VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

VZTAH ZMĚN SÁZKOVÝCH KURZŮ A VÝSLEDKŮ FOTBALOVÝCH ZÁPASŮ

RELATIONSHIP BETWEEN CHANGES IN BETTING ODDS AND RESULTS OF FOOTBALL MATCHES

BAKALÁRSKA PRÁCA
BACHELOR'S THESIS

AUTOR PRÁCE JURAJ JURKOVIČ
AUTHOR

VEDOUCÍ PRÁCE doc. Ing. JAROSLAV ZENDULKA, CSc.
SUPERVISOR

BRNO 2019

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání bakalářské práce

22232

Student: **Jurkovič Juraj**
Program: Informační technologie
Název: **Vztah změn sázkových kurzů a výsledků fotbalových zápasů**
Relationship between Changes in Betting Odds and Results of Football Matches
Kategorie: Data mining
Zadání:

1. Seznamte se se základy získávání znalostí z dat (data mining) a získáváním strukturovaných dat z webu (web scraping).
2. Seznamte se rámcově s dostupnými prostředky na podporu získávání znalostí z dat a strukturovaných dat z webu v jazyce Python.
3. Po dohodě s vedoucím zvolte vhodnou případovou studii zaměřenou na analýzu vztahu změn sázkových kurzů a výsledků fotbalových zápasů.
4. Po dohodě s vedoucím specifikujte přesněji zvolenou analytickou úlohu a podrobněji se seznamte s přístupy a algoritmy pro její řešení, včetně podpory dostupné pro jazyk Python.
5. Řešení úlohy naprogramujte a pro dostupná data vyhodnoťte.
6. Zhodnoťte dosažené výsledky a posuďte jejich praktické využití.

Literatura:

- Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Third Edition. Morgan Kaufmann Publishers, 2012, 703 p., ISBN 978-0-12-381479-1. (Kapitola 1 a další informace relevantní ke zvolené úloze a použitým datům).
- Vettigli, G.: Practical Data Mining with Python. Dostupné na http://guidetodatamining.com/.
- What is web scraping - Part 1 - Beginner's guide. Dostupné na https://www.scrapehero.com/a-beginners-guide-to-web-scraping-part-1-the-basics/.
- Practical Introduction to Web Scraping in Python. Dostupné nahttps://realpython.com/python-web-scraping-practical-introduction/.
- KDnuggets Datasets for Data Mining and Data Science. Dostupné na https://www.kdnuggets.com/datasets/index.html.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Zendulka Jaroslav, doc. Ing., CSc.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 3. prosince 2018

Abstrakt

Cieľom tejto práce je demonštrovať techniky pre riešenie úloh extrakcie dát z webu a získavania znalostí z dát. Prípadová štúdia je zameraná na zber dát zo stávkových kancelárií a následnú analýzu týchto dát. Práca demonštruje riešenie úlohy získavania dát z webu v jazyku Python. Práca popisuje vybrané implementačné detaily pre vytvorenie systému na zber dát z webu a navrhuje schému databázy, ktorá môže byť v tomto systéme použitá. Získané dáta sú analyzované štatistickými metódami a vo vývojoch cien kurzov sú objavované frekventované vzory pomocou apriori algoritmu. Pozorované vzťahy a objavené frekventované vzory sú prezentované koncovému užívateľovi.

Abstract

The goal of this thesis is to demonstrate techniques for solving web scraping and knowledge discovery tasks. The case study is focused on the extraction of data from bookmaker websites and subsequent analysis of collected data. The thesis demonstrates the implementation of web scraping task in Python language. The thesis describes selected implementation details for developing such a system and proposes a database schema that can be used for this purpose. Collected data is analyzed using statistical methods and frequent patterns are discovered in odds movements using apriori algorithm. Discovered relationships and frequent patterns are presented to the end user.

Kľúčové slová

získavanie dát z webu, data mining, získavanie informácií z databáz, kurzové stávkovanie, analýza časových rád, získavanie frekventovaných vzorov, analýza vzorov

Keywords

web scraping, data extraction, data mining, data collection, knowledge discovery in databases, sports betting, time series analysis, pattern analysis, frequent pattern mining

Citácia

JURKOVIČ, Juraj. *Vztah změn sázkových kurzů a výsledků fotbalových zápasů.* Brno, 2019. Bakalárska práca. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Jaroslav Zendulka, CSc.

Vztah změn sázkových kurzů a výsledků fotbalových zápasů

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením doc. Jaroslava Zendulku. Uviedol som všetky literárne zdroje a publikácie, z ktorých som čerpal.

.....
Juraj Jurkovič
15. mája 2019

Poďakovanie

Ďakujem vedúcemu práce doc. Zendulkovi za odborné rady a usmernenie pri vypracovávaní tejto práce, a za jeho ochotu.

Obsah

1 O čom je táto práca **3**

2 Úvod do problematiky **4**
2.1 Dolovanie a získavanie znalostí z dát 4
2.1.1 Delenie metód dolovania dát 7
2.1.2 Podpora v jazyku Python 8
2.2 Získavanie dát z webu 9
2.2.1 Časté problémy 9
2.2.2 Podpora v jazyku Python 10

3 Prípadová štúdia - návrh **15**
3.1 Úvod do kurzového stávkovania 15
3.1.1 Druhy stávok 15
3.1.2 Formáty kurzov 17
3.1.3 Marže stávkových kancelárií a pravdepodobnosť 18
3.2 Získavanie dát z webu 18
3.2.1 Analýza dostupných dát 18
3.2.2 Návrh databázy 20
3.2.3 Analýza backendu webstránky 21
3.2.4 Získavanie výsledkov stávok 27
3.2.5 Rozlišovanie totožných zápasov z rôznych zdrojov 30
3.3 Získavanie znalostí a analýza nazbieraných dát 30
3.3.1 Predspracovanie a čistenie 31
3.3.2 Transformácia 31
3.3.3 Data mining 34

4 Implementácia navrhnutého riešenia **37**
4.1 WebScraping 37
4.1.1 Framework 37
4.1.2 Získavanie dát 39
4.1.3 Správa a plánovanie procesov 41

5 Analýza nazbieraných dát **42**
5.1 Frekventované vzory 42
5.2 Celková zmena kurzu 43

6 Záver **45**

Literatúra **46**

A Obsah priloženého média **49**

Kapitola 1

O čom je táto práca

Cieľom tejto práce je preskúmať a demonštrovať riešenie úloh pre analýzu a získavanie znalostí z dát (data mining) a úloh pre extrakciu dát z webu (web scraping) pre účely tejto analýzy. Budú popísané aplikovateľné postupy pre zber dát z webových stránok, ako aj rôzne problémy, ktoré pri tomto procese môžu nastať, a ich riešenia. V kapitole 2 budú spomenuté niektoré často používané knižnice jazyka Python, ktoré je pre riešenie takýchto úloh možné využiť.

Prípadová štúdia bude zameraná na zber dát o kurzovom stávkovaní z webových stránok stávkových kancelárii a následnú analýzu a extrakciu znalostí z týchto dát. V kapitole 3 bude opísaný návrh programu pre zber dát v ponuke stávkovej kancelárie *Niké*. Návrh bude zameraný na identifikáciu štruktúry webovej stránky, ktorá je použitá ako zdroj dát a návrh procesu pre získavanie potrebných dát z tejto stránky. Taktiež bude vytvorený návrh schémy databázy pre ukladanie týchto dát.

V kapitole 4 budú následne demonštrované konkrétne riešenia v jazyku Python. Bude opísaný mikro-framework implementovaný pre účely zberu dát zo stávkových kancelárií a vysvetlené rôzne optimalizácie pre zníženie počtu dotazov, ako aj odchýlky od návrhu, ktoré majú za cieľ urýchliť následnú pracú s dátami.

Na zhromaždených historických vývojoch cien kurzov budú demonštrované možné techniky analýzy časových postupností, ktoré reprezentujú vývoje cien kurzov pre jednotlivé stávky. Na týchto dátach bude demonštrovaný algoritmus pre frekventované vzory a závislosti medzi týmito vzormi, ako aj medzi celkovou zmenou kurzu počiatočného zverejnenia a výsledkami pozorovaných stávok.

Kapitola 2

Úvod do problematiky

2.1 Dolovanie a získavanie znalostí z dát

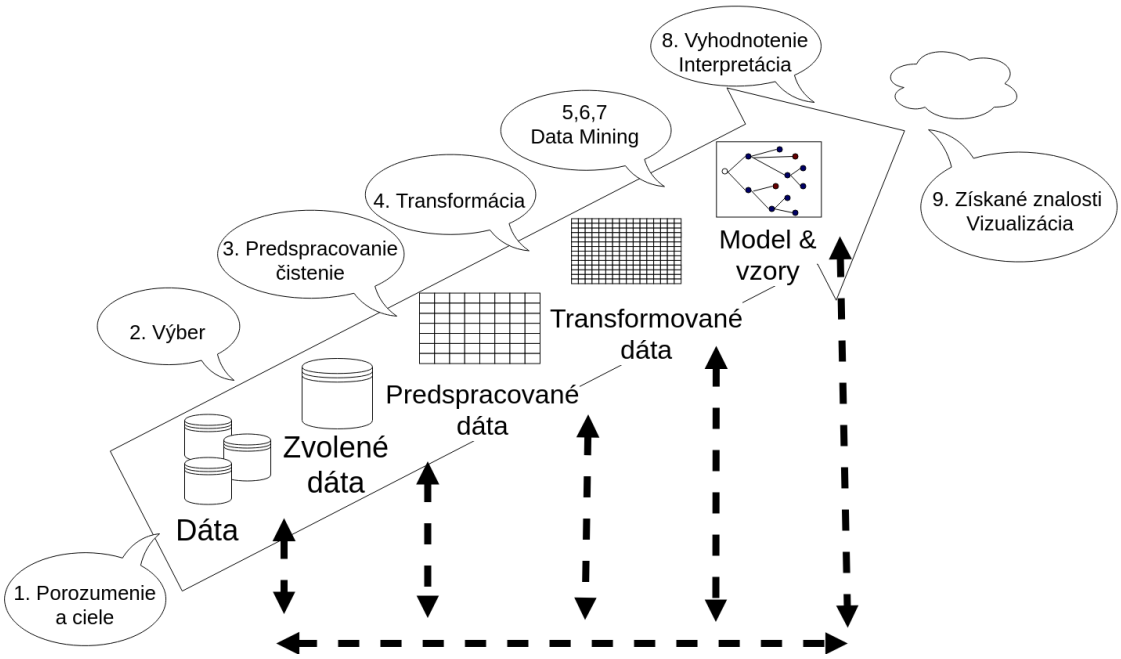
Dolovanie dát (data mining) a získavanie znalostí z dát (knowledge discovery in databases, tiež KDD) sú procesy, ktoré spracovávajú a transformujú rozsiahle kolekcie dát a premieňajú ich na znalosti a informácie. Tieto pojmy sa často zamieňajú alebo sa kolektívne označujú spoločným pojmom *data mining*. Jedná sa však o mierne odlišné procesy. Dolovanie dát je súčasťou KDD a jeho matematickým jadrom. Jeho úlohou je odvodiť algoritmy pre preskúmanie databáze, definovať matematický model a objaviť významné vzory v dátach. Toto následne predstavuje samotné získane znalostí[12].

Schopnosť získať užitočné informácie z dát sa stáva čoraz dôležitejšou. Vo svete, kde sa generuje na dennej báze obrovské množstvo dát, je potrebné vedieť tieto dáta efektívne analyzovať. Firmy, jednotlivci, štátne organizácie a vedecko-výskumné inštitúcie denne generujú veľké množstvo dát. Tieto dáta môžu mať najrôznejšiu podobu. Môže sa jednať o informácie o predajoch, vývoje cien na burze, pohyby kurzov v stávkových kanceláriách, zamestnanecké profily na profesijných sociálnych sieťach, informácie o produktoch v internetových obchodoch alebo multimédiá ako videá, hudba, či fotky. Zoznam zdrojov, ktoré dnes vytvárajú na dennej báze množstvo nových dát, je nespočetný.

V súčasnosti je pristúp k týmto dátam jednoduchší ako kedykoľvek predtým. Odhaduje sa, že množstvo dostupných dát sa zdvojnásobí približne každé dva roky[34]. Bohužiaľ, množstvo a rozsah týchto dát ďaleko presahuje to, čo sme ako ľudia schopní spracovať a porozumieť. Výsledkom je, že tieto obrovské kolekcie dát sú len veľmi zriedka opätovne použité a ostávajú nedotknuté ležať v archívoch. Súčasná situácia sa často označuje ako bohatá na dáta, ale chudobná na informácie. Vzhľadom na túto neschopnosť držať krok s množstvom generovaných dát sú často dôležité rozhodnutia prijímané nie na základe týchto rozsiahlych dát bohatých na informácie, ale na základe ľudskej intuície doménových expertov. Toto je bohužiaľ veľmi časovo neefektívny a nákladný proces. Tento zväčšujúci sa rozdiel medzi množstvom dostupných dát a ľudskou schopnosťou týmto dátam porozumieť je hlavnou motiváciou pre vývoj nástrojov, ktoré tieto zabudnuté dátové archívy premenia na cenné informácie[9].

Samotný proces získavania znalostí z dát je iteratívny a interaktívny. Kroky tohto procesu sú znázornené na obrázku 2.1. Vidíme, že dolovanie dát je len jednou zo súčastí celého procesu získavania informácií. Treba podotknúť, že proces je iteratívny v každom z týchto krokov, čo znamená, že v ktoromkoľvek kroku môže byť nutné zmeniť predchádzajúci krok na základe novonadobudnutých znalostí. Samotný proces má mnoho prvkov, ktoré spoliehajú na intuíciu osoby, ktorá ho navrhuje v tom zmysle, že neexistuje všeobecný postup,

ktorý by vytvoril optimálny proces pre ľubovolný typ aplikácie a doménu. Z toho dôvodu je nutné porozumieť jednotlivým krokom a tomu, aké dopady majú na celkový výsledok.[12]



Obr. 2.1: Proces získavania znalostí z dát

1. Vývoj a porozumenie aplikačnej doméne. Toto je prvý krok prípravy, v ktorom je nutné porozumieť aké rozhodnutia bude potrebné učiniť v ďalších krokoch. Je nutné definovať ciele procesu získavania informácií a prostredie a kontext, v ktorom budu dáta prezentované koncovému užívateľovi. Po ustanovení cieľov je možné prejsť k predspracovaniu dát tak, ako bude opísané v nasledujúcich krokoch.

2. Voľba a vytvorenie dátovej množiny, z ktorej budeme informácie získavať. Po tom, čo sme definovali ciele je potrebné ustanoviť, aké dáta použijeme pre dolovanie znalostí. Toto zahŕňa prieskum toho, aké dáta sú nám dostupné, prípadne doplnenie potrebných dát z iných zdrojov a následnú integráciu do jednotnej dátovej množiny. V prípade, že niektoré dôležité atribúty budu chýbať, môže sa to negatívne odraziť na samotnom výsledku. V tomto bode je preto vhodné uvažovať najvyšší možný dostupný počet atribútov. Na druhej strane, s narastajúcim počtom atribútov sa zvyšujú aj režijné náklady. Ide tak o iteratívnu činnosť, pričom k tejto fáze môže byť potrebné vrátiť sa znova po nadobudnutí nových poznatkov o požiadavkách systému.

3. Predspracovanie a čistenie. V tejto fáze sa zvyšuje spoľahlivosť dát. Tato fáza zahŕňa kroky čistenia, ako napríklad odstránenie nekompletných dát alebo ošetrenie chýbajúcich hodnôt. V prípade kurzového stávkovania by sa jednalo napríklad o zápasy, pre ktoré nepoznáme výsledky alebo ponuky, pre ktoré sa nám nepodarilo zachytiť celý vývoj cien kurzov. Spôsob do akej miery musíme v tomto kroku do dát zasahovať môže sám o sebe prezrádzať zaujímavé informácie.

4. Transformácia dát. V tomto kroku je cieľom pripraviť dáta pre data mining algoritmus. Metódy v tomto kroku môžu zahŕňať redukciu dimenzií (výber konkrétnych atribútov

alebo vzorkovanie), transformáciu atribútov (napr. diskretizácia) a agregačné operácie. Tento krok je veľmi dôležitý pre úspešné dolovanie, no aj pokiaľ nezvolíme správne transformácie môžeme získať znalosť o tom, aké transformácie sú potrebné, prípadne ako ich zmeniť. Samotný proces dolovania teda pomáha iteratívne získať informácie o tom, ako tento proces upraviť.

5. Voľba dolovacej úlohy. V tomto kroku sa rozhodujeme, aký druh dolovania dát použijeme. Toto závisí predovšetkým na cieľoch, ktoré sme si stanovili. Postupy dolovania dát sa delia na dve hlavné skupiny: deskriptívne a prediktívne. Bližšie sú popísane v sekcii 2.1.1.

6. Voľba algoritmu. V tomto kroku volíme konkrétnu metódu pre vyhľadávanie vzorov. Pre každú zo stratégií zvolených v predchádzajúcich krokoch existuje niekoľko spôsobov, ako docieliť požadované výsledky. Pri tomto prístupe sa snažíme porozumieť, ktorý algoritmus použiť a prečo je najvhodnejší. Každý algoritmus má parametre a taktiky učenia, ktoré je nutné zohľadniť.

7. Aplikácia dolovacieho algoritmu. Tento krok môže byť nutné opakovať niekoľkokrát, pričom algoritmus alebo jeho parametre upravujeme, kým nedosiahneme uspokojivý výsledok.

8. Vyhodnotenie. V tomto kroku získané vzory interpretujeme s ohľadom na ciele stanovené v prvom kroku. Tento krok je zameraný na použiteľnosť získaného modelu. Spätnú väzbu na získaný model získame v poslednom kroku.

