 7 základných, ktoré budú najlepšie predikovať emocionálny stav. V nástroji Weka rozdelil všetky získané dáta na tréningové a testovacie a na natréňovanie modelov využil tieto algoritmy strojového učenia: *jednoduchá regresia*, *neurónová sieť viacvrstvový Perceptrón* a *Random forest (J48 a BF)*. Následne na testovacích dátach otestoval presnosť daných modelov. Autor však

neuvádza, akým algoritmom získal danú presnosť, uvádza len najvyššiu dosiahnutú pre každú z emócií. Priemerne, pre obe úlohy iba pomocou metódy dynamiky stláčania klávesov dosiahol najvyššiu presnosť 84,5% pre emóciu radosti a najnižšiu presnosť 65,5% pre smútok. Druhou metódou bolo identifikovať emócie na základe analýzy textu. Pre tento účel použil súbor dát ISEAR, ktorý obsahuje viac ako 7500 viet spolu s označením emocionálnej kategórie. Z druhej úlohy vyexportoval všetky vety, z ktorých odstránil všetky stop slová (často sa vyskytujúce, no samy o sebe nenesú žiadnu významovú informáciu) a neemotívne slová. Na klasifikáciu viet použil VSM (vector space model je algebrický model na reprezentáciu textových dokumentov ako vektory) s Jaccardovou podobnosťou. Kombináciou metód dynamiky stláčania klávesov a analýzy textu sa dokázala zlepšiť presnosť identifikácie emócií, konkrétne najlepšie sa identifikovala radosť s 87% a najhoršie emócia hanby s 69%.

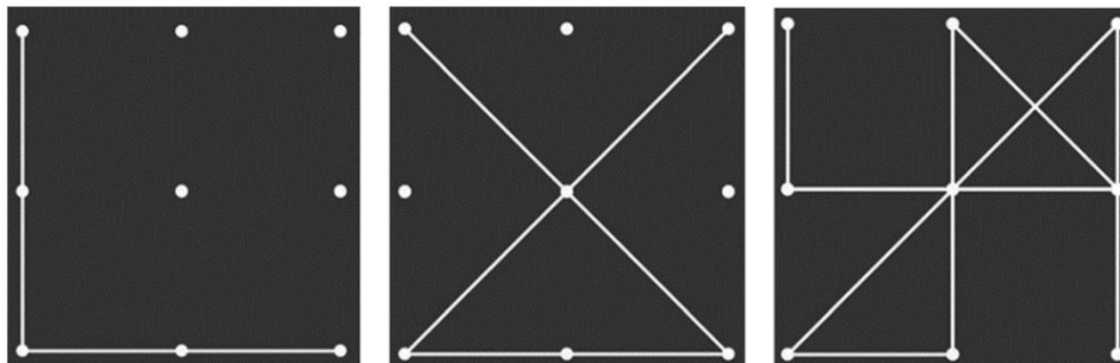
Ďalšia práca, v ktorej sa **Wolf** (2017), autor zo Slovenska, tiež zaoberá identifikáciou emocionálnych stavov na základe charakteristík získaných pri práci s počítačom, konkrétne pri práci s klávesnicou a myšou počítača. Zrealizoval niekoľko experimentov v rôznych prostrediach, ktoré mali zaznamenávať interakcie s klávesnicou a myšou a ukladať ich do logovacieho súboru. Cieľom prvého experimentu bolo prepísať text, konkrétne tri správy, kde každá obsahovala 2 až 3 vety a prepísať ich bolo potrebné za stanovený časový limit. Druhý experiment pozostával z jednoduchej hry, kde sa na grafickom plátne vygenerovalo na náhodnej pozícii 30 (očíslovaných) kruhov, ktoré mali používatelia v správnom poradí poklikať. Ak tak urobili v čase limite, spustilo sa druhé kolo, pri ktorom všetky nezakliknuté kruhy menili svoju pozíciu každých 10 sekúnd. Tretí experiment sa odohrával vo vlastnom webovom prehliadači, ktorý obsahoval navyše panel s otázkou. Autor sa domnieval, že v používateľoch vyvolá emóciu nudy, ak prehliadač pri načítavaní stránok pridá 2 sekundy oneskorenia. Otázky v tomto experimente sa týkali väčšinou vyhľadávania časti textov na internete. Štvrtý experiment bol neriadený a program na pozadí zaznamenával, čo robia, v akom okne sa pohybujú a snímali ich interakcie s klávesnicou a myšou počítača. Po všetkých experimentoch mali používatelia ohodnotiť svoj emocionálny stav zakliknutím jedného z troch stavov, a to buď pozitívny, neutrálny a negatívny. Experimentov sa zúčastnilo 22 používateľov. Z nespracovaných logovacích súborov neskôr pre myš získal 46 charakteristík a pre klávesnicu 21. Výsledky z experimentov autor vyhodnocoval pomocou jazyka R v aplikácii R Studio a tvoril modely strojového učenia

prostredníctvom algoritmov rozhodovacích stromov (*C4.5*, *C5.0* a *Random Forest*). Pre prvý experiment (pri prepise textu na klávesnici) dosiahol natrénovaním modelu rozhodovacieho stromu algoritmom *C4.5* najvyššiu presnosť 52,54%. Natrénovaný model, tým istým algoritmom pre druhý experiment (hra zameraná na prácu s myšou počítača) dosahoval presnosť až 70% a najnižšiu presnosť algoritmom *Random Forest* len 44,44%. Vyhodnotením výsledkov z tretieho experimentu (práca vo webovom prehliadači) skončila presnosťou modelu algoritmu *C4.5* najvyššou hodnotou 68,85%. Vo štvrtom (neriadenom) experimente dokázal pomocou klávesnice určiť emóciu s presnosťou 53,12% použitím algoritmu *Random Forest*.

Okrem klasifikácie a rozpoznávania emócií sa dá pomocou dynamiky stlačenia klávesov na klávesnici aj autentifikovať používateľa. Tejto téme sa venoval aj **Elliot** (2019) s jeho kolektívom a vytvoril jednoduchý program v programovacom jazyku Python, ktorý reagoval na stlačenie a uvoľnenie klávesov a pomocou toho dokázal vypočítať čas držania stlačeného klávesu, čas medzi stlačením prvého do stlačenia druhého klávesu a čas od pustení prvého klávesu do stlačenia druhého klávesu. Všetky tieto charakteristiky, pre každého z používateľov, uložil do CSV súboru. Cieľom úloh v tomto programe bolo 20-krát prepísať: reťazec desiatich znakov (označené ako silné heslo), jednu krátku vetu a dve krátke vety. Úlohu sa zúčastnilo 23 dobrovoľníkov. Po zozbieraní dát autor práce rozdelil dáta náhodne v pomere 80:20 na tréningové a testovacie. Na natrénovanie modelov autor použil 4 algoritmy strojového učenia: rozhodovacie stromy, *Random Forest*, *Support Vector Machine* (SVM) a *neurónové siete*. Výsledné modely otestoval na testovacích dátach. Algoritmus *Random Forest* dosahoval presnosť modelu klasifikácie 68% pri písaní silného hesla, 99,8% pri písaní jednej vety a 100% pri písaní dvoch viet. Autor ďalej uvádza, že pre *neurónovú sieť* bola presnosť pri písaní silného hesla len 40%, pre *rozhodovací strom* 41% a pre SVM len 35%. Všetky algoritmy dosahovali pomerne vysokú presnosť pri klasifikácii používateľa na základe písania dvoch viet.

Všetky predošlé analýzy prác sa týkali identifikácie emócií na základe správania sa používateľa pri práci s klávesnicou a myšou počítača. Existujú však aj práce, ktoré hľadajú závislosti medzi emóciami a správaním sa pri práci so smartfónom. Prvou prácou, ktorá využíva metódu dynamiky dotyku pri používaní dotykového zariadenia, je práca od **Balducciho** (2020) a jeho kolektívu, ktorý vyvinul aplikáciu s názvom *The EmoTouch lock app*, pre systém Android, ktorá pozostáva z jednej úlohy s tromi

rôznymi náročnosťami (nízka, stredná, ťažká). Cieľom tejto úlohy bolo v každej kategórii náročnosti 10-krát vytvoriť ťahaním prsta po obrazovke vopred zobrazený vzor na grafickom plátne. Podmienkou bolo, aby bol vzor vytvorený ťahaním prsta jedným ťahom. Príklady takýchto vzorov sú zobrazené na Obrázku 5.



Obrázok 5 Príklady vzorov s rôznou náročnosťou⁵

Tohto experimentu sa zúčastnilo 115 študentov a ešte pred touto úlohou bolo potrebné vyplniť dotazník DASS (Depression, Anxiety and Stress Scales), ktorý obsahuje 42 otázok (14 otázok pre každú emóciu: depresiu, úzkosť a stres). Na základe odpovedí sa vypočítala hodnota depresie, úzkosti a stresu pre používateľa. Na druhej strane, autor urobil okrem tejto formy dotazníka aj sebahodnotenie po každej úlohe, ktoré obsahovalo len 7 otázok, v ktorých sa pýtal na ruku, v ktorej držali smartfón, prstom, ktorým absolvovali úlohu a podobne a posledné 4 otázky sa týkali ohodnotenia vnímanej emócie pri danej úlohe. Súradnice (x a y) boli zaznamenávané počas toho, ako používateľ ťahá prstom po obrazovke. Po zozbieraní týchto dát, bolo z každej úlohy extrahovaných až 56 charakteristík. Medzi najdôležitejšie autor uviedol odchýlku čiary, smer potiahnutia, vzdialenosť a rýchlosť pohybu. Pri rozdelení dát na tréningové a testovacie využil autor metódu krížovej validácie, čím zaistil, že následnú klasifikáciu nebude ovplyvňovať nezávislé rozdelenie dát. Následne urobil klasifikáciu emócií na základe týchto dát pomocou algoritmov strojového učenia. Porovnal dva algoritmy: *rozhodovací strom (C4.5)* a *Random Forest*. *Random Forest* klasifikoval presnejšie ako *C4.5* až o 9% pre všetky kategórie emócií, v prípade stresu a úzkosti dosiahol presnosť klasifikácie 73,4% a depresiu zaklasifikoval s 74,1% presnosťou.

⁵ Zdroj Obrázka 5: <https://link.springer.com/article/10.1007%2Fs11042-020-09146-4>

Ďalšia práca, ktorá nielen klasifikuje, ale aj predikuje emocionálny stav používateľa podľa vzorov správania sa na mobilnom dotykovom zariadení používa k identifikácii emócií okrem softvérovej klávesnice aj senzor akcelerometer. Autor tejto práce, **Shapsough** (2016) s jeho kolektívom, vytvoril softvérovú klávesnicu, ktorá nie je viazaná na žiadnu aplikáciu a môže byť použitá ako textový vstup k akejkoľvek aplikácii. Okrem toho, že zbiera na pozadí dáta z klávesnice, keď na nej používateľ píše, zbiera dáta aj zo senzora akcelerometra, ktorý dokáže pomocou súradníc (x, y a z) zistiť v akej polohe sa smartfón nachádza. V prvej fáze, ešte pred zostavením predikčného modelu, klávesnica funguje tak, že ak používateľ píše na tejto klávesnici je vyzvaný, aby uviedol svoj aktuálny emocionálny stav (zakliknutím jedného z týchto stavov: neutrálny, nahnevaný, šťastný, smutný). Primárnym vstupom do algoritmu strojového učenia sú vektory funkcií odvodených od správania sa používateľa pri písaní. Každý vektor pozostáva z priemerného zrýchlenia, priemerného časového oneskorenia medzi stlačenými klávesmi, počtu medzier a označených emócií používateľov. Pre nájdenie najlepšieho klasifikátora, autor tejto práce vyskúšal algoritmy *Naive Bayes*, *rozhodovací strom (J48)*, *K-Nearest Neighbor*, *viacnásobnú lineárnu regresiu* a *Support Vector Machine (SVM)*. Najlepšie výsledky dosahoval rozhodovací strom (*J48*) s 90% presnosťou a najhoršie výsledky dosiahol algoritmom *SVM*. V druhej fáze, pomocou knižnice od Weka implementoval algoritmus *J48* v aplikácii. Akonáhle sa z písania používateľa zhromaždí dostatok dát, algoritmy strojového učenia dokážu predpovedať jeho emocionálny stav. Používateľ môže predikovaný stav klasifikátora kedykoľvek zmeniť (ak sa cíti inak, ako predikoval model) a tým upraviť jeho budúcu predikciu.

Nasledujúca práca, od **Ghosh**a (2017) a jeho kolektívu, je podobná predošlej práci, ktorú sme analyzovali, ale nevyužíva dáta zo senzorov, využíva len dáta z dynamiky stláčania klávesov na softvérovej klávesnici na mobilnom zariadení. Autor navrhol a implementoval pre systém Android detekčný systém emócií (*TapSense*), založený na písaní, použiteľný v ktorejkoľvek aplikácii, ktorý sa skladá z dvoch základných komponentov. Prvým je modul *TapLogger*, ktorý sleduje interakciu používateľa s klávesnicou. Skladá sa z jednoduchej QWERTY klávesnice, ktorú implementoval pomocou Android Input Method Editor (IME). Jediné, čo táto klávesnica uchováva, sú časové pečiatky každej udalosti ťuknutia na danú klávesu a tiež aj názov aplikácie, v ktorej sa používateľ práve nachádza. Druhý modul je *ESMLogger*, ktorý zbiera hodnotenia vlastných emócií používateľmi. Tento modul sa aktivuje iba

vtedy, ak používateľ napísal dostatočné množstvo textu v konkrétnej aplikácii (autorom počítané ako jedno sedenie) alebo, ak od posledného hodnotenia svojho emocionálneho stavu už uplynulo dostatok času. Používatelia mali na výber zo štyroch emocionálnych stavov (smutný, šťastný, vystresovaný a zrelaxovaný). Po ohodnotení boli všetky dáta poslané na server. Na strane servera boli dáta očistené od neúplných dát. Celkovo sa experimentu zúčastnilo 30 študentov, z toho len 22 z nich experiment dokončili. Experiment trval 3 týždne a od používateľov bolo požadované, aby klávesnicu tejto aplikácie nastavili na predvolenú klávesnicu pre daný smartfón. Po zozbieraní všetkých dát, vytvoril autor niekoľko charakteristík, ako napríklad: rýchlosť písania, interval medzi ťuknutiami na klávesy, počet špeciálnych znakov za sedenie, dĺžka sedenia, dĺžka napísaného textu a podiel mazacích klávesov. Nad týmto dátovým súborom autor vytvoril prispôbené modely na klasifikáciu emócií pre každého používateľa, pomocou algoritmu strojového učenia *Random Forest* implementovaného vo Weke. Testoval aj iné algoritmy strojového učenia (*logistická regresia*, *Support Vector Machine*), ale spomínaný *Random Forest* dosahoval najvyššiu presnosť. Priemerná presnosť klasifikácie pre všetkých používateľov dosahovala 73% a maximálna presnosť pre jedného z používateľov bola až 94%.

2 CIELE DIPLOMOVEJ PRÁCE

Hlavným cieľom tejto diplomovej práce je využiť prvky umelej inteligencie pri klasifikácii emocionálneho stavu používateľa na dátach tvorených behaviorálnymi charakteristikami, získaných z vlastnej desktopovej a mobilnej aplikácie. Súčasťou dát bude aj hodnotenie vlastného emocionálneho stavu používateľov na dimenziách valencie a vzrušenia v Russelovom kruhom modeli emócií.

PODCIELE ALEBO ČIASTKOVÉ CIELE:

- analýza najpoužívanějších metód strojového učenia pri klasifikačných úlohách,
- analýza prác autorov venujúcich sa klasifikácii emócií zo vzorov správania sa, získaných prostredníctvom klávesnice, myši a smartfónu,
- tvorba aplikácií zberajúcich behaviorálne charakteristiky pre počítače a smartfóny,
- zozbieranie dostatočného počtu dát pre ich ďalšiu analýzu,
- analýza dát a aplikovanie metód strojového učenia na tieto dáta,
- zhodnotenie výsledkov klasifikácie.

3 TVORBA APLIKÁCIÍ A ZBER DÁT

V našej predošlej práci (Držík, 2020) sme vytvorili prototyp desktopovej aplikácie Emotnizer, ktorá zberala dáta behaviorálnych charakteristík používateľa a ukladala ich lokálne do SQLite databázy v podobe súboru. Aplikácia pozostávala z jednej ucelenej aktivity, ktorá obsahovala tvorbu profilu a prepis vopred definovaného textu. V tejto aktivite sme naraz zberali dáta behaviorálnych charakteristík používateľa pri používaní klávesnice aj myši.

Následná analýza dát ukázala, že charakteristiky získané z klávesnice nemožno kombinovať s charakteristikami získanými z myši počítača. Preto sme sa rozhodli, že by sme v tejto súčasnej práci mohli vytvoriť viacero aktivít, nazývajme ich experimenty, ktoré budú zvlášť zberať dáta charakteristík správania pri používaní klávesnice a zvlášť pri používaní myši počítača. Vo verzii pre smartfóny by sme zberali dáta pri pohybe prsta používateľa po dotykovom displeji.

V nasledujúcich podkapitolách rozoberieme úpravu už existujúceho kódu v desktopovej aplikácii, tvorbu mobilnej aplikácie, databázový návrh pre uloženie získaných dát na serveri a prepojenie aplikácií s databázou pomocou PHP skriptov.

3.1 ÚPRAVA EXISTUJÚCEHO KÓDU

V našej aplikácii sme ponechali implementáciu knižnice JNativeHook, ktorá dokáže v jazyku Java odchytiť všetky udalosti z klávesnice a myši počítača pomocou natívnych metód. Avšak niektoré charakteristiky, ktoré sme pomocou týchto metód získavali, sme museli upraviť alebo úplne odstrániť. Napríklad pre myš počítača sme mali charakteristiky: *frekvencia klikania myšou*, *dobu držania tlačidla myši*, *rolovanie kolieskom myši*, *rýchlosť pohybu kurzora myši* a *rýchlosť pohybu kurzora myši pri kliknutí*. Úplne sme odstránili rolovanie kolieskom myši, pretože v minulej práci len málo používateľov použilo koliesko myši, čo prinieslo neúplné dáta. A z toho istého dôvodu sme odstránili aj charakteristiku rýchlosť pohybu kurzora myši pri kliknutí. Pridali sme charakteristiku **pauza medzi kliknutiami** pre myš, čo je vlastne doba medzi posledným uvoľnením tlačidla myši až po následné stlačenie tlačidla, pričom táto doba nesmie presiahnuť interval dĺžky spánku vlákna, čo je 5000 milisekúnd.

