VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

KLASIFIKACE HUDEBNÍCH SOUBORŮ POMOCÍ STROJOVÉHO UČENÍ

CLASSIFICATION OF MUSIC FILES USING MACHINE LEARNING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MATYÁŠ SLÁDEK**
AUTHOR

VEDOUCÍ PRÁCE — **Doc. Ing. VLADIMÍR JANOUŠEK, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Sládek Matyáš**
Program: Informační technologie
Název: **Klasifikace hudebních souborů pomocí strojového učení**
Classification of Music Files Using Machine Learning
Kategorie: Umělá inteligence

Zadání:

1. Prostudujte způsoby extrakce metadat z hudebních souborů (BPM, spektrogram, apod.). Seznamte se s existujícími knihovnami vhodnými pro extrakci těchto metadat.
2. Seznamte se s metodami klasifikace hudebních souborů. Prostudujte vhodné algoritmy pro klasifikaci.
3. Navrhněte způsob získání a zpracování metadat pro potřeby klasifikace hudebních souborů z rozsáhlého lokálního hudebního archivu s ohledem na možnost tvorby playlistů, například podle hudebního a tanečního stylu.
4. Navržené prostředky realizujte, ověřte jejich praktickou použitelnost a vyhodnoťte dosažené výsledky.

Literatura:

- Music Information Retrieval URL: https://musicinformationretrieval.com/
- Audio Content Analysis URL: https://www.audiocontentanalysis.org/
- Neural Networks and Deep Learning URL: http://neuralnetworksanddeeplearning.com/

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Janoušek Vladimír, doc. Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 31. července 2020
Datum schválení: 31. října 2019

Abstrakt

Tato práce se zabývá klasifikací hudebních souborů pomocí algoritmů strojového učení. V práci bylo porovnáno sedm klasifikačních algoritmů z hlediska úspěšnosti klasifikace a rychlosti zpracování na třech datových sadách. Využity byly dvě metody pro extrakci atributů, dvě metody pro selekci atributů a dvě metody optimalizace parametrů. Nejvíce se osvědčil model *XGBClassifier*, který dosáhl úspěšnosti klasifikace 87.56 % na datové sadě „Extended Ballroom Dataset“, 64.56 % na datové sadě „FMA: A Dataset For Music Analysis“ a 83.50 % na datové sadě „GTZAN“. Tento model může být využit při tvorbě seznamů skladeb či kategorizaci hudební databáze.

Abstract

This thesis is focused on classification of music files using machine learning algorithms. Seven classifiers were compared in this thesis, based on classification accuracy and speed. Two feature extraction methods, two feature selection methods and two parameter optimization methods were used. The best classifier proved to be *XGBClassifier*, which had reached accuracy of 87.56 % on dataset „Extended Ballroom Dataset“, 64.56 % on dataset „FMA: A Dataset For Music Analysis“ and 83.50 % on dataset „GTZAN“. This model could be used for playlist creation or music database categorization.

Klíčová slova

strojové učení, předzpracování dat, extrakce atributů, selekce atributů, optimalizace parametrů, klasifikace

Keywords

machine learning, data preprocessing, feature extraction, feature selection, parameter optimization, classification

Citace

SLÁDEK, Matyáš. *Klasifikace hudebních souborů pomocí strojového učení.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. Ing. Vladimír Janoušek, Ph.D.

Klasifikace hudebních souborů pomocí strojového učení

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. Ing. Vladimíra Janouška Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Matyáš Sládek
29. července 2020

Poděkování

Rád bych poděkoval panu Doc. Ing. Vladimíru Janouškovi Ph.D. za skvělé vedení a cenné rady, které mi poskytl při tvorbě této bakalářské práce.

Obsah

Kapitola 1

Úvod

Rozvoj informačních technologií umožnil uchovávat a šířit hudební skladby zcela novým způsobem. Lidé tak z praktických důvodů postupně upouštěli od fyzických médií a shromažďovali skladby do hudebních databází v digitální formě. Tyto databáze je vhodné přehledně organizovat, například rozdělením skladeb do kategorií na základě jejich hudebního obsahu. Tato kategorizace je velmi důležitá obzvláště z pohledu tvorby takzvaných „playlistů“, tedy seznamů skladeb s podobným obsahem, se kterými se můžeme setkat například u online streamovacích databází jako Spotify či Tidal.

Anotace obsahu hudebních souborů je v dnešní době prováděna ručně. Tato práce se zabývá návrhem a implementací systému pro automatickou anotaci hudebních skladeb pomocí analýzy jejich obsahu.

V kapitole 2 jsou čtenáři přiblíženy základní informace nutné k pochopení problematiky klasifikace digitálních hudebních souborů. V úvodních podkapitolách jsou popsány některé ze základních veličin popisujících zvukové vlnění. Přiblíženy jsou také způsoby záznamu a uchování zvukových stop a některé z pojmů hudební teorie. Závěr kapitoly se pak věnuje způsobům kategorizace hudebních databází a také použitým metodám a dosaženým výsledkům u několika z dostupných studií v oboru klasifikace hudebních souborů.

Kapitola 3 přibližuje čtenáři vybrané způsoby předzpracování dat vhodné pro klasifikaci hudebních souborů. Popsány jsou metody extrakce vhodných atributů ze zvukových stop a také metody selekce nejvhodnějších z těchto atributů pro efektivní zpracování klasifikačními algoritmy. Na konci kapitoly jsou popsány nejpoužívanější z metod kódování a škálování dat.

Čtvrtá kapitola 4 seznamuje čtenáře s přístupy ke klasifikaci dat a zmíněny jsou i některé z nejznámějších klasifikačních algoritmů. Zmíněny jsou pravděpodobnostní metody, rozhodovací stromy, metody učení založené na instancích, metody podpůrných vektorů a neuronové sítě. V poslední podkapitole 4.6 této kapitoly jsou pak přiblíženy přístupy k vytváření takzvaných „ensemble metod“, což jsou metody kombinující několik klasifikátorů dohromady.

V kapitole 5 jsou popsány některé z volně dostupných datových sad, které lze pro trénování modelů využít. Dále jsou popsány využité způsoby předzpracování dat, zahrnující kódování a škálování dat, extrakci atributů a selekci atributů. Zmíněn je také výběr klasifikačních algoritmů a nakonec je popsán způsob optimalizace parametrů těchto algoritmů.

V předposlední kapitole 6 je pak vyhodnocena úspěšnost klasifikace na testovacích datech všech tří datových sad a v závěrečné 7. kapitole je pak shrnuta provedená práce a možnosti budoucího vývoje.

Kapitola 2

Úvod do problematiky

Tato kapitola pojednává o základních vlastnostech zvuku a způsobech jeho záznamu a uchování. Dále pak vysvětluje základní pojmy hudební teorie používané v následujících kapitolách a přiblíženy jsou i způsoby kategorizace hudebních souborů na základě jejich obsahu. Nakonec je pak zmíněno několik z dosavadních prací na téma klasifikace muziky pomocí strojového učení.

2.1 Zvuk

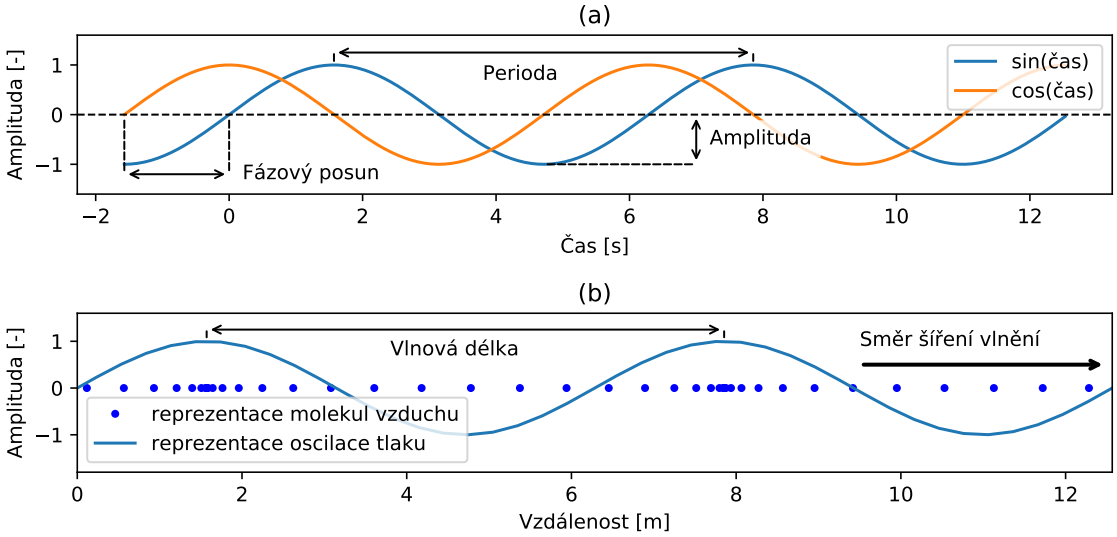
Zvuk je možné popsat jako změnu tlaku v závislosti na čase a šíří se jako mechanické vlnění (oscilace molekul) prostřednictvím elastického média, jako je plyn, kapalina, pevná látka nebo plazma. Ve všech těchto skupenstvích se zvuk šíří jako podélné (také longitudální či tlakové) vlnění, při kterém dochází k oscilaci částic ve směru rovnoběžném se směrem šíření vlnění. V pevném skupenství se však dle některých zdrojů může zvuk šířit i jako vlnění příčné, při kterém dochází k oscilaci částic ve směru kolmém na směr šíření vlnění.[12][23]

Zvukové vlny lze popsat pomocí následujících vlastností:

- Perioda - udává délku trvání jednoho cyklu vlny. Její jednotkou je obvykle sekunda a značí se T.
- Frekvence - inverzní hodnota periody, udává počet cyklů za jednu sekundu. Její jednotkou je Hertz a značí se f.
- Amplituda - u zvukových vln značí rozsah pohybu částic vůči jejich rovnovážné poloze.
- Fáze - udává pozici počátku cyklu vlny.
- Rychlost zvuku - udává rychlost pohybu zvukových vln v médiu, měří se obvykle v metrech za sekundu a značí se c.
- Vlnová délka - udává vzdálenost, kterou vlna překoná za dobu jedné její periody, měří se obvykle v metrech a značí se λ.

Znázornění zvukových vln je možné vidět v obrázku 2.1. Lidské ucho je citlivé na frekvence zvuků mezi přibližně 20 Hz a 20 kHz a změnu frekvence zvuků vnímá spíše logaritmicky, než lineárně. Stejně jako u změn frekvence je lidské ucho citlivé i na změny tlaku

logaritmicky, proto se používá pro vyjádření hlasitosti jednotka logaritmické stupnice decibel (dB). Decibel je bezrozměrná jednotka vyjadřující poměr dvou vstupních veličin. U hlasitosti zvuků přenášených vzduchem se jako referenční vstupní hodnota používá nejnižší lidmi slyšitelná změna tlaku vzduchu.[23][1]



Obrázek 2.1: **Grafy zvukových vln.** (a) Graf funkcí sinus a cosinus. Z grafu je patrné, že funkce kosinus je fázově posunutou funkcí sínus. (b) Ukázka podélného vlnění. Částice oscilující ve směru rovnoběžném se směrem šíření vlnění tvoří místa s vyšší koncentrací (vyšší tlak) a nižší koncentrací (nižší tlak.)

2.2 Způsoby záznamu, uchování a reprodukce zvuku

Záznam, uchování a reprodukci zvuku je možné rozdělit do dvou kategorií, a to analogové a digitální. Přes to, že dnes dominuje digitální způsob uchování dat, je stále možné setkat se s nahrávkami analogovými, které jsou důležité i z hlediska historického. V následujících odstavcích je proto krátce zmíněna i tato varianta.[26][13]

Zvukové vlny je možné zaznamenat pomocí membrány mikrofonu, která snímá změny atmosferického tlaku a následně tyto pomocí mechanického či elektronického systému transformuje na analogový či digitální signál, který zapisuje na nějaké médium. Následná reprodukce je obráceným procesem záznamu, kde je signál z média převeden pomocí membrány reproduktoru zpět na změny atmosferického tlaku.[26][13]

Analogový signál je spojitou funkcí spojitého času. Analogový signál může být uchován na médiích mechanických, jako je například vinylová deska, nebo magnetických jako je například magnetická páska. Přes to, že samotné zvukové vlny jsou analogovým signálem, v dnešní době se ve většině případů z praktických důvodů zvukové vlny uchovávají jako signál digitální.[26][13]

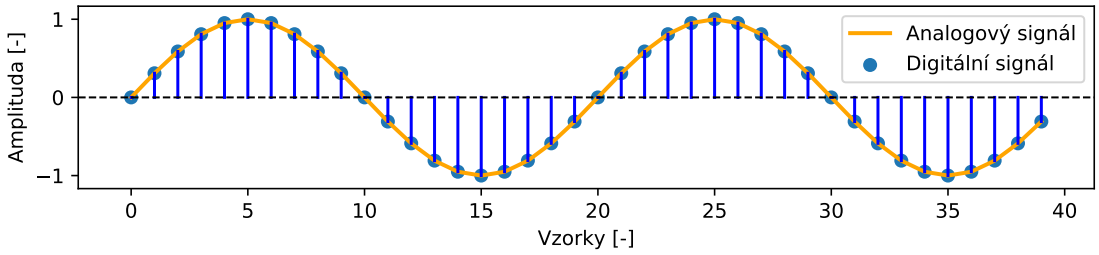
Digitální signál je oproti analogovému signálu funkcí diskrétního času, tedy je definován pouze v diskrétních časových okamžicích. Tento signál je pak reprezentován posloupností funkčních hodnot. Převod analogového signálu na digitální probíhá pomocí periodického čtení hodnot analogového signálu a následného zapisování těchto hodnot na digitální mé-

dium, jako například pevný disk. Tento proces se nazývá vzorkování a frekvence, se kterou jsou hodnoty čteny a zapisovány se nazývá *vzorkovací frekvence*. Obrázek 2.2 obsahuje ukázku vzorkovaného signálu. Dle vzorkovacího teorému je pro bezchybné vzorkování signálu s maximální frekvencí f_{max} je nutné zvolit vzorkovací frekvenci f_S tak, aby platil vztah

$$f_{max} < f_S/2. \tag{2.1}$$

Pokud tento vztah dodržen není, dochází k jevu zvanému *aliasing*, kdy při vzorkování signálu dochází k transformaci frekvencí na jiné.[26][13]

Následná reprodukce je dosažena převedením těchto hodnot zpět na analogový signál. O tyto funkce se starají převodníky *ADC (analog to digital converter)* a *DAC (digital to analog converter)*. Uchování zvukových signálů v digitální podobě má oproti podobě analogové řadu výhod, mezi které se řadí například možnost bezztrátového kopírování, menší velikost, vyšší odolnost vůči poškození, mnohem vyšší rychlost zápisu a čtení u digitálních médií a další.[26]



Obrázek 2.2: **Graf reprezentací analogového a digitálního signálu.** Graf znázorňuje rozdíl mezi analogovým signálem sinusoidy s frekvencí 400 Hz a její digitální reprezentací, kde při vzorkovací frekvenci 8000 Hz jednu periodu zachycuje 20 vzorků.

2.3 Hudba

Jednoznačná definice hudby neexistuje, obecně se však dá říci, že se jedná o uměleckou formu, jejíž médiem je soubor zvuků uspořádaný v čase. Hudba má mnoho forem, může obsahovat zpěv, zvuky hudebních nástrojů, zvuky přírody a další. V hudební teorii se zvuky dělí na hudební (tóny) a nehudební (hluky), kde tóny se vyznačují pravidelnou oscilací a hluky nepravidelnou oscilací zvukové vlny. Čistý tón obsahuje pouze jedinou frekvenci, avšak tóny produkované hudebními nástroji obsahují několik frekvencí. A to základní frekvenci a frekvence harmonické, které jsou celočíselným násobkem frekvence základní. Jako výška tónu se v hudební teorii označuje subjektivní vnímání jeho frekvence. Tóny produkují například strunné či dechové nástroje, zatím co hluky produkují převážně nástroje perkusní. V hudební teorii se hudba nejčastěji popisuje pomocí rytmu, melodie, harmonie, barvy a textury, které jsou popsány v následujícím seznamu.[23][22]

- Rytmus popisuje rozložení zvuků v čase.
- Melodie značí posloupnost několika tónů po sobě.

- Harmonie označuje několik tónů různé výšky ve stejném okamžiku.
- Barva popisuje všechny vlastnosti zvuku mimo výšku, hlasitost a dobu trvání, jako například poměr amplitud frekvencí u tónu.
- Textura popisuje komplexitu hudby, tedy kolik melodií, zvůků, nástrojů a dalších obsahuje v daném časovém okamžiku.

V hudební teorii je také možné se setkat s pojmy jako stupnice, kde u západní muziky se používá stupnice diatonická, která vychází ze stupnice chromatické. *Chromatická stupnice* obsahuje dvanáct tónů v jedné oktávě, jedná se tedy o stupnici dvanáctitónovou. Oproti tomu *diatonická stupnice* je sedmitónová, tedy obsahuje pouze vybraných sedm z dvanácti těchto tónů. Kterých sedm tónů diatonická stupnice obsahuje, určuje takzvaná *tónina*.[22]

2.4 Kategorizace hudebních skladeb

Nedílnou součástí organizace hudební databáze je rozdělení jejího obsahu do kategorií, tedy shlukování skladeb na základě určitých společných vlastností. Takováto kategorizace je vhodná hned z několika důvodů, umožňuje uživatelům snadnější orientaci v databázi, snadnější vyhledání podobných skladeb, případně umožňuje databázi vizualizovat. Uspořádání hudebních souborů do kategorií s sebou však nese mnoho úskalí. Nejprve je potřeba určit, na základě jakých vlastností je vhodné skladby kategorizovat. Skladby je možné rozdělit do kategorií například podle roku vydání, či podle jejich autora, tyto kategorie nám však o samotné hudební estetice skladby mnoho neřeknou. Z toho důvodu lidé začali přicházet s dalšími kategoriemi, zakládajících se na obsahu skladeb, tedy pocitu z poslechu, použití určitých hudebních nástrojů, tempu a tak dále. Příkladem takovýchto kategorií mohou být například hudební žánry.

Hudební žánr je dnes nejpoužívanější formou popisu hudební estetiky muziky. Lze jej popsat jako druh muziky dodržující určitá pravidla, přijímaný nějakou komunitou. Mezi vlastnosti charakterizující hudební žánry patří tempo, použité hudební nástroje, popularita muziky, regionální výskyt muziky a spousta dalších. Hudební žánr nemá žádnou formální definici a každý jej tak vnímá do jisté míry subjektivně. Z tohoto důvodu u spousty skladeb není možné jednoznačně určit, pod jaký hudební žánr spadá a často se dnes setkáváme s případy, kdy je jedna skladba označena hned několika žánry. Hudebních žánrů jsou v dnešní době tisíce, avšak spousta z nich je považována za „podžánry“ žánrů obecnějších. Například žánry Techno, Trance a House spadají pod žánr Dance. Hudební žánry vznikají zároveň s nově vznikajícími formami a styly muziky, nebo kvůli potřebě jejich přesnější klasifikace.[17][13]

Dalším možným kritériem pro klasifikaci jsou emoce, které skladby při poslechu evokují. Nejvíce zjevné, jak hudba evokuje určité emoce může být například ze zvukových stop k filmům, které významným způsobem dokreslují atmosféru scény. Tento způsob klasifikace je ale opět do značné míry subjektivní a také je propojen s žánry. Důležitými parametry pro tento způsob klasifikace jsou zejména tempo a tónina.[14][13]

Další množinou kategorií mohou být taneční styly. Tento způsob klasifikace je vhodné využít zejména u databází určených pro diskotéky nebo taneční soutěže. U klasických tanců je nejvíce směrodatný rytmus, u modernějších tanců rytmus nemusí hrát roli a jediným parametrem pro klasifikaci pak zůstává tempo.

2.5 Metody automatické klasifikace hudebních souborů

Automatické klasifikaci a doporučování podobné muziky je se stále rychleji narůstajícím počtem skladeb věnováno čím dál více pozornosti. V dostupných studiích zabývajících se touto problematikou je možné narazit převážně na dva přístupy, prvním z nich je klasifikace na základě nízkoúrovňových charakteristických dat získaných algoritmicky ze zvukové stopy a druhým je klasifikace na základě vysokoúrovňových charakteristických dat vytvořených lidmi. Oba tyto přístupy mají své výhody i nevýhody a u každého z nich byla demonstrována řada metod dosahujících různě přesných výsledků. Jelikož klasifikace na základě popisných vlastností vytvořenými lidmi je mimo rozsah této práce, tento přístup dále zmíněn není.

Klasifikace muziky podle obsahu audio stopy

S klasifikací podle obsahu audio stopy se v dostupných studiích můžeme setkat častěji. Jde o způsob klasifikace na základě informací získaných ze samotné zvukové stopy, které jsou následně předány ke zpracování klasifikačním algoritmům. Jelikož tento přístup pracuje pouze se samotnou zvukovou stopou, jednou z jeho výhod je nezávislost na datech vytvořených lidmi. Mezi jeho nevýhody však patří nutnost provádět výpočetně náročné operace extrakce směrodatných informací ze zvukové stopy a trénování klasifikačních algoritmů. Dále je pak potřeba mít k dispozici dostatečně obsáhlou datovou sadu, na které je možné klasifikátory natrénovat. V následujících odstavcích jsou zmíněny a krátce popsány dostupné výzkumy zabývající se tímto způsobem klasifikace.

