FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Software pro odhad věku na základě snímku pánevní kosti
Student:	Natália Pohanková
Vedoucí:	Ing. Michal Štepanovský, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2021/2022

Pokyny pro vypracování

Cílem BP je vytvořit jednoduchý software, který umožní odhadnout věk jedince na základě skenu význačné plochy pánevní kosti. V antropologii je z kosterních pozůstatků dospělého člověka stav pánevní kosti velmi častým indikátorem věku dožití. Na základě povrchového 3D skenu lze pozorovat tvar a opotřebení význačné plochy pánevní kosti a odhadnout tak věk jedince. Pokyny pro vypracování:

1. Seznamte se s formátem STL, ve kterém jsou uložena vstupní data.
2. Vytvořte data-mining model, pomoci kterého lze odhadnout věk jedince.
3. Navržený model implementujte.
4. Analyzujte výsledky / schopnost odhadu věku jedince.

Jednotlivé kroky a obsah práce konzultujte s vedoucím BP.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 5. listopadu 2020 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalárska práca

Software na odhad veku na základe snímky panvovej kosti

Natália Pohanková

Katedra softwarového inženýrství
Vedúci práce: Ing. Michal Štepanovský, Ph.D.

7. mája 2021

Poďakovanie

Touto cestou by som sa chcela najprv poďakovať vedúcemu mojej bakalárskej práce Ing. Michalovi Štepanovskému, Ph.D. za usmernenie, priateľský prístup a cenné rady. Veľká vďaka patrí aj mojej rodine a priateľom, ktorí ma podporovali počas celej doby môjho štúdia.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 7. mája 2021 . . . . . .

Odkaz na túto prácu

Pohanková, Natália. *Software na odhad veku na základe snímky panvovej kosti.* Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

Abstrakt

Táto bakalárska práca popisuje proces tvorby softvéru slúžiaceho na odhad veku dospelého jedinca na základe skenu jeho panvovej kosti. Strojové učenie má v súčasnosti rozsiahle uplatnenie v rôznych oblastiach vrátane v oblastiach ako sú forenzná antropológia, paleontológia či história. Používanie tohto softvéru ich časť práce zameranej na určenie veku jedincov značne uľahčí. V práci sa zameriavam na analýzu vstupných dát, návrh rôznych modelov strojového učenia, ktoré slúžia na predikciu veku jedincov, ich následnú validáciu a napokon aj na implementáciu a testovanie používateľského rozhrania.

Klíčová slova panvová kosť, softvér, strojové učenie, odhad veku, python

Abstract

This bachelor thesis describes the process of development of a software application used to estimate the age of an adult based on a scan of his pelvis bone. Machine learning algorithms are currently being widely used in various fields including in forensic anthropology, paleontology and history. Using this software will make their part of the job focusing on age estimation significantly easier. In this thesis I focus on the analysis of input data, design of multiple machine learning models, which are used to estimate the age of individuals, their subsequent validation and lastly on the implementation and testing of the user interface.

Keywords pelvic bone, software, machine learning, age estimation, python

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Medzi problémy modernej doby nepochybne patria dáta. Ich zber, spracovanie či následná analýza a vyvodenie záverov. Pokroky, ktoré priniesol rozvoj umelej inteligencie, teórie strojového učenia a neurónových sietí môžeme veľmi efektívne využiť na automatizáciu v mnohých oblastiach každodenného života. Viaceré odvetvia si osvojili používanie strojového učenia, alebo nejakého druhu umelej inteligencie, za účelom zefektívniť pracovné procesy a zredukovať personálne náklady.

Popularita delegovania práce neobišla ani zamerania akými sú napríklad archeológia, história, forenzná antropológia alebo paleontológia. Počas posledných rokov v týchto odvetviach došlo k dôležitým posunom v otázkach odhadu veku jedinca na základe jeho kostrových pozostatkov. Proces určovania závisí na mnohých faktoroch a je pomerne zdĺhavý, urýchlenie je preto žiadúce.

Tému bakalárskej práce považujem za veľmi zaujímavú a aktuálnu, nakoľko s nadšením pozorujem rozmach umelej inteligencie a strojového učenia naprieč všetkými oblasťami ľudského života.

V mojej práci som sa zaoberala predikciou veku jedinca na základe panvovej kosti, ktorá je ako jedna z mála, väčšinou takmer neporušená časť kostrových pozostatkov.

Cieľ práce

Hlavným cieľom bakalárskej práce je navrhnúť a implementovať jednoduchý softvér, ktorý umožní na základe skenu určitej plochy panvovej kosti odhadnúť vek dospelého jedinca.

Cieľom prvej časti je oboznámiť sa so strojovým učením, jeho metódami, postupmi a rôznymi spôsobmi merania úspešnosti.

V ďalších častiach bakalárskej práce budem postupne prechádzať všetkými fázami životného cyklu vývoja softvérového produktu. Najprv sa zoznámim s problematikou určovania veku dospelého jedinca na základe kostrových po-

zostatkov. Následne sa budem zaoberať analýzou vstupných dát, oboznámim sa s formátom, v ktorom sú uložené, dôležité bude vedieť ako dáta správne načítať, zobraziť a vedieť ich transformovať do podoby vhodnej na nasledujúce spracovanie. Zároveň budem v dátach identifikovať vlastnosti, ktoré majú kosti vyobrazené na skenoch spoločné alebo naopak rozdielne. Budem pozorovať ich rôzny tvar, opotrebenie či iné morfologické zmeny.

Mojím ďalším cieľom je popísanie požiadaviek, ktoré sú na výsledný softvér kladené a priblíženie technológií, pomocou ktorých bude riešenie, ako aj samotný softvér, implementovaný.

Potom budem pokračovať prípravou dát a vytvorením návrhov niekoľkých rôznych modelov strojového učenia, porovnaním ich odhadov a následným výberom tých najperspektívnejších. Ďalej sa budem venovať vylepšovaniu niektorých modelov a napokon aj implementácií grafického používateľského prostredia.

Záverom práce sa budem zaoberať analýzou schopnosti predikčných modelov odhadnúť vek jedinca na testovacích dátach a testovaním softvérového produktu.

Cieľom bakalárskej práce nie je vytvoriť bezchybný predikčný model, ale vytvoriť softvér, ktorý zautomatizuje proces odhadovania veku na základe skenu a bude vedieť uspokojivo určiť vek dospelého jedinca.

Kapitola 1

Úvod do strojového učenia

Umelá inteligencia (angl. Artificial Intelligence, AI) je disciplína zaoberajúca sa snahou v programoch, alebo programom riadených strojoch (robotoch), vytvoriť schopnosť prejaviť vlastnosti, ktoré sú zväčša spájané s človekom. Vlastnosti ako napríklad uvažovanie, učenie, tvorivosť či plánovanie.

Pod pojmom strojové učenie (angl. Machine Learning, ML) sa rozumie určitá podoblasť umelej inteligencie, ktorá sa zaoberá algoritmami a tiež metódami, ktoré umožňujú programu učiť sa z dát a vplyvom cieľavedomých vylepšení alebo prísunu ďalších dát zlepšovať svoju schopnosť robiť predikcie presnejšie bez toho, aby boli na danú úlohu explicitne naprogramované. Modely strojového učenia sú výsledkom behu algoritmov, ktoré hľadajú spojitosti, vzory a vlastnosti vo veľkom množstve dát a na základe týchto údajov sú modely potom schopné predikovať a rozhodovať o nových dátach. Dáta sa skladajú z určitého počtu samostatných vzoriek, obsahujú rôzne črty a popisujú práve jeden objekt alebo situáciu.

Kľúčovú úlohu pri učení zohráva matematická štatistika a hĺbková analýza dát – vyťažovanie údajov (angl. Data mining). Pod pojmom Data mining sa rozumie proces extrahovania použiteľných údajov z veľkého množstva dát. Použitím vhodného softvéru hľadáme spojitosti a opakujúce sa vzory, ktoré nemusia byť na prvý pohľad zrejmé.

1.1 Typy strojového učenia

Na základe zloženia dát a úlohy, ktorú chceme pomocou algoritmov strojového učenia riešiť, rozlišujeme rôzne prístupy k forme učenia. Ak máme k dispozícii dáta aj s predikovanými premennými, čiže vieme aký má byť žiadaný výsledok, ide o učenie s učiteľom (angl. Supervised learning). Naopak, ak nevieme, aký výsledok očakávame a chceme len nájsť súvislosti v dátach, ktoré neboli predtým detegované, tak ide o učenie bez učiteľa (angl. Unsupervised learning). Ďalším prístupom je učenie s posilňovaním (angl. Reinforcement

learning), čo znamená, že algoritmus je trénovaný tak, aby vykonal sériu rozhodnutí. Model, v tomto prípade označovaný ako agent, pozoruje určité komplexné prostredie, v rámci ktorého vykonáva rozhodnutia, ktoré sú buď odmenené alebo penalizované. Agent sa učí sám a snaží sa zvoliť takú stratégiu, postupnosť rozhodnutí, aby maximalizoval množstvo odmien získaných za určitý čas.

Základnými úlohami, ktoré môže algoritmus vykonávať sú regresia, klasifikácia a zhlukovanie. Regresia je vhodná vtedy, keď chceme predikovať konkrétnu numerickú hodnotu spojitých veličín, alebo množina konečných výsledkov je príliš veľká. Príkladom regresnej úlohy môže byť predikcia ceny produktu či výšky človeka.

Naopak klasifikáciu použijeme vtedy, keď chceme objekt zaradiť do určitej triedy alebo kategórie a vieme, že množina kategórií je konečná a dostatočne malá. Prvky obsiahnuté v jednej triede majú podobné vlastnosti a líšia sa od prvkov z iných tried. Príkladom klasifikačného algoritmu môže byť rozhodnutie, či je na obrázku mačka alebo pes, človek má/nemá cukrovku atp.

Zhlukovanie je úloha podobná klasifikácií, keď sa algoritmus snaží dáta rozdeliť na zhluky s podobnými vlastnosťami, avšak ide o príklad učenia bez učiteľa. Využívajú ho napríklad marketingové spoločnosti, ktoré chcú zamerať svoje reklamy na zákazníkov s podobnými záujmami alebo predchádzajúcimi kúpami.

1.2 Základné kroky vývoja ML modelu

Prvým krokom je pochopiť problém, ktorý chceme vyriešiť a aké očakávania máme. Tento krok je dôležitý pre správne zvolenie predikčného modelu ako aj hodnotiacich parametrov.

Aby bolo možné výsledný model v poslednom kroku ohodnotiť, je nevyhnutné si vyčleniť určitú časť vstupných dát práve na zmeranie presnosti predikcií. Preto ako prvé by malo nastať rozdelenie dát na trénovacie a testovacie dáta. Testovací set sa až do posledného kroku nebude používať a akékoľvek vylepšovanie parametrov bude prebiehať len na dátach patriacich do trénovacej množiny.

Pre dosiahnutie najlepších výsledkov sú významným faktorom práve trénovacie dáta. Ich množstvo, forma, kvalita a spracovanie. Preto je žiadúce im porozumieť, vedieť čo jednotlivé vlastnosti predstavujú, v akom formáte a kde sú uložené a mnohé ďalšie faktory. Vhodné je odstránenie extrémnych hodnôt, ktoré sa od ostatných veľmi líšia (angl. outliers) a mohli by presnosť značne ovplyvniť. Tiež je adekvátne prispôsobenie formátu dát konkrétnemu algoritmu, odstránenie prebytočných čŕt, ktoré sa ukazujú ako irelevantné.

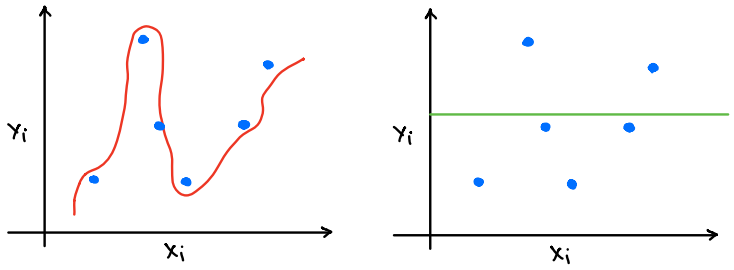
Ďalším krokom je výber modelu strojového učenia a prípadné vyladenie jeho voliteľných parametrov. V neposlednom rade nastáva vyhodnotenie presnosti odhadov pomocou zvolenej hodnotiacej funkcie na testovacej vzorke dát.