Frekvenciu klikania myšou, ale aj **frekvenciu stláčania klávesov** na klávesnici sme museli upraviť, pretože pri predošlom výpočte frekvencie sme iba počet jednotlivých stlačení vydělili dobou spánku (v sekundách to je 5 sekúnd) a dostali

sme tak hodnotu stlačení klávesov/tlačidiel za sekundu. To nebol celkom správny prístup, pretože výsledok môže v niektorých prípadoch nadobúdať zvláštne hodnoty. Pre príklad na Obrázku 6, prvú klávesu stlačíme v behu vlákna v 500 milisekundách a uvoľníme ju v 1500 ms. Druhú klávesu stlačíme v 3000 ms a uvoľníme v 4600 ms. Podľa starej metódy výpočtu by výsledok vyzeral nasledovne:

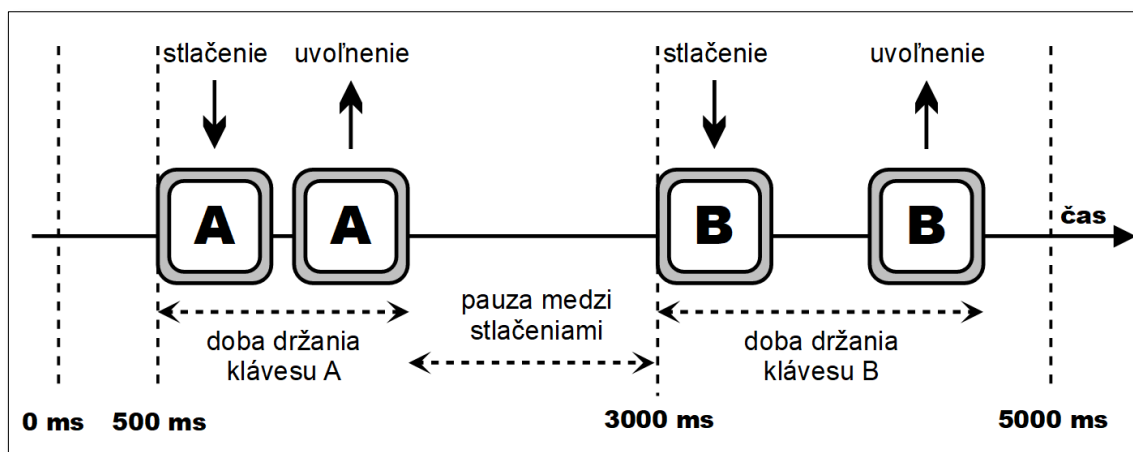
$$y_{old} = \frac{\text{počet stlačení}}{\text{doba spánku vlákna (v sekundách)}} = \frac{2}{5} = 0,4.$$

Keby sme ešte do výsledku zahrnuli aj čas stlačenia, výsledná frekvencia by bola oveľa presnejšia:

$$y_{new} = \frac{\text{počet stlačení}}{t_{max} - t_{min} \text{ (v sekundách)}} = \frac{2}{3 - 0,5} = 0,8,$$

kde t_{max} je čas posledného stlačenia a t_{min} je čas prvého stlačenia. V prípade, že by počas doby spánku vlákna nastalo len jedno kliknutie, výsledná frekvencia klikania by bola 1 kliknutie za sekundu. Podľa starého výpočtu by to bolo 0,2 kliknutí za sekundu.

Na Obrázku 6 sú zobrazené aj ďalšie charakteristiky pre klávesnicu, okrem spomínanej *frekvencie stláčania klávesov* je tu aj *doba držania klávesu* a *pauza medzi stlačeními klávesov*.



Obrázok 6 Doba držania klávesu a pauza medzi stlačeními

Všetky ostatné zdrojové kódy ako napríklad pripojenia k databáze SQLite, tvorba profilu v aplikácii, ale aj samotná aktivita, pri ktorej sa zbierali dáta, sme zmazali. V ďalšej podkapitole si rozoberieme tvorbu profilu na vzdialenom serveri.

3.2 WEBOVÝ SERVER

Našou prvotnou myšlienkou bolo, aby sme vedeli prideliť dáta získané z desktopovej a mobilnej aplikácie jednotlivým používateľom. To je možné len pomocou toho, že sa používatelia do aplikácie prihlásia pod svojím účtom, respektíve pod svojím emailom a heslom. Preto sme sa rozhodli vytvoriť registračný formulár na webe s ďalšími dotazníkovými otázkami. Hľadali sme voľne dostupný webový

hosting, ktorý by obsahoval aj MySQL databázu pre ukladanie dát a podporu jazyka PHP pre komunikáciu aplikácií s databázou. Podarilo sa nám nájsť **WebZdarma.cz**, ktorý toto všetko poskytuje. Následne sme si zaregistrovali doménu **emotnizer.wz.sk** a vytvorili sme prvú tabuľku *user* v našej databáze (podrobnejšie v ďalšej podkapitole). Následne sme vytvorili registračný formulár v jazyku HTML5 a pomocou PHP skriptu (zobrazený na Obrázku 7) sme ukladali registračné údaje do MySQL databázy.

```
<?php
require "conn.php";
$email = str_replace(' ', '', $_POST['email']);
$password = mysqli_real_escape_string($conn, $_POST["password"]);
...ďalšie atribúty

$sql = "SELECT * FROM user WHERE BINARY email = '$email'";
$result = mysqli_query($conn, $sql);
$password = password_hash($password, PASSWORD_DEFAULT);
$sql = "INSERT INTO user (email, password, ...) VALUES ('$email', '$password', ...)";
if ($conn->query($sql) === TRUE) {
    echo "Registrácia prebehla úspešne!";
} else {
    echo "Registrácia prebehla NEÚSPEŠNE!";
}
$conn->close();
?>
```

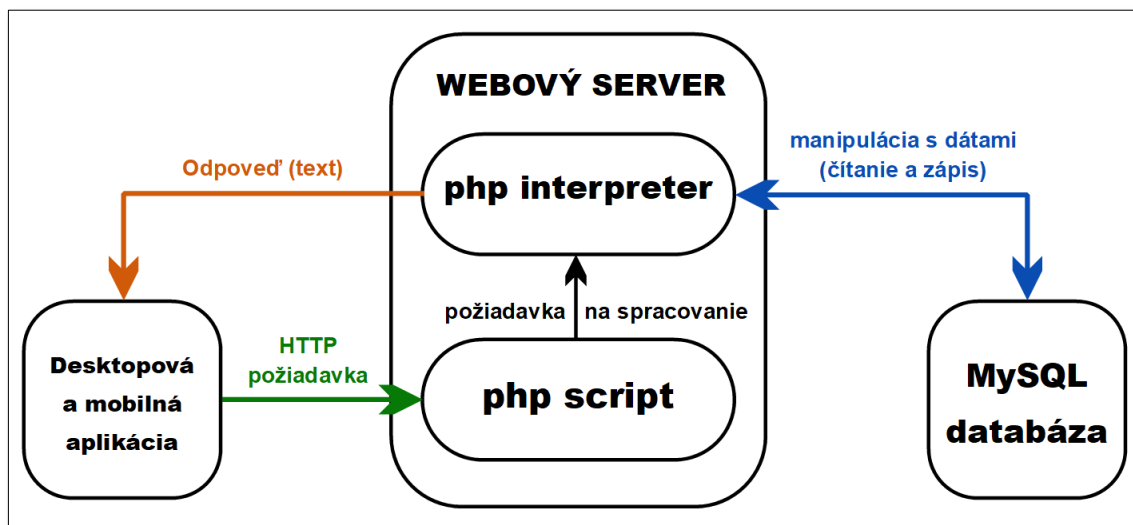
Obrázok 7 PHP skript – registrácia

Okrem prihlasovacích údajov (email a heslo) sme pridali aj zopár dotazníkových otázok ohľadom ich veku, pohlavia, dominantnej ruky (pravá/ľavá), národnosti, znalosti slovenského jazyka, emocionálnej citlivosti a vnímaní emócií, koľko času venujú písaniu textu na klávesnici a či ovládajú prstoklad.

Použili sme elegantné riešenie pre prácu s databázou v PHP. Pri všetkých skriptoch, ktoré sa potrebujú pripojiť k databáze sa používa ďalší PHP súbor (my sme si ho nazvali *conn.php*), ktorý obsahuje informácie o **názve databázy, prihlasovacom mene a hesle do databázy** a obsahuje aj **názov servera**.

Všetky PHP skripty, ktoré slúžia na registráciu, prihlásenie sa do aplikácií, ale aj uloženie dát z experimentov sú uložené na webovom serveri. Webový server obsahuje aj PHP interpret, pomocou ktorého sa prekladá kód našich PHP skriptov. Ak používateľ v desktopovej alebo v mobilnej aplikácii spustí HTTP požiadavku na webový server, server rozpozna PHP príponu súboru, interpret preloží PHP súbor a podľa toho, či je to skript na čítanie z databázy alebo na zapisovanie do databázy, prípadne ich kombinácia, táto akcia sa vykoná a spätne sa používateľovi v aplikácii

zobrazí odpoveď v podobe textu, ktorý je vypisovaný cez PHP príkaz `echo`. Táto komunikácia medzi databázou a aplikáciami je ilustrovaná na Obrázku 8.



Obrázok 8 Komunikácia medzi MySQL, PHP a aplikáciami

V desktopovej aj mobilnej aplikácii sme sa snažili použiť rovnaký prístup na zápis dát do databázy, ale aj na čítanie dát z nej. Použili sme abstraktnú triedu **HttpURLConnection** (časti kódu sme implementovali z dokumentácie Google, 2021), kde sme na začiatku museli definovať URL adresu (zobrazené na Obrázku 9). Potom sme vytvorili spojenie, nastavili sme metódu posielania (POST). Vytvorili sme výstupný prúd a pomocou *BufferedWritera* pošleme dáta na URL adresu skriptu.

```

String email = params[1];
String password = params[2];
URL url = new URL(https://emotnizer.wz.sk/login.php);
HttpsURLConnection httpURLConnection = (HttpsURLConnection) url.openConnection();
httpURLConnection.setRequestMethod("POST");
httpURLConnection.setDoOutput(true);
httpURLConnection.setDoInput(true);
OutputStream outputStream = httpURLConnection.getOutputStream();
BufferedWriter bufferedWriter = new BufferedWriter(
    new OutputStreamWriter(outputStream, "UTF-8"));

```

Obrázok 9 Posielanie dát, časť 1

Dáta pošleme v jednom textovom reťazci, kde zabalíme email a HASH hesla v kódovaní UTF-8 (postup je ilustrovaný na Obrázku 10). Po odoslaní tohto textového reťazca očakávame od webového servera nejakú odpoveď, tiež v tvare textu, ktorú nám server vráti pomocou PHP príkazu `echo`. Aby sme mohli takúto odpoveď prijať, musíme si vytvoriť vstupný prúd a pomocou *BufferedReadera* si načítame výslednú odpoveď a všetky spojenia uzavrieme.

```

String post_data = URLEncoder.encode("email","UTF-8")+"="+ URLEncoder.encode(email,"UTF-8")
    + "&" + URLEncoder.encode("password","UTF-8")+"="+ URLEncoder.encode(password,"UTF-8");
bufferedWriter.write(post_data);
bufferedWriter.flush();
bufferedWriter.close();
outputStream.close();
InputStream inputStream = httpURLConnection.getInputStream();
BufferedReader bufferedReader = new BufferedReader(
    new InputStreamReader(inputStream, "iso-8859-1"));

String result = "";
String line = "";
while ((line = bufferedReader.readLine()) != null) {
    result += line;
}
bufferedReader.close();
inputStream.close();
httpURLConnection.disconnect();
return result;

```

Obrázok 10 Posielanie dát, časť 2

V tomto prípade sme si na ukážku vybrali práve login.php skript (zobrazený na Obrázku 11), pretože obsahuje len 2 parametre, pri iných skriptoch je postup rovnaký, len parametrov, ktoré treba poslať, je oveľa viac. Ako odpoveď od servera môžeme podľa nášho kódu dostať len 2 možné odpovede. Ak sa daný používateľ s takýmto emailom a heslom našiel, tak odpoveď bude „yes“ a id toho používateľa, ale ak sa v databáze nenašiel, tak sa pošle „no“. Kód, pomocou ktorého posielame HTTP požiadavku je v jazyku Java, takže bude fungovať aj pre desktopovú, aj pre mobilnú aplikáciu na Android.

```

<?php
require "conn.php";
$user_email = $_POST['email'];
$user_pass = mysqli_real_escape_string($conn, $_POST["password"]);

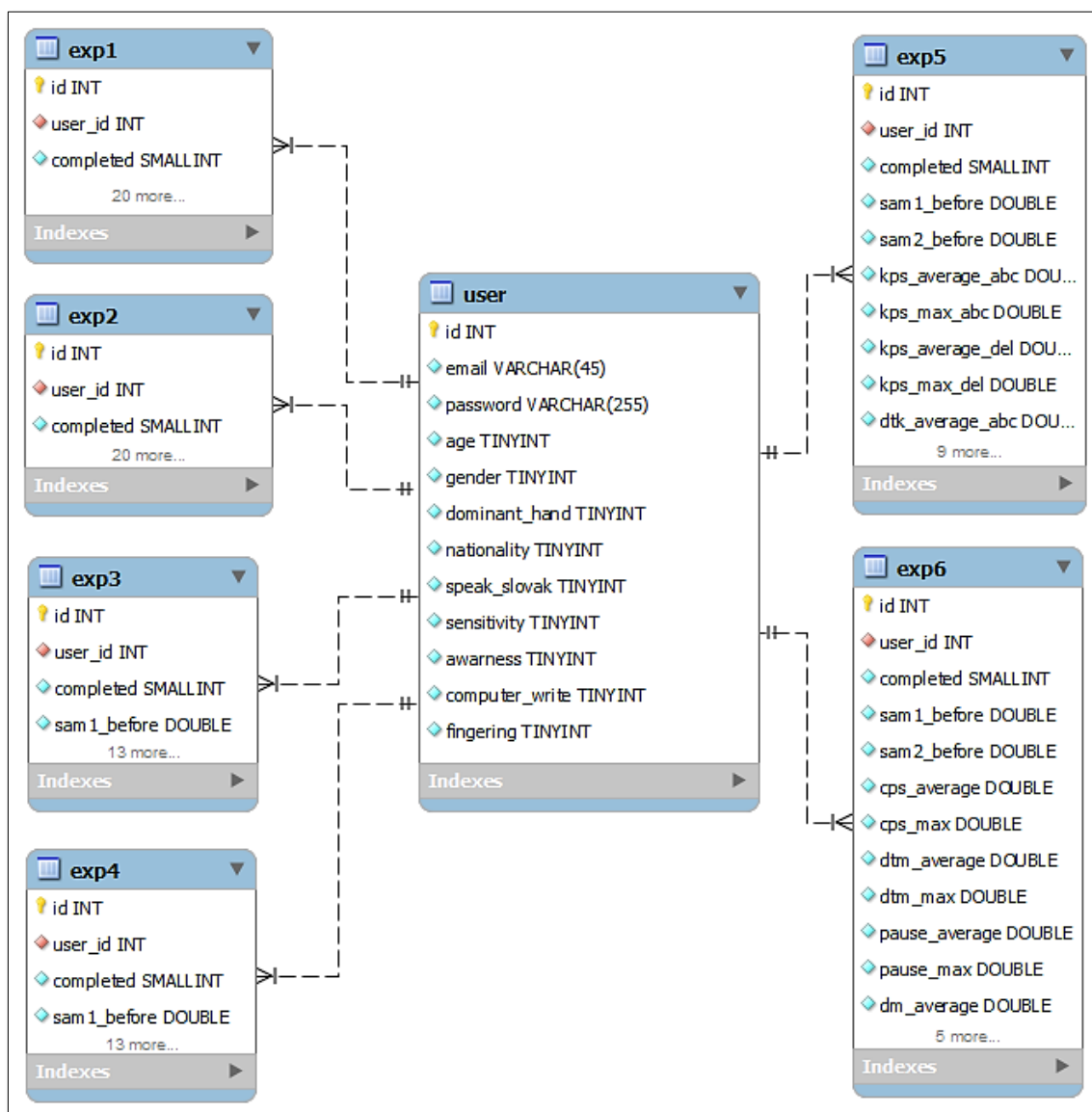
$sql = "SELECT * FROM user WHERE BINARY email = '$user_email'";
$result = mysqli_query($conn, $sql);
if ($result->num_rows > 0) {
    while ($row = $result->fetch_assoc()) {
        if (password_verify($user_pass, $row["password"])) {
            echo "yes;" . $row["id"];
        } else {
            echo "no";
        }
    }
} else echo "no";
$conn->close();
?>

```

Obrázok 11 PHP skript – prihlásenie

3.3 DATABÁZA

Návrh entitno-relačného diagramu (ERD) našej databázy sme tvorili v programe MySQL Workbench. Diagram pozostáva zo 7 entít (tabuliek), kde základnou tabuľkou je tabuľka *user*, ktorá obsahuje atribúty z registrácie z predchádzajúcej podkapitoly. Ďalších 6 entít prislúcha jednotlivým experimentom, prvé 4 sú pre experimenty desktop aplikácie a ďalšie 2 sú pre mobilnú aplikáciu. Keďže každá z tabuliek obsahuje priveľa atribútov, nie je možné všetky vizualizovať (zobrazené na Obrázku 12).



Obrázok 12 ERD diagram

Všetky ďalšie entity obsahujú niektoré rovnaké atribúty, menovite: id používateľa (*user_id*), koľkokrát bol experiment absolvovaný daným používateľom (*completed*), sebahodnotenie pred aj po experimente na dimenziách valencie a vzrušenia

v desatinnom tvare na intervale $<-1,1>$ (*sam1*, *sam2*), úspešnosť dokončenia experimentu v percentách (*accuracy*) a čas trvania experimentu v sekundách (*time*).

Experimenty 1 a 2 sú v desktopovej aplikácii zamerané na písanie textu pomocou **klávesnice**. Atribúty entít týchto experimentov sú rovnaké a obsahujú okrem vyššie spomínaných atribútov aj priemernú a maximálnu frekvenciu stláčania klávesov za sekundu pre všetky druhy klávesov (*kps*), priemernú a maximálnu dobu držania klávesov v milisekundách pre všetky druhy klávesov (*dtk*) a priemernú a maximálnu pauzu medzi stlačeniami klávesov (*pause*). Pre experiment 5 v mobilnej aplikácii sú atribúty rovnaké ako pre experiment 1 a 2, ale len s tým rozdielom, že tu je o jeden druh klávesov menej (bližšie vysvetlené v podkapitole *Tvorba experimentov v mobilnej aplikácii*) a prepis textu prebieha prostredníctvom nami vytvorenej virtuálnej klávesnice.

Experimenty 3 a 4 sú v desktopovej aplikácii zamerané na klikanie pomocou **myši** počítača a experiment 6 je v mobilnej aplikácii zameraný na ťukanie **prsta** po grafickej ploche s tlačidlami. Všetky atribúty týchto troch experimentov majú rovnaké označenie pre jednoduchosť a prehľadnosť danej databázy. Okrem vyššie spomínaných atribútov obsahujú aj tieto atribúty: priemerná a maximálna frekvencia klikania/ťukania za sekundu (*cps*), priemerná a maximálna doba držania tlačidla/prsta v milisekundách (*dtm*), priemerná a maximálna pauza medzi kliknutiami/ťuknutiami v milisekundách (*pause*) a priemerná a maximálna vzdialenosť v pixeloch, ktorú prejde kurzor myši alebo vzdialenosť medzi dvoma ťuknutiami v mobilnej aplikácii (*dm*).

3.4 SEBAHODNOTENIE V APLIKÁCIÁCH

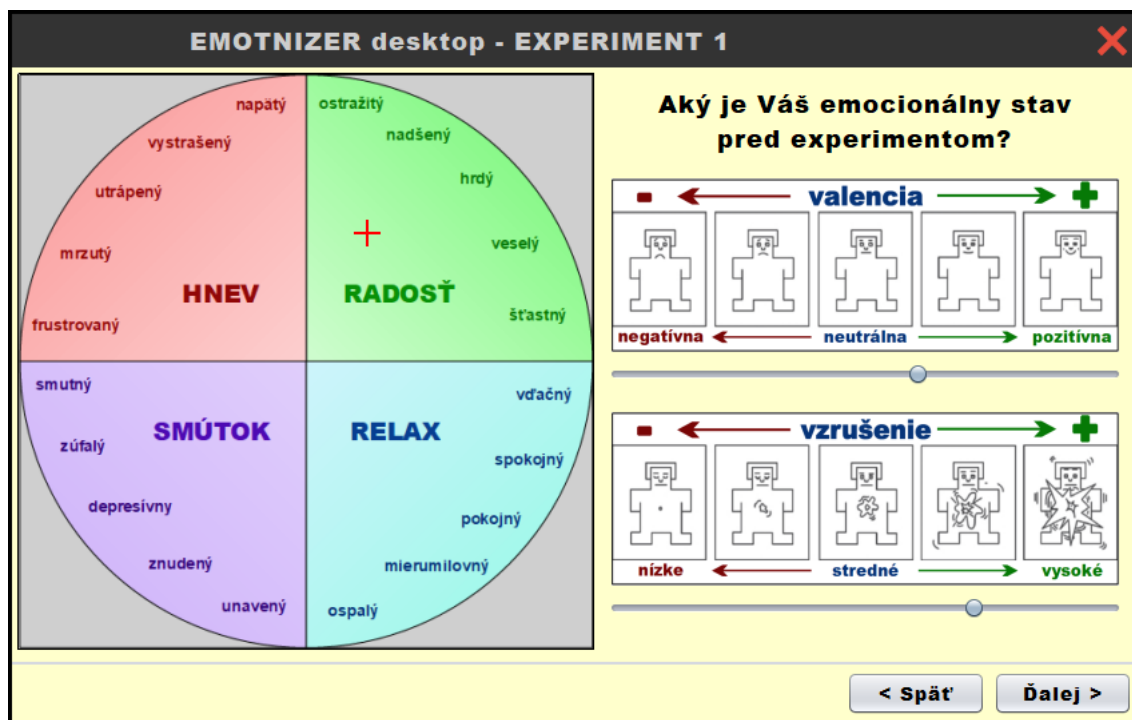
V predošlej práci sme riešili sebahodnotenie na dvoch základných dimenziách valencie a vzrušenia v Russelovom kruhovom modeli emócií. Tento prístup sa nám osvedčil, preto sme ho ponechali, ale používatelia pri hodnotení svojho emocionálneho stavu mali k dispozícii len 5 bodové ohodnotenie na obidvoch dimenziách a navyše, priamo v aplikácii nevideli, akú emóciu to v Russelovom kruhovom modeli predstavuje. Preto sme sa rozhodli viacnásobne rozšíriť hodnotiacu škálu pre obe dimenzie, čo bude mať za následok, že používatelia budú môcť presnejšie vyjadriť svoj emocionálny stav a následne sme tieto hodnoty premietli v Russelovom kruhovom modeli emócií priamo v aplikáciách (zobrazené na Obrázkoch 13 a 14).