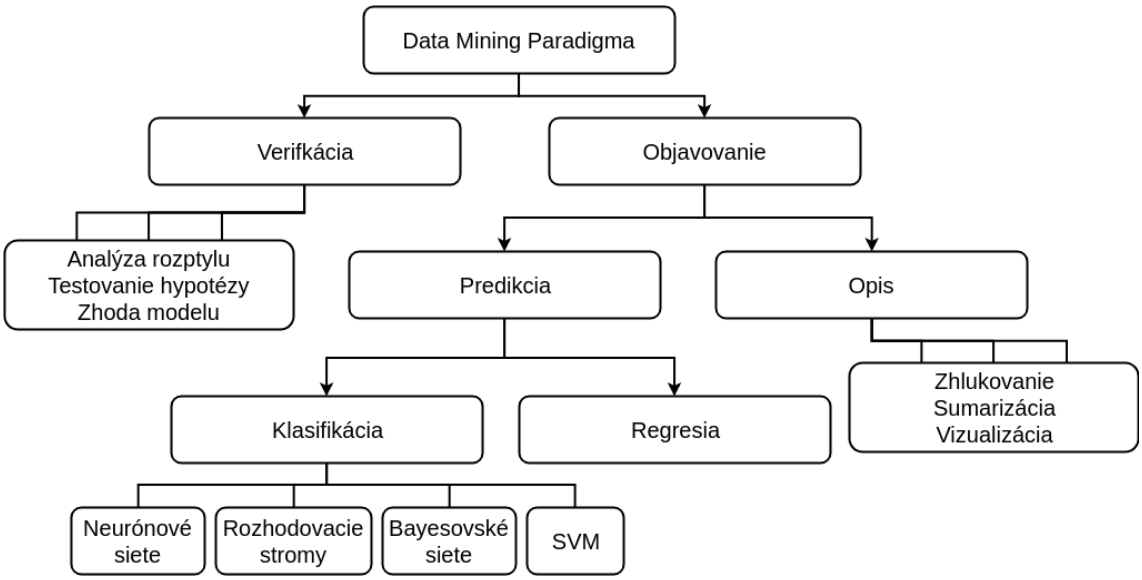
9. Použitie získaných znalostí. V tomto kroku sme pripravení poskytnúť získané znalosti inému systému pre ďalšie použitie. Znalosti môžu byť využité napríklad tak, že na základe nich vykonáme zmeny v inom systéme a budeme pozorovať ako tento systém zareaguje. V tomto kroku sa prejaví samotná účinnosť celého procesu. Hlavnou výzvou je fakt, že znalosti budú aplikované na reálne dáta, ktoré sa môžu správať odlišne ako sa správali v nami kontrolovaných podmienkach. Príkladom môže byť nedostatočne reprezentatívna vzorka, z ktorej sme vedomosti získavali.

Znalosti je možné získavať z najrôznejších zdrojov dát. Môže sa jednať o databázy, dátové sklady alebo aj o prúdy dát, ktoré prichádzajú do systému. Vo všeobecnosti sa dá povedať, že informácie vieme získavať z akéhokoľvek druhu dát pokiaľ dáta majú určitý význam pre cieľovú aplikáciu. Z týchto dát je možné získavať rôzne poznatky. Môže ísť napríklad o analýzu trendov v návštevnosti, alebo veľkosti objednávok v prevádzke rýchleho občerstvenia, ktorá na základe týchto znalostí môže prispôsobiť smeny zamestnancov. Sledovaním vývojov cien akcií na burze môže odhaliť vzory, ktoré môžu pomôcť pri vývoji investičnej stratégie. Vznikajú firmy, ktoré pomocou dolovania znalostí z biometrických dát (napríklad rýchlosť písania na klávesnici, pohyby myšou a pod.) a informácií o užívateľských zariadeniach vytvárajú bezpečnostné systémy pre detekciu neoprávnených užívateľov alebo detekciu odcudzenia prístupových údajov. Väčšina internetových obchodov v dnešnej dobe sleduje, ktoré produkty zákazníci nakupujú a na základe toho im odporúčajú produkty, ktoré by si potenciálne tiež mohli kúpiť. Robia tak na základe analýzy množstva predošlých transakcií, pričom sledujú, ktoré položky boli zakúpené spolu.

Aplikácií pre uplatnenie techník získavania znalostí z dát je skutočne nespočetne mnoho a s narastajúcim množstvom dostupných dát budú len pribúdať.[9]

2.1.1 Delenie metód dolovania dát

Existuje mnoho spôsobov a metód dolovania dát, pričom tieto sú vhodné pre rôzne účely a ciele. Dolovanie dát delíme na dva hlavné druhy: zamerané na verifikáciu, kde systém verifikuje predloženú hypotézu, a zamerané na objavovanie, kde systém hľadá vzory a nové pravidlá. Rozdelenie metód dolovania dát je znázornené na obrázku 2.2.



Obr. 2.2: Delenie metód dolovania dát

Objavovacie metódy sú také, ktoré automaticky identifikujú vzory v dátach. Táto skupina metód sa ďalej delí na prediktívne a deskriptívne metódy.

Deskriptívne metódy sú zamerané na interpretáciu dát, ktorá má za cieľ porozumenie (napríklad pomocou vizualizácie) tomu, aké sú vzťahy medzi dátami.

Prediktívne metódy majú za cieľ zostrojiť behaviorálny model, ktorý bude získavať nové a predtým nepozorované vzorky a je schopný predpovedať hodnoty jedného alebo viacerých atribútov tejto vzorky. Niektoré prediktívne metódy môžu taktiež pomôcť porozumieť dátam. Väčšina objavovacích metód je založená na logickej indukcii, kde je model zostrojený na základe zobecňovania dostatočného množstva trénovacích vzoriek. Základným predpokladom tohto induktívneho prístupu je, že získaný model je aplikovateľný na nové vzorky.

Na druhej strane sa verifikačné metódy snažia vyhodnocovať predloženú hypotézu. Toto zahŕňa tradičné štatistické metódy. Tieto metódy sa menej často spájajú s dolovaním dát, nakoľko väčšina úloh dolovania dát má za úlohu objaviť hypotézu a nie dokázanie konkrétnej hypotézy.

Prediktívne metódy sú často označované ako metódy s učiteľom, zatiaľ čo deskriptívne metódy sa niekedy označujú ako metódy bez učiteľa. Učenie bez učiteľa označuje modelovanie distribúcie jednotlivých položiek vo vstupnom priestore a väčšinou predstavuje techniky, ktoré zoskupujú inštancie bez predom definovaného závislého atribútu. Metódy s učiteľom sa pokúšajú objaviť vzťahy medzi vstupnými atribútmi (nezávislými premennými) a cieľovými atribútmi (závislými premennými). Tento vzťah je reprezentovaný štruktúrou, ktorú nazývame model. Model zvyčajne popisuje a vysvetľuje určitý jav, ktorý sa v dátovej sade nachádza a môže byť použitý pre predvídanie hodnoty závislej premennej na základe nezávislých premenných.

Dva hlavné druhy prediktívnych modelov, ktoré rozlišujeme sú klasifikácia a regresia. Regresia prevádza vstupný priestor na obor reálnych hodnôt. Regresor napríklad môže predpovedať dopyt po istom produkte na základe jeho vlastností. Na druhej strane klasifikátory prevádzajú vstupný priestor na predom definované triedy. Typickými zástupcami klasifikátorov sú: rozhodovacie stromy, metódy podporných vektorov (support vector machines, tiež SVMs) a algebraické funkcie. [12]

2.1.2 Podpora v jazyku Python

Jazyk Python ponúka veľké množstvo knižníc, ktoré je možné použiť pre účely získavania informácií z dát a data miningu. Jedná sa o knižnice pre prácu s veľkými dátovými sadami, ktoré definujú dátové typy reprezentujúce rôzne typy dátových množín, ponúkajú rôzne štatistické metódy a metódy pre transformácie dát, knižnice pre strojové učenie, ktoré môžu byť použité pri prediktívnych úlohách, a knižnice pre vizualizáciu dát pomocou rôznych typov grafov. V kontexte tejto práce stoja za spomenutie knižnice:

- **NumPy** - táto knižnica pridáva podporu pre širokú radu numerických dátových typov, n-rozmerné polia a matice. Zároveň ponúka veľké množstvo funkcií pre prácu s týmito dátovými typmi. [15]
- **pandas** - knižnica pre manipuláciu a analýzu rozsiahlych dátových štruktúr. Je postavená na knižnici *NumPy* a je vysoko optimalizovaná na rýchlosť a výkon. Najvýznamnejšími dátovými štruktúrami sú časové rady (Series) a dátové rámce (DataFrame), ktoré reprezentujú tabuľkové dáta. Pre tieto štruktúry ponúka metódy pre vzorkovanie, výpočet okienkových funkcií a štatistík, filtrovanie, transpozície, interpoláciu dát a mnoho ďalších.[17]
- **matplotlib** - knižnica pre vizualizáciu dát a vykresľovanie grafov. Je integrovaná s knižnicami *NumPy* a *pandas*, takže dokáže pracovať priamo s dátovými typmi týchto knižníc. Ponúka široké množstvo grafov, vizualizáciu najrôznejších druhov dát. Jedná sa o *de facto* štandard pre vizualizáciu dát v jazyku Python.[26]
- **SciPy** - knižnica, ktorá je postavená primárne na knižnici *NumPy* a používa *NumPy* n-rozmerné polia ako hlavnú štruktúru. Je to knižnica pre vedecké a matematické operácie. Obsahuje balíčky pre optimalizáciu, lineárnu algebru, prácu so signálmi, definuje matematické a fyzikálne konštanty a mnoho dalších.
- **scikit-learn** - knižnica pre strojové učenie a dolovanie dát. Knižnica používa a stavia na knižniciach *SciPy*, *NumPy* a *pandas*. Knižnica ponúka široké množstvo algoritmov pre modelovanie dát ako sú: [21]
    - regresná analýza
    - zhlukovanie
    - rozhodovacie stromy
    - neurónové siete
    - support vector machines

2.2 Získavanie dát z webu

Získavanie dát z webu (tiež web scraping) je proces, pri ktorom automatizovaným spôsobom získavame dáta dostupné na webovej stránke a pretvárame ich na štruktúrované dáta. Program, ktorý toto získavanie dát vykonáva sa zvyčajne nazýva „bot“ alebo „scraper“.

Okrem pojmu web scraping narazíme často na pojem crawling. Tieto pojmy sú často zamieňané. Aj keď základný princíp, získavanie dát z webu, je rovnaký, jedná sa o mierne rozdielne procesy. Crawling je proces, kde prechádzame veľké množstvo stránok tým, že sledujeme jednotlivé odkazy, ktoré sa na stránke nachádzajú a obsahy jednotlivých stránok ukladáme. Pri web scrapingu väčšinou implementujeme bota pre konkrétnu štruktúru stránky, z ktorej chceme dáta získavať. Väčšinou sa jedná o dáta ktoré budeme chcieť získavať pravidelne, prípadne o veľké množstvo dát rozdelených na niekoľkých stránkach, kde štruktúra stránok je rovnaká (napríklad pomocou stránkovania). Celý proces web scrapingu pozostáva z dvoch hlavných úloh: Získanie obsahu web-stránky od webservera a extrakcia štruktúrovaných dát z tohto obsahu. Každá z týchto úloh so sebou nesie určité výzvy a špecifické problémy, s ktorými je nutné sa vysporiadať.

Web scraping podľa tejto definície existuje takmer od samotného začiatku existencie webu. Prvým takýmto botom bol *World Wide Web Wanderer*, ktorého úlohou bolo monitorovať veľkosť webu. Neskôr v tom istom roku bol spustený prvý webový vyhľadávač JumpStation, ktorý spoliehal na informácie nazbierané pomocou botov.[16] Web scraping je vo veľkej miere používaný dodnes a niektoré zdroje uvádzajú, že viac ako polovica aktivity na webe je spôsobená takýmito botmi. [37] Web scraping je v súčasnosti využívaný pre najrôznejšie potreby. Niektoré z mnohých prípadov použitia sú napríklad:

- Webové vyhľadávače - dá sa povedať že všetky vyhľadávače sú existenčne odkázané na web scraping. Web ako ho poznáme dnes by bez toho, aby vyhľadávače pravidelne zbierali informácie o webstránkach, neexistoval.

- Agregátori obsahu - či už sa jedná o vyhľadávače hotelov, leteniek, pracovných pozícií alebo prenájmov. Agregátori obsahu zbierajú dáta z niekoľkých webstránok a ponúkajú svojim užívateľom všetok obsah v jednotnom formáte na jednom mieste.

- Monitorovacie nástroje - pravidelne zbierajú dáta zo stránky a upozornia užívateľa na určité zmeny. Môže ísť napríklad o ceny produktov na internetovom obchode a vyhľadávače zliav. Prípadne môže ísť o program, ktorý nás upozorní, že na internetovom bazáre pribudol nový inzerát, ktorý by pre nás mohol byť zaujímavý.

- Zber dát pre analýzu alebo trénovanie - niektoré druhy dát nie sú ako dátové sady verejne dostupné v takom rozsahu alebo granularite, ako by sme potrebovali, prípadne historické dáta nie sú dostupné vôbec. Web scraping je často používaný na dlhodobý zber dát za účelom vytvárania rozsiahlych dátových množín. Tieto potom môžu byť použité pre účely trénovania modelov alebo z nich môžu byť získavané znalosti.

2.2.1 Časté problémy

Nie je úplne nezvyčajné, že niektoré stránky podstupujú isté opatrenia preto, aby zabránili alebo znepríjemnili automatizované získavanie dát z ich stránok. Tieto môžu byť rôzne sofistikované a ich obchádzanie môže byť v niektorých prípadoch značne komplikované. Niekedy až natoľko, že tomu je nutné do značnej miery prispôsobiť štruktúru programu a spôsob akým dáta získavame. Predtým, ako z akejkoľvek stránky začneme dáta extrahovať

je dobré preskúmať, či daná služba neposkytuje verejnú API. Uľahčíme tak prácu sebe a pravdepodobne aj webserveru danej stránky. Častými spôsobmi obchádzania obmedzení sú napríklad [20]:

- User-Agent *spoofing*, teda predstieranie, že dotazy sú posielané z webového prehliadača. Toto je docielené zmenou hlavičky `User-Agent` v dotazoch na takú, akú používa niektorý z populárnych webových prehliadačov. Týmto spôsobom tiež môžeme predstierať, že dotaz prichádza z mobilného zariadenia.

- Zakázanie/mazanie *cookies*. Niektoré web-stránky môžu detekovať automatizované správanie pomocou cookies. Napríklad ak príliš veľa dotazov v krátkom čase prichádza z jedného sedenia (session).

- Zníženie frekvencie dotazov a odstránenie paralelných dotazov. Ak pre jednu doménu vytvárame niekoľko dotazov súčasne alebo vo veľmi krátkom časovom úseku, niektoré web-stránky môžu túto „nadľudskú“ rýchlosť detekovať a rozpoznať automatizované správanie.

- Rotácia IP adries/použitie HTTP proxy. Niekedy stránka detekuje, že dotazy prichádzajú z rovnakého zariadenia na základe IP adresy. Rotáciou IP adries docielime, že z jednej IP adresy nebude prichádzať príliš veľa dotazov v príliš krátkom časovom úseku.

- Niektoré stránky používajú opatrenia vo forme CAPTCHA[29]. Existuje niekoľko komerčných služieb pre obchádzanie tohto opatrenia. Niektoré sú implementované pomocou umelej inteligencie a algoritmov pre rozpoznávanie obrazu, zatiaľ čo niektoré využívajú tzv. „klikfarmy“ v krajinách tretieho sveta.

2.2.2 Podpora v jazyku Python

Pre účely web scrapingu je jedným z veľmi často používaných implementačných jazykov jazyk Python. Jedným z dôvodov je, že sa jedná o vysokoúrovňový interpretovaný jazyk, čo umožňuje rýchly vývoj takýchto systémov. Nakoľko zber dát z webu väčšinou nie je časovo kritická operácia, relatívne nízka rýchlosť programov v tomto jazyku nie je zvyčajne problémom. Ďalším dôvodom je aj to, že existuje veľké množstvo knižníc, ktoré môžu byť použité pri riešení takýchto úloh. Tieto knižnice navyše majú vo väčšine prípadov veľmi dobrú podporu v online komunite a ich použitie je dobre zdokumentované. Niektoré z najčastejšie používaných knižníc pre tieto účely sú:

- **requests** - táto knižnica je pravdepodobne najpoužívanejšia knižnica pre vytváranie HTTP dotazov a čítanie odpovedí a je to *de facto* štandardná knižnica pre prácu s HTTP protokolom. Abstrahuje celú aplikačnú vrstvu sieťovej komunikácie pre HTTP protokol a ponúka užívateľsky priateľské rozhranie, ktoré dovoľuje jednoducho komunikovať s webovými službami a konzumovať dáta.[18] Knižnica uľahčuje prácu s HTTP protokolom niekoľkými spôsobmi:
    - kódovanie URL parametrov
    - kódovanie slovníkov ako formulárových parametrov v dotazoch typu POST
    - správa sedení (sessions) a cookies
    - automatické dekódovanie odpovedí vo formáte JSON do Python slovníkov

  - metódy pre uľahčenie práce s najpoužívanejšími spôsobmi HTTP autentifikácie
  - možnosť automaticky nasledovať presmerovania
  - podpora pre automatickú kompresiu a dekompresiu dotazov a odpovedí
  - opätovné pripojenie v prípade krátkodobého výpadku alebo nedostupnosti webového servera

- **websockets** - knižnica pre prácu s websocketmi[7]. Uľahčuje vytvorenie spojenia pomocou HTTP a následne čítanie dát z websocketu. Ponúka funkcie pre zabezpečenú komunikáciu cez SSL ako aj nezabezpečenú komunikáciu. Knižnica zároveň ponúka interaktívneho klienta pre príkazový riadok, užitočného pre rýchle testovanie a ladenie.