1.3 Problémy strojového učenia

Samozrejme, ako v každej oblasti, aj v oblasti strojového učenia dochádza k mnohým problémom. Jedným z nich je napríklad nedostatok relevantných dát, z ktorých by sa program mohol učiť. Veľakrát sú potrebné stovky či tisícky údajov, ktoré často nie sú dostupné, a preto program nie je schopný poskytovať presné predikcie. Ďalším problémom, s ktorým sa dátoví analytici potykajú je, že dáta nie sú v konzistentnom formáte, sú na rôznych miestach alebo niektoré vlastnosti chýbajú. Pred samotným trénovaním treba myslieť na zloženie dát. Veľmi dôležité je mať k dispozícii také dáta, ktoré odzrkadľujú vzorku, na ktorej chceme byť schopní v budúcnosti robiť predikcie, nakoľko to veľmi ovplyvní samotný predikčný model.

Problém môže prameniť aj z nevhodne zvoleného modelu, konkrétne nevhodne nastavených hyperparametrov či parametrov, čo môže vyústiť do prílišnej citlivosti modelu a jeho následnej schopnosti detegovať aj najmenšie zmeny v trénovacích dátach. Tento problém sa nazýva *overfitting*, nastáva vtedy, keď model nie je schopný správne generalizovať na nových dátach a dochádza k ich memorovaniu. Veľakrát tak model reaguje na rôzne šumy, prípadne hodnoty, ktoré sú ostatným veľmi vzdialené. Model na trénovacej vzorke vykazuje až príliš vysokú, ideálne vyzerajúcu presnosť, avšak na testovacej vzorke je jeho presnosť výrazne nižšia. Riešením môže byť zhromaždenie ďalších vzoriek, čo bude model skôr viesť k detegovaniu spojitostí a nie memorovaniu, prípadné odstránenie šumu a extrémnych hodnôt, alebo zvolenie menej komplexného modelu.

Opačným problémom je *underfitting*, ku ktorému dochádza najmä vtedy, ak model nie je schopný odhaliť žiadne súvislosti, je príliš jednoduchý pre daný problém. Preto je žiadúce bilancovať na hrane týchto dvoch problémov a vybrať model tak, aby nebol príliš jednoduchý ale ani príliš komplexný.



Obr. 1.1: Zjednodušený príklad. Modré body predstavujú dvoj-dimenzionálne dáta. Červená funkcia znázorňuje problém overfitting, zelená underfitting.

1.4 Metriky

Výsledkom každého procesu je zistenie, či bol proces úspešný alebo nie. V oblasti strojového učenia tomu nie je inak. Existuje niekoľko spôsobov ako zistiť presnosť predikcií výsledného modelu a tým aj jeho kvalitu.

1.4.1 Regresné metriky

Nasledujúce metriky sa používajú k ohodnoteniu výkonnosti modelov pri regresných úlohách.

Mean Square Error (MSE)

Veľmi používaná metrika, ktorú vypočítame ako priemer rozdielov predikovanej hodnoty – y od skutočnej hodnoty – $\hat{y}$, umocnený na druhú. N predstavuje celkový počet pozorovaní.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y - \hat{y})^2 \tag{1.1}$$

Mean Absolute Error (MAE)

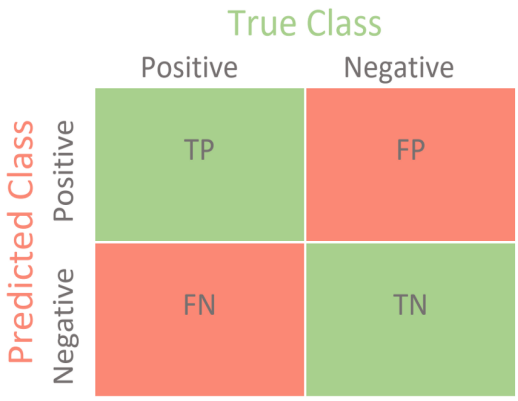
Narozdiel od predošlej metódy, počítame sumu rozdielov a nezáleží, či je rozdiel hodnôt kladný alebo záporný, vždy berieme kladný.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y - \hat{y}| \tag{1.2}$$

1.4.2 Klasifikačné metriky

Klasifikačné metriky sú založené na počte modelom správne, prípadne nesprávne, zaradených dátových vzoriek do príslušnej triedy. Bližšie túto situáciu opisuje *Confusion matrix*, ktorá je definovaná na obrázku 1.2.
Napríklad majme binárny klasifikačný problém, kedy vzorky môžu patriť do triedy s označením „1“ alebo „0“. Potom platí nasledovné:

- True positives (TP) – zahŕňa vzorky, ktorých skutočná hodnota triedy je 1 a modelom predikovaná trieda je tiež 1.
- True negatives (TN) – zahŕňa vzorky, ktorých skutočná hodnota triedy je 0 a modelom predikovaná trieda je tiež 0.
- False positives (FP) – zahŕňa vzorky, ktorých skutočná hodnota triedy je 0, ale modelom predikovaná trieda je 1.
- False negatives (FN) – zahŕňa vzorky, ktorých skutočná hodnota triedy je 1, ale modelom predikovaná trieda je 0.



Obr. 1.2: Confusion matrix pre binárnu klasifikáciu. [1]

Jedny z najčastejšie používaných metrík [2] pre binárne klasifikačné úlohy, vychádzajúce z pojmov obsiahnutých v confusion matrix, sú definované nasledovne:

Accuracy

Metrika popisuje celkovú presnosť modelu.

$$\frac{TP + TN}{TP + TN + FP + FN} \tag{1.3}$$

Precision

Dáva do pomeru počet správne predikovaných pozitívnych vzoriek k celkovému počtu pozitívnych predikcií.

$$\frac{TP}{FP + TP} \tag{1.4}$$

Recall/Sensitivity/True Positive Rate (TPR)

Udáva pomer správne predikovaných pozitívnych vzoriek.

$$\frac{TP}{TP + FN} \tag{1.5}$$

F1 score

Je harmonický priemer precision a recall.

$$2 * \frac{Precision * Recall}{Precision + Recall} \tag{1.6}$$

Specificity

Reprezentuje pomer správne predikovaných negatívnych vzoriek.

$$\frac{TN}{TN + FP} \tag{1.7}$$

False Positive Rate (FPR)

Metrika dáva do pomeru počet nesprávne predikovaných pozitívnych vzoriek a celkový počet negatívnych vzoriek.

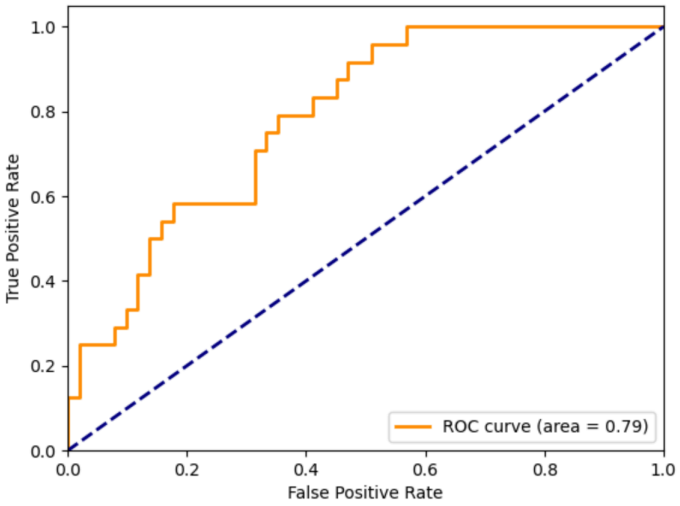
$$1 - Specificity = \frac{FP}{FP + TN} \tag{1.8}$$

AUC, krivka ROC

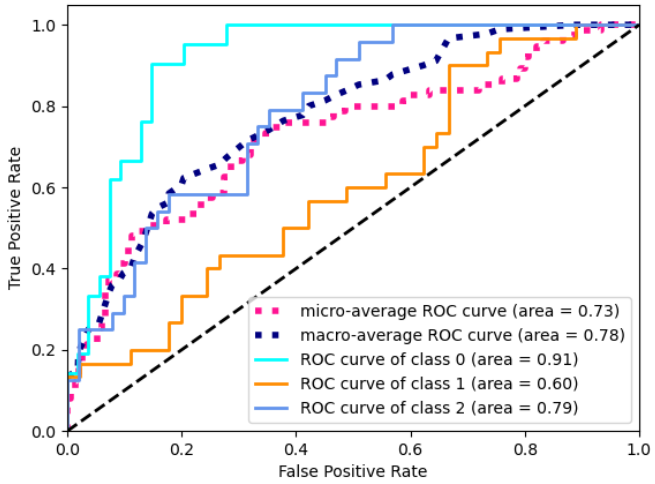
AUC (Area Under Curve) a ROC (Receiver Operating Characteristics) patria medzi často používané klasifikačné metriky. Krivka ROC dáva do závislosti True Positive Rate (TPR) a False Positive Rate (FPR) v rôznych klasifikačných medziach. Ak znížime klasifikačnú hranicu, model označí viac vzoriek ako pozitívnych, čo vyústi do väčšieho počtu FP a aj TP. Na vykreslenie ROC je nutné klasifikačný model natrénovať niekoľkokrát vždy s inou hranicou oddeľujúcou pozitívne od negatívnych vzoriek. Tento spôsob nemusí byť vždy efektívny, preto je vhodné použiť metriku AUC, ktorá zmeria celú dvojrozmernú oblasť pod touto krivkou. AUC je meradlom výkonnosti vo všetkých možných medzných hodnotách klasifikácie. Udáva, do akej miery je model schopný rozlišovať medzi jednotlivými triedami. Čím vyššia je AUC, tým lepší je model pri odhadovaní správnych tried. AUC nadobúda hodnoty na intervale od 0 po 1. [3]

- V prípade, že AUC = 0, model nebol schopný klasifikovať ani jednu vzorku správne.
- Ak AUC = 0.5, model nie je schopný rozlíšiť medzi jednotlivými triedami.
- Ak AUC = 1, potom platí, že model správne predikoval všetky pozitívne vzorky a všetky negatívne vzorky.

Väčšina zo spomenutých metrík je použiteľná aj pre klasifikačné úlohy, kedy máme na výber z viacerých tried (angl. multiclass classification). Pojmy ako TP a TN avšak nie sú také jednoznačné, je potrebné ich vypočítať pre jednotlivé triedy a potom vypočítať ich priemer. Priemer môžeme vypočítať viacerými spôsobmi, ako napríklad: „mikro“, „makro“ a „vážený“. Pre mikro spriemerovanie platí, že všetky vzorky sa rovnako podieľajú na výpočte priemeru, pri makro spôsobe sa všetky triedy podieľajú na výpočte priemeru rovnako, tj. majú rovnaké váhy a pri váženom spriemerovaní sa každá trieda podieľa podľa svojej veľkosti v celkovej vzorke údajov. [5]



Obr. 1.3: Príklad ROC krivky (oranžová) odzrkadľujúcej presnosť binárneho klasifikátoru, prerušovanou čiarou je vyznačená predikčná schopnosť modelu AUC = 0.5. [4]



Obr. 1.4: Graf zobrazuje predikčnú schopnosť určitého modelu pri klasifikačnom probléme, kde je viacero tried. Model je najpresnejší v predikciách triedy s označením „0“. [4]

1.5 Trénovacie a testovacie dáta

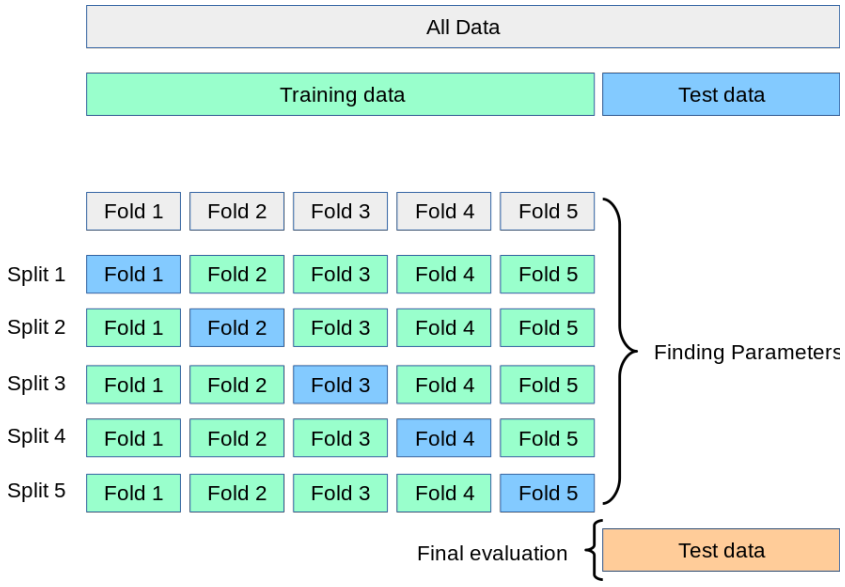
Ako bolo spomenuté v predchádzajúcej časti, na to, aby sme mohli vyhodnotiť predikčné schopnosti modelu je nutné si na začiatku vyčleniť určitú časť vstupných dát na testovanie. Existuje viacero prístupov, ako to docieliť.