Pôvodná hodnotiacia škála obsahovala len 5 bodov, zatiaľ čo novú sme 40-krát násobne rozšírili. Vo výsledku obe dimenzie môžu nadobúdať hodnoty od -1 po 1

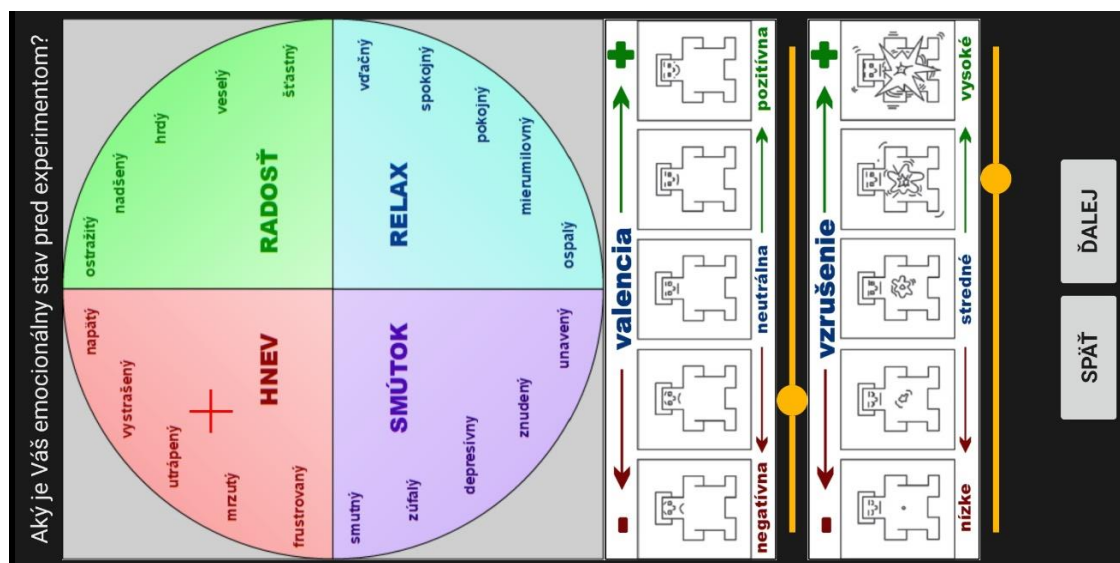
s krokom 0,01. Jedinou podmienkou bolo, aby vyznačená emócia patrila kruhu, pretože aj samotný Russel (1980) tvrdí, že všetky emócie sa dajú vpísať len do tohto kruhového modelu a nie mimo neho. Preto sme v aplikáciách využili jednoduchú rovnicu kružnice, ktorou sa takéto stavy dali jednoducho rozlíšiť.

$$valencia^2 + vzrušenie^2 \leq 1$$

Rovnica 1 Rovnica kružnice



Obrázok 13 Sebahodnotenie v aplikácii Emotnizer desktop



Obrázok 14 Sebahodnotenie v aplikácii Emotnizer mobile

Rozhodli sme sa, že používatelia budú hodnotiť svoj emocionálny stav pred aj po každom experimente a to prostredníctvom bežcov (sliderov) pre valenciu aj pre vzrušenie. Vypracovali sme k tomu aj podrobný PDF návod, ktorý je k dispozícii v aplikáciách. Russelov kruhový model sme trochu farebne prerobili, aby bol výraznejší a celý model je rozdelený do štyroch oblastí:

- **hnev** (negatívna valencia a vysoké vzrušenie),
- **smútok** (negatívna valencia a nízke vzrušenie),
- **radosť** (pozitívna valencia a vysoké vzrušenie),
- **relax** (pozitívna valencia a nízke vzrušenie).

3.5 TVORBA EXPERIMENTOV V DESKTOPOVEJ APLIKÁCII

V našej práci sme sa rozhodli pokračovať vo vyvíjaní desktopovej aplikácie v jazyku Java, ktorý je stále veľmi používaný v tvorbe desktopových aplikácií. Aplikáciu sme vyvíjali vo vývojovom prostredí NetBeans IDE 8.2 a používali sme verziu JDK 8 (Java Development Kit). Pre tvorbu používateľského rozhrania, ako sú napríklad okná, dialógové okná a tlačidlá sme použili vstavanú knižnicu Swing.

V predchádzajúcich podkapitolách sme urobili centrálnu registráciu používateľov, aby sme ich neskôr vedeli zaradiť dátam z jednotlivých experimentov. Práve preto sa musia používatelia pri vstupe do aplikácie prihlásiť pod svojím emailom a heslom. Po úspešnom prihlásení sa používateľovi zobrazí jednoduché okno, ktoré obsahuje niekoľko kariet, najdôležitejšie sú karta *Domov* a karta *Experimenty*. Na karte *Domov* používateľ vidí pod akým emailom je prihlásený a má možnosť si prejsť návody v PDF forme pre tvorbu experimentov a porozumieť im, čo sa on neho bude neskôr požadovať. Takisto je v aplikácii prístupný aj návod pre hodnotenie svojho emocionálneho stavu, pretože hodnotiť svoj stav musia používatelia pred aj po absolvovaní experimentu a pre neskoršiu klasifikáciu je veľmi potrebné, aby používatelia svoj emocionálny stav ohodnotili čo najsprávnejšie.

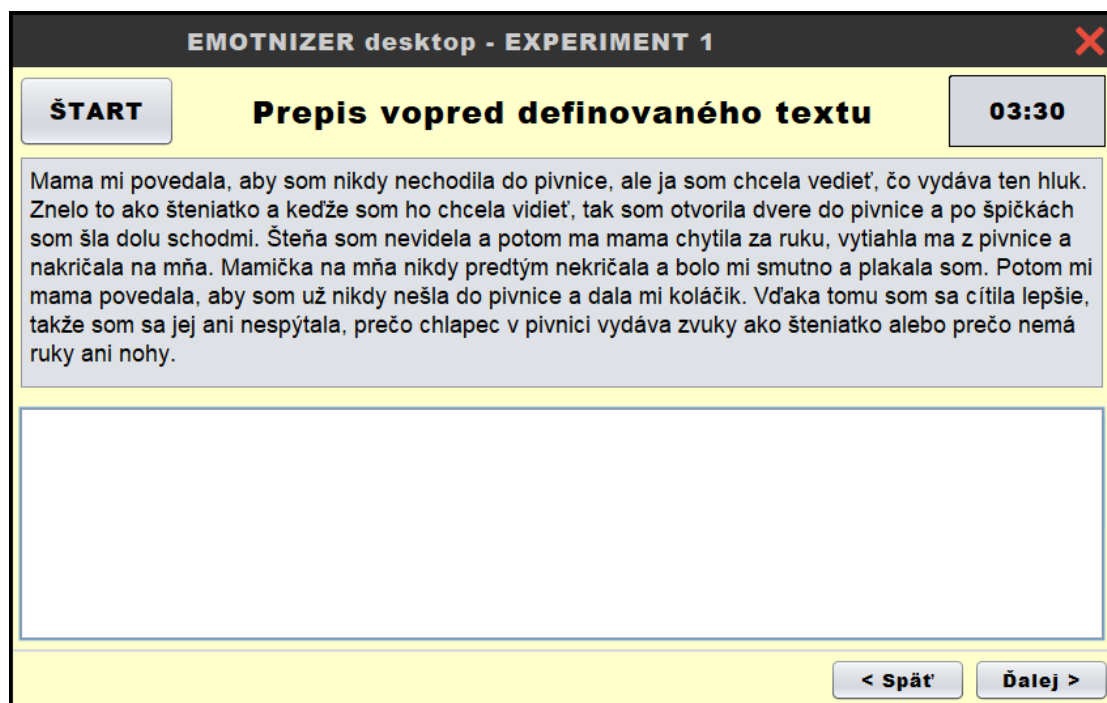
Na karte *Experimenty* sme vytvorili 4 experimenty a ako sme už v predchádzajúcich podkapitolách spomínali, dva experimenty sú zamerané na prácu s klávesnicou počítača a dva experimenty sú zamerané na prácu s myšou počítača. Po spustení experimentu sa otvorí dialógové okno, ktoré je zarovnané na stred obrazovky a je znemožnené takýmto oknom pohybovať. V ďalších podkapitolách si podrobne rozoberieme tvorbu a cieľ experimentov v aplikácii Emotnizer desktop.

3.5.1 Experiment 1 – Prepis vopred definovaného textu

Cieľom experimentu je prepísať text, pomocou klávesnice, z horného textového poľa do spodného poľa vo vopred stanovenom časovom limite (3 minúty a 30 sekúnd), pričom sú používatelia upozornení, že text musí byť prepísaný čo najpresnejšie, pretože textové polia sa neskôr porovnávajú. Textové polia sa porovnávajú na základe **Levenshteinovej vzdialenosti**, čo je vlastne počet znakov, v ktorých sa líšia dva textové reťazce. Čím je toto číslo väčšie, tým sú tieto dva porovnávané reťazce odlišnejšie, a naopak. Pre príklad slov PES a LES. Levenshteinova vzdialenosť medzi týmito slovami je 1, pretože najmenší počet úprav slova PES na LES je jeden znak. Pod úpravou sa myslí vloženie, zmazanie alebo nahradenie znaku. V tomto jednoduchom príklade sme znak nahradili. Takéto porovnávanie reťazcov sa zvyčajne robí pomocou matíc a aj my sme v aplikácii takéto matice využili (Jlordiales, 2014; Cuelogic.com, 2017; Nam, 2019). Text, ktorý mali používatelia prepísať obsahuje 612 znakov vrátane medzier. Presnosť prepísaného textu sme počítali na základe tohto vzorca:

$$\text{presnosť} = \frac{\text{počet znakov pôvodného textu} - \text{Levenshteinova vzdialenosť reťazcov}}{\text{počet znakov pôvodného textu}}$$

Vzorec 4 Presnosť prepísaného textu



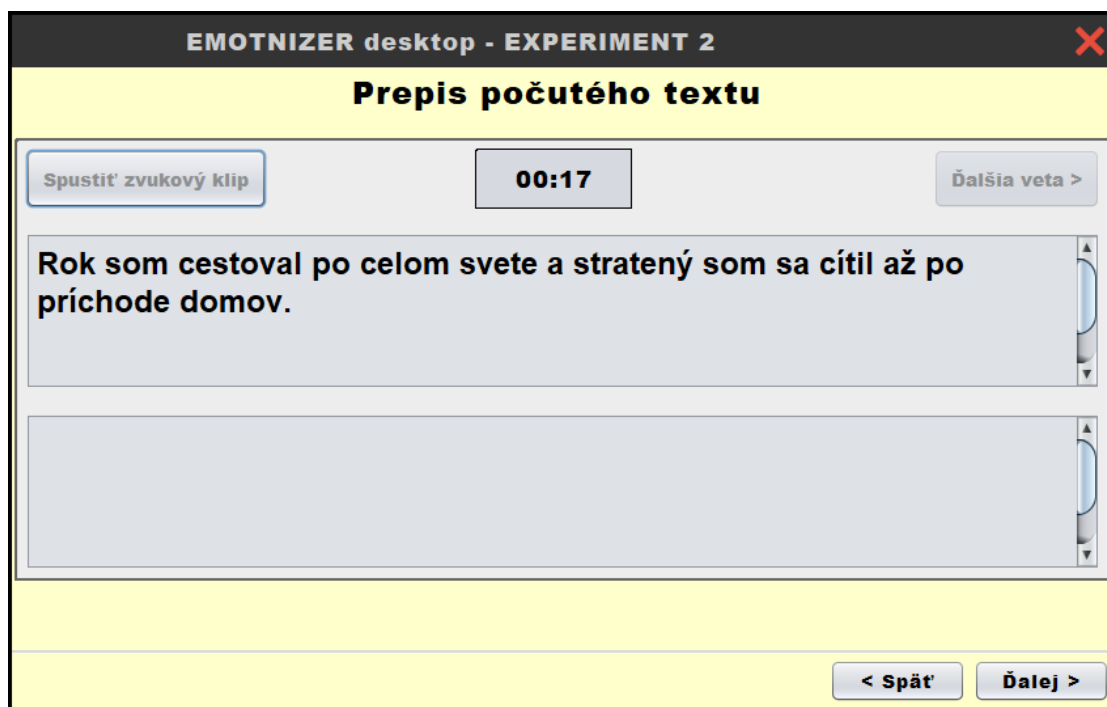
Obrázok 15 Experiment 1 v desktopovej aplikácii

Text, ktorý sme vybrali mal vyvolať v používateľoch nejakú negatívnu emóciu (od strachu, znechutenia až po napätosť). Celý text, ale aj grafické používateľské rozhranie Experimentu 1 je možné vidieť na Obrázku 15.

3.5.2 Experiment 2 – Prepis textu zvukovej nahrávky

Aj v druhom experimente bolo hlavným cieľom prepisovať text. Na začiatku používateľ stlačí tlačidlo *Spustiť zvukový klip* a aplikácia prehrá krátku zvukovú nahrávku danej vety, ktorú musí používateľ neskôr napísať. Hneď potom, ako zvuková nahrávka skončí, prehrá sa znova a tentokrát sa spolu s nahrávkou zobrazí aj daný text v hornom textovom okne. Úlohou používateľa je po skončení nahrávky a zmiznutí textu napísať text, ktorý si zapamätal, do spodného textového okna v určenom časovom limite, v závislosti od dĺžky danej prepisovanej vety. O blížiacom sa konci prepisovania je používateľ upozornený zvukovým signálom (alarmom) 3 sekundy pred koncom.

Celkovo tento experiment obsahoval až 5 zvukových nahrávok. Snažili sme sa vybrať predovšetkým známe vety, väčšina z nich sú ľudové múdrosti. Všetky vety dokopy obsahovali až 318 znakov (vrátane medzier) a čas na prepis všetkých viet bol dokopy 72 sekúnd. V porovnaní s prvým experimentom je to na počet znakov oveľa menej času, pretože samotné vety v používateľoch nevyvolajú nejaký druh emócie, ale nestihnúť prepísať ich, vyvolať emóciu môže. Experiment je zobrazený na Obrázku 16.



Obrázok 16 Experiment 2 v desktopovej aplikácii

Takisto aj v tomto experimente sme textové polia porovnávali a zisťovali sme presnosť v prepisovaní textu. Využívali sme Levenshteinovu vzdialenosť pre každú z pôvodných a prepísaných viet, postupne sme si počítali presnosť prepísaného textu, pomocou Vzorca 1 z predchádzajúcej podkapitoly, a na záver sme všetky presnosti spriemerovali. Aj v tomto experimente museli používatelia ohodnotiť svoj emocionálny stav pred a po absolvovaní experimentu.

Výstupné charakteristiky z Experimentov 1 a 2 sú v rovnakej štruktúrovanej podobe, v databáze majú podobu atribútov. Prehľadne sú uvedené aj s popisom v Tabuľke 1.

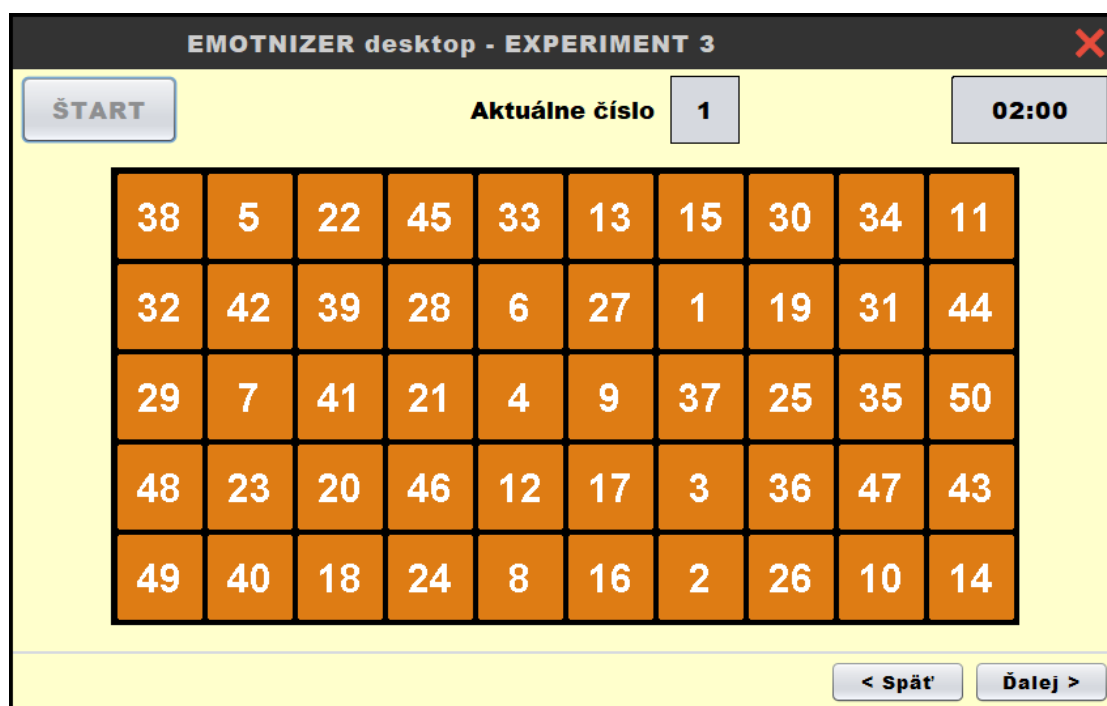
Tabuľka 1 Charakteristiky ukladané do databázy pre Experiment 1 a 2

Názov atribútu	Popis atribútu
COMPLETED	koľkokrát bol experiment dokončený daným používateľom
SAM1_BEFORE	hodnota valencie pred experimentom
SAM2_BEFORE	hodnota vzrušenia pred experimentom
KPS_AVERAGE_ABC	priemerná frekvencia stláčania bežných klávesov za sekundu
KPS_MAX_ABC	maximálna frekvencia stláčania bežných klávesov za sekundu
KPS_AVERAGE_CTRL	priemerná frekvencia stláčania funkčných klávesov za sekundu
KPS_MAX_CTRL	maximálna frekvencia stláčania funkčných klávesov za sekundu
KPS_AVERAGE_DEL	priemerná frekvencia stláčania mazacích klávesov za sekundu
KPS_MAX_DEL	maximálna frekvencia stláčania mazacích klávesov za sekundu
DTK_AVERAGE_ABC	priemerná dĺžka držania bežných klávesov v milisekundách
DTK_MAX_ABC	maximálna dĺžka držania bežných klávesov v milisekundách
DTK_AVERAGE_CTRL	priemerná dĺžka držania funkčných klávesov v milisekundách
DTK_MAX_CTRL	maximálna dĺžka držania funkčných klávesov v milisekundách
DTK_AVERAGE_DEL	priemerná dĺžka držania mazacích klávesov v milisekundách
DTK_MAX_DEL	maximálna dĺžka držania mazacích klávesov v milisekundách
PAUSE_AVERAGE	priemerná pauza medzi stlačeniami
PAUSE_MAX	maximálna pauza medzi stlačeniami
SAM1_AFTER	hodnota valencie po experimente
SAM2_AFTER	hodnota vzrušenia po experimente
ACCURACY	presnosť správnosti prepísaného textu
TIME	čas trvania experimentu (v sekundách)

3.5.3 Experiment 3 – V správnom poradí

V treťom experimente desktopovej aplikácie sme sa zamerali na charakteristiky, ktoré sa získavajú pri práci s myšou počítača. Cieľom tohto experimentu bolo kliknúť vzostupne na všetky zobrazené tlačidlá, ktoré boli označené číslom. Celkovo tento experiment obsahoval 50 tlačidiel a vždy hore sa zobrazovalo aktuálne číslo, ktoré používateľ musí stlačiť. Dôležité je tiež spomenúť, že tlačidlá boli rozmiestnené

náhodne a po stlačení konkrétneho tlačidla, ktoré bolo na rade sa danému tlačidlu zmení farba z oranžovej na zelenú. Pri tomto experimente sme sa inšpirovali prácou Pentela (2015), ktorého prácu sme opísali aj v našej minulej práci, kde použil 24 tlačidiel, na ktoré mali používatelia kliknúť v správnom poradí. My sme na rozdiel od neho pridali viac tlačidiel, čo môže v používateľovi po chvíli vyvolať emóciu nudy, ale keďže na tento experiment je vymedzený čas iba 2 minúty, v používateľoch to môže vyvolať nejakú negatívnu emóciu alebo dokonca aj strach. Našich 50 tlačidiel bolo rozmiestnených náhodne do tvaru obdĺžnika 10 x 5 tlačidiel. Úspešnosť dokončenia tohto experimentu sme merali ako podiel počtu už zakliknutých tlačidiel a počtu všetkých tlačidiel. Všetky charakteristiky získané týmto experimentom sú zobrazené v Tabuľke 2.