Jednu z prvních prací na toto téma publikovali roku 2002 Tzanetakis a Cook. Ve své práci navrhli způsob klasifikace pomocí třech sad nízkoúrovňových atributů popisujících určitou vlastnost zvukové stopy. První sada obsahovala atributy popisující barvu zvuku, druhá rytmus a třetí výšku a tóninu. Tyto sady byly zpracovány pomocí Gaussova klasifikátoru, Gaussova smíšeného modelu a metody K-nejbližších sousedů. K vyhodnocení byly použity tři datové sady dohromady obsahující dvacet žánrů, kde každý žánr byl reprezentován stem třiceti sekundových úseků skladeb. Výsledky klasifikace je možné vidět v následujícím seznamu.[29]

- Žánry: rozdělení do 10 skupin: 61 %
- Classical: rozdělení do 4 skupin: 85 %
- Jazz: rozdělení do 6 skupin: 67 %

Gwardys a Grzywczak ve své práci využili pro klasifikaci výhradně spektrogram, čímž problém klasifikace zvukové stopy transformovali na problém klasifikace obrazu. Ke klasifikaci byly využity tři formy spektrogramu, a to původní obecný spektrogram, harmonický spektrogram a perkusní spektrogram. Pro klasifikaci byly využity konvoluční neuronové sítě. Nejlepších výsledků bylo dosaženo kombinací všech tří spektrogramů, kde pro každý z nich byl zvlášť natrénován jeden klasifikátor a výsledky všech byly vyhodnoceny pomocí metody podpůrných vektorů. Tímto způsobem bylo dosaženo 78% úspěšnosti při klasifikaci sta skladeb do desíti žánrů.[6]

Práce Murauera a Spechta porovnává více způsobů klasifikace. Jedním z nich je klasifikace pomocí konvolučních neuronových sítí pracujících se spektrogramem, druhým je klasifikace pomocí algoritmů rozhodovacích stromů, posilování gradientu a hlubokých neuronových sítí pracujících s numerickými parametry. Výsledky experimentů této práce naznačují, že pomocí klasifikátoru posilování gradientu XGBoost je možné dosáhnout ještě

přesnějších výsledků, než pomocí neuronových sítí. A to jak konvolučních pracujících se spektrogramem, tak hlubokých pracujících s numerickými parametry. Jak však autoři sami zmiňují, tyto výsledky mohly být ovlivněny nedostatkem prostředků pro kvalitnější implementaci neuronových sítí.[19]

Kaur a Kumar vybrali pro klasifikaci čtyři nízkoúrovňové vlastnosti, které na základě svého průzkumu vyhodnotili jako nejvíce přínosné. Byly to Mel-frekvenční kepstrální koeficienty, spektrální klesání, počet průchodů nulou a spektrální tok. Pro klasifikaci využili Gaussova smíšeného modelu a pro vyhodnocení výsledků využili čtyř datových sad, kde u každé z nich jejich způsob klasifikace dosáhl přibližně 70% úspěšnosti.[10]

S klasifikací do žánrů se můžeme ve výzkumech komunity získávání hudebních informací (music information retrieval, MIR) setkat nejčastěji, v některých případech je ale možné se setkat i s klasifikací do jiných kategorií, jako například emocí. Tento způsob zvolili ve své práci Lin, Yang, Chen, Liao a Ho, kde pro klasifikaci emocí nejprve využili klasifikace žánrů, neboť žánr je dle výsledků jejich experimentů s emocemi do značné míry propojen. Klasifikace tedy probíhala na dvou úrovních, kdy v první úrovni byl klasifikován žánr a v druhé emoce. Pro klasifikaci žánrů zvolili dva způsoby, a to klasifikaci jednoho žánru a klasifikaci více žánrů, kde pravděpodobnost náležitosti k určitému žánru byla vyjádřena procentuálně. U obou způsobů byly následně informace získané z klasifikace žánrů použity pro klasifikaci emocí a pro porovnání byl demonstrován i klasifikátor pracující s reálnými informacemi o žánru získaných z hudebních databází. Výsledky experimentů ukázaly, že klasifikace s využitím procentuální náležitosti k žánrům byla jednak přesnější než klasifikace s využitím jednoho žánru, ale také, že její úspěšnost se blížila výsledkům klasifikace s použitím reálných informací o žánru.[14]

Kapitola 3

Předzpracovaní dat

Předzpracování dat (data preprocessing) je jedním z nejdůležitějších kroků v oboru strojového učení. Jedná se o proces, při kterém se původní data transformují tak, aby je vybraný algoritmus byl schopný lépe interpretovat a dosáhl lepších výsledků. Předzpracování dat se skládá z několika kroků, ne vždy je však nutné provést všechny. Jaké kroky je vhodné provést záleží na formě a velikosti dat původních a na použitém algoritmu. Prvním krokem obvykle bývá kontrola kvality dat. Tento krok zahrnuje ošetření chybějících hodnot buďto jejich doplněním či odstraněním příslušných položek a také odstranění duplikátních a chybných hodnot. V následujících odstavcích jsou blíže rozebrány vhodné kroky předzpracování dat pro klasifikaci hudebních souborů.[11][13]

Přestože s využitím neuronových sítí na neupravené zvukové stopě bylo dosaženo uspokojivých výsledků, pro většinu klasifikačních algoritmů je digitální reprezentace zvukové stopy příliš komplexní na to, aby ji byly shopny efektivně zpracovat. Je tedy nejprve nutné ji vhodně upravit. Tento proces se nazývá redukce dimenzionality či snížení počtu rozměrů a dá se rozdělit do dvou hlavních kategorií. Extrakce atributů a selekce atributů. Atribut (vlastnost) je v tomto kontextu hodnota určitým způsobem popisující původní objekt. Atributy se dělí na numerické a kategorické. Numerické atributy jsou takové, jejichž hodnoty jsou spojité nebo celočíselné. Tyto atributy nemají definovanou konečnou množinu hodnot. Mezi atributy numerické patří například tempo skladby. Atributy kategorické jsou takové, které mají předem definovanou konečnou množinu hodnot. Patří mezi ně například dominantní tónina skladby.[22][20][13]

3.1 Extrakce atributů

Extrakci atributů je vhodné použít v případě, kdy původní data jsou příliš rozsáhlá nebo obsahují redundantní či irelevantní hodnoty. Extrakce atributů si klade za cíl tato data transformovat pomocí různých funkcí na data nová, méně rozsáhlá, avšak se zachováním co největšího množství informací. Přístupů k extrakci atributů existuje mnoho a vhodné způsoby závisí na konkrétní situaci. Existuje spousta algoritmů implementovaných pro tento účel, které jsou dostatečně obecné na to, aby se daly použít napříč řadou oborů. Mezi zmíněné metody patří například *Analýza hlavních komponent (Principal component analysis, PCA)* či poměrně nová metoda *Jednotná mnohočetná aproximace a projekce (Uniform manifold approximation and projection, UMAP)*.[2][16][13]

V oboru získávání hudebních informací však existuje celá řada algoritmů vytvořených specificky za účelem extrakce vhodných atributů ze zvukové stopy. Tyto atributy je pak

možné rozdělit do tříd například dle úrovně abstrakce na nízkoúrovňové a vysokoúrovňové sémantické atributy, dle časového rozsahu na okamžité, segmentové či globální, případně dle aspektů hudební teorie na atributy týkající se melodie, rytmu, harmonie a dalších. Označení proměnných v rovnicích této kapitoly jsou uvedeny v tabulce 3.1.[22][13]

Tabulka 3.1: **Označení využitá v rovnicích podkapitoly extrakce atributů**

Označení	**Popis**
$x[n]$	Hodnota amplitudy vzorku n
N	Celkový počet vzorků signálu
$X[k]$	Hodnota amplitudy frekvence k
K	Celkový počet frekvencí spektra
r	Označení okna r
s	Označení filtračního pásma s

Časová a frekvenční oblast

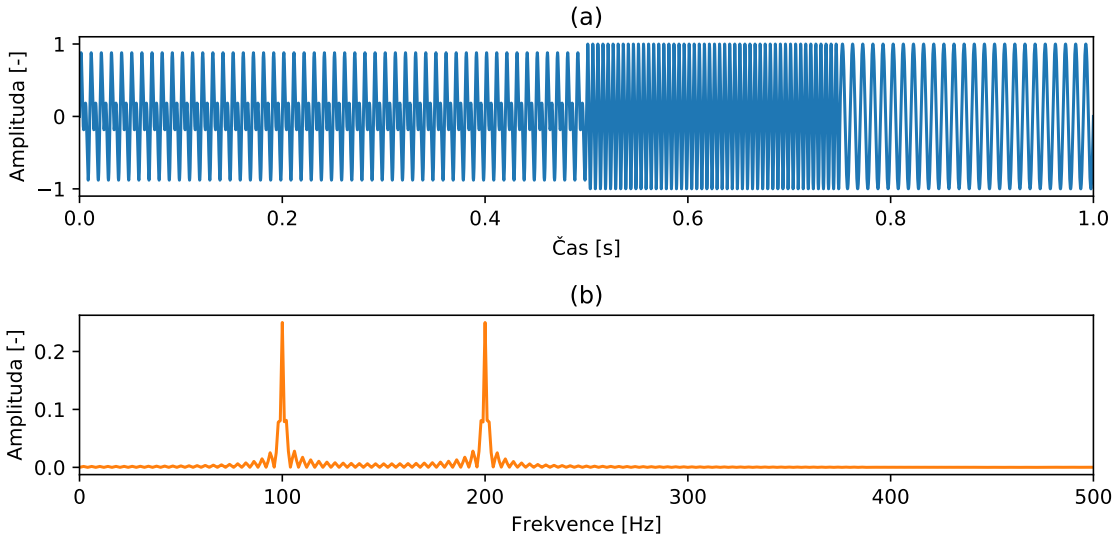
Digitální reprezentace zvukové stopy je funkcí času a jedná se tedy o reprezentaci v časové oblasti. Tato reprezentace zvukové stopy je vhodná pro extrakci atributů týkajících se vývoje amplitudy zvukové vlny v čase. Frekvenční spektrum signálu v časové oblasti je pak reprezentací tohoto signálu ve frekvenční oblasti, tedy je funkcí frekvence. Taková reprezentace je pak vhodná k extrakci atributů týkajících se zastoupení jednotlivých frekvencí ve zvukové vlně. Zobrazení signálu v časové a frekvenční oblasti znázorňuje obrázek 3.1. U diskrétních signálů se pro generaci frekvenčního spektra obvykle používá *diskrétní Fourierova transformace (discrete Fourier transform, DFT)*. Rovnice diskrétní Fourierovy transformace má tvar

$$DFT[k] = \sum_{n=0}^{N-1} x[n] e^{\frac{-i2\pi kn}{N}}. \tag{3.1}$$

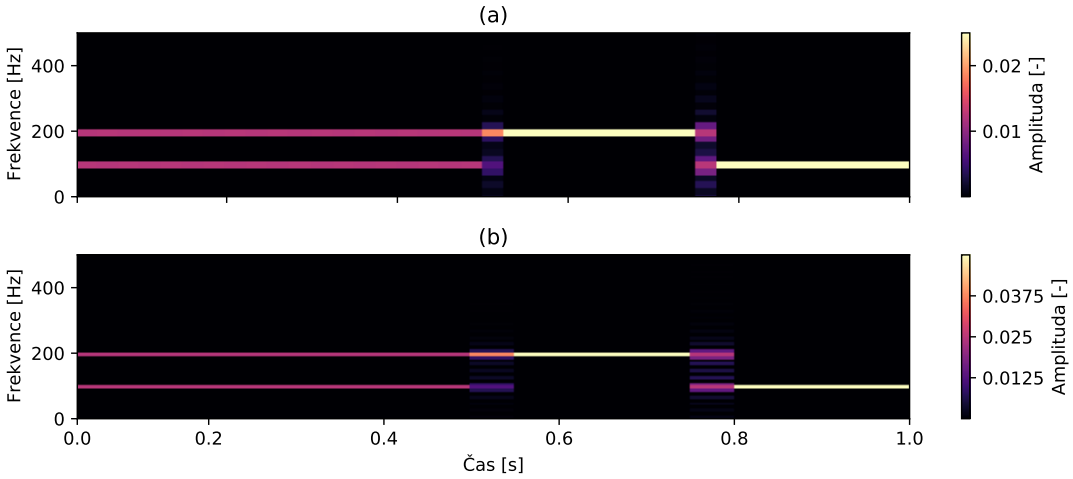
[22][28][13]

Výsledné hodnoty pak vyjadřují amplitudu a fázi vůči frekvenci a jsou obvykle reprezentovány ve dvoudimenzionálním grafu. Pro výpočet DFT se používají efektivní algoritmy nazývané *rychlá Fourierova transformace (fast Fourier transform, FFT)*, které snižují časovou složitost výpočtu z $O(N^2)$ na $O(NlogN)$, kde N značí velikost dat. Protože je ale u zvukových stop důležité sledovat i změnu zastoupení frekvencí v čase, pro spektrální analýzu se nejčastěji používá algoritmus nazývaný *krátkodobá Fourierova transformace (short-time Fourier transform (STFT))*.[22][28][13]

Algoritmus *STFT* nejprve rozdělí původní signál na krátké, obvykle částečně se překrývající, úseky o určitém počtu po sobě jdoucích vzorků nazývané *okna*. Tyto mohou být následně vynásobeny vybranou *okénkovou funkcí* jako je například *Hanningovo okno* či *Hammingovo okno*. Okénkové funkce zpravidla mívají zvonovitý tvar a mimo daný interval nabývají nulových či nízkých hodnot, čímž vyhlazují hrany okna a snižují tak nepřesnost výsledků Fourierovy transformace. Každé z těchto oken je následně zpracováno algoritmem *FFT*. Výsledkem algoritmu *STFT* je reprezentace zvukové stopy v časově-frekvenční oblasti nazývaná *spektrogram*. Ukázky spektrogramů s odlišnými hodnotami délky okna je možné vidět v obrázku 3.2. [22][21][28][13]



Obrázek 3.1: **Ukázka signálu v časové a frekvenční oblasti.** (a) Zobrazení signálu složeného ze dvou sinusoid o frekvenci 100 Hz a 200 Hz v časové oblasti. (b) Zobrazení signálu z obrázku (a) ve frekvenční oblasti. Z obrázku je patrné, že signál obsahuje frekvence 100 Hz a 200 Hz, není však možné určit rozložení těchto frekvencí v čase.



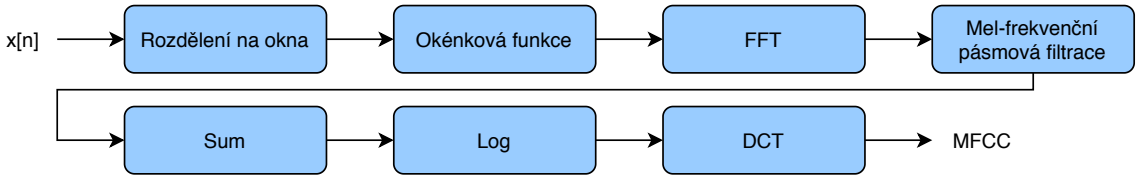
Obrázek 3.2: **Ukázka signálu v časově-frekvenční oblasti.** Spektrogramy signálu z obrázku 3.1(a). Na rozdíl od zobrazení ve frekvenční oblasti je ze zobrazení v časově-frekvenční oblasti možné rozpoznat rozložení frekvencí v čase. Delší okno umožňuje přesněji určit frekvence, zatím co kratší okno umožňuje přesněji určit rozložení frekvencí v čase. (a) Spektrogram se šířkou okna 2000 vzorků a posunem okna 1000 vzorků. (b) Spektrogram se šířkou okna 1000 vzorků a posunem okna 500 vzorků.

Nízkoúrovňové atributy

Nízkoúrovňové atributy lze vypočítat přímo ze zvukové stopy, či odvozeně například z její reprezentace ve frekvenční oblasti. Tyto atributy z pohledu uživatelů nemají velký význam, avšak počítače je jsou schopny snadno využít a dále pomocí nich vypočítat atributy vyšší úrovně. Nízkoúrovňových atributů je mnoho, proto je dále přiblíženo jen několik z nejpoužívanějších.[22]

Mel-frekvenční kepstrální koeficienty

Mel-frekvenční kepstrální koeficienty (Mel-frequency cepstrum coefficients, MFCC) jsou kompaktní formou popisu spektra stopy. Jejich výpočet probíhá filtrováním frekvenčního spektra sadou trojúhelníkových filtrů s šířkou pásma odpovídající *Mel-frekvenční škále*. Tato škála byla vytvořena tak, aby simulovala vnímání frekvencí lidským uchem. Její nárůst je lineární po 1 kHz a dále logaritmický. Pro výpočet koeficientů je pro každý z filtrů vypočítána pásmová energie odmocněním a sečtením amplitud odpovídající části spektra. Logaritmus této energie je nakonec zpracován pomocí *Diskrétní kosinové transformace (Discrete cosine transform, DCT)*, čímž je snížena míra korelace pásem způsobená částečně se překrývajícími filtry. Výpočet *MFCC* se u hudebních souborů typicky provádí nad frekvenčními spektry každého z oken spektrogramu, nikoli nad frekvenčním spektrem celé stopy. Schéma výpočtu Mel-frekvenčních kepstrálních koeficientů je možné vidět v obrázku 3.3.[22][21][13]



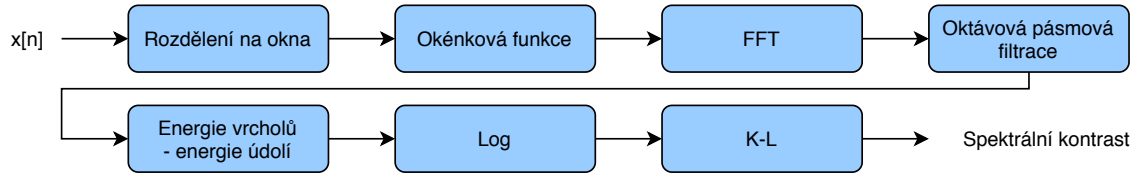
Obrázek 3.3: **Blokové schéma výpočtu Mel-frekvenčních kepstrálních koeficientů.** Obrázek znázorňuje výpočet *MFCC* pro každé z oken spektrogramu. *Sum* značí součet, *Log* značí logaritmus a *DCT* značí Diskrétní kosinovou transformaci.

Spektrální kontrast

Spektrální kontrast (Spectral contrast) vyjadřuje rozdíl energií mezi vrcholy a údolími spektra. Obdobně jako u *MFCC* je při výpočtu spektrálního kontrastu frekvenční spektrum zpracováno pomocí sady filtrů. U spektrálního kontrastu však šířka pásma filtrů odpovídá oktávám chromatické stupnice. Pro každé z pásem je pak vypočítán rozdíl energií mezi vrchním a spodním kvantilem spektra. Pokud jako $\{X_s[1], X_s[2], \ldots, X_s[K]\}$ označíme sestupně seřazené hodnoty amplitud frekvencí filtračního pásma s jednoho okna spektrogramu, pak rovnice pro výpočet spektrálního kontrastu SC_s má tvar

$$SC_s = log(\frac{1}{\alpha K}\sum_{k=1}^{\alpha K} X_s[k]) - log(\frac{1}{\alpha K}\sum_{k=1}^{\alpha K} X_s[K-k+1]), \tag{3.2}$$

kde α značí velikost kvantilů. Pro získání finálních hodnot spektrálního kontrastu jsou hodnoty SC_s zpracovány pomocí *Karhunen-Loeveovy transformace (Karhunen-Loeve transform, K-L)*. Schéma výpočtu Mel-frekvenčních kepstrálních koeficientů je možné vidět v obrázku 3.4.[9][28]



Obrázek 3.4: **Blokové schéma výpočtu spektrálního kontrastu.** Obrázek znázorňuje výpočet spektrálního kontrastu pro každé z oken spektrogramu. *Log* značí logaritmus a *K-L* značí Karhunen-Loeveovu transformaci.

Počet průchodů nulou

Počet průchodů nulou (zero crossing rate (ZCR)) udává, kolikrát amplituda signálu změní znaménko z pozitivního na negativní či naopak. U úzkopásmových signálů jako jsou sinusoidy je hodnota *ZCR* propojena s jejich základní frekvencí. Tento atribut může být využit například při klasifikaci perkusních zvuků. Pokud jako ZCR_r označíme počet průchodů nulou snímku r, rovnice pro výpočet pak má tvar

$$ZCR_r = \frac{1}{2}\sum_{n=1}^{N} |sign(x_r(n)) - sign(x_r(n-1))|, \tag{3.3}$$

kde

$$sign(x) = \begin{cases} 1 & \text{pro} \quad x \geq 0, \\ -1 & \text{pro} \quad x < 0. \end{cases} \tag{3.4}$$

[22][5][21][28][13]

Spektrální tok

Spektrální tok (Spectral flux) je atribut udávající míru korelace spekter po sobě jdoucích oken. Vysoké hodnoty spektrálního toku naznačují náhlou změnu amplitud spektra a tento atribut je tak možné využít k detekci počátků událostí. Rovnice pro výpočet spektrálního toku SF_r pro okno r má tvar

$$SF_r = \sum_{k=1}^{K} (|X_r(k))| - |X_{r-1}(k)|)^2. \tag{3.5}$$

[21][3][13]

Spektrální centroid

Spektrální centroid (Spectral centroid) vyjadřuje geometrický střed distribuce spektra. Umístění centroidu ve vyšší části spektra značí přítomnost velkého počtu vysokých frekvencí. Rovnice pro výpočet centroidu má tvar

$$SC_r = \frac{\sum_{k=1}^{K} X_r[k] \cdot k}{\sum_{k=1}^{K} X_r[k]}. \tag{3.6}$$

[21][3][28][13]

Spektrální klesání

Spektrální klesání (Spectral roll-off) udává frekvenci, pod kterou spadá většina energie spektra. Většinou se uvádí hodnota 85 %. Z matematického hlediska se jedná o nejvyšší frekvenci R_r, pro kterou platí

$$\sum_{k=1}^{R_r} X_r[k] \leq 0.85 \cdot \sum_{k=1}^{K} X_r[k]. \tag{3.7}$$

[21][3][28][13]

Vysokoúrovňové atributy

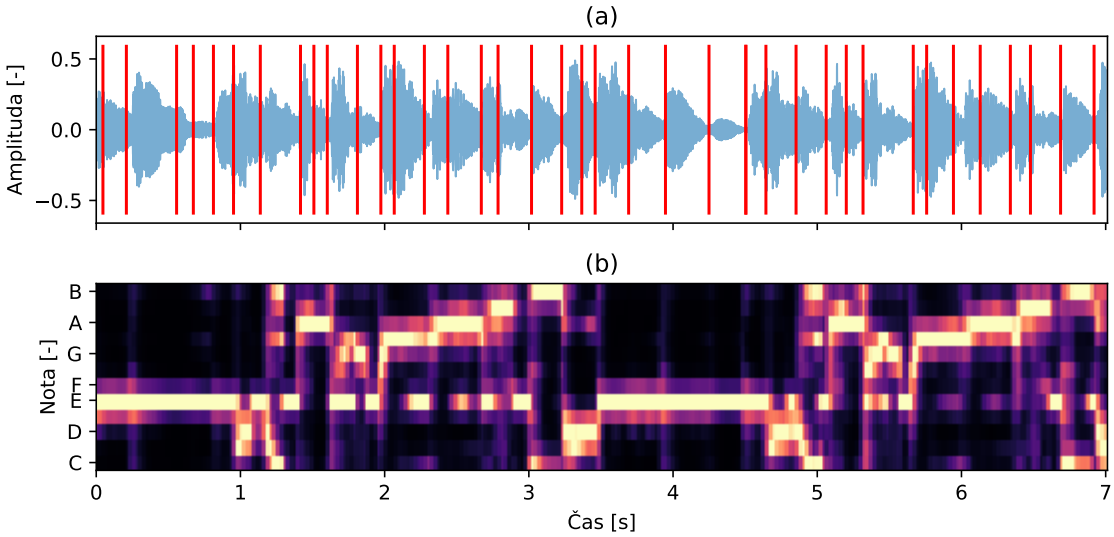
Vysokoúrovňové atributy popisují vlastnosti stopy snadno interpretovatelné lidmi. Jde například o důležité vlastnosti stopy z pohledu hudební teorie, jako jsou melodie, harmonie či rytmus. Může se ale jednat i o více abstraktní atributy jako je například hodnota udávající vhodnost skladby k tanci. Pro extrakci melodie a harmonie je nejprve nutné vybrat vhodnou metodu pro extrakci výšky tónů. Dále také některou z metod pro detekci počátků událostí, která se používá i pro extrakci atributů týkajících se rytmu.[22][13]

Melodie a harmonie a rytmus

Pro extrakci výšky u monofonních signálů se nejčastěji používají metody pro měření periodicity signálu, a to pomocí měření korelace vybraných atributů v navazujících časových úsecích. Je také možné se setkat s aproximací výšky pomocí ideální harmonické řady. U polyfonních signálů se pro aproximaci všech výšek nejčastěji používají iterativní metody aproximace výšky dominantního tónu nalezením nejvýraznější základní frekvence a její harmonické řady. Tyto jsou následně odečteny od původního spektra stopy a proces je opakován, dokud nejsou aproximovány výšky všech tónů. Další způsoby spočívají v nalezení takové sady základních frekvencí a jejich harmonických řad, pomocí které je co nejpřesněji možné aproximovat původní spektrum.[22][21][13]

Pro následnou extrakci atributů týkajících se melodie či rytmu se používají algoritmy pro detekci počátků událostí. Tyto počátky se většinou vyznačují náhlým nárůstem amplitudy a jejich detekce tak probíhá sledováním výrazných změn energie v časové oblasti, případně také nárůstu vysokofrekvenčních komponent. U současných systémů pro detekci počátků je největší limitací obtížná detekce počátků u zvukových stop, které neobsahují zdroj výrazných změn amplitud, jako jsou například perkusní nástroje. V takovém případě pro detekci počátků mohou být využity právě hodnoty výšek tónů, kde jejich změna může značit počátek události.[22][13]

Výše zmíněné postupy je pak možné využít k extrakci atributů vyšší úrovně týkající se melodie či rytmu. U extrakce melodie je stopa rozdělena na segmenty na základě detekovaných počátků. Pro každý segment je pak provedena detekce výšek tónů. Výsledkem je pak řada různých po sobě jdoucích tónu značící melodii. Detekce počátků se používá i pro extrakci atributů týkajících se rytmu. Tempo skladby je aproximováno pomocí měření periodicity těchto počátků, často je však výsledkem násobek tempa reálného, například kvůli detekci půldob. Počátky mohou být využity i pro detekci úderů perkusních nástrojů. Obrázek 3.5 znázorňuje detekci počátků a not.[22][13]



Obrázek 3.5: **Ukázka detekce počátků událostí a not.** (a) Detekované počátky událostí, vyznačující se náhlou změnou amplitudy, jsou znázorněny svislými červenými čarami. (b) Vyšší intenzita detekovaných not je znázorněna světlejší barvou. Po sobě jdoucí noty pak značí detekovanou melodii.