- **BeautifulSoup** - knižnica pre prácu a extrakciu dát z HTML a XML dokumentov. Využíva niektorý z použiteľných HTML/XML parserov a ponúka niekoľko možností ako navigovať, vyhľadávať a meniť tieto dokumenty poskytnutým Python rozhraním. Jedná sa o často používanú knižnicu v prípadoch, keď webserver priamo poskytuje statickú odpoveď vo forme HTML alebo XML dokumentu, v ktorom sa nachádzajú dáta ktoré sa pokúšame získať. Funkcionalita, ktorú táto knižnica poskytuje zahŕňa:

  - množstvo funkcií pre vyhľadávanie prvkov spracovaného dokumentu. Napríklad:
    - vyhľadávanie podla HTML/XML značiek
    - vyhľadávanie podla atribútov uzlov
    - navigovanie hierarchie, možnosť získať odkaz na rodičovský uzol a odkazy na poduzly
    - vyhľadávanie uzlov pomocou regulárnych výrazov
    - navigácia pomocou CSS selektorov
    - navigácia pomocou XPath[23].
  - formátovanie spracovaného dokumentu
  - automatické rozpoznanie kódovania dokumentu[2]

- **json/ultrajson** - ako názov napovedá, jedná sa o knižnice pre prevod reťazcov vo formáte JSON[3] na Python objekty. Javascript objekty sú prevádzané na slovníky jazyka Python, Javascript polia na Python zoznamy a primitívne dátové typy `string`, `long` a `float` sú prevedené na príslušné primitívne dátové typy v jazyku Python. Tieto knižnice sú často používané v prípadoch, keď webserver, z ktorého dáta získavame, tieto dáta poskytuje dynamicky. Výhodou knižnice *json* je, že sa jedná o štandardnú knižnicu jazyka dostupnú v štandardnej inštalácii interprétu. Knižnica *ultrajson* je implementovaná v jazyku C a je zameraná na rýchlosť. Podľa benchmarkov, v závislosti od operácie, dosahuje tri až šesť-násobné zrýchlenie oproti štandardnej knižnici. [11][6]

- **Selenium** je framework, ktorý je primárne určený pre automatizované testovanie webov a webových aplikácií. *Selenium* poskytuje rozhranie pre väčšinu populárnych programovacích jazykov vrátane jazyka Python. Hlavnou súčasťou tohto framewroku je *Selenium Webdriver*, teda ovládač, s ktorým je možné komunikovať práve vďaka Python API. Táto komponenta umožňuje komunikovať s konkrétnou inštanciou webového prehliadača pomocou ovládačov špecifických pre daný prehliadač. Dostupné

sú ovládače pre najpopulárnejšie prehliadače ako *Goolge Chrome*, *Safari*, *Firefox*, *Internet Explorer* a *Microsoft Edge*. Zároveň sú dostupné ovládače pre tzv. „headless“ prehliadače, teda také, ktoré nemajú žiadne grafické rozhranie, no dovoľujú interpretovať klientský kód. Pomocou Selnium API je možné posielať prehliadaču príkazy pre kontrolu jednotlivých prvkov, zadávanie vstupu a interakciu s užívateľskými prvkami. Zároveň API umožňuje získavať informácie z prehliadača ako aktuálny DOM, cookies a pod. Týmto efektívne dovoľuje simulovať správanie bežného užívateľa na webovej stránke. Selenium sa často používa pre získavanie informácií z webových stránok, kde nie je možné informácie priamo získať z webového servera, ale sú poskytnuté až klientskym kódom na danej webovej stránke alebo v prípade, že dotazy na webserver sú príliš komplikované, alebo z rôznych iných dôvodov, keď dáta nie je možné získať iba pomocou HTTP dotazov. Nevýhodou je, že tento spôsob získavania dát je relatívne pomalý a pamäťovo náročný, nakoľko program musí vytvoriť inštanciu webového prehliadača, ktorý potom interpretuje klientský Javascript kód, ktorý beží na webovej stránke.[22]

- **Scrapy** je aplikačný framework pre web scraping a crawling. Framework je primárne určený pre web scraping, no je možné ho použiť aj pre získavanie dát z verejných API. Jednou z hlavných výhod je, že je postavený na sieťovom frameworku *Twisted*[35], ktorý je asynchrónny. Framework je teda navrhnutý tak, že dotazy posiela paralelne, čím sa zvyšuje rýchlosť a efektivita programu pri získavaní dát z viacerých zdrojov. Toto nám síce dovoľuje vytvárať veľké množstvo dotazov v krátkom čase. Zároveň však framework dovoľuje definovať, ako „slušne“ sa bot má správať, napríklad zvýšením oneskorenia medzi jednotlivými dotazmi alebo obmedzením maximálneho počtu paralelných dotazov v rámci jednej domény, tak aby sme neprimerane nezaťažovali webserver dotyčnej stránky. Framework dokonca ponúka možnosť použiť rozšírenie, ktoré sa pokúsia tieto parametre odvodiť automaticky na základe odpovedí od servera. [20]

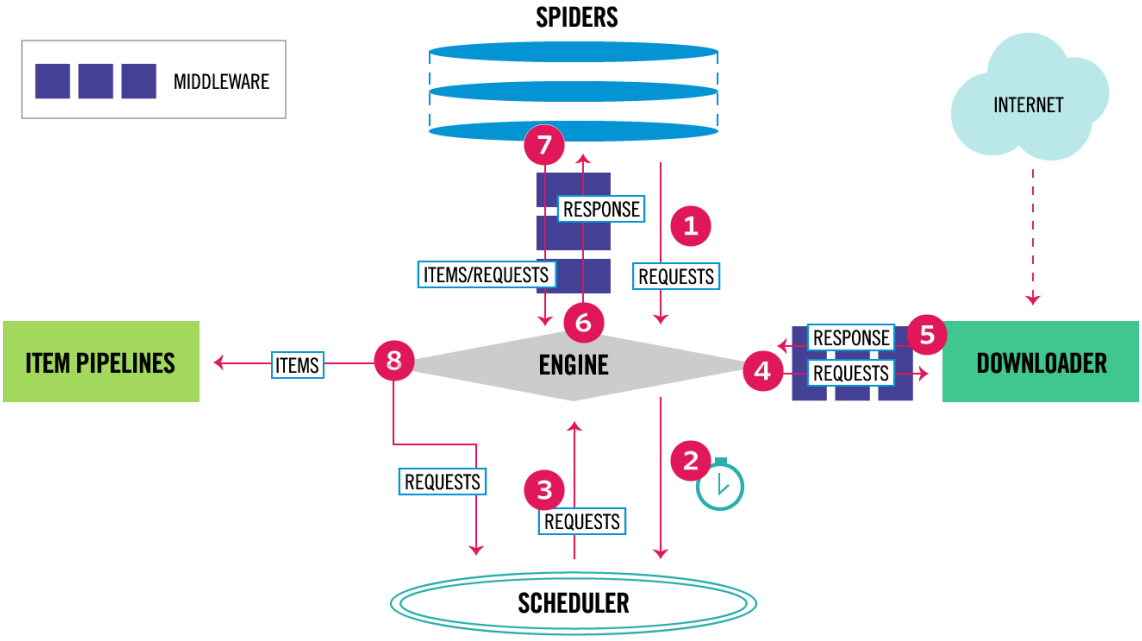
  Ďalšími možnosťami tohto frameworku, ktoré stoja za spomenutie sú:

  - vstavaná podpora pre spracovanie XML/HTML s použitím CSS selektorov a XPath ako aj pomocné metódy pre vyhľadávanie a podpora pre regulárne výrazy.
  - interaktívny režim pre ladenie CSS a XPath dotazov
  - vstavaná podpora pre generovanie exportov vo formátoch CSV, JSON, XML
  - podpora pre automatické kódovanie a dekódovanie ako aj detekcia nesprávnych deklarácií kódovania.
  - rozšíriteľnosť pomocou middlewarov, sériového spracovania, rozšírení a signálov
  - široké spektrum vstavaných rozšírení pre: prácu s cookies a sedeniami, HTTP možnosti kompresie autentifikácie a cachovania, user-agent spoofing.
  - možnosť nasadiť na vzdialenom serveri a ovládať pomocou REST API

  Architektúra tohto frameworku je znázornená na obrázku 2.3, pričom jeho komponenty sú:

  - `ENGINE` je riadiacou jednotkou frameworku a jeho zodpovednosťou je riadiť dátový tok medzi jednotlivými komponentami.

- `SCHEDULER`, teda plánovač dostáva dotazy od `ENGINu` a radí ich do fronty tak, aby ich neskôr mohol v naplánovanom čase opäť poskytnúť `ENGINu`.
- `DOWNLOADER` má ako hlavnú úlohu získavať odpovede od webserverov a preposielať ich `ENGINu`, ktorý ich následne prepošle príslušným pavúkom.
- `SPIDER` alebo pavúk je hlavnou užívateľskou jednotkou v tomto frameworku. `SPIDER` ako trieda definuje vstupný bod stránky, z ktorej chceme dáta získavať, prípadne ako prechádzať hierarchiu stránky a ako vyvárať ďalšie dotazy. Okrem toho definuje aj to, ako získané dáta spracovať. Väčšinou definujeme nového pavúka pre každú stránku, z ktorej chceme dáta získavať.
- `ITEM PIPELINE` (zreťazené spracovanie položiek) má za úlohu spracovať položky, ktoré sú poskytnuté pavúkmi. Typicky sa jedná o úlohy ako validácia získaných dát, odstránenie duplicít, čistenie dát, prípadne ukladanie do relačnej databázy alebo iného úložiska
- `DOWNLOADER MIDDLEWARE` je sekvencia funkcií nachádzajúca sa medzi komponentami `ENGINE` a `DOWNLOADER`. Tieto funkcie budú postupne aplikované na dotazy prichádzajúce z `ENGINu` a na odpovede, ktoré vracia `DOWNLOADER`. Typickým prípadom použitia môže byť napríklad:
  - ignorovanie niektorých dotazov
  - vytvorenie nového dotazu na základe odpovede od servera bez toho, aby sme pôvodnú odpoveď vracali pavúkovi
  - vytvorenie odpovede bez toho, aby sme dotaz reálne posielali na webserver
  - úprava dotazu pred odoslaním na webserver (napríklad pridanie `User-Agent` hlavičiek, proxy hlavičiek a pod.)
- `SPIDER MIDDLEWARE` je sekvencia funkcií nachádzajúca sa medzi komponentami `ENGINE` a `SPIDER`. Tieto budú postupne aplikované na vstupy (odpoveď od webservera) a výstupy (spracovane položky a dotazy) pavúka. Typickým prípadom použitia je napríklad spracovanie výnimiek z pavúka.



Obr. 2.3: Scrapy architektúra[20]

Tok dát medzi jednotlivými komponentami znázornenými na obrázku 2.3 je nasledovný:

1. `ENGINE` získa počiatočný dotaz od pavúka
2. `ENGINE` naplánuje tento dotaz pomocou plánovača `SCHEDULER` a zároveň od neho získa ďalší naplánovaný dotaz
3. `SCHEDULER` vráti ďalší naplánovaný dotaz z fronty.
4. `ENGINE` posiela dotaz do `DOWNLOADERu`, pričom dotaz prechádza cez `DOWNLOADER MIDDLEWARES`
5. po tom, čo `DOWNLOADER` získa odpoveď od servera, posiela túto odpoveď `ENGINu` pričom odpoveď prechádza cez `DOWNLOADER MIDDLEWARES`
6. `ENGINE` obdrží odpoveď a prepošle ju pavúkovi, ktorý ju následne môže spracovať, pričom táto odpoveď prechádza cez `SPIDER MIDDLEWARE`
7. `SPIDER` spracuje odpoveď a vracia extrahované položky a nové dotazy `ENGINu`, pričom tieto prechádzajú cez `SPIDER MIDDLEWARE`
8. `ENIGNE` preposiela spracované položky do `ITEM PIPELINES` a nové dotazy plánovaču
9. Proces sa opakuje (od kroku 1) kým fronta dotazov v plánovači nie je prázdna

Kapitola 3

Prípadová štúdia - návrh

Prípadová štúdia bude zameraná na analýzu vývojov cien kurzov pre futbalové stávky. Zdrojom dát pre túto analýzu budú kurzové ponuky spoločnosti *Niké* a *Pinnacle*. Dáta zo stránky spoločnosti *Niké* budú získavané pomocou web scrapingu, zatiaľ čo dáta spoločnosti *Pinnacle* budeme získavať pomocou ich verejne dostupnej API. V rámci športov sa obmedzíme iba na futbalové zápasy v ponuke týchto kancelárií. Zbierať budeme kurzy pre stávky typu „1x2“ a „počet gólov“ - popísané nižšie. Pre informácie o výsledkoch podujatí použijeme službu spoločností *Pinnacle*.

3.1 Úvod do kurzového stávkovania

Športové stávkovanie je forma hazardu a ide o činnosť, pri ktorej sa hráč, nazývaný stávkar alebo tiež tipér, snaží správne tipnúť, či nastane konkrétna situácia, ktorá je reprezentovaná stávkou, pričom na tento tip staví istú sumu. Hráč hrá proti stávkovej kancelárii. Stávková kancelária zverejňuje ponuku možných stávok, ktoré sa viažu na konkrétne športové udalosti spolu s kurzami pre konkrétne tipy. Kurz je číselná hodnota, ktorá hovorí o tom, akú veľkú výhru vzhľadom na výšku stávky stávková kancelária vyplatí v prípade, že stávkar tipne víťazný tip, teda že nastane udalosť, na ktorú stavil.

Tip môže mať dva výsledky. Ak nastala udalosť, ktorú stávkar tipol, vyhráva a bude mu vyplatená výhra, ktorá pozostáva zo samotnej stávky (vkladu) a zisku podľa výšky vkladu a kurzu alebo prehráva a prichádza o svoju stávku. [1]

3.1.1 Druhy stávok

Jednotlivé druhy stávok môžme kategorizovať niekoľkými spôsobmi. Jedným z nich je delenie podla času:

- predzápasové stávkovanie (pre-game) - jedná sa o stávky, pre ktoré platí že udalosti, na ktoré sa viažu, ešte nezačali. Ide o najrozsiahlejší druh stávkovania čo sa týka objemu ponúkaných zápasov a počtu stávkových príležitostí. Typicky stávkové kancelárie zverejňujú takéto ponuky jeden až dva týždne pred začiatkom udalosti a stávkovanie je väčšinou možné až do samotného začiatku udalosti.

- živé stávkovanie (live) - jedná sa o stávky, ktoré sa uzatvárajú počas priebehu udalosti, na ktorú sa tieto stávky viažu. Je typické, že kurzy sa pri týchto stávkach rapídne

[1] Existujú stávky kde je aj tretia možnosť, a síce že stávkar dostane naspäť iba svoju stávku (vklad), teda ani nevyhráva, ani neprehráva. V tejto práci sa takéto typy stávok nevyskytujú.

menia podľa diania na ihrisku a stavu zápasu. Vyplýva z toho, že takéto udalosti sú ponúkané v oveľa menšom objeme, keďže je potrebné sledovať zápas a upravovať kurzy podľa diania na ihrisku.

- priame stávkovanie (outrights) - stávky sa nevzťahujú na jednu konkrétnu udalosť, ale na celkové umiestenie v rámci ligy alebo súťaže. Takéto stávkové príležitosti majú väčšinou väčšie množstvo možností ako dve alebo tri. Typickým príkladom je stávka na celkového víťaza ligy majstrov, kde možnosťami sú jednotlivé tímy a väčšinou sú v ponuke všetky tímy ligy. Tieto stávky sa zvyčajne zverejňujú vo veľkom predstihu, a teda na vyhodnotenie sa často čaká niekoľko mesiacov.

Pre konkrétne zápasy pri live a predzápasovom stávkovaní existuje niekoľko druhov stávok, ktoré môžu pre daný zápas byť v ponuke. Jedná sa o konkrétne situácie alebo fakty spojené s daným zápasom, ktoré sa tipér snaží tipnúť, či nastanú. Najčastejšími druhmi stávok sú[25][19]:

- 1x2 (anglicky často nazývané aj threeway). Na prvý pohľad možno trochu mätúci názov reprezentuje tri možné tipy pri tejto stávke. Stávkar sa v tomto prípade snaží tipnúť, kto vyhrá stretnutie. Možné výsledky sú teda: „vyhrá domáci tím“ (1), „remíza“ (x) „vyhrá druhý tím“ (2). Tento typ stávky sa môže vzťahovať aj na výsledok niektorej periódy zápasu. Napríklad pre futbal sa môže jednať o stávku na to, kto vyhrá prvý polčas.

- Dvojitá šanca (angl. double chance). Jedná sa o veľmi podobný typ stávky ako v prvom prípade. Rozdiel je v tom, že miesto tipovania ktorá z troch možností nastane, tipuje sa ktorá z možností nenastane. Teda možnosti sú „nevyhrá prvý tím“, „nebude remíza“, „nevyhrá druhý tím“. Názov dvojitá šanca vyplýva z faktu, že každá z možností zahŕňa 2 z možností 1x2. Napríklad „nevyhrá prvý tím“ je možné vyjadriť ako „vyhrá druhý tím alebo remíza“

- Víťaz zápasu (angl. moneyline) je podobný typ stávky ako *1x2* s tým, že možnosť remízy neuvažujeme. V prípade futbalu by sa jednalo napríklad o víťaza zápasu v nadstavenom čase alebo po penaltovom rozstrele. V športoch, kde remíza neexistuje, napríklad tenis, sa tento typ stávky v angličtine označuje aj ako „twoway“.

- Párne/Nepárne (angl. even/odd) je stávka s dvoma možnými tipmi, pri ktorej stávkar tipuje, či celkový počet gólov/bodov v zápase bude párny alebo nepárny.

- Počet gólov (angl. totals alebo over/under) je druh stávky s dvoma možnými výsledkami. Stávkar tipuje, či celkový počet bodov/gólov v zápase bude vyšší alebo nižší ako daná prahová hodnota. Napríklad pre ponuku „Počet gólov 2.5“ sú možnosti „menej ako 2.5“, ktorá zahŕňa 0, 1 a 2 góly, a „viac ako 2.5“, ktorá zahŕňa všetky ostatné výsledky.

- Štatistické stávky - tento pojem zahŕňa niekoľko druhov stávok. Jedná sa o stávky, kde tipujúci tipuje hodnoty niektorej štatistiky zápasu. Jednotlivé tipy môžu mať formát „viac/menej“, ako je tomu pri počte gólov, interval alebo presná hodnota. Príkladmi takýchto stávok sú napríklad: „počet striel na bránu“, „počet červených kariet“, „počet rohových kopov“ a pod.

Existuje mnoho ďalších druhov stávok, tieto sú však najtipovanejšie. Väčšina druhov stávok navyše existuje v rôznych variáciách, kde sa výsledok vzťahuje iba na konkrétnu

periódu zápasu alebo len na konkrétny tím. V tejto prípadovej štúdii budeme sledovať hlavne stávky typu *1x2* a *počet gólov* a to pre periódy prvý polčas, druhý polčas a koniec zápasu.

Jednotlivé stávky je možné kombinovať, pričom výsledný kurz je produktom kurzov jednotlivých stávok.[2]. Toto je však možné len pokiaľ sa nejedná o navzájom závislé stávky. Napríklad nie je možné v rámci jednej stávky staviť na tip „menej ako 2.5 gólov“ a zároveň „menej ako 3.5 gólov“. Takéto stávky sa nazývajú kombinované stávky alebo akumulátory. Výsledná stávka sa považuje za víťaznú jedine v prípade, že všetky dielčie stávky sú víťazné.

3.1.2 Formáty kurzov

Samotné kurzy môžu byť reprezentované niektorým z najčastejšie používaných formátov. Pre porovnanie ponúk z dvoch kancelárií, kde každá uvádza kurzy v inom formáte je potrebné vedieť medzi jednotlivými formátmi prevádzať. Najčastejšie používané formáty kurzov sú [36] [27]:

- Desatinný formát (európsky) (angl. decimal) je najčastejšie používaný v Európe, Kanade a Austrálii. Má formu desatinného čísla. Číslo reprezentuje výšku potenciálnej výhry vzhľadom na vklad. Výška potenciálnej výhry je spočítaná podľa vzorca $vyhra = kurz * stavka$. Pri stávke 100$ s kurzom 1.5 by teda výhra bola 150$ s tým, že čistý zisk tvorí 50$. Pri tomto formáte je jednoduché spočítať výšku kurzu pre akumulátorové stávky, pretože celková výška kurzu je súčin jednotlivých kurzov.

- Zlomkový formát (angl. fractional), nazývaný tiež britský, je najčastejšie používaný v krajinách Veľkej Británie a Írska. Je to číslo v tvare zlomku a indikuje výšku potenciálneho čistého zisku vzhľadom na stávku. Napríklad pri stávke 100$ pri kurze 2/1 (slovom dva ku jednej), by potenciálna výhra bola 300$, teda zisk 200$. Pri tej istej stávke a kurze 1/2 (slovom jedna ku dvom), by bola potenciálna výhra 150$, teda zisk 50$. Prevod na Európsky formát docielime tak, že zlomok prevedieme na desatinné číslo a pričítame 1.

