
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV INFORMAČNÍCH SYSTÉMŮ**
**DEPARTMENT OF INFORMATION SYSTEMS**


# SYSTÉM PRO ANALÝZU BIATLONOVÝCH STATISTIK

**SYSTEM FOR ANALYSIS OF BIATHLON STATISTICS**

**BAKALÁŘSKÁ PRÁCE**
**BACHELOR'S THESIS**


**AUTOR PRÁCE** **ONDŘEJ ZEMAN**
**AUTHOR**

**VEDOUCÍ PRÁCE** **Ing. VLADIMÍR BARTÍK, Ph.D.**
**SUPERVISOR**



**BRNO 2022**

**Vysoké učení technické v Brně**
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2021/2022

# Zadání bakalářské práce


Student: **Zeman Ondřej**
Program: Informační technologie
Název: **Systém pro analýzu biatlonových statistik**
**System For Analysis of Biathlon Statistics**
Kategorie: Data mining

Zadání:

1. Prostudujte problematiku analýzy dat a získávání znalostí z dat.
2. Analyzujte různé možnosti a způsoby získávání dat z biatlonových serverů a navrhněte nejvhodnější postup. Navrhněte příslušnou databázi.
3. Navrhněte koncepci aplikace, která bude stahovat data biatlonových statistik a bude provádět jejich analýzu. Po konzultaci s vedoucím navrhněte vhodné analytické úlohy.
4. Navrženou aplikace implementujte, ověřte její funkčnost a proveďte experimenty.
5. Zhodnoťte dosažené výsledky a další možnosti pokračování tohoto projektu.

Literatura:


- Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Third Edition. Morgan Kaufmann Publishers, 2012, 703 p., ISBN 978-0-12-381479-1.
- Layton, R.: Learning Data Mining with Python: Use Python to manipulate data and build predictive models. 2nd Edition. Packt Publishing, 2017, 358 p., ISBN 978-1787126787.


Pro udělení zápočtu za první semestr je požadováno:

- Body 1-3.

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

Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 20. října 2021

# Abstrakt

Cílem této práce je vytvořit webovou aplikaci, která stáhne biatlonová data a statistiky a využije techniky pro získání znalostí z dat pro získání zajímavých a neobvyklých informací. V rámci práce jsou řešeny deskriptivní i prediktivní dolovací úlohy. Shlukovací algoritmy byly využity pro deskriptivní dolovací úlohy a pro nalezení vzorů v běžeckých a střeleckých statistikách. Predikce výsledků závodů je vyřešena pomocí vícenásobné lineární regrese. Aplikace je implementována v jazyce Python. Výsledná webová aplikace je dostupná na adrese https://analysisofbiathlonstatistics.herokuapp.com.



# Abstract

The goal of this bachelor's thesis is to create a web application that downloads biathlon data and statistics and use the technique to get knowledge from data to get interesting and unusual information. In this work are solving descriptive and predictive mining tasks. Clustering algorithms have been used for descriptive mining tasks and for search patterns in course and shooting statistics. Prediction of race results is solved by using multiple linear regression. The application is implemented in Python. Web application is available at https://analysisofbiathlonstatistics.herokuapp.com.



# Klíčová slova

Data mining, shlukování, predikce, kmeans, DBSCAN, vícenásobná lineární regrese, Python, Django, datová analýza, biatlon

# Keywords

Data mining, clustering, prediction, kmeans, DBSCAN, multiply linear regression, Python, Django, data analysis, biathlon


# Citace

ZEMAN, Ondřej. *Systém pro analýzu biatlonových statistik*. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

# Systém pro analýzu biatlonových statistik

## Prohlášení

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

.....
Ondřej Zeman
11. května 2022


## Poděkování

Děkuji panu Ing. Vladimíru Bartíkovi Ph.D., za ochotu při vytváření a přidělení tématu, za odborné a cenné rady, vstřícný přístup a věnovaný čas při řešení bakalářské práce. Také děkuji Veronice Bednářové za pomoc s jazykovou korekturou této práce.

# Obsah

**1 Úvod** **2**

**2 Data mining** **3**
2.1 Úvod do Data miningu . . . 3
2.2 Klasifikace . . . 5
2.3 Predikce . . . 9
2.4 Shluková analýza . . . 10

**3 Získávání dat** **15**
3.1 Návrh databáze . . . 15
3.2 Stahování dat . . . 17

**4 Implementace** **22**
4.1 Jazyk Python . . . 22
4.2 Knihovny jazyka Python . . . 22
4.3 Django . . . 25
4.4 Ajax . . . 26
4.5 ChartJs . . . 26
4.6 Struktura aplikace . . . 26

**5 Shlukovací experimenty** **31**
5.1 Porovnání běžeckého času - disciplíny . . . 31
5.2 Vývoj běžeckého času v průběhu závodu . . . 35
5.3 Srovnání rychlosti střelby a úspěšnosti střelby . . . 39
5.4 Úspěšnost výstřelů při střelbě . . . 41
5.5 Vyhodnocení shlukovacích experimentů . . . 43

**6 Prediktivní analýza** **45**
6.1 Výběr dat . . . 45
6.2 Vytvoření modelu . . . 45
6.3 Predikce výsledků . . . 46
6.4 Úprava predikce . . . 46
6.5 Zhodnocení modelu . . . 46

**7 Závěr** **48**

**Literatura** **50**

**A Obsah přiloženého média** **53**

# Kapitola 1

# Úvod

V posledních letech roste důraz na zpracování dat a získávání znalostí z dat ve všech odvětvích. Důraz na zpracování dat klade i sport a většina jeho disciplín. Zpracování dat je každým rokem častější ve více sportech a prohlubuje se do větších detailů. Při analýze dat ve sportu můžeme získat znalosti, kterých si nevšimneme pouhým pozorováním. Data pomáhají zjistit, který sportovec předvedl dobrý výkon, jaké aspekty výkonu jsou důležité pro vítězství, nebo mohou predikovat výsledky dalších závodů a zápasů.

Tato bakalářská práce se zabývá zpracováním biatlonových dat a získávaním znalostí z nich. Z biatlonových závodů můžeme získat velké množství zajímavých a typově odlišných dat. Cílem této práce je prostudovat metody pro zpracování dat a vhodné metody využít k predikci biatlonových výsledků a k vytvoření shluků, které nám pomohou interpretovat biatlonové statistiky. Tato práce může být užitečná pro všechny fanoušky biatlonu, kteří hledají zajímavé doplnění znalostí pro sledování závodů. Užitečná je také pro samotné biatlonisty, kteří na základě této práce mohou lépe srovnávat své výkony s ostatními biatlonisty a najít slabé nebo silné stránky jejich výkonů.

# Kapitola 2

# Data mining

Tato kapitola se zabývá základy Data miningu. Nejprve jsou v sekci 2.1 popsány základy Data miningu. Další sekce 2.2 a 2.3 popisují klasifikaci a predikci dat a metody, které byly použity při implementaci. Poslední sekce 2.4 se zabývá shlukováním dat a vlastnostmi jednotlivých shlukovacích metod. V této sekci jsou popsány také shlukovací metody použité při implementaci. Tato kapitola byla napsána podle zdrojů [20] a [24].

## 2.1 Úvod do Data miningu

Data mining neboli získávání znalostí z dat je proces, ve kterém objevujeme zajímavé a netriviální informace z velkého množství dat. Informace a modely, které získáváme pomocí dolování dat, by neměly být snadno získatelné například pomocí SQL dotazů na data v databázi a zároveň by se nemělo jednat o modely, které jsou snadno viditelné a zjistitelné. Při dolování dat naopak využíváme pokročilé techniky, které nám pomohou získat znalosti, které nejsme schopni snadno objevit. Potenciální užitečnost získaných znalostí je měřena významem pro rozhodnutí, které děláme na základě těchto znalostí. Proces získávání znalostí z dat je běžně tvořen z těchto částí:

- Čištění dat
  Při čištění dat je cílem odstranění odlehlých hodnot, chybějících dat a vyřešiní nekonzistence dat. Při čištění dat máme několik možností jak zacházet s chybnými daty. Jednou z možností je ignorování n-tice, ale tím se zmenšuje množina dat pro řešení dolovací úlohy. Častěji chybnou hodnotu nahradíme průměrnou hodnotou atributu, což nám umožní zachovat n-tici dat a zároveň neovlivnit výsledky dolovací úlohy. Další možností je manuální nahrazení, při kterém musíme znát informace o datech, nebo nahrazení globální konstantou, které může zkreslit výsledky.

- Integrace dat
  Integraci dat provádíme, pokud data pocházejí z různých zdrojů. Data z různých zdrojů musíme správně spojit, aby nedošlo k nekonzistenci.

- Selekce dat
  Při selekci dat vybereme jen potřebná data pro řešení dolovacích úloh. Pro řešení dolovací úlohy je důležité vybrat pouze podstatná data, aby nedocházelo k prodloužení času řešení úlohy. Redukování dat by mělo zachovat integritu a charakter původních dat a nemělo by ovlivnit výsledek dolování.

- Transformace dat
  Transformace dat je proces převedení dat do vhodného tvaru pro řešení dolovací úlohy. Může se jednat o normalizaci některých dat nebo převedení dat na stejný formát. Nekteré údaje mohou být agregovány. Při transformaci dat můžeme vytvořit nové atributy, které jsou odvozené od současných atributů. Nové atributy slouží k jednoduššímu řešení dolovací úlohy.

- Dolování dat
  Nejdůležitější část procesu získávání znalostí je vytvoření modelů dat nebo získání zajímavých vzorů dat.

- Hodnocení modelů a vzorů
  Při dolování můžeme získat spoustu modelů a vzorů dat. Modely a vzory dat je potřeba ohodnotit a odlišit, které modely a vzory jsou užitečné pro řešení dolovací úlohy a které nikoliv.

- Prezentace znalostí
  Cílem je prezentovat výsledky dolování uživateli využitím technik vizualizace a reprezentace znalostí.

Dolovací úlohy můžeme rozdělit na dva základní typy.

- Deskriptivní
  Cílem deskriptivních dolovacích úloh je charakterizovat obecné vlastnosti analyzovaných dat. Tato práce řeší několik deskriptivních dolovacích úloh. Deskriptivní dolovací úlohy byly řešeny napríklad pomocí shlukování 2.4. Další možné řešení deskriptvních úloh je pomocí asociačních pravidel a frekventovaných množin, ale tyto metody v této bakalářské práci nejsou použity.

- Prediktivní
  Na základě analýzy současných dat dochazí k předpovědi budoucího chování. V rámci bakalářské práce byly predikovány výsledky biatlonových závodů. K predikci výsledků můžeme využít metody pospané v sekci 2.2 a 2.3.

V rámci práce jsem řešil oba typy úloh. Jejich konkrétní popis je v kapitolách 5 a ref 6.

### 2.1.1 Sumarizující popisné charakteristiky dat

Pro úspěšné předzpracovní dat potřebujeme získat informace o datech, které budeme využívat při řešení dolovací úlohy. Pro získání těchto informací používáme sumarizující popisné charakteristiky dat. Pro získání popisné charakteristky dat pro datový soubor používáme následující popisné míry.

#### Míry polohy

Míra polohy určuje hodnotu, která zaujímá významnou polohu v souboru dat. Hodnota míry pro soubor dat může, ale nemusí existovat. Mezi míry polohy patří například aritmetický průměr nebo medián.

#### Míra variace

Míra variace určuje hodnotu, která vypovída o rozptylu hodnot v datovém souboru. Mezi míry variace může patřit rozsah hodnot, rozdíl mezi kvartily, rozptyl nebo směrodatná odchylka.

### 2.1.2 Typy atributů

Při řešení dolovacích úloh pracujeme s různými typy atributů. S každým z atributů pracujeme při řešení dolovací úlohy rozdílně. Typy atributů určují, jaké algoritmy a modely můžeme pro řešení úloh využít. Z pohledu oboru hodnot rozlišujeme tyto atributy kvantitativní, ordinální, kategorické a binární.

#### Kvantitavní atributy

Tyto atributy je reprezentován reálnými numerickými hodnotami. Příkladem může být výška.

#### Ordinální atributy

Jedná se o atributy, kterým jsou přiřazeny hodnoty a záleží na jejich pořadí. Příkladem může být známkování ve škole.

#### Kategorické atributy

Tyto atributy jsou spojeny se jmény. Hodnota kategorického atributu je symbolem nebo jménem pro věc. Každá hodnota reprezentuje druh kategorie. Na rozdíl od ordinálních atributů zde nezáleží na jejich pořadí. Mezi kategorické ordinální atributy patří například barvy.

#### Binární atributy

Binární atributy jsou kategorické atributy pouze se dvěma hodnotami, kde 0 znamená nepravda a 1 reprezentuje pravdu.

## 2.2 Klasifikace

Klasifikace je proces, který umožňuje přiřazovat data na základě jejich vlastností do jistých tříd, kterých je konečný počet. Klasifikace má tři části. První částí je fáze učení, druhou fází je testování klasifikátoru a třetí fází je jeho využití.

#### Fáze učení

Ze souboru dat je vybrána množina dat, která se nazývá trénovací množinou. U těchto dat musíme znát do jaké třídy jsou zařazeny. Tyto vzorky dat jsou vstupem pro klasifikátor. Klasifikátor má za úkol určit klasifikační pravidla, podle kterých můžeme jednotlivé objekty klasifikovat do jistých tříd.

#### Testování klasifikátoru

Jsou vybrány vzorky dat, u kterých je předem známa klasifikační třída. Tyto vzorky nazveme testovací množinou. Vzorky musí být plně nezávislé na datech z fáze učení. Data z trénovací množiny jsou pomocí klasifikačních pravidel zařazeny do tříd. Vyhodnotíme, v kolika procentech byla správně určena klasifikační třída

#### Využití klasifikátoru

Naučený model je využíván pro nové vstupy a generuje výstupy podle klasifikačních pravidel.

### 2.2.1 Porovnání klasifikačních metod

Kritéria pro porovnávání klasifikační metod mohou být různá. Různé klasifikační metody mají různé vlastnosti. Mezi základní vlastnosti klasifikačních metod řadíme následující:

- Přesnost předpovědi
  Vyhodnocuje v kolika procentech řadí klasifikátor vstupní objekt do správné třídy.
- Rychlost
  Výpočetní složitost pro vygenerování a používání klasifikačních pravidel.
- Robustnost
  Robustnost je schopnost vytvořit správný model, pokud data obsahují šum a chybějící hodnoty.
- Stabilita
  Jedná se o schopnost vytvořit model pro velké množství dat.
- Interpretovatelnost
  Určuje složitost modelu pro jeho pochopení.

### 2.2.2 Klasifikační metody

Pro klasifikaci můžeme využít různé klasifikační metody. Každá z metod má lepší využití pro různé typy úloh a má své výhody a nevýhody. V této sekci jsou popsány základní a zřejmě nejpoužívanější metody.

#### Rozhodovací stromy

Rozhodovací strom je strom grafové struktury, ve kterém každý vnitřní uzel reprezentuje test vnitřní hodnoty atributu a koncové listy reprezentují třídu, do které je objekt klasifikován. Rozhodovací strom může být snadno převeden na klasifikační pravidla, která rozhodnou, do jaké třídy objekt patří. Klasifikace probíhá tak, že vstupní objekt je postupně testován podle rozhodovacího stromu. Objekt postupně prochází celým stromem, dokud nedojde ke koncovému listu, který reprezentuje jeho třídu.

Pro rozhodovací stromy je důležité pořadí testování atributů. Pro maximálně efektivní a přesný rozhodovací strom potřebujeme testovat atributy v pořadí podle nejvyšší rozhodovací schopností. Pro výpočet atributu s nejvyšší rozhodovací schopností si zavedeme následující označení:
$S$ je množina všech vzorků (trénovací data)

$C_1, C_2, \ldots, C_m$ - značí jednotlivé třídy
$s_i$ je počet prvků z množiny S, který je klasifikován do třídy $C_i$
$m$ - je celkový počet tříd
$A_1, A_2, \ldots, A_n$ - značí jednotlivé atributy
$a_1, a_2, \ldots, a_v$ - značí jednotlivé hodnoty daného atributu
$v$ - je celkový počet atributů
$S_j$ - značí množinu obsahující pouze vzorky z množiny $S$, jejichž atribut $A$ má hodnotu $a_j$
$s_{ij}$ - značí počet vzorků z množiny $S_j$, které jsou klasifikovány do třídy $C_i$
Pro určení atributu s největší rozhodovací schopností nejprve spočítáme očekávanou informaci potřebnou pro klasifikaci daného vzorku:

$$I(s_1, \ldots, s_m) = -\sum_{i=1}^{m} p_i \log_2 p_i \tag{2.1}$$

$p_i$ je pravděpodobnost, že náhodně vybraný vzorek z množiny $S$ je klasifikován do třídy $C_i$. Pro každý atribut dále definujeme jeho entropii:

$$E(A) = \sum_{j=1}^{v} \frac{s_{1j} + \ldots + s_{mj}}{\|S\|} I(s_{1j} + \ldots + s_{mj}) \tag{2.2}$$

$I(s_{1j} + \ldots + s_{mj})$ je vypočítáno takto:

$$I(s_{1j} + \ldots + s_{mj}) = -\sum_{i=1}^{m} p_{ij} \log_2 p_{ij} \tag{2.3}$$

$p_{ij}$ je pravděpodobnost, že náhodně vybraný vzorek z množiny $S_j$ je klasifikován do třídy $C_i$. Pro každý atribut následně spočítáme hodnotu gain:

$$Gain(A) = I(s_1, \ldots, s_m) - E(A) \tag{2.4}$$

Atribut, který má největší *Gain*, má největší rozhodovací schopnost a ve stromu bude prvním uzelem. Ze zbylých atributů je vypočítaný atribut, který má nejvyšší rozhodovací schopnost a ten je v rozhodovacím stromu další v pořadí.

Rozhodovací stromy jsou nejčastěji používanou klasifikační metodou. Jedná se o metodu jednoduchou na implementaci, rychlou a vysoce přesnou. Rozhodovací stromy umějí pracovat pouze s diskrétními hodnotami, což je největší nevýhodou této metody.

### Jednoduchá bayesovská klasifikace

Bayesovská klasifikace je založená na statistice. Pro každý objekt je podle statistických metod určeno do jaké třídy bude daný objekt patřit. Objekt je následně zařazen do třídy, do které patří s největší pravděpodobností.

Pro určení třídy se používá podmíněná pravděpodobnost neboli Bayesův vzorec. Nechť $X$ a $Y$ jsou dva nezávislé jevy. Pro výpočet podmíněné pravděpodobnosti využijeme vzorec:

$$P(X|Y) = \frac{P(X \cap Y)}{P(Y)} \tag{2.5}$$

$P(X \cap Y)$ je pravděpodobnost, že jevy $X$ a $Y$ nastanou současně. $P(Y)$ je pravděpodobnost, že nastane jev Y.