3.2 Selekce atributů

Selekce atributů je dalším ze způsobů dimenzionální redukce a steně jako u extrakce atributů je jejím cílem odstranit redundantní či irelevantní data. Na rozdíl od extrakce atributů, jejíž výstupem jsou transformovaná data pomocí funkcí, výstupem této metody je podmnožina dat původních. Selekci je možné provádět manuálně či algoritmicky, dále bude rozebrána pouze druhá z těchto variant. Metody selekce je možné rozdělit na základě jejich přístupu k vyhodnocení vhodných atributů, a to na metody filtrační, obalovací a vestavěné. Diagram těchto metod je možné vidět v obrázku 3.6.[2][13]

Filtrační metody

Filtrační metody se skládají ze dvou kroků, kde v prvním kroku je vhodnost atributů vyhodnocena pomocí některé z metrik jako vzdálenost, korelace, vzájemná informace nebo konzistence. V druhém kroku je pak vybrán stanovený počet atributů s nejvyšším hodnocením pro předání klasifikačním algoritmům. Atributy mohou být vyhodnoceny samostatně nebo v kontextu množiny atributů, kde v druhém případě jsou pak metody schopny rozpoznat redundantní data. Tyto metody nezávisí na žádném klasifikačním algoritmu a jsou tak obecně schopny lépe popsat vztah mezi atributy.[2][11][13]

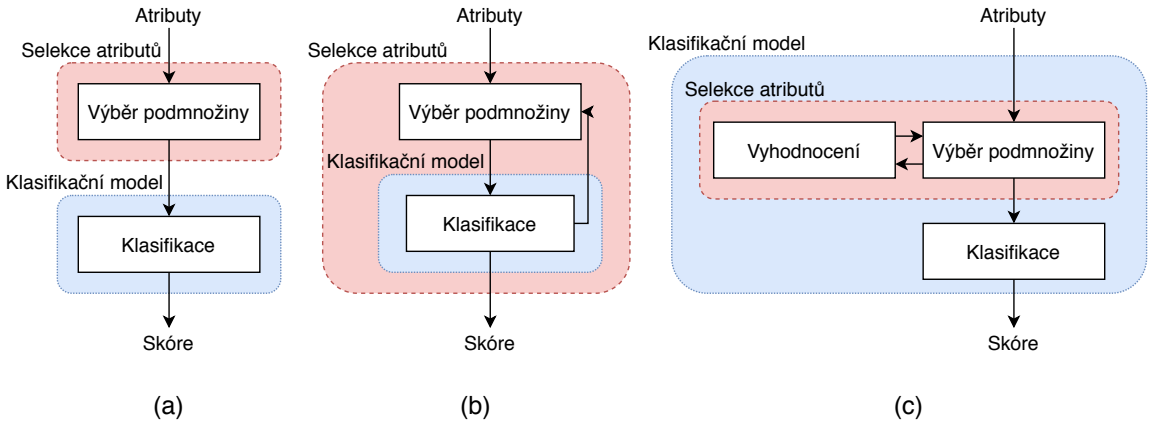
Obalovací metody

Obalovací metody využívají k ohodnocení atributů chybovost predikce určitého modelu. Berou tedy v potaz, že každý klasifikátor může dosáhnout nejlepších výsledků pomocí odlišných atributů. V prvním kroku metody vybírají podmnožiny z množiny atributů za pomoci nějakého z vyhledávacích algoritmů a v druhém kroku je předají modelu k ohodnocení pomocí chybovosti predikce. Tento proces je opakován, dokud není nalezena podmnožina

dosahující nejvyšší úspěšnosti, či úspěšnosti převyšující stanovenou hranici. Vzhledem k nutnosti natrénovat model pro každou z podmnožin atributů jsou tyto metody velmi výpočetně náročné. Lze však pomocí nich pro daný model dosáhnout vyšší úspěšnosti klasifikace, než pomocí metod filtračních.[2][11][13]

Vestavěné metody

Vestavěné metody stejně jako metody obalovací pro vyhodnocení vhodnosti atributů využívají vybraného prediktivního modelu, na rozdíl od metod obalovacích jsou však přímo jeho součástí. Selekce probíhá již při konstrukci modelu a není tak nutné jej opakovaně trénovat. Vestavěné metody lze rozdělit na tři varianty podle postupu selekce, který využívají. První varianta těchto metod natrénuje model na všech atributech a následně nastavením koeficientů či vah náležícím k vybraným atributům testuje, zda-li jejich absence má vliv na úspěšnost predikce modelu. Další variantou je využití algoritmů jako *ID3* či *C4.5* pro výběr atributů, jako je tomu u rozhodovacích stromů. Poslední variantou je pak regularizace pomocí penalizačních funkcí, které na základě vybraného kritéria již v průběhu trénování snižují koeficienty či váhy vybraných atributů.[2]



Obrázek 3.6: **Diagram metod selekce atributů.** (a) Diagram průběhu filtračních metod. Selekce probíhá zcela nezávislá na vybraném klasifikačním algoritmu. (b) Diagram obalovacích metod. Úspěšnost selekce je vyhodnocována pomocí vybraného klasifikačního algoritmu. Není však jeho součástí. (c) Diagram vestavěných metod. Selekce je přímo součástí klasifikačního algoritmu.

3.3 Kódování kategorických dat

Kategorická data často bývají ve formě textových řetězců, a protože většina klasifikátorů textová data zpracovat neumí, může být nutné je nejprve převést do číselné podoby. Nejprve je potřeba určit, jestli kategorická data jsou ordinální či nominální. Nominální kategorická data jsou taková, která mezi sebou nelze porovnat, protože nezastávají žádnou hodnotu. Příkladem nominálních kategorických dat může být například barva. Oproti tomu ordinální data lze porovnat nebo je seřadit. Příkladem ordinálních dat může být například vzdělání, kde vysokoškolské vzdělání má vyšší hodnotu, než vzdělání středoškolské. Nominální kategorická data není vhodné zakódovat tak, aby výsledné hodnoty byly ordinální. V následujících

odstavcích jsou popsány nejpoužívanější způsoby kódování. Ukázky kódování kategorických dat je možné vidět v tabulce 3.2.[27]

Ordinální kódování (ordinal encoding) je vhodné pro ordinální data a funguje na principu přiřazení unikátní číselné hodnoty každé z možných kategorických hodnot. Tato přiřazená číselná hodnota musí odpovídat důležitosti původní kategorické hodnoty. Výhodou tohoto způsobu kódování je, že nezvyšuje velikost dat.[27]

Binární kódování (Binary encoding) nejprve transformuje atributy na jejich číselné reprezentace, následně tyto převede do binární podoby a nakonec vytvoří vektor jedniček a nul pro každý řád hodnot binární reprezentace. Výsledkem je tedy vektor o délce počtu prvků v kódovaných datech a počet těchto vektorů je $log_2(N)$, kde N vyjadřuje počet unikátních kategorických hodnot.[24]

One-hot kódování (One-hot encoding) transformuje každou kategorickou hodnotu na vektor jedniček a nul od délce rovné počtu prvků v kódovaných datech, kde jednička u příslušného prvku značí přítomnost dané kategorické hodnoty a nula naopak. Pokud označíme počet unikátních kategorických hodnot písmenem N, výsledný počet těchto vektorů bude $N - 1$, protože jeden z vektorů je vždy možné odvodit z vektorů ostatních a typicky tak bývá odstraněn. Nevýhody tohoto kódování spočívají ve velkém nárůstu dat a s tím spojenou možností představení fenoménu „řídkých dat“. Tento způsob kódování tak není vhodné použít pro velký počet kategorických dat.[27][24]

Tabulka 3.2: **Ukázka kódování kategorických dat**

Původní hodnota	Ordinální kódování	Binární kódování		One-hot kódování		
a	0	0	0	0	0	0
b	1	0	1	1	0	0
c	2	1	0	0	1	0
d	3	1	1	0	0	1

3.4 Škálování dat

Vstupní data často bývají ve formě, kde hodnoty jednoho atributu jsou řádově jinde, než hodnoty atributu jiného. Některé z klasifikačních algoritmů pak mohou mít problém takovým atributům přiřadit adekvátní váhu. Z tohoto důvodu se v praxi můžeme u předzpracování dat setkat se *škálováním dat (data scaling)*. V následujících odstavcích jsou dále přiblíženy nejpoužívanější z těchto metod. Metody škálování dat jsou znázorněny v obrázku 3.7.[7][11]

Jedna z metod škálování je metoda *min-max normalizace*, kde jsou data lineárně transformována do určitého intervalu, obvykle $\langle 0, 1 \rangle$. Rovnice pro převod hodnoty x_i z množiny X do intervalu $\langle a, b \rangle$ má tvar

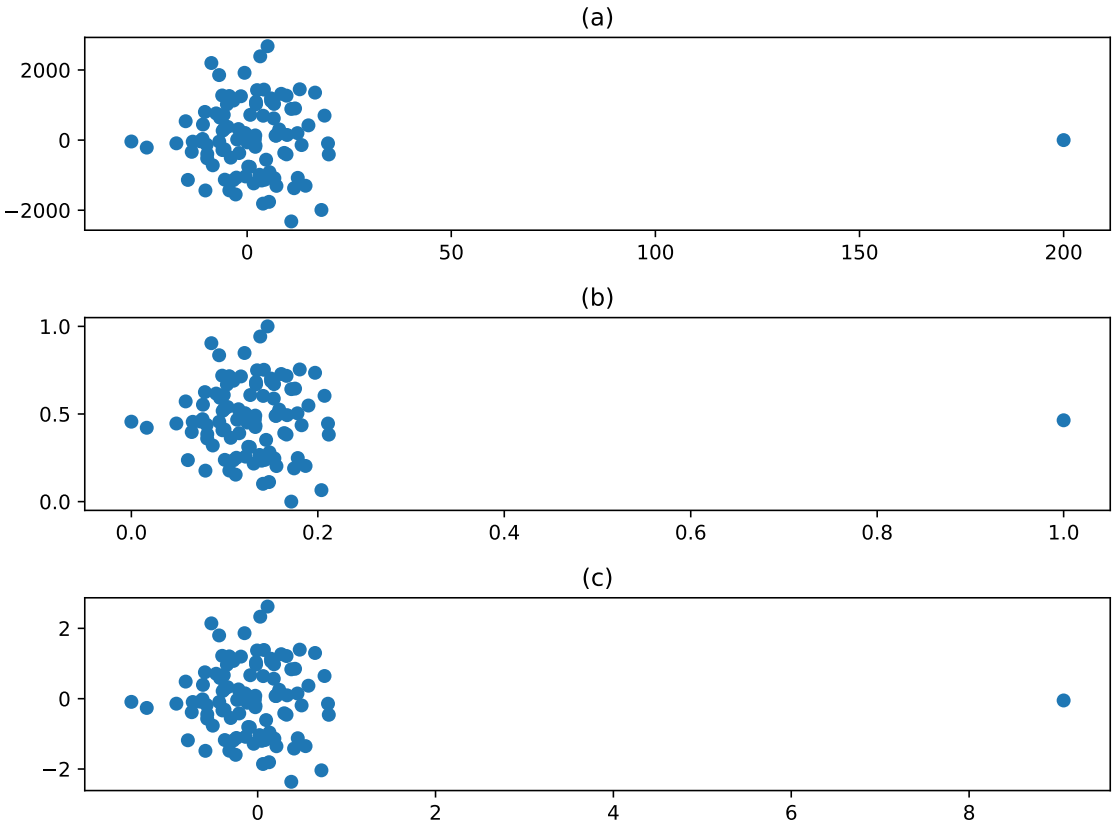
$$x'_i = a + \frac{(x_i - min(X))(b - a)}{max(X) - min(X)}, \tag{3.8}$$

kde funkce $min()$ a $max()$ vracejí nejnižší a nejvyšší hodnotu ze vstupní množiny. Tato metoda je však citlivá na *odlehlé hodnoty (outliers)*.[7][11]

Další metoda se nazývá *Z-score normalizace* jejím cílem je transformovat vstupní hodnoty tak, aby jejich průměr byl roven nule a standardní deviace rovna jedné. Rovnice Z-score normalizace má tvar

$$x'_i = \frac{x_i - mean(X)}{std(X)}, \tag{3.9}$$

kde funkce *mean*() vrací průměrnou hodnotu a funkce *std*() vrací hodnotu standardní deviace vstupního vektoru hodnot.[7][11]



Obrázek 3.7: **Ukázka metod škálování dat.** (a) Znázornění neupravených hodnot atributů v prostoru prvků. Většina hodnot na ose y spadá do rozmezí (−2000–2000) a většina hodnot na ose x spadá do rozmezí (−25–25). Zatím co nárůst hodnoty o 10 bude na ose x podstatný, stejný nárůst bude na ose y poměrně zanedbatelný. (b) Data transformovaná metodou *min-max normalizace*. Zatím co většina hodnot na ose y spadá do rozmezí (0.2–0.8), odlehlá hodnota na ose x způsobila, že většina hodnot na ose x spadá do rozmezí (0–0.2). Nárůst o stejnou hodnotu bude tedy stále podstatně výraznější na ose x. (c) Data transformovaná metodou *Z-score normalizace*. Většina hodnot na obou dvou osách spadá do rozmezí (−2–2), nárůst o stejnou hodnotu je tedy na obou osách srovnatelný.

Kapitola 4

Klasifikační algoritmy

Klasifikace je proces učení s učitelem, tedy algoritmy při učení pracují s daty rozdělenými do tříd, u kterých jsou známé jejich správné hodnoty. Cílem tohoto učení je vytvořit funkci, která bude na základě určitých vlastností dat shopna správně určit třídu, do které tato data patří. Výsledná funkce se pak nazývá klasifikační model (klasifikátor). Cílem klasifikace je vytvořit takový model, který bude schopný rozřadit neznámá data do tříd s co nejvyšší úspěšností. To však vyžaduje, aby byl model schopný generalizovat, tedy brát v potaz to, že u neznámých dat mohou pro klasifikaci být důležité jiné z vlastností, než u dat známých, na kterých byl natrénován. V případě, kdy je model schopný klasifikovat trénovací data s velmi vysokou úspěšností, avšak u dat neznámých dosahuje podstatně nižší úspěšnosti, se nazývá *přetrénování (overfitting)*. V následujících podkapitolách je přiblíženo několik z nejznámějších klasifikátorů. Speciálním případem je pak takzvaný meta-klasifikátor, což je klasifikátor využívající několik klasifikátorů dohromady. Metody využívající meta-klasifikátory se nazývají *ensemble metody* a jsou rozebrány v poslední podkapitole.[2]

4.1 Pravděpodobnostní metody

Pravděpodobnostní metody (Probabilistic methods) využívají principů statistiky pro klasifikaci dat. Cílem těchto metod je pro každou vstupní položku X nalézt takovou třídu C_k z K tříd, pro kterou je pravděpodobnost, že k ní náleží, nejvyšší. V následujících odstavcích jsou krátce popsány dva nejznámější klasifikátory využívající pravděpodobnostních metod pro klasifikaci položek.[2]

Naivní Bayesův klasifikátor (Naive Bayes classifier) je generalizační model, který pro klasifikaci využívá Bayesova teorému. Naivní se nazývá pro to, že předpokládá takzvanou *podmíněnou nezávislost*. Tedy, pokud je již dána třída, předpokládá, že všechny atributy klasifikovaného prvku jsou vzájemně nezávislé. Pokud jako x_n označíme n-tý z N atributů klasifikovaného prvku X, pak rovnice Naivního Bayesova klasifikátoru má tvar

$$NB(X) = \underset{C_k}{argmax}(P(C_k) \prod_{n=0}^{N-1} P(x_n|C_k)). \tag{4.1}$$

[2][18]

Logistická regrese (Logistic regression) je metoda binární klasifikace, jejíž cílem je nalézt vhodnou diskriminační funkci, pomocí které je na základě vektoru atributů klasifikovaného prvku možné určit, zda náleží k vybrané třídě, či nikoliv. Rovnice logistické regrese má tvar

$$P(Y = 1|X) = g(\theta^T X) = \frac{1}{1 - e^{-\theta^T X}}, \tag{4.2}$$

kde

$$g(z) = \frac{1}{1 - e^{-z}} \tag{4.3}$$

se nazývá logistická funkce a platí

$$\theta^T X = \theta_0 + \sum_{n=0}^{N-1} \theta_{n+1} x_n, \tag{4.4}$$

kde $\{\theta_0, \ldots, \theta_N\}$ značí parametry logistické funkce. Pro nalezení adekvátních hodnot těchto parametrů se používá *metoda maximální věrohodnosti*. Pro klasifikaci do více tříd lze využít metodu *jeden proti všem (one-vs-all)*, nebo upravenou verzi logistické regrese nazývanou *Multinomiální logistická regrese*.

[2][25]

4.2 Rozhodovací stromy

Rozhodovací stromy (Decision trees) rekurzivním dělením transformují vstupní data do stromové struktury, kde každý z koncových uzlů je přiřazen k určité třídě. Existuje několik algoritmů pro automatickou generaci rozhodovacích stromů, jako například algoritmy *ID3*, *C4.5* a *CART*. Dělení každého z vnitřních uzlů probíhá na základě *dělícího kritéria (split criterion)*, které pomocí podmínky či predikátu vyhodnotí buďto hodnotu jednoho z atributů, pak se jedná o univariační dělení, nebo hodnoty více atributů, pak se jedná o multivariační dělení. Cílem tohoto dělení je maximalizovat příslušnost prvků v každém poduzlu k určité třídě. Vhodnost atributů k dělení se vyhodnocuje pomocí různých funkcí jako jsou *entropie* či *Gini index*. Entropie a Gini index se počítají podle vzorců

$$Entropie = -\sum_{k=0}^{K-1} p(C_k) log_2 p(C_k) \tag{4.5}$$

a

$$Gini = 1 - \sum_{k=0}^{K-1} p(C_k)^2, \tag{4.6}$$

kde $p(C_k)$ vyjadřuje pravděpodobnost klasifikace do třídy C_k z celkového počtu K tříd. Entropie se používá pro výpočet informačního zisku, který pro vyhodnocení vhodnosti atributu pro dělení uzlu používají například algoritmy *ID3* a *C4.5*. Gini index pak pro toto vyhodnocení používá například algoritmus *CART*.