Pre jednoduchosť, môžeme dáta rozdeliť na trénovacie, na ktorých sa

bude model učiť, a testovacie, na ktorých v poslednom kroku vyhodnotíme jeho kvalitu. Je vhodné dáta rozdeliť s ohľadom na výpočtové nároky pri trénovaní ako aj pri hodnotení, a takým spôsobom, aby bola testovacia aj trénovacia vzorka reprezentatívna. Môžeme dáta rozdeliť napríklad na 80% predstavujúcich trénovacie a 20% dát, ktoré budú testovacie.

Ďalším spôsobom ako rozdeliť dáta je, že ich rozdelíme na tri skupiny. Trénovacie, overovacie a testovacie. Pre trénovacie a testovacie platí to isté ako v predchádzajúcom príklade. Overovacie dáta použijeme vtedy, keď chceme porovnať ako sa daný model zlepšuje/zhoršuje vplyvom zmien v nastavení jeho voliteľných hyperparametrov.

Cross–validation je sofistikovanejšia technika ako získať informácie o tom, ako je model kvalitný. Namiesto toho, aby sme rozdelili dáta na trénovacie a testovacie, rozdelíme ich na k častí, približne rovnakej veľkosti. Následne použijeme $k-1$ častí na natrénovanie modelu a zvyšnú časť na otestovanie jeho presnosti. Tento proces opakujeme *k-krát*, čím docielime, že každá z častí bude práve raz použitá na otestovanie predikčných schopností modelu. Týmto spôsobom, využijeme všetky dáta na natrénovanie aj otestovanie modelu, čo je výhodou keď je vstupných dát menší počet. Nevýhodou je, že trénovací proces musí prebehnúť niekoľkokrát, a to môže byť problematické pri modeloch, ktorých trénovanie je časovo náročné.



Obr. 1.5: Schéma techniky rozdelenia dát nazývaná cross–validation. [6]

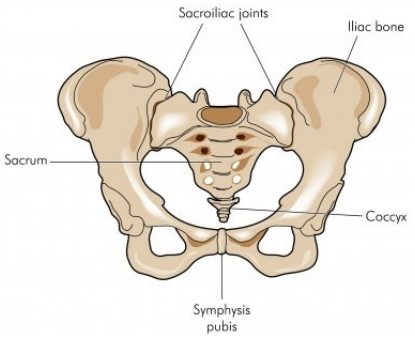
O všetky z vyššie spomenutých pojmov a metód som sa opierala pri riešení problému, ktorý bližšie popisujem v nasledujúcich kapitolách.

Kapitola 2

Analýza

2.1 Odhad veku z kostrových pozostatkov

Forenzní antropológovia, historici a archeológovia sa odhadom veku zaoberajú stovky rokov. Metód na odhad veku dospelých jedincov je niekoľko, zahŕňajú hodnotenie mozgovej časti lebky, panvových kostí či hodnotenie stavu koncov rebier. [7] V tejto práci som sa ďalej zaoberala analýzou panvových kostí, konkrétne lonovej spony (lat. symphysis pubica), ktorá sa nachádza v prednej časti stretu panvový polovíc.



Obr. 2.1: Ilustrácia panvy s vyznačenými časťami. [8]

Prvú štúdiu zaoberajúcu sa morfologickými zmenami lonovej kosti publikoval v roku 1920 americký osteológ T. Todd. [9] Išlo o analýzu 306 vzoriek mužských lonových kostrových pozostatkov, ktoré na základe znakov zaradil do desiatich fáz. Táto štúdia bola počas nasledujúcich rokov niekoľkokrát revidovaná a bolo odhalených viacero nedostatkov. Za najznámejšiu a najpoužívanejšiu sa dnes považuje metóda Suchey a Brooks z roku 1990 [10], ktorá vychádza z revidovanej Toddovej metódy. Analýza hodnotí 1225 vzoriek, vrátane ženských lonových kostí. Narozdiel od Toddovej metódy, Suchey

a Brooks zaraďujú pozostatky do šiestich fáz, a aj napriek rozdielom medzi mužskou a ženskou vzorkou popisujú najmä zmeny, ktoré sú kľúčové pre obe pohlavia. Táto metóda bola aplikovaná aj na kostrové pozostatky ľudí z viacerých kútov sveta ako napríklad z Poľska, Bosny a Hercegoviny, Austrálie a vykazovala pomerne vysokú presnosť, prípadne bola podkladom k niektorým vylepšeniam. [11]

Obr. 2.2: Ilustrácia zobrazujúca jednotlivé fázy zmien plochy mužskej lonovej kosti podľa Suchey a Brooks. [12]

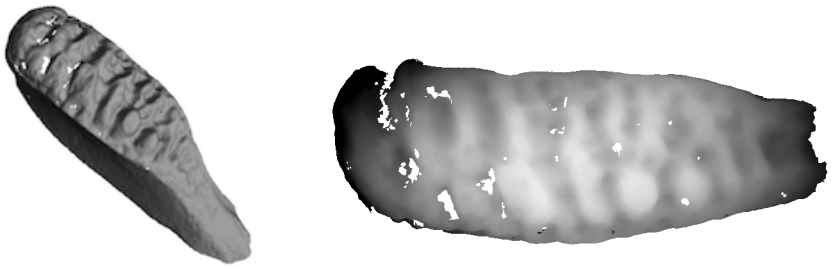
Z vyššie spomínaných štúdií vyplýva, že mladší jedinci majú povrch lonovej kosti viac členitý, je vlnitého charakteru a čím je jedinec starší, tak sa plocha vyhladzuje. S rastúcim vekom pribúdajú v kostiach malé fraktúry. Taktiež je dôležité si povšimnúť, že jednotlivé fázy sa vyskytujú so stále sa zväčšujúcim intervalom kalendárneho veku a posledná fáza je určená pomerne širokým intervalom, z čoho vyplýva, že približne po štyridsiatom roku života nám stav lonovej kosti vek nijak nehodnotí.

2.2 Analýza vstupných dát

Vstupné dáta, ktoré dodal vedúci bakalárskej práce, boli uložené vo formáte STL, čo je skratka pre „Standard Triangle Language“ alebo „stereolitografia“. Formát slúži na uchovanie informácie o 3D modeloch. Povrch modelu je mozaikovitý alebo logicky rozdelený na sériu malých trojuholníkov (faziet). Každá fazeta je opísaná kolmým smerom a tromi bodmi predstavujúcimi vrcholy trojuholníka. [13]

Po konzultácií s vedúcim bakalárskej práce sme usúdili, že je vhodnejšie ďalej pracovať a na vstupe do aplikácie prijímať, súbory vo formáte PNG (Portable Network Graphics) vytvorenými vhodnou transformáciou z STL súborov – otočením a použitím pohľadu „zhora“ na plochu lonovej kosti.

Obrázky na vstupe boli čiernobiele, pixely nadobúdali hodnoty od *0*, čo reprezentuje čierny pixel, až po hodnotu *1*, čo je naopak biely pixel. Rozmery obrázka boli 900 na 1200 pixelov a všetky plochy boli zarovnané na stred obrázka. Transformáciou 3D objektu na objekt dvoj-dimenzionálny došlo k strate niektorých informácií, ale samotné spracovanie dát to značne uľahčilo a zrýchlilo.



Obr. 2.3: Vstupné dáta v STL (vľavo) a v PNG (vpravo) formáte.

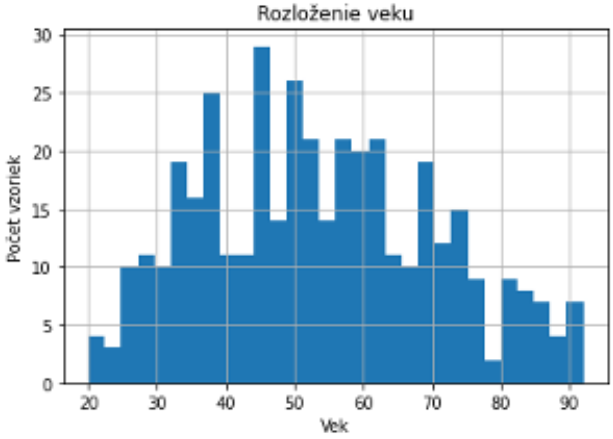
K dispozícií som mala 399 vzoriek kosťových skenov. Všetky súbory mali názov v tvare *„identifikátor_typKosti_stranaKosti_PohlavieVek_orez.png“*. Poskytujú tak ďalšie informácie ako je špeciálny identifikátor, pohlavie, typ kosti, vek a či sa jedna o ľavú alebo pravú stranu plochu kosti. Tieto informácie som extrahovala zo všetkých vzoriek do textového súboru a jednotlivé hodnoty oddelila pomocou „;“ aby sa mi s nimi v ďalšej fáze dobre pracovalo.

	image	id	bone	side	sex	age
1	BIE25S_sym_sin_F34.stl.png	BIE25S	sym	sin	female	34
2	BIE25S_sym_dex_F34.stl.png	BIE25S	sym	dex	female	34
3	BIE14S_sym_sin_M62_orez_rucne.stl.png	BIE14S	sym	sin	male	62
4	BEX7S_sym_dex_M46_orez.stl.png	BEX7S	sym	dex	male	46
5	BEX2S_sym_dex_M65_orez_rucne.stl.png	BEX2S	sym	dex	male	65

Obr. 2.4: Prehľad tabuľky reprezentujúcej vstupné dáta.

Identifikátor ani názov kosti neobsahovali žiadnu pridanú hodnotu (nakoľko všetky vzorky boli vzorky lonovej kosti), preto som s týmito vlastnosťami ďalej nepracovala. Po detailnom preskúmaní tabuľky zobrazenej na obrázku 2.4 vyplýva, že vlastnosti *sex* a *side* nadobúdajú len dve hodnoty, a to hodnoty female/male pre označenie ženskej/mužskej vzorky a sin/dex pre označenie ľavej/pravej lonovej kosti.

Súbor obsahuje 235 mužských vzoriek a 164 ženských vzoriek kostí. Najdôležitejšou hodnotou je pochopiteľne hodnota veku, preto som sa na ňu pozrela dôkladnejšie. Graf na obrázku 2.5 predstavuje vekové rozloženie vzoriek v rámci množiny dát. Z údajov na obrázku 2.6 ďalej vyplýva, že stredná hodnota je 53.6 rokov, štandardná odchýlka má hodnotu 17, najmenšia hodnota je 20 rokov a najväčšia 92 rokov. Je očividné, že sú vo väčšom počte zastúpené skôr vzorky starších jedincov. Podľa Suchey a Brooks väčšina vzoriek tak patrí do poslednej fázy, čo značí veľmi nevyváženú množinu vstupných údajov. Predikčný model by to mohlo viezť k predikovaniu len väčšinovej triedy. Vzoriek nad 40 rokov je 301, kým pod 40, kedy dochádza k signifikantnejším zmenám, je menej ako 100.



Obr. 2.5: Histogram pre zobrazenie rozloženia veku vo vzorkách obsiahnutých vo vstupných dátach.

	age
count	399.000000
mean	53.659148
std	17.044845
min	20.000000
25%	40.000000
50%	52.000000
75%	66.000000
max	92.000000

	age
count	98.000000
mean	32.336735
std	4.961592
min	20.000000
25%	29.000000
50%	34.000000
75%	36.750000
max	39.000000

Obr. 2.6: Štatistické informácie o veku všetkých vzoriek vstupných dát (vľavo), vzoriek kde je vek nižší ako 40 rokov (vpravo).

2.3 Vymedzenie problému

Vychádzajúc z analýzy vstupných dát, kedy okrem samotných obrázkov a ďalších vlastností, som mala k dispozícií aj predikované premenné – údaj veku, jedná sa o *učenie s učiteľom*. Na základe samotných štúdií, ktoré som opísala v predchádzajúcej časti, nie je úplne možné morfologické zmeny a opotrebenia kosti pripísať vždy konkrétnemu roku života jedinca, obe štúdie určujú intervaly, alebo fázy, v ktorých ku konkrétnym zmenám môže dochádzať. V tomto prípade bolo možné úlohu riešiť ako klasifikačný problém, zaradenie vzorky do triedy, ktorá bude charakterizovaná intervalom veku, ale aj ako regresný problém, nakoľko išlo o intervaly, ktoré sú spojitého charakteru. Ja som zvolila prvý prístup a k jednotlivým intervalom som pristupovala ako ku triedam a označila preto úlohu ako *klasifikačný problém*.

Po konzultácií s vedúcim bakalárskej práce, som zvolila 3 rôzne rozdelenia na triedy popisujúce rôzne intervaly veku.