Pro výpočet pravděpodobnosti pro jednoduchou bayesovskou klasifikaci definujeme následující parametry:
$S$ je množina všech vzorků - trénovací data.
$C_1, C_2, \ldots, C_m$ - značí jednotlivé třídy
$s_i$ je počet prvků z množiny S, který je klasifikován do třídy $C_i$
$m$ - je počet tříd
$A_1, A_2, \ldots, A_n$ - značí jednotlivé atributy
$X = (x_1, x_2, \ldots, x_n)$, kde $x_i$ je hodnota atributu $A_i$, značí testovací vzorek, který má být klasikován do nějaké třídy.

Podle Bayesova vzorce platí:

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)} \tag{2.6}$$

$P(C_i|X)$ udává pravděpodobnost toho, že vzorek $X$ bude zařazen do třídy $C_i$. Cílem je najít třídu, pro kterou je vypočítaná pravděpodobnost nejvyšší, a vzorek bude do této třídy zařazen.

Bayesovská klasifikace potřebuje, aby jevy byly nezávislé. Ve skutečném světě je většina jevů na sobě závislých, proto je občas označována jako bayesovská klasifikace jako naivní a její výsledky mohou být poměrně nepřesné.

### Neuronová síť Backpropagation

Ke klasifikaci pomocí Backpropagation využíváme neuronovou síť. Neuronová síť je skupina jednotek, které mají propojený vstup a výstup. Těmto jednotkám říkáme neurony. Každé propojení mezi dvěma neurony má váhu. Během fáze učení nastavujeme pro neuronovou síť váhy jednotlivých spojení mezi neurony, tak aby neuronová síť dosáhla očekáváného chování. Vytvoření neuronové sítě je početně náročné a interpretace neuronové sítě je oproti jiným klasifikačním metodám složitá. Výhodou neuronové sítě je naopak přesnost její předpovědi. Neuronová síť dokáže zpracovat i nečistá data a najít vzory, pro která nebyla trénována. Na rozdíl od jiných klasifikačních metod umí kvalitně zpracovat i kvantitavní atributy.

Při metodě backpropagation se nejčastěji využívá vícevrstvá neuronová síť. Vícevrstvá neuronová síť se skládá ze vstupní vrstvy, z jedné nebo více skrtých vrstev a z výstupní vrstvy. Každé vrstva je tvořena neurony. Vstupní vrstva odpovídá atributům pro každý testovací vstup. Výstupy neuronů jsou násobeny váhovou funkcí a vstupují do neuronů první skryté vrstvy. Výstupy neuronů ze skryté vrstvy jsou mapovány na vstupy dalších neuronů ve skrytých vrstvách. Výstupy z poslední skryté vrstvy jsou vstupem pro výstupní vrstvu. Výstup každého neuronu získáme tak, že všechny vstupní hodnoty vynásobíme váhovou funkcí pro daný vstup, hodnoty sečteme a přičteme bias neboli vnitřní konstantu neuronu. Výpočet provedeme pomocí následující rovnice:

$$\sum_{i=1}^{n} w_i x_i + \emptyset \tag{2.7}$$

Učení neuronové sítě pomocí metody Backpropagation je iterativní proces, ve kterém jsou výstupy pro tréninková data porovnány s výstupy trénované neuronové sítě. Na začatku trénovacího procesu jsou váhy jednotlivých funkcí nastaveny na malé hodnoty. Pro každý vstup jsou upraveny hodnoty váhových funkcí. Tato úprava probíhá ve zpětném směru přes všechny skryté vrstvy. Hodnoty jsou upraveny tak, aby výstupní hodnoty z neuronové sítě odpovídali očekáváným výstupním hodnotám z dat z trénovací množiny. Tento proces

se opakuje, dokud neumí neuronová síť správně reagovat na všechny vzorky z trénovací množiny.

## 2.3 Predikce

Predikce je proces, který umožní přiřazovat datům hodnoty, které mají obecně spojitý charakter. Predikce má tři fáze stejně jako u klasifikace 2.2. Nejprve je prediktivní model vytrénován, následně je otestován a nakonec dochází k jeho využití.

### 2.3.1 Lineární jednoduchá regrese

Data pro lineární regresi jsou ve tvaru $(x_1, y_1), (x_2, y_2), \ldots, (x_s, y_s)$, kde $x$ je vstupní atribut a $y$ je výstupní atribut. Data jsou aproximována přímkou o rovnici:

$$Y = aX + b \tag{2.8}$$

$Y$ je výsledná hodnota. Konstanty $a$ a b určíme pomocí metody nejmenších čtverců podle následujících vzorců:

$$a = \frac{\sum_{i=1}^{s}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{s}(x_i - \overline{x})^2} \tag{2.9}$$

$$b = \overline{y} - a\overline{x} \tag{2.10}$$

Nevýhodou lineární jednoduché regrese je, že její hodnota je určena na základě jednoho atributu. Výsledná hodnota většinou závisí na více argumentech a predikce pomocí jednoduché lineární regrese jsou často nepřesné.

### 2.3.2 Lineární vícenásobná regrese

U vícenásobné lineární regrese můžeme zobecnit metodu pro $v$ vstupních parametrů na rozdíl od jednouché lineární regrese. Data pro vícenásobnou lineární legresi jsou ve tvaru $(x_{11}, x_{12}, \ldots, x_{1v}, y_1)$, $(x_{21}, x_{22}, \ldots, x_{2v}, y_2), \ldots, (x_{s1}, x_{s2}, \ldots, x_{sv}, y_s)$ Aproximace je provedena pomocí rovnice:

$$Y = a_0 + a_1X_1 + a_2X_2 + \ldots + a_3X_3 \tag{2.11}$$

Pro zjištení výsledné hodnoty Y potřebujeme zjistit koeficienty $a_0, a_1, \ldots, a_v$. Z jednotlivých koeficientů vytvoříme matice:

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_v \end{pmatrix} \quad X = \begin{pmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1v} \\ 1 & x_{21} & x_{22} & \ldots & x_{2v} \\ \vdots & & & & \vdots \\ 1 & x_{s1} & x_{s2} & \ldots & x_{sv} \end{pmatrix} \quad A = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_v \end{pmatrix}$$

Matici A vypočítáme podle následujícího vzorce:

$$A = (X^TX)^{-1}X^TY \tag{2.12}$$

Lineární vícenásobná regrese se používá k predikci mnohem častěji než jednoduchá lineární regrese a také s větší úspěšností. Ve své práci pracuji na predikci biatlonových výsledků a k predikci jsem využil lineární vícenásobnou regresi. K využití vícenásobné lineární regrese mě inspiroval úspěšný hokejový prediktivní model Game Score, který je založený na

vicesobné lineární regresi. Tento model je predikuje hokejové výsledky pro Tipsport Extraligu ledního hokeje. Model dosahuje úspěšnosti 65 % a jedná se zřejmě o nejlepší systém pro predikci hokejových zápasů, který je veřejně dostupný [19].

## 2.4 Shluková analýza

Shluková analýza je proces rozdělení skupiny objektů do podmnožin. Každá podmnožina je shluk objektů, ve kterém si jsou objekty navzájem podobné a nejsou podobné objektům v jiných shlucích. Různé shlukovací algoritmy mohou rozdělit stejné objekty do různých shluků. Proces shlukování je založený na učení bez učitele. Data jsou rozdělena do shluků pouze podle algoritmů, nikoliv v závislosti na lidském faktoru nebo podle předem daných tříd, jak je tomu u klasifikace a predikce. Shlukování může vést k nalezení různých, předem neznámých skupin nebo vzorů, které mohou přinést nový pohled na zkoumaná data. Shluková analýza nám může také pomoci detekovat odlehlé hodnoty.

### 2.4.1 Vlastnosti shlukovacích metod

Algoritmy pro shlukovou analýzu jsou různorodé a pro řešení dolovací úlohy je potřeba vybrat správný shlukovací algoritmus. Typické vlastnosti shlukovacích metod, které potřebujeme pro řešení dolovacích úloh, jsou následující:

- Škálovatelnost:
  Některé shlukovací alogritmy mohou fungovat na malém množství dat. Často pracujeme s velkými databázemi, které obsahují tisíce objektů, proto musí být shlukovací algoritmy vysoce škálovatelné.

- Schopnost zpracování různých typů atributů:
  Pro zpracování objektů s numerickými atributy existuje spousta algoritmů. Při shlukové analýze pracujeme s objekty, které obsahují i jiné typy atributů a shlukové algoritmy musí umět pracovat i s těmito objekty.

- Vytváření shluků různého tvaru:
  Nejvíce shlukovacích algoritmů je založeno na Euklidovské nebo Manhattonovské vzdálenosti. Tyto algoritmy nejčastěji nacházejí shluky stejného kruhového tvaru, stejné velikosti a stejné hustoty. Shluky můžou být různého tvaru a může být potřebova vytvořit i shluky jiného tvaru než kruhového tvaru.

- Minimální požadavky na znalost problému při určování parametrů:
  Mnoho algoritmů potřebuje předem určit vstupní parametry. Jedním z parametrů může být například počet shluků. Tyto vstupní parametry mohou výrazně ovlivnit výsledky shlukování.

- Schopnost vyrovnat se s daty obsahujícími šum:
  Většina dat obsahuje odlehlé hodnoty. Některé hodnoty mohou chybět nebo mohou být chybné. Tyto chybné hodnoty mohou ovlivnit výstup shlukové analýzy, proto potřebujeme algoritmy, které se dokaží vyrovnat s těmito problémy.

- Necitlivost na pořadí vstupních záznamů:
  Shlukovací algoritmy mohou být závislé na pořadí jednotlivých objektů. Při změně pořadí objektů můžeme dostat velmi rozdílné výsledky shlukové analýzy.

- Schopnost zpracovávat vysokodimenzionální data:
  Objekty pro rozdělení do tříd mohou obsahovat i objekty s velkým počtem atributů. Pro shlukovou analýzu potřebujeme algoritmy, které zvládnou zpracovat i vysokodimenzionální data. Právě u vysokodimenziálních objektů nejsme schopni data rozřadit bez použití výpočetní techniky a potřebujeme využít shlukovací algoritmy.
- Schopnost shlukování na základě omezování:
  Výsledné shluky mohou obsahovat různá omezení, například minimální nebo maximální počet objektů v jedné podmnožině. Omezení se mohou vzrahovat i na různé atributy. Cílem je nalézt takové rozdělení do tříd, aby splňovalo předem daná omezení.
- Interpretovatelné a použitelné shluky:
  Výsledkem shlukové analýzy musí být interpretovatelné a použitelné shluky. Je důležité, aby výsledné shluky umožnily aplikaci cíle shlukové analýzy.

Samozřejmě nepotřebujeme pro řešení dolovací úlohy, aby algoritmus, pomocí kterého shlukování vytváříme, měl všechny tyto vlastnosti. Vždy musíme vybrat algoritmus, který splňuje vlastnosti, které potřebujeme pro konktérní dolovací úlohu.

### 2.4.2 Rozdělení shlukovacích metod

#### Metody založené na rozdělování

Metody založené na rozdělování rozdělují $n$ objektů do $k$ tříd, přičemž musí platit $k \leq n$. Objekt musí patřit právě do jedné třídy a v každé třídě musí být alespoň jeden objekt. U těchto metod musíme vždy dopředu zadat počet tříd $k$. V prvním kroku vybereme $k$ objektů, které reprezentují jednotlivé třídy, a ostatní objekty se na základě vzdalenostní funkce rozdělí do jednotlivých tříd. V dalších krocích se iterativně hledají objekty, které nejlépe reprezentují dané třídy. Konečný výsledkem je, pokud nedošlo k přesunu, žádného objektu z jedné třídy do třídy jiné. Tyto metody nejsou schopny nalézt shluky složitějšího tvaru a další nevýhodou je určení počtu tříd $k$. Mezi nejznámější a nejpoužívanější heuristiky patří k-means neboli metoda založená na centrálním bodě a k-medoids neboli metoda založená na reprezentujícím objektu.

#### Hiearchické metody

Hiearchické metody vytvářejí hiearchický rozpad předem dané množiny objektů. Hiearchické metody rozdělujeme na shlukující a rozdělující podle toho, jak probíhá rozklad.

- Shlukující hiearchické metody:
  Rozklad probíhá od zdola nahoru. Každý objekt se umístí do jedné zvláštní třídy. Postupně se slučují nejpodobnější třídy, dokud nedosáhneme požadovaného rozdělení.
- Rozdělující hierarchické metody:
  Rozklad probíhá od shora dolů. Nejprve jsou umístěny všechny objekty do jedné třídy. Následně se jednotlivé třídy rozdělují na menší třídy, dokud nedosáhneme předem daného počtu tříd.

Problémem těchto metod je, že pokud třídy spojíme, tak je již nemůže rozdělit, nebo naopak pokud třídy rozdělíme, nemůžeme je znovu spojit. Výpočetní složitost hiearchických metod je nižší než u metod založených na rozdělování. Naopak hiearchických metode nemusí být dostatečně přesné, protože nejsme schopni rozdělovat a spojovat třídy poté, co jsou jednou spojeny.

#### Metody založené na hustotě

Nejvíce shlukovacích metod je založeno na vzdálenosti mezi objekty. Tyto shlukovací metody nalézají shluky kruhovitého tvaru. Shlukovací metody založené na hustotě vytváří výsledné shluky různých tvarů, což je jejich největší výhodou. Shlukovací algoritmy jsou založené na shlukování objektů v prostoru dat. Tyto metody jsou založeny na myšlence, že jednotlivé třídy rostou do té doby, dokud má některý objekt ze třídy v prostoru do prahové vzdálenosti jiný objekt. Tento objekt se poté připojí k dané třídě a pokračuje hledání objektů v prostoru, které jsou v menší vzdálenosti, než je prahová vzdálenost. Objekty, které jsou odlehlé v prostoru dat, se považují za šum.

### 2.4.3 Shlukovací metody

#### k-Means

Metoda k-Means patří mezi metody založené na rozdělování. Pro metodu k-Means předpokládáme soubor dat $D$, který obsahuje $n$ objektů. Metoda k-means rozdělí $n$ objektů do $k$ tříd $C_1, \ldots, C_k$, kde platí $C_i \subset D, C_i \cap C_j = \emptyset$ pro $0 <= i, j <= k$.

Tato metoda reprezentuje třídu pomocí fiktivního centrálního bodu, jehož atributy jsou určeny jako střední hodnoty atributů objektů, které byly přiřazeny do dané třídy. Centrální bod může být definován různými způsoby. Nejčastěji je definován jako průměr hodnot atributů objektů, které jsou zařazeny do dané třídy. Objekty jsou na základě vzdáleností od těchto fiktivních centrálních bodů. Objekty jsou rozdělovaný, tak, aby si objekty v jedné třídě byly, co nejvíce podovné, a naopak, aby nebyly podobné objektům v jiných třídách. Cílem je vytvořit třídy, které jsou co nejvíce kompaktní, a zároveň jsou co nejvíce oddělitelné od sebe.

Vzdálenost objektů $p \in C_i$ a centrálního bodu $c_i$ je měřena jako $dist(p, c_i)$, kde $dist(x, y)$ je Euklidovská vzdálenost mezi body $x$ a $y$. Kvalita jednotlivých tříd $C_i$ je měřena pomocí shlukové variace. Shluková variace je součet středních kvadratických chyb mezi všemi objekty ve třídě $C_i$ a centrálním bodem $c_i$. Součet středních kvadratických chyb je definována jako:

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} dist(p, c_i)^2 \tag{2.13}$$

Pro každý objekt v dané třídě je spočítaná vzdálenost mezi objektem a centrálním bodem a všechny tyto vzdálenosti jsou sečteny. Metodu k-means je možné pomocí střední kvadratické chyby optimalizovat, ale je to obtížné a může docházet k vysoké časové složitosti.

K-means nejprve vybere náhodně $k$ objetů, přičemž každý z objektů reprezentuje centrální bod dané třídy. Zbývající objekty jsou přiřazeny do třídy, kde je jejich Euklidovská vzdálenost od centrálního bodu nejmenší. Pro každou třídu je spočítaný nový centrální bod. Následně jsou všechny objekty opět přiřazeny do třídy, ve které je vzdálenost mezi objekt a centrálním bodem třídy nejmenší. Tento postup se opakuje, dokud při přiřazení alespoň jeden z objektů změní třídu.

**Algoritmus 1:** K-MEANS

**Input:** $D$: soubor dat, obsahuje $n$ objektů
$k$: počet tříd
**Output:** $k$ shluků

1: Náhodné vybrání $k$ objektů a přiřazení do tříd
2: **while** *došlo k přesunu objektu mezi třídami* **do**
3:  **for** *pro každý objekt* **do**
4:  přiřaď objekt do třídy, která je nejvíce podobná na základě vzdálenosti mezi objektem a středem třídy
5:  **for** *pro každou třídu* **do**
6:  spočítej nový centrální bod.

### DBSCAN

Density-Based Clustering Based on Connected Regions with High Density neboli DBSCAN patří mezi metody, které rozdělují objekty do tříd, podle hustoty objektů v prostoru. Hustota objektu $o$ je měřena jako počet objektů v blízkém prostoru objektu $o$. DBSCAN hledá středové objekty. Středové objekty jsou objekty, které mají ve svém okolí vysokou hustotu sousedních objektů. Objekt je pro jiný objekt sousedním objektem, pokud jeho vzdálenost od sousedního objektu je menší než $\epsilon$. $\epsilon$ je parametr větší než 0, který zadá uživatel a specifikuje poloměr, který určuje do jaké vzdálenosti jsou objekty sousedy. Uživatel kromě parametru $\epsilon$ zadává také parametr $MinPts$, který určuje minimální hodnotu hustoty objektů. Objekt je středovým objektem, pokud v jeho okolí ve vzdálenosti $\epsilon$ je hustota nejméně $MinPts$. Algoritmus DBSCAN pracuje tak, že ze vstupního souboru dat s $n$ objekty vybere náhodně nenavštívený objekt. Objekt je po vybrání označený jako navštívený. Pokud má v okolí $\epsilon$ alespoň $MinPts$ objektů, tak je vytvořena nová třída $C$ a objekt je přidán do třídy $C$. Naopak je objekt označený jako šum. Všechny objekty z okolí vybraného objektu jsou označeny jako kandidátní objekty a jsou označeny za navštívené. Pokud mají v okolí $\epsilon$ alespoň $MinPts$ objektů, tak jsou objekty přidané do třídy $C$ a objekty v jejich okolí jsou označeny jako kandidátní objekty. Pokud neexistují žadné další kandidátní objekty, je náhodně vybrán nový nenavštívený objekt

**Algoritmus 2:** DBSCAN

**Input:** $D$: soubor dat, obsahuje $n$ objektů
$\epsilon$: poloměr, který určuje blízké okolí
$MinPts$: minimální hustota objektů pro určení jádra
**Output:** Vytvořené shluky podle hustoty

```
1:  označ všechny objekty jako nenavštivené
2:  while nejsou všechny objekty navštívené do
3:      náhodně vyber nenavštívený objekt p
4:      označ objekt p jako navštívený
5:      if pokud v okolí ϵ objektu p je alespoň MinPts objektů then
6:          vytvoř nový shluk C a přidej do shluku C objekt p
7:          objekty z okolí objektu p přidej do množiny N
8:          for pro každý objekt p′ v množině N do
9:              if objekt p′ je nenavštívený then
10:                 označ p′ jako navštívený
11:                 if pokud má objekt p′ v okolí ϵ alespoň MinPts objektů then
12:                     objekty z okolí objektu p′ přidej do množiny N
13:                 if pokud objekt p′ není v žádném shluku then
14:                     přidej objekt p′ do shluku C
```

# Kapitola 3

# Získávání dat

Pro řešení dolovacích úloh bylo potřeba získat data o biatlonových závodech, výsledky ze závodů, rozšířené statistiky a v neposlední řadě bylo potřeba získat data o samotných biatlonistech a biatlonistkách.