Dělení uzlů probíhá, dokud není splněno jedno z ukončovacích kritérií. Ukončovacím kritériem může být například náležitost všech prvků v uzlu ke stejné třídě. V praxi se však obvykle používají jiná kritéria, jako náležitost určité části prvků uzlu ke stejné třídě, či pouze určitý počet prvků v uzlu. Tyto navíc pomáhají zamezit přetrénovaní modelu. Není však možné předem určit hodnoty těchto kritérií tak, aby byl růst stromu zastaven ve vhodnou

chvíli. Tento problém řeší metoda *prořezávání stromu (tree pruning)*, která zpětně prochází vytvořený strom a odstraňuje uzly, které by k přetrénování mohly vést. Jedním ze způsobů prostřihávání stromů je například vyčlenění části trénovacích dat a jejich následné použití při vyhodnocování úspěšnosti klasifikace stromů po odstranění určitých uzlů.[2][25][8][18]

4.3 Učení založené na instancích

Metody *učení založené na instancích (instance based learning)* fungují na principu vyhodnocení podobnosti klasifikovaného prvku mezi již známými prvky (instancemi) z trénovacích dat, a to za pomoci vybrané metriky. Tyto metody na základě trénovacích dat nevytváří model, pouze tato data uloží a samotná klasifikace proběhne až ve chvíli, kdy je metodě předán prvek z dat testovacích. Tento přístup se nazývá *líné učení (lazy learning)*. Tyto metody mají řadu výhod, jednak pro klasifikaci využívají pouze část všech dat, což umožňuje snadněji klasifikovat velmi rozsáhlá data, dále každý nový prvek z testovacích dat je možné zařadit mezi již známé prvky dat trénovacích. To může vylepšit úspěšnost klasifikace a také umožní klasifikovat i třídy, které trénovací data neobsahovala. Mezi nevýhody tohoto přístupu se řadí vysoká citlivost na irelevantní data a nevyváženost počtu prvků v jednotlivých třídách.[2][18][8]

Nejznámější z metod založených na instancích je metoda *K-nejbližších sousedů (K-nearest neighbors)*. Tato metoda je založena na předpokladu, že podobnost dvou prvků může být vyjádřena pomocí jejich vzdálenosti v prostoru prvků. Metrikou pro vyhodnocení podobnosti je *Euklidovská vzdálenost*, jejíž rovnice má tvar

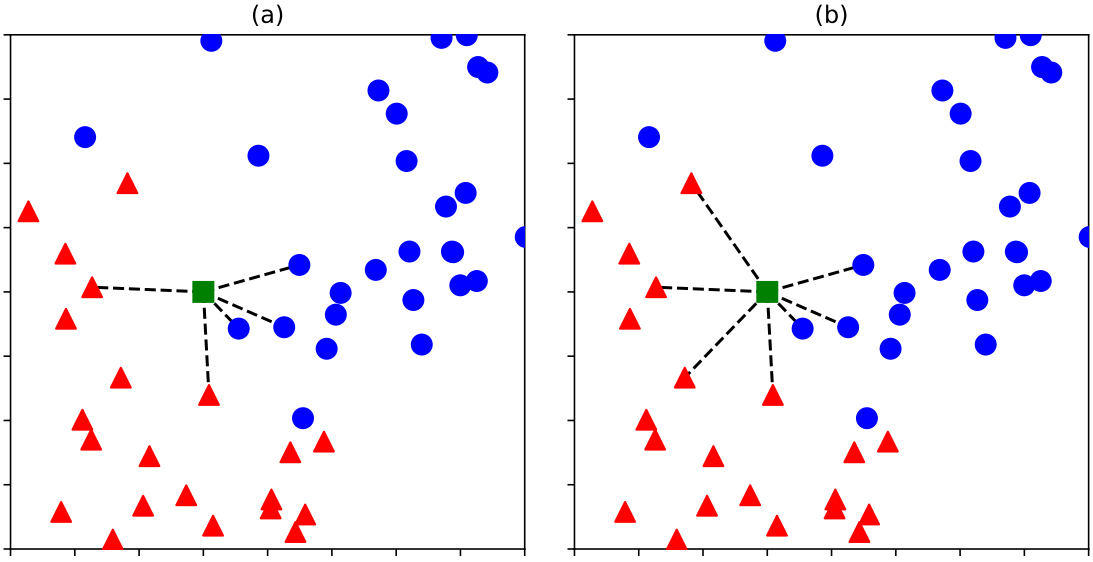
$$d(X,Y) = \sqrt{\sum_{n=0}^{N-1}((x_n) - (y_n))^2}, \tag{4.7}$$

kde $d(X,Y)$ značí výslednou vzdálenost mezi prvkem X s atributy $\{x_0, \ldots, x_{N-1}\}$ a prvkem Y s atributy $\{y_0, \ldots, y_{N-1}\}$. Při klasifikaci nového prvku tato metoda pomocí dané metriky nalezne k nejbližších prvků, a na základě nejčastěji se vyskytující třídy mezi těmito prvky přiřadí třídu prvku novému. Princip fungování metody K-nejbližších sousedů je znázorněn v obrázku 4.1.[8][18]

4.4 Metody podpůrných vektorů

Metody podpůrných vektorů (Support vector machines, SVM) separují dvě třídy pomocí stanovení lineární rozhodovací hranice (nadroviny) tak, aby na každé její straně ležely prvky náležící ke stejné třídě. Jedná se tedy o binární klasifikaci a pro klasifikaci do více tříd je možné využít metodu jeden proti všem. V případě, kdy jsou prvky lineárně separovatelné, se pro stanovení hranice používá metoda nazývaná *hard-margin*. Pomocí této metody je nadrovina umístěna tak, aby byl maximalizován její odstup od prvků každé z tříd. Taková nadrovina se pak nazývá *nadrovina maximálního odstupu (maximum margin hyperplane)* a prvky, na základě kterých byla pozice nadroviny stanovena, se nazývají *podpůrné vektory (support vectors)*.[2][25]

V případě, kdy prvky nejsou lineárně separovatelné, ale přesto je možné umístit lineární rozhodovací hranici tak, aby pomocí ní bylo možné prvky klasifikovat s uspokojivou úspěšností, je možné použít metodu nazývanou *soft-margin*. Vhodnost umístění nadroviny touto metodu je vyhodnocena pomocí funkcí, které kladně hodnotí správně klasifikované prvky



Obrázek 4.1: **Ukázka klasifikace pomocí metody K-nejbližších sousedů.** V obrázcích je znázorněn prostor prvků definovaný dvěma atributy. (a) Ukázka metody s hodnotou $k = 5$. Klasifikovaný prvek reprezentovaný zeleným čtvercem by v tomto případě byl klasifikován jako modrý kruh. (b) Ukázka metody s hodnotou $k = 7$. Klasifikovaný prvek reprezentovaný zeleným čtvercem by v tomto případě byl klasifikován jako červený trojúhelník.

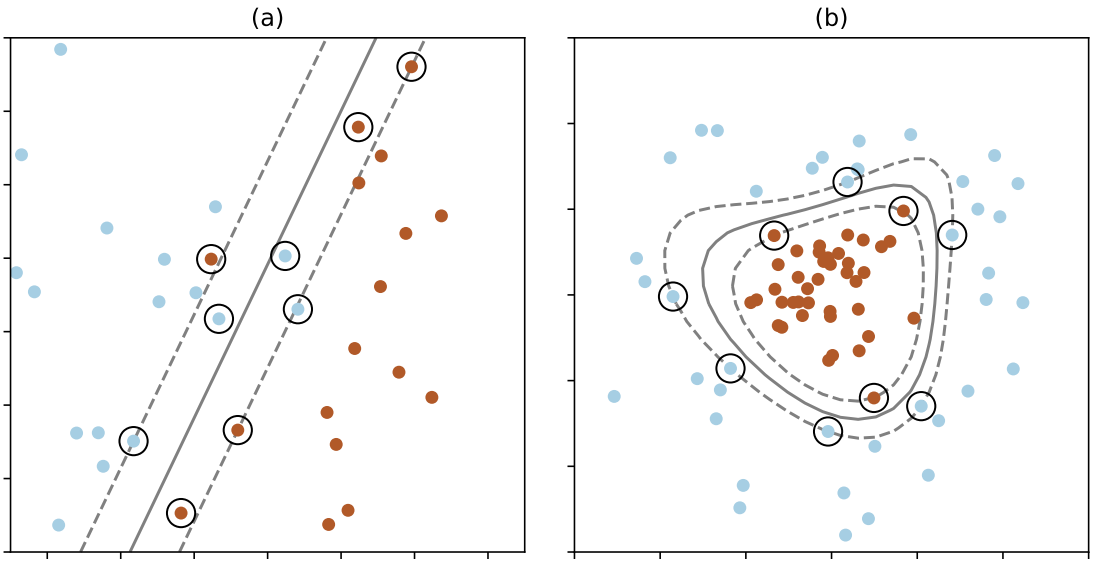
splňující stanovený odstup od hranice, ale záporně hodnotí prvky, které odstup nesplňují, či hranici překročí a jsou tak chybně klasifikovány.

V případě, kdy jsou prvky lineárně neseparovatelné a není možné umístit nadrovinu tak, aby bylo dosaženo dostatečné úspěšnosti klasifikace, je možné využít některé z *jádrových metod (kernel methods)*. Tyto metody transformují prostor prvků na jiný, obvykle vícedimenzionální prostor, který umožňuje umístit nadrovinu tak, aby úspěšnost klasifikace byla uspokojivá. Protože však transformace prvků do nového prostoru není lineární, rozhodovací hranice v původním prostoru bude také nelineární, jak je možné vidět v obrázku 4.2.

4.5 Neuronové sítě

Neuronové sítě (Neural networks) simulují biologické struktury lidského mozku. Architekturu neuronových sítí tvoří neurony a propojení mezi nimi. Neurony se podle funkce, kterou v síti vykonávají, dělí na vstupní, výstupní a výpočetní. Mohou však plnit více funkcí najednou. Výpočetní neurony jsou základní výpočetní jednotky neuronových sítí. Jejich vstupem je několik hodnot a jim přiřazených vah, které transformují na výstupní hodnotu pomocí dvou funkcí. První funkce zpracuje vstupní hodnoty a váhy výsledek předá druhé funkci, která se nazývá *aktivační funkce (activation function)*, která jej transformuje na výstupní hodnotu neuronu. Model neuronu je možné vidět v obrázku 4.3.[2][18]

Neuronová síť se obvykle skládá z několika vrstev, a to vrstvy vstupní, vrstev skrytých a vrstvy výstupní. Trénování takovýchto neuronových sítí pak probíhá ve dvou fázích, kde v první se vstupní hodnoty postupně transformují a propagují dále v síti až do výstupní vrstvy a v druhé se v případě chybné klasifikace upraví váhy a *biasy* příslušných neuronů pomocí metody zpětné propagace. Rychlost trénování je možné ovlivnit pomocí parametru



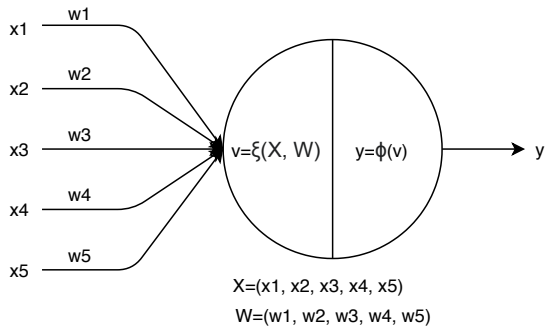
Obrázek 4.2: **Ukázka umístění nadroviny metodou podpůrných vektorů.** (a) Přesto, že data nejsou lineárně separovatelná je možné pomocí metody *soft-margin* umístit nadrovinu tak, aby rozdělila prvky obou tříd s uspokojivou úspěšností. (b) Ukázka případu, kdy ani pomocí metody *soft-margin* není možné umístit nadrovinu tak, aby bylo dosaženo uspokojivé úspěšnosti klasifikace. Umístění nadroviny po transformaci prostoru prvků pomocí jádrové metody *Radial basis function (RBF)* je však možné všechny prvky klasifikovat správně. Protože však tato transformace není lineární, po zpětné transformaci na původní prostor prvků je i stanovená hranice nelineární.

λ. V praxi se nejdříve parametru λ přiřadí vyšší hodnota, což vede k rychlejšímu nalezení hodnot vah blízkých hodnotám optimálním a následně se λ snižuje, kvůli zabránění oscilace okolo optimálních hodnot vah.[2][18]

Pro klasifikaci do více tříd pak lze využít metodu jeden proti všem, nebo navrhnout architekturu sítě tak, aby počet neuronů ve výstupní vrstvě odpovídal počtu klasifikovaných tříd. Výhody neuronových sítí spočívají jejich široké oblasti uplatnění, schopnosti zpracovat rozsáhlá data a jejich schopnosti implicitně provádět dimenzionální redukci. Nevýhody spočívají ve vyšší citlivosti na irelevantní data a možnosti přetrénování.[2][18]

4.6 Ensemble metody

Ensemble metody využívají různé způsoby kombinování několika klasifikátorů dohromady. Cílem těchto metod je dosáhnout lepší schopnosti generalizace a snížení chybovosti klasifikace. Meta-klasifikátory mohou využívat několik stejných klasifikátorů pracujících s různými částmi původních dat, případně různými způsoby do klasifikátorů uvést náhodnost, čímž se zaručí, že tyto nebudou vracet totožné výsledky. Mohou ale také využívat několika různých klasifikátorů a využít tak předností každého z nich. Vyhodnocení výsledků klasifikátorů je možné dosáhnout například pomocí metody většinového hlasování nebo je možné pro tento úkol využít dalšího klasifikátoru. Případně pak mohou využívat všechny z výše zmíněných přístupů dohromady. Ensemble metod existuje mnoho, v následujících odstav-



Obrázek 4.3: **Model neuronu.** Množina X značí vstupní hodnoty neuronu, množina W značí váhy pro každou ze vstupních hodnot.

cích jsou tak přiblíženy pouze tři z nejpoužívanějších přístupů ke kombinaci klasifikátorů, a to *Bagging*, *Boosting* a *Stacking*.

Bagging

Bagging, zkrácenina „Bootstrap aggregating“, je metoda u které je vytvořena nová datová sada pro každý ze základních klasifikátorů obvykle metodou *vzorkování s náhradou (sampling with replacement)*. Tato metoda vytváření nové datové sady spočívá v kopírování náhodně vybraných prvků do datové sady nové, a to až do chvíle, než dosáhne stejné dimenzionality, jako datová sada původní. Nově vytvořená datová sada tak kvůli náhodnému výběru typicky obsahuje některé prvky původní datové sady vícekrát a některé neobsahuje vůbec. Navíc, pokud je původní datová sada dostatečně obsáhlá, je velmi malá šance, že by nově vytvořené datové sady byly totožné. Každý ze základních klasifikátorů je tak natrénován na jiných datech, což způsobí, že jejich predikce na nových datech může být odlišná.

Pro tento způsob kombinování klasifikátorů je vhodné jako základní klasifikátory použít takové, které jsou citlivé na změny trénovacích dat, aby variabilita jejich predikcí byla dostatečně vysoká. V závislosti na vybraných základních klasifikátorech je pak pro zpracování jejich výsledků možné využít například metodu většinového hlasování, či metodu váženého hlasování. Jako základní klasifikátory je možné použít klasifikátory stejné i odlišné, kde použití odlišných klasifikátorů je vhodné obzvláště u malých datových sad, kde variabilita nových datových sad získaných metodou vzorkování s náhradou je nízká.[2]

Bagging metodu využívá například meta-klasifikátor *Náhodný les (Random forest)*, který jako základní klasifikátory používá rozhodovací stromy. Tento klasifikátor však metodu bagging dále obohacuje upravením rozhodovacích stromů tak, že neprovádí výběr vhodného atributu pro dělení z množiny všech atributů, ale pouze z její náhodně vybrané podmnožiny určité velikosti. Právě velikost této podmnožiny ovlivňuje míru náhodnosti stromu. Pokud podmnožina bude zahrnovat pouze jediný atribut, strom bude kompletně náhodný, pokud bude podmnožina zahrnovat všechny atributy, bude strom kompletně deterministický, stejně jako je neupravený rozhodovací strom. Pomocí této míry náhodnosti u základních klasifikátorů v náhodném lesu je možné nadále zvýšit variabilitu jejich predikcí.[2]

Boosting

Boosting je metoda spočívající v propojení několika slabých základních klasifikátorů do jednoho silného meta-klasifikátoru. Slabý klasifikátor je takový, který má nízkou schopnost generalizace a úspěšnost predikce. Silný klasifikátor má naopak vysokou schopnost generalizace a úspěšnost predikce. Na rozdíl od bagging metody, u boosting metody jsou základní klasifikátory přidávány iterativně a pracují s celou množinou trénovacích dat. Cílem iterativního přidávání klasifikátorů je zajistit, že nově přidaný klasifikátor je schopný správně klasifikovat co nejvíce prvků, které předchozí klasifikátor klasifikoval chybně. Existují dva hlavní přístupy k vytváření boosting meta-klasifikátorů, a to *adaptive boosting* a *gradient boosting*.

Adaptive boosting (AdaBoost) je metoda, která v každé iteraci přidá nový slabý klasifikátor a následně zjistí, které z prvků je schopen klasifikovat správně a které chybně. Na základě toho pak přiřadí vyšší váhu prvkům, které klasifikátor přidaný v poslední iteraci klasifikoval chybně a naopak. To zajistí, že klasifikátor v následující iteraci bude mít vyšší pravděpodobnost správně klasifikovat prvky, které byly v poslední iteraci klasifikovány chybně. Nakonec také stejným způsobem upraví váhy přiřazené samotným klasifikátorům na základě jejich úspěšnosti predikce. Tento iterativní proces je opakován, dokud je možné najít klasifikátor s vyšší úspěšností predikce, než je náhodná šance. Nakonec jsou sečteny všechny váhy klasifikátorů, jež prvku přiřadili stejnou třídu a samotnému prvku je přiřazena třída, jejíž součet vah je nejvyšší.

Gradient boosting (GBM) v první iteraci přidá jeden slabý klasifikátor a pro každý prvek vypočítá hodnoty směrodatné odchylky. Pro každý prvek je tedy vypočítána míra chybovosti predikce. V následujících iteracích pak algoritmus *GBM* využije sadu atributů nikoli k predikci třídy, ale k predikci právě této míry chybovosti klasifikace každého z prvků z předchozí iterace. Postupným přidáváním základních klasifikátorů a predikováním chybovosti tak tato metoda v každé iteraci snižuje míru chybovosti predikce. Bez včasného zastavení či regularizace je však na rozdíl od metody *Bagging* náchylná na přetrénování.

Stacking

Stacking je metoda, která pro vyhodnocení predikcí základních klasifikátorů využívá dalšího klasifikátoru. Základní klasifikátory se u této metody obvykle nazývají klasifikátory první úrovně a klasifikátor vyhodnocující jejich výsledky se nazývá klasifikátor druhé úrovně. Zatím co vstupem klasifikátorů první úrovně jsou atributy původní datové sady, vstupem klasifikátoru druhé úrovně jsou predikované třídy klasifikátorů první úrovně. Klasifikátor druhé úrovně se tedy učí, jak zpracovat výsledky základních klasifikátorů. Klasifikátory obou úrovní mohou být i jiné ensemble klasifikátory.[2]

Protože použití metody *Stacking* může často vést k přetrénování modelu, obvykle bývá pro trénování využita metoda křížové validace. Pokud model obsahuje k základních klasifikátorů, je původní datová sada rozdělena do k částí a každému základnímu klasifikátoru je přiřazena jedna z těchto částí jako validační datová sada. Každý z klasifikátorů první úrovně je následně natrénován na zbylých $k - 1$ sadách. Výsledky klasifikace základních klasifikátorů na validační sadě jsou pak použity pro natrénování klasifikátoru druhé úrovně. Tímto způsobem tak nejsou využita stejná trénovací data pro klasifikátory obou úrovní, což zamezí možnosti přetrénování.[2]

Kapitola 5

Návrh a implementace systému

Cílem této práce je vytvořit systém pro klasifikaci hudebních souborů s co nejvyšší úspěšností. Proto byl vytvořen tak, aby bylo možné porovnat úspěšnost klasifikace několika z nejpoužívanějších klasifikátorů a pro následnou optimalizaci pak vybrat nejúspěšnější z nich. Zároveň byly vybrány tři datové sady s různým způsobem kategorizace a dvě knihovny pro extrakci atributů. V první podkapitole jsou blíže popsány vybrané datové sady, ve druhé podkapitole je popsán způsob předzpracování dat a výběr klasifikačních algoritmů a v poslední podkapitole je popsán způsob optimalizace parametrů klasifikačních algoritmů.

Výpočty probíhaly na osobním počítači s osmijádrovým procesorem Intel 9700K s taktovací frekvencí 3.60 GHz a možností přetaktování až na 4.90 GHz. Procesor má k dispozici 16 GB operační paměti DDR4. Pro akceleraci algoritmů byl využit grafický čip Nvidia Geforce GTX 1080Ti s 3584 jádry CUDA a 11GB GDDR5 video paměti. Na stroji byl nainstalován 64 bitový operační systém Linux Ubuntu (verze 20.04).

Pro implementaci byl zvolen jazyk Python (verze 3.8) a vývojové prostředí Jupyter lab, kvůli jeho možnostem snadné vizualizace dat. Implementace byla rozdělena do pěti částí, a to tvorba seznamů skladeb a jim přiřazených žánrů, extrakce atributů, selekce atributů, optimalizace parametrů klasifikačních algoritmů, klasifikace, která zároveň umožňuje vizualizovat výsledky a porovnat úspěšnost klasifikačních algoritmů na jednotlivých sadách atributů a nakonec samotná predikce skladeb z lokálního archivu pomocí natrénovaných klasifikátorů. Struktura projektu je uvedena v příloze F.

5.1 Výběr datových sad

Jelikož většina skladeb spadá pod privátní licence a není je tak možné volně stáhnout, vhodných dostatečně rozsáhlých datových sad pro účely strojového učení není mnoho. Přesto se však členům MIR komunity v průběhu let podařilo několik dostatečně rozsáhlých datových sad vytvořit. Datové sady byly vybrány na základě jejich rozsahu, způsobu kategorizace skladeb a vyváženosti jednotlivých kategorií. Pro klasifikaci byly vybrány tři datové sady dohromady obsahující téměř třinácti tisíc třiceti sekundových úseků skladeb.