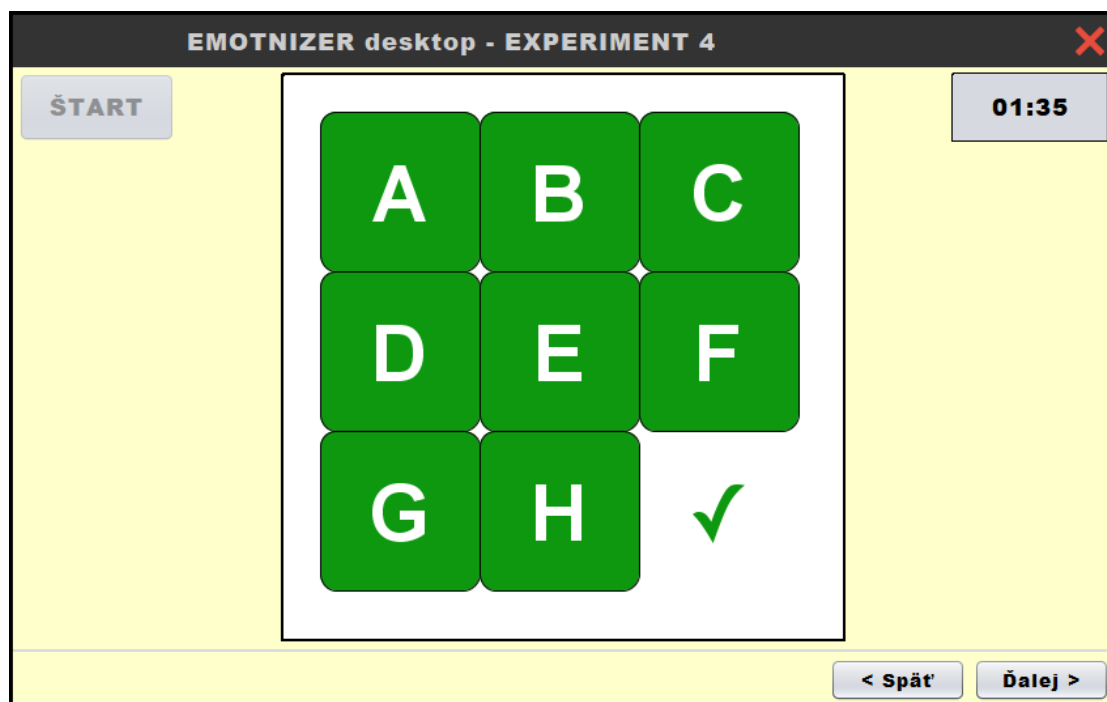


Obrázok 17 Experiment 3 v desktopovej aplikácii

3.5.4 Experiment 4 – Hlavalam 8 štvorcov

Hlavalam 8 štvorcov (alebo 8 PUZZLE problem) je hádankou, ktorú vynašiel Noyes Palmer Chapman v 70. rokoch 19. storočia. Hrá sa na hracom poli pozostávajúcom z mriežky 3x3 (obsahujúci 8 štvorcov s číslami 1 až 8 a 1 štvorec, ktorý je prázdny) (Princeton.edu, 2010). My sme však v našom experimente namiesto čísel na štvorcoch použili písmená A až H. Cieľom je posunúť všetky štvorce tak (je dovolené posúvať štvorce s písmenom horizontálne aj vertikálne do prázdneho štvorca klikaním), aby boli usporiadané abecedne, čiže usporiadané od ľavého horného

štvorca (označený písmenom A) až po pravý dolný štvorec, tak ako je zobrazené na Obrázku 18. Pri tvorbe hlavolamu sme sa inšpirovali Saurelom (2019).



Obrázok 18 Experiment 4 v desktopovej aplikácii

Opäť to používatelia musia stihnúť do vypršania časového limitu (2 minúty), ak to nestihnú, presnosť (úspešnosť) vyriešenia daného experimentu meriame ako podiel počtu správne umiestnených štvorcov a počet všetkých očíslovaných štvorcov. Všetky výstupné charakteristiky z Experimentov 3 a 4 aj s popisom sú uvedené v Tabuľke 2.

Tabuľka 2 Charakteristiky ukladané do databázy pre Experiment 3 a 4

Názov atribútu	Popis atribútu
COMPLETED	koľkokrát bol experiment dokončený daným používateľom
SAM1_BEFORE	hodnota valencie pred experimentom
SAM2_BEFORE	hodnota vzrušenia pred experimentom
CPS_AVERAGE	priemerná frekvencia klikania tlačidlom myši za sekundu
CPS_MAX	maximálna frekvencia klikania tlačidlom myši za sekundu
DTM_AVERAGE	priemerná dĺžka držania tlačidla myši v milisekundách
DTM_MAX	maximálna dĺžka držania tlačidla myši v milisekundách
PAUSE_AVERAGE	priemerná pauza medzi kliknutiami
PAUSE_MAX	maximálna pauza medzi kliknutiami
DM_AVERAGE	priemerná vzdialenosť, ktorú prejde kurzor myši v pixeloch za sekundu
DM_MAX	maximálna vzdialenosť, ktorú prejde kurzor myši v pixeloch za sekundu
SAM1_AFTER	hodnota valencie po experimente
SAM2_AFTER	hodnota vzrušenia po experimente
ACCURACY	presnosť správnosti vyriešeného experimentu
TIME	čas trvania experimentu (v sekundách)

3.6 TVORBA EXPERIMENTOV V MOBILNEJ APLIKÁCI

Pred začiatkom tvorby experimentov sme si museli vybrať, pre ktorý mobilný operačný systém vôbec takúto aplikáciu naprogramujeme. Podľa portálu StatCounter (2021) je dlhodobo najpoužívanejším operačným systémom Android s podielom vyše 72% na trhu predaja v celosvetovom rebríčku za obdobie júl 2021. Ďalšou veľkou výhodou Androidu je, že umožňuje programovanie aplikácií v jazyku Java, čo je programovací jazyk, ktorý sme využili na naprogramovanie desktopovej aplikácie. Preto dokážeme veľmi jednoducho implementovať už nami napísaný kód, ktorý zbiera dáta charakteristík správania sa používateľa z desktopovej aplikácie priamo v mobilnej aplikácii. Ako vývojové prostredie sme použili *Android Studio* založené na *IntelliJ IDEA*. Pri vytváraní nového projektu sme si museli určiť minimálnu verziu Androidu, na ktorej chceme, aby sa naša aplikácia dala spustiť. Vybrali sme si API level 22, čo je Android 5.1 a orientáciu obrazovky sme vo všetkých zobrazeniach nastavili na portrét.

Začali sme tým, že sme si vytvorili úvodnú prihlasovaciu obrazovku, ktorá je dôležitá najmä pre prepojenie dát z experimentov s používateľmi, ktorí ich absolvovali. Pri aplikovaní už nami naprogramovaného kódu do mobilnej aplikácie, ktorý komunikoval s databázou, sme narazili na viacero problémov. Prvým, tým ľahším, bolo nastaviť oprávnenie v manifeste pridaným tohto riadka kódu:

```
< uses-permission android:name="android.permission.INTERNET" />.
```

Druhým problémom bolo to, že aplikácia sa pri pokuse pripojiť na databázu pomocou PHP skriptu neustále zatvárala (zrútila sa). To bolo z dôvodu, že pripojenie k databáze vyžaduje určitý čas a všetko, čo sa nevykoná hneď, musí byť vo vnútri vlákna. Preto sme použili abstraktnú triedu **AsyncTask**, ktorá implementovala 4 povinné metódy. Využili sme len dve z nich a to bola metóda **doInBackground**, ktorá obsahovala celý kód na posielanie dát, ktorý je zobrazený na Obrázkoch 9 a 10 v kapitole 3.2, a tiež sme použili metódu **onPostExecute**, ktorá vracala výsledok z predchádzajúcej metódy prostredníctvom nejakého GUI prvku (v prípade prihlásenia, zmenilo aktivitu, v opačnom prípade oznámilo nesprávne heslo). Ešte tu boli metódy **onPreExecute** (vykoná sa pred metódou *doInBackground*) a **onProgressUpdate** (dokáže zobrazovať stav ešte počas vykonávania hlavnej metódy). Aplikovanie tejto abstraktnej triedy *AsyncTask* je zobrazené na Obrázku 19.

```

public class BackgroundWorker extends AsyncTask<String, Void, String> {
    @Override
    protected String doInBackground(String... params) {
        .....posielanie dát.....
        return result;
    }
    @Override
    protected void onPostExecute(String result) {
        ...zápis do používateľského rozhrania...
    }
}

```

Obrázok 19 Aplikovanie abstraktnej triedy AsyncTask

3.6.1 Experiment 5 – Prepis vopred definovaného textu

V prípade experimentov v desktopovej aplikácii, ktoré sú zamerané na prepis textu používali používatelia hardvérovú klávesnicu. V každom smartfóne sa dnes používa softvérová klávesnica. Preto sme najskôr chceli odchytiť behaviorálne charakteristiky priamo z takejto klávesnice. Avšak, nie každý človek používa rovnakú softvérovú klávesnicu, čo by nebol až taký problém, ale väčšina takýchto klávesníc používa automatickú opravu textu (autokorekciu), čo je v našom prípade dosť veľká komplikácia, pretože text musí byť reálne napísaný používateľom, nie vytukaný najčastejšími návrhmi pri oprave slov. A ešte väčšia komplikácia bola, že väčšina najpoužívanejších softvérových klávesníc nedovoľuje odchytyvanie stlačení týchto klávesov tretím stranám, tým pádom nedokážeme z písania na takýchto klávesniciach získať charakteristiky správania sa. Preto sme sa rozhodli všetky tieto tri problémy vyriešiť jedným komplexným riešením, ktorým je **vlastná softvérová klávesnica** priamo pre našu mobilnú aplikáciu.

Pri tvorbe našej klávesnice sme si pomohli návodom od Saurela (2018), ktorý robil iba základnú numerickú klávesnicu. Našu klávesnicu tvorí hlavne 26 základných písmen latinskej abecedy, číslice od 0 po 9, špeciálne znaky a funkčné klávesy (Shift, medzerník, Backspace a Enter). Spolu je tak na našej klávesnici až 67 tlačidiel. Klávesnica je zobrazená na Obrázku 20. Najskôr sme si vytvorili XML súbor, kde definujeme štruktúru používateľského rozhrania našej klávesnice. Potom sme vytvorili triedu v jazyku Java, ktorá implementuje rozhranie **OnTouchListener**. Pomocou tohto rozhrania dokážeme pristúpiť k metóde **OnTouch** a dokážeme tak odchytiť stlačenia aj uvoľnenia klávesov a tiež čas, kedy presne tieto akcie nastali.



Obrázok 20 Klávesnica v aplikácii Emotnizer mobile

Najskôr sme však museli prepojiť jednotlivé tlačidlá s ich reálnymi znakovými hodnotami. To sme urobili pomocou poľa a metódy *put*, kde sme posielali ako parametre *id tlačidla* a jeho hodnotu v textovej podobe. Následne po ťuknutí na tlačidlo vieme z poľa vrátiť jeho reálnu hodnotu znaku. Následne sme urobili aktivitu pre experiment, v ktorom je základ pre písanie textu pomocou našej klávesnice *EditText*. To je komponent, do ktorého sa bude reálne text po stlačení tlačidla vpisovať.

V prípade systémových klávesníc Android používa na presmerovanie klávesnice k *EditTextu* inštanciu *InputMethodManager*. Toto sme museli zablockovať, keď chceme, aby sa nám systémová klávesnica neotvorila a zároveň, aby sa nám otvorila tá naša. To sa nám podarilo pomocou príkazu *FLAG_ALT_FOCUSABLE_IM*. Potom už len získame inštanciu *InputConnection* z nášho *EditTextu* a nastavíme ju našej klávesnici. A potom pomocou tohto *InputConnection* môžeme pridávať alebo mazat' znaky pomocou ťukania na tlačidlá našej klávesnice.

```
super.onCreate(savedInstanceState);
setContentView(R.layout.experiment1);
getWindow().setFlags(WindowManager.LayoutParams.FLAG_ALT_FOCUSABLE_IM,
    WindowManager.LayoutParams.FLAG_ALT_FOCUSABLE_IM);

EditText editText = (EditText) findViewById(R.id.editText);
MyKeyboard keyboard = (MyKeyboard) findViewById(R.id.keyboard);
editText.setRawInputType(InputType.TYPE_CLASS_TEXT);
editText.setTextIsSelectable(true);

InputConnection ic = editText.onCreateInputConnection(new EditorInfo());
keyboard.setInputConnection(ic);
```

Obrázok 21 Zdrojový kód prepojenia *EditTextu* s našou klávesnicou

Po dokončení našej klávesnice sme mohli naprogramovať experiment. V tomto experimente bolo taktiež cieľom prepísať text, ako tomu bolo v prvom experimente desktopovej aplikácie. Chceli sme zistiť, či budú výsledky získané z týchto

experimentov porovnateľné. Samozrejme, aj v tomto experimente je cieľom prepísať text z horného textového poľa do spodného textového poľa v časovom limite (až 6 minút), pričom tento text obsahoval 651 znakov vrátane medzier. Asi je zrejmé, že keď sa na našej klávesnici nenachádzajú znaky s diakritickými znamienkami, budú musieť používatelia prepisovať text bez diakritiky. Takisto sme na záver porovnávali vopred definovaný text s prepísaným textom a pomocou Levenshteinovej vzdialenosti sme počítali správnosť prepísaného textu. Výstupné charakteristiky sú rovnaké ako v prípade experimentov 1 a 2 z desktopovej aplikácie. Avšak v tomto experimente máme o jeden druh klávesov menej. Rozlišujeme iba bežné klávesy a mazacie. Mazacia je len jedna a to je klávesa BackSpace, čím sa reťazec skracuje a bežnými sa reťazec predlžuje. Nerozlišujeme tu funkčné klávesy ako Shift, lebo na klávesniciach v Androide funguje Shift odlišne ako na počítačoch, čiže netreba ho držať, keď chce používateľ napísať veľké písmeno.

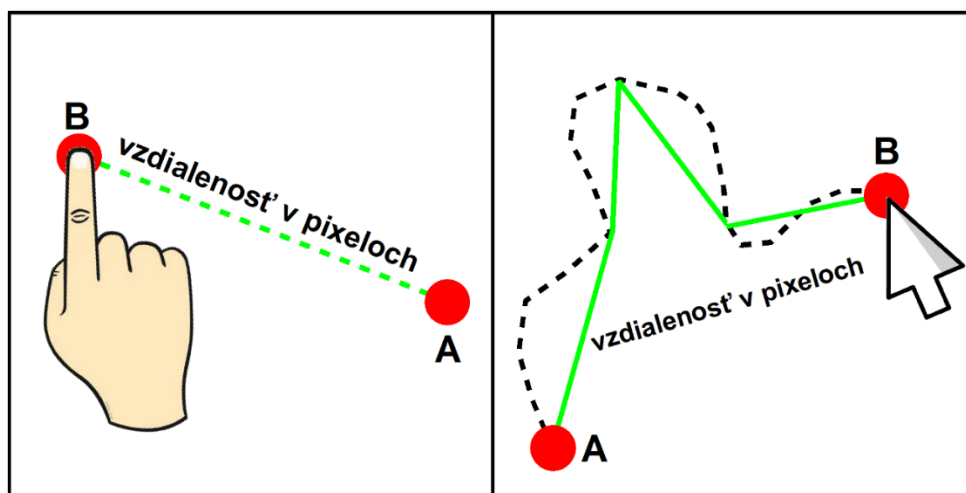
3.6.2 Experiment 6 – Hlavalam 8 štvorcov

Druhým experimentom pre mobilnú aplikáciu sme zvolili znova Hlavalam 8 štvorcov, ktorý sme mali aj v desktopovej aplikácii. Aj v tomto experimente je cieľom posunúť všetky štvorce s písmenami A až H tak, aby boli usporiadané abecedne a stanovený čas pre splnenie experimentu je 2 minúty. Jediný rozdiel medzi experimentom v desktopovej aplikácii a týmto experimentom v mobilnej aplikácii je charakteristika správania sa **DistanceMoved** (dm = vzdialenosť, ktorú prejde kurzor myši v pixeloch za sekundu). Okno experimentu desktopovej aplikácie má na každom počítači rovnakú veľkosť v pixeloch, zatiaľ čo okno mobilnej aplikácie je také veľké ako obrazovka daného zariadenia a navyše každá obrazovka zariadenia môže mať inú hodnotu DPI (počet pixelov na palec). Preto sme museli previesť hodnotu pixelov na DIP (density independent pixel), pomocou Vzorca 3, kde *pixel* je hodnota vyjadrená v pixeloch a *density* je mierka hustoty pixelov obrazovky, ktorú sme zistili pomocou príkazu priamo v aplikácii. Tento vzorec sme testovali na emulátore v Android Studio a mierka hustoty pixelov obrazovky tu bola 2,625 a testovali sme to aj na fyzickom zariadení s hustotou 3,0 a výsledky boli takmer rovnaké.

$$DIP = \frac{pixel - 0,5}{density}$$

Vzorec 5 Prevod hodnoty v pixeloch na hodnotu v DIP

Problémom však stále zostáva, že pri pohybe kurzora myši vieme povedať kadiaľ sa kurzor pohybuje a vieme vypočítať jeho priemernú vzdialenosť v pixeloch prejdenu za sekundu. To sa však takto nedá robiť v mobilnej aplikácii, lebo tu sa kurzor nenachádza a ovládať aplikáciu musíme pomocou prsta, ktorý sa stále nenachádza na obrazovke, aby sme mohli sledovať jeho trajektóriu. Preto sme sa rozhodli, že budeme počítať so vzdialenosťou medzi ťuknutiami prsta. Na Obrázku 22 je vpravo zobrazené meranie vzdialenosti pri pohybe kurzora myši, kde čiarkovaná čiara je skutočná trajektória kurzora a zelená čiara je vzdialenosť medzi dvoma bodmi na súradnicovej sústave. Keďže meranie prebieha veľmi často (každých 100 milisekúnd), výsledná vzdialenosť je veľmi presná. Vľavo na obrázku je zobrazené ťukanie prstom na súradnicovej sústave, kde vzdialenosť sa meria iba medzi jednotlivými ťuknutiami.