Techniky pro získávání dat, které byly použity jsou popsány v sekci 3.2. Data, která byla stažena a zpracována, jsou uložena v databázi, která je popsána v sekci 3.1. Při řešení dolovacích úloh se již pracuje pouze s vyčištěnými daty v databázi.

## 3.1 Návrh databáze

Po stažení dat jsou všechna data uložena do databáze. Datábaze je implementována pomocí technologie `mysql`. Jedná se o databazí, která využívá relační model. Databáze byla navržena podle potřeb pro řešení dolovacích úloh. Při návrhu databáze byl nejprve navržen ER diagram, následně byl ER diagram převeden na tabulky relační databáze a tyto tabulky byly normalizované. Tabulky byly na závěr vytvořeny v databázi. Všechna stažená data byla uložena do databáze.

### 3.1.1 Relační model dat

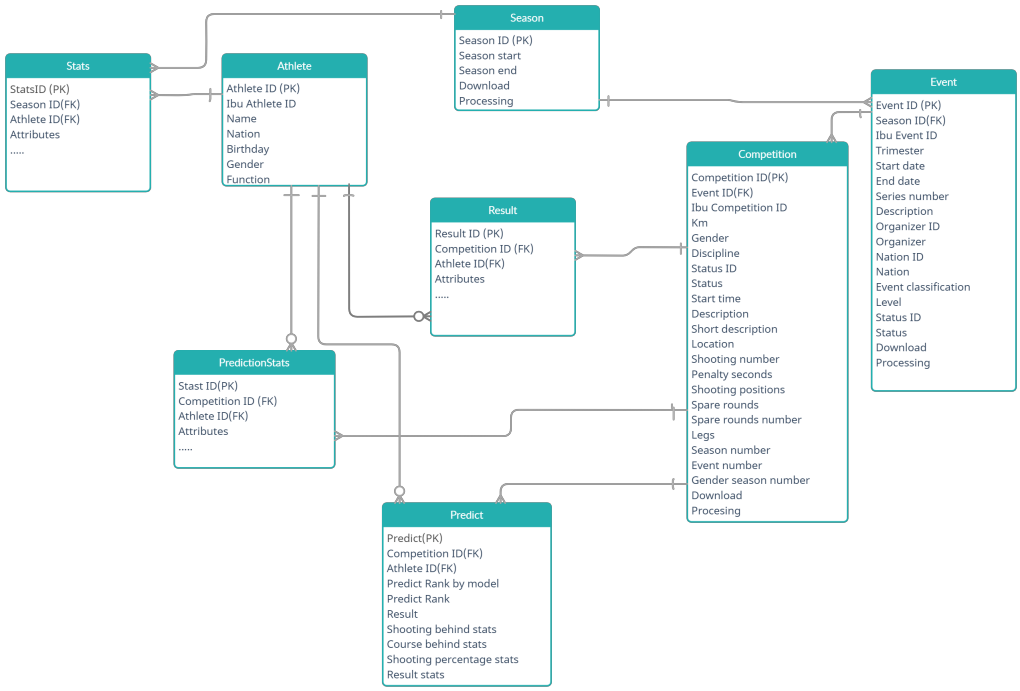
Relační model dat na logické úrovni je kolekcí relací. Relační databáze jsou inspirovány relací v matematice. Relace $R$ je v matematice definována mezi množinami $A_1, A_2, \ldots, A_n$ jako $R \subseteq A_1 \times A_2 \times \ldots \times A_n = \{(a_1, a_2, \ldots, a_n) | a_1 \in A_1, a_2 \in A_2, \ldots, a_n \in A_n\}$. Definice relace v relačním modelu na doménách $D_1, D_2, \ldots, D_n$ je dvojice $R = (R, R^*)$ , kde $R = (A_1 : D_1, A_2 : D_2, \ldots, A_n : D_n)$ je schéma relace a $R^* \subseteq D_1 \times D_2 \times \ldots \times D_n$ je tělo relace. Schéma relace zapisujeme často zjednodušeně ve tvaru $R(A_1, A_2, \ldots, A_n)$. Počet atributů $n$ relace se označuje jako stupeň relace, kardinalita těla relace $m = |R^*|$ se označuje jako kardinalita relace. Doména je množina skalárních hodnot stejného typu. Skalární hodnota je nejmenší sémantická jednotka dat.

Mezi důležité vlastnosti relačních databází patří neexistence duplicitní n-tice v realašním modelu. N-tice jsou vzájmně neuspořádané. Hodnoty jednotlivých atributů jsou v relačních modelech atomické, což znamená, že relace jsou v 1. normální formě [25].

### 3.1.2 ER Diagram

Entity Relationship Diagram je diagram, který slouží k návrhu databáze. ER diagram obsahuje entity a vztahy mezi entitami. Entita je objekt, který je jednoznačně odlišitelný od

jiných objektů. Entitní množina je množina entit stejného typu se stejnými vlastnostmi. Atributy entity jsou vlastnostmi daného objektu, o kterých potřebujeme uchovávat informaci. Atributy mohou být jednoduché a složené. Vztah je asociace mezi entitami. Každý vztah je definován pomocí jména vztahové množiny. Vztahová množina je množina vztahů stejného typu se stejnými vlastnostmi. Vztah má jednoznačně určen stupeň vztahu, členství a kardinalitu. Stupeň vyjadřuje mezi kolika entitami je vztah. Stupeň je typicky unární, binární nebo ternání. Členství určuje minimální počet vztahů daného typu, ve kterých musí entita participovat. Kardinalita je maximální počet vztahů daného typu, ve kterých musí entita participovat [1]. ER Diagram pro moji databázi obsahuje celkem 6 entit.




Obrázek 3.1: ER Diagram pro návrh databáze

- Season
  Tabulka Season má uložené infromace o jednotlivých sezónách.

- Event
  Event jsou biatlonové události neboli víkendy, které obsahují několik závodů a jsou organizovány na jednom místě. Důležitou informací je úroveň soutěží. Může se jednat o soutěže Světového, Evropského nebo Juniorského poháru. Při řešení dolovacích úloh je podstatné tyto úrovně odlišovat. V tabulce Event je uloženo místo konání závodů, datum konání nebo popis závodu.

- Competition
  Competition jsou biatlonové závody, které absolvují muži i ženy. Biatlonové závody jsou součástí biatlonových událostí.V tabulce Competition jsou uložené všechny informace o závodě například počet střelnic, délka závodu, typ disciplíny, zda závod jeli muži nebo ženy, kde se závod jel, jak dlouhý byl závod. Dále je v tabulce uloženo,

jestli se jednalo o závod, ve kterém byly za chybný výstřel uděleny trestné vteřiny, nebo se jelo trestné kolo, čas startu závodu nebo jeho popis.

- Athlete
  V tabulce Athlete jsou základní informace o biatlonistech a biatlonistkách. Každý biatlonista má svůj záznam se jménem, národností, pohlavím, datem narození a informací o tom, jestli je jeho kariéra aktivní nebo ne.

- Result
  V tabulce Result jsou uloženy výsledky ze závodů. Pro každý výsledek je v tabulce jeden záznam. Každý výsledek je spojen s konkrétním závodníkem z tabulky Athlete a závodem z tabulky Competition. Každý záznam obsahuje kromě výsledku všechny detailní statistiky, které se k závodníkovi vztahují. Mezi statistiky patří čas běhu i střelby, ztráta na nejlepšího nebo pořadí v dané statistice. V tabulce jsou dále uloženy informace o chybách na střelnících a o dalších statistikách.

- PredictStats
  Pro každého závodníka jsou po získání údajů o jeho výsledcích spočítány jeho statistiky jako čas běhu nebo přesnost střelby v různých obdobích. Ty jsou dále použity v prediktivní analýze.

- Stats
  Každý biatlonista, který se zúčastnil závodu v dané sezóně má uložené údaje v tabulce Stats. V tabulce Stats jsou uloženy statistiky pro biatlonistu, podle výsledků, kterých dosáhl v dané sezóně.

- Prediction
  V tabulce Prediction jsou uloženy informace o predikci výsledků. Každý predikovaný výsledek je spojený se závodníkem z tabulky Athlete a se závodem z tabulky Competition. V tabulce jsou uloženy také parametry podle kterých byl výsledek predikovaný.

## 3.2 Stahování dat

Biatlonové statistiky ze závodů jsou uloženy na několika webových stránkách. Oficiální výsledky a všechny informace o závodech a biatlonistech se nacházejí na webové stránce Mezinárodní biatlonové unie `www.biathlonresult.com`. Alternativou bylo získávat data například ze stránky `https://www.realbiathlon.com/`, ale tato stránka data pouze přebírá z oficiálního zdroje `www.biathlonresult.com` a není zaručená její dlouhodobá udržitelnost. Další možností byla stránka `https://www.flashscore.com/biathlon/`, kde ale nenalezneme všechny potřebné statistiky. Pro kvalitní výsledky práce byla potřebná důvěryhodná data. Z tohoto důvodu jsem se rozhodl pro získání dat z oficiálních zdrojů, i když data zde byla uložena různými způsoby a jejich získání bylo složitější než z jiných neoficiálních zdrojů. Nejkomplikovanější bylo získávání rozšířených statistik o výsledcích biatlonistů. Důležité statistiky pro tuto práci jako například bežecký nebo střelecký čas, nebylo možné získat přímo přes API jako jiná data. Tyto statistiky jsou na oficiální webové stránce uloženy pouze v PDF souborech. Další statistiky, které jsou měřeny pouze několik posledních let, jako například čas mezi výstřely při střelbě, jsou uloženy pouze v HTML a pro jejich získání bylo zapotřebí využití frameworku Selenium. Pro stahování dat z webové stránky

www.biathlonresult.com byl využit jazyk Python. Jazyk Python byl využit z důvodu velkého množství knihoven pro stahování dat. Knihovny, které byly využity, jsou popsány v sekci 4.2.

### 3.2.1 API

API neboli Application Programming Interface definuje standardizovanou syntaxi, která umožní jednomu softwarovému zařízení komunikovat s jiným, a to i když byla napsánya v jiném programovacím jazyce nebo jsou jinak strukturovaná. Webové API umožňuje především komunikaci mezi webovým serverem a internetovým prohlížečem. Dokumentace pro API typicky popisuje cesty a koncové body, jako url adresy s parametry, které lze získat pomocí metody GET s parametrem. Odpověd od webového serveru je obvykle zaslána ve formátu JSON nebo XML. Data, která byla získána, byla uložena pomocí formátu XML. Webová stránka www.biathlonresult.com umožnila získat data o biatlonistech a biatlonistkách, o biatlonových událostech a biatlonových závodech přes API ve formátu XML. Pro získání dat přes API byla využita knihovna request 4.2. [22]

#### Data uložená v XML

XML neboli Extensible Markup Language je značkovací jazyk. Jedná se o serializační formát, který má skupinu pravidel, jenž definuje sémantické tagy neboli elementy. Sémantické tagy popisují význam XML dokumentu [23]. XML dokumenty jsou obdobné jako HTML dokumenty. Jednotlivé XML elementy jsou hiearchicky zanořovány. Každý element má jméno a může mít atributy. Jazyk pro výběr uzlů v XML dokumentu se nazývá XPath. Výsledkem vyhodnocení XPath je množina elementů. Data z XML dokumentů jsem byla zpracována pomocí knihovny Beautifoulsoup4 4.2. Zpracování XML dokumentů bylo díky zavedeným standartům a praktičnostni knihovny Beautifoulsoup4 snadné. V XML souborech byla získána data o biatlonových závodech, o biatlonistech a reporty s výsledky z biatlonových závodů. [22]

Ukázky XML souborů, které byly staženy. Data o biatlonových událostech:

```
<ArrayOfSportEvent xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SportEvent>
<SeasonId>2021</SeasonId>
<Trimester>1</Trimester>
<EventId>BT2021SWRLCP01</EventId>
<StartDate>2020-11-27T12:00:00Z</StartDate>
<EndDate>2020-11-29T12:00:00Z</EndDate>
<FirstCompetitionDate>2020-11-28T10:00:00Z</FirstCompetitionDate>
<Description>BMW IBU World Cup Biathlon-Season Opening</Description>
<EventSeriesNr>1 </EventSeriesNr>
<ShortDescription>Kontiolahti</ShortDescription>
<OrganizerId>KON</OrganizerId>
<Organizer>Kontiolahti</Organizer>
<Nat>FIN</Nat>
<NatLong>Finland</NatLong>
<EventClassificationId>BTSWRLCP</EventClassificationId>
<Level>1</Level>
```

```
<UTCOffset>2</UTCOffset>
<IsActual>false</IsActual>
<IsCurrent>false</IsCurrent>
<EventStatusId>0</EventStatusId>
<EventStatus/>
</SportEvent>
</ArrayOfSportEvent>
```

Data o závodech:

```
<ArrayOfCompetition xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<Competition>
<RaceId>BT1920SWRLCP01MXSR</RaceId>
<catId>MX</catId>
<DisciplineId>SR</DisciplineId>
<StatusId>11</StatusId>
<StatusText>Final</StatusText>
<ScheduleStatus>FINISHED</ScheduleStatus>
<ResultStatus>OFFICIAL</ResultStatus>
<HasLiveData>false</HasLiveData>
<IsLive>false</IsLive>
<StartTime>2019-11-30T12:10:00Z</StartTime>
<Description>Single Mixed Relay (W+M)</Description>
<ShortDescription>Single Mixed Relay (W+M)</ShortDescription>
<Location>Swedish National Biathlon Arena</Location>
<HasAnalysis>true</HasAnalysis>
<StartMode>M</StartMode>
<NrShootings>2</NrShootings>
<NrSpareRounds>3</NrSpareRounds>
<HasSpareRounds>true</HasSpareRounds>
<PenaltySeconds>0</PenaltySeconds>
<NrLegs>4</NrLegs>
<ShootingPositions>PS</ShootingPositions>
<LocalUTCOffset>1</LocalUTCOffset>
<RSC>BTHXRELAY2------------FNL-000100--</RSC>
<GenderOrder>W+M</GenderOrder>
</Competition>
</ArrayOfCompetition>
```

URL s reportem ze závodu:

```
<ReportsReponse xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<RaceId>BT1920SWRLCP01MXSR</RaceId>
<Reports>
<ReportItem>
<Description>COMPETITION ANALYSIS</Description>
<URL>
https://ibu.blob.core.windows.net/docs/1920/BT/SWRL/CP01/MXSR/BT_C77C_1.0.pdf
</URL>
```

```
<ReportType>PDF</ReportType>
</ReportItem>
</Reports>
</ReportsReponse>
```

### Data uložená v PDF

Výsledky z jednotlivých závodů byly uloženy v souborech PDF. PDF soubory bylo možné získat pomocí API. Soubory byly po stažení uloženy na lokální úložiště. Ke zpracování PDF souborů byla využita knihovna pdfplumber. PDF soubor je pomocí knihovny pdfplumber načten řádek po řádku. Data jsou rozpoznávána a zpracovávána pomocí regulárních výrazů. Report s výsledky může mít několik různých formátů a v rámci zpracování reportu je potřeba rozpoznat, o jaký formát se jedná. Všechny formáty mají stejnou posloupnost uložení dat. Reporty mají v úvodu hlavičku s informacemi o závodě. Po hlavičce následují výsledky ze závodu. Výsledky jsou seřazeny podle pořadí. Každý výsledek začíná řádkem se jménem biatlonisty nebo biatlonistky. V dalších řádcích jsou informace o běžeckém čase, o střelbě a další statistiky. Tato posloupnost se opakuje v celém PDF souboru. Při zpracovávání bylo potřeba zachytit různé chyby. Nejčastějším problémem byli chybějící data, data ve špatném formátu a chybná data. Ukázka reportu je na obázku 3.2.