Datová sada EBD

Tuto datovou sadu, s plným názvem „Extended Ballroom Dataset“, obsahující přes čtyři tisíce třicetisekundových úseků skladeb, vytvořili Geoffroy Peeters a Ugo Marchand jako nástavbu na datovou sadu „Ballroom“ roku 2016 [15]. Tato datová sada je nevyvážená a skladby dělí do třinácti kategorií podle tanečních stylů. Právě rozřazení podle tanečních

stylů bylo důvodem pro její výběr, protože nabízí možnost porovnat důležité atributy pro klasifikaci žánrů a tanečních stylů. Skladby jsou uloženy ve formátu *MP3* se vzorkovací frekvencí 44100 Hz a bitovou hloubkou 16 bitů. Tato datová sada obsahuje skladby spadající pod následující taneční styly:

- Chacha - 455 skladeb
- Foxtrot - 507 skladeb
- Jive - 350 skladeb
- Pasodoble - 53 skladeb
- Quickstep - 497 skladeb
- Rumba - 470 skladeb
- Salsa - 47 skladeb
- Samba - 468 skladeb
- Slowwaltz - 65 skladeb
- Tango - 464 skladeb
- Viennesewaltz - 252 skladeb
- Waltz - 529 skladeb
- Wcswing - 23 skladeb

Datová sada FMA

Druhá datová sada nese název „FMA: A Dataset For Music Analysis“ a vytvořili ji v roce 2016 Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst a Xavier Bresson [4]. Tuto datovou sadu je možné stáhnout ve čtyřech verzích, z nichž první verze je vyvážená a obsahuje osm žánrů po tisíci třicetisekundových úsecích skladeb. Zbylé verze jsou nevyvážené. Druhá verze obsahuje 25000 třicetisekundových úseků skladeb rozřazených do 16 žánrů. Třetí 106574 třicetisekundových úseků skladeb rozřazených do 161 žánrů a poslední, čtvrtá verze se od třetí liší pouze v tom, že obsahuje sklady v plné délce. Tyto skladby byly staženy z databáze *Free Music Archive*, volně dostupné online databáze skladeb.[1] Skladby jsou nahrané ve studiu, z rádia i přes mikrofon a jsou uloženy ve formátu *MP3* se vzorkovací frekvencí 44100 Hz a bitovou hloubkou 16 bitů. Datová sada byla vybrána jednak kvůli vysokému počtu skladeb a také protože rozřazuje sklady dle méně známých a hůře definovaných žánrů. Pro účely této práce byla vybrána první verze obsahující osm vyvážených žánrů, a to:

- Hiphop
- Pop
- Folk

[1] https://freemusicarchive.org/

- Experimental
- Rock
- International
- Electronic
- Instrumental

Datová sada GTZAN

Již v roce 2002 Tzanetakis a Cook se svou prací zveřejnili datovou sadu s názvem „GTZAN“ obsahující tisíc třiceti sekundových úseků skladeb rovnoměrně rozřazených do desíti žánrů [29]. Stejně jako datová sada FMA obsahuje skladby nahrané za různých podmínek a všechny jsou uloženy se vzorkovací frekvencí 22050 Hz a bitovou hloubkou 16 bitů ve formátu *ULAW*. Tato datová sada byla vybrána, jelikož je často používána v dostupných studiích a slouží tak jako dobrá reference pro porovnání dosažených výsledků vůči ostatním implementacím, a také pro to, že je rozřazena do velmi známých a poměrně úzce definovaných žánrů. Tato datová sada obsahuje žánry:

- Blues
- Classical
- Country
- Disco
- Hiphop
- Jazz
- Metal
- Pop
- Reggae
- Rock

5.2 Výběr klasifikačních algoritmů a jejich parametrů

Z klasifikátorů popsaných v kapitole 4 bylo vybráno pět zástupců klasických klasifikačních algoritmů a dva zástupci ensemble klasifikátorů. Popis všech parametrů těchto klasifikačních algoritmů je možné nalézt v oficiálních dokumentacích knihoven XGBoost a Sci-kit learn.[2][3] Všechny vybrané klasifikátory je možné vidět v následujícím seznamu.

- Logistická regrese (LogisticRegression) - zástupce pravděpodobnostních metod
- K-nejbližších sousedů (KNearestNeighbors) - zástupce metod založených na instancích

[2] https://xgboost.readthedocs.io/en/latest/
[3] https://scikit-learn.org/stable/

- Vícevrstvý perceptron (MLPClassifier) - zástupce neuronových sítí
- Rozhodovací strom (DecisionTreeClassifier) - zástupce rozhodovacích stromů
- Metoda podpůrných vektorů (SVC) - zástupce metod podpůrných vektorů
- Náhodný les (RandomForestClassifier) - zástupce ensemble metody bagging
- Extrémní gradient boosting (XGBClassifier) - zástupce ensemble metody boosting

U každého klasifikátoru byly nastaveny vhodné parametry pro následnou selekci atributů, optimalizaci parametrů i klasifikaci následovně. Pro každý z algoritmů, který tyto možnosti podporoval, byl nejprve nastaven parametr *class_weight* na hodnotu *balanced*, což by mělo zaručit, že i prvky, jejichž třídy jsou v datové sadě málo početné, ovlivní trénování algoritmu na tolik, že je bude schopen klasifikovat s obdobnou úspěšností, jako prvky více početné. Dále byl nastaven parametr *n_jobs* na hodnotu *-1* tak, aby klasifikátory využívali všechna logická jádra procesoru a výpočty tak probíhaly rychleji. U klasifikátoru *XGBClassifier* pak byl nastaven parametr *tree_method* na hodnotu *gpu_hist* a parametr *n_jobs* na hodnotu *1* tak, aby výpočty byly prováděny na grafickém čipu namísto procesoru. S dostatečně výkonným grafickým čipem by tak výpočty měly probíhat podstatně rychleji. Parametr *random_state* byl nastaven na náhodně vybranou hodnotu *42*, což zaručí, že výsledky je možné zpětně reprodukovat a jako poslední pak byl nastaven parametr *max_iter* na hodnotu *10000*, aby algoritmy měly vždy, kdy to bylo možné, možnost zcela konvergovat k optimálnímu řešení.

Dále byly u klasifikátorů, které podporovali použití různých algoritmů pro nalezení optimálního řešení (Logistická regrese, Vícevrstvý perceptron), jádrových metod (Metody podpůrných vektorů) či struktur pro efektivní vyhledávání dat (K-nejbližších sousedů) otestovány všechny z těchto možností na každé z šesti sad atributů. Účelem tohoto testování bylo zjistit, která ze zmíněných možností dosahuje nejvyšší úspěšnosti klasifikace a zároveň je přijatelně výpočetně náročná pro následnou selekci atributů. Na základě těchto výsledků, dostupných v příloze B, byly vybrány následující parametry.

Pro Logistickou regresi byl ponechán výchozí algoritmus *LBFGS* a stejně tak pro vícevrstvý perceptron výchozí algoritmus *ADAM*. Pro metodu podpůrných vektorů byla vybrána původní lineární metoda i metoda s použitím jádrové funkce *RBF*, jelikož obě tyto varianty dosahovaly nejlepších výsledků a každá z nich může výrazně ovlivnit, jaká sada atributů bude vybrána jako ideální. A nakonec u algoritmu K-nejbližších sousedů byla vybrána metoda *brute*, protože žádná z datových sad není dostatečně obsáhlá na to, aby vytvoření stromových struktur pro ukládání a vyhledávání prvků bylo efektivní. Všechny zvolené parametry, které byly vyhodnoceny jako nejlepší, je možné vidět v následujícím seznamu.

- LogisticRegression (LR) - max_iter : 10000, $class\_weight$: $balanced$
- KNearestNeighbors (KNNC) - n_jobs : -1, $algorithm$: $brute$
- MLPClassifier (MLPC) - $random\_state$: 42, max_iter : 10000
- DecisionTreeClassifier (DTC) - $class\_weight$: $balanced$, $random\_state$: 42
- SVC_linear (SVCL) - $kernel$: $linear$, $class\_weight$: $balanced$
- SVC_rbf (SVCR) - $kernel$: rbf, $class\_weight$: $balanced$

- RandomForestClassifier (RFC) - $n\_jobs : -1, class\_weight : balanced, random\_state : 42$
- XGBClassifier (XGBC) - $tree\_method : gpu\_hist, n\_jobs : 1, random\_state : 42$

5.3 Předzpracování dat

Prvním krokem předzpracování dat bylo vytvoření seznamů pojících jednotlivé skladby s jejich žánrem. Vytvoření takových seznamů bylo nutné, protože u každé datové sady byly informace o žánrech zaznamenány jiným způsobem, a to ne vždy vhodným ke zpracování klasifikačními algoritmy. U datové sady FMA byly potřebné informace extrahovány z přiloženého souboru .csv, u datové sady GTZAN byly informace o žánrech extrahovány z názvu jednotlivých souborů a u datové sady EBD pak z názvů složek, ve kterých byly skladby uloženy. Pro každou z datových sad byl tedy vytvořen jeden soubor .csv pomocí knihovny Pandas, tvořený dvěma sloupci, kde první obsahuje identifikační číslo skladby a druhý informaci o žánru dané skladby.

Druhým krokem předzpracování dat byla extrakce atributů blíže rozepsaná v podkapitole 3.1 pomocí funkcí knihoven Librosa a Essentia a jejich následné zpracování pomocí statistických funkcí z knihoven Numpy a Scipy. Třetím krokem byla selekce atributů přiblížená v podkapitole 3.2, které bylo dosaženo pomocí dvou obalovacích metod, a to dopředné selekce a zpětné eliminace. Pro správné zpracování atributů klasifikačními algoritmy je však nejprve nutné provést kódování kategorických dat, čehož bylo pro potřeby selekce atributů, optimalizace parametrů i samotné klasifikace dosaženo pomocí algoritmu *One-hot kódování* popsaného v podkapitole 3.3. Tento způsob kódování byl vybrán kvůli jeho vhodnosti pro nominální kategorická data a dostupné implementaci v knihovně Sci-kit learn.

Následně byla data rozdělena na trénovací sadu obsahující 80 % skladeb a testovací sadu obsahující zbylých 20 % skladeb, a to tak, aby byl zachován poměr kategorií v obou sadách. Pro potřeby validace byla implementována křížová validace a klasická validace na validační sadě, vytvořené rozdělením původní trénovací sady opět v poměru 80/20 %. Tato rozdělená data pak byla transformována pomocí metody *Z-score normalizace* tak, že pro normalizaci byly využity hodnoty pouze sady trénovací. Testovací a validační sada pak byla transformována bez ohledu na její hodnoty stejným způsobem, jako sada trénovací, což zaručí, že se nedostanou informace o hodnotách z těchto sad do sady trénovací.

Extrakce atributů

Před samotným zpracováním zvukových stop byla provedena kontrola kvality dat, kde bylo ověřeno, že soubory nejsou poškozené a jejich délka je 30 sekund. U datové sady FMA bylo tímto způsobem nalezeno 7 skladeb, z nich 3 nedosahovaly požadované délky, 3 nebylo možné otevřít a jedna neobsahovala žádné zvuky. Všechny tyto skladby byly opětovně staženy z volně dostupných zdrojů a nahrazeny.

Pro extrakci atributů byly využity dvě knihovny, kde první z nich byla knihovna Librosa (verze 0.8.0) implementovaná v jazyce Python. Tato knihovna obsahuje řadu funkcí pro analýzu zvuků a hudby nezbytných pro vytvoření systémů pro získávání hudebních informací.[4] Druhou knihovnou byla Essentia (verze 2.1b6.dev234) implementovaná v jazyce C++. Tato knihovna obsahuje rozsáhlou sadu algoritmů pro analýzu signálů a extrakci popisných vlastností ze zvukové stopy. Je také multiplatformní a obsahuje rozhraní pro použití

[4] https://librosa.github.io/librosa/index.html

v jazyce Python. Díky těmto vlastnostem, a také zaměření na co nejefektivnější zpracování, tuto knihovnu pro zpracování audio stop využívá několik firem a hudebních databází, jako například AcousticBrainz.[5]

Samotné skladby každé z datových sad byly následně zpracovány ve dvou fázích, nejprve pomocí funkcí z knihovny Librosa, následně pomocí knihovny Essentia. V obou případech zpracování probíhalo paralelně tak, že každému logickému jádru procesoru byl přiřazen jeden proces zpracovávající jednu skladbu. Každá skladba byla načtena se vzorkovací frekvencí 44100 Hz, což je původní vzorkovací frekvence skladeb datových sad FMA a Extended Ballroom. Skladby datové sady GTZAN, se vzorkovací frekvencí 22050 Hz, byly převzorkovány, aby nebylo nutné upravovat parametry extrakčních funkcí.

Pro uchování atributů byl využit datový rámec knihovny Pandas, kde každý řádek odpovídá jedné skladbě a je indexován jejím identifikačním číslem, každý sloupec pak odpovídá hodnotě určitého atributu. Protože však některé atributy obsahují více hodnot, sloupce jsou indexovány pomocí více úrovní, kde první úroveň nese název extrahovaného atributu, druhá úroveň odpovídá použité statistické funkci a poslední úroveň slouží k očíslování jednotlivých hodnot atributů. Po zpracování všech dat byly z datového rámce odstraněny skladby obsahující chybějící či chybné hodnoty, které by nebylo možné zpracovat klasifikačními algoritmy. Nakonec pak byl datový rámec uložen ve formátu .csv, a to pro každou ze tří datových sad a každou ze dvou knihoven, což dohromady dělá šest souborů obsahující sady atributů.

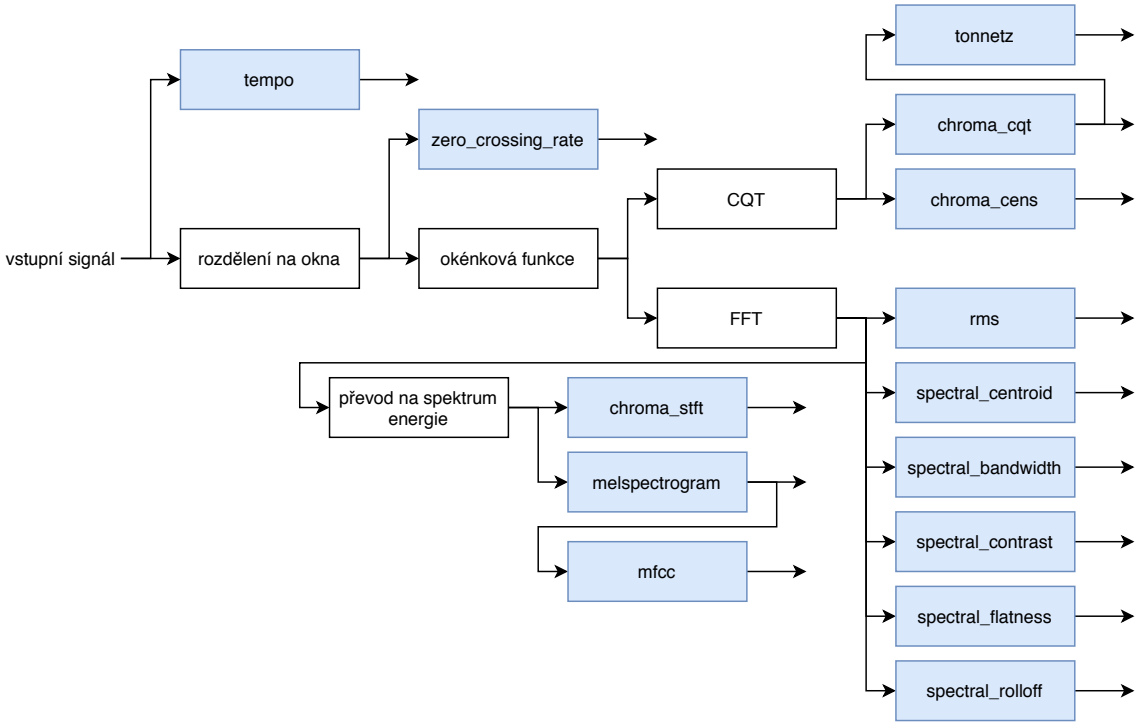
Pomocí funkcí knihovny Librosa bylo extrahováno celkem čtrnáct atributů uvedených v obrázku s diagramem průběhu extrakce 5.1. Popis těchto atributů je uveden v oficiální dokumentaci knihovny Librosa.[6] Atributy byly extrahovány z reprezentace zvukové stopy v časové oblasti a časově-frekvenční oblasti. Pro část spektrálních atributů byl převod do časově-frekvenční oblasti proveden pomocí metody *Constant-Q transform (CQT)* a pro část pomocí metody *STFT*. V obou případech byla zvolena délka okna 2048 vzorků, což při vzorkovací frekvenci 44100 Hz odpovídá úseku dlouhému přibližně 46 ms. Tato délka by měla být adekvátní pro dostatečné časové i frekvenční rozlišení časově-frekvenční analýzy. Pro rozestup jednotlivých oken byla zvolena hodnota 1024 vzorků. Každé okno se tedy z poloviny překrývá s oknem předchozím. což by mělo zaručit dostatečnou přesnost frekvenční analýzy. Nakonec byla v obou případech okna zpracována okénkovou funkcí typu Hanning.

Protože většina atributů je tvořena jednou či více hodnotami pro každé z oken, kterých je s výše zvolenými parametry pro třicetisekundovou stopu bezmála 1300, byla i po extrakci dimenzionalita dat příliš vysoká pro zpracování na osobním počítači. Hodnoty atributů napříč všemi okny tak byly zpracovány statistickými funkcemi z knihoven Numpy a Scipy. Celkem bylo vybráno 7 statistických funkcí, a to průměr, medián, minimální hodnota, maximální hodnota, standardní deviace, koeficient šikmosti distribuce a koeficient špičatosti distribuce. Po zpracování statistickými funkcemi tvoří atributy extrahované pomocí knihovny Librosa celkem 1422 hodnot pro každou skladbu. Extrakce atributů pomocí knihovny Librosa trvala celkem 1 hodinu a 16 minut.

Knihovna Essentia umožňuje extrahovat všechny dostupné atributy pomocí jediné funkce. Protože však tato funkce nepodporuje načítání audio stop ve formátu *ULAW*, u datové sady GTZAN bylo nejprve nutné pomocí knihovny Soundfile skladby načíst a následně je dočasně uložit ve formátu *WAVE*. Po zpracování byly konvertované skladby odstraněny. Funkce *MusicExtractor* extrahuje ze skladby 49 nízkoúrovňových atributů, 14 atributů týkajících se rytmu a 17 atributů týkajících se melodie a harmonie. Podrobný popis této funkce, včetně

[5] https://essentia.upf.edu/

[6] https://librosa.github.io/librosa/index.html



Obrázek 5.1: **Diagram průběhu extrakce atributů pomocí funkcí knihovny Librosa.** Z načtené zvukové stopy je nejprve extrahován atribut *tempo*, poté je stopa rozdělena na okna a je extrahován atribut *zero crossing rate*. Dále je stopa vynásobena okénkovou funkcí a převedena do časově frekvenční oblasti pomocí metody *Constant-Q transform* nebo *Short-time Fourier transform*. Pro některé z atributů je toto frekvenční spektrum nakonec převedeno na spektrum energie. Následně je provedena extrakce ostatních atributů.

všech jejích parametrů a extrahovaných atributů je možné nalézt v oficiální dokumentaci knihovny Essentia.[7]

Funkce *MusicExtractor* vrací extrahované atributy ve dvou formách, a to v neupravené podobě a po zpracování statistickými funkcemi. Pro následné zpracování byla opět kvůli příliš vysoké dimenzionalitě neupravených dat zvolena varianta druhá. Výchozí hodnoty parametrů extrakční funkce byly vyhodnoceny jako adekvátní a byly tak z větší části ponechány beze změn. Rozšířen byl pouze počet použitých statistických funkcí za účelem zachování co největšího množství informací. Všechny zvolené hodnoty parametrů pro funkci *MusicExtractor*, včetně jejich krátkého popisu, je možné vidět v příloze A. Atributy extrahované pomocí knihovny Essentia tvoří celkem 4403 hodnot pro každou skladbu. Extrakce atributů pomocí knihovny Essentia trvala celkem 41 minut.

Selekce atributů

Pro selekci atributů byly zvoleny dvě obalovací metody, a to *dopředná selekce (forward selection, FS)* a *zpětná eliminace (backward elimination, BE)*. Pro vyhodnocení úspěšnosti klasifikace byla z důvodu příliš vysoké výpočetní náročnosti využita pouze validační sada. V následujících podkapitolách jsou blíže popsány tyto metody a dosažené výsledky. Kvůli příliš vysokému počtu sad atributů a klasifikačních algoritmů nejsou vybrané atributy

[7] https://essentia.upf.edu/streaming_extractor_music.html

v této práci zahrnuty. Lze je však dohledat na přiloženém médiu. Výpis průběhu selekce byl uložen do souboru *feature_selection.log* a samotné atributy pak do souboru *optimised_feature_sets.json*.

Dopředná selekce

Počet atributů vybraných pomocí metody dopředné selekce je možné vidět v tabulce 5.1. Ze 14 atributů extrahovaných pomocí knihovny Librosa byly vybrány průměrně 4 atributy a z 80 atributů extrahovaných pomocí knihovny Essentia pak 6 atributů. Pseudokód dopředné selekce je uveden v algoritmu 1. Selekce pomocí této metody trvala na všech sadách atributů celkem 14 hodin a 1 minutu. Informace o délce běhu pro jednotlivé klasifikátory na každé ze sad atributů jsou uvedeny v příloze C.

Algorithm 1 Pseudokód dopředné selekce

```
Zkopíruj všechny atributy do remaining_features[]
Nastav max_score na 0
Nastav total_max_score na 0
while i = 0 < len(features[]) do
    for feature in remaining_features[] do
        Přidej feature do best_features[]
        Proveď klasifikaci na množine best_features[]
        if score > max_score then
            Nastav max_score na score
            Nastav best_feature na feature
        end if
        Odstraň feature z best_features[]
    end for
    if max_score <= total_max_score then
        Ukonči selekci
    end if
    Nastav total_max_score na max_score
    Přidej best_feature do best_features[]
    Odstraň best_feature z remaining_features[]
end while
```

Tabulka 5.1: **Počet atributů vybraných pomocí metody dopředné selekce**

	EBD		FMA		GTZAN		Average
	librosa	essentia	librosa	essentia	librosa	essentia	
LogisticRegression	6	9	7	11	5	6	$7.\overline{3}$
KneighborsClassifier	2	8	3	8	3	5	$4.8\overline{3}$
MLPClassifier	3	6	6	2	3	3	$3.8\overline{3}$
DecisionTreeClassifier	3	8	5	6	2	6	5
SVC_linear	4	4	7	6	2	4	4.5
SVC_rbf	7	6	4	15	4	6	7
RandomForestClassifier	3	4	3	6	1	4	3.5
XGBClassifier	5	5	4	4	4	2	4
Average	4.125	6.25	4.875	7.25	3	4.5	5

Zpětná eliminace

Počet atributů vybraných pomocí metody zěptné eliminace je možné vidět v tabulce 5.2. Ze 14 atributů extrahovaných pomocí knihovny Librosa bylo vybráno průměrně 10 atributů a z 80 atributů extrahovaných pomocí knihovny Essentia pak 67 atributů. Pseudokód zpětné eliminace je uveden v algoritmu 2. Selekce pomocí této metody trvala na všech sadách atributů celkem 3 dny, 22 hodin a 53 minut. Informace o délce běhu pro jednotlivé klasifikátory na každé ze sad atributů jsou uvedeny v příloze C.