Obrázok 22 Vzdialenosť v mobilnej a desktopovej aplikácii

Výstupné charakteristiky sú rovnaké ako v prípade experimentov 3 a 4 z desktopovej aplikácie až na atribúty *dm_average* a *dm_max*, tu sú to priemerné a maximálne vzdialenosti v dip medzi ťuknutiami prsta na obrazovke za sekundu.

4 ANALÝZA DÁT A KLASIFIKÁCIA EMÓCIÍ POUŽÍVATEĽA

V tejto kapitole si podrobne rozoberieme fázy zberu, analýzy a predspracovania dát a nasadenie viacerých klasifikačných algoritmov strojového učenia na týchto dátach vo vybranom programovacom jazyku. Dáta, s ktorými budeme pracovať, boli získané zo šiestich experimentov, ktoré sme podrobne popísali v tretej kapitole.

4.1 ZBER DÁT

Vo fáze zberu dát sme pôvodne požiadali až 76 osôb, aby sa zúčastnili našich experimentov, avšak 25 z nich sa len registrovali a pravdepodobne po zistení, že si musia niečo ďalej stiahnuť, odmietli v absolvovaní experimentov ďalej pokračovať. Našich experimentov sa zúčastnilo **51 osôb** (33 mužov a 18 žien) s priemerným vekom 19,75 rokov v období **od augusta do októbra 2021**. Ďalej sme z dotazníkových otázok pri registrácii zistili, že 46 osôb bolo pravákov a 5 ľavákov, 49 osôb bolo slovenskej národnosti a dvaja maďarskej, priemerne ovládajú slovenský jazyk na 86,63%, priemernú mieru citlivosti na podnety majú na úrovni 60,92%, svoje emócie si priemerne uvedomujú na 76,92% a 18 osôb ovláda prstoklad, avšak až 33 osôb prstoklad neovláda. Nanešťastie, ani všetci účastníci, ktorí sa podieľali na absolvovaní experimentov, neabsolvovali všetky experimenty. Najčastejším dôvodom bol iný operačný systém pre mobilnú verziu aplikácie, ako mali niektorí participant, ale aj nedostatočná motivácia pre absolvovanie daných experimentov. Celkový čas strávený absolvovaním experimentov používateľmi v oboch aplikáciách dosahuje 27 703 sekúnd (čo je po prepočte približne 7,7 hodín).

4.2 ANALÝZA DÁT

Dáta získané z experimentov sme exportovali z databázy webového servera do Excelu. Na analýzu dát, ale aj na predspracovanie dát a klasifikáciu modelmi strojového učenia sme si vybrali programovací jazyk **Python**, ktorý je flexibilný, ponúka stručný a ľahko čitateľný zdrojový kód a obsahuje obrovské množstvo knižníc, ktoré uľahčujú prácu s dátami a s implementáciou strojového učenia na týchto dátach. Ako vývojové prostredie sme si zvolili **Jupyter Notebook**, ktorý umožňuje kombináciu zdrojového kódu so štruktúrovanými komentármi, obrázkami, či grafmi v jednom ucelenom dokumente.

Pre načítanie dátových súborov (datasetov) sme použili metódu *read_excel* z knižnice **Pandas**, ktorá umožňuje, okrem iného, aj ľahkú manipuláciu s dátami. Pomocou metódy *describe* si vieme rýchlo zobrazit' základnú štatistiku nášho datasetu, ktorá nám prezradí, koľko má náš dataset záznamov, stĺpcov, či aké rozloženie hodnôt obsahuje. Na základe tejto základnej štatistiky sme jednoznačne vedeli, že musíme vo fáze predspracovania naše dáta **zoškálovať** (znormalizovať), pretože všetky vstupné premenné majú iný číselný rozsah, a pri klasifikovaní emócií by mali modely strojového učenia problém klasifikovať jednotlivé emocionálne stavy.

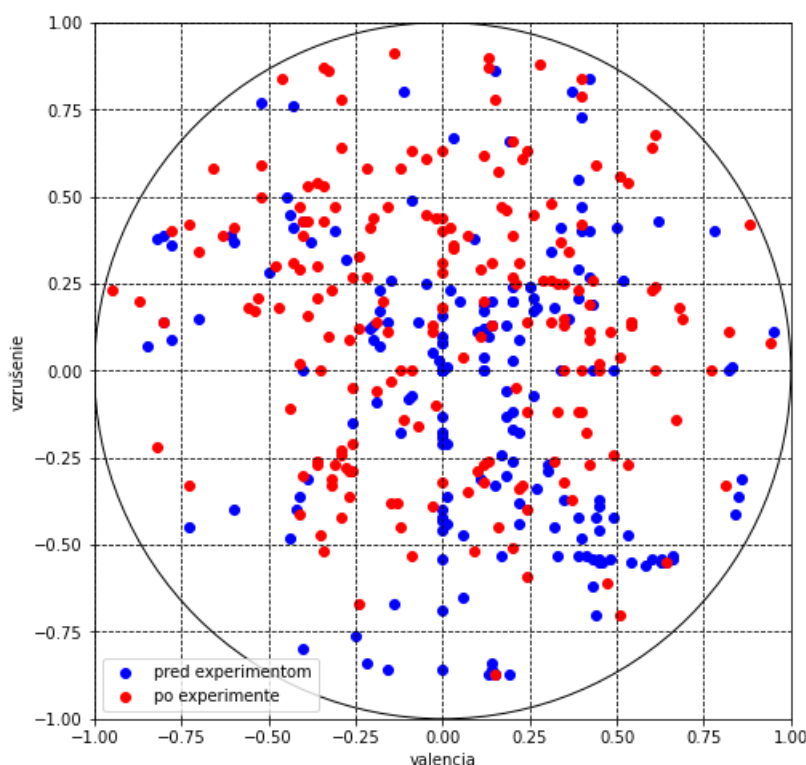
Pri prvotnom pokuse o klasifikáciu sme zistili, že výsledky by nedosahovali požadovanú úroveň a pravdepodobne máme nedostatočné množstvo vstupných premenných. Našťastie pre nás, sme počítali aj s touto možnosťou, aj keď sa do našej databázy reálne posielali charakteristiky, ktoré vypočítali aplikácie na konci experimentov (popísané v tretej kapitole), posielali sme aj celý textový reťazec (oddelený znakovými separátormi), ktorý obsahoval postupne vypočítavané hodnoty z bežiacich vlákien v našich aplikáciách. Preto sme boli schopní spätne pristupovať k dátam a pomocou externého programu v Jave sme vytvorili ďalšie vstupné premenné. Pôvodne sme pre každú charakteristiku ukladali do databázy iba **priemery** a **maximá**. Teraz sme navyše vytvorili pre všetky charakteristiky aj **minimá**, **smerodajné odchýlky** a **početnosti**. Pre ukážku, novo vytvorené charakteristiky z experimentov 3, 4 a 6 zobrazujeme v Tabuľke 3. Datasety okrem zobrazených charakteristík obsahujú aj pôvodné premenné ako: *user_id*, *completed*, *accuracy*, *time* a *sebahodnotenia pred a po experimente*.

Tabuľka 3 Doplnenie charakteristík

CPS_AVERAGE	CPS_STD	CPS_MIN	CPS_MAX	CPS_COUNT
DTM_AVERAGE	DTM_STD	DTM_MIN	DTM_MAX	DTM_COUNT
PAUSE_AVERAGE	PAUSE_STD	PAUSE_MIN	PAUSE_MAX	PAUSE_COUNT
DM_AVERAGE	DM_STD	DM_MIN	DM_MAX	DM_COUNT

Pred aj po každom experimente museli používatelia ohodnotiť svoj emocionálny stav na Russelovom kruhovom modeli emócií prostredníctvom dimenzií valencie a vzrušenia. V programe Microsoft Excel sme vypočítali rozdiely hodnôt medzi sebahodnoteniami pred a po experimentoch pre valenciu aj pre vzrušenie. Spriemerovaním týchto rozdielov sme zistili, že používatelia sa v priemere cítili nepríjemnejšie po experimente ako pred ním, čo naznačuje kladná hodnota valencie

0,081 a zároveň pocítovali vyššiu hodnotu vzrušenia po experimente ako pred ním, čo zase naznačuje záporná hodnota vzrušenia -0,210. Priemerné hodnoty nám nestačili, a preto sme vypočítali priemery z absolútnych hodnôt rozdielov. Zistili sme, že priemerne naše experimenty ovplyvnili emócie používateľov vo vysokej miere, pretože zmena v sebahodnotení pred a po experimentoch dosahuje hodnotu približne 0,44 pre obe dimenzie, pričom povolený rozsah ohodnotenia svojich emócií bol od -1 do 1. Všetky označenia svojich emocionálnych stavov pred (modré značky) aj po (červené značky) experimente sú vizualizované v Grafe 1.



Graf 1 Vizualizácia sebahodnotení emócií používateľmi

4.3 PREDSPRACOVANIE DÁT

Prvou úlohou vo fáze predspracovania dát je zmena sebahodnotenia z podoby reálnych čísel do kategóriálnej premennej. Rozhodli sme sa, že budeme pracovať už len s hodnotením používateľov po experimentoch, pretože tieto hodnotenia odzrkadľujú vplyv emócií na správanie sa pri absolvovaní experimentov. Vytvoríme si dve kategóriálne (závislé) premenné, keďže máme hodnotenia na dvoch dimenziách. Ďalším krokom bolo nájsť vhodný počet kategórií a interval rozdelenia hodnôt pre jednotlivé premenné. Postupným skúšaním a overovaním sme dospeli k záveru, že budeme mať 3 kategórie pre každú z premenných. Premenná valencie bude obsahovať kategórie: *negatívna*, *neutrálna* a *pozitívna* a premenná vzrušenia bude obsahovať kategórie: *nízke*,

stredné a *vysoké*. V Tabuľke 4 sú zobrazené jednotlivé kategórie pre cieľové premenné spolu s ich intervalom rozdelenia. Aj keď sa na základe intervalov z tabuľky zdá, že pre neutrálnu kategóriu valencie a strednú kategóriu vzrušenia je interval veľmi malý, tak v skutočnosti je kruhový priestor rozdelený takmer rovnomerne.

Tabuľka 4 Kategorizácia cieľových premenných

Premenná valencie	Premenná vzrušenia	Interval rozdelenia
NEGATÍVNA (0)	NÍZKE (0)	$-1,00 \leq \text{hodnota} < -0,25$
NEUTRÁLNA (1)	STREDNÉ (1)	$-0,25 \leq \text{hodnota} \leq 0,25$
POZITÍVNA (2)	VYSOKÉ (2)	$0,25 < \text{hodnota} \leq 1,00$

Ďalej v Tabuľke 5 uvádzame početnosti záznamov jednotlivých kategórií oboch kategoriálnych premenných (valencia, vzrušenie) pre všetkých 6 dátových súborov. Z tejto tabuľky je možné vyčítať, či bude dataset vyvážený, teda, či bude počet záznamov približne rovnaký v každej kategórii (triede). Napríklad v dátovom súbore 1 pre kategorickú premennú valencie si môžeme všimnúť, že záznamy sú rozdelené v pomere 9:18:8, čo znamená, že dataset bude mierne nevyvážený. V prípade, že by bol dataset priveľmi nevyvážený, tak pre majoritnú triedu by klasifikačný model strojového učenia fungoval dokonale, ale pre minoritné triedy by dobre nefungoval, pretože by nemal dostatočný počet záznamov na natrénovanie sa. Všeobecne sa odporúča prevzorkovať dátový súbor, čím sa počet záznamov vyrovná, ale pri väčšine modelov pri učení dôjde k preučeniu sa. Preto použijeme na vyhodnotenie výkonu modelov strojového učenia okrem tradičnej metriky Accuracy (presnosť) aj metriku F1-score, ktorá je váženým harmonickým priemerom metrík Precision a Recall, ktoré berú do úvahy počty falošne pozitívnych a falošne negatívnych prípadov. Na vyhodnotenie modelov použijeme aj maticu zmätku (Confusion matrix), ktorej princíp si vysvetlíme priamo pri použití algoritmu strojového učenia v nasledujúcej podkapitole.

Tabuľka 5 Početnosti záznamov jednotlivých kategórií všetkých dátových súborov

Kategória		Dat. 1	Dat. 2	Dat. 3	Dat. 4	Dat. 5	Dat. 6
valencia	negatívna (0)	9	12	9	13	10	9
	neutrálna (1)	18	13	14	17	9	8
	pozitívna (2)	8	8	12	11	7	13
vzrušenie	nízke (0)	8	9	7	9	6	5
	stredné (1)	14	13	13	13	12	11
	vysoké (2)	13	11	15	19	8	14

Po zmene hodnotení emócií používateľov do podoby kategoriálnych cieľových premenných je našou druhou úlohou vo fáze predspracovania dát **zoškálovať** vstupné dáta. Na to použijeme metódu *fit_transform*, triedy **StandardScaler** z knižnice **Sklearn** (najpoužívanější knižnica pri práci so strojovým učením v Pythone), kde na vstup dodáme iba vstupné dáta, čiže všetky dáta bez dvoch posledných stĺpcov, kde sú uložené cieľové kategorické premenné valencie a vzrušenia. Trieda *StandardScaler* transformuje dáta pre každú premennú do podoby, kedy je priemer hodnôt 0 a štandardná odchýlka sa rovná 1.

```
from sklearn.preprocessing import StandardScaler
standard_scaler = StandardScaler()

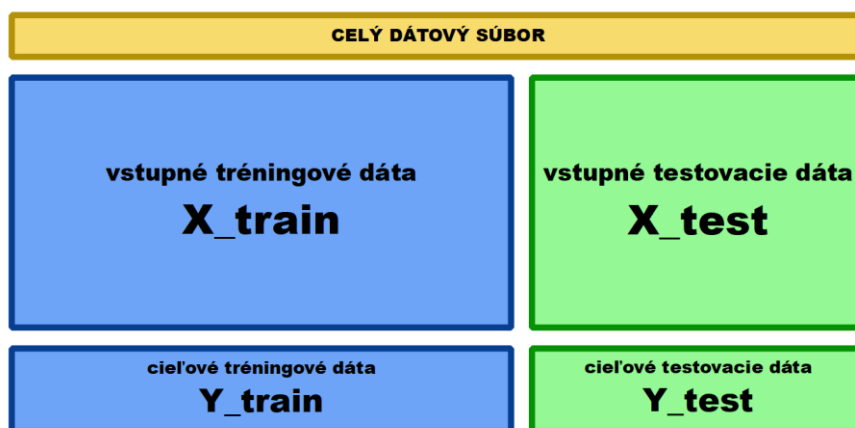
X_vstup = data.iloc[:, :-2].values
X = standard_scaler.fit_transform(X_vstup)
```

Obrázok 23 Zoškáľovanie vstupných dát

Po zoškáľovaní vstupných dát môžeme tieto dáta spolu s cieľovými premennými rozdeliť na **tréninové** a **testovacie** podmnožiny. Dôvod delenia je veľmi jednoduchý. Pomocou tejto techniky môžeme vyhodnotiť výkon daného klasifikačného modelu. Na tréninovej sade dát sa algoritmus strojového učenia pri učení s učiteľom natrénuje, učí sa a na testovacej sade sa overí presnosť daného klasifikačného modelu. Väčšinou sa celý dátový súbor delí v pomere 80:20, ale aj 70:30, v našom prípade budeme deliť dátový súbor v pomere 60:40, pre tréninové a testovacie podmnožiny, pretože množstvo našich dát nie je príliš veľké. Na Obrázku 24 zobrazujeme ukážku zdrojového kódu, kde prostredníctvom metódy *train_test_split* z knižnice Sklearn delíme dataset na vstupné tréninové a testovacie podmnožiny a cieľové tréninové a testovacie podmnožiny (rozdelenie vizualizujeme na Obrázku 25). Ako parametre do tejto metódy vstupujú *zoškáľované vstupné dáta*, *jedna cieľová kategorická premenná* (bud' valencia alebo vzrušenie), *veľkosť tréninovej podmnožiny* a parameter *stratify*, ktorý zabezpečí rozdelenie dát tak, aby jednotlivé kategórie cieľovej premennej mali rovnaké zastúpenie v oboch podmnožinách.

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,train_size=0.6,stratify=Y)
```

Obrázok 24 Rozdelenie vstupných dát na tréninové a testovacie



Obrázok 25 Tréningové a testovacie podmnožiny

4.4 POUŽITIE ALGORITMOV STROJOVÉHO UČENIA

V tejto podkapitole popíšeme použitie viacerých klasifikačných algoritmov strojového učenia s učiteľom (konkrétne v abecednom poradí: *K-Nearest Neighbors*, *Logistická regresia*, *Naive Bayes*, *Neurónová sieť*, *Random Forest* a *Support Vector Machine*) v programovacom jazyku Python. Všetky klasifikačné algoritmy musíme nasadiť na všetkých 6 dátových súborov pre valenciu aj pre vzrušenie, čím nám vznikne až 36 modelov (6 algoritmov x 6 dátových súborov), pričom všetky modely sú priložené v prílohe. Pri každom klasifikátore si podrobnejšie rozoberieme trénovanie, testovanie a vyhodnotenie modelu iba pre jeden konkrétny dátový súbor, pričom výsledky pre všetky dátové súbory uvedieme v prehľadnej tabuľke pri každom klasifikátore.

4.4.1 K-Nearest Neighbors

Dátový súbor z experimentu 1 je zložený z charakteristík správania sa pri používaní klávesnice. Celý dátový súbor obsahuje 35 záznamov a pozostáva z 39 stĺpcov, z čoho je 37 vstupných premenných a 2 sú výstupné cieľové premenné. Obe cieľové premenné (valencia a vzrušenie) pozostávajú z troch úrovní (0, 1, 2). Počet záznamov je tak v pomere 9:18:8 pre negatívnu, neutrálnu a pozitívnu valenciu a zase v pomere 8:14:13 pre nízke, stredné a vysoké vzrušenie. Rozdelenie záznamov podľa valencie vytvorilo viac nevyvážený dataset, ako pri rozdelení podľa vzrušenia. Pre premenné valencie a vzrušenia sme aj navyše pre všetky datasety počítali štatistickú mieru závislosti (koreláciu), ktorú sme počítali priamo v Pythone. S valenciou najviac korelujú vstupné charakteristiky: *pauza medzi stláčaniami klávesov*, *presnosť prepísaného textu* a *držanie mazacích klávesov*. A so vzrušením majú najvyššiu mieru závislosti taktiež charakteristiky *pauzy a presnosti prepísaného textu*, ale navyše aj

charakteristika *držania funkčných klávesov (Shift, Ctrl)*. To môže znamenať, že najmä dĺžka časového limitu pri prepise textu mohla vyvolať danú emóciu.

Na klasifikovanie emocionálnej kategórie použijeme viacero algoritmov, začneme však klasifikátorom **K-Nearest Neighbors (KNN)**, ktorý sme spomenuli aj v analýze súčasného stavu. Najdôležitejším faktorom v tomto algoritme je K počet najbližších susedov. Vo všeobecnosti platí, že počet susedných bodov by mal byť určený podľa rovnice $K = \frac{\sqrt{\text{počet tréningových záznamov}}}{2}$, (platí pre väčšie datasety). Algoritmus KNN začína vypočítaním vzdialeností medzi novým bodom a tréningovými bodmi. Na základe minimálnej vzdialenosti sa vyberie K počet najbližších susedov a trieda, ktorá obsahuje väčšinu týchto bodov sa stáva predikovanou triedou pre nový bod (Christopher, 2021). V Pythone sme tento klasifikátor implementovali pomocou knižnice Sklearn a ako parametre vstupujúce do klasifikátora sme nastavili počet susedov na 3 (pri vyššou počte klesala presnosť klasifikátora) a metriku sme zvolili *minkowski*, pomocou ktorej sa počíta vzdialenosť medzi bodmi. Ekvivalentom Minkowského vzdialenosti je Euklidovská a Manhattanská vzdialenosť. Pomocou metódy *fit* sme klasifikátor natrénovali na vopred rozdelených a zoškálovaných tréningových vstupných a cieľových dátach. A na natrénovanom modeli sme pomocou metódy *predict* a vstupných testovacích dát predikovali výsledné cieľové stavy, ktoré sme uložili do poľa (Y_pred).

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors = 3, metric = 'minkowski')
model.fit(X_train, Y_train)
Y_pred = model.predict(X_test)
sklearn.metrics.accuracy_score(Y_test, Y_pred)
sklearn.metrics.f1_score(Y_test, Y_pred, average = "macro")
```

Obrázok 26 Vytvorenie, trénovanie a testovanie modelu KNN

Prostredníctvom metrík *accuracy_score* a *f1_score* vieme vyhodnotiť presnosť klasifikačného modelu. Vstupom do týchto metód je pole testovacích cieľov (Y_test) a pole predikovaných cieľov (Y_pred). **Accuracy (presnosť)** predstavuje vlastne podiel počtu správne predikovaných (zaklasifikovaných) stavov a počtu všetkých stavov. **F1 score** predstavuje harmonický priemer metrík **Precision** (súčet skutočne pozitívnych výsledkov vo všetkých triedach vydelený súčtom skutočne pozitívnych a falošne pozitívnych výsledkov vo všetkých triedach) a **Recall** (súčet skutočne pozitívnych výsledkov vo všetkých triedach vydelený súčtom skutočne pozitívnych a falošne

negatívnych výsledkov vo všetkých triedach) (Scikit-learn, 2011). Keďže sa F1 score počíta pre každú triedu, musíme všetky F1 score spriemerovať, aby sme získali jednu konkrétnu percentuálnu hodnotu presnosti. My sme využili *macro priemer*, ktorý iba jednoducho spriemeruje všetky F1 score, čiže pristupuje k triedam, ako keby mali rovnakú váhu. Všetky naše datasety sú iba mierne nevyvážené, takže na tomto parametri toľko nezáleží, v prípade vysokej nevyváženosti sa odporúča použiť *weighted priemer*.