**COMPETITION ANALYSIS**

MEN 20KM INDIVIDUAL

SWEDISH NATIONAL BIATHLON ARENA
SAT 27 NOV 2021

START TIME: 15:00
END TIME: 16:51

| Rank | Bib | Name | Nat | Loop 1 | | Loop 2 | | Loop 3 | | Loop 4 | | Lap 5 | | T | Result | Behind | Rk |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Time | Rk | Time | Rk | Time | Rk | Time | Rk | Time | Rk | | | | |
| 1 | 34 | LAEGREID Sturla Holm | NOR | | | | | | | | | | | 0 | 51:04.0 | 0.0 | 1 |
| | | Cumulative Time | | 10:08.5 | +9.8 5 | 20:23.3 | 0.0 1 | 30:56.2 | 0.0 1 | 41:32.9 | 0.0 1 | | | | 51:04.0 | 0.0 | 1 |
| | | Loop Time | | 10:08.5 | +9.8 5 | 10:14.8 | +15.5 4 | 10:32.9 | +13.1 11 | 10:36.7 | +8.4 2 | 9:31.1 | +29.1 28 | | | | |
| | | Ski Time | | 10:08.5 | +18.2 13 | 20:23.3 | +36.6 12 | 30:56.2 | +53.2 14 | 41:32.9 | +1:18.8 14 | | | | 51:04.0 | +1:47.3 | 16 |
| | | Shooting | | 0 27.8 | +10.7 22 | 0 26.3 | +8.7 =36 | 0 31.3 | +6.9 29 | 0 25.8 | +10.3 =29 | | | 0 | 1:51.3 | +14.6 | 22 |
| | | Range Time | | 49.0 | +5.1 =15 | 46.0 | +3.8 =14 | 52.2 | +4.6 21 | 46.0 | +3.4 17 | | | | 3:13.2 | +6.9 | 11 |
| | | Course Time | | 9:19.5 | +21.1 =15 | 9:28.8 | +24.7 18 | 9:40.7 | +26.1 28 | 9:50.7 | +23.4 25 | 9:31.1 | +29.1 28 | | 47:50.8 | +1:59.0 | 22 |
| | | Penalty Time | | 0.0 | | 0.0 | | 0.0 | | 0.0 | | | | | 0.0 | | |
| 2 | 28 | BOE Tarjei | NOR | | | | | | | | | | | 2 | 52:03.2 | +59.2 | 2 |
| | | Cumulative Time | | 10:03.4 | +4.7 3 | 21:07.3 | +44.0 9 | 31:28.2 | +32.0 3 | 42:53.8 | +1:20.9 4 | | | | 52:03.2 | +59.2 | 2 |
| | | Loop Time | | 10:03.4 | +4.7 3 | 11:03.9 | +1:04.6 32 | 10:20.9 | +1.1 3 | 11:25.6 | +57.3 30 | 9:09.4 | +7.4 6 | | | | |
| | | Ski Time | | 10:03.4 | +13.1 8 | 20:07.3 | +20.6 5 | 30:28.2 | +25.2 5 | 40:53.8 | +39.7 6 | | | | 50:03.2 | +46.5 | 5 |
| | | Shooting | | 0 28.9 | +11.8 30 | 1 24.4 | +6.8 =19 | 0 32.3 | +7.9 =38 | 1 27.5 | +12.0 48 | | | 2 | 1:53.4 | +16.7 | 30 |
| | | Range Time | | 49.9 | +6.0 22 | 46.4 | +4.2 =22 | 54.3 | +6.7 40 | 49.4 | +6.8 =52 | | | | 3:20.0 | +13.7 | 27 |
| | | Course Time | | 9:13.5 | +15.1 10 | 9:17.5 | +13.4 7 | 9:26.6 | +12.0 9 | 9:36.2 | +8.9 8 | 9:09.4 | +7.4 6 | | 46:43.2 | +51.4 | 6 |
| | | Penalty Time | | 0.0 | | 1:00.0 | | 0.0 | | 1:00.0 | | | | | 2:00.0 | | |

Obrázek 3.2: Report s výsledky závodu

### 3.2.2 Selenium

Pro získání detailních statistik, které potřebujeme pro řešení některých shlukovacích úloh, bylo nutné využít framework Selenium. Všechna ostatní data bylo možné získat pomocí API a zpracování PDF reportů. Framework Selenium je popsán v sekci 4.2. Vytvoření skriptu pro automatické získávání dat pomocí frameworku Selenium nebylo složité z hlediska programování. Největším problémem je časová a paměťová náročnost, která stahování dat prodloužila. Tuto techniku bych z důvodu časové náročnosti za běžných okolností nepoužil, ale pro data, která bylo potřeba získat neexistovala jiná varianta. Naopak výhodou bylo lehčí zpracování chybných dat. Selenium stáhne postupně statistiky a výsledky ze závodu

v HTML souboru. Tento soubor je zpracován pomocí knihovny Beautifoulsoup4 4.2. Data se tímto způsobem dají získat ze závodů od roku 2018. Pro starší data je nutné výsledky získat pomocí zpracování PDF souborů.

### 3.2.3 Proces stahování dat

Informace o sezóně jsou vloženy manuálně. Následně je možné stáhnout informace o biatlonových víkendech pro konkrétní sezónu nebo všechny sezóny. Data jsou stažena pomocí API dotazu na `https://biathlonresults.com/modules/sportapi/api/Events?SeasonId=season.season_id`.


Obrázek 3.3: Stahování dat ve výsledné aplikaci

Pro všechny biatlonové víkendy jsou následně staženy data ze závodů, které se odehrávají v jednotlivých biatlonových víkendech. Závody jsou opět staženy pomocí dotazu na API url endpoint `https://biathlonresults.com/modules/sportapi/api/Competitions?EventId=event.event_ibu_id`. Stejně tak jsou následně staženy reporty s výsledky ze závodů pomocí dotazu na url endpoint `https://biathlonresults.com/modules/sportapi/api/Reports2?RaceId=competition.ibu_competition_id`. Tyto reporty jsou staženy a výsledky z nich jsou zpracovány a uloženy do databáze. Druhou možností, která je používána pro výsledky od roku 2018, je získání dat pomocí frameworku Selenium. Framework Selenium pracuje na adrese `https://biathlonresults.com/`. Selenium nalezne analýzu ke konkrétnímu závodu. Následně načte výsledky každého závodníka ze závodu včetně všech statistik a tyto infromace stáhne v HTML. HTML je zpracováno pomocí BeautifulSoup 4.2 a výsledky jsou uloženy do databáze. Pro závodníky, kteří se účastnili závodu, byly vytvořeny nebo aktualizovány jejich dlouhodobé statistiky.

Infromace o biatlonistech a biatlonistkách jsou staženy pomocí dotazu na API url endpoint `https://biathlonresults.com/modules/sportapi/api/Athletes?FamilyName=name`. Tento dotaz vrátí informace o maximálně 10 biatlonistech. Při inicializaci databáze jsou pomocí jednoduchého algoritmu postupně staženy informace o všech biatlonistech, které má server uložené v databázi. Pokud by při získávání výsledků byl nalezen biatlonista, o kterém nejsou uložena data v databázi, budou informace o něm vloženy do databáze během stahování výsledků.

# Kapitola 4

# Implementace

Cílem práce bylo vytvořit, implementovat a zhodnotit dolovací úlohy pro biatlonové statistiky. K prezentaci výsledků dolovacích úloh byla vytvořena webová aplikace, která slouží také ke stahování dat a zobrazení výsledků biatlonových závodů. Pro řešení této bakalářské práce jsem si vybral jazyk Python 4.1. Jazyk Python poskytuje širokou podporu knihoven pro stahování dat i řešení dolovacích úloh 4.2. V jazyce Python zároveň existuje velké množství frameworků, pomocí kterých lze implementovat webovou aplikaci. Pro implementaci webové aplikace jsem si vybral framework Django 4.3, který poskytuje podporu pro tvorbu jednotlivých částí webové aplikace, tak aby implementace mohla být rychlá a snadná. Na závěr této kapitoly jsou popsány detaily implementace a struktura aplikace 4.6.

## 4.1 Jazyk Python

Python je vysokoúrovňový jazyk, který poskytuje podporu pro funkcionální i objektově orientované programování. Kontrola datových typů v jazyce Python je dynamická a jedná se o interpretovatelný jazyk. Jazyk Python by mohl být zařazen také mezi skriptovací jazyky. Python vznikl v 90. letech 20. století a navrhl jej Guido van Rossum. Python je plně open-source projekt, který nabízí celou řadu baličků pro řešení různých problémů, které jsou také zdarma. Pro implementaci projektu jsem zvolil verzi 3.10.1 jazyka Python [12].

## 4.2 Knihovny jazyka Python

Jazyk Python poskytuje širokou podporu knihoven pro řešení analytických a vědeckých úloh. Knihovny byly užitečné pro řešení všech částí bakalářské práce. Knihovny byly využity při stahování, zpracovávání dat, k ukládání a načítání dat do databáze, pro řešení dolovacích úloh i pro prezentaci výsledků. V této sekci jsou popsány jednotlivé knihovny, které byly použity, jakým způsobem byly využity a jejich užtečnost pro řešení této práce.

### 4.2.1 NumPy

NumPy je základním balíčkem pro vědecké výpočty v jazyce Python. Jedná se o knihovnu jazyka Python, která poskytuje multidimenzionální vícerozměrné pole objektů a řadu operací a funkcí pro rychlou a efektivní práci s poli včetně složitých matematických operací. Kvalitní práce s NumPy poli je základem pro rychlé a efektivní řešení dolovacích úloh [9].

### 4.2.2 Pandas

Knihovna Pandas slouží pro rychlou a efektivní práci s daty, která jsou indexována v tabulce. Základním prvkem je `DataFrame`, který je tabulkou jakou známe z databáze. Pandas má širokou podporu pro načtení dat. V této práci byl využit Pandas pro načítání dat z databáze. Pandas umožňuje jednoduchý výběr dat, přidání sloupců nebo změnu struktury dat pomocí funkcí `pivot()` a `melt()`. K řešení různých výpočtu jako průměr,medián a další není potřeba procházet všechny řádky tabulky. Díky knihovně Pandas je snadné spojení více tabulek pomocí funkcí `concatenate()` a `merge()`. Knihovna Pandas se stala základem pro řešení všech dolovacích úloh, byla také využita pro čištění dat a selekci dat. Pomocí knihovny Pandas byly vyřešeny výpočty statistik nebo výběr dat pro řešení úloh. Algortimy, které byly použity pro řešení dolovacíh úloh jsem umí pracovat se vstupními daty v `DataFrame` [10].

### 4.2.3 Request

Knihovna request umožní v jazyce Python velmi snadno odesílat požadavky pomocí protokolu HTTP. Díky knihovně request není potřeba manuálně přidávat dotaz do URL rětězce nebo data zakódovat do požadavku POST. Keep-alive a HTTP spojení je plně automatické díky knihovně urllib3. Knihovna request umožní nejen získávání, ale i posílání dat. V této práci byla využita knihovna pro získávání dat během stahování dat, k tomu byla využita funkce `get()`. Pomocí návratového kódu můžeme zjistit, jestli byl dotaz úspěšný ,a následně získat obsah odpovědi. Knihovna request umožní i automatické vytvoření Session [14].

### 4.2.4 BeautifulSoup

BeautifulSoup je knihovna z jazyka Python, která slouží ke zpracovávání dat XML a HTML souborů. Knihovna má funkce pro hledání, navigování a modifikaci těchto souborů. Díky knihovně BeutifulSoup je práce s HTML a XML soubory jednoduchá a ušetří spoustu času při vývoji. BeautifulSoup nejprve zpracuje zadaný soubor a převede zadaný HTML nebo XML dokument na strom objektů v jazyce Python pomocí funkce `BeautifulSoup()`. Základním navigačním prvkem ve stromu objektů jsou tagy, které odpovídají tagům původního souboru. Každý tag má jméno a může mít i atribut. Pomocí tagů můžeme v dokumentu snadno vyhledávat, využíváme k tomu například funkce `find()` nebo `find_all()`. Knihovnu BeautifulSoup byla využita k nalezení správných dat v HTML a XML souborech při stahování [3].

### 4.2.5 Pdfplumber

Knihovna pdfplumber slouží k vyhledání podrobných informací v PDF dokumentu. Pdfplumber nejprve otevře PDF soubor. Po otevření máme dokument načtený v jazyce Python jako `pdfplumber.PDF class`, který obsahuje metadata o souboru a stránky dokumentu. Knihovnu je možné využít mnoha způsoby pro práci s PDF dokumenty. V této práci byla využita instanci třídy `pdfplumber.PDF class.pages`, pomocí které jsem získal text PDF dokumentů, který jsem dále zpracovával pomocí regulárních výrazů [11].

### 4.2.6 Re

Tato knihovna slouží k porovnávání regulárních výrazů. Řetězce, které jsou porovnávány, mohou být ve formátu Unicode i 8-bitové řetězce. Během řešení práce bylo potřebova získat

výsledky z PDF dokumentů. Nejlepším řešením bylo jednotlivé řádky rozpoznávat pomocí regulárních výrazů. Pro vytvoření regulárních výrazů jsem využil funkci `compile()` a pro porovnání textu a regulárních výrazů byla využita funkce `search()` [13].

### 4.2.7 Selenium

Selenium je knihovna, která poskytuje jednoduché API pro psaní funkčních nebo akceptačních testů za pomocí Selenium WebDriver [16]. Selenium se nejčastěji používá pro psaní testů, ale může být využito pro jakoukoliv automatickou interakci s prohlížečem, tedy i pro stahování dat. Při práci byl nejprve definoval webový ovladač, pomocí funkce `webdriver()`. Webovou stránku je možné načíst pomocí funkce `get()`. Pro měnení obsahu webové stránky a získání příslušných dat byla použitá funkce `find_element()` pro nalezení správných dat a tlačítek, která je potřeba stisknout. Pro stisknutí tlačítka byla využita funkce `click()`. Po nalezení správné konfigurace stránky byl stažen HTML zdrojový kód, který byl už dále zpracován pomocí knihovny BeautifulSoup [18].

### 4.2.8 Scikit-learn

Scikit-learn je Python knihovna, která vytváří podporu pro strojové učení. Knihovna scikit-learn nám umožní pracovat s algoritmy strojového učení bez učitele i s učitelem. Kromě samotných algoritmů podporuje knihovna také různé funkce pro přípravu a výběr modelu, předzpracování dat, vyhodnocení modelu a další utility. Knihovna scikit-learn byla využita při řešení deskriptivních i prediktivních dolovacích úloh [21].

#### scikit-learn shlukování

Pro shlukovou analýzu má knihovna scikit-learn modul `clustering`. Každý shlukovací algoritmus je implementován ve dvou fázích. První fází je funkce `fit()`, která vytvoří jednotlivé shluky na základě trénovacích dat a vrátí pole integerů, které odpovídají jednotlivým třídám. Jednotlivé třídy získáme pomocí funkce `labels_` [15].

V bakalářské práci byly využity algoritmy k-means a DBSCAN. K jejich implementaci jsem využil právě modul `clustering`. Pro přípravu dat jsem využil modul `preprocessing`, pomocí kterého byly pro oba dva algoritmy připraveny data do požadovaného formátu. Algoritmus k-means má jako vstupní parametr počet tříd $k$. Následně pomocí funkcí `fit()` a `predict()` rozřadíme data do jednotlivých tříd a pomocí funkce `labels_` získáme informace o rozřazení do tříd. Algoritmus DBSCAN má jako vstupní parametry poloměr $\epsilon$ a minimální počet shluků v okolí $MinPts$. Pomocí funkce `fit_predict` rozřadíme vstupní data do jednotlivých tříd [15].

#### Scikit-learn predikce

Knihovna scikit-learn poskytuje širokou podporu algoritmů pro klasifikaci i predikci. Algoritmy pro predikci mají dvě hlavní funkce. První funkcí je funkce `fit()`, která má na vstupu tréninková data a odpovídající třídy pro tyto vzorky. Funkce `fit()` provede trénování algoritmu a nastaví jeho vnitřní parametry. Druhou funkcí je funkce `predict()`, která predikuje výslednou třídu pro testovaná data. Výstupem funkce `predict()` je pole s výsledkem predikce. Nejvíce algoritmů využívá Numpy pole 4.2.1 [21].

V této práci byla využita vícenásobná lineární regresi 2.3. Lineární regrese je jedním z algoritmů, který je implmenentován v knihovně scikit-learn mezi lineárními modely. Data,

které jsem pro trénování modelu a pro ověření jeho správnosti byla náhodně vybrána pomocí funkce `train_test_split` v poměru 3:1 z modulu `model selection`. Stejně jako ostatní algoritmy je na vstupu trénovací množina a pomocí funkce `fit()` je model vytrénován. Následně je možné pomocí funkce `predict()` predikovat další hodnoty. Úspěšnost modelu byla hodnocena pomocí modulu `metrics`, který je implementován v knihovně scikit-learn [15].

## 4.3 Django

Framework Django je framework pro vývoj webových aplikací. Jedná se o volně dostupný framework, který poskytuje podporu při vývoji webových aplikací takm, aby bylo možné soustředit se na vývoj samotné aplikace, ale nebylo třeba při vývoji řešit problémy s vývojem webu. Aplikace Django byla navržena tak, aby bylo možné koncept aplikace, co nejrychleji implementovat a vytvořit reálný produkt. Framework poskytuje mnoho funkcí, které je možné využít. Django také zajišťuje bezpečnost výsledné aplikace a poskytuje ochranu proti SQL injection a zajišťuje integritu dat v databázi. Framework Django používá architekturu model-view-template [7].

### 4.3.1 Architektura model-view-template

Architektura model-view-template je tvořena ze tří částí model, view a template. Architektura je velmi podobná architektuře model-view-controller, která je zřejmě používanější. View má v architektuře model-view-template podobnou roli jako controller v architektuře model-view-controller, ale s malými odlišnostmi. Podobně template v architektuře model-view-template má podobnou roli jako view v arckitektuře model-view-controller [6].

#### Model

Model funguje jako rozhraní pro data. Pomocí modelu se vytváří, upravují a mažou data v celé aplikaci. Jedná se vlastně o logickou datovou strukturu na pozadí aplikace, která je reperezentována databázi. Model zajišťuje vytváření tabulek a jejich polí, stará se o nastavení omezení a jejich dodržování. Pro implementaci databáze byla vybrána technologie mysql [8].

#### View

Hlavní funkcí View je přijímat HTTP požadavky a vracet HTTP odpovědi. Odpověďí může být HTML dokument, přesměrování, obrázek, XML dokument nebo chyba s návratovým kódem 404, tedy cokoliv může prohlížeč zobrazit. View je součástí uživatelského rozhraní, které poskytuje data pro template [8].

#### Template

Template jsou šablony pro zobrazování dat, které jsou napsány v HTML, CSS a JavaScriptu v HTML souboru. Pomocí template je možné snadno vytvářet šablony pro aplikaci a udržet tak její jednotný styl [8].

## 4.4 Ajax

Ajax neboli Asynchronous JavaScript and XML používá kombinaci `XMLHttpRequest`, které jsou vestavěny do prohlížeče k získání dat z webového serveru a JavaScriptu v kombinaci s HTML pro zobrazení nebo využití získaných dat. Data mohou být, a často jsou přenášené na jako XML, ale jako prostý text nebo JSON. Ajax získává informace asynchroním způsobem. Díky tomu, není nutné aktualitovat celou webovou stránku, ale pouze její části [2]. Využití technologie Ajax bylo v tomto projektu důležité především pro stahování dat. Stahování dat může trvat i několik minut a je klíčové, aby mohl uživatel v průběhu pracovat dále a požadavek proběhl asynchronně. Ajax byl dále využit pro získávání dat při aktualizaci pouze části stránky. Nejdůležitější bylo využití asynchronního získávání dat během překreslování grafů s výsledky z dolovacích úloh.

## 4.5 ChartJs

ChartJs je knihovna jazyku Javascript, která slouží k vytváření interaktivních grafů na webových stránkách za pomocí HTML. Knihovna byla využita především k prezentaci výsledků jednotlivých experimentů ze shlukovací analýzy [5].

## 4.6 Struktura aplikace

Adresářová struktura byla vytvořena podle doporučení v dokumenteci Django frameworku [7]. Základní struktura aplikace je na obrázku níže. Při návrhu struktury jsem dbal na oddělení jednotlivých modulů a především na rozdělení funkčnosti do různých částí aplikace.

```
Application
├── Biathlon
│   ├── migrations
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── Miningapp
│   ├── models.py
│   ├── urls.py
│   └── views.py
├── Clustering
│   ├── coursetime.py
│   └── shooting.py
├── Download
│   ├── download.py
│   └── downloadresult.py
├── Prediction
│   ├── linearregression.py
│   └── predict.py
├── Stats
├── Static
└── Templates
```

### 4.6.1 Biathlon

Je hlavní složkou Django projektu, která obsahuje soubory důležité pro nastavení a konfiguraci celé aplikace.