Algorithm 2 Pseudokód zpětné eliminace

```
Zkopíruj všechny atributy do best_features[]
Proveď klasifikaci na množine best_features[]
Nastav max_score na 0
Nastav total_max_score na score
while i = 0 < len(features[]) do
    for feature in best_features[] do
        Odstraň feature z best_features[]
        Proveď klasifikaci na množine best_features[]
        if score >= max_score then
            Nastav max_score na score
            Nastav best_feature na feature
        end if
        Přidej feature do best_features[]
    end for
    if max_score < total_max_score then
        Ukonči selekci
    end if
    Nastav total_max_score na max_score
    Nastav max_score na 0
    Odstraň best_feature z best_features[]
end while
```

Tabulka 5.2: **Počet atributů vybraných pomocí metody zpětné eliminace**

	EBD		FMA		GTZAN		Average
	librosa	essentia	librosa	essentia	librosa	essentia	
LogisticRegression	12	48	7	73	12	37	31.5
KneighborsClassifier	7	66	12	73	8	72	$39.\overline{6}$
MLPClassifier	10	80	13	79	11	80	45.5
DecisionTreeClassifier	13	75	12	80	10	79	$44.8\overline{3}$
SVC_linear	11	58	12	75	9	26	$31.8\overline{3}$
SVC_rbf	7	47	8	67	6	51	31
RandomForestClassifier	12	76	13	79	12	75	44.5
XGBClassifier	9	76	13	78	13	62	$41.8\overline{3}$
Average	10.125	65.75	11.25	75.5	10.125	60.25	$38.8\overline{3}$

Zhodnocení výsledků selekece atributů

Vliv selekce atributů na skóre jednotlivých klasifikátorů na validačních sadách je možné vidět v tabulce 5.3 pro atributy extrahované pomocí knihovny Librosa a v tabulce 5.4 pro atributy extrahované pomocí knihovny Essentia. Vliv na dobu trénování a klasifikace pak v příloze C. Z výsledků je patrné, že skóre na podmnožině atributů získané pomocí metody dopředné selekce bylo v drtivé většině vyšší, než na celé množině atributů. Na atributech extrahovaných pomocí knihovny Librosa bylo dosaženo zvýšení skóre ve všech případech, na atributech extrahovaných pomocí knihovny Essentia pak už jen v devatenácti ze čtyřiadvaceti případů. Pomocí metody zpětné eliminace pak bylo dosaženo zvýšení skóre ve všech případech. Jelikož však ve spoustě případů pomocí metody dopředné selekce bylo dosaženo vyššího nárůstu skóre, než pomocí metody zpětné eliminace, nelze prohlásit, že by metoda zpětné eliminace byla obecně lepší. Z toho důvodu byly pro následnou optimalizaci parametrů využity podmnožiny atributů získané pomocí obou metod.

Dále je také nutné podotknout, že testování úspěšnosti klasifikace bylo prováděno na stejné validační sadě, na jaké byla testována úspěšnost při samotné selekci atributů. Dá se tedy předpokládat, že výsledky klasifikace na sadě testovací budou odlišné. V neposlední řadě je z výsledků klasifikace patrné, že na sadách atributů extrahovaných pomocí knihovny Essentia bylo obecně dosaženo vyššího skóre, než na sadách atributů extrahovaných pomocí knihovny Librosa. Jediným případem, kdy nejvyššího skóre bylo dosaženo pomocí atributů knihovny Librosa, je klasifikátor *MLPClassifier* na datové sadě GTZAN. Z toho důvodu byly atributy extrahované pomocí knihovny Librosa vyhodnoceny jako nedostatečné, a pro následnou optimalizaci parametrů a klasifikaci byly využity pouze atributy extrahované pomocí knihovny Essentia.

Tabulka 5.3: **Rozdíl skóre po selekci atributů extrahovaných pomocí knihovny Librosa na validační sadě**

		LR	KNNC	MLPC	DTC	SCVL	SVCR	RFC	XGBC	Average
EBD	FS	3.44%	14.50%	6.88%	1.94%	0.45%	8.52%	10.16%	0.45%	5.79%
	BE	1.35%	8.22%	5.38%	2.24%	2.84%	8.52%	6.43%	2.24%	4.65%
FMA	FS	8.98%	4.06%	0.47%	2.11%	4.45%	2.34%	0.94%	1.25%	3.08%
	BE	8.98%	3.20%	1.41%	1.80%	0.86%	3.44%	0.70%	2.27%	2.83%
GTZAN	FS	4.38%	6.25%	5.00%	5.00%	3.75%	16.88%	12.50%	3.75%	7.19%
	BE	7.50%	0.62%	7.50%	7.50%	7.50%	16.25%	11.25%	1.88%	7.50%
Average	FS	5.60%	8.27%	4.12%	3.02%	2.88%	9.25%	7.87%	1.82%	5.35%
	BE	5.94%	4.01%	4.76%	3.85%	3.73%	9.40%	6.13%	2.13%	5.00%

Tabulka 5.4: **Rozdíl skóre po selekci atributů extrahovaných pomocí knihovny Essentia na validační sadě**

		LR	KNNC	MLPC	DTC	SCVL	SVCR	RFC	XGBC	Average
EBD	FS	0.75%	30.64%	8.07%	10.31%	-2.69%	10.61%	3.59%	0.90%	7.77%
	BE	3.74%	14.20%	4.19%	6.58%	1.49%	13.75%	2.39%	2.09%	6.05%
FMA	FS	10.70%	6.17%	-4.61%	2.42%	6.33%	5.47%	0.62%	0.94%	3.51%
	BE	3.52%	1.33%	2.03%	4.22%	2.34%	3.05%	1.17%	3.28%	2.62%
GTZAN	FS	0.00%	13.75%	5.63%	6.88%	-1.25%	7.50%	3.13%	-3.12%	4.07%
	BE	3.75%	15.00%	5.00%	6.88%	4.38%	6.88%	5.00%	3.75%	6.33%
Average	FS	3.82%	16.85%	3.03%	6.54%	0.80%	7.86%	2.45%	-0.43%	5.11%
	BE	3.67%	10.18%	3.74%	5.89%	2.74%	7.89%	2.85%	3.04%	5.00%

5.4 Optimalizace parametrů klasifikačních algoritmů

Pro optimalizaci parametrů klasifikátorů byla využita knihovna Optuna (verze 1.5.0). Tato knihovna je implementována v jazyce Python a obsahuje několik algoritmů pro vyhledání optimálních hodnot parametrů. Za účelem optimalizace byla vytvořena třída *HPoptimise* obsahující optimalizační funkce pro každý z klasifikačních algoritmů využitých v této práci. parametry každého z osmi použitých klasifikátorů byly následně optimalizovány pomocí jednoho ze dvou vybraných optimalizačních algoritmů.

Prvním z nich je algoritmus *GridSampler*, který metodou *vyčerpávajícího hledání (exhaustive search)* prohledává předem definovaný prostor parametrů. Tato metoda tedy vyzkouší každou z možných kombinací parametrů z definovaného prostoru, a je tak vždy schopna nalézt optimální řešení. Její nevýhodou je však vysoká výpočetní náročnost, a je tak vhodné ji využít pouze v případě, kdy prostor parametrů není příliš obsáhlý, či samotný klasifikační algoritmus je velmi málo výpočetně náročný. Tento algoritmus byl využit pro optimalizaci parametrů klasifikátorů *LogisticRegression*, *KNeighborsClassifier*, *SVC_linear* a *SVC_rbf*.

Druhým algoritmem je *Tree-structured Parzen Estimator (TPE)*, který nejprve provádí náhodný výběr parametrů z definovaného prostoru a následně na základě výsledků klasifikace vybírá takové kombinace parametrů, na nichž bylo v předešlých iteracích dosaženo nejlepších výsledků. Tímto způsobem je tak algoritmus schopen konvergovat k optimálnímu řešení a je tak vhodný i pro rozsáhlejší prostory parametrů, čí výpočetně náročnější klasifikační algoritmy. Jeho nevýhodou však je, že v případě, kdy v prvních iteracích s náhodným

výběrem nejsou zvoleny vhodné kombinace parametrů, může konvergovat k řešení, které je pouze lokálně optimální. tento algoritmus byl využit pro optimalizaci parametrů klasifikátorů *MLPClassifier*, *DecisionTreeClassifier*, *RandomForestClassifier* a *XGBClassifier*.

Stejně jako u selekce atributů byla z důvodu příliš vysoké výpočetní náročnosti úspěšnost vybraných parametrů testována pomocí validační sady. Samotné prostory parametrů byly zvoleny pomocí řady experimentů na různých kombinacích parametrů, které byly, na základě oficiálních dokumentací knihoven obsahující využité klasifikační algoritmy, vyhodnoceny jako potenciálně blízké hodnotám optimálním. Optimalizace pomocí algoritmu *TPE* byla ukončena vždy, když proběhlo sto iterací, při kterých nebylo dosaženo navýšení skóre. U obou algoritmů pak na konci každé iterace byly uloženy dosavadní výsledky a také vypsány aktuální informace o stavu optimalizace. Optimalizované parametry byly uloženy do souboru souboru *optimised_hyper_parameters.json*. Optimalizace parametrů všech klasifikátorů na všech sadách atributů trvala celkem 3 dny, 19 hodin a 19 minut. Informace o délce běhu pro jednotlivé klasifikátory na každé ze sad atributů jsou uvedeny v příloze D. Všechny parametry, jejichž hodnoty byly optimalizovány, je možné vidět v následujícím seznamu.

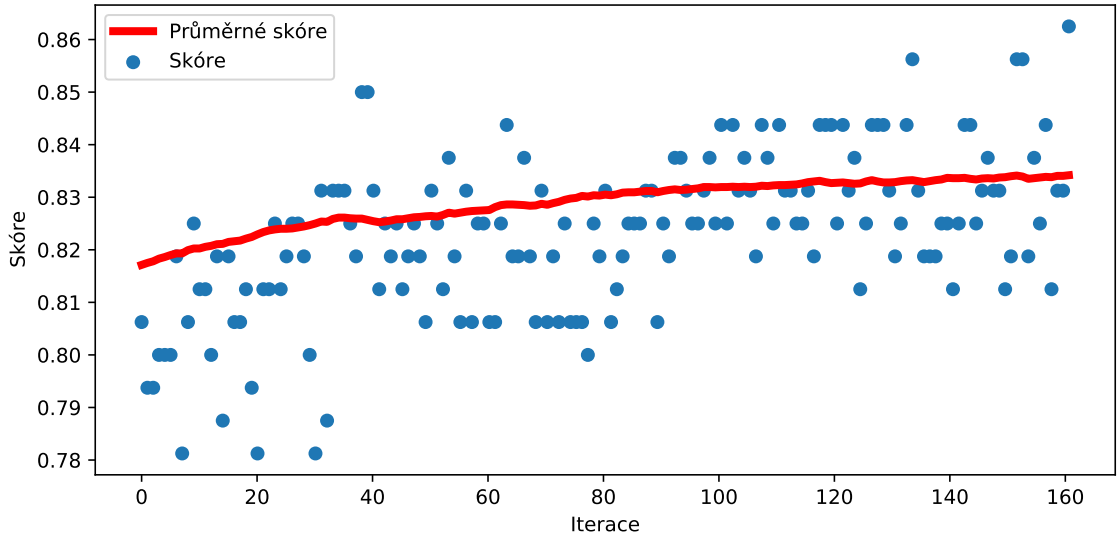
- LogisticRegression (LR) - *penalty*, *fit_intercept*, *C*
- KNearestNeighbors (KNNC) - *n_neighbors*, *weights*, *p*
- MLPClassifier (MLPC) - *solver*, *hidden_layer_sizes*, *activation*, *alpha*, *momentum*, *epsilon*
- DecisionTreeClassifier (DTC) - *criterion*, *min_samples_split*, *min_samples_leaf*, *max_features*, *ccp_alpha*
- SVC_linear (SVCL) - *C*
- SVC_rbf (SVCR) - *C*, *gamma*
- RandomForestClassifier (RFC) - *criterion*, *min_samples_split*, *min_samples_leaf*, *max_features*, *ccp_alpha*, *max_samples*
- XGBClassifier (XGBC) - *n_estimators*, *max_depth*, *min_child_weight*, *subsample*, *colsample_bytree*, *reg_alpha*, *reg_lambda*, *gamma*

Zhodnocení výsledků optimalizace parametrů

Vliv optimalizace parametrů na skóre na validačních sadách je možné vidět v tabulce 5.5 a vliv na dobu trénování a klasifikace pak v příloze E. Průměrné zvýšení skóre na množině všech atributů dosahovalo 5 %, což je velmi podobný výsledek, jakého bylo dosaženo pomocí metod selekce atributů. Na samotných podmnožinách atributů však optimalizace parametrů dosahovala mnohem nižšího nárůstu skóre, kdy na podmnožině získané metodou dopředné selekce nárůst skóre dosahoval pouze 0.32 % a na podmnožině získané metodou zpětné eliminace 1.35 %. Tento výsledek je možné vysvětlit tím, že vyhodnocení u selekce atributů bylo prováděno pomocí klasifikátorů s výchozími hodnotami parametrů. Vybrané podmnožiny tak byly ideální pro tyto hodnoty parametrů a jejich následná optimalizace již nebyla tak přínosná.

Tabulka 5.5: **Rozdíl skóre klasifikace po optimalizaci parametrů na validační sadě**

		LR	KNNC	MLPC	DTC	SVCL	SVCR	RFC	XGBC	Average
EBD	All	0.15%	11.06%	8.82%	4.19%	0.00%	11.06%	4.04%	2.54%	5.23%
	FS	0.15%	0.00%	0.00%	0.45%	0.00%	0.30%	0.00%	0.00%	0.11%
	BE	0.00%	5.23%	4.63%	0.60%	0.00%	0.00%	1.20%	0.75%	1.55%
FMA	All	8.52%	3.98%	4.77%	7.42%	6.80%	2.27%	0.86%	4.30%	4.87%
	FS	0.00%	0.55%	1.25%	5.39%	0.00%	0.00%	-0.70%	2.19%	1.09%
	BE	5.23%	2.58%	1.56%	3.20%	3.83%	0.00%	-0.23%	0.78%	2.12%
GTZAN	All	0.00%	11.88%	8.75%	5.63%	0.00%	6.25%	3.75%	5.00%	5.16%
	FS	0.00%	1.25%	1.25%	-4.38%	0.00%	0.00%	-1.88%	1.88%	-0.24%
	BE	0.00%	1.88%	3.75%	-1.88%	0.00%	0.62%	-3.13%	1.88%	0.39%
Average	All	2.89%	8.97%	7.45%	5.75%	2.27%	6.53%	2.88%	3.95%	5.09%
	FS	0.05%	0.60%	0.83%	0.49%	0.00%	0.10%	-0.86%	1.36%	0.32%
	BE	1.74%	3.23%	3.31%	0.64%	1.28%	0.21%	-0.72%	1.14%	1.35%



Obrázek 5.2: **Ukázka průběhu optimalizace parametrů pomocí algoritmu *TPESampler*.** V obrázku je možné vidět průběh optimalizace klasifikátoru *XGBClassifier* na datové sadě GTZAN až po nejlepší iteraci. Červená čárna značí průměrné skóre sta následujících iterací. Z obrázku je patrný postupný nárůst skóre, jak optimalizační algoritmus konverguje k optimálním hodnotám parametrů.

Kapitola 6

Klasifikace a zhodnocení výsledků

Úspěšnost selekce atributů na testovacích sadách je možné vidět v tabulce 6.1. Z výsledků je patrné, že na testovacích sadách selekce atributů nedosáhla zdaleka tak vysokého nárůstu úspěšnosti klasifikace, jako na sadách validačních. Na podmnožině atributů získaných pomocí metody dopředné selekce bylo dokonce průměrné skóre o 0.36 % horší, než na celé množině atributů. Na podmnožině získané pomocí metody zpětné eliminace pak bylo průměrně skóre o 1.09 % vyšší. Toto dramatické snížení úspěšnosti klasifikace na testovací sadě je pravděpodobně způsobeno nedostatečnou velikostí datových sad a také použitím validace na jediné sadě, namísto křížové validace na několika sadách.

Tabulka 6.1: **Rozdíl skóre klasifikace po selekci atributů na testovací sadě**

		LR	KNNC	MLPC	DTC	SCVL	SVCR	RFC	XGBC	Average
EBD	FS	-3.11%	25.00%	1.79%	-2.75%	-4.07%	4.78%	3.11%	-2.99%	2.72%
	BE	0.12%	9.45%	-0.24%	-1.20%	-0.36%	6.22%	1.79%	0.24%	2.00%
FMA	FS	4.06%	1.88%	-0.75%	-1.31%	2.25%	1.38%	-1.13%	-2.19%	0.52%
	BE	-0.13%	0.12%	6.50%	2.63%	-0.75%	1.50%	0.75%	0.62%	1.41%
GTZAN	FS	-5.00%	-1.50%	-2.00%	-6.50%	-7.00%	2.00%	-9.00%	-5.50%	-4.31%
	BE	-1.00%	0.50%	-1.00%	-3.50%	-0.50%	3.00%	0.50%	1.00%	-0.13%
Average	FS	-1.35%	8.46%	-0.32%	-3.52%	-2.94%	2.72%	-2.34%	-3.56%	-0.36%
	BE	-0.34%	3.36%	1.75%	-0.69%	-0.54%	3.57%	1.01%	0.62%	1.09%

Úspěšnost optimalizace parametrů na testovacích sadách je možné vidět v tabulce 6.2. V porovnání se selekcí atributů byla úspěšnost optimalizace na testovací sadě výrazně vyšší. Průměrný nárůst skóre činil 2.83 % na celé množině atributů, 0.78 % na podmnožině získané pomocí metod dopředné selekce a 2.28 % na podmnožině získané pomocí metody zpětné eliminace.

Tabulka 6.2: **Rozdíl skóre klasifikace po optimalizaci parametrů na testovací sadě**

		LR	KNNC	MLPC	DTC	SVCL	SVCR	RFC	XGBC	Average
EBD	All	-0.48%	7.89%	2.63%	-0.36%	0.00%	7.18%	2.39%	0.12%	2.42%
	FS	1.56%	0.00%	0.12%	3.11%	0.00%	1.67%	-0.36%	1.08%	0.90%
	BE	0.00%	7.89%	1.91%	1.91%	0.12%	0.00%	2.51%	-0.36%	1.75%
FMA	All	6.38%	4.31%	12.00%	5.75%	7.13%	1.94%	0.44%	1.69%	4.96%
	FS	0.00%	0.56%	4.75%	4.50%	0.00%	0.00%	0.06%	1.25%	1.39%
	BE	5.44%	4.75%	4.31%	3.13%	6.69%	0.00%	0.31%	0.69%	3.17%
GTZAN	All	1.00%	-3.00%	1.50%	0.50%	0.00%	5.00%	0.00%	4.00%	1.13%
	FS	0.00%	1.00%	2.00%	2.00%	0.00%	0.00%	-3.50%	-1.00%	0.06%
	BE	0.00%	4.00%	2.00%	4.00%	0.00%	4.00%	1.50%	0.00%	1.94%
Average	All	2.30%	3.07%	5.38%	1.96%	2.38%	4.71%	0.94%	1.94%	2.83%
	FS	0.52%	0.52%	2.29%	3.20%	0.00%	0.56%	-1.27%	0.44%	0.78%
	BE	1.81%	5.55%	2.74%	3.01%	2.27%	1.33%	1.44%	0.11%	2.28%

Úspěšnost klasifikace klasifikátorů s výchozími parametry je možné vidět v tabulce 6.3 a klasifikátorů s optimalizovanými parametry v tabulce 6.4. Dobu trénování a klasifikace pak v příloze E. Nejvyšší skóre na datové sadě EBD činilo 86.48 % a bylo dosaženo pomocí klasifikátoru *XGBClassifier* s výchozími parametry na podmnožině atributů získané pomocí metody zpětné eliminace. Na datové sadě FMA bylo nejlepšího skóre 64.56 % dosaženo opět pomocí klasifikátoru *XGBClassifier*, tentokrát však s optimalizovanými parametry a na celé množině atributů. Na datové sadě GTZAN pak bylo nejlepšího skóre 82.00 % dosaženo hned ve čtyřech případech. V prvním z nich pomocí optimalizovaného klasifikátoru *MLPClassifier* na celé množině atributů, v druhém a třetím pak pomocí klasifikátoru *SVC* bez použití jádrové metody, a to jak s výchozími, tak optimalizovanými parametry na celé množině atributů. V posledním případě pak opět pomocí klasifikátoru *SVC*, tentokrát však s použitím jádrové metody *RBF* a optimalizovanými parametry na podmnožině atributů získané pomocí metody zpětné eliminace.

Tabulka 6.3: **Skóre na testovacích sadách klasifikátorů s výchozími parametry**

		LR	KNNC	MLPC	DTC	SVCL	SVCR	RFC	XGBC
EBD	All	85.53%	53.11%	83.73%	72.85%	84.57%	78.23%	79.55%	86.24%
	FS	82.42%	78.11%	85.53%	70.10%	80.50%	83.01%	82.66%	83.25%
	BE	85.65%	62.56%	83.49%	71.65%	84.21%	84.45%	81.34%	86.48%
FMA	All	54.56%	47.63%	50.69%	37.06%	53.81%	60.00%	58.06%	62.88%
	FS	58.63%	49.50%	49.94%	35.75%	56.06%	61.38%	56.94%	60.69%
	BE	54.44%	47.75%	57.19%	39.69%	53.06%	61.50%	58.81%	63.50%
GTZAN	All	80.00%	71.00%	80.50%	57.50%	82.00%	75.00%	76.50%	77.00%
	FS	75.00%	69.50%	78.50%	51.00%	75.00%	77.00%	67.50%	71.50%
	BE	79.00%	71.50%	79.50%	54.00%	81.50%	78.00%	77.00%	78.00%

Tabulka 6.4: **Skóre na testovacích sadách klasifikátorů s optimalizovanými parametry**

		LR	KNNC	MLPC	DTC	SVCL	SVCR	RFC	XGBC
EBD	All	85.05%	61.00%	86.36%	72.49%	84.57%	85.41%	81.94%	86.36%
	FS	83.97%	78.11%	85.65%	73.21%	80.50%	84.69%	82.30%	84.33%
	BE	85.65%	70.45%	85.41%	73.56%	84.33%	84.45%	83.85%	86.12%
FMA	All	60.94%	51.94%	62.69%	42.81%	60.94%	61.94%	58.50%	64.56%
	FS	58.63%	50.06%	54.69%	40.25%	56.06%	61.38%	57.00%	61.94%
	BE	59.88%	52.50%	61.50%	42.81%	59.75%	61.50%	59.13%	64.19%
GTZAN	All	81.00%	68.00%	82.00%	58.00%	82.00%	80.00%	76.50%	81.00%
	FS	75.00%	70.50%	80.50%	53.00%	75.00%	77.00%	64.00%	70.50%
	BE	79.00%	75.50%	81.50%	58.00%	81.50%	82.00%	78.50%	78.00%

Na základě výsledků uvedených v posledních dvou kapitolách je tak možné prohlásit, že na daných třech datových sadách se extrahované atributy ukázaly být všechny hodnotné a na místo jejich selekce tak bylo výhodnější je ponechat všechny, pouze pak optimalizovat parametry klasifikačních algoritmů, které by pomocí vestavěných metod selekce případné nevhodné atributy vyřadily až v průběhu trénování. Jako nejúspěšnější klasifikační algoritmy se ukázaly být *XGBClassifier*, *MLPClassifier* a *SVC*.