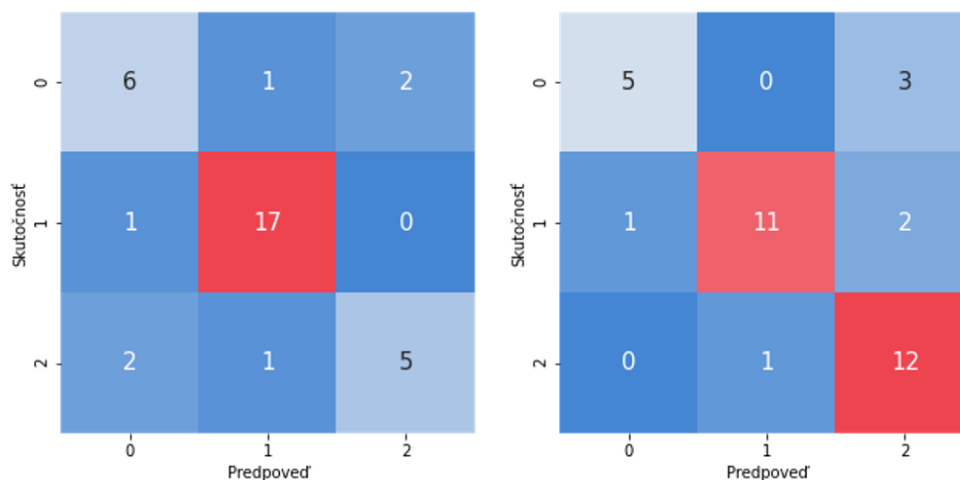
Pri spustení dosahujeme priemerne presnosť (pre dataset 1) klasifikovaním valencie 78,57% (metrikou accuracy), 73,61% (metrikou F1 score) a klasifikovaním vzrušenia dosahujeme 78,57% (accuracy), 79,48% (F1 score). Pre lepšie overenie modelu sme využili metódu *k-násobnej krížovej validácie*, ktorej princíp si vysvetlíme v nasledujúcej podkapitole. Touto metódou sme sa dostali ku komplexnejšiemu vyhodnoteniu modelu, pričom sme dostali veľmi porovnateľné výsledky, ktoré sú zobrazené v Tabuľke 6, nie len pre dataset 1, ale aj pre všetky naše datasety.

Tabuľka 6 Vyhodnotenie presnosti klasifikačného modelu KNN

K-Nearest Neighbors (%)		Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6
VALENCIA	ACC	79,80	75,76	76,52	75,27	72,69	73,33
	F1	75,11	74,29	71,66	73,04	69,22	71,66
VZRUŠENIE	ACC	80,05	78,79	71,47	83,15	79,63	83,33
	F1	78,52	78,22	68,30	80,21	78,89	74,24

Na vyhodnotenie presnosti modelu sa používa aj **Matica zmätku** (Confusion matrix), čo je vlastne jednoduchá tabuľka, v našom prípade veľkosti 3x3 (pretože máme 3 stavy), ktorá pozostáva zo správnych a nesprávnych predpovedí klasifikátora. Na hlavnej diagonále sú správne predikované stavy a všetky ostatné sú nesprávne predikované stavy (Scikit-learn, 2011). Na Obrázku 27 je zobrazená Matica zmätku, ktorú sme vygenerovali pri použití metódy 3-násobnej krížovej validácie pomocou funkcie *confusion_matrix* a parametrov cieľových testovacích dát (*Y_test*) a predikovaných stavov (*Y_pred*). Vľavo je matica pre valenciu, ktorá vo výstupe zobrazuje správne klasifikovanú negatívnu valenciu pre 6 prípadov, nesprávne klasifikovanú ako neutrálnu pre 1 prípad, a taktiež nesprávne klasifikovanú ako pozitívnu pre 2 prípady. Analogicky klasifikovalo 17 správne ako neutrálnu valenciu a 5 prípadov správne ako pozitívnu valenciu. Celkovo si môžeme všimnúť, že

najpočetnejšie triedy klasifikovalo najlepšie a minoritné mali najviac nesprávne klasifikovaných prípadov.



Obrázok 27 Matice zmätka pre valenciu (vľavo) a pre vzrušenie (vpravo)

4.4.2 Logistická regresia

Všeobecne sme logistickú regresiu popísali už v analýze súčasného stavu, teraz sa budeme tejto metóde strojového učenia sa venovať prakticky. Najskôr však treba spomenúť, že my budeme pracovať s **multinominálnou logistickou regresiou**, pretože klasifikujeme až 3 stavy (0, 1, 2) pre valenciu aj vzrušenie. Pri tomto klasifikátore priblížime pozornosť na druhý dataset, ktorý pozostáva z charakteristík správania sa pri používaní klávesnice v experimente 2. Podobne ako prvý dataset, obsahuje tie isté charakteristiky, líši sa však v počte záznamov, ktorých je 33. Počet záznamov je tu rozdelený v pomere 12:13:8 pre negatívnu, neutrálnu a pozitívnu valenciu a pre nízke, stredné a vysoké vzrušenie je pomer 9:13:11. S valenciou najviac korelujú vstupné charakteristiky: *držanie všetkých typov klávesov* a *pauza medzi stláčaniami klávesov*. A so vzrušením majú najvyššiu mieru závislosti charakteristiky: *držanie a stláčanie bežných a mazacích klávesov* a *čas prepisovania jednotlivých viet*.

Algoritmus logistickej regresie spočíva v tom, že najskôr sa snaží odhadnúť parametre modelu (váhy, ktoré sa v procese učenia sa aktualizujú k lepšej presnosti klasifikácie) a neskôr sa násobia s reálnymi hodnotami jednotlivých premenných. Výsledkom sčítania všetkých súčinov váh so vstupmi sú logity (prirodzené logaritmy podielu dvoch pravdepodobností). Z logitov sú následne vypočítavané pravdepodobnosti pre všetky stavy pomocou aktivačnej funkcie Softmax (zovšeobecnená Sigmoidálna funkcia). Trieda s najvyššou pravdepodobnosťou sa stáva predikovanou triedou (Polamuri, 2017).

V Pythone sme model multinominálnej logistickej regresie vytvorili pomocou triedy **LogisticRegression** a ako parametre do modelu sme určili, že pôjde o multinominálny typ triedneho rozdelenia a ako optimalizačný algoritmus na odhad parametrov sme použili *Limited-memory BFGS*. Pomocou metódy *fit* sme klasifikátor natrénovali na tréningových dátach a na vstupných testovacích dátach sme pomocou metódy *predict* model otestovali. Pri viacerých spusteniach pozorujeme, že percentuálna miera presnosti klasifikátora nie je vždy rovnaká. Priemerne dosahujeme (pre dataset 2) klasifikovaním valencie presnosť 78,57% (accuracy), 79,21% (F1score) a klasifikovaním vzrušenia dosahujeme 85,71% (accuracy), 85,54% (F1 score). Tento jav je spôsobený tým, že pri delení dát na tréningové a testovacie sa dáta delia náhodne v určenom pomere a pri každom spustení sa dáta rozdelia inak. Práve preto, ak by sme chceli zabezpečiť, aby sa jednotlivé záznamy objavili aj v tréningovom a v testovacom súbore, musíme použiť metódu k-násobnej krížovej validácie.

K-násobná krížová validácia funguje na princípe, že sa celá vstupná množina dát rozdelí na *k* podmnožín. Jedna podmnožina slúži na testovanie, ostatné podmnožiny slúžia ako tréningové množiny. Klasifikátor natrénuje model na tréningovej množine a pomocou testovacej množiny sa testuje presnosť modelu. Tento proces sa k-krát opakuje, zakaždým však s inou testovacou a tréningovou množinou. My sme použili Stratifikovanú k-násobnú krížovú validáciu (inicializovaním triedy *StratifiedKFold*), ktorá navyše využíva proporčné zastúpenie jednotlivých tried vo výsledných tréningových a testovacích množinách. Použili sme 3-násobnú krížovú validáciu, čiže 67% bude tréningových a 33% bude testovacích záznamov, pričom výsledky všetkých troch validácií sme spriemerovali do jednej presnosti pre obe metriky (accuracy score a F1 score) (Scikit-learn, 2011).

```
model = LogisticRegression(multi_class='multinomial', solver='lbfgs')
skf = StratifiedKFold(n_splits = 3, shuffle = True)
for train_index, test_index in skf.split(X, Y):
    x_train_fold, x_test_fold = X[train_index], X[test_index]
    y_train_fold, y_test_fold = Y[train_index], Y[test_index]
    model.fit(x_train_fold, y_train_fold)
    y_pred_fold = model.predict(x_test_fold)
```

Obrázok 28 Ukážka kódu stratifikovanej krížovej validácie

Vyhodnotenie presnosti klasifikácie modelu Logistickej regresie metódou stratifikovanej 3-násobnej krížovej validácie uvádzame v Tabuľke 7 pre premennú valencie aj pre vzrušenia (accuracy score aj F1 score).

Tabuľka 7 Vyhodnotenie presnosti klasifikačného modelu Logistickej regresie

Logistická regresia (%)		Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6
VALENCIA	ACC	82,83	78,79	74,49	80,59	76,85	73,33
	F1	78,32	77,28	70,72	79,84	73,97	67,59
VZRUŠENIE	ACC	85,35	84,85	74,24	83,15	81,02	83,33
	F1	84,01	84,75	72,95	81,17	72,63	74,23

4.4.3 Naive Bayes

Ako sme už spomínali v analýze, Naivný Bayesov klasifikátor používa Bayesovu vetu na vypočítanie pravdepodobnosti, že konkrétny záznam patrí konkrétnej triede. Pomenovanie má naivný preto, pretože predpokladá, že všetky vstupné premenné (prediktory) sú navzájom nezávislé, aj keď vo väčšine prípadov tomu tak nie je. Prostredníctvom už spomínaného Vzorca 2 z prvej kapitoly, vie tento algoritmus predikovať nové záznamy do konkrétnej triedy (Gandhi, 2018).

Tretí dátový súbor pozostáva z charakteristík správania sa pri používaní myši v experimente 3. Obsahuje 35 záznamov a 24 stĺpcov (22 vstupných premenných). Počet záznamov po rozdelení pre premennú valencie je v pomere 9:14:12 pre negatívnu, neutrálnu a pozitívnu valenciu a pre nízke, stredné a vysoké vzrušenie je pomer záznamov 7:13:15. S valenciou aj so vzrušením najviac korelujú tieto vstupné charakteristiky: *úspešnosť a čas, za ktorý bol experiment dokončený a pauza medzi kliknutiami*. A navyše so vzrušením má ešte podstatnú mieru závislosti aj charakteristika *rýchlosť pohybu kurzora myši*.

Na vytvorenie modelu v Pythone sme použili triedu **GaussianNB** z knižnice Sklearn, pretože tento typ sa používa v prípade, že vstupné premenné obsahujú spojité hodnoty. Metódou *fit* dodáme danému modelu vstupné a cieľové tréningové dáta a metódou *predict* a vstupnými testovacími dátami predpovedáme triedy jednotlivých záznamov.

```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train, Y_train)
Y_pred = model.predict(X_test)
```

Obrázok 29 Vytvorenie, trénovanie a testovanie modelu Naive Bayes

Priemerne dosahujeme, datasetom 3, klasifikovaním valencie presnosť 71,43% (accuracy), 70,90% (F1 score) a klasifikovaním vzrušenia dosahujeme 78,57% (accuracy), 70,77% (F1 score). V nasledujúcej tabuľke sú zobrazené výsledky z 3-násobnej krížovej validácie pre všetky datasety. Už na prvý pohľad je možné vidieť, že tento klasifikátor nie je vhodný pre náš typ problému a aj výsledky klasifikácie emócií týmto modelom sú oproti predošlým algoritmom strojového učenia nedostačujúce.

Tabuľka 8 Vyhodnotenie presnosti klasifikačného modelu Naive Bayes

Naive Bayes (%)		Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6
VALENCIA	ACC	74,49	66,67	73,99	73,08	69,91	70,00
	F1	72,26	66,04	72,48	68,20	65,69	69,21
VZRUŠENIE	ACC	77,27	69,70	71,21	75,82	73,61	76,67
	F1	74,45	65,67	63,66	70,53	67,05	68,87

4.4.4 Neurónová sieť

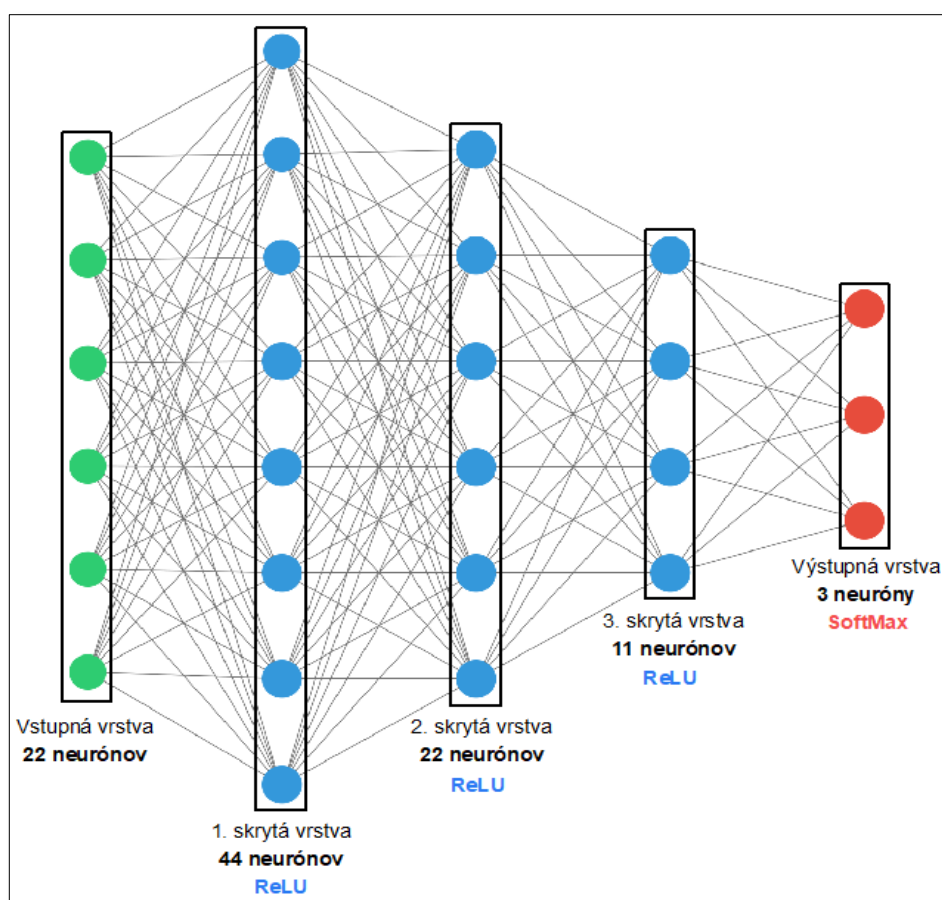
Na klasifikovanie emocionálneho stavu sme si tentokrát vybrali algoritmus strojového učenia model Neurónovej siete, konkrétne sekvenčný model hlbokkej Neurónovej siete **knižnice Keras**. Keras je vysoko výkonná a ľahko použiteľná knižnica. Obsahuje množstvo implementácií bežne používaných stavebných blokov neurónových sietí ako sú vrstvy, stratové a chybové funkcie, aktivačné funkcie, optimalizátory a množstvo ďalších nástrojov na zjednodušenie zápisu modelu neurónových sietí (Keras, 2015). Tentokrát sa bližšie pozrieme na štvrtý dataset z experimentu 4. Obsahuje 41 záznamov (čo je pre neurónovú sieť primálo) a inak obsahuje tie isté charakteristiky pri práci s myšou počítača ako tretí dátový súbor. Po rozdelení záznamov podľa valencie dostaneme počet záznamov v pomere 13:17:11 pre negatívnu, neutrálnu a pozitívnu valenciu a rozdelenie podľa vzrušenia v pomere 9:13:19 pre nízke, stredné a vysoké vzrušenie. Rozdelenie podľa premennej vzrušenia nám vytvorilo viac nevyvážený dataset, ako po rozdelení podľa premennej valencie. S premennou valencie najviac korelujú vstupné charakteristiky: *frekvencia klikania*, *pauza medzi kliknutiami* a *rýchlosť pohybu kurzora myši*. A s premennou vzrušenia majú najvyššiu mieru závislosti charakteristiky: *pauza medzi kliknutiami* a *držanie tlačidla myši*.

Neurónová sieť pozostáva z neurónov, ktoré sú zatriedené do vrstiev. Do vstupnej vrstvy vstupujú vstupné premenné z nášho dátového súboru, preto sa počet neurónov vo vstupnej vrstve musí rovnať počtu vstupných premenných. Každý neurón v sieti je prepojený so všetkými neurónmi nasledujúcej vrstvy, toto prepojenie sa nazýva aj váha. Hodnota každého ďalšieho neurónu v skrytých vrstvách alebo aj vo výstupnej vrstve je tvorená súčtom všetkých súčinov hodnôt neurónov a prislúchajúcich váh predchádzajúcej vrstvy a následne je hodnota daného neurónu upravená aktivačnou funkciou, ktorá je zvyčajne rovnaká pre všetky neuróny danej vrstvy (výpočet je zobrazený vo Vzorci 3 v prvej kapitole). Táto technika je známa ako **dopredné šírenie vstupného signálu**. Výstupná vrstva obsahuje vždy taký počet neurónov, aký je počet tried cieľovej premennej. U nás bude výstupná vrstva obsahovať 3 neuróny pre všetky datasety. Ešte pred tým musíme v rámci predspracovania cieľovú premennú zbinarizovať, čiže previesť celé čísla na matice binárnych tried. Trieda označená pôvodne ako 0 je prevedená do podoby [1, 0, 0], trieda 1 je teraz [0, 1, 0] a analogicky trieda 2 je v podobe [0, 0, 1]. Túto úpravu sme museli urobiť, pretože vo výstupnej vrstve sme ako aktivačnú funkciu nastavili Softmax, ktorá normalizuje výstup siete tak, že vracia pravdepodobnosti pre jednotlivé klasifikačné triedy na základe hodnôt neurónov. Proces učenia sa funguje tak, že v každej iterácii (epoche) sú jednotlivé pravdepodobnosti získané výstupnou vrstvou siete porovnávané so skutočnými výsledkami a vypočíta sa chyba pomocou stratovej funkcie. Cieľom neurónovej siete je nastaviť hodnoty váh tak, aby bola chyba čo najmenšia. To sa dá doceliť pomocou metódy **spätného šírenia chyby (backpropagation)**, ktorá je založená na zmene hodnôt váh (spojení) medzi jednotlivými neurónmi na základe získanej chyby a po úprave váh sa v novej iterácii proces opakuje, až pokiaľ sa chyba ďalej nezmenšuje (Keras, 2015; Great Learning, 2020).