#### Settings.py

Soubor `settings.py` obsahuje kompletní nastavení celého Django projektu. V nastavení bylo nastaveno, v jaké složce jsou static a template soubory. Dále je definována technologie databáze, která je využita, časová zóna nebo přístupová adresa aplikace.

#### Urls.py

Soubor `urls.py` slouží pro zpracování url adres a určuje způsob zpracování url adresy. Po zaslání pořadavku se adresy přesměřují do správné aplikace. Každá aplikace má svůj `urls.py` soubor.

#### Wsgi.py

Tento soubour obsahuje konfiguraci rozhraní pro nasazení aplikace na webový server.

### 4.6.2 MiningApp

MiningApp je jedinou a hlavní aplikací celého projektu. Aplikace byla vytvořena pomocí příkazu `python manage.py startapp MiningApp`. Tento Django příkaz vytvoří aplikaci a soubory, které jsou potřebné pro používání této aplikace.

#### Apps.py

Tento soubor obsahuje konfiguraci pro nastavení aplikace.

#### Models.py

V tomto souboru jsou definovány všechny databázové tabulky včetně všech informací o polích a integritních omezeních. Důležité je i správné nastavení cizích klíčů. Tabulky jsou definovány podle návrhu databáze 3.1. Tabulky jsou následně vytvořeny pomocí příkazů `python manage.py makemigrations` a `python manage.py migrate`.

#### Urls.py

Soubor `urls.py` slouží pro zpracování url adres. Hledá požadovaný vzor url adresy v tomto souboru. Když je vzor nalezen, tak je zavolána Python funkce ze souboru views. Kvalitní a čistý `urls.py` je základem, aby aplikace byla přehledná.

#### Views.py

Ve `views.py` jsou Python funkce, které jsou volané ze souboru `urls.py`. Tyto funkce zajišťují komunikaci mezi modelem a templatem. Views při stahování dat volá funkce pro a ukládá stažená data pro databáze. Druhou typickou funkcí je získávání dat z databáze, které jsou odeslané do šablon a dále se vykreslí uživateli.

#### Migrations

Tato složka obsahuje soubory, které vznikly příkazem `python manage.py makemigrations`. Tyto soubory jsou vytvořeny podle definic databázových tabulek v souboru `models.py`. Pomocí příkazu `python manage.py migrate` jsou podle definicí v těchto souborech vytvořeny tabulky v databázi.

### 4.6.3 Download

Modul Download slouží pro stahování dat ze serveru www.biathlonresult.com. Modul Download obsahuje soubor `download.py` a soubor `downloadresult.py`.

#### Download.py

Tento soubor obsahuje třídu `DataDownloader`, která se využívá pro stažení dat a jejich validaci. Při potřebě stáhnout data se vytvoří instance třídy `DataDownloader`. Pro stažení dat pomocí API slouží funkce `get_data`, která vytvoří sezení a stáhne data podle parametru URL. Pro stažení eventů slouží funkce `get_events`, která zpracuje stažená data jako pole slovníků. Funkce vrátí zpracované pole slovníků. Pro stažení informací o závodech je funkce `get_competitions`. Funguje stejně jako funkce `get_competitions`, zpracuje stažená data jako pole slovníků a to vrátí pomocí příkazu `return`. Pro stažení výsledků je funkce `get_race_result`. Tato funkce stáhne PDF soubor s výsledky z daného závodu. Následně je PDF soubor načten řádek po řádku pomocí knihovny pdfplumber a zpracován. Řádky jsou rozpoznávány a zpracovávány pomocí regulárních výrazů. Pro ověření validity stažených informací slouží funkce `check_res_line()`, `check_cumulative()`, `check_shooting()`, `check_course()`. Výsledkem je opět pole slovníků se staženými informacemi o výsledcích závodu. Poslední funkcí je funkce `get_athletes()`. Tato funkce stáhne informace o všech biatlonistech a biatlonistkách, které jsou dostupné na webu www.biathlonresult.com. Data o biatlonistech jsou stažena při inicializaci celé aplikace. Během každého dotazu na API jsou získány data pouze o 10 biatlonistech. Pro získání informací o všech biatlonistech byl využit jednoduchý algoritmus, který postupně získá data o všech biatlonistech. Výsledkem funkce je pole slovníků, ve kterém jsou uloženy všechny informace pro vložení údajů do databáze. V případě, že ve výsledcích bude nový biatlonista, o kterém nejsou uložena data v databázi jsou informace o biatlonistovi staženy dodatečně pomocí funkce `get_athlete()`.

#### Downloadresult.py

Soubor `downloadresult.py` slouží pro stažení výsledků a detailních statistik pomocí techniky `Selenium`. Pro získání těchto dat slouží třída `RaceResult`. Pro stažení výsledků ze závodu je vytvořena instance této třídy. Následně funkce `download_race_result()` řídí logiku stahování dat ze závodu a vrátí zpracované výsledky. Funkce `session_init()` vytvoří okno prohlížeče a připraví webovou stránku a funkce `get_result()` stáhne všechny výsledky a statistiky.

### 4.6.4 Clustering

Modul Clustering slouží pro řešení dolovacích úloh pomocí shlukování. Pro každý typ dolovacích úloh obsahuje modul Clustering jeden soubour.

#### Coursetime.py

Soubor coursetime.py obsahuje třídu `Course`. Při řešení dolovací úlohy je vytvořena instance třídy `Course`. Tato třída slouží pro řešení dolovacích úloh pro shlukování pro běžecký čas. Dolovací úlohy pro shlukování jsou popsány v sekci 5. Funkce `discipline_course_behind_count` a funkce `lap_course_behind_count` slouží pro výběr dat pro shlukovací úlohu, které uloží do DataFrame 4.2.2. Funkce `course_behind_prepare` a `lap_behind_prepare` připraví data v DataFramu podle typu dolovací úlohy a využije funkce `kmeans`,`dbscan` a `hiearchy` pro rozdělení dat do tříd. Jako výsledek vrátí DataFrame, ve kterém každý řádek patří do jedné třídy. Funkce `kmeans`, `dbscan` a `hiearchy` využívají pro rozdělení do tříd funkce z knihhovny `scikit-learn` 4.2.8. Funkce `discipline_course_behind_count` a funkce `lap_course_behind_count` vyberou z DataFrame data pro zobrazení uživateli, upraví je a vrátí ve formátu JSON.

#### Shooting.py

Tento soubor slouží pro řešení shlukovacích dolovacích úloh, ve kterých se zkoumají střelecké výkony. Třída `Shooting` funguje velmi podobně jako třída `Course`. Při řešení shlukovacích experimentů funkce `time_compare_accurancy` a `shot_accurancy` slouží pro výběr a úpravu dat, které vrátí ve formátu JSON. Všechny dolovací úlohy využívají funkce `kmeans`,`dbscan` a `hiearchy` pro rozřazení objektů do tříd.

### 4.6.5 Stats

Modul Stats obsahuje soubor `create_stats.py` s třídou `CreateStats`. Tato třída má dvě funkce, které jsou zavolány po každém uložení nového výsledku do tabulky `result`. Funkce `predict_stats` slouží pro vytvoření statistik pro závodníka. Funkce nejprve načte data do DataFrame a následně spočítá všechny statistiky a uloží je do tabulky `PredictStats`. Tyto statistiky slouží pro predikci závodů. Další fukce `create_stats` z třídy `CreateStats` spočítá statistiky závodníka pro danou sezónu a uloží je do tabulky `Stats` nebo statistiky aktualizuje.

### 4.6.6 Prediction

Tento modul slouží pro vytvoření predikce výsledků a k zhodnocení této predikce.

#### Linearregression.py

Tento soubor tvoří třída `Regression`, která vytváří model pro predikci výsledků. Nejprve jsou vybrány statistiky z databáze z tabulky `PredicStats`. Pro každý z parametrů (bežecký a střelecký čas, umístění v minulých závodech a střeleckou úspěšnost) je vypočítán průměr. Tyto čtyři základní parametry jsou vstupem pro funkci `fit()` z třídy `LinearRegression` z knihovny `scikit-learn`. Vytrénovaný model je následně uložený pomocí funkce `gzip` a použit k predikci výsledků.

#### Predict.py

Tento soubor slouží pro predikci výsledků pro konkrétní závod. Třída `Predict` obsahuje funkci `predict_result`, která nejprve zavolá funkci `get_start_list`, která stáhne startovní listinu pro daný závod. Startovní listina je následně načtena a pro každého ze závod-

níků jsou načteny jeho pokročilé statistiky a upraveny tak, aby mohly sloužit jako vstup pro vytrénovaný model. Model, který byl vytrénován a uložen pomocí třídy `Regression`, určí výsledné umístění závodníka v daném závodu. Umístení závodníků jsou následně seřazena a uložena do tabulky `Prediction`.

### 4.6.7 Static

Tato složka obsahuje soubory, které potřebuje webová aplikace k vykreslení stránky. Nejedná se o HTML soubory, které jsou ve složce Template, ale o CSS soubory, JavaScript a obrázky, které jsou použité na webové stránce. Tyto soubory jsou označovány jako statické. CSS soubory a JavaScript v této složce byly převzaty z open-source šablony pro tvorbu webových stránek. Zdrojové kódy jsou okomentovány a byly dodrženy všechny licenční požadavky pro použití těchto souborů.

### 4.6.8 Templates

Složka `Templates` obsahuje všechny šablony pro vykreslovaní webové stránky. Šablony byly vytvořeny podle open-source šablony, stejně jako u složky Static.

# Kapitola 5

# Shlukovací experimenty

Shlukovací metody byly využity pro řešení deskriptivní dolovacích úloh. Cílem u shlukování bylo najít vzory a modely dat, které nejsou snadno zjistitelné. Také bylo cílem najít v datech zajímavé vzory, které by mohly zajímavat biatlonové nadšence. Tyto informace by mohly zajímat i samotné biatlonisty a biatlonistky, kteří by se mohli dozvědět nové informace o svých výkonech a na základě těchto informací své výkony upravit. Experimenty byly navrženy tak, aby pokryly různé statistické oblasti biatlonu a nebyly zaměřeny jen na jednu oblast. Všechny experimenty byly provedeny pomocí tří shlukovacích metod jedná se o metody založené na rozdělování, hierarchické metody a metody založené na hustotě. Z metod založených na rozdělování byl vybrán pro shlukovací experimenty algoritmus kmeans. Shlukování za pomocí hiearchických metod proběhlo s rozkladem zdola nahoru. DBSCAN byl algroritmus, který vytvořil shluky na základě hustoty objektů. Hodnoty min a max na výsledných grafech jsou hodnoty první a třetího kvartilu.

## 5.1 Porovnání běžeckého času - disciplíny

Při porovnání bežeckého času jsem se zaměřil na rozdíly, které mají biatlonisté napříč jednotlivými disciplíny. Jaký biatlonista dosahuje a nedosahuje dobrých bežeckých časů lze zjistit jednoduchým SQL dotazem. Každý biatlonista, ale může mít disciplínu, která mu vyhovuje více a dosahuje v ní lepších běžeckých časů než v jiné disciplině. Tento rozdíl může způsobit rozdílná délka závodu nebo jiný typ startu závodu, kde se může jednat o intervalový start nebo hromadný start. Cílem bylo zjistit skupinu závodníku, kteří mají bežecký čas na stejné úrovni v každé disciplíně a další skupiny závodníku, kteří dosahují v některé z disciplín horší nebo lepší výsledky.

### 5.1.1 Experiment s disciplínami I.

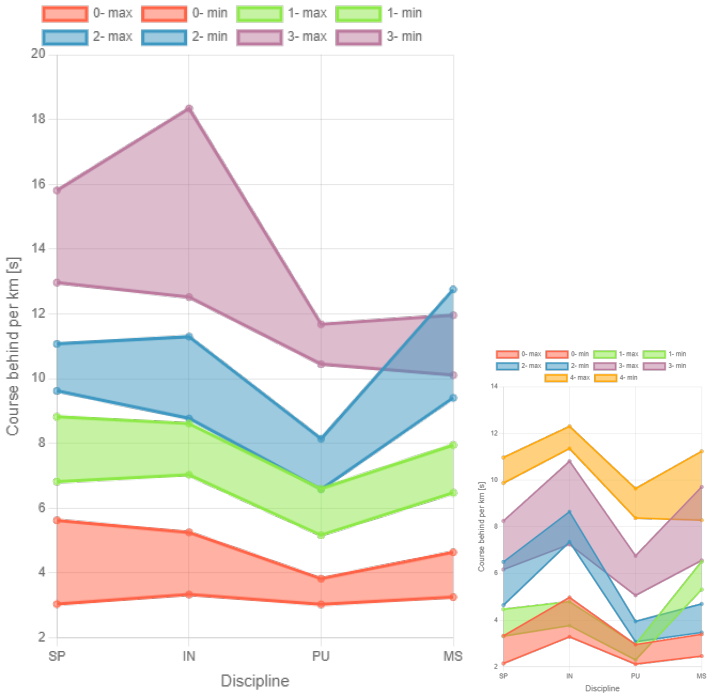
#### Výběr dat

Při prvním experimentu byly porovnány průměrné běžecké ztráty v přepočtu na 1km v rámci jednotlivých disciplín. Pro každého biatlonistu nebo biatlonistku byla spočítána jejich průměrná ztráta v běhu na nejlepší bežecký čas ve sprintu, stíhacím závodě, v inviduálním závodě a v závodě s hromadným startem. Zahrnuté jsou pouze výsledky biatlonistů, kteří se zúčastnili alespoň jednoho závodu v každé disciplíně. Tato data byla vstupem pro shlukovací algoritmy.

### Zhodnocení experimentu

Při tomto experimnetu jsem nezískal výsledky, jaké jsem očekával. Biatlonisté byli rozděleni do tříd podle celkových bežeckých ztrát. Pomocí tohoto experimentu se nepovedlo zjistit, kteří biatlonisté mají lepší nebo horší bežecké časy v různých disciplinách a o kolik času. V rámci experimentu bylo zjištěno, že v závodech s intervalovým startem dochazí k větším bežeckým ztrátám na nejlepšího závodnika oproti závodům, kde startují závodníci ve skupině. Tento experiment může pomoci pro rozřazení závodníku podle celkové běžecké výkonnosti do různých skupin.

Nejlepší výsledky byly získány od algoritmu kmeans, který při nastavení vhodného počtu tříd rozřadil závodníky do velmi podobných a velikostně stejných tříd. Algoritmus DBSCAN nezvládl rozřadit závodníky do stejného počtu tříd, ale dokázal najít menší skupiny závodníků, u kterých jsou jejich výsledky naprosto totožné. Naopak některé skupiny jsou příliš velké a nelze z nich nic vyčíst. Shlukováním pomocí hiearchické metody jsem získal podobné výsledky jako u algoritmu kmeans, ale jednotlivé třídy si nejsou tak podobné, jak u algoritmu kmeans. Výsledky z algortmu kmeans jsou na obrázku 5.1.




Obrázek 5.1: Experiment I.: Rozdělení pomocí algoritmu kmeans a hierarchické metody

### 5.1.2 Experiment s disciplínami II.

Při prvním experimentu jsem nezískal rozdělení, jaké bylo cílem. Pro druhý experiment jsem upravil výběr dat pro shlukovací algoritmy, aby bylo dosažené očekávaného rozdělení.

### Výběr dat

Pro každého závodníka byla vypočítána průměrná ztráta v běhu na 1km během všech závodů. Následně byla spočítána pro každého biatlonistu průměrná ztráta běžeckého času

na nejlepší bežecký čas v přepočtu na 1km stejně jako u prvního experimentu. Pro každou disciplínu jsem následně odečetl průměrnou ztrátou v dané disciplíně s průměrnou ztrátu napříč všemi disciplínami. Opět byla vybrána data pouze u závodníku, kteří se zúčastnili všech disciplín. Takto upravená data byla vstupem pro shlukovací algoritmy.

#### Zhodnocení experimentu

U druhého experimentu se povedlo najít skupiny závodníků, kteří dosahují různých běžeckých časů napříč všemi disciplínami v porovnání se svým průměrným bežeckým časem. Většina závodníků dosahuje lepších bežeckých časů ve stíhacích závodech a v závodech s hromadným startem. Při tomto experimentu se povedlo najít různorodé skupiny závodníků. Mezi některými skupinami nejsou velké rozdíly, ale podařilo se najít skupiny, které jsou odlišitelné. Dokázal jsem nejen najít skupiny, které dosahují horších výsledků v závodech s intervalovým startem, ale také odlišit skupiny závodníků, kteří dosahují lepší nebo horší výsledky ve sprintu nebo v individuálním závodě. Podobně je tomu při porovnání stíhacího závodu s hromadným startem. Rozdíly mezi běžeckými výkony se v některých skupinách blíží až 5 sekundám na 1 kilometr. V takových případech už je rozdíl velmi markantní a projeví se na výsledcích biatlonistů. Při experimentu se povedlo najít i skupinu závodníků, kteří mají vyrovnané běžécké výkony napříč všemi disciplínami. Nevýhodou tohoto experimentu je, že došlo k porovnání pouze výsledků závodníků v poměru s jejich výkony. Ztratil jsem tak informaci o jejich celkovém bežeckém času, přičemž v jednotlivých skupinách jsou závodnici s velmi odlišnými běžeckými výsledky. Pro dosažení kvalitních výsledků je lepší provádět shlukování pouze pro data z jedné sezóny.

V tomto experimentu jsem získal nejzajímavější výsledky pomocí algoritmu DBSCAN. DBSCAN sice některé závodníky označí za šum, ale při správném nastavení epsilon dokáže rozdělit závodníky do velmi podobných shluků. Výsledky z tohoto algoritmu jsou na obrázku 5.2. Velmi dobré výsledky jsem získal i z algoritmu kmeans.