Vzhledem k těmto poznatkům tak byla pro každou z datových sad provedena finální optimalizace parametrů těchto nejúspěšnějších klasifikátorů na celé množině atributů. Tentokrát však byla úspěšnost zvolených parametrů vyhodnocována pomocí pětinásobné křížové validace, což by mělo zaručit, že optimalizace bude ještě úspěšnější. Tento předpoklad se však ukázal být pravdivý pouze v některých případech. Konkrétně bylo dosaženo zvýšení skóre o 0.12 % u klasifikátoru *MLPClassifier* na datové sadě FMA a u klasifikátoru *XGBClassifier* o 1.2 % u datové sady EBD a o 2.5 % u datové sady GTZAN. V ostatních případech dosahovalo skóre stejné či nižší úspěšnosti, jako u optimalizace s využitím validace pomocí validační sady. Zmíněný nárůst skóre u klasifikátoru *XGBClassifier* však byl dostatečný na to, aby byl tento nyní nejúspěšnější ze všech klasifikačních algoritmů na všech datových sadách.

Tato finální optimalizace parametrů trvala celkem 3 dny, 3 hodiny a 24 minut. Podrobné informace o finální optimalizaci a také matice predikcí nejúspěšnějších klasifikátorů je možné nalézt v tabulce v příloze E. Optimalizované hodnoty parametrů nejúspěšnějších klasifikátorů je možné vidět v tabulce 6.5. Nejúspěšnější natrénované klasifikátory byly uloženy a je možné je využít pro predikci a anotaci žánrů skladeb z lokálního archivu.

Tabulka 6.5: **Nejúspěšnější parametry pro klasifikátor *XGBClassifier* pro každou z datových sad.** Parametr *n_estimators* značí počet iterací (počet základních klasifikátorů). Parametr *learning_rate* značí rychlost učení, tento parametr nebyl optimalizován, jeho hodnota byla pouze snížena z výchozí hodnoty kvůli možnosti přesnější optimalizace. Parametr *max_depth* značí maximální hloubku základních klasifikátorů (rozhodovacích stromů). Parametr *min_child_weight* značí minimální součet vah prvků v poduzlech děleného uzlu. Parametr *subsample* značí velikost množiny prvků, na které je každý základní klasifikátor natrénován. Parametr *colsample_bytree* značí velikost množiny atributů, na které je každý základní klasifikátor natrénován. Parametr *reg_alpha* značí hodnotu „intenzity“ L1 regularizace. Parametr *reg_lambda* značí hodnotu „intenzity“ L2 regularizace. Parametr *gamma* značí značí hodnotu „intenzity“ pseudo regularizace.

	EBD	FMA	GTZAN
n_estimators	459	454	271
learning_rate	0.1	0.1	0.1
max_depth	8	3	8
min_child_weight	8	0	3
subsample	0.8	0.6	0.5
colsample_bytree	1.0	0.75	0.75
reg_alpha	0.095	0.44	0.092
reg_lambda	3.235	6.268	0.226
gamma	0.09	0.209	0.102

Tabulka 6.6: **Tabulka nejvyššího dosaženého skóre a doby trvání trénování a klasifikace pro každou z datových sad.** Ve všech případech bylo dosaženo pomocí klasifikačního algoritmu *XGBClassifier* s optimalizovanými parametry a na množině všech atributů. Nižší úspěšnost klasifikace na datové sadě FMA je možné vysvětlit tím, že dělí skladby do velmi nejednoznačně definovaných žánrů.

	Úspěšnost klasifikace	Čas trénování	Čas klasifikace
Datová sada EBD	87.56 %	0:02:51	0:00:00
Datová sada FMA	64.56 %	0:06:50	0:00:00
Datová sada GTZAN	83.50 %	0:00:26	0:00:00

Kapitola 7

Závěr

Tato práce se zabývá klasifikací hudebních souborů dle žánrů či tanečních stylů pomocí algoritmů strojového učení. Je zde popsán teoretický základ k pochopení šíření, záznamu a uchování zvukových vln a také několika klasifikačních algoritmů. Porovnány byly také dvě metody selekce atributů, a to dopředná selekce a zpětná eliminace. Na ani jedné z datových sad se však selekce neosvědčila a klasifikátory s optimalizovanými parametry byly v drtivé většině případů schopny na celé množině atributů dosáhnout stejného či vyššího skóre.

Jako nejúspěšnější se ukázal být model *XGBClassifier*, který dosáhl nejvyšší úspěšnosti klasifikace na všech třech datových sadách. Výsledky klasifikace dosáhly 86.48 % na datové sadě „Extended Ballroom Dataset“ rozřazené dle třinácti tanečních stylů, 64.56 % na datové sadě „A Dataset For Music Analysis“ rozřazené dle osmi hudebních žánrů a 82.00 % na datové sadě „GTZAN“ rozřazené dle desíti hudebních žánrů.

Velmi dobrých výsledků však dosáhl i klasifikátor *MLPClassifier*, který byl zároveň podstatně rychlejší, než klasifikátor *XGBClassifier*. Jelikož klasifikátor *MLPClassifier* je pouze velmi jednoduchým případem vícevrstvého perceptronu, zajímavou oblastí dalšího výzkumu by mohla být klasifikace pomocí pečlivě navržených neuronových sítí, které by mohly dosáhnout stejné či vyšší úspěšnosti klasifikace, a to rychleji, než metody posilování gradientu. Pomocí neuronových sítí by také bylo potenciálně možné klasifikovat neupravené hudební soubory a zcela se tak vyhnout extrakci atributů.

Dalšího navýšení úspěšnosti klasifikace by mohlo být dosaženo použitím rozsáhlejší datové sady obsahující celé skladby. Klasifikační algoritmy by tak měly jednak možnost natrénovat se na více datech, ale také rozpoznat více vlastností skladeb týkajících se jejich vývoje v čase, jako například jestli skladba obsahuje refrén.

Automatická klasifikace hudebních souborů se ukázala jako proveditelná a s dostatečným množstvím trénovacích dat a budoucím vývojem klasifikačních algoritmů by tak jednou mohla částečně či úplně nahradit ručně prováděnou anotaci skladeb.

Literatura

[1] *DB: What is a decibel?* [online]. University of New South Wales [cit. 2020-04-02]. Dostupné z: https://www.animations.physics.unsw.edu.au/jw/dB.htm.

[2] Aggarwal, C. C. *Data Classification: Algorithms and Applications*. 1. vyd. Chapman & Hall/CRC, 2014. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. ISBN 1466586745.

[3] Chu, W.-T. *Musical Genre Classification* [online]. [cit. 2020-04-10]. Dostupné z: https://www.cs.ccu.edu.tw/~wtchu/courses/2014f_MCA/Lectures/Lecture%209%20Audio%20and%20Music%20Analysis%202.pdf.

[4] Defferrard, M., Benzi, K., Vandergheynst, P. a Bresson, X. FMA: A Dataset for Music Analysis. In: *18th International Society for Music Information Retrieval Conference (ISMIR)*. 2017. Dostupné z: https://arxiv.org/abs/1612.01840.

[5] Gouyon, F., Pachet, F. a Delerue, O. On the Use of Zero-Crossing Rate for an Application of Classification of Percussive Sounds. In: *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00*. 2000.

[6] Gwardys, G. a Grzywczak, D. Deep Image Features in Music Information Retrieval. *International Journal of Electronics and Telecommunications*. 2014, sv. 60, č. 4, s. 321–326.

[7] Han, J., Kamber, M. a Pei, J. *Data Mining: Concepts and Techniques*. 3. vyd. Elsevier Science, 2011. The Morgan Kaufmann Series in Data Management Systems. ISBN 9780123814807.

[8] Honzík, P. *Strojové učení* [online]. Říjen 2006 [cit. 2020-04-10]. Dostupné z: http://midas.uamt.feec.vutbr.cz/STU/Lectures/Honzik%20-%20Strojove_uceni_S.pdf.

[9] Jiang, D.-N., Lu, L., Zhang, H.-J., Tao, J.-H. a Cai, L.-H. Music type classification by spectral contrast feature. In: *Proceedings. IEEE International Conference on Multimedia and Expo*. 2002, sv. 1, s. 113–116.

[10] Kaur, C. a Kumar, R. Study and analysis of feature based automatic music genre classification using Gaussian mixture model. In: *2017 International Conference on Inventive Computing and Informatics (ICICI)*. Listopad 2017, s. 465–468. DOI: 10.1109/ICICI.2017.8365395.

[11] Kotsiantis, S., Kanellopoulos, D. a Pintelas, P. Data Preprocessing for Supervised Learning. *International Journal of Computer Science*. Leden 2006, sv. 1, s. 111–117.

[12] Leisure, R. G. Acoustic Waves in Solids. In: *Ultrasonic Spectroscopy: Applications in Condensed Matter Physics and Materials Science*. Cambridge University Press, 2017, s. 56–93. DOI: 10.1017/9781316658901.004.

[13] Lerch, A. *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*. 1st. Wiley-IEEE Press, 2012. ISBN 111826682X.

[14] Lin, Y., Yang, Y., Chen, H. H., Liao, I. a Ho, Y. Exploiting genre for music emotion classification. In: *2009 IEEE International Conference on Multimedia and Expo*. červen 2009, s. 618–621. DOI: 10.1109/ICME.2009.5202572. ISSN 1945-7871.

[15] Marchand, U. a Peeters, G. The Extended Ballroom Dataset. *ISMIR 2016 Late-Breaking Session*. 2016. New-York, USA.

[16] McInnes, L., Healy, J. a Melville, J. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. 2018 [cit. 2020-04-10].

[17] McKay, C. a Fujinaga, I. Musical Genre Classification: Is It Worth Pursuing and How Can It be Improved? In:. Leden 2006.

[18] Mitchell, T. M. *Machine Learning*. 1. vyd. USA: McGraw-Hill, Inc., 1997. ISBN 0070428077.

[19] Murauer, B. a Specht, G. Detecting Music Genre Using Extreme Gradient Boosting. In: *Companion Proceedings of the The Web Conference 2018*. 2018, s. 1923–1927. DOI: 10.1145/3184558.3191822.

[20] Orjesek, R., Jarina, R., Chmulik, M. a Kuba, M. DNN Based Music Emotion Recognition from Raw Audio Signal. In: *2019 29th International Conference Radioelektronika (RADIOELEKTRONIKA)*. 2019, s. 1–4.

[21] Rao, P. Audio Signal Processing. In: *Speech, Audio, Image and Biomedical Signal Processing using Neural Networks*. Springer Publishing Company, Incorporated, 2007.

[22] Schedl, M., Gomez, E. a Urbano, J. *Music Information Retrieval: Recent Developments and Applications*. Hanover, MA, USA: Now Publishers Inc., 2014. ISBN 1601988060.

[23] Schmidt Jones, C. *Understanding Basic Music Theory*. USA: 12th Media Services, 2018. ISBN 1680921541.

[24] Seger, C. *An investigation of categorical variable encoding techniques...* Stockholm, Sweden, 2018. Diplomová práce. School of Electrical Engineering and Computer Science.

[25] Shalev Shwartz, S. a Ben David, S. *Understanding Machine Learning: From Theory to Algorithms*. 1. vyd. USA: Cambridge University Press, 2014. ISBN 1107057132.

[26] Smith, S. W. *The Scientist and Engineer's Guide to Digital Signal Processing*. 1. vyd. USA: California Technical Publishing, 1997. ISBN 0966017633.

[27] Takayama, K. *Encoding Categorical Variables with Ambiguity* [online]. [cit. 2020-04-13]. Dostupné z: http://www.di.uniba.it/~loglisci/NFMCP2019/NFMCP/nfMCP2019_paper_8.pdf.

[28] Tjoa, S. *Notes on Music Information Retrieval* [online]. [cit. 2020-07-25]. Dostupné z: https://musicinformationretrieval.com/.

[29] Tzanetakis, G. a Cook, P. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*. Červenec 2002, sv. 10, č. 5, s. 293–302. DOI: 10.1109/TSA.2002.800560. ISSN 1063-6676.

Příloha A

Přílohy k extrakci atributů

V následujícím seznamu jsou uvedeny zvolené hodnoty parametrů extrakční funkce *MusicExtractor* knihovny Essentia.

- **'analysisSampleRate': 44100** - Vzorkovací frekvence skladeb pro analýzu, 44100 Hz (CD kvalita), skladby jsou případně převzorkovány.
- **'endTime': 1e+06** - Jak dlouhý úsek skladby má být analyzován, nastaveno na celou skladbu.
- **'gfccStats':** [**"mean", "cov", "icov"**] - Jaké statistické funkce použít pro zpracování hodnot gamma tónových kepstrálních koeficientů, nastaveno na průměr, kovarianci a inverzní kovarianci.
- **'loudnessFrameSize': 88200** - Délka okna pro časově-frekvenční analýzu pro odhad hlasitosti, nastaveno na hodnotu 88200 vzorků, což odpovídá úseku dlouhému 2 sekundy. Delší okno je vhodné pro přesnější odhad.
- **'loudnessHopSize': 44100** - Posun okna pro časově-frekvenční analýzu pro odhad hlasitosti, nastaveno na délku poloviny okna, takže se každé okno z poloviny překrývá z oknem předchozím.
- **'lowlevelFrameSize': 2048** - Délka okna pro časově-frekvenční analýzu pro výpočet nízko-úrovňových atributů, nastaveno na hodnotu 2048 vzorků, což odpovídá úseku dlouhému 46 milisekund. Kratší okno je vhodné pro přesnější časovou analýzu.
- **'lowlevelHopSize': 1024** - Posun okna pro časově-frekvenční analýzu pro výpočet nízko-úrovňových atributů, nastaveno na délku poloviny okna, takže se každé okno z poloviny překrývá z oknem předchozím.
- **'lowlevelSilentFrames': 'noise'** - Zpracování oken neobsahujících zvuk, ponechána výchozí hodnota 'noise' – doplnění šumem.
- **'lowlevelStats':** [**"mean", "var", "stdev", "median", "min", "max", "dmean", "dmean2", "dvar", "dvar2"**] - Jaké statistické funkce použít pro zpracování hodnot nízkoúrovňových atributů, nastaveno na průměr, rozptyl, směrodatná odchylka, medián, minimální hodnotu, maximální hodnotu, průměr první derivace, průměr druhé derivace, rozptyl první derivace a rozptyl druhé derivace.

- **'lowlevelWindowType': 'blackmanharris62'** - Typ okénkové funkce pro zpracování oken spektrogramu pro nízkoúrovňové atributy. Nastaveno typ blackmanharris62.
- **'lowlevelZeroPadding': 0** - Odsazení oken pro nízkoúrovňové atributy, nepoužito.
- **'mfccStats': ["mean", "cov", "icov"]** - Jaké statistické funkce použít pro zpracování hodnot Mel-frekvenčních kepstrálních koeficientů, nastaveno na průměr, kovarianci a inverzní kovarianci.
- **'profile': ''** - Profil pro načtení parametrů, nepoužito.
- **'requireMbid': False** - Vyžadovat tag MusicBrainz databáze, nepoužito.
- **'rhythmMaxTempo': 208** - Maximální hodnota tempa, nastaveno na hodnotu 208 úderů za minutu, což by mělo zamezit odhadům tempa ze čtvrtdob a podobně.
- **'rhythmMethod': 'degara'** - Metoda pro odhad tempa, nastaveno na metodu degara.
- **'rhythmMinTempo': 40** - Minimální hodnota tempa, nastaveno na hodnotu 40 úderů za minutu.
- **'rhythmStats': ["mean", "var", "stdev", "median", "min", "max", "dmean", "dmean2", "dvar", "dvar2"]** - Jaké statistické funkce použít pro zpracování hodnot rytmických atributů, nastaveno na průměr, rozptyl, směrodatná odchylka, medián, minimální hodnotu, maximální hodnotu, průměr první derivace, průměr druhé derivace, rozptyl první derivace a rozptyl druhé derivace.
- **'startTime': 0** - Od jakého času začít analýzu, nastaveno na hodnotu 0 sekund.
- **'tonalFrameSize': 4096** - Délka okna pro časově-frekvenční analýzu pro výpočet melodických atributů, nastaveno na hodnotu 4096 vzorků, což odpovídá úseku dlouhému 92 milisekund. Delší okno je vhodné pro přesnější frekvenční odhad.
- **'tonalHopSize': 2048** - Posun okna pro časově-frekvenční analýzu pro výpočet melodických atributů, nastaveno na délku poloviny okna, takže se každé okno z poloviny překrývá z oknem předchozím.
- **'tonalSilentFrames': 'noise'** - Zpracování oken neobsahujících zvuk, ponechána výchozí hodnota 'noise' - doplnění šumem.
- **'tonalStats': ["mean", "var", "stdev", "median", "min", "max", "dmean", "dmean2", "dvar", "dvar2"]** - Jaké statistické funkce použít pro zpracování hodnot melodických atributů, nastaveno na průměr, rozptyl, směrodatná odchylka, medián, minimální hodnotu, maximální hodnotu, průměr první derivace, průměr druhé derivace, rozptyl první derivace a rozptyl druhé derivace.
- **'tonalWindowType': 'blackmanharris62'** - Typ okénkové funkce pro zpracování oken spektrogramu pro melodické atributy. Nastaveno typ blackmanharris62.
- **'tonalZeroPadding': 0** - Odsazení oken melodické atributy, nepoužito.

Příloha B

Přílohy k výběru klasifikačních algoritmů a jejich parametrů

Tabulka B.1: **Výsledky klasifikace pro výběr parametrů klasifikačních algoritmů na validační sadě**

	EBD		FMA		GTZAN	
	librosa	essentia	librosa	essentia	librosa	essentia
LogisticRegression_newton-cg	60.24%	87.14%	44.77%	50.47%	75.63%	84.38%
LogisticRegression_lbfgs	60.24%	87.14%	44.77%	50.39%	75.63%	84.38%
LogisticRegression_liblinear	58.15%	86.70%	46.88%	50.78%	70.00%	77.50%
LogisticRegression_sag	61.29%	85.95%	48.05%	56.17%	73.13%	84.38%
LogisticRegression_saga	62.18%	85.95%	49.61%	56.80%	73.13%	84.38%
KNeighborsClassifier_ball_tree	41.70%	54.11%	43.98%	45.31%	65.63%	58.13%
KNeighborsClassifier_kd_tree	41.70%	54.11%	43.98%	45.31%	65.63%	58.13%
KNeighborsClassifier_brute	41.70%	54.11%	43.98%	45.31%	65.63%	58.13%
MLPClassifier_lbfgs	59.94%	82.96%	49.69%	54.30%	75.00%	82.50%
MLPClassifier_sgd	60.99%	83.86%	52.73%	51.72%	74.38%	68.75%
MLPClassifier_adam	59.94%	80.27%	52.58%	57.03%	76.88%	78.13%
SVC_linear	60.24%	87.29%	47.81%	52.89%	75.63%	84.38%
SVC_poly	38.57%	40.36%	34.06%	32.97%	34.38%	36.88%
SVC_rbf	57.70%	76.68%	53.67%	57.81%	68.13%	78.75%
SVC_sigmoid	46.94%	73.84%	42.34%	53.05%	68.13%	80.00%

Tabulka B.2: **Doba trénování a klasifikace pro výběr parametrů klasifikačních algoritmů**

	EBD		FMA		GTZAN	
	librosa	essentia	librosa	essentia	librosa	essentia
LogisticRegression_newton-cg	0:00:07	0:00:15	0:00:23	0:01:22	0:00:00	0:00:02
LogisticRegression_lbfgs	0:00:06	0:00:14	0:00:07	0:00:25	0:00:00	0:00:05
LogisticRegression_liblinear	0:00:24	0:01:11	0:01:11	0:04:39	0:00:02	0:00:10
LogisticRegression_sag	0:05:24	0:15:23	0:12:37	0:31:35	0:00:19	0:01:59
LogisticRegression_saga	0:08:24	0:22:26	0:17:24	0:43:51	0:00:34	0:03:01
KNeighborsClassifier_ball_tree	0:00:00	0:00:01	0:00:01	0:00:05	0:00:00	0:00:00
KNeighborsClassifier_kd_tree	0:00:00	0:00:02	0:00:02	0:00:09	0:00:00	0:00:00
KNeighborsClassifier_brute	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00
MLPClassifier_lbfgs	0:00:00	0:00:01	0:00:03	0:00:08	0:00:00	0:00:00
MLPClassifier_sgd	0:00:03	0:00:11	0:00:07	0:00:18	0:00:00	0:00:03
MLPClassifier_adam	0:00:02	0:00:05	0:00:06	0:00:09	0:00:00	0:00:01
SVC_linear	0:00:08	0:00:23	0:00:34	0:01:42	0:00:00	0:00:01
SVC_poly	0:00:18	0:00:59	0:01:00	0:03:09	0:00:00	0:00:02
SVC_rbf	0:00:12	0:00:39	0:00:43	0:02:14	0:00:00	0:00:02
SVC_sigmoid	0:00:10	0:00:29	0:00:36	0:01:54	0:00:00	0:00:01

Příloha C

Přílohy k selekci atributů

Tabulka C.1: **Doba běhu selekce atributů pomocí dopředné selekce**

	EBD		FMA		GTZAN		Total
	librosa	essentia	librosa	essentia	librosa	essentia	
LogisticRegression	00:01:27	00:14:28	00:02:13	00:36:29	00:00:10	00:02:01	00:56:48
KneighborsClassifier	00:00:01	00:00:48	00:00:08	00:01:49	00:00:00	00:00:07	00:02:53
MLPClassifier	00:06:09	00:38:00	00:08:04	00:43:26	00:00:57	00:06:14	01:42:50
DecisionTreeClassifier	00:00:19	00:02:19	00:03:55	00:51:36	00:00:02	00:01:04	00:59:15
SVC_linear	00:01:49	00:06:08	00:27:39	01:12:36	00:00:03	00:00:46	01:49:01
SVC_rbf	00:03:38	00:19:29	00:07:34	04:46:00	00:00:08	00:01:56	05:18:45
RandomForestClassifier	00:00:16	00:01:25	00:00:53	00:06:18	00:00:03	00:00:56	00:09:51
XGBClassifier	00:04:13	00:26:45	00:11:51	02:12:49	00:01:12	00:04:50	03:01:40
Total	00:17:52	01:49:22	01:02:17	10:31:03	00:02:35	00:17:54	14:01:03