V Pythone sme model sekvenčného modelu hlbkej neurónovej siete (zobrazené na Obrázku 31) inicializovali zavolaním triedy *Sequential*. Pomocou metódy *add* sme pridali jednotlivé vrstvy (*Dense*) do modelu. Pre všetky datasety pre použili 1 vstupnú, 3 skryté a 1 výstupnú vrstvu, pričom pre neuróny vo vstupnej a v skrytých vrstvách sme nastavili aktivačnú funkciu **ReLU** ($f(x) = \max(0, x)$) a pre výstupnú vrstvu sme nastavili aktivačnú funkciu **Softmax**. Počet neurónov vo vrstvách závisí od konkrétneho dátového súboru, preto pre prehľadnosť uvádzame počty neurónov jednotlivých vrstiev pre všetky datasety v Tabuľke 9 a vzhľad Neurónovej siete zobrazujeme na Obrázku 30.

Tabuľka 9 Počty neurónov v jednotlivých vrstvách pre všetky datasety

	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6
Vstupná vrstva	37	37	22	22	27	22
Skrytá vrstva (1.)	74	74	44	44	54	44
Skrytá vrstva (2.)	37	37	22	22	27	22
Skrytá vrstva (3.)	18	18	11	11	11	11
Výstupná vrstva	3	3	3	3	3	3
Počet epoch	50	50	80	80	80	80



Obrázok 30 Vzhľad Neurónovej siete pre dataset 3,4 a 6

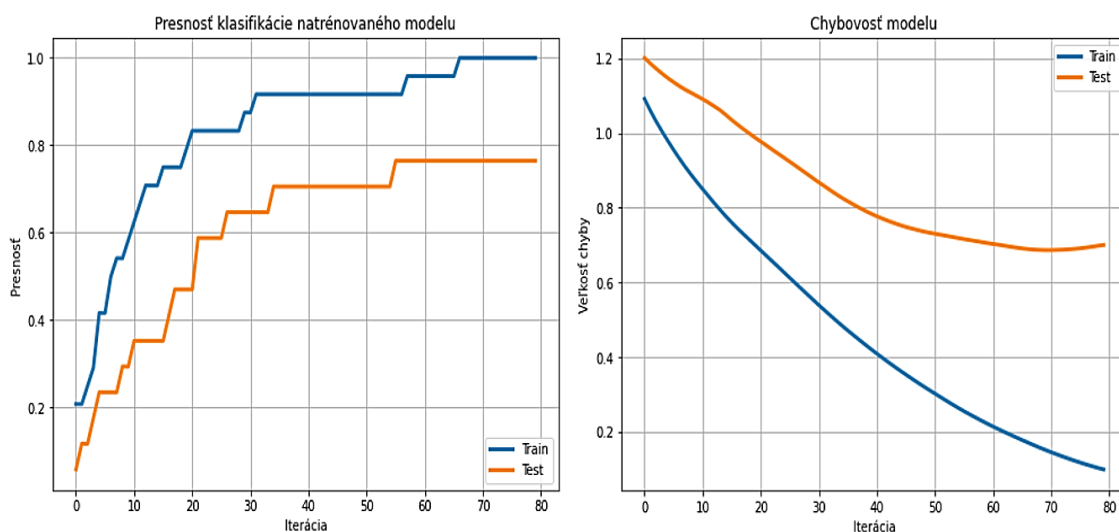
Po vytvorení vrstiev sme model skompilovali pomocou *compile* metódy, v ktorej sme určili stratovú funkciu *categorical_crossentropy* (stratová funkcia, ktorá vypočíta chybu pre viacriednu klasifikáciu), optimalizér *adam* (výpočtovo najefektívnejší optimalizér, ktorý sa snaží znížiť stratovú funkciu, určuje aj rýchlosť učenia sa) a metriku sme určili *accuracy* (aby sa model trénoval k lepšej presnosti). Skompilovaný model sme mohli konečne natrénovať. Trénovali sme ho pomocou metódy *fit* a pomocou parametrov vstupných a cieľových tréningových dát (*X_train*, *Y_train*), ako

validačné dáta (testovacie) sme zadefinovali vstupné a cieľové testovacie dáta (X_test, Y_test) a počet iterácií (epoch) sme nastavili v prípade prvých dvoch datasetov na 50 a v ostatných až na 80. Ukážka kódu nastavenia počtu neurónov, vrstiev a iterácií pre štvrtý dátový súbor je zobrazená na Obrázku 31.

```
model = Sequential()
model.add(Dense(44, input_dim=22, activation="relu", name="Skryta_vrstva_1"))
model.add(Dense(22, activation="relu", name="Skryta_vrstva_2"))
model.add(Dense(11, activation="relu", name="Skryta_vrstva_3"))
model.add(Dense(3, activation="softmax", name="Vystupna_vrstva"))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
output = model.fit(X_train, Y_train, validation_data = (X_test,Y_test), epochs=80)
Y_predikcia = model.predict(X_test)
```

Obrázok 31 Vytvorenie, tréningovanie a testovanie modelu Neurónovej siete

Výhoda použitia knižnice Keras pri modeli Neurónovej siete je, že pri tréningovaní po každej iterácii vypisuje chybu a presnosť na tréningových dátach (loss, accuracy) a chybu aj presnosť na testovacích dátach (val_loss, val_accuracy), keď validuje model. Na Grafe 2 vizualizujeme presnosť klasifikácie a chybovosť modelu pre premennú valencie datasetu 4 v jednotlivých iteráciách, kde modrá krivka je pre tréningové dáta a oranžová je pre testovacie dáta.



Graf 2 Presnosť klasifikácie a chybovosť natrénovaného modelu Neurónovej siete

Priemerne dosahujeme (datasetom 4) klasifikovaním valencie presnosť 76,47% (accuracy), 73,49% (F1 score) a klasifikovaním vzrušenia dosahujeme 88,24% (accuracy), 88,23% (F1 score). V nasledujúcej tabuľke uvádzame výsledky z 3-násobnej krížovej validácie pre všetky datasety. Neurónová sieť pracuje zatiaľ najlepšie pre prvé 3 datasety v porovnaní s ostatnými klasifikátormi.

Tabuľka 10 Vyhodnotenie presnosti klasifikačného modelu Neurónovej siete

Neurónová sieť (%)		Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6
VALENCIA	ACC	85,61	81,82	76,77	77,66	76,85	73,33
	F1	81,63	82,83	72,99	76,81	76,61	73,34
VZRUŠENIE	ACC	86,11	87,88	79,80	85,71	80,56	80,00
	F1	84,70	86,20	70,45	84,50	76,08	71,70

4.4.5 Random Forest

Ďalším klasifikačným algoritmom strojového učenia, ktorý sme si vybrali na klasifikáciu emócií je Random Forest. V analýze sme skôr opisovali klasifikátor Rozhodovací strom a Random Forest sme spomenuli ako algoritmus, ktorý kombinuje viacero Rozhodovacích stromov, čím dosahuje lepšiu klasifikačnú presnosť a nie je náchylný na preučenie sa.

Piaty dátový súbor sme získali z experimentu 1 mobilnej aplikácie, kde úlohou participantov bolo prepísať text prostredníctvom softvérovej klávesnice. Tento dataset obsahuje rovnaké charakteristiky ako datasety 1 a 2, líši sa od nich iba tým, že neobsahuje charakteristiky pri stláčaní a držaní funkčných klávesov (napr. Shift, Ctrl). Celý dataset obsahuje 26 záznamov a pozostáva z 27 vstupných premenných. Po rozdelení záznamov podľa valencie dostaneme počet záznamov v pomere 10:9:7 pre negatívnu, neutrálnu a pozitívnu valenciu a rozdelenie podľa vzrušenia v pomere 6:12:8 pre nízke, stredné a vysoké vzrušenie. S premennou valencie najviac korelujú vstupné charakteristiky: *pauza medzi ťuknutiami*, *frekvencia ťukania* a *doba držania prsta bežných a mazacích klávesov (na obrazovke)*. A s premennou vzrušenia majú najvyššiu mieru závislosti charakteristiky: *čas absolvovania experimentu*, *frekvencia stláčania a doba držania bežných a mazacích klávesov a pauza medzi ťuknutiami*.

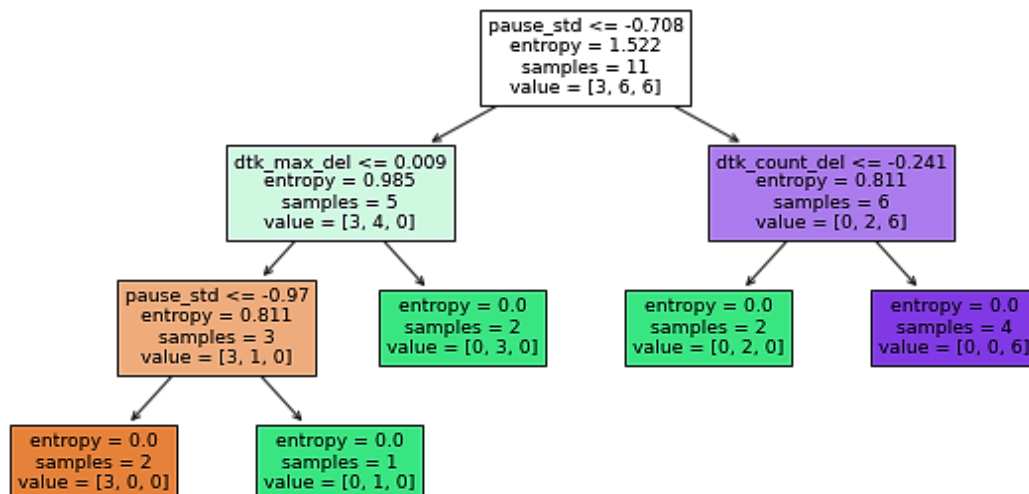
Algoritmus klasifikátora Random Forest je veľmi jednoduchý. V prvom kroku algoritmus vytvorí **nových N-dátových súborov** z pôvodného datasetu (vo výsledku bude N stromov). Vytvára ich tak, že vyberie *náhodne niekoľko vzoriek z tréningovej množiny dát*, pričom vzorky sa môžu opakovať (metóda známa ako bootstrapping) a každý takto novovytvorený dataset obsahuje iba podmnožinu zo všetkých vstupných charakteristík (zabezpečuje sa tak menšia korelácia medzi stromami). V druhom kroku sa pre každý takýto dataset vytvorí Rozhodovací strom. V treťom kroku sa pre jednotlivé testovacie dáta vytvárajú predpovede spriemerovaním predpovedí každého

Rozhodovacieho stromu na týchto dátach. Výsledky dosiahnuté algoritmom sa porovnávajú so skutočnými výsledkami a vypočíta sa presnosť klasifikácie (Koehrsen, 2018).

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=100, criterion="entropy", max_depth=5)
model.fit(X_train, Y_train)
Y_pred = model.predict(X_test)
```

Obrázok 32 Vytvorenie, tréning a testovanie modelu Random Forest

Implementácia algoritmu Random Forest je v Pythone ešte jednoduchšia ako jeho vysvetlenie. Použitím triedy **RandomForestClassifier** vytvoríme model algoritmu. Ako vstupné parametre sme zadefinovali počet stromov (`n_estimators`), ktorý sme nastavili na 100, kritérium, podľa ktorého sa vyberajú charakteristiky pre optimálne rozdelenie dát pri vytváraní stromov sme nastavili na **entropy** a určili sme maximálnu hĺbku stromu (`max_depth`), ktorú sme nastavili na 5. Pomocou metódy `fit` sme model úspešne natrénovali na tréningových dátach a pomocou metódy `predict` a vstupných testovacích dát sme predikovali výslednú emocionálnu triedu (ukážka kódu je zobrazená na Obrázku 32). Všetky vytvorené stromy si môžeme vizualizovať, jeden takto vytvorený strom pri klasifikácii vzrušenia zobrazujeme na Obrázku 33.



Obrázok 33 Ukážka jedného rozhodovacieho stromu z Random Forest

Pri priemernom spustení dosahujeme (datasetom 5) klasifikovaním valencie presnosť 72,73% (accuracy), 68,52% (F1 score) a klasifikovaním vzrušenia dosahujeme 81,82% (accuracy), 79,19% (F1 score). Výsledky klasifikácie všetkých našich datasetov klasifikátorom Random Forest metódou 3-násobnej krížovej validácie uvádzame v Tabuľke 11. Výsledky dosiahnuté týmto algoritmom sú porovnateľné

s výsledkami Neurónovej siete, dokonca v niektorých prípadoch klasifikuje lepšie a jednoznačne rýchlejšie ako Neurónová sieť.

Tabuľka 11 Vyhodnotenie presnosti klasifikačného modelu Random Forest

Random Forest (%)		Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6
VALENCIA	ACC	83,08	78,79	73,74	78,21	77,31	76,67
	F1	79,45	78,10	73,32	77,27	76,43	74,69
VZRUŠENIE	ACC	85,86	84,85	71,97	85,16	76,85	83,33
	F1	85,93	84,64	65,18	80,14	74,69	72,65

4.4.6 Support Vector Machine

Posledným algoritmom, ktorý sme aplikovali na naše dáta je Support Vector Machine (SVM). Cieľom algoritmu je nájsť nadrovinu (hyperplane), ktorá je schopná rozdeliť n-rozmerný priestor do dvoch kategórií tak, že sa maximalizuje vzdialenosť k bodom v oboch kategóriách. Táto vzdialenosť sa nazýva okraj (margin) a body, ktoré sa dotýkajú tohto okraja, sa nazývajú podporné vektory. Podľa týchto podporných vektorov sa vlastne určuje poloha nadroviny. Najjednoduchší typ tohto algoritmu nepodporuje klasifikáciu viacerých tried, podporuje len binárnu klasifikáciu, čo je pre nás značný problém, pretože naše cieľové premenné nadobúdajú až 3 stavy. Pre dosiahnutie viactriednej klasifikácie sa použije rovnaký princíp rozdelenia celého problému na viaceré binárne klasifikačné problémy. Striedajú sa tu dva prístupy nájdenia vhodných nadrovín. V prvom prístupe potrebujeme nadrovinu na oddelenie medzi každé dve triedy, pričom si nevšímame body tretej triedy. V druhom prístupe potrebujeme nadrovinu na oddelenie medzi triedou a všetkými ostatnými triedami naraz. Využitím oboch prístupov je možné nájsť nadroviny, ktoré dokážu vykonať viactriednu klasifikáciu (Baeldung.com, 2020).

Šiesty a zároveň posledný dátový súbor sme získali z experimentu 2 mobilnej aplikácie, ktorého cieľom bolo vyriešiť hlavolam ťukaním prsta po obrazovke. Tento dataset obsahuje rovnaké charakteristiky ako datasety 3 a 4 (odlišuje sa iba pomenovaním niektorých charakteristík). Celý dataset obsahuje 30 záznamov a pozostáva z 22 vstupných premenných. Po rozdelení záznamov podľa valencie dostaneme počet záznamov v pomere 9:8:13 pre negatívnu, neutrálnu a pozitívnu valenciu a rozdelenie podľa vzrušenia v pomere 5:11:14 pre nízke, stredné a vysoké

vzrušenie. S premennou valencie najviac korelujú vstupné charakteristiky: *pauza medzi ťuknutiami* a *rýchlosť pohybu prsta* (merané ako vzdialenosť medzi dvoma ťuknutiami v čase po obrazovke). A s premennou vzrušenia majú najvyššiu mieru závislosti charakteristiky: *rýchlosť pohybu prsta*, *frekvencia ťukania* a *doba držania prsta na obrazovke*.

Implementácia klasifikátora SVM v Pythone je nasledovná. Vytvoríme si model inicializovaním triedy **SVC** (ako *Support Vector Classification*) knižnice Sklearn. Ako vstupný parameter sme nastavili *kernel* na funkciu Radial Basis Function (RBF), ktorá určuje plynulosť a efektívnosť oddelenia tried. Následne sme natrénovali model klasifikátora na tréningových dátach pomocou metódy *fit* a pomocou metódy *predict* a vstupných testovacích dát sme predikovali cieľové dáta. Porovnaním výsledkov modelu s reálnymi výsledkami získame klasifikačnú presnosť. Pri priemernom spustení dosahujeme (datasetom 6) klasifikovaním valencie presnosť 75% (accuracy), 74,53% (F1 score) a klasifikovaním vzrušenia dosahujeme 91,67% (accuracy), 86,32% (F1 score).