### 5.1.3 Experiment s disciplínami III.

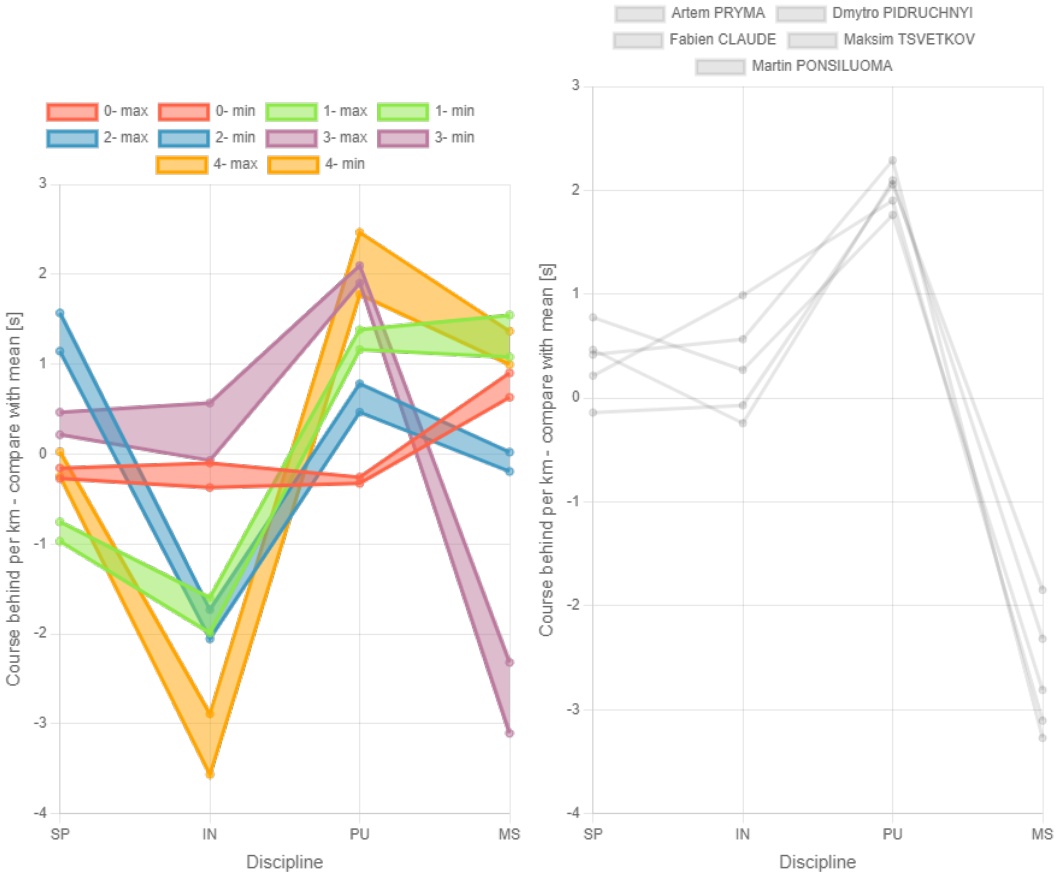
Ve třetím experimentu jsem se soustředil na odstranění nedostatků z druhého experimentu. U tohoto experimentu jsem se soustředil nato, aby biatlonisté a biatlonistky byli běžci na podobné úrovni a zároveň dosahovali stejných výsledků napříč všemi disciplínami.

#### Výběr dat

U tohoto experimentu jsem data, vybral a upravil stejně jako u druhého experimentu. Rozdíl u tohoto experimentu spočívá v tom, že kromě porovnání časů s průměrem napříč všemi disciplínami byl pro shlukovací algoritmy vstup atribut s průměrnou ztrátou ze všech disciplín.

#### Zhodnocení experimentu

U tohoto experimentu jsou použitelné pouze výsledky z algoritmu DBSCAN. U algortimu DBSCAN záleží na jednotlivých třídách. Některé vzniklé třídy jsou příliš velké a nepřehledné. Povedlo se ale vytvořit třídy, které nalézají závodníky se stejnými běžeckými výkony a jejich rozložením napříč disciplínami. Těchto případů je minimum, navíc spoustu závodníků bylo označeno za šum. U algoritmů kmeans a hieachického shlukování se mi nepovedlo získat kvalitní výsledky.




Obrázek 5.2: Experiment II.: Rozdělení podle algoritmu DBSCAN a jedna z tříd

### 5.1.4 Experiment s disciplínami IV.

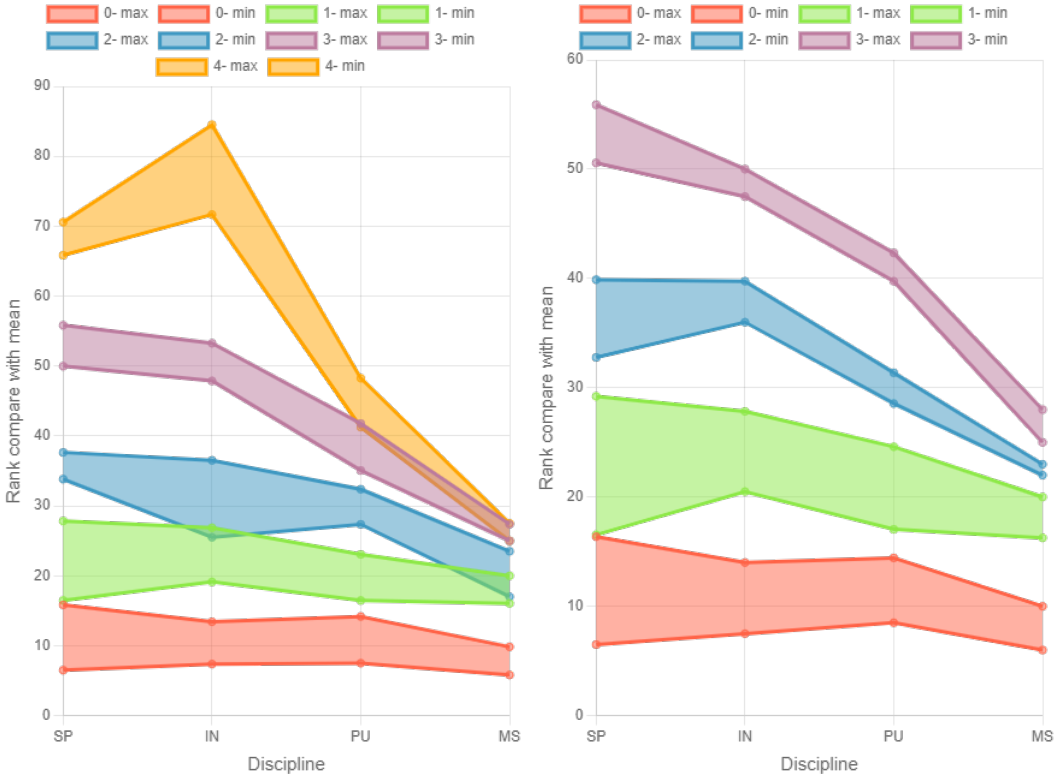
První tři experimenty byly provedeny z hlediska porovnání běžeckých ztrát na nejlepšího závodníka. Při dalších experimentech jsem porovnával skupiny závodníků podle umístění běžeckého času.

#### Výběr dat

V tomto experimentu jsem porovnal průměrné umístění běžeckého času v rámci jednotlivých disciplín. Data byla vybrána stejně jako u prvního experimentu, poze místo běžecké ztráty, bylo vybráno umístění běžeckého času.

#### Zhodnocení experimentu

Výsledky experimentu jsou velice podobné jako u prvního experimentu, pouze je větší rozdíl v závodech s hromadným startem a závodech s intervalovými starty. To je způsobeno tím, že v závodech s hromadným startem startuje méně závodníků. Výsledky shlukování jsou na obrázku 5.3.




Obrázek 5.3: Experiment IV.: Rozdělení podle algoritmu kmeans a DBSCAN

### 5.1.5 Experiment s disciplínami V.

Tento experiment slouží pro porovnání s druhým experimentem přičemž cílem získat lepší skupinu závodníků, kteří mají konzistentní výkony při všech závodech.

#### Výběr dat

Výběr dat byl stejný jako u druhého experimentu s tím rozdílem, že místo běžecké ztráty na kilometr bylo vybráno průměrné umístění závodníků v běžeckém čase ve všech disciplínách v porovnání s průměrným umístěním závodníků v jednotlivých disciplínách.

#### Zhodnocení experimentu

Rozdílný počet startujících závodníků v jednotlivých disciplínách způsobuje velmi rozdílné umístění napříč disciplínami. Z tohoto experimentu, nejsou kvalitní výsledky.

## 5.2 Vývoj běžeckého času v průběhu závodu

V dalších dolovacích úlohách jsem se zaměřil na vývoj běžeckého času v průběhu závodu u jednotlivých biatlonistů a biatlonistek. Cílem bylo zjistit skupiny závodníku podle průběhu běžeckého času během závodu. Očekávaným výsledkem byly skupiny závodníků, mezi nimiž je skupina závodníků, která má konstantní bežecký výkon v průběhu celého závodu a další skupiny závodníků, v nichž mají závodníci lepší výkon na začátku celého závodu nebo naopak na konci závodu.

### 5.2.1 Experiment s vývojem běžeckého času I.

#### Výběr dat

U prvním experimentu byly vybrány průměrné běžecké ztráty v přepočtu na 1 km v rámci jednotlivých okruhů pro závodníky. U každého biatlonisty nebo biatlonistky jsem spočítal jejich průměrnou ztrátu v běhu na nejlepší bežecký čas v každém z okruhů. Závodník se musel zúčastnit minimálně dvou závodů v dané disciplíně, aby byl zařazen do shlukování. U sprintu, který se jezdí nejvíce, se musí závodník zúčastnit minimálně 4 závodů pro zařazení do shlukování. Tato data byly vstupem pro shlukovací algoritmy.

#### Zhodnocení experimentu

V prvním experimentu dojde k rozdělení závodníků podle celkové výkonnosti. U všech algoritmů je jasné, že v průběhu závodů většina závodníků má větší a větší ztrátu na biatlonisty s nejlepším bežeckým časem. U závodů s intervalovým startem jsou patrné běžecké rozdíly, hned od prvního kola a v průběhu závodů se rozestupy zvětšují. Ve stíhacích závodech a závodech s hromadným startem jsou rozestupy v prvních kolech minimální, ale v průběhu závodů rozdíly rostou s větší rychlostí a nejpomalejší závodníci mají v posledních okruzích závodů velké ztráty.

U tohoto experimentu byly získány zatím nejlepší výsledky za pomocí hierarchické shlukování. Velmi kvalitní jsou i výsledky ze shlukování za pomocí algoritmu kmeans. Algoritmus DBSCAN fungoval různě na různý typ disciplín. U shlukování u disciplín jako jsou sprint a vytrvalostní závod, ve kterém závodí velký počet sportovců, algoritmus vytvářel shluky, které byly příliš velké a nepřehledné. Naopak pro stíhací závod a závod s hromadným startem jsou třídy malé a spoustu objektů bylo označeno za šum.


Obrázek 5.4: Experiment I.: Rozdělení Kmeans pro závod s hromadným startem a sprint

### 5.2.2 Experiment s vývojem běžeckého času II.

Obdobně jako v rámci experimentování s porovnáním běžeckého času během disciplín, jsem se zde pokusil porovnat výsledky v jednotlivých kolech s celkovým průměrným časem.

#### Výběr dat

Pro každého závodníka jsem spočítal jeho průměrnou ztrátu běžeckého času v rámci dané disciplíny v přepočtu na 1 kilometr. Následně jsem spočítal jejich průměrnou ztrátu běžeckého času ve všech běžeckých kolech v průběhu závodu na 1 kilometr. Pro všechna kola jsem od celkového průměru pro danou disciplínu odečetl průměrnou ztrátu na jedno kolo. Parametry pro výběr závodníků byly stejné jako u prvního experimentu.

#### Zhodnocení experimentu

U tohoto experimentu se mi povedlo dosáhnout požadovaného rozdělení do skupin. Nejlépe se toto rozdělení podařilo u disciplíny sprint a to jak pro ženy, tak pro muže. Důvodů může být více, jedním z nich je, že se jedná o závod s intervalovým startem. Dalším důvodem může být, že se ve sprintu se jezdí nejvíce závodů a v databázi je k nim uloženo nejvíce dat. U rozdělení do jednotlivých tříd se povedlo najít skupinu nebo skupiny závodníků, kteří mají nejkonstantnější výkony běžeckého času v průběhu celého závodu. Další skupiny závodníků v průběhu závodů svůj běžecký výkon v porovnání s nejrychlejším běžcem závodu postupně zhoršovali. Některé skupiny závodníků se zhoršují v průběhů celého závodu. Na obrázku 5.5 je vidět další skupina závodníků, která v prvních dvou kolech závodu drží konstantní běžecký čas, ale v posledním kole závodu výrazně zpomalí.

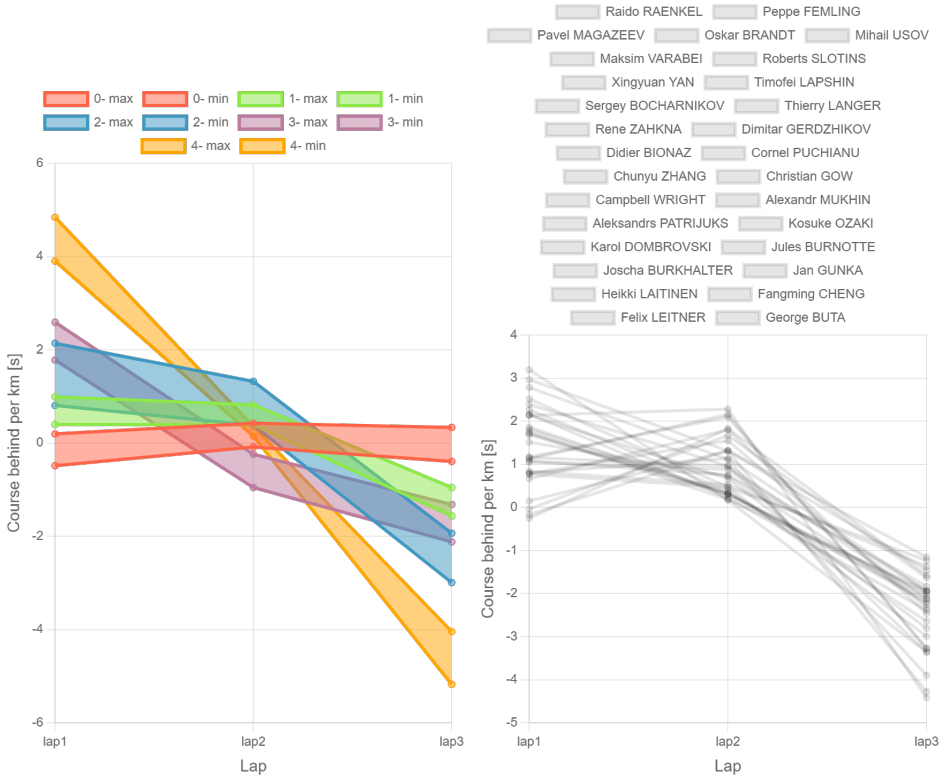
Nejlepších výsledků shlukování v tomto experimentu se mi povedlo dosáhnout pomocí algoritmu kmeans. Algoritmus kmeans dokázal rozřadit biatlonisty a biatlonistky do velmi zajímavých tříd. Podařilo se velmi dobře třídy od sebe odlišit. U hierarchického shlukování bylo rozřazení do tříd podobně úspěšné jako u algoritmu kmeans, ale podobnost v rámci jednotlivých tříd byla menší. Algoritmus DBSCAN dokázal opět najít skupiny závodníků, ve kterých jsou výkony velmi podobné. Některé skupiny, jsou ale velmi malé a naopak některé skupiny jsou velké a nelze se v nich jednoznačně orientovat.

### 5.2.3 Experiment s vývojem běžeckého času III.

U porovnávání běžeckého času napříč disciplínami nefungovala dobře při srovnání podle pořadí bežeckého času. Tato skutečnost byla dána tím, že jednotlivých disciplín se účastnil velmi rozdílný počet závodníků. Během závodu ale zůstává počet biatlonistů stejný a porovnání by, tak mělo fungovat lépe. Od tohoto experimentu jsem očekával podobné výsledky jako první experiment.

#### Výběr dat

U tohoto experimentu jsem pro každého závodníka porovnal jeho umístení běžeckého času v rámci jednotlivých okruhů s jeho celkovým průměrným umístením. Do shlukování vstoupili pouze závodníci, kteří splnili stejné podmínky jako u prvního a druhého experimentu.




Obrázek 5.5: Experiment II.: Kmeans pro sprint a vybraná třída

#### Zhodnocení experimentu

V tomto experimentu došlo k jasnému rozdělení podle běžecké výkonnosti. Výsledky jsou velmi obdobné jako u prvního experimentu pouze ještě rozdělení do jednotlivých tříd je markantnější.

### 5.2.4 Experiment s vývojem běžeckého času IV.

Tento experiment je obdobný jako druhý experiment, jen místo porovnání běžecké ztráty dochází k porovnání umístění běžeckého času.

#### Výběr dat

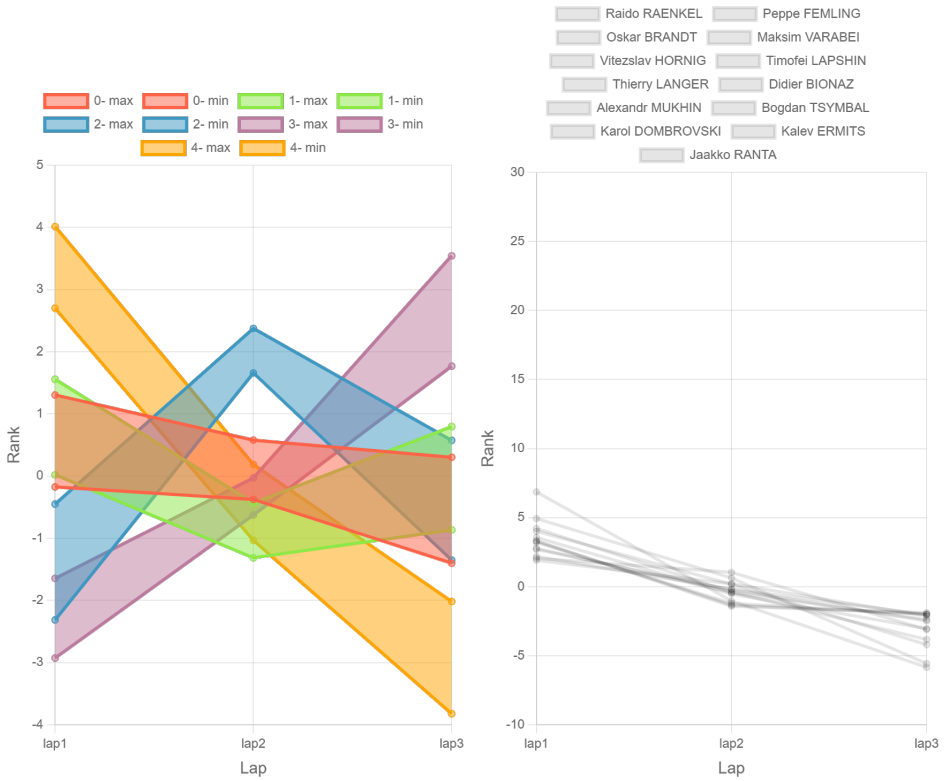
Pro každého závodníka jsem nejprve vypočítal jeho průměrné umístění v rámci disciplíny. Následně bylo pro každý okruh během závodu bylo vypočítáno průměrné umístění v rámci daného okruhu a pro každý okruh bylo průměrné umístení odečteno. Po této úpravě data vstupují do shlukovacích algoritmů.