1. Klasifikácia dát do 2 tried
    - 18 – 40 rokov (1)
    - 41 a viac rokov (2)
2. Klasifikácia do 3 tried
    - 18 – 29 rokov (1)
    - 30 – 40 rokov (2)
    - 41 a viac rokov (3)
3. Klasifikácia do 4 tried
    - 18 – 39 rokov (1)
    - 40 – 55 rokov (2)
    - 56 – 70 rokov (3)
    - 71 a viac rokov (4)

V prvom prípade som otestovala predikčné schopnosti modelu, ktoré vychádzajú z predpokladu, že po 40. roku života už vek človeka lonová kosť nijak neurčuje. Druhý prípad je rozšírením prvého, kedy som prvý interval rozdelila na dva, čím je klasifikácia konkrétnejšia. V poslednom rozdelení som pozorovala, či je model schopný odhaliť nejaké vlastnosti aj vo fázach, kedy stav lonovej kosti nemá výpovednú hodnotu s ohľadom na vek.

Avšak rozdelenie na tieto triedy viedlo k problému, že niektoré triedy boli zastúpené vo výrazne väčšom počte ako ostatné. Riešení existuje niekoľko. Ako prvý spomeniem spôsob, kedy jednotlivým triedam priradíme číslo – váhu, zvyčajne na základe pomeru výskytu každej triedy. Alebo môžeme z triedy,

ktorá v rámci rozdelenia obsahuje zreteľne viac vzoriek odstrániť určitý počet vzoriek, čím docielime vyrovnanie počtu v každej triede. Tento prístup sa nazýva *undersampling* a v tomto prípade ho nepovažujem za vhodný, nakoľko vstupných dát je dosť málo a takto by som prišla o trénovacie dáta. Opakom je tzv. *oversampling*, kedy do množiny vstupných dát niektoré vzorky z minoritnej triedy vložíme viackrát. Tento prístup môže viezť k tomu, že je model náchylnejší na overfitting. Ja som sa zvolila prístup nazývaný „syntetický“ oversampling, ktorý bližšie popíšem v nasledujúcej kapitole.

2.4 Požiadavky na softvér

2.4.1 Funkčné požiadavky

- Softvér umožní používateľovi nahrať práve jeden súbor vo formáte PNG. Užívateľ bude upozornený v prípade, že nahral súbor v inom formáte ako je akceptovaný. (F1)
- Softvér umožní používateľovi zobraziť nahratý súbor. (F2)
- Používateľ bude môcť doplniť pohlavie a stranu kosti. (F3)
- Používateľ si bude môcť vybrať, či chce vek klasifikovať do dvoch tried a/alebo do troch. (F4)
- Program zobrazí s akou pravdepodobnosťou vzorka patrí do danej triedy v rámci zvoleného rozdelenia. (F5)

2.4.2 Nefunkčné požiadavky

- Softvér bude mať jednoduché používateľské prostredie. (N1)
- Program bude napísaný tak, aby sa data-mining model dal kedykoľvek vymeniť za iný, bez rozsiahlejšieho zásahu do iných častí programu. (N2)

Kapitola 3

Návrh a realizácia

V rámci tejto kapitoly popíšem použité technológie a predprípravu vstupných dát, ale hlavne predstavím niekoľko modelov strojového učenia, ktoré som zvažovala použiť ako riešenie klasifikačnej úlohy odhadu veku jedincov. Následne opíšem ako som vylepšila presnosť niektorých modelov pomocou metód optimalizácie hyperparametrov.

3.1 Použité nástroje

Python

Problém som sa rozhodla implementovať v jazyku Python, nakoľko v posledných rokoch sa ukazuje ako najpopulárnejší programovací jazyk v oblasti strojového učenia. [14] Za jeho hlavné výhody považujem jeho jednoduchú syntax a rozsiahly počet knižníc na prácu s dátami. Veľkým prínosom je aj množstvo podporných materiálov, ako aj moja predchádzajúca skúsenosť s programovaním v tomto jazyku.

PySimpleGUI

Existuje niekoľko grafických frameworkov postavených na jazyku Python. Veľakrát sú ale príliš komplikované, nedostatočne zdokumentované alebo staré. V mojej bakalárskej práci som na implementáciu grafického používateľského rozhrania použila knižnicu PySimpleGUI [15], ktorá v sebe zahŕňa funkcionality z niekoľkých známych Python frameworkov ako sú Tkinter, Qt, Remi a WxPython. Je veľmi dobre zdokumentovaná a jej používanie je intuitívne.

Jupyter notebook

Na samotnú prípravu a vizualizáciu dát, či výber a natrénovanie jednotlivých modelov som použila webovú aplikáciu *Jupyter notebook* [16], ktorá umožňuje vytvárať interaktívne súbory. Veľkou výhodou je živé spustenie softvérového

kódu v kombinácií s výstupmi, multimediálnymi súbormi a doplňujúcim textom.

Podporné knižnice jazyka Python

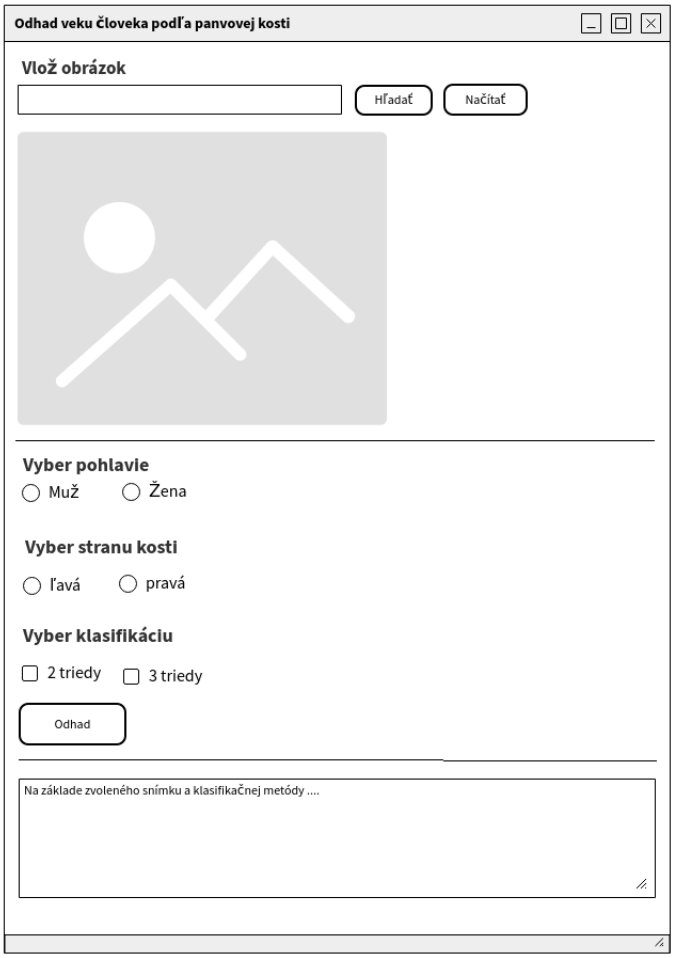
- *Scikit-learn* [17] je populárna open source knižnica zameraná na strojové učenie. Túto knižnicu som v mojej práci použila na trénovanie data-mining modelov kvôli jej intuitívnemu používaniu a rozsiahlej dokumentácií.
- Knižnica *Pandas* [18] je vhodná na prácu a manipuláciu s dátami. V práci som ju využila na uloženie dát do štruktúry Dataframe a na následnú analýzu.
- Na vizualizáciu dát som použila knižnicu *matplotlib* [19] a *scikit-plot* [20]. Tieto knižnice poskytujú na výber z veľkej škály grafov a diagramov.
- Na prácu s obrázkami som použila knižnice *OpenCV* [21] a *Scikit-image* [22], čo sú knižnice vhodné práve na manipuláciu s obrázkami, ich predspracovanie a strojové videnie.
- Knižnica *NumPy* [23] mi v bakalárskej práci poslúžila pri niektorých matematických výpočtoch.
- Knižnicu *imbalanced-learn* [24] som použila na vysporiadanie sa s problémom malého počtu vstupných vzoriek. Táto knižnica pokrýva rôzne spôsoby ako sa vyrovnať s nevyváženými množinami vstupných dát (napr. oversampling, undersampling).

3.2 Návrh obrazovky

Na obrázku 3.1 je zobrazený návrh obrazovky používateľského rozhrania vychádzajúci z požiadaviek kladených na softvér. Bol skonštruovaný pomocou webovej aplikácie `www.mockflow.com`

3.3 Optimalizácia hyperparametrov

Modely strojového učenia majú okrem interných parametrov, ktorých hodnota závisí od dát, na ktorých je model natrénovaný, aj tzv. *hyperparametre*, čo sú externé parametre modelov a ich hodnota nevychádza z trénovacích dát. Tieto hyperparametre musia byť explicitne nastavené programátorom ešte pred tým, ako samotné trénovanie modelu začne. Vhodné hodnoty hyperparametrov závisia od konkrétneho problému a daného modelu. Každý model má iné množstvo a typ voliteľných hyperparametrov a ich ručné nastavovanie je veľmi zdĺhavé a pracné. Python knižnica *scikit-learn* však poskytuje dva generické prístupy, ktoré celý proces zautomatizujú.



Obr. 3.1: Prototyp obrazovky.

GridSearchCV

Prvým prístupom je metóda GridSearchCV (GS), ktorá dostane na vstupe model a množinu hyperparametrov s možnými hodnotami, ktoré má algoritmus na danom modeli otestovať. Hyperparametre, ktoré nie sú zahrnuté v množine si ponechávajú v tomto procese defaultnú hodnotu. GridSearchCV potom určený model postupne natrénuje s využitím jednotlivých kombinácií hodnôt hyperparametrov a ohodnotí ich predikčnú schopnosť pomocou metódy cross–validation. Vo výsledku máme prehľad o tom akú má daný model presnosť v predikciách s rôznymi hodnotami hyperparametrov a môžeme vybrať takú kombináciu, ktorú vyhodnotí ako najlepšiu.

RandomizedSearchCV

Druhým prístupom je metóda RandomizedSearchCV (RS), ktorá funguje na rovnakom princípe avšak s tou výnimkou, že neskúša kombinácie z celej množiny, ale len z náhodne zvolenej podmnožiny hyperparametrov. Na vstupe je nutné určiť koľko rôznych kombinácií má algoritmus vyskúšať.

V oboch prípadoch je možné si zvoliť metriku, pomocou ktorej bude algoritmus cross–validation vyhodnocovať predikčné schopnosti, ako aj počet častí, na ktoré sa rozloží množina trénovacích dát.

3.4 Príprava vstupných dát

Všetky obrázky lonových kostí som načítala ako dvojrozmerné pole hodnôt pixelov. Nakoľko rozmer každého obrázka je 900 na 1200 pixelov vo výsledku by som pracovala s takmer 1.1 miliónom numerických hodnôt pre každú vzorku. Nevyhnutné bolo obrázky orezať na takú veľkosť, aby boli viditeľné plochy lonových kostí a zároveň, aby neostával prebytočný okraj. Výsledná veľkosť obrázkov po orezaní bola 450 na 880 pixelov.

Ďalším krokom by bola dôkladná úprava a predspracovanie vstupných dát ako napríklad aplikácie filtrov, detekcia hrán, rotácia, úprava jasu, kontrastu, segmentácia a prípadne iné techniky. To však nebolo cieľom mojej bakalárskej práce a po konzultácií s vedúcim práce som pokračovala bez aplikácie týchto techník.

Hodnoty jednotlivých pixelov som uložila do štruktúry *DataFrame* spolu s ostatnými vlastnosťami jednotlivých vzoriek. Nie všetky atribúty objektov boli čisto numerické, vlastnosti *sex* a *side* bolo nutné previezť do číselnej podoby. Vytvorila som namiesto nich nové atribúty a to: *female*, *male*, *left* a *right*. Hodnoty jednotlivých atribútov som vyplnila spôsobom, že ak mala vzorka príznak *sex* s hodnotou „female“ tak nová hodnota príznaku *female* bude „1“ inak „0“. Týmto spôsobom pre všetky novovzniknuté vlastnosti.