- Moneyline (americký) formát najčastejšie používaný v USA má formu celého kladného alebo záporného čísla, pričom znamienko je väčšinou explicitne uvedené. V prípade, že je kurz záporný, indikuje, aký musí byť vysoký vklad na to, aby čistý potenciálny profit bol 100$. Ak je teda kurz napríklad -200, znamená to, že pri stávke 200$ je potenciálna výhra 300$, a teda zisk 100$. Ak je kurz kladný, indikuje to, aká bude výška zisku pri stávke 100$. Napríklad pre kurz +200 bude pri stávke 100$ výhra 300$ a čistý zisk 200$. Prevod na európsky formát závisí od toho, či je kurz kladný alebo záporný. Pre záporné kurzy $-100/us_kurz + 1$. Pre kladné kurzy $us_kurz/100 + 1$

- Hongkongský, často nazývaný aj ázijský, je jeden z niekoľkých formátov, ktoré sú často používané v ázijských krajinách. Jedná sa o desatinné číslo, ktoré reprezentuje čistý zisk v závislosti na výške stávky. Napríklad pri kurze 0.5 a stávke 100$, bude potenciálna výhra 150$ a čistý zisk 50$. Tento formát je veľmi podobný európskemu s tým rozdielom, že sa nevzťahuje na výhru, ale na čistý zisk. Pre prevod na európsky formát stačí ku kurzu pričítať 1.

Tabuľka 3.1 pre lepšie znázornenie ukazuje niekoľko prípadov ekvivalentných kurzov v rôznych formátoch. V tejto práci budú všetky kurzy konvertované na európsky formát, a teda všetky spomenuté kurzy budú uvádzané v tomto formáte.

[2] Platí pre kurzy uvádzané v európskom formáte

európsky	ázijský	britský	americký
1.01	0.01	1/100	-10000
1.2	0.2	5/1	-500
2.0	1.0	1/1	-100
2.5	1.5	3/2	+150
10.0	9.0	9/1	+900

Tabuľka 3.1: Ukážka prevodu kurzov v odlišných formátoch

3.1.3 Marže stávkových kancelárií a pravdepodobnosť

Stávkové kancelárie zarábajú tým, že prijímajú stávky od hráčov a upravujú kurzy tak, aby stávkari tipovali jednotlivé možnosti v správnom pomere a tým si zabezpečili zisk bez ohľadu na to, ako daná udalosť dopadne. Dosiahnu toho tým, že ponúkajú kurzy, ktoré nereprezentujú skutočnú štatistickú pravdepodobnosť, že daná situácia nastane. Táto odchýlka od skutočnej pravdepodobnosti sa nazýva marža. Implikovanú pravdepodobnosť spočítame ako prevrátenú hodnotu kurzu (v európskom formáte).

Jednoduchým príkladom na demonštrovanie tohto konceptu je hod mincou. Pri hode mincou existujú dve situácie, ktoré môžu nastať a obe sú rovnako pravdepodobné. Ak by stávkové kancelárie ponúkali férové kurzy, bol by kurz na obidve možnosti 2.0, teda implikovaná pravdepodobnosť je 50%. Súčet pravdepodobností je teda 100% a odchýlka od reálnej pravdepodobnosti, že nastane jedna z 2 možností je 0. Stávkové kancelárie nám však neponúknu kurz 2.0, ale napríklad 1.92 na obidve možnosti. To implikuje pravdepodobnosť približne 52% a v súčte 104%. Marža stávkovej kancelárie je teda pre túto ponuku približne 4%. Toto je spôsob, akým si stávkové kancelárie zaisťujú zisk bez ohľadu na to, ktorá z možných situácií nastane.[24] [3]

3.2 Získavanie dát z webu

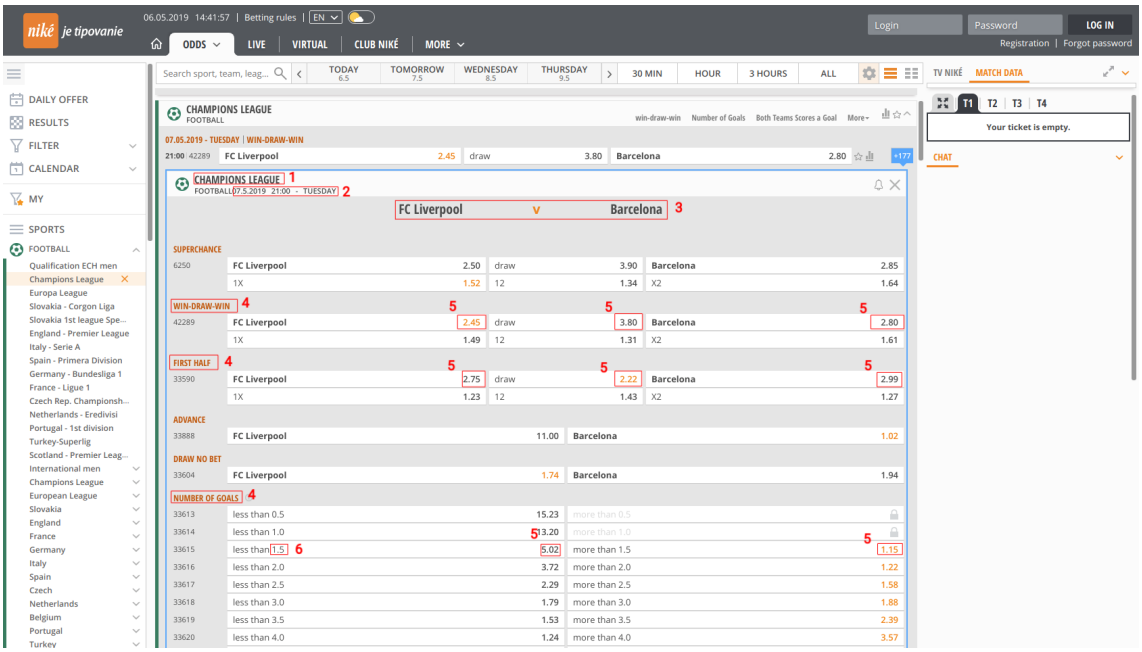
3.2.1 Analýza dostupných dát

Jedným z prvých krokov je zamyslieť sa nad tým, aké dáta chceme z webstránky získavať. Je nutné zistiť, či sú dáta na stránke nejakým spôsobom logicky štruktúrované a formátované alebo ich budeme musieť získavať z voľného textu, obrázku alebo iného neštandardného formátu. Zároveň je v tejto fáze dobré porozmýšľať nad tým, ako sú na sebe dáta závislé a to, či sú všetky potrebné údaje dostupné na jednom mieste alebo je nutné pre kompletný záznam získať dáta z viacerých miest na danej webovej lokalite. Je tiež vhodné zistiť, aká je hierarchia stránky a ako sú dáta usporiadané. V našom prípade sú jednotlivé zápasy najprv rozdelené podľa športov, následne každý šport pod sebou má niekoľko líg a konkrétne zápasy sú umiestnené pod príslušnými ligami.

V poslednom rade je potom vhodný čas rozhodnúť sa, aké dátové typy budu reprezentovať získané informácie ideálne tak, aby boli tieto typy univerzálne použiteľné bez ohľadu

[3] Tento príklad sa môže zdať ako nadmerné zjednodušenie, no stávkové kancelárie takéto stávky skutočne ponúkajú. Pri zápasoch na neutrálnej pôde rozhoduje o tom, ktorý tím získa loptu hod mincou a stávkové kancelárie na toto ponúkajú možnosť staviť. De facto tipujúci teda tipuje výsledok hodu mincou. [5].

na konkrétny zdroj. V prípade zberu stávkových kurzov a sledovania ich vývoja v čase nás budú zaujímať tieto údaje znázornené na obrázku 3.1.



Obr. 3.1: Ukážka ponuky konkrétneho zápasu na stránke nike.sk

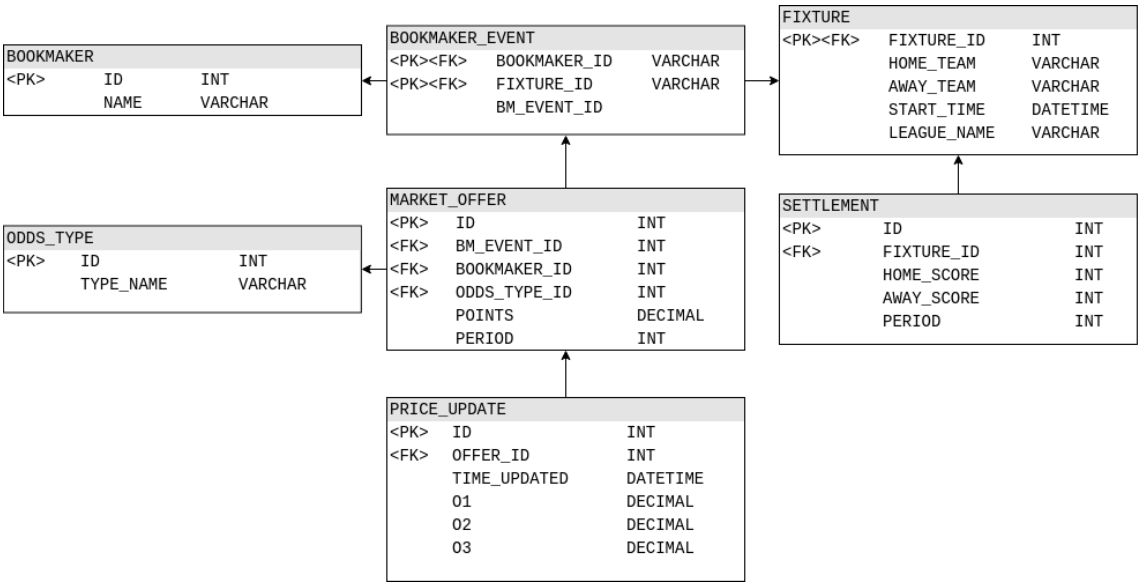
1. názov ligy, pod ktorú konkrétny zápas patrí
2. dátum a čas začiatku udalosti
3. názvy tímov domácich a hostí
4. typ stávky a perióda zápasu (1. polčas, 2. polčas, koniec zápasu)
5. kurzy pre jednotlivé typy danej stávky
6. v prípade stávok „počet gólov“ prahová hodnota pre danú stávku

Ďalej môžeme definovať obory hodnôt jednotlivých atribútov. Tento krok nie je absolútne nevyhnutný pre návrh dátového modelu, ale neskôr nám môže pomôcť detekovať chyby a nesprávne spracované údaje. V prípade kurzových stávok sú možnými obmedzeniami:

- jednotlivé kurzy nesmú mať hodnotu nižšiu alebo rovnú hodnote 1.0, teda stávka, u ktorej je potenciálna výhra nulová. Stávkové kancelárie občas používajú túto hodnotu v prípade, že sa zápas zruší alebo odloží.
- Marža stávkovej kancelárie nebude záporná alebo nulová. To by znamenalo, že by bolo možné garantovať výhru tým, že by sme stavili na všetky možné výsledky udalosti. Stávkové kancelárie z pochopiteľných dôvodov takéto kurzy neponúkajú, a tak toto pravdepodobne indikuje chybu pri spracovaní.
- Dátum a čas začiatku zápasu, pre ktorý získavame kurzy nebude v minulosti. Táto podmienka je špecifická pre náš prípad a musí byť platná najmä kvôli tomu, že v tejto práci nás zaujíma iba vývoj kurzov pred samotným začiatkom zápasu. Pokiaľ sa tomu teda dá vyhnúť, nebudeme zbierať kurzy pre live stávky.

3.2.2 Návrh databázy

Po analýze dostupných dát a ich formátu je potrebné navrhnúť schému databázy, do ktorej budeme nazbierané dáta ukladať. Schéma navrhnutej databázy je zobrazená na diagrame 3.2. Sémantika jednotlivých tabuliek a stĺpcov je nasledovná:



Obr. 3.2: Schéma databázy

- `BOOKMAKER` - číselník, reprezentuje jednotlivé stávkové kancelárie, napríklad `Niké`, `Pinnacle`, `Tipsport` ...
- `ODDS_TYPE` - číselník, reprezentuje možné typy stávok, napríklad `1x2`, `totals`, `double chance` ...
- `FIXTURE` - jednotlivé záznamy predstavujú jeden konkrétny zápas a vychádza z reálneho sveta. Zápasy sú jednoznačne reprezentované názvami tímov, názvom ligy a dátumom a časom, kedy sa tento zápas odohráva.
- `SETTLEMENT` - táto tabuľka reprezentuje výsledky zápasov v jednotlivých periódach stretnutia. Každý záznam sa viaže na najviac jeden záznam tabuľky `FIXTURE`. Každý záznam nesie informácie o tom, ktorý tím nastrieľal koľko gólov v danej perióde. Na jeden záznam tabuľky `FIXTURE` sa tak môže viazať niekoľko záznamov tabuľky `SETTLEMENT` Konkrétne to budú záznamy pre výsledok prvého polčasu, výsledok druhého polčasu a výsledok v riadnom hracom čase.
- `BOOKMAKER_EVENT` - záznamy tabuľky sa viažu na práve jeden záznam tabuľky `FIXTURE`. Táto tabuľka mapuje reálne udalosti z tabuľky `FIXUTRE` na udalosti v ponuke konkrétnej stávkovej kancelárie. Keďže ponuky zápasov medzi jednotlivými stávkovými kanceláriami sa do veľkej miery zhodujú, na jeden záznam `FIXTURE` sa zvyčajne bude viazať viacero záznamov tejto tabuľky. V tejto tabuľke zároveň uchovávame interné ID zápasu tak, ako ho definuje konkrétna stávková kancelária. To nám pri ladení pomôže dohľadať daný zápas na stránke stávkovej kancelárie.

- `MARKET_OFFER` - predstavuje konkrétne stávky, ktoré stávková kancelária pre danú udalosť ponúka. Ponuky sa viažu na konkrétnu udalosť v tabuľke `BOOKMAKER_EVENT`. Atribút `ODDS_TYPE_ID` definuje, o aký typ stávky sa jedná (`1x2`, `počet gólov`). Atribút `PERIOD` definuje, o ktorú periódu sa pri tejto jedná. V prípade, že sa jedná o druh stávky `počet gólov`, atribút `POINTS` udáva prahovú hodnotu počtu gólov pre túto stávku.

- `PRICE_UPDATE` - predstavuje kurzy pre jednotlivé tipy danej stávky v konkrétnom čase. `O1, O2, O3` predstavujú konkrétne kurzy. V prípade stávok s 3 možnými tipmi predstavujú v tomto poradí tipy: vyhrá domáci, vyhrajú hostia, remíza. Pre stávky s 2 možnými tipmi, napríklad `počet gólov`, `O1` reprezentuje kurz pre tip „viac ako `POINTS`“ a `O2` kurz pre tip „menej ako `POINTS`“. Keďže kurzy sa v čase menia, na jeden záznam `MARKET_OFFER` sa viaže niekoľko záznamov tejto tabuľky.

3.2.3 Analýza backendu webstránky

Teraz je potrebné zistiť, ako potrebné dáta získame z webstránky. Prvým krokom v tomto prípade je otvoriť stránku, z ktorej dáta ideme získavať a otvoriť vývojársku konzolu v ľubovoľnom prehliadači[4]. Teraz je nutné zistiť, ako je backend stránky postavený. Vo vývojárskych nástrojoch zvolíme záložky „Network“, obnovíme stránku a sledujeme, aké dotazy poslal a odpovede obdržal náš prehliadač. Snažíme sa identifikovať jeden z 3 najpoužívanejších spôsobov, ktorým sú webstránky postavené:

- Stránka môže byť postavená dynamicky pomocou klientskeho kódu v jazyku JavaScript. V záložke `XHR (XML HTTP Request)` vývojárskych nástrojov hľadáme, či stránka posiela asynchrónne dotazy na svoj backend server. Ak je stránka postavená dynamicky, v tejto záložke väčšinou nájdeme dotazy, ktoré vracajú odpovede vo formáte JSON. Tieto dáta sú následne spracované a vykreslené na stránke klientskym kódom.

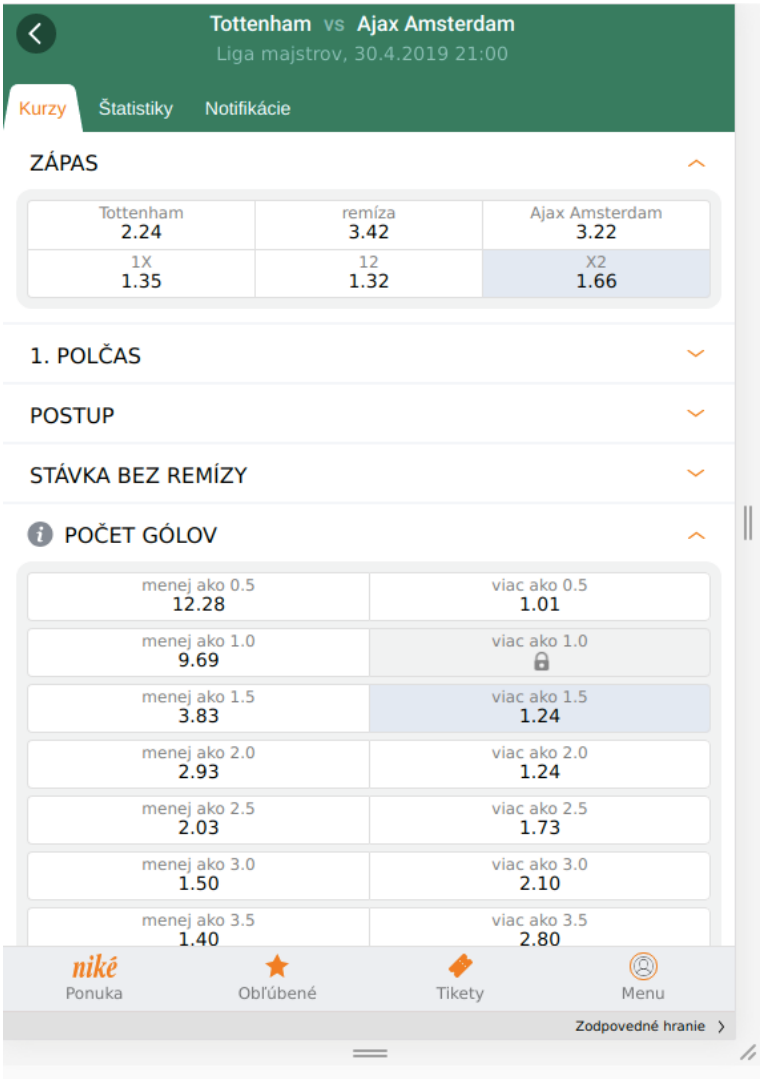
- Pri statických stránkach väčšinou posiela webserver všetky údaje, ktoré sa majú zobraziť na stránke, vo forme HTML. Záložka `XHR` v tomto prípade väčšinou neobsahuje žiadne dôležité dotazy. Zdrojové súbory HTML od webservera v tomto prípade nájdeme v záložke `Doc`.