```
from sklearn.svm import SVC
model = SVC(kernel='rbf')
model.fit(X_train, Y_train)
Y_pred = model.predict(X_test)
```

Obrázok 34 Vytvorenie, trénovanie a testovanie modelu SVM

Výsledky klasifikácie všetkých našich datasetov klasifikátorom Support Vector Machine metódou 3-násobnej krížovej validácie uvádzame v Tabuľke 12. Aj keď prvotná presnosť klasifikácie bola vyššia, overením klasifikátora 3-násobnou krížovou validáciou nám presnosť mierne klesla.

Tabuľka 12 Vyhodnotenie presnosti klasifikačného modelu SVM

Support Vector Machine (%)		Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6
VALENCIA	ACC	80,05	75,76	74,24	75,46	76,85	73,33
	F1	74,51	75,82	72,73	72,78	71,70	72,59
VZRUŠENIE	ACC	82,83	84,85	74,49	82,78	77,31	83,33
	F1	81,23	83,10	60,12	70,37	74,38	72,85

5 VÝSLEDKY PRÁCE A DISKUSIA

V našej diplomovej práci sme si za hlavný cieľ stanovili využiť prvky umelej inteligencie pri klasifikácii emócií používateľa na základe dát správania sa pri práci s počítačom a so smartfónom. To sa nám aj podarilo a pomocou viacerých klasifikačných algoritmov strojového učenia pri učení s učiteľom sme dokázali úspešne klasifikovať jednotlivé emocionálne stavy na dimenziách valencie a vzrušenia. Všetky výsledky sme uvádzali už v predošlej kapitole pre konkrétne klasifikačné algoritmy. Teraz si vyhodnotíme už len výsledky tých algoritmov, ktoré dosahovali najvyššiu presnosť klasifikácie pre konkrétne datasety na oboch emocionálnych dimenziách, metódou 3-násobnej krížovej validácie (vizuálne sme najúspešnejšie klasifikátory uviedli v Tabuľke 13).

Tabuľka 13 Najúspešnejšie klasifikačné algoritmy

Najúspešnejšie klasifikátory	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6
VALENCIA	NS	NS	NS, RF, K-NN	LR	RF, NS, LR, SVM	RF
VZRUŠENIE	NS, RF	NS	NS, LR	NS, RF, LR	LR, NS, K-NN	K-NN, RF, LR, SVM

Začneme datasetom 1, kde sa ukázalo, že najlepším klasifikátorom je *Neurónová sieť* (NS) s dosiahnutou presnosťou 85,61% pre valenciu a 86,11% pre vzrušenie. Algoritmus *Random Forest* (RF) bol iba o niečo slabší, no v prípade F1 score dosahoval presnosť 85,93%. *Neurónová sieť* dominovala aj v druhom datasete, kde presnosť klasifikácie pre valenciu dosahovala 81,82% a pre vzrušenie až **87,88%**, čo je zároveň aj **najvyššia dosiahnutá klasifikačná presnosť** všetkých našich modelov. Pre dataset 3 nemôžeme jednoznačne povedať, ktorý klasifikátor je najlepší, pretože presnosti viacerých klasifikačných algoritmov boli približne rovnaké. Algoritmy *Neurónovej siete*, *Random Forest* a *K-Nearest Neighbors* (K-NN) klasifikujú valenciu s približne 75% presnosťou a v prípade vzrušenia najlepšie klasifikátory sú *Neurónová sieť* (79,80%) a *Logistická regresia* (LR) (74,24%). Model klasifikátora *Logistickej regresie* vo štvrtom dátovom súbore klasifikuje najlepšie spomedzi ostatných modelov s presnosťou 80,59% pre valenciu a modely *Neurónovej siete*, *Random Forest* a *Logistickej regresie* dokázali klasifikovať vzrušenie s presnosťou vyššou ako 83%.

Úroveň valencie v piatom dátovom súbore správne klasifikovali algoritmy *Random Forest*, *Neurónová sieť*, *Logistická regresia* a *Support Vector Machine* (SVM) s presnosťou takmer 77%. Úroveň vzrušenia dosiahla opäť vyššiu presnosť v porovnaní s valenciou, približne 80%, algoritmami *Logistická regresia*, *Neurónové siete* a *K-Nearest Neighbors*. A výsledky pre posledný, šiesty dataset naznačujú, že najlepším algoritmom pre klasifikáciu valencie je *Random Forest* s presnosťou 76,67% a úroveň vzrušenia klasifikovali algoritmy *K-Nearest Neighbors*, *Random Forest*, *Logistická regresia* a *Support Vector Machine* so zhodnou presnosťou 83,33%. Naive Bayes je ako jediný klasifikačný algoritmus v našej práci nedostačujúcim klasifikátorom, pretože takmer vo všetkých vytvorených modeloch dosahoval najnižšiu presnosť v porovnaní s inými klasifikátormi.

Po vyhodnotení našich výsledkov sa opláca porovnať ich s výsledkami autorov, ktorých sme spomenuli už v analýze súčasného stavu v prvej kapitole. Porovnanie však nebude dokonalé, pretože tak ako aj títo autori, tak aj v našej práci sme používali vlastné metódy merania behaviorálnych charakteristík (experimenty), čím sme vytvorili vlastné dátové súbory, keďže sme nemohli žiadny verejne dostupný súbor nájsť. Ak porovnáme prácu **Nahina** (2014) s našou prácou, zistíme, že tento autor vytvoril dve úlohy, kde v prvej len snímal dynamiku stláčania klávesov na klávesnici pri prepise textu a pri druhej úlohe prepisovaný text aj analyzoval. Nás zaujíma hlavne tá prvá úloha. Autor ňou priemerne dosiahol presnosť klasifikácie 84,5% pre emóciu radosti a najnižšiu presnosť 65,5% pre smútok, pričom rozlišoval až sedem konkrétnych emócií. Je dôležité spomenúť, že autor tvoril zvlášť modely pre každého z používateľov, pričom my sme sa snažili vytvoriť spoločný model pre všetkých používateľov.

Ak porovnáme prácu **Wolfa** (2017), tak zistíme, že máme dva podobné experimenty, pretože jeho prvý experiment sa týkal prepisu dvoch až troch viet za časový limit a jeho druhý experiment sa týkal poklikania 30-tich očíslovaných kruhov pomocou myši počítača. Autor sa zameral iba na valenciu a rozlišoval stavy emócií pozitívne, negatívne alebo neutrálne. Takisto ako my, tak aj tento autor robil jeden model pre všetkých používateľov, no však jeho výsledky sa od tých našich značne odlišujú. Pri prepise textu dosiahol najvyššiu klasifikačnú presnosť len 52,54% algoritmom *Rozhodovacieho stromu* a pre hru, kde mali používatelia poklikať 30 kruhov dosiahol presnosť klasifikovania 70% *Rozhodovacím stromom* a 44,44% presnosť pri použití klasifikátora *Random Forest*.

Práca **Ghosha** (2017) je veľmi podobná nášmu prvému experimentu v mobilnej aplikácii. Autor vytvoril aplikáciu pozostávajúcu len zo softvérovej klávesnice, ktorá slúžila ako vstup do ktorejkoľvek inej aplikácie pre zadávanie textu. Používatelia museli túto klávesnicu nastaviť ako predvolenú pre všetky aplikácie. Po napísaní dostatku textu, mali používatelia ohodnotiť svoj emocionálny stav ako: smutný, šťastný, vystresovaný alebo zrelaxovaný. Dôležité je, že po zozbieraní dát vytvoril autor klasifikačný model pre každého používateľa zvlášť. Klasifikátorom *Random Forest* dosahoval najvyššiu priemernú presnosť až 73% a maximálnu presnosť pre jedného z používateľov dosahovala úroveň 93%. Naše výsledky sú celkom porovnateľné s výsledkami tohto autora, keďže naša dosiahnutá presnosť klasifikácie sa pohybovala od 77% do 80% pre dataset 5 (prepis textu na mobilnom zariadení).

Ako možným vylepšením alebo rozšírením našej práce by mohlo byť vytvorenie modelov pre každého používateľa zvlášť tak, ako to robí väčšina autorov, čo by znamenalo, že používatelia by museli danú úlohu niekoľkokrát opakovať, čo by mohlo spôsobiť problém vo vyvolaní emócií, keďže používateľ už druhýkrát nebude na podnet reagovať rovnako, skôr neutrálne. Alebo práve naopak, zlepšila by sa klasifikačná presnosť, ak by sme model postavili na povahe používateľa a nezovšeobecňovali by sme správanie všetkých na správanie jednotlivca.

Ďalšie vylepšenie práce, týkajúce sa skôr mobilnej aplikácie zahŕňa použitie zabudovaného senzora akcelerometer, ktorý by nám detegoval pohyb, náklon, ale aj orientáciu mobilného zariadenia. Tento senzor by sme mohli využiť, podobne ako súradnice kurzora na obrazovke počítača, k vypočítaniu rýchlosti pohybu smartfónu, či napríklad k zmene uhlovej rýchlosti otočenia smartfónu.

A posledným vylepšením, ktoré nám napadlo, je prepojiť už existujúce riešenia predikcie emócií pomocou kamery, mikrofónu a/alebo telesných senzorov (napr. EKG senzor) spolu s naším riešením klasifikácie emócií pomocou charakteristík správania sa pri práci s klávesnicou a myšou počítača, či dokonca smartfónom.

ZÁVER

V súčasnom svete sa prvky umelej inteligencie vyskytujú už azda v každej oblasti moderného života človeka. V poslednom desaťročí sa začali prvky umelej inteligencie, konkrétne modely strojového učenia, využívať aj pri klasifikácii, či dokonca predikcii emocionálneho stavu ľudí. Existujú viaceré riešenia, ktoré klasifikujú emócie používateľov na základe mimiky tváre, tónu hlasu, či dokonca na základe iných fyziologických prejavov. Cieľom našej diplomovej práce bolo klasifikovať emocionálny stav používateľa prostredníctvom jeho behaviorálnych charakteristík, získaných pri práci s počítačom a smartfónom, s využitím prvkov umelej inteligencie. V teoretickej časti práce sme sa venovali súčasnému pohľadu na umelú inteligenciu s prihliadnutím na najpoužívanějšíe klasifikačné algoritmy strojového učenia sa s učiteľom. Analyzovali sme aj práce autorov, ktorí sa venovali klasifikácii emócií podľa vzorov správania sa a využili pri tom algoritmy strojového učenia.

V aplikačnej časti práce sme vytvorili desktopovú a mobilnú aplikáciu, ktoré na pozadí vypočítavali charakteristiky správania sa. Pre desktopovú aplikáciu sme vymysleli a implementovali až 4 experimenty. Dva z nich boli zamerané na prepis textu pomocou klávesnice a zvyšné dva boli zamerané na poklikanie objektov pomocou myši počítača k splneniu úlohy daného experimentu. Mobilná aplikácia obsahovala 2 experimenty, z toho cieľom prvého bolo prepísať text pomocou softvérovej klávesnice a cieľom druhého bolo pomocou ťukania prsta splniť danú úlohu. Po zozbieraní dostatočného počtu dát sme dáta analyzovali a predspracovali. Sebahodnotenie používateľov sme z podoby reálnych čísel transformovali do troch kategórií emocionálnych stavov pre obe dimenzie Russelovho kruhového modelu emócií.

V programovacom jazyku Python sme na tieto dáta nasadili až 6 klasifikačných algoritmov strojového učenia sa s učiteľom. Klasifikovali sme zvlášť mieru valencie (negatívna, neutrálna, pozitívna) aj vzrušenia (nízke, stredné, vysoké) metódou 3-násobnej krížovej validácie. Najvyššiu klasifikačnú presnosť, až 87,88%, sme dosiahli klasifikovaním vzrušenia na dátovom súbore z experimentu 2 desktopovej aplikácie pomocou algoritmu *Neurónovej siete*. Z výsledkov klasifikácie valencie všetkých dátových súborov najpresnejšie klasifikuje taktiež klasifikátor *Neurónovej siete* pre prvý dátový súbor s presnosťou 85,61%. Zistili sme, že klasifikátor neurónovej siete je najvhodnejší pre prvé tri dátové súbory, pre zvyšné dátové súbory sa podľa výsledkov javia ako lepšie algoritmy *Random Forest*, *Logistická regresia* a *K-Nearest Neighbors*.

ZOZNAM BIBLIOGRAFICKÝCH ODKAZOV

- ADLER, S. 2019. A Quick Guide to Artificial Intelligence (AI). [online]. 2019. [cit. 2021-08-25]. Dostupné na internete: <<https://www.intelligentautomation.network/decision-ai/news/a-basic-guide-to-ai>>.
- BAELDUNG.COM. 2020. Multiclass Classification Using Support Vector Machines. [online]. 2020. [cit. 2021-09-02]. Dostupné na internete: <<https://www.baeldung.com/cs/svm-multiclass-classification>>.
- BALDUCCI, F. – IMPEDOVO, D. – MACCHIARULO, N. – PIRLO, G. 2020. Affective states recognition through touch dynamics. Multimedia Tools and Applications. doi: 10.1007/s11042-020-09146-4
- BROWNLEE, J. 2019. What is Deep Learning?. [online]. 2019. [cit. 2021-08-27]. Dostupné na internete: <<https://machinelearningmastery.com/what-is-deep-learning/>>.
- BUKOVČANOVÁ, A. 2021. Natural language processing alebo spracovanie prirodzeného jazyka je čoraz podstatnejšie. [online]. 2021. [cit. 2021-08-25]. Dostupné na internete: <<https://www.translata.sk/blog/natural-language-processing-alebo-spracovanie-prirodzeneho-jazyka-je-coraz-podstatnejšie>>.
- CUELOGIC.COM. 2017. The Levenshtein Algorithm. [online]. 2017. [cit. 2021-08-03]. Dostupné na internete: <<https://www.cuelogic.com/blog/the-levenshtein-algorithm>>.
- DRŽÍK, D. 2020. Klasifikácia emocionálneho stavu používateľa na základe jeho behaviorálnej charakteristiky. 47 s. Univerzita Konštantína Filozofa v Nitre FPV UKF KI
- ELLIOT, K. – GRAHAM, J. – YASSIN, Y. – WARD, T. – CALDWELL, J. – ATTIE, T. 2019. A Comparison of Machine Learning Algorithms in Keystroke Dynamics. 2019. International Conference on Computational Science and Computational Intelligence (CSCI), 2019, pp. 127-132, doi: 10.1109/CSCI49370.2019.00028.

- ESCOTT, E. 2017. What are the 3 types of AI? A guide to narrow, general, and super artificial intelligence. [online]. 2017. [cit. 2021-08-22]. Dostupné na internete: <<https://codebots.com/artificial-intelligence/the-3-types-of-ai-is-the-third-even-possible>>.
- EUROPA.EU. 2020. What is artificial intelligence and how is it used?. [online]. 2020. [cit. 2021-08-16]. Dostupné na internete: <<https://www.europarl.europa.eu/news/en/headlines/society/20200827STO85804/what-is-artificial-intelligence-and-how-is-it-used>>.
- EXPERT.AI. 2020. What is Machine Learning? A Definition.. [online]. 2020. [cit. 2021-08-23]. Dostupné na internete: <<https://www.expert.ai/blog/machine-learning-definition/>>.
- GANDHI, R. 2018. Naive Bayes Classifier. [online]. 2018. [cit. 2021-09-02]. Dostupné na internete: <<https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c>>.
- GHOSH, S. – GANGULY, N. – MITRA, B. – DE, P. 2017. Evaluating effectiveness of smartphone typing as an indicator of user emotion. 2017. Seventh International Conference on Affective Computing and Intelligent Interaction (ACII), 2017, pp. 146-151, doi: 10.1109/ACII.2017.8273592.
- GOOGLE. (N.D.). 2021. HttpURLConnection. [online]. 2021. [cit. 2021-07-30]. Dostupné na internete: <<https://developer.android.com/reference/java/net/HttpURLConnection>>.
- GREAT LEARNING. 2020. Types of Neural Networks and Definition of Neural Network. [online]. 2020. [cit. 2021-09-06]. Dostupné na internete: <<https://www.mygreatlearning.com/blog/types-of-neural-networks/>>.
- CHAKRABORTY, M. 2021. Types of artificial intelligence – weak, strong and super ai. [online]. 2021. [cit. 2021-08-22]. Dostupné na internete: <<https://www.analyticsinsight.net/types-of-artificial-intelligence-weak-strong-and-super-ai/>>.
- CHRISTOPHER, A. 2021. K-Nearest Neighbor. [online]. 2021. [cit. 2021-09-02]. Dostupné na internete: <<https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4>>.

- IBM.COM. 2020. What is computer vision?. [online]. 2020. [cit. 2021-08-25].
Dostupné na internete: <<https://www.ibm.com/topics/computer-vision>>.
- JLORDIALES. 2014. Easy to understand Dynamic Programming – Edit distance. [online]. 2014. [cit. 2021-08-03]. Dostupné na internete: <<https://jlordiales.wordpress.com/2014/03/01/dynamic-programming-edit-distance/>>.
- KERAS. 2015. Keras. [online]. 2015. [cit. 2021-11-30]. Dostupné na internete: <<https://keras.io/>>.
- KOŽDOUSKOVÁ, B. 2021. Umělá inteligence, hluboké a strojové učení, v čem je rozdíl. [online]. 2021. [cit. 2021-08-22]. Dostupné na internete: <<https://www.rascasone.com/cs/blog/umela-inteligence-strojove-hluboke-uceni-rozdil>>.
- KOEHRSEN, W. 2018. An Implementation and Explanation of the Random Forest in Python. [online]. 2018. [cit. 2021-12-05]. Dostupné na internete: <<https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76>>.
- KUMAR, CH. 2018. Artificial Intelligence: Definition, Types, Examples, Technologies. [online]. 2018. [cit. 2021-08-22]. Dostupné na internete: <<https://chethankumargn.medium.com/artificial-intelligence-definition-types-examples-technologies-962ea75c7b9b>>.
- MARR, B. 2020. What Is The Difference Between Weak (Narrow) And Strong (General) Artificial Intelligence (AI)?. [online]. 2020. [cit. 2021-08-22]. Dostupné na internete: <<https://bernardmarr.com/what-is-the-difference-between-weak-narrow-and-strong-general-artificial-intelligence-ai/>>.
- MARR, B. 2021. What are the Four Types of AI?. [online]. 2021. [cit. 2021-08-22]. Dostupné na internete: <<https://bernardmarr.com/what-are-the-four-types-of-ai/>>.
- MISHRA, S. 2017. Unsupervised Learning and Data Clustering. [online]. 2017. [cit. 2021-08-26]. Dostupné na internete: <<https://towardsdatascience.com/unsupervised-learning-and-data-clustering-eeecb78b422a>>.

- NAHIN, N. – ALAM, J. – MAHMUD, H. – HASAN, K. 2014. Identifying emotion by keystroke dynamics and text pattern analysis, *Behaviour & Information Technology*, 33:9, 987-996, doi: 10.1080/0144929X.2014.907343
- NAM, E. 2019. Understanding the Levenshtein Distance Equation for Beginners. [online]. 2019. [cit. 2021-08-03]. Dostupné na internete: <<https://medium.com/@ethannam/understanding-the-levenshtein-distance-equation-for-beginners-c4285a5604f0>>.
- NAVLANI, A. 2018. Decision Tree Classification in Python. [online]. 2018. [cit. 2021-09-03]. Dostupné na internete: <<https://www.datacamp.com/community/tutorials/decision-tree-classification-python>>.
- ONGSULEE, P. 2017. Artificial intelligence, machine learning and deep learning, 2017 15th International Conference on ICT and Knowledge Engineering (ICT&KE), 2017, pp. 1-6, doi: 10.1109/ICTKE.2017.8259629.
- OSIŃSKI, B. – BUDEK, K. 2018. What is reinforcement learning? The complete guide. [online]. 2018. [cit. 2021-08-26]. Dostupné na internete: <<https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/>>.
- PENTEL, A. 2015. Employing think-aloud protocol to connect user emotions and mouse movements. [online]. 2015. [cit. 2021-08-04]. Dostupné na internete: <<https://ieeexplore.ieee.org/document/7387970>>.
- POLAMURI, S. 2017. HOW MULTINOMIAL LOGISTIC REGRESSION MODEL WORKS IN MACHINE LEARNING. [online]. 2017. [cit. 2021-09-01]. Dostupné na internete: <<https://dataaspirant.com/multinomial-logistic-regression-model-works-machine-learning/>>.
- PRINCETON.EDU. 2010. 8 Puzzle. [online]. [cit. 2021-08-05]. Dostupné na internete: <<https://www.cs.princeton.edu/courses/archive/spr10/cos226/assignments/8puzzle.html>>.
- RUSSELL, J. 1980. A Circumplex Model of Affect. *Journal of Personality and Social Psychology*. 39. 1161-1178. doi: 10.1037/h0077714.

- SAUREL, S. 2018. Creating an In-App Keyboard for your Android Apps. [online]. 2018. [cit. 2021-08-06]. Dostupné na internete: <<https://ssaurel.medium.com/creating-an-in-app-keyboard-for-your-android-apps-a0a6c6e1e289>>.
- SAUREL, S. 2019. Developing a 15 Puzzle — Game of Fifteen in Java. [online]. 2019. [cit. 2021-08-06]. Dostupné na internete: <<https://ssaurel.medium.com/developing-a-15-puzzle-game-of-fifteen-in-java-dfe1359cc6e3>>.
- SCIKIT-LEARN. 2011. Machine Learning in Python. [online]. 2011. [cit. 2021-11-24]. Dostupné na internete: <<https://scikit-learn.org/>>.
- SEUFERT, E. B. 2014. Quantitative Methods for Product Management. Freemium Economics, 47–82. doi: 10.1016/b978-0-12-416690-5.00003-8
- SHAPSOUGH, S. – HESHAM, A. – ELKHORAZATY, Y. – ZUALKERNAN, I. A. – ALOUL, F. 2016. Emotion recognition using mobile phones. 2016. IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom), 2016, pp. 1-6, doi: 10.1109/HealthCom.2016.7749470.
- STATCOUNTER.COM. 2021. Mobile Operating System Market Share Worldwide. [online]. 2021. [cit. 2021-08-05]. Dostupné na internete: <<https://gs.statcounter.com/os-market-share/mobile/worldwide>>.
- WOLF, M. 2017. Model používateľa pre identifikáciu jeho emocionálneho stavu. 86 s. Slovenská technická univ. v Bratislave FIIT UISI (FIIT)
- ZWASS, V. 2015. Expert system. [online]. 2015. [cit. 2021-08-25]. Dostupné na internete: <<https://www.britannica.com/technology/expert-system>>.

ZOZNAM PRÍLOH

Prílohy sú dostupné v súkromnom GitHub repozitári:

- <https://github.com/daviddrzik/diplomovka>

Priložené prílohy obsahujú:

- diplomovú prácu vo formáte PDF,
- zdrojový kód a spustiteľný exe súbor aplikácie Emotnizer desktop,
- zdrojový kód a inštalačný apk súbor aplikácie Emotnizer mobile,
- zdrojový kód webovej stránky <https://emotnizer.wz.sk/>,
- exportované (spracované i nespracované) dátové súbory v xlsx formáte,
- klasifikačné algoritmy strojového učenia pre každý z datasetov v ipynb a html formáte prostredia Jupyter notebook.