Následne som si vstupné dáta rozdelila na trénovaciu množinu a testovaciu množinu pomocou funkcie *train_test_split* z knižnice *scikit-learn*, v pomere 7:3. Podľa rozdelenia v sekcií 2.3 aj analýzy, mali niektoré triedy výrazne vyšší počet vzoriek ako iné. Preto som na vyváženie trénovacej množiny dát použila techniku nazývanú *SMOTE*, čo je skratka pre *Synthetic Minority Oversampling Technique*, čo je technika na syntetické vytváranie vzoriek minoritnej triedy za účelom vyváženia dát. [24] Narozdiel od klasického použitia techniky *oversampling*, ktorá len zduplikuje náhodné vzorky z minoritnej triedy a nepridá do množiny dát žiadnu novú informáciu, SMOTE funguje tak, že vyberie náhodnú vzorku z minoritnej triedy, v priestore k nej nájde jej najbližších susedov (vlastnosťami podobné vzorky), náhodnú zo susediacich vzoriek vyberie a syntetickú vzorku vytvorí na ľubovoľnom mieste v priestore ležiacom

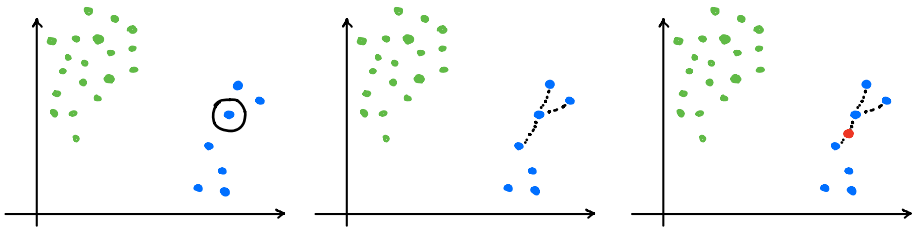
medzi tými dvoma vzorkami. Tento proces sa opakuje toľkokrát, koľko je nevyhnutné pre vyváženie množín dát. Nevýhodou tejto techniky je, že vôbec neberie do úvahy majoritnú triedu, a preto môže dôjsť k vzniku zvláštnych vzoriek, najmä vtedy ak dochádza v priestore k prekrytiu jednotlivých tried. Počty dát po použití techniky SMOTE v jednotlivých množinách odpovedajúcim rozdeleniam z 2.3 sú obsiahnuté v tabuľkách 3.1.

	1	2
Trénovacie dáta	206	206
Testovacie dáta	32	88

	1	2	3
Trénovacie dáta	205	205	205
Testovacie dáta	8	23	89

	1	2	3	4
Trénovacie dáta	88	88	88	88
Testovacie dáta	29	38	31	22

Tabuľka 3.1: Rozdelenie a pomery dát podľa jednotlivých intervalov.



Obr. 3.2: Zjednodušená ilustrácia procesu SMOTE, minoritná trieda je označená modrými bodmi, majoritná zelenými, červený bod predstavuje novovzniknutú synteticky vytvorenú vzorku.

3.5 Uvažované modely predikcie veku jedincov

3.5.1 Logistická regresia

Aj keď názov skôr evokuje model, ktorý sa zaoberá regresnými problémami, logistická regresia (angl. Logistic regression) sa využíva aj na klasifikáciu dát,

nakoľko odhaduje s akou pravdepodobnosťou vybraná inštancia patrí do konkrétnej triedy. Ak je táto pravdepodobnosť väčšia ako 50%, tak model označí, že táto vzorka patrí do danej triedy, alebo v opačnom prípade, že nepatrí. Tento opis odpovedá binárnej klasifikácií, pri viacerých triedach je to však podobné, model objekt zaradí do triedy, do ktorej patrí s najväčšou pravdepodobnosťou. Logistická regresia patrí medzi lineárne klasifikátory, čiže vychádza z lineárnej funkcie

$$f(x) = b_0 + b_1 x_1 + ... + b_r x_r \tag{3.1}$$

kde $b_0, b_1, ..., b_r$ sú koeficienty a $x_1, x_2, ..., x_r$ je vektor vlastností vzorky na vstupe. Avšak narozdiel od lineárnej regresie, na výstupe nie je priamo výsledok, ale logistická funkcia – *sigmoida* tohto výsledku. [25]

3.5.2 Metóda najbližších susedov

Metóda supervizovaného učenia označovaná aj KNN (z anglického K Nearest Neighbors). Narozdiel od mnohých ostatných modelov, tento sa nesnaží vytvoriť generalizujúci vnútorný model, ale jednotlivé trénovacie dáta si uchováva. Pri predikcií triedy pre dátový bod x sa vyberie k jeho najbližších susedov, kde k je celočíselný kladný parameter, ktorý vopred zadáme. Na základe hlasu majority k zvolených susedov je dátovému bodu priradená príslušnosť ku konkrétnej triede. Zvolenie hodnoty parametru k závisí od konkrétnej množiny dát, ale vo všeobecnosti platí, že čím je k väčšie, tým dochádza k aktívnejšiemu potlačeniu šumu v dátach ale aj nejasným hraniciam medzi jednotlivými triedami. [26]

3.5.3 Rozhodovací strom

Metóda rozhodovacích stromov (angl. Decision tree) je vhodná na riešenie regresných ako aj klasifikačných problémov. Spočíva vo vytvorení modelu, ktorý je schopný predikovať hodnotu/príslušnosť k triede na základe množiny rozhodovacích pravidiel. Tieto pravidlá sú v tvare if-then-else tvrdení o vlastnostiach „naučených“ z trénovacích dát. Knižnica *scikit-learn* na to používa algoritmus, ktorý je pomerne jednoduchý: najprv rozdelí trénovaciu množinu dát na dve časti podľa nejakej črty k a medze tejto črty t_k, dvojicu (k, t_k) zvolí tak, aby vytvorila „najčistejšie“ možné rozdelenie, čo znamená, že všetky trénovacie inštancie, pre ktoré platí pravidlo dané dvojicou (k, t_k), spadnú do rovnakej triedy. Proces pokračuje rozdelením vzniknutých častí na ďalšie dve časti, čo sa opakuje až kým strom nedosiahne maximálnu povolenú hĺbku alebo už neexistuje také rozdelenie, ktoré by znížilo „nečistotu“ danej triedy. [27]

Vo všeobecnosti platí, že čím je strom hlbší, tým je komplexnejší a viac sa približuje množine trénovacích dát. Rozhodovacie stromy sú na tieto dáta veľmi citlivé a len malé zmeny v nich môžu viesť k signifikantným zmenám vo výsledkoch. Medzi výhody tejto metódy patrí možnosť si rozhodovací strom

vykresliť, je jednoduchý na pochopenie. Nie je nutné dáta vopred predpripraviť, narozdiel od iných modelov trénovacie dáta nemusia byť normalizované. Nevýhodou rozhodovacích stromov je ich častá náchylnosť na overfitting, modely sú veľakrát príliš komplexné a neschopné generalizácie. Aby k tomu nedochádzalo je vhodné obmedziť niektoré hyperparametre modelu, napríklad hĺbku stromu.

3.5.4 Náhodný les

Ako už z názvu vyplýva, náhodný les (angl. Random forest) sa skladá z veľkého počtu individuálnych rozhodovacích stromov, ktoré vytvárajú jeden celok (angl. ensemble). Náhodný les je metóda učenia s učiteľom, vhodná na klasifikačné aj regresné úlohy.

Ensemble metódy sú založené na *„wisdom of the crowd“*, kedy jednotlivé modely v rámci súboru poskytnú svoje predikcie pre danú vzorku a výslednou predikovanou trieda je tá, ktorú určil najväčší počet modelov. Veľakrát takto agregované predikcie vedú k presnejším výsledkom ako majú jednotlivé modely samostatne, obzvlášť vtedy, keď sú jednotlivé modely natrénované odlišnými trénovacími algoritmami. Rozlišujeme medzi dvoma princípmi vytvárania takýchto súborov:

- „averaging method“ – metóda založená na natrénovaní niekoľkých jednoduchých predikčných modelov a následnom spriemerovaní ich odhadov do výsledku.

- „boosting method“ – metóda založená na sekvenciálnom vylepšovaní predošlého modelu, za cieľom vylepšiť schopnosť predikcií.

Náhodný les je príkladom prvej metódy, jednotlivé stromy sú natrénované z podmnožiny vstupných trénovacích dát.

3.5.5 Dummy estimator

Pri trénovaní modelov učenia s učiteľom je veľakrát užitočné ich porovnať s veľmi jednoduchými pravidlami. Tieto pravidlá sú implementované v *scikit-learn* knižnici pomocou Dummy klasifikátora. Tento klasifikátor poskytuje predikcie na základe zvolenej stratégie. Príkladom takejto stratégie je, že model vždy predikuje triedu, ktorá je v množine dát najviac zastúpená, alebo predikuje čisto náhodne, alebo naopak nech je na vstupe čokoľvek, vždy predikuje tú istú triedu. Vďaka tomuto je možné vyhodnotiť, či sa model naučil rozpoznávať nejaké vzory z dát a či je schopný relevantných predikcií.

3.6 Výsledky predikcií na trénovacej množine dát

Každý z modelov spomenutých v predošlej časti som postupne natrénovala pomocou trénovacích dátach rozdelených do príslušných tried intervalov zo sekcie 2.3. Všetko to boli modely s defaultnými hodnotami parametrov aké poskytuje knižnica *scikit-learn*. Ich presnosť som porovnala pomocou metriky *f1 score* s *mikro* spriemerovaním hodnôt, čo znamená, že každá vzorka sa do priemeru počíta rovnako. Tiež som použila techniku cross–validation, kedy sa trénovací set rozdelí na päť častí, pomocou štyroch častí sa model natrénuje a zvyšná časť sa použije na ohodnotenie. Tento proces sa opakuje päťkrát a jednotlivé výsledky sa spriemerujú. Výsledky pre jednotlivé rozdelenia som zaznamenala do tabuliek 3.2 až 3.4.

Klasifikátor	Priemer f1 score	Smerodajná odchýlka
DummyEstimator	0.498	0.003
LogisticRegression	**0.886**	0.043
KNN	0.580	0.038
DecisionTree	0.707	0.064
RandomForest	**0.862**	0.036

Tabuľka 3.2: F1 score výsledky cross–validation techniky na trénovacích dátach jednotlivých modelov pre rozdelenie do dvoch tried.

Klasifikátor	Priemer f1 score	Smerodajná odchýlka
DummyEstimator	0.333	0.000
LogisticRegression	**0.941**	0.021
KNN	0.649	0.024
DecisionTree	0.798	0.045
RandomForest	**0.928**	0.027

Tabuľka 3.3: F1 score výsledky cross–validation techniky na trénovacích dátach jednotlivých modelov pre rozdelenie na tri triedy.

Klasifikátor	Priemer f1 score	Smerodajná odchýlka
DummyEstimator	0.241	0.002
LogisticRegression	0.597	0.119
KNN	0.540	0.127
DecisionTree	0.441	0.089
RandomForest	0.591	0.134

Tabuľka 3.4: F1 score výsledky cross–validation techniky na trénovacích dátach jednotlivých modelov pre rozdelenie do štyroch tried.

Z tabuliek som jasne pozorovala, že v prvom a druhom rozdelení sa ako najpresnejšie javia klasifikátory LogisticRegression a RandomForest. Smerodajná odchýlka pri hodnoteniach modelov posledného rozdelenia je vyššia ako v prvých dvoch prípadoch, napriek tomu ale platí, že LogisticRegression a RandomForest poskytujú najlepšie výsledky. F1 score všetkých modelov je lepšie ako f1 score DummyEstimator, čo som považovala za znamenie, že každý model bol schopný sa niečo zo vstupných trénovacích dát naučiť.

3.7 Výsledky aplikovania optimalizácie

Oba spomenuté prístupy som aplikovala pri snahe vylepšiť presnosť modelov, ktorých predikčné schopnosti som považovala za perspektívne. Vo všetkých troch prípadoch rozdelenia určeného v časti 2.3 sa ukázali ako najlepšie modely RandomForest a LogisticRegression. Navrhla som príslušné množiny hyperparametrov pre oba modely strojového učenia, ktorých kombinácie som pomocou algoritmov GridSearch a RandomizedSearch preskúmala.

Za hyperparametre pre LogisticRegression som z dokumentácie *scikit-learn* zvolila *solver, penalty* a *C*. Kde *solver* predstavuje algoritmus, pomocou ktorého chceme aby sa optimalizačný problém počítal, hyperparametrom *penalty* určíme regularizačnú techniku a hyperparameter C nám špecifikuje silu regularizácie. V tomto prípade som použila prístup RandomizedSearchCV, kde som nastavila hľadanie desiatich náhodne vybraných kombinácií v troch iteráciách. Proces bol výpočetne veľmi náročný, trénovanie modelu s niektorými kombináciami hyperparametrov trvalo aj niekoľko hodín.

Pre model RandomForest som vybrala množinu hyperparametrov *bootstrap*, *min_samples_leaf*, *max_depth*, *min_samples_split* a *n_estimators*. Hyperparameter *bootstrap* predstavuje techniku, kedy na natrenénovanie modelov nie sú použité všetky vstupné dáta, ale vždy je použitá len určitá náhodne vybraná podmnožina s tým, že konkrétne vzorky môžu byť vybraté viackrát. Pomocou *min_samples_leaf* špecifikujem minimálny počet vzoriek, ktorý je potrebný na to, aby už nedochádzalo k ďalšiemu deleniu uzlov v stromoch, a strom sa tak v tejto vetvi zastavil. Naopak *min_samples_split* určuje minimálny počet vzoriek, ktorý je potrebný k ďalšiemu deleniu vnútorných uzlov stromu. Hyperparameter *max_depth* obmedzuje hĺbku stromu a *n_estimators* určuje počet rozhodovacích stromov tvoriacich ensemble. Nakoľko trénovanie náhodného lesu nebolo časovo náročné použila som techniku GridSearchCV, ktorá prehľadala 144 kombinácií hyperparametrov tiež v troch iteráciách.