- Poslednou z populárnych možností sú websockety. Websocket je pretrvávajúce spojenie, ktorým webstránka prijíma dáta z web servera. Tieto stránky sú opäť dynamické. Webová stránka pomocou websocketu prijíma dáta, ktoré následne pomocou klientskeho kódu aktualizuje v zobrazení stránky bez nutnosti znova načítať stránku. Prípadne otvorené websockety nájdeme v záložke `WS`

Ďalšou technikou, ktorú môžeme vyskúšať, je zmeniť pomocou vývojárskych nástrojov `user-agent` hlavičku na takú, ktorá reprezentuje mobilné zariadenie. Nakoľko webstránky obsahujú pomerne veľké množstvo informácií a hustota informácií je na desktopových verziách vysoká, ako je možné vidieť na obrázku 3.1, stávkové kancelárie často implementujú zvlášť rozhranie pre mobilné zariadenia. Nová hierarchia stránky niekedy znamená, že backend musí byť tiež prispôsobený tak, aby dodával dáta vo vhodnom formáte pre toto nové zobrazenie. Pre niektoré stávkové kancelárie, ktoré na trhu existujú dlhšiu dobu, sa tiež často stáva, že mobilná verzia stránky prišla podstatne neskôr, a tak používa modernejšiu alebo úplne inú verziu backendu.

[4] Tu používame Google Chrome

V prípade nike.sk vidíme, že stávková kancelária poskytuje zvlášť mobilnú verziu na doméne m.nike.sk, ktorá vyzerá oveľa jednoduchšie, lineárnejšie, obsahuje menej rušivých prvkov a zbytočných informácií, a napriek tomu ponúka tie isté dáta ako desktop verzia, ako je možné vidieť na obrázku 3.3 v porovnaní s obrázkom 3.1.



Obr. 3.3: Nike mobile

Ak teda pomocou vývojárskych nástrojov preskúmame, aké dotazy a odpovede stránka posiela a dostáva naspäť, vidíme, že stránka je postavená dynamicky a klientsky kód zobrazuje dáta, ktoré získava asynchrónne vo formáte JSON. Pri navigovaní stránky postupne podľa hierarchie *šport -> liga -> zápas* vidíme, že prehliadač vytvára nasledujúce dotazy[5]:

- pre získanie líg `:/prematch/menu?live=false&prematch=true`
- pre získanie jednotlivých zápasov ligy: `/prematch/boxes/mobile?boxId=<id>`
- pre získanie kurzov: `/prematch/v2/boxes/extended/bet-number/<id>`

[5]spoločný prefix api.nike.sk/api

Pri skúmaní stránky si tiež všimneme, že stránka ponúka možnosť zmeniť jazyk na angličtinu. Dôležité pre nás je, že po zmene jazyka sa okrem navigačných prvkov a názvov jednotlivých typov stávok zmenia aj názvy líg a tímov. Ak by sa nám dáta zo všetkých zdrojov podarilo získavať v rovnakom jazyku, neskôr by nám to výrazne uľahčilo prácu pri porovnávaní jednotlivých udalostí z rôznych zdrojov za účelmi detekcie identických udalostí. Po zmene jazyka na angličtinu znova preskúmame jednotlivé dotazy a porovnáme s pôvodnými dotazmi. Vidíme, že dotazy, ktoré posiela prehliadač po novom obsahujú HTTP hlavičku `Content-Language: en`.

Ak už poznáme konkrétne endpointy, je čas analyzovať parametre dotazu a aké déta sa nám vracajú v odpovediach od webservera. Naskytá sa tiež príležitosť experimentovať s jednotlivými parametrami a vyskúšať dotazy, ktoré sa na stránke prirodzene nevyskytujú, ale backend API môže podporovať.

Získanie všetkých futbalových líg

Pre získanie všetkých športov v ponuke, a zároveň všetkých líg pre tieto športy, použijeme už spomenutý endpoint */prematch/menu?live=false&prematch=true*. Zaujímavé sú parametre prematch a live. Tie sú nastavené tak, že od webservera získavame iba zápasy, na ktoré je možné predzápasové stávkovanie, a nie live stávkovanie. Je dobré tieto parametre použiť, nakoľko to ušetrí počet dotazov na webový server. Navyše nebudeme získavať dáta, ktoré by sme v neskorších fázach odstránili. Odpoveď od servera je zobrazená v ukážke 1[6].

[6] Pre prehľadnosť boli niektoré položky odpovede vynechané

```
{
 "items": [
  {
   "itemId": "164",
   "boxId": "bi-1-null-null",
   "slug": "/futbal",
   "label": "Football",
   "items": [
    {
     "itemId": "256",
     "boxId": "bi-1-44-null",
     "slug": "/futbal/medzistatny-muzi",
     "label": "International men",
     "items": [
      {
       "itemId": "2488",
       "boxId": "bi-1-44-t:2280",
       "slug": "/futbal/medzistatny-muzi/kvalifikacia-me-muzi",
       "label": "Qualification ECH men",
       "items": [],
      },
     ],
    },
   ],
  }
 ]
}
```


Ukážka 1: Opdoveď na dotaz pre získanie jednotlivých líg

Vidíme, že odpoveď tvorí stromovú štruktúru, kde k jednotlivým podstromom sa dostaneme cez položku `items`. Nás budú zaujímať iba listové položky, pretože práve tieto reprezentujú konkrétne ligy, zatiaľ čo nelistové uzly môžu reprezentovať šport alebo kategóriu líg (napríklad Nemecko, Anglicko, Medzinárodné zápasy muži). Listové uzly, ako vidíme, zodpovedajú ponuke líg na stránke stávkovej kancelárie. Pre získanie všetkých líg pre futbal teda použijeme tento endpoint.

Získavanie informácií o zápasoch

Pre získanie informácií o zápasoch pre jednotlivé ligy webová stránka volá už spomenutý endpoint `/prematch/boxes/mobile?boxId=<id>`, pričom `boxId` je ID konkrétneho listového uzlu, a teda zodpovedá ID konkrétnej ligy alebo súťaže. V ukážke odpovede od webservera 1 vidíme položku `sportEvents`. Jednotlivé prvky podľa všetkého reprezentujú konkrétne udalosti danej ligy. Toto je nutné verifikovať manuálne a porovnať záznamy v odpovedi s dátami, ktoré sa zobrazujú na webstránke. Keďže používame ten istý dotaz, ako vytvára samotný prehliadač pri bežnom používaní stránky, dáta by sa mali zhodovať.

Vidíme, že konkrétne položky poskytujú potrebné informácie o udalosti ako sú názvy tímov, meno danej ligy a čas začiatku. Toto sú dáta potrebné pre jednoznačné identifikovanie zápasu.


```
{
 "boxes": [...],
 "bets": [...],
 "markets": [...],
 "sportEvents": [
  {
   "boxIds": [
    "bi-1-98-t:3851"
   ],
   "sportEventId": "m:27057483",
   "sportId": "1",
   "isLive": false,
   "participants": [
    "Chelsea",
    "Eintracht Frankfurt"
   ],
   "sportName": "Futbal",
   "tournamentName": "Európska liga",
   "conflictingBetEvents": []
  },
 ],
}
```

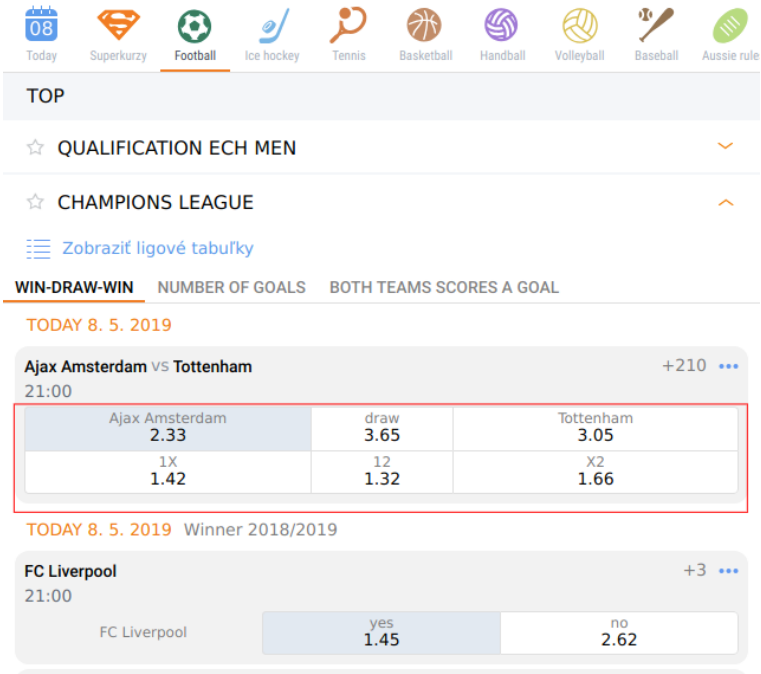

Ukážka 2: Odpoveď na dotaz pre získanie zápasov v konkrétnej lige

Parametrom dotazu je ID ligy. To je predávané ako URL parameter `GET` dotazu. V takýchto prípadoch sa väčšinou naskytá možnosť vyskúšať, či tomuto parametru nie je možné predať zoznam položiek. Ak skúsime vytvoriť dotaz, ktorému predáme čiarkami oddelený zoznam ID líg, vidíme, že od servera dostávame platnú odpoveď. Je nutné však overiť, či naozaj dostávame dáta pre všetky ligy. Nie je úplne nezvyčajné, že webserver príjme dotaz s viacerými hodnotami jedného parametra, no prebytočné hodnoty ignoruje a uvažuje iba prvú. Ak takýto dotaz vytvoríme, vidíme, že počet položiek v poli `sportEvents` sa skutočne zvýšil a vidíme, že zodpovedá počtu zápasov vo vybraných ligách tak, ako tomu je na stránke stávkovej kancelárie. Pri implementácii môžme teda posielať dotazy pre niekoľko líg zároveň a znížime tak celkový počet dotazov nutný k získaniu informácií o všetkých zápasoch, ktoré sú v danej chvíli v ponuke. V tomto momente máme informácie o šprotovom podujatí (angl. fixture), a následne potrebujeme získať konkrétne stávkové príležitosti a výšky kurzov pre tieto podujatia.

Získavanie stávok a kurzov

Ako sme spomenuli, stránka Niké pre získanie aktuálnych kurzov pre konkrétnu udalosť volá endpoint `/prematch/v2/boxes/extended/bet-number/<id>`. Vidíme, že identifikátor uda-

losti je v tomto prípade priamo súčasťou URL, no nie ako parameter. Pri pokusoch vložiť do URL niekoľko ID naraz vždy dostaneme od servera chybovú odpoveď. Tieto dotazy však potrebujeme volať často a pre každú udalosť. Bolo by dobré mať možnosť znížiť počet dotazov, aby sme predišli tomu, že nás stránka začne blokovať. Predtým, ako sa rozhodneme použiť pre získanie kurzov toto URL, je dobré skúsiť alternatívne možnosti. Vieme, že udalosti sú jednoznačne identifikované identifikátorom `sportEventId` tak, ako sme videli v odpovedi 2. Ak skúsime použiť rovnaké URL ako v predchádzajúcom prípade, ale zmeníme parameter `boxId` na `sportEventId`, získame URL `/prematch/boxes/mobile?sportEventId=<id>`. Ak skúsime vytvoriť dotaz `GET` na toto URL a predáme platné ID udalosti, ktoré sme získali predchádzajúcim dotazom, zistíme, že dostávame platnú odpoveď. Keďže pre získanie ponúk a kurzov pre konkrétnu udalosť používame rovnaký endpoint, nie je zrejme úplným prekvapením, že schéma JSON odpovede je v obidvoch prípadoch rovnaká. Rozdiel je však v tom, že každý z dotazov ponúkne v odpovedi iné dáta. Pri dotaze na zápasy v danej lige vidíme v odpovedi tiež položku `bets`. Tá však v tomto prípade neobsahuje všetky ponúkané stávky, ale iba tie, ktoré sú viditeľné na karte udalosti v náhľade ligy, ako je zobrazené na obrázku 3.4.



Obr. 3.4: Náhľad zápasov v lige

Ak však zavoláme endpoint s parametrom `sportEventId`, v odpovedi dostaneme v poli `bets` oveľa viac záznamov. Vieme teda, že pre získanie kurzov môžeme použiť tento endpoint miesto toho, ktorý používa stránka pri používaní z prehliadača. Hlavným dôvodom ale bolo znížiť počet dotazov. Ak skúsime tomuto parametru predať viac hodnôt ako v predošlom prípade, vidíme, že opäť dostávame platnú odpoveď. Opäť musíme manuálne overiť, že dáta, ktoré sme dostali korešpondujú s dátami na stránke tak, ako sú zobrazené v obrázku 3.3. Navyše je nutné overiť, že dostávame stávky pre všetky udalosti, na ktoré sa dotazujeme. Toto overenie je v tomto prípade obzvlášť dôležité, nakoľko používame dotaz, ktorý stránka pri bežnom používaní nepoužíva. Štruktúra odpovede, ktorú takýmto dotazom získame je znázornená v ukážke 3.

```
{
 "bets": [
  {
   "boxIds": [...],
   "sportEventId": "m:27057480",
   "header": "Number of Goals",
   "game": "Prematch",
   "selections": [
    {
     "name": "less than 2.5",
     "odds": 1.79,
     ...
    },
    {
     "name": "more than 2.5",
     "odds": 1.88,
     ...
    }
   ],
   "section": "Prematch",
   "participants": [
    "Arsenal FC",
    "Valencia"
   ],
   "sportId": "1"
  }
 ]
}
```


Ukážka 3: Odpoveď na dotaz pre získanie dostupných stávok pre danú udalosť

Informácie, ktoré sú pre nás v tomto prípade zaujímavé, sú jednotlivé položky pola `bets`. Každá položka obsahuje tieto informácie. `sportEventId` - je už spomínané ID udalosti. `header` - podľa hlavičky vieme zistiť o aký typ stávky sa jedná. V našom prípade budeme hľadať stávky na víťaza zápasu, víťaza daného polčasu a počet gólov. Samotné tipy a kurzy sa potom v rámci každej položky nachádzajú v poli `selections`.

V tomto momente teda vieme, ako získať informácie o všetkých zápasoch, ktoré v danom momente stávková kancelária ponúka, ako aj všetky ponúkané stávky a výšky kurzov jednotlivých tipov.

3.2.4 Získavanie výsledkov stávok

Výsledky zápasov nie sú viazané na konkrétnu stávkovú kanceláriu a zodpovedajú realite. Preto nie je potrebné získavať výsledky od každej stávkovej kancelárie zvlášť. Stačí nájsť jeden obsiahly a spoľahlivý zdroj výsledkov a ten potom použiť pre vyhodnocovanie všetkých zápasov.

V našom prípade použijeme službu stávkovej spoločnosti *Pinnacle*. Hlavnými dôvodmi sú, že spoločnosť poskytuje verejne dostupnú a zdokumentovanú API, a zároveň sa jedná o jednu z najväčších stávkových kancelárií na svete čo sa týka objemu zápasov, ktoré pokrývajú. Výsledky, ktoré získame týmto spôsobom, by mali pokrývať drvivú väčšinu zápasov, ktoré nazbierame z ostatných stávkových kancelárií. Ak sa vyskytnú zápasy, pre ktoré nevieme získať výsledky z tohto zdroja, môžme dané zápasy považovať za „nedostatočne zaujímavé“. Táto služba poskytuje iba skóre jednotlivých tímov pre konkrétne periódy zápasu. Nie je ju teda možné použiť pre štatistické stávky, napríklad „počet striel na bránu“. Pre naše účely však postačuje, pretože pomocou skóre vieme vyhodnotiť väčšinu typu stávok ako sú *1x2*, *počet gólov*, *párne/nepárne*.

Pre získavanie výsledkov pošleme GET dotaz na endpoint `/v1/fixtures/settled`[7]. Jeho jediným povinným parametrom je ID športu, pre ktorý chceme výsledky získať. Tento parameter prijíma iba jednu hodnotu, takže pre získanie všetkých výsledkov túto metódu budeme musieť volať pre každý šport. Endpoint navyše umožňuje získavať výsledky iba pre konkrétne ligy, ale tento parameter nepoužijeme, nakoľko sa snažíme získať všetky dostupné výsledky, a to čo najmenším počtom dotazov. Ďalej je možné poskytnúť parameter `since`. Ten je určený pre inkrementálne aktualizácie, teda v odpovedi by sme dostali iba výsledky, ktoré pribudli od dátumu a času špecifikovaného v parametri `since`. Tento parameter je vhodné použiť ak by sme chceli získavať výsledky s nízkou odozvou a často volali túto metódu. Znížila by sa tak veľkosť odpovedí. Nám výsledky stačí získavať oneskorene, takže tento parameter nebudeme využívať. Zber týchto dát budeme vykonávať s veľmi nízkou frekvenciou. Raz za 12 hodín bude postačujúce na to, aby sme nazbierali všetky výsledky, čo potrebujeme. Zároveň však ponechá určitú rezervu v prípade výpadku niektorých služieb bez toho, aby sme niektoré výsledky zmeškali.

V odpovedi získame zoznam všetkých zápasov (aj jednotlivých periód), ktoré boli vyhodnotené za posledných 24 hodín. Formát odpovede je v ukážke 4

[7] všetky dotazy na pinnacle API majú prefix api.pinnacle.com

```
{
 "sportId": 0,
 "last": 0,
 "leagues": [
  {
   "id": 0,
   "events": [
    {
     "id": 0,
     "periods": [
      {
       "number": 0,
       "status": 0,
       "settlementId": 0,
       "settledAt": "2017-09-03T18:21:22.3846289-07:00",
       "team1Score": 0,
       "team2Score": 0,
       "cancellationReason": {...}
      }
     ]
    }
   ]
  }
 ]
}
```


Ukážka 4: Získavanie výsledkov zápasov

Pre každý zápas každej ligy v odpovedi dostávame pole `period`, v ktorom sú výsledky jednotlivých polčasov a celého zápasu. `number` indikuje číslo periódy. `status` potvrdzuje správnosť výsledkov. `team1Score` a `team2Score` je počet gólov jednotlivých tímov.