#### Zhodnocení experimentu

Tento experiment přinesl zajímavé výsledky. Výsledky jsou velmi podobné jako u druhého experimentu. Podařilo se závodovníky rozdělit do různých skupin podle jejich strategie běžeckého výkonu v průběhu závodu. Na rozdíl od druhého experimentu jsou výsledky lépe interpretovatelné pro všechny disciplíny. Na obrázku 5.6 je vidět rozdělení závodníků do 5

tříd ve sprintu podle algoritmu kmeans. Každá třída obsahuje jiný typ závodníků. Podařilo se najít závodníky s konstantní výkonností, zhoršující i zlepšující se výkonností běhu během závodu. Na druhém obrázku je zobrazena skupina závodníků, u kterých se jejich běžecké umístění v průběhu závodu zlepšuje.

Nejlepší výsledky opět přinesl algoritmus kmeans. Při správném nastavení parametrů byly získány dobré výsledky i u algoritmu DBSCAN, ale je poměrně obtížné najít tyto správné parametry. Tyto parametry se navíc pro jednotlivé disciplíny mění u mužů i žen. Hierarchická metoda rozdělí závodníky také velmi solidně, ale opět si jsou objekty v jednotlivých třídách více podobné u kmeans.




Obrázek 5.6: Experiment IV.: Kmeans pro sprint a vybraná třída

## 5.3 Srovnání rychlosti střelby a úspěšnosti střelby

První dvě skupiny experimentů byly zaměřeny na porovnávání běžeckého času. V následujících experimentech jsem se soustředil na porovnání střeleckých výkonů. Další skupina experimentů byla provedena pro porovnání úspěšnosti střelby v souvislosti s její rychlostí. Cílem těchto experimentů bylo najít optimální čas střelby pro závodníka. Tento experiment může sloužit pro závodníky k určení optimální rychlosti jejich střelby. V této dolovací úloze je cílem určit, jestli závodník při kratších střeleckých časech dosahuje horších střeleckých výsledků a vyplatí se mu střílet, pomaleji nebo naopak, jestli při kratších střeleckých časech dosahuje stejných nebo lepších výsledků a biatlonistům se tak vyplatí střílet rychleji. Všechny experimenty byly provedeny ve třech variantách: pro střelbu komplexně a pro střelbu vleže a vestoje oddělene.

### 5.3.1 Experiment s porovnáním rychlosti a úspěšnosti střelby I.

#### Výběr dat

Při výběru dat byly vybrány data od závodníků, kteří odstříleli více než 40 střeleb celkově nebo 20 střeleb vleže a 20 střeleb vestoje. Pro každého ze závodníků je nejprve spočítán průměrný čas střelby. Střelba je následně rozdělena do 4 skupin podle času střelby. První skupina je pokud čas střelby je menší než 75% průměrného času střelby, druhá skupina je v intervalu 75-100% průměrného času střelby, třetí skupina je v invervalu 100-125% času střelby a v poslední skupině jsou střelby s časem větším než 125% průměrného času střelby. Pro každého závodníka je následně spočítána průměrná úspěšnost střelby. V dalším kroku je spočítána průměrná úspěšnost střelby v každé ze čtyř skupin. Od průměrné střelby v každé skupině je odečtena celková průměrná úspěšnost střelby. Takto upravená data jsou vstupem pro shlukovací algoritmy.

#### Zhodnocení experimentu

Při tomto experimentu se mi povedlo rozlišit závodníky do různých skupin. Prvním zjištěním bylo, že téměř všechny skupiny závodníků při delším střeleckém čase měli horší střelecké výsledky. Tato skutečnost, která je na první pohled překvapivá je nejspíše způsobena tím, že k delší střelbě dochází kvůli problémům při střelbě. Tyto problémy následně způdobí i nižší úspěšnost střelby. Z tohoto důvodu se těžko dají interpretovat výsledky související s touto skupinou střelby. U dvou středních typů střelby jsou výsledné třídy většinou podobné. Největší rozdíly vznikají pokud je střelba kratší než obvykle. U některých biatlonistů zůstává střelba velmi podobná jako u dalších skupin. Při shlukování, ale vznikly i třídy u kterých při nejkratším čase dochází k zhoršení střelby nebo naopak i k zlepšení střelby. Jedním z příkladů může být český biatlonista Mikuláš Karlík, který pokud střílí rychleji v leže, tak střílí s úspěšností o 30% méně než je jeho průměrný střelecký výkon. Naopak švýcarský biatlonista Serafin Wiestner při kratší střelbě střílí o 25% lépe než je jeho střelecký průměr 5.7.

Algortmis kmeans funguje dobře při nižším počtu tříd. Při větším počtu tříd se začnou jednotlivé třídy překrývat 5.7. U algoritmu DBSCAN je opět spousta objektů označeno za šum, ale vzniklé třídy jsou přehledné a objekty ve třídách jsou téměř totožné.

### 5.3.2 Experiment s porovnáním rychlosti a úspěšnosti střelby II.

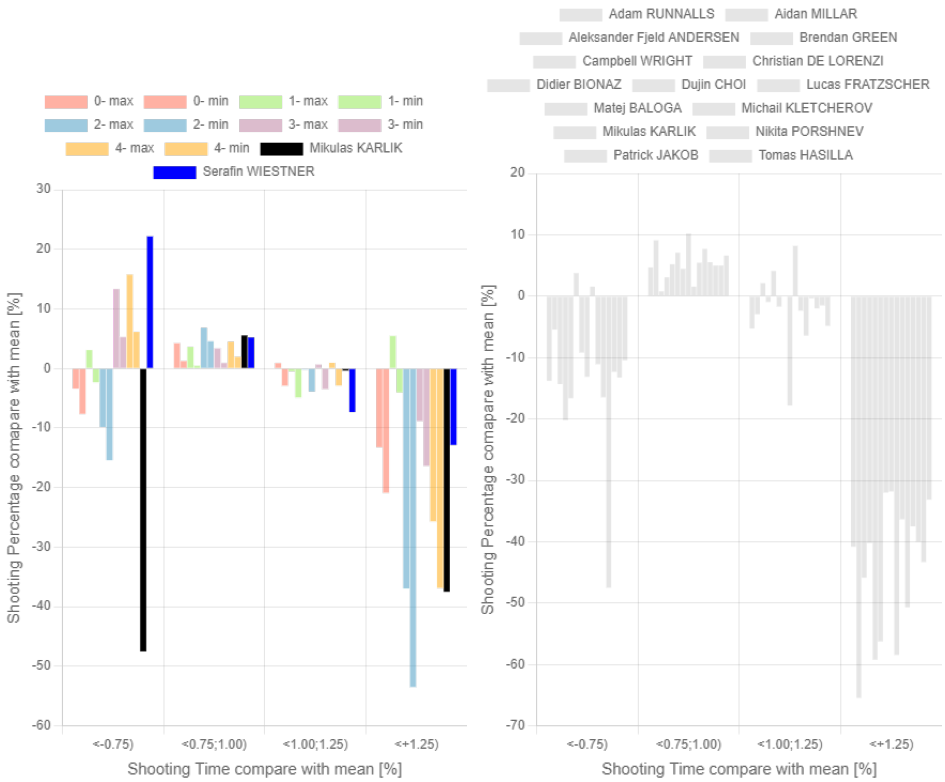
U střeleckých časů je problém s odlehlými hodnotami. Některé časy střelby jsou velmi dlouhé a ovlivní výpočet průměru střelby. Z tohoto důvodu jsem v tomto experimentu místo průměru střeleckého času využil medián.

#### Výběr dat

Výběr dat proběhl stejně jako u prvního experimentu jen u místo průměrného střeleckého času je vypočítán medián střeleckého času.

#### Zhodnocení experimentu

Při tomto experimentu jsem získal velmi podobné výsledky jako u prvního experimentu. Jelikož medián má nižší hodnotu než průměr u střeleckých časů, tak i střelby ze skupiny jsou s kratším časem. Při tomto experimentu jsou už skupiny, která dosahují lepších střeleckých

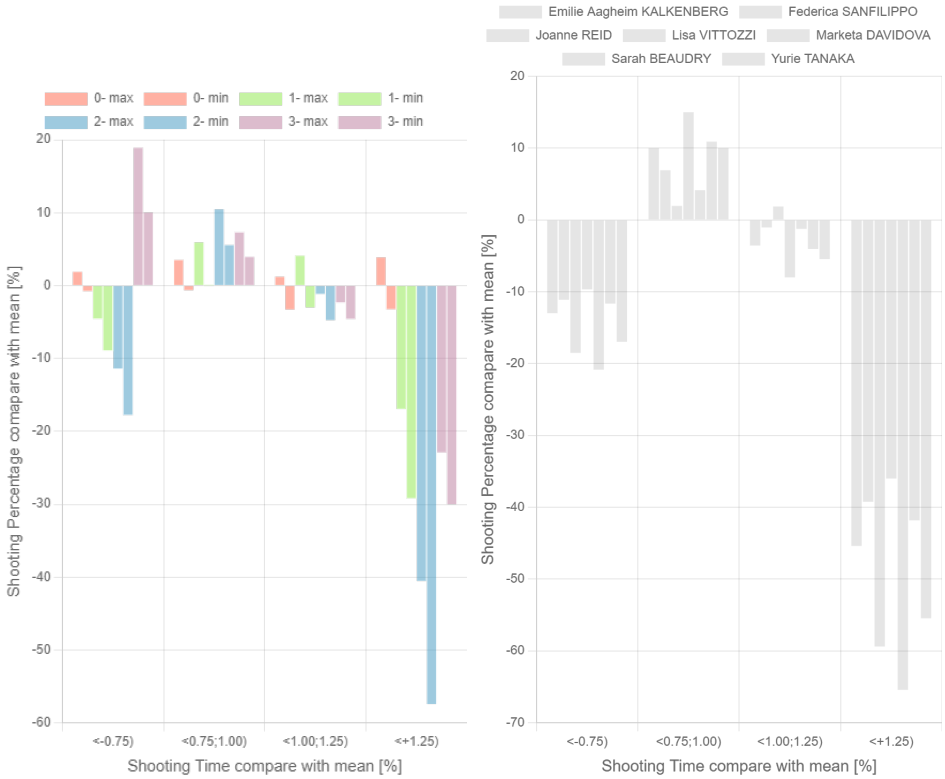



Obrázek 5.7: Experiment I.:Střelba v leže algortimus kmeans a jedna z tříd

výkonů při nejkratších střelbách menší. Většina biatlonistů se při kratším střeleckým čase dopouští více chyb, jak je vidět na druhém obrázku 5.8, kde je vidět jedna z výsledných tříd. Při tomto způsobu rozdělení fungoval velmi dobře i algortmus kmeans 5.8, který dokázal rozřadit biatlonisty a biatlonistky do zajimavých skupin. Hiearchické shlukování rozdělilo biatlonisty do tříd velmi dobře, ale jednotlivé objekty ve třídách jsou zašuměné a střelecká úspěšnost závodníků není dostatečně podobná.

## 5.4 Úspěšnost výstřelů při střelbě

V další dolovací úloze zaměřené na střelbu bylo mým cílem porovnáno úspěšnost jednotlivých výstřelů biatlonistů při střelbě. U tohoto experimentu je cílem zjistit, jestli při některé z ran dosahuje závodník lepší střelecké úspěšnosti, nebo naopak horší střelecké úspěšnosti. Očekával jsem především velké výkyvy při první nebo poslední vystřelené ráně. Tento experiment může biatlonistům pomoci zjistit, zda při některém z výstřelů mají dlouhodobě horší střeleckou úspěšnost, a zaměřit se na odstranění tohoto nedostatku, který zhoršuje jejich celkový střelecký výsledek. Tento experiment byl proveden pro střelbu vleže i vestoje dohromady a pro střelby vleže a vestoje zvlášť.




Obrázek 5.8: Experiment II.:Střelba v leže algortimus kmeans a jedna z tříd

### 5.4.1 Experiment s úspěšností výstřelů při střelběy I.

#### Výběr dat

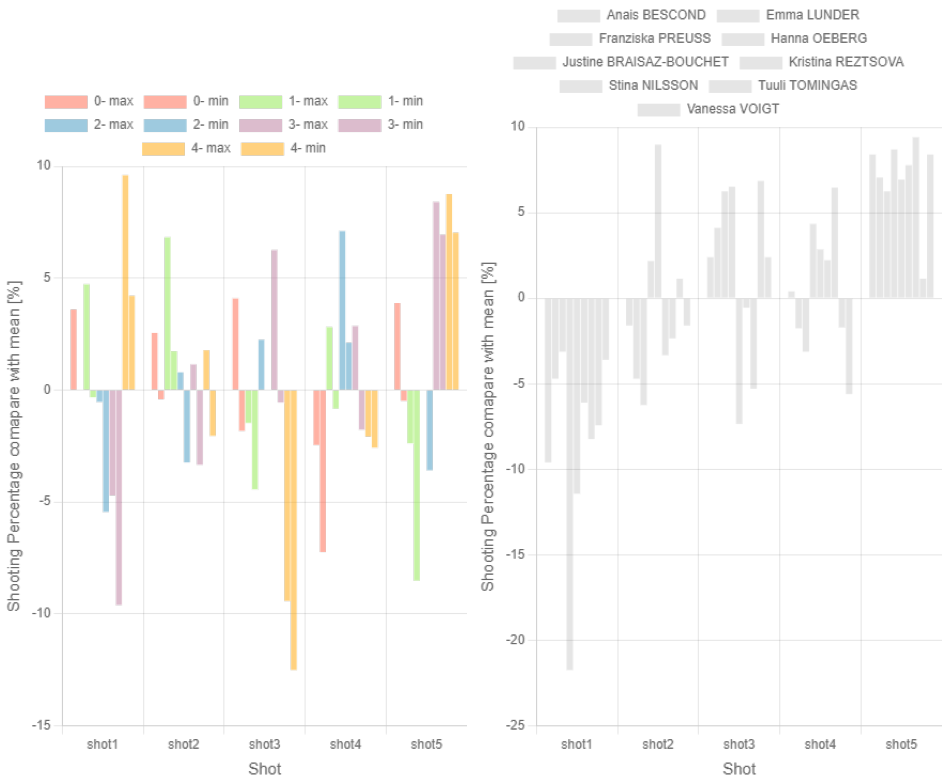
U každého závodníka, který se zúčastnil minimálně 15 závodů, jsem spočítal jeho celkovou průměrnou střeleckou úspěšnost. Následně jsem vypočítal průměrnou střeleckou úspěšnost při každém z výstřelů a od ní odečetl celkovou průměrnou střeleckou úspěšnost.

#### Zhodnocení experimentu

U tohoto experimentu došlo k velmi zajimavým rozdělením závodníků. Povedlo se vytvořit různorodé třídy a každá z tříd má různé vlasnosti. Zajimavostí je, že žádný z biatlonistů nemá všechny rány se stejnou nebo alespoň podobnou úspěšností. U všech tříd můžeme vidět u některé z ran výkyv do lepších nebo horších hodnot než je závodníkův průměr.

V rámci zkoumání výsledných tříd jsem nalezl některé velmi překvapivé vzory. Při vytváření experimentu jsem očekával velké výkyvy především při první a poslední ráně, ale některé třídy ukazují velmi nízkou úspěšnost u třetího výstřelu. Naopak podle očekávání se povedlo najít skupiny závodníků, kteří mají problém s první ránou. U třídy číslo 4 na obrázku 5.9 je tento výkyv do záporných hodnot nejmarkantnější. Například švédská biatlonistka Hanna Oeberg, která byla přidělena do této skupiny má při první ráně o 22 % nižší úspěšnost než je její průměr. V prediktivní analýze 6 jsem zjistil, že i jednotky procent hrají velkou roli v konečném výsledku. Naopak závodnice z třídy 5.9 mají nejhorší úspěšnost na posledním výstřelu. Pomocí tohoto experimentu mohou biatlonisté a biatlonistky snadno odhalit případnou slabinu ve střelbě a snažit se úpravou střelby tuto slabinu odstranit.

Nejlepší výsledky jsem opět získal pomocí algoritmu kmeans. Algoritmus kmeans i při nastavení většího počtu tříd dokázal závodníky rozdělit do dobře rozlišitelných tříd. Naopak u alogoritmu DBSCAN bylo opět složité nastavit parametry tak, aby se mi povedlo získat kvalitní výsledky. Při výběru některých dat funguje algortimus DBSCAN velmi dobře, naopak pro některé datasety se mi vůbec nepovedlo najít optimální nastavení. Velmi dobře fungovalo i hierarchické shlukování, pomocí kterého jsem získal podobné výsledky jako u algroritmu kmeans.




Obrázek 5.9: Experiment II.: Střelba vleže algortimus kmeans a jedna z tříd

## 5.5 Vyhodnocení shlukovacích experimentů

V rámci shlukovacích experimentů se povedlo najít neobvyklé a zajímavé vzory. Původně jsem se obával, že shluky budou vzájemně splývat a nebude možné je navzájem rozlišit. Při správné úpravě dat a nastavení vhodných parametrů pro shlukovací experimenty se mi povedlo těmto obavám vyhnout. Pro samotné biatlonisty mohou být zajimavé především výsledky ze střeleckých analýz. U střeleckých experimentů se mi podařilo získat nejzajímavější výsledky , kde vzniklé rozdíly mezi třídami byli i více než 10 % v úspěšnosti střelby.

Pro všechny experimenty jsem využil tři různé shlukovací metody. U každého z experimentů jsem se snažil tyto metody porovnat a jednoznačně nejlepší výsledky jsem získal pomocí algoritmu kmeans. Kmeans fungoval velmi dobře pro většinu z navržených experimentů a i při nastavení různého počtu tříd se mi povedlo získat kvalitní výsledky. Algoritmus DBSCAN bylo pro spoustu úloh obtížné nastavit. U některých úloh se mi správné nastavení tohoto algoritmu vůbec nepovedlo a výsledky shlukování byly nepoužitelné. Naopak

algoritmus DBSCAN dokázal velmi dobře najít téměř totožné objekty a odlišit je od ostatních. U některých dolovacích úloh mi to pomohlo najít zajímavé vzory. Pomocí hierarchické metody jsem získal často velmi podobné výsledky jako u algortimu kmeans. Při prvním pohledu na rozdělení tříd byly výsledky téměř nerozeznatelné. Větší rozdíly mezi kmeans a hierarchickou metodou jsem vždy odhalil až při detailním zkoumání jednotlivých tříd. U hiearchické metody se ve výsledných třídách častěji objevovali objekty, které do dané třídy nezapadaly.