Výsledkom oboch prístupov sú natrénované modely s takou kombináciou hyperparametrov, ktorá sa v cross–validation hodnotení ukázala ako najlepšia. Jednotlivé výsledky som zahrnula do tabuľky 3.5.

Je nutné podotknúť, že výsledky vyzerajú pomerne „ideálne“ v prvých dvoch kategóriách a je dosť možné, že modely sú „pretrénované“ a nastal over-

Klasifikátor	1. f1 score	2. f1 score	3. f1 score
LogisticRegression	0.860	0.933	0.600
RandomForest	0.830	0.919	0.580

Tabuľka 3.5: F1 score výsledky modelov s kombináciami hyperparametrov, ktoré RandomizedSearchCV a GridSearchCV označili za najlepšie.

fitting. Finálne výsledky presností som otestovala pomocou testovacej množiny dát v nasledujúcej kapitole.

3.8 Implementácia používateľského rozhrania

Na tvorbu grafického rozhrania som použila balíček *PySimpleGUI*. Základ tvorí skript v jazyku Python a element *Window* spolu s ďalšími elementami ako: *Button, RadioButton, Text, Input, CheckBox* a iné. Beh programu je vymedzený behom cyklu, ktorý je riadený pomocou tlačidiel, ktoré spúšťajú konkrétne udalosti. Okrem hlavného skriptu, ktorý reprezentuje grafické rozhranie a jednotlivé elementy spúšťajúce rozličné procesy, sa program skladá z ďalších pomocných skriptov jazyka Python. Tieto skripty obsahujú podporné funkcie na spracovanie vstupu, prípravu obrázka, predikcie modelov a následný výpis výsledkov.

Používatelia majú možnosť informáciu o strane kosti a pohlaví jedinca zadať, pokiaľ ňou disponujú. Ak nie, program umožňuje pokračovať aj bez zadania týchto informácií. Na to, aby prebehla estimácia je nutné nahrať len obrázok a zvoliť typ odhadu. Toto je implementované pomocou možnosti zakázať určité funkcionality, kým nie sú zadané všetky dáta.

Príprava vývojového prostredia v jazyku Python je veľakrát pomerne komplikovaná. Dôvodom sú rôzne verzie jazyka, rôzna verzia podporných knižníc ako aj ich kompatibilita. Pokiaľ sú všetky závislosti v poriadku, skript je jednoduché pomocou konzoly spustiť. Avšak ak nie sú, vyžaduje to väčšie úsilie.

Výstupom mojej bakalárskej práce je aj spustiteľný *.exe* súbor, ktorý umožňuje spustiť program, bez nutnosti inštalácie všetkých knižníc a závislostí vopred, na operačnom systéme Windows 10. Aby som toto docielila použila som knižnicu *PyInstaller* [28], ktorá vyrieši všetky závislosti a zabalí ich do jedného spustiteľného súboru.

Všetky informácie ako spustiť program na Windows 10, prípadne ako ho spustiť na iných operačných systémoch sú obsiahnuté v *Používateľskej príručke*, ktorá je prílohou B tejto bakalárskej práce.

Kapitola 4

Výsledky predikcií na testovacích dátach

V tejto kapitole porovnám presnosť predikcií niektorých perspektívnych modelov z predošlej kapitoly na testovacej množine dát. Následne vyberiem dva, pomocou ktorých bude estimácia veku prebiehať vo výslednom programe.

Na výkon jednotlivých modelov som sa pozrela pomocou metrík: ROC krivka, f1 score s mikro spriemerovaním, AUC a confusion matrix.

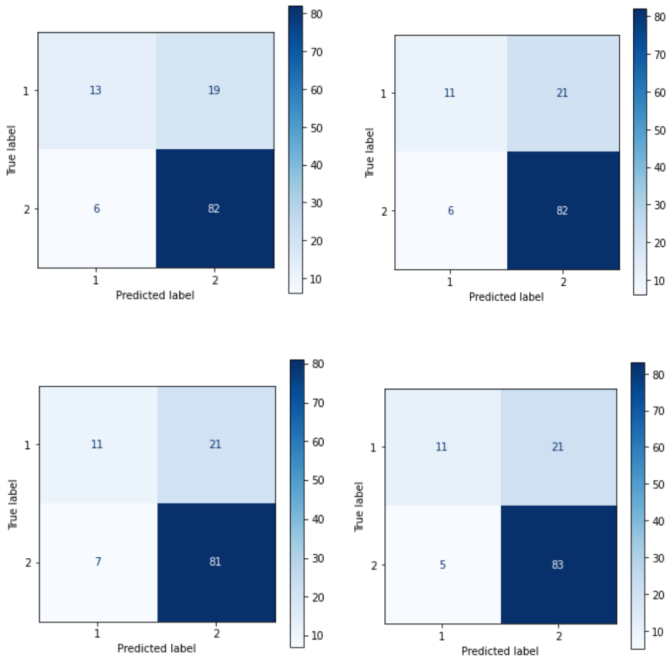
4.1 Rozdelenie dát na dve triedy

Výsledky jednotlivých modelov v rámci tohto rozdelenia boli rôzne, preto som považovala za dôležité ich porovnať pomocou viacerých metrík.

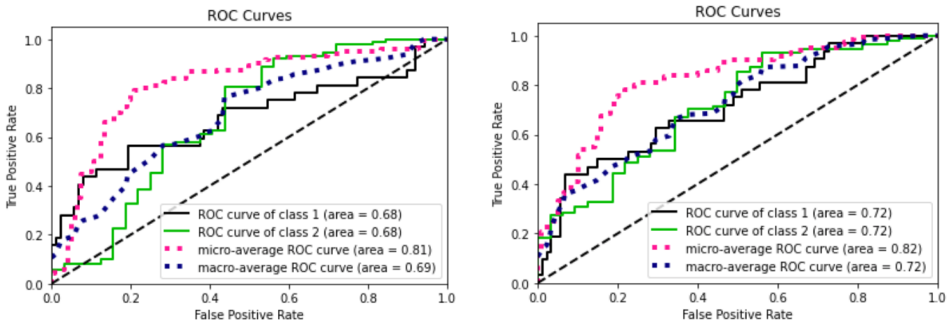
Klasifikátor	f1 score	AUC
LogisticRegression	0.792	0.676
LogisticRegression RS	0.775	0.675
RandomForest	0.767	0.702
RandomForest GS	0.783	0.716

Tabuľka 4.1: Výsledky f1 score a AUC modelov na testovacej množine dát, pri rozdelení dát na dve triedy.

Z tabuľky 4.1 je vidieť, že model RandomForest vykazoval lepšie AUC výsledky ako LogisticRegression. Pre zobrazenie detailnejších údajov o predikciách som použila confusion matrix na obrázku 4.1, z ktorého vyplýva, že modely sú presnosťou predikcií veľmi porovnateľné. Z väčšinovej triedy „2“ boli správne klasifikované takmer všetky vzorky. Výsledný model som vybrala pomocou ROC krivky na obrázku 4.2, porovnala som modely LogisticRegression a RandomForest GS, ktoré som na základe predošlých metrík vyhodnotila ako najlepšie.



Obr. 4.1: Confusion matrix z testovacej množiny dát pre jednotlivé modely. Hore vľavo LogisticRegression, vpravo LogisticRegression RS. Dole vľavo RandomForest, vpravo RandomForest GS.



Obr. 4.2: ROC krivka, hodnota AUC pre modely LogisticRegression (vľavo), RandomForest GS (vpravo) na testovacej množine dát.

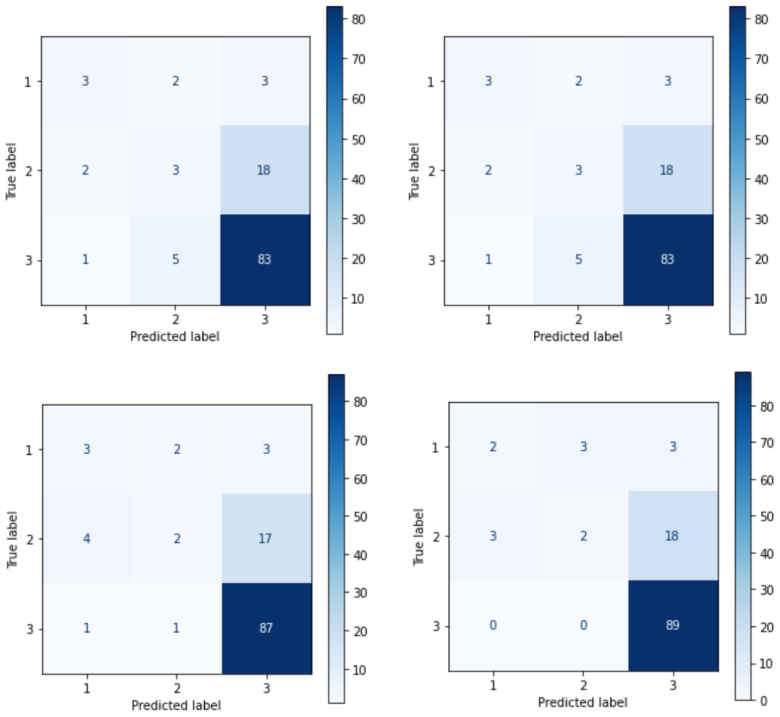
Z grafov na obrázku 4.2 som sa rozhodla vo výslednom programe na klasifikáciu veku do dvoch kategórií použiť **RandomForest GridSearch**, nakoľko jeho AUC skóre s mikro aj makro spriemerovaním sa ukázalo ako najperspektívnejšie, ako aj jeho jednotlivé AUC skóre pre každú z kategórií.

4.2 Rozdelenie dát na tri triedy

Výsledky obsiahnuté v tabuľke 4.2 poukazujú na to, že modely sú veľmi vyrovnané, takmer identické. Modely RandomForest dosahovali lepšie výsledky, a jeden od druhého sa líšili len jemne. Bližšie informácie o predikciách som pozorovala v confusion matrix na obrázku 4.3.

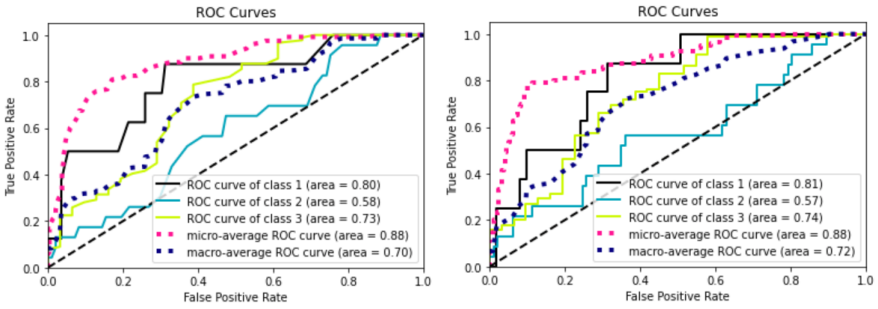
Klasifikátor	f1 score	AUC
LogisticRegression	0.742	0.608
LogisticRegression RS	0.742	0.608
RandomForest	0.767	0.705
RandomForest GS	0.775	0.723

Tabuľka 4.2: Výsledky f1 score a AUC modelov na testovacej množine dát, pri rozdelení dát na tri triedy.



Obr. 4.3: Confusion matrix z testovacej množiny dát rozdelenej na tri triedy pre jednotlivé modely. Hore vľavo LogisticRegression, vpravo LogisticRegression RS. Dole vľavo RandomForest, vpravo RandomForest GS.

Všetky modely nevykazovali vysokú presnosť v odhadoch vzoriek patriacich do prvej triedy. Odhady v druhej triede neboli presnejšie, modely pred-



Obr. 4.4: ROC krivka, hodnota AUC pre modely RandomForest (vľavo), RandomForest GS (vpravo) na testovacej množine dát.

ikovali vo väčšine prípadov nesprávne. V tretej kategórií dosahovali modely najvyššiu presnosť, čo môže byť spôsobené aj veľkým množstvom pôvodných vstupných dát patriacich práve do tejto kategórie. Na základe údajov vyplývajúcich z obrázka 4.4 som sa rozhodla opäť pre **RandomForest GridSearch** aj pre klasifikáciu do 3 tried intervalov, nakoľko AUC skóre tohto modelu je najvyššie spomedzi ostatných.