Pre získanie ID športu použijeme `GET` metódu na endpoint `/v2/sports`, ktorý neprijíma žiadne parametre. V odpovedi dostávame v poli `sports` zoznam položiek, ktoré reprezentujú jednotlivé športy. Nakoľko chceme zbierať všetky výsledky len pre futbal a je veľmi nepravdepodobné, že stávková kancelária zmení ID tohto športu, úplne postačí ak metódu zavoláme ručne a ID športu uložíme priamo do zdrojového kódu.

```
{
 "sports": [
  {
   "id": 0,
   "name": "string",
   "hasOfferings": true,
   "leagueSpecialsCount": 0,
   "eventSpecialsCount": 0,
   "eventCount": 0
  }
 ]
}
```

Ukážka 5: Športy v ponuke kancelárie Pinnacle

3.2.5 Rozlišovanie totožných zápasov z rôznych zdrojov

Na to aby sme nazbierané dáta mohli nejakým spôsobom vyhodnocovať potrebujeme mať informáciu o výsledkoch zápasov, pre ktoré sme zbierali vývoje cien kurzov. Keďže výsledky zápasu sú na rozdiel od kurzov vždy rovnaké pre jeden zápas bez ohľadu na to, v ktorej stávkovej kancelárii bol daný zápas v ponuke, stačí zbierať výsledky zápasov iba z jedného zdroja. Navyše nie každá stávková kancelária zverejňuje konkrétne výsledky zápasu a väčšinou bude zverejňovať iba výsledky konkrétnych stávok, a aj to iba pre hráčov, ktorí stávku zrealizovali. V našom prípade ako náš jediný zdroj výsledkov používame API od kancelárie *Pinnacle*. Tieto výsledky vieme potom v databáze priamo namapovať pomocou interného ID na konkrétne zápasy. Potrebujeme teda nejaký mechanizmus, ktorý dokáže určiť, že dve ponuky z rôznych stávkových kancelárii sa odkazujú na ten istý zápas v skutočnom svete.

V kontexte detekcie totožných udalostí považujeme za zdroj pravdy dáta spoločnosti *Pinnacle*, pretože ich ponuka je najobsiahlejšia, a preto, že výsledky zápasov vieme priamo priradiť k jednotlivým zápasom podľa interného ID spoločnosti bez toho, aby sme museli vzťah nejakým spôsobom odvodzovať. Odvodzovanie vzťahu medzi zápasmi v ponuke ostatných kancelárií vzhľadom k zápasom v ponuke *Pinnacle* bude založené na podobnosti reťazcov jednotlivých atribútov porovnávaných zápasov a bližšie je popísané v sekcii 4.1.2

3.3 Získavanie znalostí a analýza nazbieraných dát

Dáta, ktoré sme získali zo stávkových kancelárií teraz máme príležitosť analyzovať a pokúsiť sa z nich extrahovať informácie. Analyzovali sme doménu a vybrali sme dátovú sadu, z ktorej sa budeme pokúšať informácie získavať. V tejto prípadovej štúdii sa zameriame na vývoje kurzov spoločnosti Pinnacle. Dôvodom je objem nazbieraných dát a objem stávok, ktoré táto spoločnosť denne uzatvára v porovnaní s domácim trhom. V nazbieraných dátach sa pokúsime vyhľadať frekventované vzory vývoja kurzov a vzťah medzi týmito frekventovanými vzormi a výsledkami zápasov. Druhou úlohou bude štatistické vyhodnotenie celkového rozdielu kurzu od prvotného zverejnenia zápasu a kurzu tesne pred začiatkom zápasu a závislosť výsledku zápasu od tejto zmeny. Toto je metrika, ktorú mnoho stávkarov používa pri rozhodovaní. Podľa definície procesu získavania znalostí z kapitoly 2.1 zodpove-

dajú doterajšie kroky s krokom 1 a 2 tohto procesu. Nakoľko máme vybranú zvolenú dátovú sadu, nasleduje ďalší krok, ktorým je predspracovanie dát.

3.3.1 Predspracovanie a čistenie

Keďže sledujeme závislosti výsledkov udalostí a vývojov kurzov, jedným z krokov čistenia je, že odstránime všetky zápasy, následne aj všetky nazbierané ponuky pre tieto zápasy, pre ktoré nepoznáme výsledok. V tomto prípade nemáme inú možnosť ako na chýbajúce hodnoty reagovať, nakoľko výsledky zápasov nemôžu byť žiadnym spôsobom odvodené z existujúcich dát. Ďalším krokom čistenia je odstránenie všetkých ponúk, ktoré boli ponúknuté po čase začiatku zápasu. V tomto prípade odstraňujeme iba ponuky, nie samotné zápasy, nakoľko vývoje kurzov pred samotným začiatkom zápasu sú platné dáta, o ktoré máme záujem. Ďalšou podmienkou, ktorá musí platiť je, že všetky kurzy majú hodnotu vyššiu ako 1.0 počas celého vývoja. Ak pre niektorú ponuku kurzy nadobúdali hodnotu menšiu alebo rovnú 1.0, pravdepodobne bol zápas zrušený alebo odložený. Tieto ponuky aj zápasy, ku ktorým sa viažu môžeme odstrániť. Nakoniec môžeme odstrániť aj všetky zápasy, pre ktoré nemáme žiadne ponuky kurzov. V tomto momente by dáta mali byť vyčistené, ale ako bolo spomenuté v kapitole 2.1, tento proces je iteratívny a je možné, že sa k nemu pri samotnej implementácii vrátime.

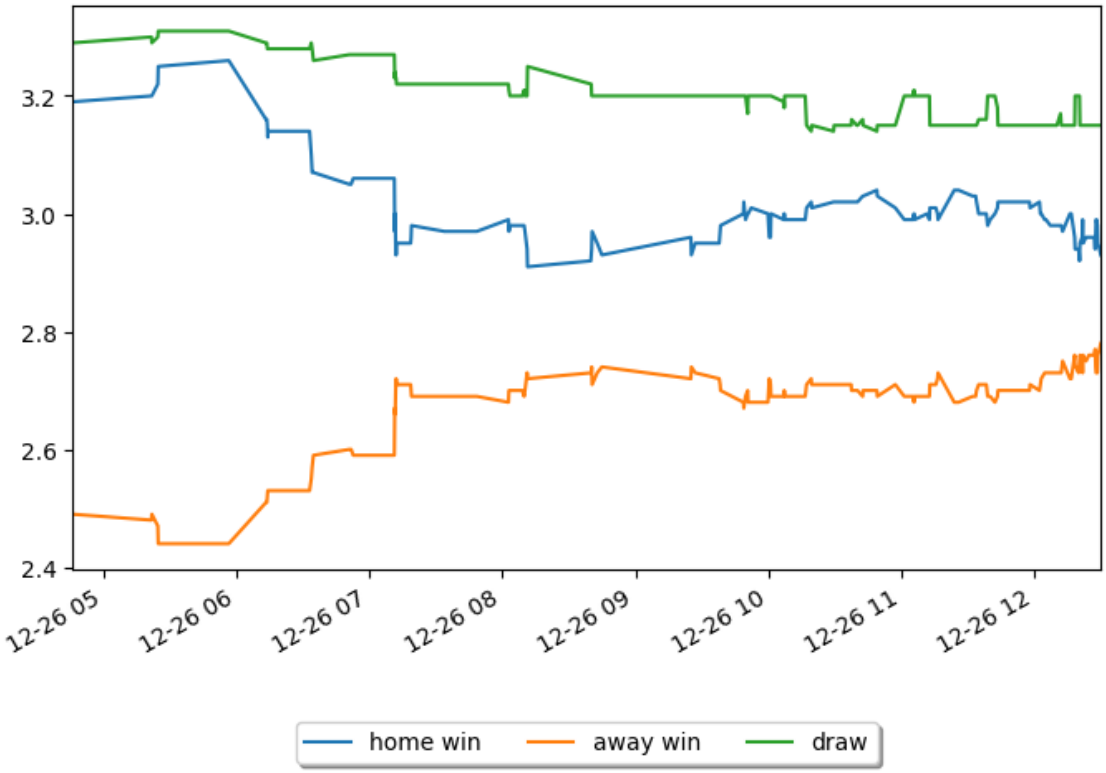
Nasledujúcim krokom je transformácia dát. Tento krok bude pomerne obsiahly, ale je potrebné, aby sme dáta transformovali do takej podoby, aby sme s nimi vedeli jednoducho a hlavne efektívne pracovať. Jedným z hlavných cieľov bude minimalizovať počet databázových operácií, ktoré je potrebné s nad transformovanými dátami vykonávať počas data miningu. Docielime toho hlavne tak, že štruktúru dát denormalizujeme a vytvoríme plochú štruktúru.

3.3.2 Transformácia

Prvým krokom transformácie je agregácia vývojov kurzov pre jednotlivé ponuky. Pre každú ponuku z tabuľky `OFFER` získame všetky kurzy. Získavame tým časovú radu s 3 atribútmi `ODDS1`, `ODDS2`, `ODDS3`. Túto časovú radu následne rozdelíme tak, aby vývoj kurzov pre každú z možností bol reprezentovaný samostatnou radou. Napríklad pre ponuku tím A vs. tím B `1x2` získame 3 časové rady, kde prvá časová rada bude reprezentovať vývoj kurzu na možnosť vyhrá tím A (1), druhá časová rada reprezentuje možnosť vyhrá tím B (2) a tretia remíza (x). Príklad takýchto časových rád vidíme na obrázku 3.5.

Takouto transformáciou získame časové rady, pričom každá bude reprezentovať vývoj kurzu jedného tipu jednej konkrétnej stávky pre jednu udalosť. Príklad takejto časovej rady by bol vývoj kurzov pre tip „Vyhrá domáci“ pre stávku typu `1x2`. Máme teraz k dispozícii časové rady, kde každá reprezentuje vývoj kurzov pre jeden konkrétny tip. Každú časovú radu teraz transformujeme tak, že z dostupných hodnôt ponecháme iba lokálne extrémy. Týmto získame rady, ktoré reprezentujú to, akým spôsobom sa kurz menil v kompaktnejšom formáte. Odstránime týmto úseky v časových radách, kde sa kurz nemenil a zároveň agregujeme čiastočné prírastky alebo úbytky na jednu hodnotu. Príklad takto transformovanej rady znázorňuje obrázok 3.6.

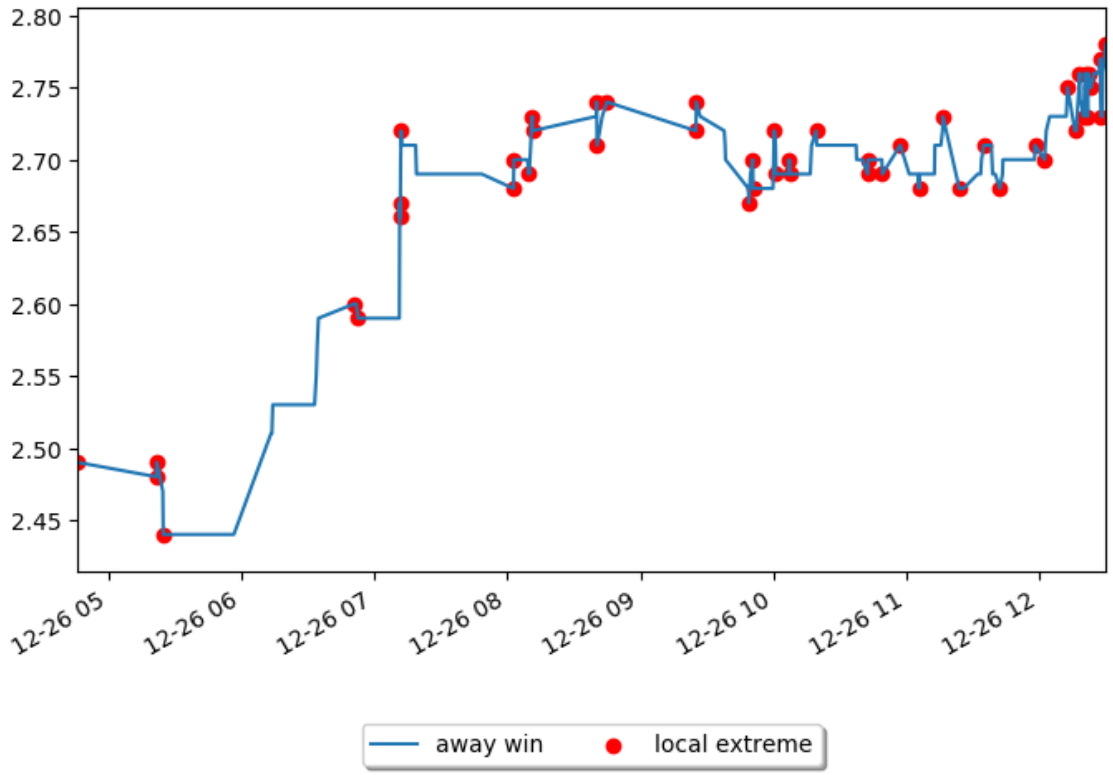
Následne túto časovú radu transformujeme na radu relatívnych rozdielov po sebe nasledujúcich hodnôt. Získame tak rady, kde každá hodnota reprezentuje relatívnu zmenu kurzu voči poslednému lokálnemu extrému. Zároveň v tomto kroku zahadzujeme informáciu o čase a získavame tak namiesto časovej rady postupnosť zmien, ako vidíme na obrázku 3.7. Na y-ovej osi už nie sú časové značky, ale iba poradové čísla dátových bodov. Normalizá-



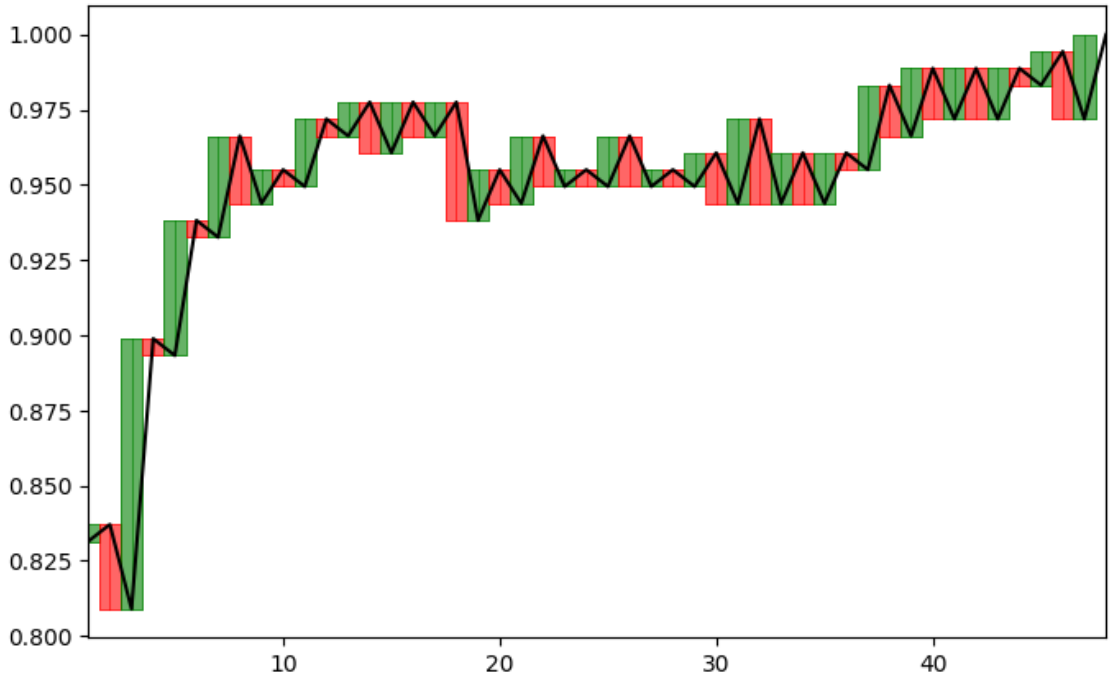
Obr. 3.5: časová rada 3v1

ciou sme zaistili, že rovnaké absolútne prírastky pri nižších kurzoch majú vyššiu relatívnu hodnotu ako rovnaké absolútne prírastky pri vyšších kurzoch.

Postupnosti, ktoré máme k dispozícii sa stále viažu na jeden konkrétny tip. V ďalšom kroku týmto možnostiam priradíme výsledky. Výsledky budeme reprezentovať binárne (ako vieme, stávky, ktoré sme zbierali, môžu mať len dva výsledky - výhra alebo prehra stávkujúceho). Inými slovami, ku každej časovej rade priradíme výsledok podľa toho, či táto možnosť bola pre danú stávku výherná. Keďže v predchádzajúcom kroku predspracovania sme vyradili udalosti, pre ktoré nepoznáme výsledky, vieme, že pre každú z časových rád, ktoré nám ostali vieme zistiť výsledok. Výsledky udalostí sa nachádzajú v tabuľke `SETTLEMENT`. Pre každú možnosť musíme tento výsledok vyhodnotiť. Napríklad, ak máme časovú radu, ktorá sa viaže na možnosť padne viac ako 3.5 gólov pre stávku typu počet gólov v prvom polčase a z tabuľky settlement vidíme, že stav v prvom polčase pre danú udalosť bol 1:1, vyhodnotíme túto možnosť ako prehrávajúcu a priradíme do tohto atribútu hodnotu 0. V tomto momente nám iba ostáva takto transformované dáta uložiť. Schéma tabuľky pre predspracované dáta je na obrázku 3.8. Vidíme, že výsledný formát je veľmi kompaktný napriek tomu, že sme neprišli o žiadne potrebné informácie. `OFFER_ID` odkazuje na pôvodnú ponuku. `ODDS` je zoradené pole pôvodných kurzov serializované ako reťazec, `ODDS_DIFFS` obsahuje našu serializovanú časovú radu zakódovanú ako reťazec. `OUTCOME` je výsledok danej stávky, má binárnu hodnotu. Tieto dáta predávame ďalej do fázy dataminingu.