Tabulka C.2: **Doba běhu selekce atributů pomocí zpětné eliminace**

	EBD		FMA		GTZAN		Total
	librosa	essentia	librosa	essentia	librosa	essentia	
LogisticRegression	00:03:33	06:23:10	00:03:54	04:37:50	00:00:17	01:58:18	13:07:02
KneighborsClassifier	00:00:07	00:05:16	00:00:14	00:10:41	00:00:02	00:00:42	00:17:02
MLPClassifier	00:02:33	00:16:24	00:02:37	00:52:06	00:00:37	00:04:00	01:18:17
DecisionTreeClassifier	00:01:16	00:55:11	00:03:20	00:47:59	00:00:15	00:04:54	01:52:55
SVC_linear	00:03:27	10:45:42	00:21:58	13:49:21	00:00:16	00:55:39	25:56:23
SVC_rbf	00:06:26	07:30:54	00:21:06	27:25:49	00:00:25	00:42:39	36:07:19
RandomForestClassifier	00:00:25	00:09:50	00:00:39	00:12:11	00:00:06	00:02:37	00:25:48
XGBClassifier	00:10:03	05:17:07	00:09:37	06:36:02	00:01:24	03:34:36	15:48:49
Total	00:27:50	31:23:34	01:03:25	54:31:59	00:03:22	07:23:25	94:53:35

Tabulka C.3: **Celková doba běhu trénování a klasifikace na atributech extrahovaných pomocí knihovny Librosa**

		LR	KNNC	MLPC	DTC	SVCL	SVCR	RFC	XGBC
EBD	All	0:00:05	0:00:00	0:00:03	0:00:02	0:00:08	0:00:12	0:00:00	0:00:13
	FS	0:00:00	0:00:00	0:00:12	0:00:00	0:00:01	0:00:02	0:00:00	0:00:09
	BE	0:00:05	0:00:00	0:00:01	0:00:02	0:00:02	0:00:02	0:00:00	0:00:03
FMA	All	0:00:07	0:00:00	0:00:06	0:00:05	0:00:33	0:00:43	0:00:01	0:00:22
	FS	0:00:01	0:00:00	0:00:04	0:00:04	0:00:20	0:00:07	0:00:01	0:00:15
	BE	0:00:01	0:00:00	0:00:04	0:00:04	0:00:33	0:00:09	0:00:01	0:00:22
GTZAN	All	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:04
	FS	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01
	BE	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01

Tabulka C.4: **Celková doba běhu trénování a klasifikace na atributech extrahovaných pomocí knihovny Essentia**

		LR	KNNC	MLPC	DTC	SVCL	SVCR	RFC	XGBC
EBD	All	0:00:14	0:00:00	0:00:05	0:00:07	0:00:23	0:00:39	0:00:01	0:00:40
	FS	0:00:01	0:00:00	0:00:01	0:00:00	0:00:00	0:00:02	0:00:00	0:00:03
	BE	0:00:09	0:00:00	0:00:05	0:00:05	0:00:22	0:00:08	0:00:00	0:00:40
FMA	All	0:00:25	0:00:00	0:00:09	0:00:17	0:01:42	0:02:14	0:00:02	0:01:14
	FS	0:00:02	0:00:00	0:00:07	0:00:06	0:00:07	0:00:15	0:00:00	0:00:27
	BE	0:00:22	0:00:00	0:00:12	0:00:17	0:01:28	0:01:24	0:00:02	0:01:14
GTZAN	All	0:00:05	0:00:00	0:00:01	0:00:01	0:00:01	0:00:02	0:00:00	0:00:12
	FS	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01
	BE	0:00:02	0:00:00	0:00:01	0:00:01	0:00:00	0:00:01	0:00:00	0:00:08

Příloha D

Přílohy k optimalizaci parametrů

Tabulka D.1: **Doba běhu optimalizace parametrů na atributech extrahovaných pomocí knihovny Essentia**

		LR	KNNC	MLPC	DTC	SVCL	SVCR	RFC	XGBC	Total
EBD	All	00:01:29	00:00:37	00:35:36	00:29:30	00:02:22	00:48:03	03:11:00	04:44:32	09:53:09
	FS	00:00:15	00:00:03	00:10:24	00:00:39	00:00:08	00:03:29	00:01:57	00:21:30	00:38:25
	BE	00:01:00	00:00:18	00:22:05	00:15:16	00:02:18	00:11:13	02:19:16	04:57:46	08:09:12
FMA	All	00:08:03	00:02:13	01:07:24	00:42:54	00:10:22	02:33:06	09:47:20	15:59:44	30:31:06
	FS	00:09:08	00:00:05	00:23:20	00:11:05	00:15:45	00:18:00	01:26:22	03:17:07	06:00:52
	BE	00:08:47	00:02:11	02:45:05	00:44:03	00:08:54	01:35:59	14:27:07	12:14:05	32:06:11
GTZAN	All	00:00:37	00:00:03	00:09:03	00:02:00	00:00:10	00:02:22	00:34:43	01:22:07	02:11:05
	FS	00:00:03	00:00:01	00:02:24	00:00:20	00:00:00	00:00:15	00:01:12	00:11:57	00:16:12
	BE	00:00:19	00:00:02	00:24:58	00:01:44	00:00:05	00:01:03	00:22:02	00:42:38	01:32:51
Total		00:29:41	00:05:33	06:00:19	02:27:31	00:40:04	05:33:30	32:10:59	43:51:26	91:19:03

Tabulka D.2: **Celková doba běhu trénování a klasifikace na atributech extrahovaných pomocí knihovny Essentia po optimalizaci parametrů**

		LR	KNNC	MLPC	DTC	SVCL	SVCR	RFC	XGBC
EBD	All	0:00:02	0:00:01	0:00:37	0:00:13	0:00:23	0:00:24	0:00:53	0:01:44
	FS	0:00:00	0:00:00	0:00:01	0:00:00	0:00:00	0:00:02	0:00:00	0:00:07
	BE	0:00:09	0:00:00	0:00:33	0:00:05	0:00:22	0:00:08	0:01:08	0:01:37
FMA	All	0:00:04	0:00:07	0:00:44	0:00:19	0:01:47	0:01:59	0:03:45	0:02:27
	FS	0:00:02	0:00:00	0:00:04	0:00:03	0:00:07	0:00:15	0:00:24	0:01:38
	BE	0:00:07	0:00:06	0:00:34	0:00:19	0:01:33	0:01:24	0:04:42	0:03:15
GTZAN	All	0:00:02	0:00:00	0:00:08	0:00:01	0:00:01	0:00:02	0:00:09	0:00:15
	FS	0:00:00	0:00:00	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01
	BE	0:00:02	0:00:00	0:00:07	0:00:00	0:00:00	0:00:00	0:00:04	0:00:06

Příloha E

Přílohy ke klasifikaci a zhodnocení výsledků

Tabulka E.1: **Celková doba běhu trénování a klasifikace na atributech extrahovaných pomocí knihovny Essentia s výchozími parametry na testovací sadě**

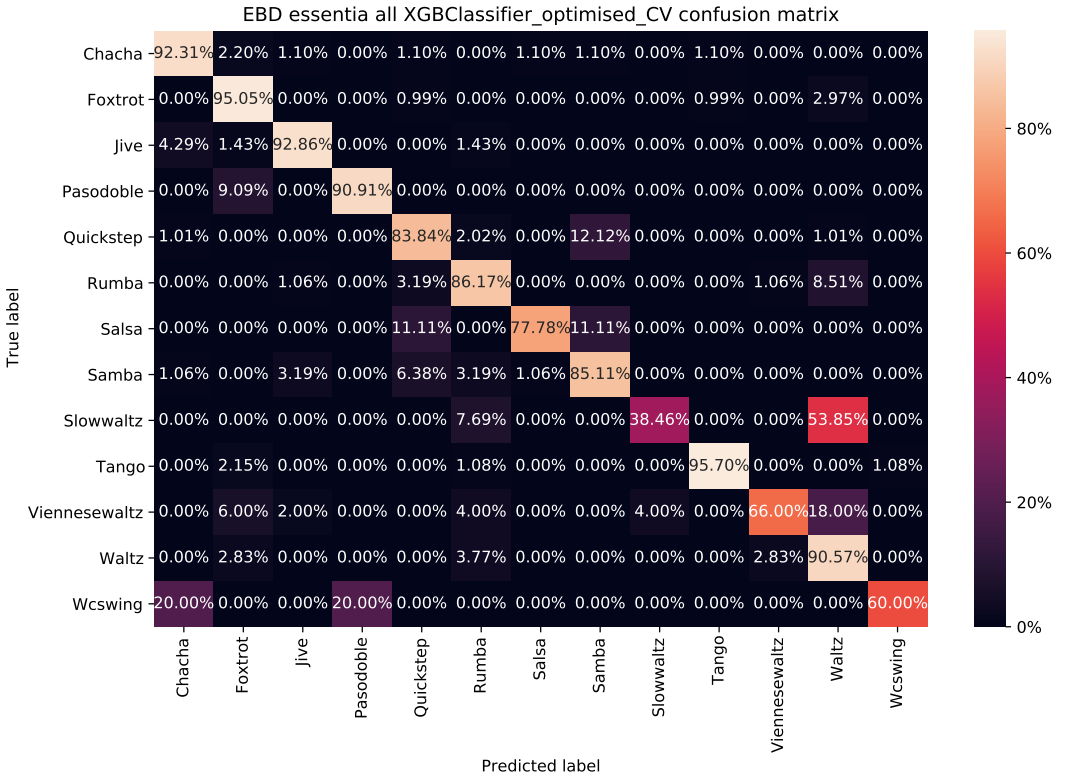
		LR	KNNC	MLPC	DTC	SVCL	SVCR	RFC	XGBC
EBD	All	0:00:17	0:00:00	0:00:06	0:00:10	0:00:34	0:00:58	0:00:01	0:00:48
	FS	0:00:01	0:00:00	0:00:02	0:00:00	0:00:01	0:00:03	0:00:00	0:00:04
	BE	0:00:11	0:00:00	0:00:06	0:00:06	0:00:33	0:00:12	0:00:01	0:00:48
FMA	All	0:00:34	0:00:00	0:00:14	0:00:22	0:02:33	0:03:22	0:00:03	0:01:27
	FS	0:00:03	0:00:00	0:00:12	0:00:08	0:00:11	0:00:23	0:00:01	0:00:32
	BE	0:00:32	0:00:00	0:00:14	0:00:21	0:02:13	0:02:07	0:00:03	0:01:27
GTZAN	All	0:00:07	0:00:00	0:00:01	0:00:01	0:00:02	0:00:03	0:00:00	0:00:15
	FS	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01
	BE	0:00:02	0:00:00	0:00:01	0:00:01	0:00:01	0:00:01	0:00:00	0:00:10

Tabulka E.2: **Celková doba běhu trénování a klasifikace na atributech extrahovaných pomocí knihovny Essentia s optimalizovanými parametry na testovací sadě**

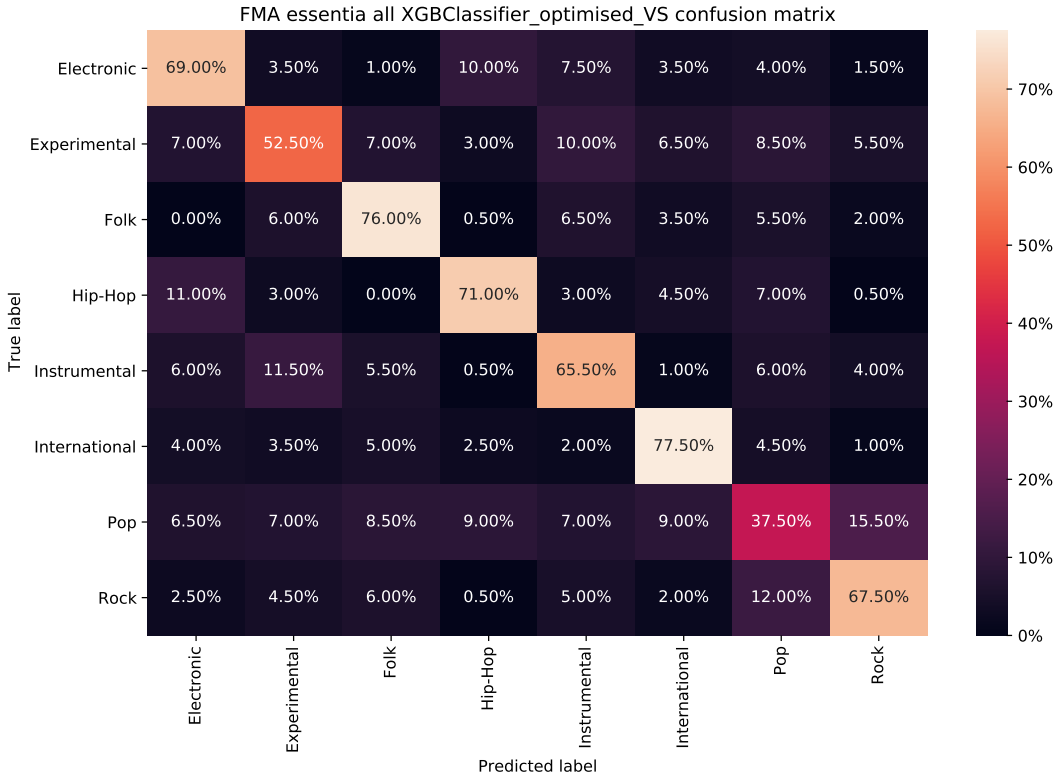
		LR	KNNC	MLPC	DTC	SVCL	SVCR	RFC	XGBC
EBD	All	0:00:04	0:00:02	0:00:45	0:00:18	0:00:34	0:00:35	0:01:05	0:02:07
	FS	0:00:00	0:00:00	0:00:02	0:00:00	0:00:01	0:00:03	0:00:00	0:00:09
	BE	0:00:11	0:00:01	0:00:40	0:00:06	0:00:33	0:00:12	0:01:27	0:01:58
FMA	All	0:00:06	0:00:10	0:00:53	0:00:26	0:02:41	0:02:59	0:05:11	0:02:55
	FS	0:00:03	0:00:00	0:00:06	0:00:04	0:00:11	0:00:23	0:00:34	0:02:01
	BE	0:00:11	0:00:10	0:00:45	0:00:26	0:02:20	0:02:07	0:06:21	0:03:49
GTZAN	All	0:00:02	0:00:00	0:00:08	0:00:01	0:00:02	0:00:03	0:00:14	0:00:18
	FS	0:00:00	0:00:00	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01
	BE	0:00:02	0:00:00	0:00:09	0:00:01	0:00:01	0:00:01	0:00:07	0:00:07

Tabulka E.3: **Tabulka s podrobnými informacemi o finální optimalizaci**

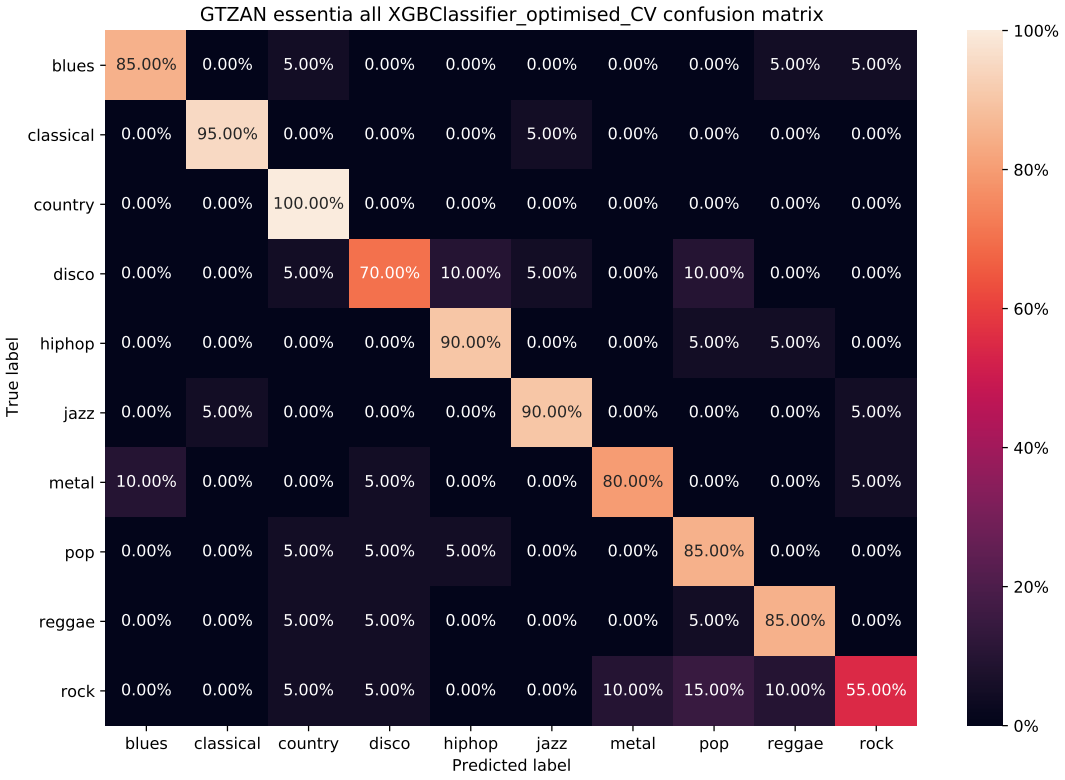
		MLPC	SVCL	XGBC
Doba běhu optimalizace parametrů	EBD	02:46:07	00:03:27	26:07:10
	FMA	07:32:58	00:15:15	29:48:28
	GTZAN	01:31:48	00:00:14	07:18:35
Úspěšnost klasifikace na testovací sadě	EBD	85.89%	84.57%	87.56%
	FMA	62.81%	60.94%	64.50%
	GTZAN	82.00%	82.00%	83.50%
Doba trénování a klasifikace	EBD	0:00:37	0:00:35	0:02:50
	FMA	0:00:44	0:02:42	0:06:50
	GTZAN	0:00:09	0:00:02	0:00:26



Obrázek E.1: **Matice predikcí nejúspěšnějšího klasifikátoru *XGBClassifier* na datové sadě EBD.**



Obrázek E.2: **Matice predikcí nejúspěšnějšího klasifikátoru *XGBClassifier* na datové sadě FMA.**



Obrázek E.3: **Matice predikcí nejúspěšnějšího klasifikátoru *XGBClassifier* na datové sadě GTZAN.**

Příloha F

Obsah přiloženého CD

- `/` ..... kořenový adresář přiloženého média
  - `project` ..... adresář obsahující soubory projektu
    - `data` ..... adrasář obsahující testovací skladby
      - `local_archive` ..... adrasář reprezentující lokální hudební archiv
        - ...
    - `metadata` ..... adresář obsahující vygenerovaná data
      - `misc` ..... adresář obsahující ostatní soubory
        - `columns_fix.json` . názvy sloupců atributů pro natrénované klasifikátory
        - `feature_extraction.out` ..... výpis průběhu extrakce atributů
        - `feature_selection.out` ..... výpis průběhu selekce atributů
        - `optimised_feature_sets.json` ..... vybrané podmnožiny atributů
        - `optimised_hyper_parameters.json` ..... optimalizované parametry
      - `optuna_studies` ..... adresář obsahující soubory k optimalizaci parametrů
        - `EBD_essentia_all_XGBClassifier_CV.pkl`
        - `FMA_essentia_all_XGBClassifier_VS.pkl`
        - `GTZAN_essentia_all_XGBClassifier_CV.pkl`
      - `test_data` ..... adresář obsahující testovací data
        - `EBD_genres.csv` ..... žánry testovacích dat EBD
        - `EBD_X_test.csv` ..... atributy testovacích dat EBD
        - `FMA_genres.csv` ..... žánry testovacích dat FMA
        - `FMA_X_test.csv` ..... atributy testovacích dat FMA
        - `GTZAN_genres.csv` ..... žánry testovacích dat GTZAN
        - `GTZAN_X_test.csv` ..... atributy testovacích dat GTZAN
      - `track_genre_lists` ..... adresář se soubory s informacemi o žánrech skladeb
        - `EBD_track_genre_list.csv`
        - `FMA_track_genre_list.csv`
        - `GTZAN_track_genre_list.csv`
      - `trained_classifiers` ..... adresář obsahující natrénované klasifikátory
        - `EBD_essentia_all_XGBClassifier_optimised_CV_TS.joblib.dat`
        - `FMA_essentia_all_XGBClassifier_optimised_VS_TS.joblib.dat`
        - `GTZAN_essentia_all_XGBClassifier_optimised_CV_TS.joblib.dat`
    - `src` ..... adresář obsahující zdrojové soubory projektu
      - `classification.ipynb` ..... trénování a klasifikace
      - `feature_extraction.ipynb` ..... extrakce atributů
      - `feature_selection.ipynb` ..... selekce atributů

- hpoptimise_optuna.ipynb ..... třída s funkcemi pro optimalizaci parametrů
- images.ipynb ..... generace obrázků do technické zprávy
- local_archive_predictions.ipynb ... predikce a anotace lokálního archivu
- misc.ipynb ..... funkce pro výpisy výsledků selekce/optimalizace/klasifikace
- parameter_tuning.ipynb ..... optimalizace parametrů
- track_genre_list_creation.ipynb tvorba souborů s informacemi o žánrech

xslade21-latex ..... adresář obsahující soubory pro generaci technické zprávy

- bib-styles ..... adresář obsahující styly pro latex
  - ...
- obrazky ..... adresář obsahující obrázky technické zprávy
  - ...
- template-fig ..... adresář obsahující loga fakulty
  - ...
- fitthesis.cls ..... latex třída pro technickou zprávu
- Makefile ..... makefile pro vytvoření technické zprávy
- xslade21-Klasifikace-hudebnich-souboru.tex
- xslade21-Klasifikace-hudebnich-souboru-01-kapitoly-chapters.tex
- xslade21-Klasifikace-hudebnich-souboru-20-literatura-bibliography.bib
- xslade21-Klasifikace-hudebnich-souboru-30-prilohy-appendices.tex
- zadani.pdf ..... zadání projektu

install.sh ..... skript pro instalaci potřebných knihoven

makefile ..... makefile pro instalaci Python knihoven (pro install.sh)

Návod-k-instalaci-a-použití.txt ..... Návod k instalaci a použití

requirements.txt ..... seznam potřebných knihoven (pro makefile)

Struktura-obsahu-média.pdf ..... soubor s tímto obsahem

xslade21-Klasifikace-hudebnich-souboru.pdf ..... soubor s technickou zprávou