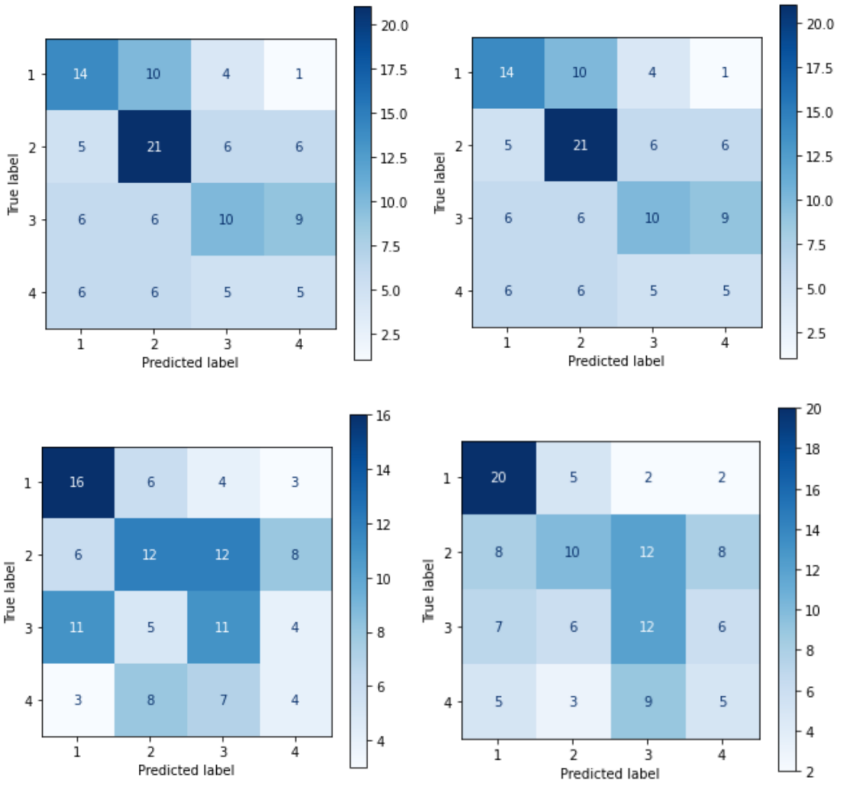
4.3 Rozdelenie dát na štyri triedy

Toto rozdelenie nebolo zahrnuté do výsledného softvéru. Preto som len porovnala jednotlivé výsledky. Z tabuľky 4.3 som pozorovala výrazne nižšie f1 score ako v predchádzajúcich rozdeleniach, naopak AUC skóre je porovnateľné. Predikcie lepšie vystihujú confusion matrix na obrázku 4.5.

Klasifikátor	f1 score	AUC
LogisticRegression	0.417	0.620
LogisticRegression RS	0.417	0.620
RandomForest	0.358	0.637
RandomForest GS	0.392	0.640

Tabuľka 4.3: Výsledky f1 score a AUC modelov na testovacej množine dát, pri rozdelení dát na štyri triedy.

Z confusion matrix na obrázku 4.5 je zrejmé ako modely chybovali a o koľko tried sa mýlili. V diagonále, idúcej z ľavého horného rohu, sú zaznamenané hodnoty, kedy sa predikcie jednotlivých modelov zhodovali so skutočnými hodnotami. Pozorovala som, že predikcie sa vo väčšine líšili od skutočných hodnôt o jednu triedu. Najpresnejší v predikciách prvého intervalu bol model RandomForest GS, v odhade vzoriek patriacich do druhého intervalu sa naopak darilo logistickým regresiám. Predikcie tried tretieho a štvrtého in-



Obr. 4.5: Confusion matrix z testovacej množiny dát rozdelených do 4 tried predstavujúcich intervaly pre jednotlivé modely. Hore vľavo LogisticRegression, vpravo LogisticRegression RS. Dole vľavo RandomForest, vpravo RandomForest GS.

tervalu boli porovnateľné naprieč všetkými modelmi, avšak RandomForest v predikciách príslušnosti do tretieho intervalu chyboval s rozdielom dvoch tried až jedenásťkrát.

Kapitola 5

Testovanie a validácia

Počas každej z fáz vývoja softvérového produktu dochádzalo k určitej forme testovania či validácie. Počas analýzy išlo o ujasnenie správnosti požiadaviek, v nasledujúcej fáze som validovala, že všetky požiadavky boli v návrhu zapracované a zároveň, že všetky vybraté technológie a modely strojového učenia sú vhodné na použitie pri riešení problému estimácie veku jedincov. Počas implementácie grafického rozhrania išlo najmä o test funkcionality, kedy som testovala rôzne priechody programom a ako program reaguje na neplatné operácie či vstupy. V závere som program poslala niekoľkým potencionálnym používateľom, za účelom overiť používateľskú zrozumiteľnosť a jednoduchosť ovládania.

5.1 Validácia splnenia požiadaviek

Cieľom testovania návrhu, prípadne už výsledného produktu, je overenie splnenia všetkých požiadaviek, ktoré sú na program kladené. Požiadavky na softvér som opísala v časti 2.3 a ich splnenie som zhrnula do tabuľky 5.1.

Požiadavka	Splnenie
F1	splnená
F2	splnená
F3	splnená
F4	splnená
F5	splnená
N1	splnená
N2	čiastočne splnená

Tabuľka 5.1: Vyhodnotenie splnenia funkčných a nefunkčných požiadaviek.

Všetky funkčné požiadavky F1 – F5 boli splnené pomocou nástrojov grafického rozhrania. Ich funkčnosť som overila v teste funkcionality programu,

ktorý som opísala v nasledujúcej časti.

Splnenie nefunkčnej požiadavky N1 vychádzalo z hodnotenia oslovenými používateľmi, celý proces a výsledky som opísala v časti 5.3. Pri požiadavke N2 je hodnotenie splnenia komplexnejšie. Ak súčasné modely strojového učenia nahradíme modelmi, ktoré prijímajú na vstupe dáta v rovnakom formáte ako doterajšie modely, požiadavka bude splnená. Avšak ak nový model, bude prijímať iný formát dát, bude nutné urobiť v programe zmeny na viacerých miestach.

5.2 Testovanie funkcionality programu

Pri testovaní funkcionality programu som vyskúšala niekoľko rôznych scenárov a priechodov programom tak, aby som pokryla všetky možné situácie. Táto časť testovania spočívala aj v ošetrení niektorých operácií, ktoré mohli nastať vplyvom činnosti používateľa.

Na to, aby mohla prebehnúť estimácia veku, je nutné načítať obrázok. Podľa F1 uvedenej v časti 2.3 na vstupe je vždy práve jeden obrázok vo formáte PNG. Z toho vyplýva, že operácie ako načítanie neexistujúceho súboru, či súboru v nesprávnom formáte, nie sú validné. Je nutné používateľa po prevedení týchto operácií upozorniť a ďalšie kroky, ako zadanie pohlavia a zvolenie klasifikácie znefunkčniť, kým vstup nie je zadaný korektne. Toto ošetrenie som implementovala, čím som splnila požiadavku F1. Následne sa načítaný obrázok používateľovi zobrazí – splnenie požiadavky F2. Pokiaľ používateľ disponuje ďalšími informáciami ako sú pohlavie a strana kosti, vie ich pomocou grafického rozhrania programu doplniť – splnenie F3. Grafické rozhranie pracuje s možnosťou zakázať určitú akciu, kým nie sú splnené konkrétne podmienky. Napríklad samotná estimácia veku nemôže nastať kým nie je načítaný obrázok a zvolený druh klasifikácie. Používateľovi je umožnené vybrať zaradenie do dvoch a/alebo troch tried, čím je uspokojená požiadavka F4. Výsledkom je pravdepodobnosť príslušnosti vzorky do triedy podľa zvolenej klasifikácie – splnená požiadavka F5.

Program som priebežne testovala počas celej doby vývoja. Všetky chyby, ktoré sa vyskytli som opravila a ošetrila. Okrem prípadov spomenutých vyššie, program nevykazoval žiadne ďalšie problémy alebo náhle ukončenia, preto som testovanie funkcionality programu považovala za úspešné.

5.3 Testovanie používateľského prostredia

Testovanie užívateľského prostredia programu prebiehalo za pomoci 10 oslovených používateľov, ktorí mali za úlohu si program vyskúšať a zhodnotiť jeho použiteľnosť. Cieľom bolo zistiť či je používanie aplikácie intuitívne a používateľsky prívetivé a či používatelia nenarazia na situáciu, kedy by sa aplikácia nečakane ukončila.

Používatelia sa zhodli, že aplikácia a jej použitie je intuitívne. Nemali problém s načítaním vstupu ani výberom klasifikácie a získaním výsledkov. Žiaden z používateľov nezaznamenal nečakané ukončenie programu. Niekoľkí používatelia sa sťažovali na dlhšie čakanie po spustení programu. Tento problém je legitímny ale vzhľadom na veľkosť a množstvo podporných knižníc a balíčkov jazyka Python a ich následnú správnu konfiguráciu, je menšie zdržanie na niektorých počítačoch pochopiteľné a nedá sa mu vyhnúť. Používatelia tiež ocenili zrozumiteľnosť používateľskej príručky, ktorá bola podľa nich stručná, zrozumiteľná a obsahovala dostatočné množstvo informácií potrebných na použitie programu.

Záver

Hlavným cieľom mojej bakalárskej práce bolo navrhnúť model strojového učenia, ktorý na základe obrázku plochy lonovej kosti odhadne vek dospelého jedinca a navrhnutý model následne implementovať. Výsledkom je spustiteľný program, ktorý poskytuje používateľovi ľahko ovládateľné grafické rozhranie, pomocou ktorého môže vybrať obrázok, doplniť o ňom dodatočné údaje a vo výsledku dostane predikciu veku jedinca. Cieľ bakalárskej práce preto považujem za splnený.

Počas písania práce som si vyskúšala prechod životným cyklom vývoja softvérového produktu. Najprv som popísala teoretické základy k problematike strojového učenia, následne som sa venovala analýze a príprave vstupných dát, ako aj požiadavkám na softvér, z ktorých som vychádzala aj pri výbere použitých technológií. Ďalej som popísala a navrhla niekoľko modelov strojového učenia, pokračovala som optimalizáciou ich hyperparametrov a najperspektívnejšie modely som otestovala a porovnala s cieľom vybrať ten najpresnejší. Potom som implementovala používateľské rozhranie a zasadila doň výsledný predikčný model. Počas celej doby vývoja som program priebežne testovala a opravovala zistené nedostatky.

Práca zahŕňa výsledky predikčných schopností niekoľkých modelov. Tieto výsledky poukazujú na to, že automatizované predikovanie veku jedincov na základe snímky panvovej kosti je realizovateľné, ale pre dosiahnutie presnejších predikcií je potrebné ďalšie testovanie na oveľa väčšej množine vstupných dát. Väčšina z použitých modelov na trénovacej množine dát vykazovala podstatne lepšie výsledky ako potom na testovacej množine, čo by mohol prísun nových vzoriek vyriešiť. Ďalšou možnosťou ako dosiahnuť presnejšie predikcie je využitie konvolučných neurónových sietí alebo iného komplexnejšieho predikčného modelu. Možnosť zlepšenia vidím aj v rozsiahlejšom predspracovaní dát, prípadne v možnosti vymedzenia problému ako regresnej úlohy.

Literatúra

[1] MOHAJON, Joydwip. *Confusion Matrix for Your Multi-Class Machine Learning Model.* In: Towards data science [online]. [cit. 2021-04-04]. Dostupné z: https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826

[2] AGARWAL, Rahul. *The 5 Classification Evaluation metrics every Data Scientist must know.* Towards Data Science [online]. [cit. 2021-04-05]. Dostupné z: https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226.

[3] NARKHEDE, Sarang. *Understanding AUC - ROC Curve.* Towards Data Science [online]. [cit. 2021-04-05]. Dostupné z: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

[4] *Receiver Operating Characteristic.* Scikit-learn [online]. [cit. 2021-04-04]. Dostupné z: https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html?highlight=roc+auc

[5] SHMUELI, Boaz. *Multi-Class Metrics Made Simple, Part II: the F1-score.* Towards Data Science [online]. [cit. 2021-04-06]. Dostupné z: https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1

[6] *Cross-validation: evaluating estimator performance.* Scikit-learn [online]. [cit. 2021-04-06]. Dostupné z: https://scikit-learn.org/stable/modules/cross_validation.html

[7] *Young or Old?* In: National museum of natural history [online]. [cit. 2021-04-04]. Dostupné z: https://naturalhistory.si.edu/education/teaching-resources/written-bone/skeleton-keys/young-or-old

[8] *SPD-Symphysis Pubis Dysfunction.* In: The Whitchurch Clinic [online]. [cit. 2021-04-04]. Dostupné z: https://www.thewhitchurchclinic.co.uk/2013/04/13/spd-symphysis-pubis-dysfunction/

[9] TODD, T. Wingate. *Age changes in the pubic bone. I. The male white pubis.* American Journal of Physical Anthropology, 1920, 3.3: s. 285-334.