Obr. 3.6: Filtrované extrémy časovej rady



Obr. 3.7: Postupnosť relatívnych rozdielov

Obr. 3.8: Formát dát po predspracovaní

3.3.3 Data mining

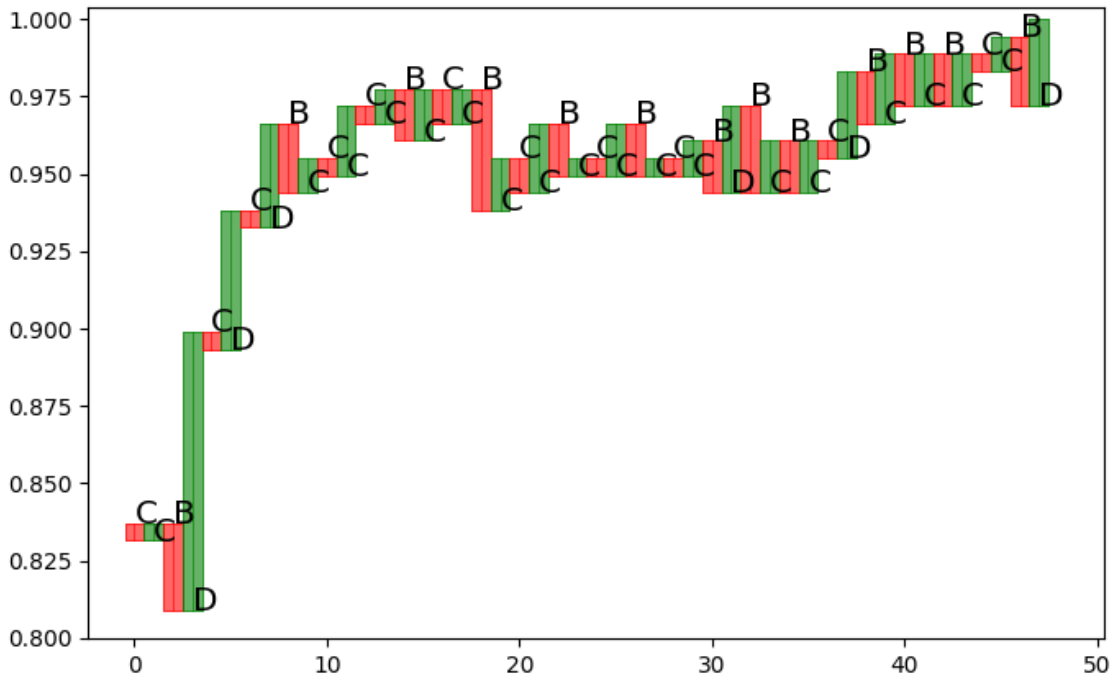
Ako bolo spomenuté v kapitole 2.1, úlohy dolovania dát delíme na deskriptívne a prediktívne. Našim zámerom je pozorovať vzory vo vývoji dát a závislosti medzi výsledkami, jedná sa teda o deskriptívny druh dataminingu. V našom prípade budeme medzi vývojmi kurzov hľadať frekventované vzory pomocou apriori algoritmu. Následne budeme štatisticky vyhodnocovať príležitosti, u ktorých takýto vývoj nastal. Pre porovnanie tiež zhodnotíme jednoduchý celkový rozdiel otvárajúceho a záverečného kurzu.

Príprava dát pre apriori algoritmus

Cieľom tohto algoritmu je nájsť často vyskytujúce sa vzory. Ako prvý krok potrebujeme dáta pripraviť pre algoritmus. Relatívne zmeny kurzov potrebujeme previesť na symboly, pričom každá hodnota bude reprezentovaná jedným z N symbolov (N je parametrom nášho data mining algoritmu). Z transformovaných dát vyberieme záznamy, kde výsledok stávky je 1, nakoľko sa pokúšame nájsť často vyskytujúce sa správanie pre víťazné prípady. Na základe všetkých relatívnych zmien z týchto záznamov spočítame q-kvantily, pričom $q = N$. Inými slovami, interval hodnôt rozdelíme tak, aby jednotlivé intervaly mali rovnakú hĺbku. Každému intervalu, ktorý takto získame, priradíme jeden symbol. Získame tak tabuľku prevodov zmien kurzov na symboly. Pomocou tejto tabuľky zakódujeme všetky vývoje kurzov vo vybraných záznamoch. Vývoje kurzov tak budú reprezentované postupnosťou symbolov. Príklad zakódovaného vývoja kurzov vidíme na obrázku 3.9. Sekvencia bola zakódovaná pomocou tabuľky 3.2.

zmena	symbol
$(-\infty, -0.1212)$	A
$< -0.1212, -0.0124)$	B
$< -0.0124, 0.0231)$	C
$< 0.0231, 0.1114)$	D
$< 0.1114, \infty)$	E

Tabuľka 3.2: Príklad tabuľky prevodov pre 5 symbolov



Obr. 3.9: Prevedenie na symboly

Hľadanie frekventovaných vzorov pomocou apriori algoritmu

Tento algoritmus použijeme pre vyhľadanie frekventovaných vzorov v sekvenciách symbolov, ktoré sme získali zakódovaním jednotlivých postupností relatívnych zmien pomocou tabuľky prevodov. Výstupom algoritmu sú sekvencie, ktoré spĺňajú minimálnu podporu a spoľahlivosť. Podpora v našom prípade hovorí o tom, s akou pravdepodobnosťou sa vzor nachádza v množine vzorov pre víťazné stávky.

Algoritmus pracuje nasledovne:

1. Zvolíme minimálnu podporu. Táto hodnota môže byť absolútna alebo relatívna.

2. Vytvoríme množinu všetkých frekventovaných vzorov o dĺžke 1, teda množinu symbolov, ktoré sa medzi všetkými sekvenciami vyskytujú s minimálnou podporou. Keďže jednotlivé sekvencie sme kódovali do symbolickej podoby na základe kvantilov, vieme, že všetky symboly sa vyskytujú približne v rovnakom počte. Ako počiatočnú množinu frekventovaných vzorov teda zvolíme množinu všetkých symbolov. V tomto kroku máme teda množinu vzorov, kde každý má podporu približne $1/N$, pričom N je počet symbolov z tabuľky prevodov.

3. Každý z frekventovaných vzorov rozšírime nasledovným spôsobom. Z množiny frekventovaných vzorov vyberieme všetky vzory, ktoré majú s vybraným vzorom spoločný prefix o dĺžke $K - 1$, kde K je dĺžka vzoru. Pre daný vzor vygenerujeme kandidátne vzory tak, že daný vzor rozšírime o posledný znak reťazca, s ktorým vzor zdieľa prefix. Množinu kandidátnych vzorov rozšírime o takto získané vzory.

4. Z kandidátnych vzorov vyradíme vzory, pre ktoré neplatí, že všetky podreťazce sú frekventované vzory. V praxi to znamená, že pre každý kandidátny reťazec skontrolu-

jeme, či sa všetky jeho podreťazce dĺžky $K-1$, kde K je dĺžka kandidátneho vzoru, nachádzajú v predchádzajúcej množine frekventovaných vzorov.

5. Z množiny kandidátnych vzorov získame množinu frekventovaných vzorov dĺžky K tak, že vyradíme vzory, ktoré nespĺňajú minimálnu podporu. Ak je nová množina prázdna, vraciame predchádzajúcu množinu frekventovaných vzorov.

6. Opakujeme od kroku 3 pre novú množinu frekventovaných vzorov.

Zo získaných frekventovaných vzorov môžeme následne odvodiť asociačné pravidlá v tvare $A => win$, a následne z týchto pravidiel ponechať len tie, ktoré spĺňajú minimálnu spoľahlivosť. Spoľahlivosť pre frekventovaný vzor A spočítame ako podmienenú pravdepodobnosť

$$P(win|A)$$

Minimálna podpora bude vzhľadom na sémantiku našich dát 33.3% pre stávky `1x2` a 50% pre stávky `počet gólov`.

Ako už bolo spomenuté v kapitole 2.1, jedná sa o iteratívny proces. Parametre, ako je počet znakov použitých pre kódovanie sekvencií, minimálna podpora, minimálna spoľahlivosť (tá vychádza zo samotnej domény a cieľov), minimálna dĺžka frekventovaného vzoru a pod. môže byť nutné upravovať, aby sme sa dostali k použiteľným výsledkom. Výsledky experimentov s rôznymi parametrami sú uvedené v kapitole 5.

Pre analýzu zmeny kurzu voči otvárajúcemu kurzu opakujeme tie isté kroky KDD procesu ako pri hľadaní frekventovaných vzorov. Rozdielom je, že fázu vzorkovania po lokálnych extrémoch vynechávame. Miesto toho berieme prvú a poslednú vzorku časovej rady. Kódovanie na symboly taktiež nie je nutné. Histogramy pre nazbierané dáta sú ukázané v kapitole 5.

Kapitola 4

Implementácia navrhnutého riešenia

V tejto kapitole budú popísané implementačné detaily vybraných prvkov systému navrhnutého v kapitole 3. Budú spomenuté niektoré rozdiely medzi návrhom a implementáciou, ako aj zdôvodnenie týchto rozdielov a prípadne nepredvídané situácie, s ktorými bolo nutné sa vysporiadať. Budu opísané použité algoritmy, dátové štruktúry, knižnice a programové štruktúry jazyka Python použité pri riešení jednotlivých úloh.

4.1 WebScraping

V tejto sekcii bude opísaná samotná implementácia jednotlivých krokov procesu zberu a ukladania dát.

4.1.1 Framework

Pre zber dát bol navrhnutý jednoduchý framework, ktorý dovolí rozšírenie o zber dát z ďalších zdrojov bez nutnosti meniť celkovú štruktúru systému. Jadrom systému je databáza, do ktorej framework zapisuje nazbierané údaje z jednotlivých zdrojov. Zároveň je táto databáza zdrojom dát pre fázu dataminingu. Užívateľskými jednotkami v tomto frameworku sú jednotliví boti, ktorí definujú, ako sa budú dáta z jednotlivých stránok získavať. Hlavnými zodpovednosťami frameworku sú:

1. Spracovávanie a validácia dát od botov a následné ukladanie dát do databázy
2. Vytváranie HTTP dotazov na základe požiadaviek botov.
3. Spúšťanie konkrétnych procesov jednotlivých botov v daných intervaloch.
4. Vytváranie metrík a štatistík
5. Interakcie s databázou, dávkové vkladanie dát.

Práca s databázou

Návrh databázy v sekcii 3.2.2 je dobrý z hľadiska odporúčaného návrhu databáze a normalizácie. Databáza by bola vhodná pre ukladanie veľkého množstva dát ak by sme dopredu poznali všetky väzby medzi dátami a vedeli by sme ktoré zápasy sú identické. Ak by sme

túto schému databázy použili pri implementácii, zistili by sme, že takáto schéma kladie isté obmedzenia na spôsob, akým sme schopní dáta získavať. Výsledná databáza použitá pre implementáciu sa teda mierne odlišuje. Schéma použitej DB je na obrázku 4.1 a vzhľadom na pôvodný návrh má túto sémantiku:

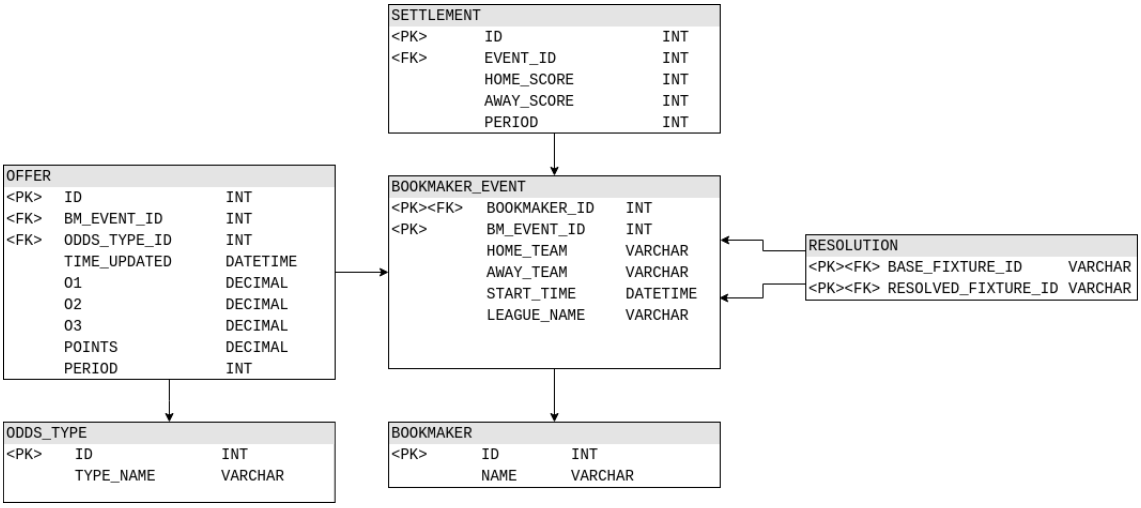
1. Tabuľky `BOOKMAKER_EVENT` a `FIXTURE` boli zlúčené do jednej tabuľky. Dôvodom je, že pri pôvodnom návrhu nebolo možné vložiť záznam do tabuľky `BOOKMAKER_EVENT` bez toho, aby sme vedeli na akú reálnu udalosť odkazuje. To teda vyžadovalo pre každý novoobjavený zápas v ponuke niektorej z kancelárií detekovať, na ktorý zápas reálneho sveta odkazuje. Navyše, táto detekcia musela byť vykonaná okamžite. Zlúčením tabuliek sme docielili toho, že každý novoobjavený zápas uložíme do spojenej tabuľky `BOOKMAKER_FIXTURE` s dátami, ktoré ponúka stávková kancelária a detekciu totožných zápasov z rôznych kancelárií vykonáme neskôr.

2. Tabuľky `MARKET_OFFER` a `PRICE_UPDATE` boli zlúčené do jednej. Hlavným dôvodom je, že to výrazne zjednodušuje implementáciu nášho mini frameworku a botov samotných. V prvom rade to zjednoduší prípad, kedy sa konkrétna ponuka pre konkrétny zápas objaví prvýkrát. Nie je nutné riešiť logiku, ktorá vytvorí záznam pre `MARKET_OFFER` ak neexistuje. V ostatných prípadoch vytvárať iba `PRICE_UPDATE` záznamy. Ďalšou výhodou je, že bot, ktorý bude zbierať informácie o kurzoch a generovať záznamy pre tabuľku `PRICE_UPDATE` nepotrebuje zisťovať ID záznamu `MARKET_OFFER`, ku ktorému sa tieto kurzy viažu.

3. Vytvorená bola tabuľka `RESOLUTION`. Túto použijeme pre ukladanie informácie o tom, ktoré udalosti od rôznych stávkových kancelárií sa zhodujú. Nakoľko výsledky zápasov zbierame od Pinnacle, použijeme zápasy od Pinnacle ako smerodajné. V tabuľke `RESOLUTION` teda budu záznamy, ktoré mapujú zápasy z kancelárií iných ako Pinnacle, na jeden konkrétny zápas Pinnacle.

4. Použitím zlúčenej tabuľky `OFFER` môže bot priamo generovať záznamy do tejto tabuľky z dát, ktoré získa z webu bez toho, aby potreboval nejaké informácie navyše z databázy. Záznamy v tabuľke `OFFER` sa viažu na záznamy `FIXTURE` pomocou názvu stávkovej kancelárie a interného ID danej udalosti u tejto stávkovej kancelárie. Dôležité je podotknúť, že toto integritné obmedzenie nie je vynucované. Znamená to, že záznam `OFFER` je možné vložiť bez toho, aby existoval príslušný záznam `FIXTURE`. To je z dôvodu, že nové zápasy sú objavované v pomalšom intervale, ako sú získavané aktuálne kurzy. Takto máme istotu, že nezahadzujeme kurzy, pretože druhý proces ešte neobjavil daný zápas a že kurzy zbierame od samého začiatku odkedy sú v ponuke stávkovej kancelárie. Toto bude mať vplyv najmä pre dáta Pinnacle. Keďže endpoint vracia všetky zmeny kurzov poslednej aktualizácie, nepotrebujeme poznať ID jednotlivých zápasov, a nie je teda problém získavať kurzy skôr, ako sme schopní identifikovať, ku ktorým zápasom sa vzťahujú. V prípade dát Niké táto absencia integritných obmedzení de facto nehrá úlohu, nakoľko kurzy začneme získavať až po tom, čo `discover` proces „objaví“ identifikátory jednotlivých zápasov a tie uloží do cache spolu s vložením spracovaných zápasov do databázy. `pull` proces potom môže získavať kurzy až na základe týchto identifikátorov v cache.

Ako databázový backend náš framework používa SQLite. Jedná sa o jeden z najpoužívanejších databázových systémov na svete (podľa vývojárov najpoužívanejší) [31]. Výhodou pre naše použitie okrem výbornej podpory online je, že sa jedná o vstavaný systém, ktorý



Obr. 4.1: Schéma databázy pre zber dát

funguje bez servera a jeho implementácia je založená na čítaní a zápise do súboru. Tento SRBD bol teda zvolený najmä kvôli jednoduchosti nasadenia. Jednotlivé funkcie frameworku pre prácu s DB sú implementované v triede `SqliteAdapter`.

Databáza je zároveň použitá ako dočasné úložisko pre transformované dáta pre použitie pri extrakcii znalostí z týchto dát. Keďže v tejto fáze sa pracuje s veľkým objemom dát naraz, bolo nutné uskutočniť isté optimalizácie. Po predspracovaní dát pre data mining budeme tieto dáta ukladať do databázy. Môžeme očakávať, že sa bude jednať o pomerne veľké množstvo dát. Knižnica pre prácu s SQLite databázou bohužiaľ neimplementuje metódu, ktorá vloží do databázy naraz viacero záznamov. V spoločnom rozhraní pre databázy jazyka Python[13] existuje metóda `insert_many()`. Tá je bohužiaľ v knižnici SQLite implementovaná tak, že prijíma zoznam položiek, ktoré chceme vložiť, no vkladá ich po jednom. Pre optimalizáciu tohto procesu bolo nutne navrhnúť vlastnú metódu, ktorá vygeneruje SQL dotaz, ktorý vloží naraz niekoľko záznamov. Nakoľko maximálny počet záznamov, ktoré je možné vložiť jedným volaním metódy `INSERT` je v SQLite 500 záznamov, bolo zároveň nutné implementovať metódu, ktorá veľkú vstupnú kolekciu rozdelí na podkolekcie o maximálnej veľkosti 500 a následne tieto podkolekcie vloží naraz do databázy. Táto optimalizácia mala významný dopad na rýchlosť fázy predspracovania dát.

4.1.2 Získavanie dát

Ako už bolo spomenuté, samotné získavanie dát je implementované v užívateľských triedach (botoch), ktoré rozširujú triedu *Bookmaker*. Samotné získavanie stávkových kurzov rozdelíme na dva procesy. Nazveme ich `pull_odds` a `discover_events`. `discover_events` process bude mať za úlohu prehľadať štruktúru stránky a identifikovať a vyhľadať na nej všetky udalosti, ktoré nás zaujímajú. V našom prípade sa bude jednať o všetky futbalové zápasy. Každú takúto udalosť spracuje a vytvorí objekt triedy `Fixture`. Táto trieda je izomorfná so záznamami v tabuľke `BOOKMAKER_FIXTURE`. Kolekciu týchto objektov následne odovzdá frameworku, ktorý tieto údaje vloží do databázy do tabuľky `BOOKMAKER_FIXTURE`. Pre každý objekt `Fixture` zároveň vráti url, prípadne iba hodnotu url parametra, pomocou ktorého je možné získať konkrétne stávky a kurzy pre danú udalosť. V prípade statických stránok sa môže jednať o cele URL, prípadne URL sufix daného zápasu alebo v prípade dy-

namických stránok môže ísť o hodnotu `POST` alebo `GET` parametra. Tieto hodnoty framework uloží do cache, kde budú prístupné `pull` procesu.

`pull` proces bude mať za úlohu získať aktuálne ceny kurzov pre udalosti, ktoré boli objavené `discover` procesom. Proces získa informácie o tom, odkiaľ má tieto dáta získať z cache. Pre jednotlivé záznamy z cache získa dostupné dáta z webstránky. Tie potom spracuje a vytvorí z nich kolekciu objektov triedy `Offer`. Táto trieda je izomorfná so záznamami v tabuľke `OFFER`. Túto kolekciu následne predá frameworku, ktorý túto kolekciu vloží do databázy do tabuľky `OFFER`.