Výsledky shlukovací analýzy hodnotím jako úspěšné z důvodu nalezení zajimavách informací, vzorů a rozdělení. Bohužel existuje málo shlukovacích analýz s biatlonovými statistikami, se kterými bych mohl srovnat své výsledky. Z jiných projektů jsem zkoušel moje výsledky srovnat například s projektem https://biathlonanalytics.com/ [4], který se v několika analýzach zaměřil na střelecké výkony závodníků. Tento projekt přinesl také srovnání úspěšnosti výstřelů při střelbě, ale jiným způsobem, a nelze tedy srovnat s mými výsledky shlukovací analýzy. Moje shlukovací experimenty můžou, tak sloužit jako nové doplnění těchto experimentů.

# Kapitola 6

# Prediktivní analýza

Kromě shlukovací analýzy jsem se zaměřil na predikitvní analýzu biatlonových výsledků. Mým cílem v rámci prediktivní analýzy bylo vytvořit model, který bude pro závody predikovat výsledky. Pro vytvoření modelu jsem zvolil vícenásobnou lineární regresi. Vícenásobnou lineární regresi jsem vybral z několika důvodů. Prvním důvodem bylo, že regrese dokáže dobře zpracovat numerické atributy se spojitou hodnotou. Dalším důvodem byla inspirace u prediktivních modelů v jiných sportech. Vícenásobná lineární regrese se používá pro predikci hokejových nebo baseballových výsledků. U hokeje se používá model pro predikci výsledků Tipsport Extraligy Českého hokeje nebo NHL [19].

## 6.1 Výběr dat

Data, která byla jako vstupní atributy pro vícenásobnou lineární regresi, jsem postupně měnil a upravoval tak, aby výsledná úspěšnost modelu byl co největší. Největší úspěšnost modelu, se mi povedlo získat, pokud do modelu vstoupily statistiky běžecké ztráty na 1km, střelecká úspěšnost, ztráta času střelby při jedné střelbě a úmístění v minulých závodech. Každý z těchto atributů byl vypočítán jako průměr ze statistik v minulých závodech. Pro střeleckou úspěšnost byly důležité především dlouhodobé statistiky a při přidání krátkodobých statistik dosahoval model horší úspěšnost. Naopak pro běžecké výkony bylo důležité přidat výsledky z posledních závodů.

## 6.2 Vytvoření modelu

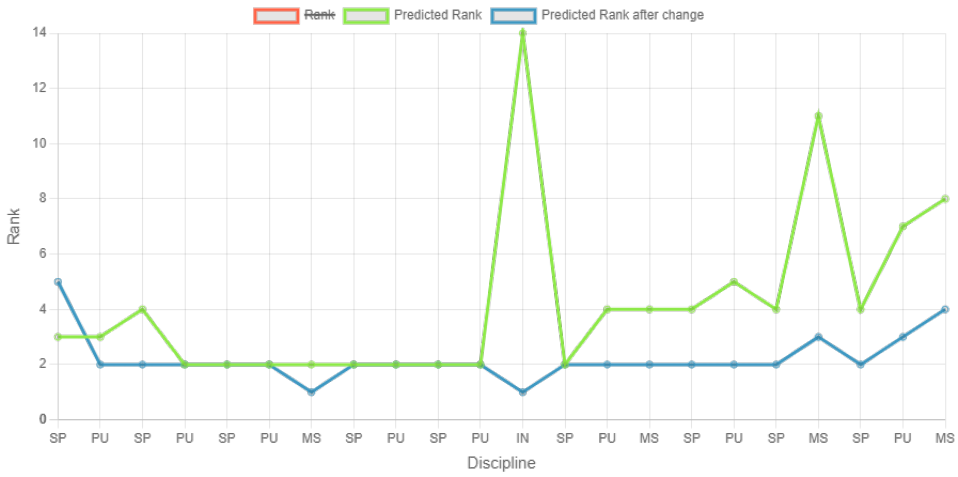
V databázi mám uložená data od sezóny 2015/2016. Pro vytvoření modelu byla použity data, od sezóny 2017/2018. Po stažení výsledků z posledních dvou sezón jsem měl v databázi dostatek výsledků a statistik pro vytvoření modelu. Při vytváření modelu byla nejprve vybrána data z databáze a upravena do tvaru tak, aby mohla být vstupem pro vícenásobnou lineární regresi. Data byla náhodně rozdělena v poměru 4:1 na data, která budou vstupem pro lineární regresi, a data, která budou sloužit k testování modelu. Model byl vytvořen pomocí funkce `fit()` ze třídy `LinearRegression` z knihovny `scikit-learn`. Model je neustále pro každý závod aktualizován a postupně se stává robustnější a zůstává aktuální.

## 6.3 Predikce výsledků

Při predikci výsledků je nejprve vytvořený model podle popisu výše. Následně je stažena startovní listina závodu. U každého ze závodníků jsou načteny jeho statistiky a upraveny pro vstup do vytrénovaného modelu. Závodníci jsou následně seřazeni podle lineárního modelu a data o jejich predikci jsou uložena do databáze.

## 6.4 Úprava predikce

Ve výsledné aplikaci existuje funkcionalita s možností zobrazit graf s predikovanými a skutečnými výsledky pro každého z biatlonistů. Kromě vizuální možnosti porovnání predikovaných a skutečných výsledků nabízí tento graf možnost upravit vstupní parametry pro vybraného biatlonistu a nově spočítat predikce. Každý z biatlonistů tak může vyzkoušet, jak se upraví jeho výsledky při zlepšení střelecké úspěšnosti nebo při zlepšení běžeckého času. Ve shlukovacích experimenetech 5 jsem zjistil, že biatlonistka Hanna Oeberg má problémy při první vystřelené ráně. Pokud by tento nedostatek dokázala odstranit, tak se její střelecká úspěšnost zlepší o 5 %. Na obrázku 6.1 je vidět, jak by se vylepšily výsledky Hanny Oeberg, pokud by dokázala nedostatek odstranit.




Obrázek 6.1: Upravená predikce Hanny Oeberg

## 6.5 Zhodnocení modelu

Vytvořený model pro predikci výsledků funguje dobře především pro závody s individuálním startem. Model dosahuje větší přesnosti u mužů než u žen. Výsledky ohodnocení modelu jsou v tabulce 6.1. Výsledný model dosahuje spíše průměrné úspěšnosti. Jedním z důvodů je nevyrovnanost výkonů biatlonistů. Velmi často se stává, že se biatlonistovi závod nepovede, nebo že během závodu fouká biatlonistovi na střelnici vítr. Tyto aspekty, nejsem schopnen zahrnout do prediktivního modelu. Další nevýhodou modelu je, že nezvládne kvalitně predikovat výsledky při porovnání dvou biatlonistů, pokud je jejich umístění predikováno blízko u sebe. Například pokud u mužského sprintu je rozdíl v predikovaném umístění menší než 3 místa, tak v závodě zvítězí lépe hodnocený biatlonista v 53 % případů. Druhý nedostatek by bylo možné odstranit za pomoci robustnější a přesnější metody než je vícenásobná lineární

regrese. Model naopak dosahuje velmi dobrých výsledků, pokud se rozdíl v predikovaných místech závodníků pohybuje ve větších jednotkách. Pokud zůstaneme u mužského sprintu, tak pokud je rozdíl v predikovaném umístení 10 - 14 míst, tak lépe hodnocený závodník zvítězí v 69% případů. Při větších rozdílech již dosahuje model kvalitních výsledků.

Při pokusu porovnat tento model s jinými již existujícími modely jsem objevil práci, která hodnotí, jestli lze predikovat výsledky v biatlonu ve světovém poháru podle výsledků v evropském poháru [17]. Tato práce neobsahuje konkrétní model a jeho hodnocení, ale potvrzuje, že biatlonový výsledek je více zavíslý na běžeckém výkonu než na střeleckém. Můj model také přiřadil větší důležitost běžeckému výkonu než střelecké úspěšnosti, a především střeleckému času. Výsledkem výše zmíněné studie je, že model může být využitý pro biatlonové trenéry, ale při jeho využití je potřeba být opatrný. Podobný závěr bych vyvodil i z mé práce. Pomocí modelu můžeme sledovat vývoj formy závodníků, porovnání hlavních rivalů a další informace ale vzhledem k nižší úspěšnosti se nemůžeme na model spolehnout s jistotou.

| Disciplína | M/Ž | Úspěšnost | Absolutní chyba | Střední kvad. chyba |
|---|---|---|---|---|
| Sprint | M | 57.17 | 16.12 | 397.56 |
| Stíhací závod | M | 57.99 | 8.65 | 115.54 |
| Inviduální závod | M | 48.26 | 18.02 | 493.21 |
| Závod s hromadným startem | M | 26.22 | 6.33 | 56.44 |
| Sprint | Ž | 52.97 | 15.99 | 391.47 |
| Stíhací závod | Ž | 47.37 | 9.66 | 136.42 |
| Inviduální závod | Ž | 46.27 | 16.55 | 423.6 |
| Závod s hromadným startem | Ž | 23.92 | 6.59 | 59.0 |

Tabulka 6.1: Hodnocení modelu

# Kapitola 7

# Závěr

V rámci této bakalářské práce jsem vytvořil webovou aplikaci, která stahuje data z oficiálního webu Mezinárodní biatlonové federace, data ukládá do databáze a na základě těchto dat provádí analýzu pomocí různých dolovacích úloh. Webová aplikace je dostupná na adrese https://analysisofbiathlonstatistics.herokuapp.com/.

V kapitole 2 jsem popsal různé možnosti jak přistoupit k analýze dat a popsal základní techniky pro získávání znalostí z dat. V kapitole 3 jsem se zabýval možnostmi jak získat biatlonová data, stahováním biatlonových dat a navržením databáze pro tato data. V kapitolách 5 a 6 jsem popsal řešení jednotlivých dolovacích úloh.

V rámci řešení práce trvalo delší dobu, než jsem očekával, stahování samotných dat. Při stahování dat jsem narážel na různé problémy. Největším problémem byla validace dat při zpracovávání PDF souborů. Nakonec se mi povedlo vytvořil obsáhlou databázi, která je naplněna výsledky ze závodů, statistikami a informacemi o závodech a biatlonistech a biatlonistkách.

V rámci získávání znalostí z dat jsem řešil dva typy dolovacích úloh. Prvním typem dolovacích úloh byly dolovací úlohy řešené za pomoci shlukování. Při řešení shlukovací analýzy se mi povedlo získat velmi zajímavé vzory v datech a informace, které mohou být užitečné pro analyzování výkonů biatlonistů. Shlukovací experimenty bych vyhodnotil jako úspěšné. Při případném rozšíření této bakalářské práce by bylo zajimavé se zaměřit na další shlukovací experimenty především u střeleckých výkonů biatlonistů, které by mohly přinést další zajimavé informace o střeleckých výkonech biatlonistů. Shlukovací experimenty jsem řešil pomocí různých algoritmů a nejlépe z použitých algoritmů fungoval algoritmus kmeans. Tento algortimus fungoval dobře pro všechny typy experimentů a doporučil bych jej pro případné řešení další dolovacích úloh.

V rámci prediktivní analýzy, která byla druhým typem dolovacích úloh, jsem vytvořil jednoduchý model, který využívá vícenásobnou lineární regresi pro predikci výsledků. U vícenásobné lineární regrese jsem narazil na omezení přesnosti pravděpodobnosti, kterého je tato metoda schopna dosáhnout. Výsledná úspěšnost modelu nedosahuje, takové výše, jakou bych očekával a jaké jsem chtěl dosáhnout. Pro přesnější předpověď výsledků by bylo potřeba implementovat predikci pomocí neuronové sítě backpropagation nebo jiné podobně robustní metody.

V rámci implementace jsem využil jazyk Python. Volbu jazyka Python bych ohodnotil jako velmi dobrou volbu pro řešení této bakalářské práce. Jazyk Python mi poskytl širokou podporu knihoven pro stahování dat, řešení dolovacích úloh i pro implementaci výsledné aplikace. Při řešení dolovacích úloh jsem využíval knihovnu scikit-learn, která mi poskytla funkce pro všechny dolovací techniky, které jsem použil. Jazyk Python mi usnadil řešení

všech částí mé bakalářské práce a doporučil bych jej pro použití především u řešení dolovacích úloh. Jedinou nevýhodou jazyka Pyton, na kterou jsem, narazil byla jeho rychlost, která byla omezující.

Pro pokračování tohoto projektu by bylo vhodné vytvořit prediktivní model, který bude pro predikci výsledků používat jinou metodu, než je vícenásobná lineární regrese, a srovná tuto metodu právě s vícenásobnou lineární regresí. Tato práce by mohla být dále rozšířená o další shlukovací experimenty, které budou zkoumat výsledky biatonistů a jejich statistiky a najít další nové zajimavé vzory. Dolovací úlohy se zaměřily pouze na individuální výkony biatlonistů a nezkoumaly výsledky ze štafetových závodů. Další rozšíření tohoto projektu je možné v oblasti vizualizace stažených dat ve webové aplikaci. Tato práce má spoustu možností pro další rozšíření a rozvinutí výsledné webové aplikace, která se tak může stát zajímavější a hodnotnější.

# Literatura


[1] *What is Entity Relationship Diagram (ERD)?* [online]. 2019 [cit. 2022-04-02]. Dostupné z: https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/.

[2] *Asynchronous JavaScript Technology and XML (Ajax) With the Java Platform* [online]. 2022 [cit. 2022-04-27]. Dostupné z: https://www.w3schools.com/js/js_ajax_intro.asp.

[3] *Beautiful Soup Documentation* [online]. 2022 [cit. 2022-03-25]. Dostupné z: https://www.crummy.com/software/BeautifulSoup/bs4/doc/.

[4] *Biathlon Analytics Home Page* [online]. 2022 [cit. 2022-05-01]. Dostupné z: https://biathlonanalytics.com/.

[5] *Chart.js* [online]. 2022 [cit. 2022-04-26]. Dostupné z: https://www.chartjs.org/docs/latest/.

[6] *Difference between MVC and MVT design patterns* [online]. 2022 [cit. 2022-04-17]. Dostupné z: https://www.geeksforgeeks.org/difference-between-mvc-and-mvt-design-patterns//.

[7] *Django documentation* [online]. 2022 [cit. 2022-04-17]. Dostupné z: https://docs.djangoproject.com/en/4.0/.

[8] *Django Project MVT Structure* [online]. 2022 [cit. 2022-04-17]. Dostupné z: https://www.geeksforgeeks.org/django-project-mvt-structure/.

[9] *NumPy documentation* [online]. 2022 [cit. 2022-04-15]. Dostupné z: https://numpy.org/doc/stable/.

[10] *Pandas documentation* [online]. 2022 [cit. 2022-04-15]. Dostupné z: https://pandas.pydata.org/docs/.

[11] *Pdfplumber* [online]. 2022 [cit. 2022-04-15]. Dostupné z: https://gitee.com/hailei_yan/pdfplumber/.

[12] *Python (programming language)* [online]. 2022 [cit. 2022-04-26]. Dostupné z: https://www.w3schools.com/js/js_ajax_intro.asp.

[13] *Re — Regular expression operations* [online]. 2022 [cit. 2022-04-14]. Dostupné z: https://docs.python.org/3/library/re.html.

[14] *Requests: HTTP for Humans* [online]. 2022 [cit. 2022-04-07]. Dostupné z: https://docs.python-requests.org/en/latest/.

[15] *Scikit-learn Machine Learning in Python* [online]. 2022 [cit. 2022-04-02]. Dostupné z: https://scikit-learn.org/stable/getting_started.html.

[16] *The Selenium Browser Automation Project* [online]. 2022 [cit. 2022-04-16]. Dostupné z: https://www.selenium.dev/documentation/.

[17] Dzhilkibaeva, N., Ahrens, M. a Laaksonen, M. S. *Can performance in biathlon world cup be predicted by performance analysis of biathlon IBU cup?* [online]. 2019 [cit. 2022-05-02]. Dostupné z: https://www.tandfonline.com/doi/full/10.1080/24748668.2019.1665884.

[18] Fordham, S. *5 Top Tips for Data Scraping Using Selenium in Python* [online]. 2022 [cit. 2022-04-16]. Dostupné z: https://towardsdatascience.com/5-top-tips-for-data-scraping-using-selenium-d8b83804681c.

[19] Hudec, O. *Hokej v číslech: Úvod do Game Score. Jak měřit výkony hráčů v zápasech* [online]. 2021 [cit. 2022-04-15]. Dostupné z: https://www.hokej.cz/hokej-v-cislech-uvod-do-game-score-jak-merit-vykony-hracu-v-zapasech/5055065.

[20] HAN, J., KAMBER, M. a PEI, J. *Data mining: concepts and techniques.* 3. vyd. Elsevier, 2012. ISBN 978-0-12-381479-1.

[21] Layton, R. *Learning Data Mining with Python.* 1. vyd. Packt Publishing Ltd., 2015. ISBN 978-1-78439-605-3.

[22] Mitchell, R. *Web Scraping with Python.* 2. vyd. O'Reilly Media, 2018. ISBN 978-1-4919-8557-1.

[23] Slonneger, K. *Introduction to XML* [online]. 2006 [cit. 2022-04-10]. Dostupné z: http://homepage.divms.uiowa.edu/~slonnegr/xml/01.IntroXML.pdf.

[24] Zendulka, J. *Získávání znalostí z databází ZZN Studijní opora* [online]. 2009 [cit. 2022-01-18]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FZZN-IT%2Ftexts%2FZZN.pdf.

[25] Zendulka, J. a Bartík, V. *Relační model dat* [online]. 2022 [cit. 2022-04-02]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FIDS-IT%2Flectures%2Fcz%2F3_relmod.pdf&cid=13974.

# Seznam obrázků


3.1 ER Diagram pro návrh databáze . . . . . 16
3.2 Report s výsledky závodu . . . . . 20
3.3 Stahování dat ve výsledné aplikaci . . . . . 21

5.1 Experiment I.: Rozdělení pomocí algoritmu kmeans a hierarchické metody . 32
5.2 Experiment II.: Rozdělení podle algoritmu DBSCAN a jedna z tříd . . . . . 34
5.3 Experiment IV.: Rozdělení podle algoritmu kmeans a DBSCAN . . . . . 35
5.4 Experiment I.: Rozdělení Kmeans pro závod s hromadným startem a sprint 36
5.5 Experiment II.: Kmeans pro sprint a vybraná třída . . . . . 38
5.6 Experiment IV.: Kmeans pro sprint a vybraná třída . . . . . 39
5.7 Experiment I.:Střelba v leže algortimus kmeans a jedna z tříd . . . . . 41
5.8 Experiment II.:Střelba v leže algortimus kmeans a jedna z tříd . . . . . 42
5.9 Experiment II.: Střelba vleže algortimus kmeans a jedna z tříd . . . . . 43

6.1 Upravená predikce Hanny Oeberg . . . . . 46

# Příloha A

# Obsah přiloženého média

```
CD
├── WebApp - zdrojové kódy
├── latexsrc - zdrojové kódy pro technickou zprávu
├── xzeman63-tz.pdf - technická zpráva
└── readme.md - návod na instalaci
```