[10] BROOKS, Sheilagh; SUCHEY, Judy M. *Skeletal age determination based on the os pubis: a comparison of the Acsádi-Nemeskéri and Suchey-Brooks methods.* Human evolution, 1990, 5.3: s. 227-238.

[11] PRIYA, Ekta. *Methods of Skeletal Age Estimation used by Forensic Anthropologists in Adults: A Review* [online]. 2017, 4(2) [cit. 2021-04-06]. ISSN 24692794. Dostupné z: doi:10.15406/frcij.2017.04.00104

[12] SHERIDAN, Susan G. *Pubic Symphysis Morphology.* The Byzantine St. Stephen's Project [online]. [cit. 2021-04-04]. Dostupné z: https://www3.nd.edu/~stephens/pubsymphysis.html

[13] SPECIFICATION, StereoLithography Interface. *3D Systems.* Inc., October, 1989, 22.

[14] *Python leads the 11 top Data Science, Machine Learning platforms: Trends and Analysis.* KDnuggets [online]. [cit. 2021-04-04]. Dostupné z: https://www.kdnuggets.com/2019/05/poll-top-data-science-machine-learning-platforms.html

[15] *PySimpleGUI v4.38.0* [software]. 2021. Dostupné z: https://github.com/PySimpleGUI/PySimpleGUI

[16] *Jupyter notebook* [software]. 2015. Dostupné z: https://jupyter.org/index.html

[17] PEDREGOSA, Fabian, et al. *Scikit-learn: Machine learning in Python.* the Journal of machine Learning research, 2011, roč. 12: s. 2825-2830.

[18] REBACK, Jeff, et al. *pandas-dev/pandas: Pandas 1.0.3.* Zenodo, 2020. Dostupné z: https://doi.org/10.5281/zenodo.3509134

[19] HUNTER, John D. *Matplotlib: A 2D Graphics Environment* [online]. 2007, roč. 9, č. 3, s. 90-95 [cit. 2021-04-04]. ISSN 1521-9615. Dostupné z: doi:10.1109/MCSE.2007.55

[20] REIICHIRO Nakano. *reiinakano/scikit-plot: v0.2.1* [software]. 2017. [cit. 2021-04-04]. Dostupné z: https://doi.org/10.5281/zenodo.293191

[21] BRADSKI, Gary. The OpenCV Library. Dr Dobb's J. Software Tools, 2000, 25: 120-125.

[22] VAN DER WALT, Stéfan, et al. *Scikit-image: image processing in Python.* PeerJ [online]. 2014, 2 [cit. 2021-04-04]. ISSN 2167-8359. Dostupné z: doi:10.7717/peerj.453

[23] HARRIS, Charles R., et al. *Array programming with NumPy.* Nature, 2020, 585.7825: 357-362.

[24] LEMAÎTRE, Guillaume; NOGUEIRA, Fernando; ARIDAS, Christos K. *Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning.* The Journal of Machine Learning Research, 2017, 18.1: 559-563.

[25] STOJILJKOVIĆ, Mirko. *Logistic Regression in Python.* Real Python [online]. [cit. 2021-04-06]. Dostupné z: https://realpython.com/logistic-regression-python/

[26] *Nearest Neighbors Classification.* Scikit-learn [online]. [cit. 2021-04-06]. Dostupné z: https://scikit-learn.org/stable/modules/neighbors.html

[27] GÉRON, Aurélien. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.* Boston: O'Reilly, 2017. ISBN 978-1-4919-6229-9.

[28] *PyInstaller*. Dostupné z: http://www.pyinstaller.org/

Dodatok **A**

Zoznam použitých skratiek

AI Artificial Intelligence

ML Machine Learning

MSE Mean Square Error

MAE Mean Absolute Error

TP True Positives

TN True Negatives

FP False Positives

FN False Negatives

TPR True Positive Rate

FPR False Positive Rate

AUC Area Under Curve

ROC Receiver Operating Characteristics

STL Standard Triangle Language

PNG Portable Network Graphics

SMOTE Synthetic Minority Oversampling Technique

KNN K Nearest Neighbour

GS GridSearchCV

RS RandomizedSearchCV

Dodatok B

Používateľská príručka

Používanie na operačnom systéme Windows 10

Používatelia majúci operačný systém Windows nemusia pre beh programu urobiť nič špeciálne iba ho spustiť. Všetky moduly a potrebné knižnice sú už súčasťou *estimator_win.exe* súboru. Po spustení, nastáva pár sekundové otváranie a potom je už program pripravený na používanie.

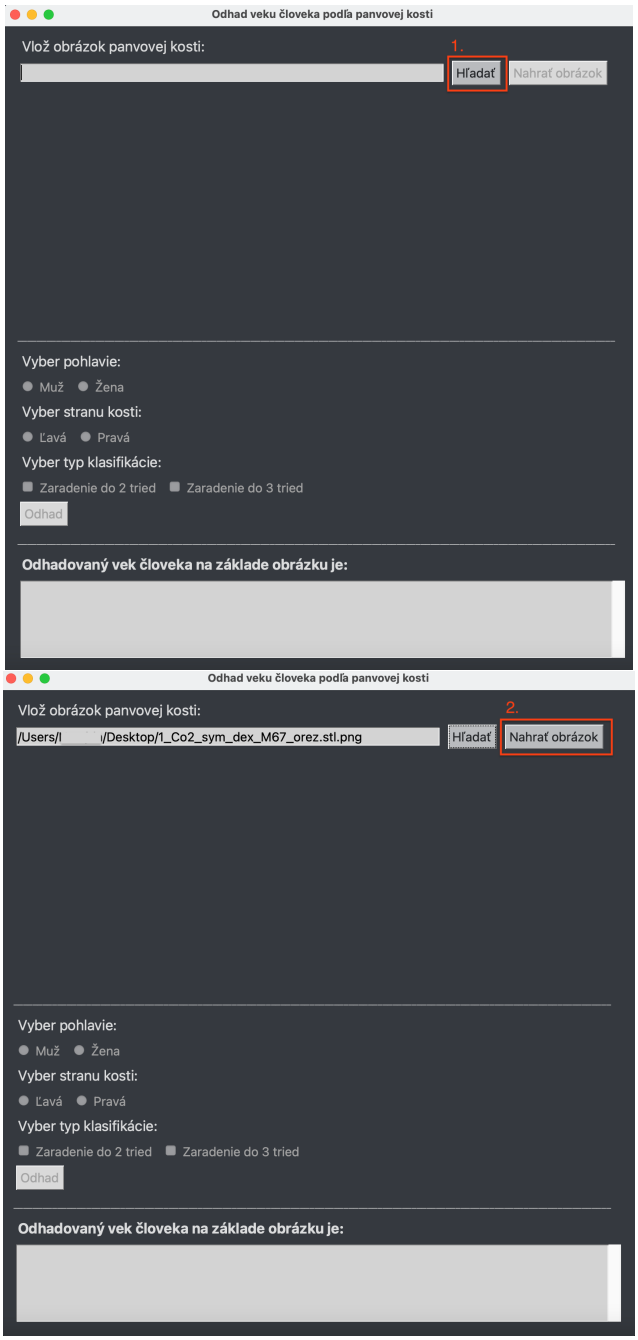
Používanie na iných operačných systémoch - UNIX based

Podobne ako pri operačnom systéme Windows, program na svoj beh nepotrebuje inštalovať žiadne balíčky predom. Pomocou príkazovej riadky (angl. Command Line Interface, CLI) sa presunieme do súboru, v ktorom sa nachádza spustiteľný skript *estimator_linux*. Následne program spustíme zadaním príkazu *./estimator_linux*. Tento postup je otestovaný na OS Ubuntu 20.04, avšak nemal by sa priveľmi líšiť od postupu na iných linuxovo-orientovaných operačných systémoch.

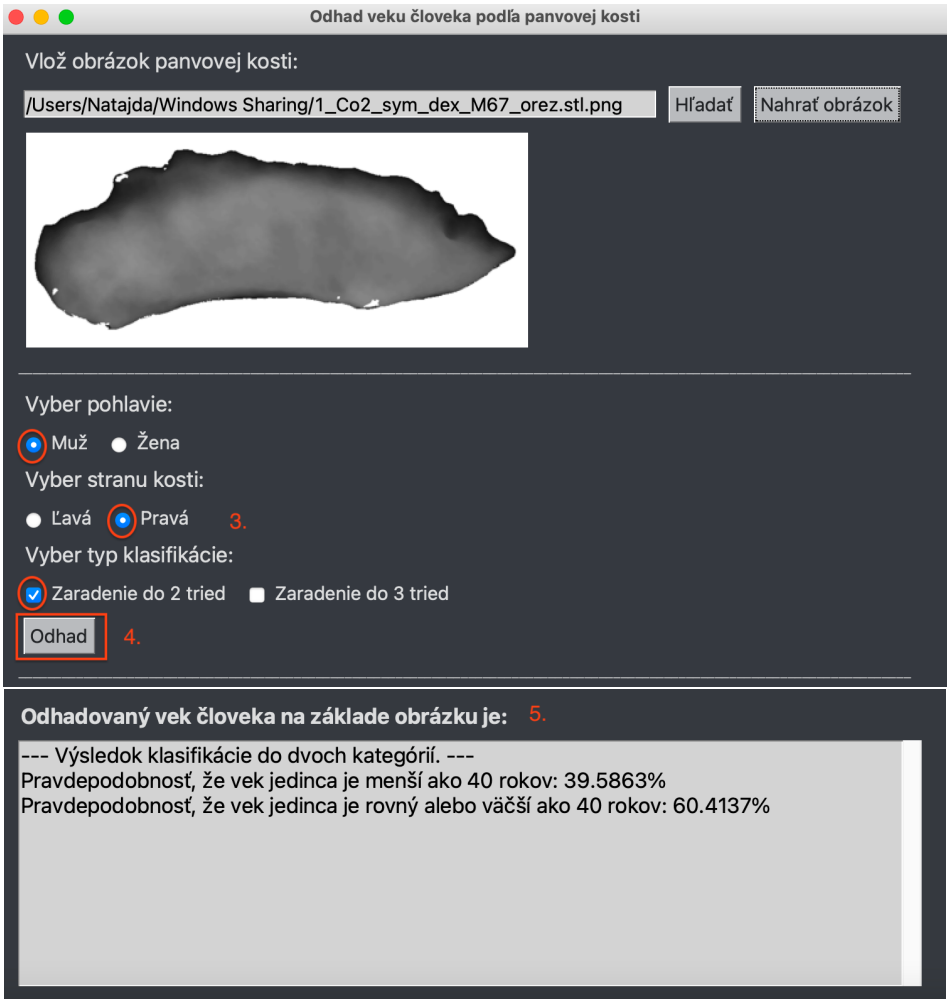
Ukážka použitia programu

Po zapnutí je nutné vyhľadať scan, ktorý chce používateľ načítať, a to pomocou tlačidla *Hľadať*. Následne keď je s výberom vstupného súboru spokojný, potvrdí výber tlačidlom *Nahrať obrázok*. Potom sa zobrazí nahraný obrázok.

V ďalšom kroku môže používateľ doplniť údaje, ktoré o kosti na obrázku vie. Pohlavie a strana kosti sú údaje, ktoré sú voliteľné a pokiaľ ich používateľ nepozná, nemusí ich uviezť. Povinný je však údaj o type klasifikácie. Je možné vybrať práve jednu alebo dve možnosti zaradenia vzorky do triedy. Obrázok s dodatočnými údajmi je pripravený na klasifikáciu, čo používateľ potvrdí kliknutím na tlačidlo *Odhad*. Odhadovaná pravdepodobnosť príslušnosti do danej triedy je zobrazená používateľovi.



Obr. B.1: Ukážka postupu ako vyhľadať a načítať vstupný obrázok vo formáte PNG.



Obr. B.2: Grafické rozhranie programu a postup ako vybrať a označiť dodatočné informácie k obrázku – pohlavie a stranu kosti. Príklad výstupu, ktorý poskytuje program ako odhad príslušnosti do daných kategórií.

Dodatok C

Obsah priloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresár so spustiteľnou formou implementácie na rôznych OS
- `src`
  - `impl` ..... zdrojové kódy implementácie
    - `jup` ..... analýza dát, trénovanie modelov v jupyter notebookoch
    - `dev` ..... skripty výsledného softvéru
  - `thesis` ..... zdrojová forma práce vo formáte LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce vo formáte PDF
  - `thesis.ps` ..... text práce vo formáte PS