Takéto rozdelenie na dva procesy má za následok, že pre získanie kurzov nemusíme zakaždým prehľadávať celú stránku, ale môžeme pre každú udalosť priamo dotazovať adresu pre kurzy. Zároveň sa nám ponúka možnosť definovať rôzne frekvencie procesov pre dáta, ktoré sa menia menej často (zápasy v ponuke) a dáta, ktoré sa menia často (ceny kurzov). To má za následok ďalšie zníženie počtu zbytočných dotazov. Znížením počtu zbytočných dotazov znižujeme taktiež pravdepodobnosť, že stávková kancelária detekuje túto aktivitu a vystaví na naše pripojenie nejaké obmedzenie.

V prípade niektorých stránok môžeme docieliť ďalšie zníženie počtu dotazov tým, že ceny kurzov budeme získavať po dávkach. To je samozrejme podmienené tým, že samotný webserver danej stránky niečo takéto dovoľuje. Toto je väčšinou možné len pre stránky ktoré sú dynamické. Ako sme zistili v kapitole 3, práve webserver stránky *nike.sk* umožňuje dotazovať sa o kurzy pre niekoľko udalostí naraz. Tento fakt zohľadníme aj pri implementácií `pull` procesu pre tohto bota. Pri získavaní kurzov rozdelíme kolekciu ID zápasov na podkolekcie o maximálnej veľkosti 30 prvkov.[1]. Dotazy na stránku následne budeme posielať naraz pre všetkých 30 zápasov. Týmto výrazne znížime počet vytváraných dotazov.

Pre vytvorenie procesu, ktorý bude dáta zbierať je použitá trieda `BotBuilder` a jej továrenská metóda `build()`, ktorá zostrojí inštanciu triedy `Bot` pomocou niektorej z implementácií triedy `Bookmaker`. Táto inštancia zároveň nesie informáciu o tom, ktorú z implementovaných metód triedy `Bookmaker` má volať a s akým intervalom sa spúšťať.

Rozpoznávanie identických zápasov

Ako už bolo spomenuté, nie každá stávková kancelária nám poskytuje možnosť získať výsledok stretnutia priamo pomocou interného ID danej udalosti. V našom prípade použijeme iba jeden zdroj výsledkov a tým je stávková kancelária *Pinnacle*. Tieto výsledky sa priamo viažu na zápasy v ponuke tejto stávkovej kancelárie ich interným ID. Ľubovolnú stávku v tejto kancelárii teda vieme vyhodnotiť na základe ID udalosti, ku ktorej sa táto stávka vzťahuje. Pre vyhodnotenie podujatia v ponuke inej stávkovej kancelárie teda potrebujeme zistiť, či sa toto isté podujatie nachádza v ponuke stávkovej kancelárie *Pinnacle* a ktoré to je.

Komponentu, ktorá má za úlohu identifikovať totožné zápasy v našom frameworku, nazveme `EventResolver` a funguje následovne. Má jedinú metódu `resolve()`. Ako vstup prijíma udalosť `Fixture`, ktorú sa snažíme identifikovať. Jej výstupom je `Optional[Fixture]`, ktorá v prípade, že existuje udalosť v ponuke *Pinnacle*, je totožná so vstupnou.

Algoritmus pre detekciu totožných udalostí je založený na reťazcovej vzdialenosti. Pre samotné spočítanie podobnosti reťazcov sa nám ponuka niekoľko algoritmov, ako napríklad: Levenshteinova vzdialenosť[14], Damerau-Levenshteinova vzdialenosť[10], Jaro-Winklerova vzdialenosť [4].

[1] Zistené experimentálne

Naša komponenta bude používať práve Levenshteinovu vzdialenosť, pretože knižnica `FuzzyWuzzy`, ktorú pre tieto účely použijeme, používa túto metriku[8][1].

Pre vstupné podujatie komponenta vyberie z databázy kandidátne podujatie na základe týchto kritérií. Zdrojová stávková kancelária pre kandidátne podujatie je `Pinnacle`. Dátum a čas začiatku kandidátneho podujatia je v rozsahu $< -4, +4 >$ hodiny vzhľadom k času a dátumu začiatku vstupného podujatia. Tento časový rozsah je implementovaný preto, že občas sa môže stať, že niektorá stávková kancelária má nesprávne údaje o začiatku stretnutia. Napríklad ak sa podujatie posunie kvôli počasiu a stávková kancelária tento fakt nezohľadní. Časové nezrovnalosti môžu vznikať taktiež kvôli nezrovnalostiam v časových pásmach. Konkrétne dôvody nie sú tak dôležité ako fakt, že k týmto prípadom skutočne dochádza, a to častejšie ako by sa človek mohol domnievať. Týmto zmenšíme prehľadávaný priestor, čím zároveň nechávame určitý priestor pre odchýlky.

Po získaní kandidátnej množiny sa pokúsime nájsť identickú udalosť nasledovne. Pre kandidátne udalosti a vstupnú udalosť vytvoríme množinu identifikátorov pozostávajúcich z názvu domáceho tímu, názvu tímu hostí a názvu ligy. Následne pomocou knihovnej metódy `process.extracOne` vyberieme z kandidátnych udalostí tú najpodobnejšiu vstupnej ak táto splňuje prahovú podobnosť.[1] Prahovú podobnosť nastavíme relatívne vysokú aj za cenu toho, že sa môže stať, že niektoré udalosti nebudeme vedieť rozlíšiť aj napriek tomu, že medzi kandidátnymi sa nachádza totožná udalosť. V našom prípade môže mať „false positive“ oveľa horšie následky a negatívny vplyv na výsledné spracovanie ako to, že niektoré udalosti zostanú nerozpoznané. Pri vývoji tejto komponenty bolo testované rozpoznávanie 200 udalostí od spoločnosti *Niké* na udalosti v ponuke stávkovej kancelárie *Pinnacle*. Pri prahovej hodnote 85 komponenta úspešne rozpoznala 198 týchto udalostí.

4.1.3 Správa a plánovanie procesov

Keďže zber dát nie je jednorázová záležitosť, ale procesy musia bežať dlhodobo, je vhodné zamyslieť sa nad tým, ako budeme jednotlivých botov spravovať a spúšťať. Zároveň by bolo vhodné, aby sa v prípade nejakých neočakávaných udalostí, napríklad dočasný výpadok siete, vedeli boti zotaviť. K dispozícii máme niekoľko systémov pre kontrolu procesov ako *supervisord*[32], *runit*[30] alebo *puppet*[28]. V našom prípade použijeme štandardný systém pre správu unixových služieb, systemd[33] a jednoduchý skript, ktorý bude generovať konfiguračné súbory pre *systemd*.

Systemd nám okrem iného dovoľuje definovať správanie v prípade, že sa kontrolovaný proces nečakane zastaví a skončí s chybou, a taktiež nám umožňuje automaticky spúšťať procesy po prípadnom reštarte servera, na ktorom naše procesy pre zber dát budú bežať (napr. pri dočasnom výpadku elektrického prúdu). *Systemd* bude spravovať procesy pre zber dát od jednotlivých stávkových kancelárií. Popis správania jednotlivých procesov je definovaný v konfiguračných súboroch pre *systemd*. Pre vytvorenie týchto súborov bol vytvorený jednoduchý skript, ktorý na základe zoznamu položiek triedy `BotConfig` vygeneruje súbory pre *systemd*. Trieda `BotConfig` definuje ako často a ktorá metóda ktorého bota má bežať.

V prípade, že teda chceme rozšíriť náš systém o dáta z ďalšieho zdroja, pridať ďalšiu stávkovú kanceláriu, z ktorej budeme zbierať dáta, stačí pridať bota pre túto stávkovú kanceláriu do adresára `data_retrieval/odds/books`. Táto trieda musí rozširovať triedu `Bookmaker` a implementovať abstraktné metódy `get_odds()` a `get_events()`. Ďalej je nutné pridať konfiguráciu do súboru `data_retrieval/scripts/service_config.py`, kde pomocou záznamu triedy `BotConfig` vygenerujeme konfiguračný súbor pre nového bota.

Kapitola 5

Analýza nazbieraných dát

Analýzy boli vykonané na vzorke dát o veľkosti 12 000 stávok typu `1x2` a 12 000 stávok typu `počet gólov` pre futbalové zápasy z ponuky stávkovej kancelárie *Pinnacle*. Výsledky pre zvolené metódy získavania znalostí z dát sú uvedené v nasledujúcich sekciách. Tieto výsledky môžu stávkarom pomôcť pri rozhodovaní pri uzatváraní stávok. Interpretácia týchto výsledkov je do istej miery ponechaná na užívateľoch.

5.1 Frekventované vzory

V tejto sekcii budú prezentované výsledky algoritmu pre dolovanie frekventovaných vzorov pre vybrané parametre. Úpravou parametrov sa snažíme nájsť čo najdlhšie vzory s dostatočnou podporou a spoľahlivosťou, ktorá je lepšia ako náhodná, teda v prípade stávok typu `1x2` viac ako 33% a v prípade stávok typu `počet gólov` viac ako 50%. Tabuľka 5.1 zobrazuje nájdené frekventované vzory, pričom parametre dolovacieho algoritmu boli nasledovné:

- počet kódových symbolov: 6
- minimálna podpora: 0.075
- minimálna dĺžka frekventovaného vzoru: 4
- minimálna spoľahlivosť závisi od typu stávky (viď vyššie)

Stĺpec tabuľky `vzor` obsahuje konkrétne frekventované podvzory nájdené v množine všetkých vzorov, ktoré reprezentujú vývoj cien kurzov. Stĺpec `podpora` obsahuje pravdepodobnosť výskytu tohto vzoru medzi vzormi pre víťazné stávky. Stĺpec `spoľahlivosť` obsahuje podmienenú pravdepodobnosť, že je stávka víťazná ak obsahuje daný vzor. Typ stávky je uvedený v poslednom stĺpci. Podpora a spoľahlivosť je počítaná vrámci stávok rovnakého typu. Vývoje kurzov boli kódované podľa tabuľky prevodov symbolov 5.2.

vzor	podpora	spoľahlivosť	typ stávky
`BEBD`	0.077	0.360	`1x2`
`EBEB`	0.100	0.367	`1x2`
`BEBE`	0.099	0.377	`1x2`
`EBEC`	0.082	0.398	`1x2`
`DCDCE`	0.080	0.513	`počet gólov`
`CDBD`	0.099	0.517	`počet gólov`
`BECD`	0.078	0.518	`počet gólov`

Tabuľka 5.1: Nájdené frekventované vzory

zmena	symbol
$(-\infty, -0.050)$	A
$< -0.050, -0.016)$	B
$< -0.016, -0.001)$	C
$< -0.001, 0.019)$	D
$< 0.019, 0.051)$	E
$< 0.051, \infty)$	F

Tabuľka 5.2: Tabuľka prevodov pre 6 symbolov

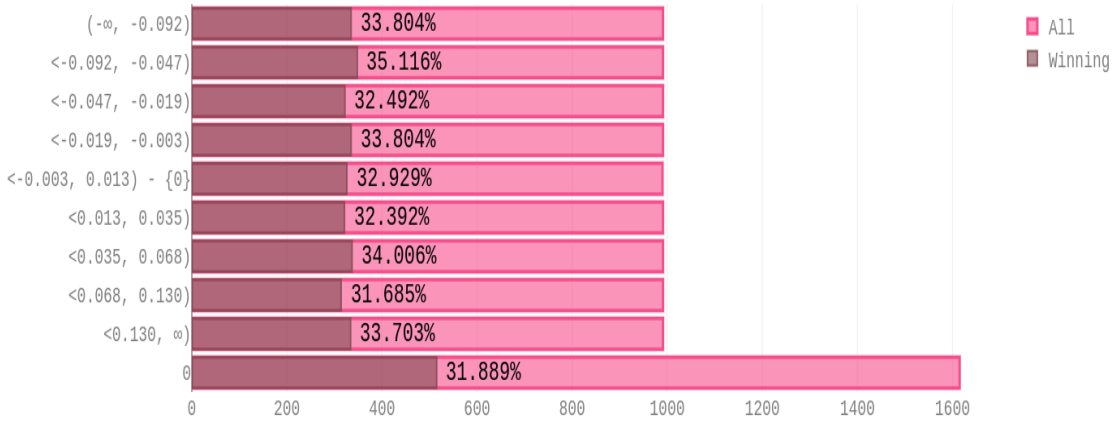
5.2 Celková zmena kurzu

Závislosť celkovej zmeny kurzu a výsledkov stávky pre stávky typu `1x2` je znázornená na histograme 5.1. Y-ová os označuje jednotlivé intervaly celkovej zmeny kurzu. X-ová os označuje celkový počet stávok v danom intervale a počet víťazných stávok v danom intervale. Popisky jednotlivých stĺpcov navyše uvádzajú percentuálny podiel víťazných stávok voči všetkým stávkam, u ktorých bola zmena kurzu v danom intervale.

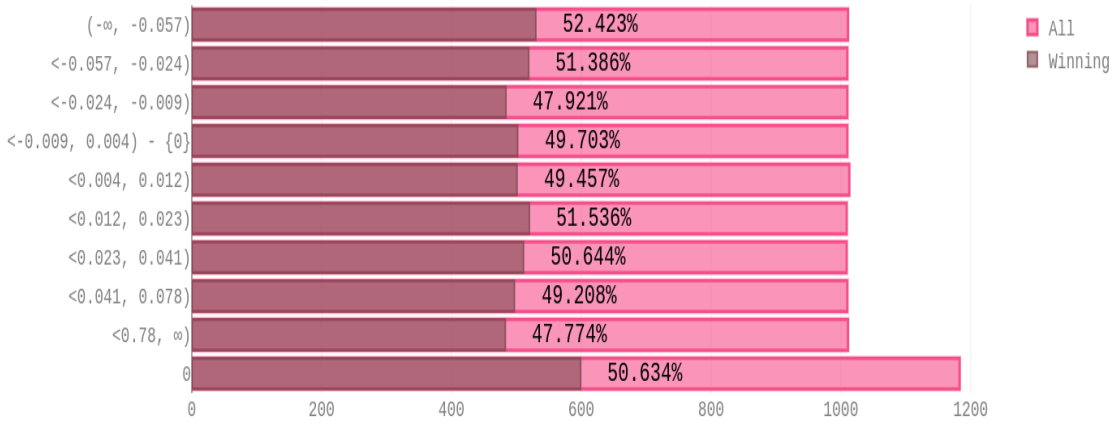
Nakoľko sa jedná o stávky s troma možnými tipmi, pri náhodnom vyhodnotení stávok by bol pomer víťazných tipov 33.3%. Vidíme, že pre našu vzorku sa pravdepodobnosť výhry pohybuje v rozmedzí 31.685% - 35.116% v závislosti na veľkosti zmeny kurzu.

Pre stávky typu `počet gólov` je závislosť zmien kurzov a výsledkov stávok znázornená na histograme 5.2. Sémantika osí a popisiek jednotlivých stĺpcov je rovnaká, ako tomu bolo v predchádzajúcom prípade.

Keďže sa jedná o stávky s dvoma možnými tipmi, pri náhodnom rozdelení bude pravdepodobnosť výhry 50%. Pre našu vzorku sa pravdepodobnosť výhry pohybuje v rozmedzí 47.774% - 52.423%. V oboch prípadoch vidíme, že pravdepodobnosť výhry pre jednotlivé zmeny kurzov je blízka náhodnej pravdepodobnosti. To, či sú tieto odchýlky od náhodnej pravdepodobnosti dostatočne významné, je otázne. Túto hypotézu by bolo vhodné verifikovať na väčšej vzorke, ktorú momentálne nemáme k dispozícii.



Obr. 5.1: Histogram pohybov kurzov pre stávky typu `1x2`



Obr. 5.2: Histogram pohybov kurzov pre stávky typu `počet gólov`

Kapitola 6

Záver

Práca priblížila a demonštrovala postupy pre návrh a implementáciu systémov pre získavanie dát z webu. Na konkrétnom prípade bolo potom ukázané, aké kroky je nutné podstúpiť pre vytvorenie takéhoto systému. Na nazbieraných dátach boli vykonané jednoduché analýzy. Bolo ukázané, aké operácie je potrebné na dáta v jednotlivých krokoch procesu dolovania znalostí aplikovať. Pomocou apriori algoritmu sa podarilo v dátach detekovať frekventované vzory. Tieto môžu stávkarovi pomôcť pri rozhodovaní v prípade, že tieto vzory bude pozorovať v budúcnosti. Pomocou histogramov bol preskúmaný a prezentovaný vzťah medzi celkovým rozdielom kurzu a pravdepodobnosťou výhry stávky.

Práca by mohla byť rozšírená o dáta z ďalších zdrojov. Dali by sa tak demonštrovať techniky získavania dát z webu zo stránok, ktoré sú statické alebo používajú websockety. Rozšírili by sa okrem toho tiež možnosti analýzy dát a bolo by možné sledovať vzťahy medzi ponukami jednotlivých stávkových kancelárií.

Napriek tomu, že boli všetky body zadania splnené, práca sa trocha príliš sústredila na fázu web scrapingu. V dôsledku toho bolo časti získavania znalostí venované nižšie úsilie než by bolo ideálne.

Bolo by dobré, ak by zber dát prebiehal dlhšiu dobu a bolo by tým pádom k dispozícii väčšie množstvo dát. Výsledky analýz by tak boli spoľahlivejšie.

V časti získavania znalostí by bolo dobré skúsiť niektoré z mnohých ďalších metód dolovania znalostí - napríklad niektoré z prediktívnych metód. Taktiež by bolo dobré podobnú prácu konzultovať s doménovým expertom z oblasti kurzového stávkovania a na základe toho odvodiť vhodné metódy získavania znalostí.

Literatúra

[1] Adam Cohen: *FuzzyWuzzy: Fuzzy String Matching in Python*. [Online; navštívene 01.03.2019].
URL https://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/

[2] BeautifulSoup authors: *BeautifulSoup documentation*. [Online; navštívene 01.03.2019].
URL https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[3] Bray, T.: The JavaScript Object Notation (JSON) Data Interchange Format. STD 90, December 2017.
URL https://tools.ietf.org/html/rfc7159

[4] Cohen, W.; Ravikumar, P.; Feinberg, S. E.: A comparison of string distance metrics for name-matching tasks. In *KDD Workshop on Data Cleaning and Object Consolidation*, ročník 3, 2003, s. 73–78.

[5] Cronin, B.: *Super Bowl coin toss odds: Everything you need to know*. [Online; navštívene 01.03.2019].
URL https://www.pinnacle.com/en/betting-articles/Football/Super-Bowl-coin-toss-betting/JTB2LE2M5AKT48QC

[6] Dataweave authors: *json vs simplejson vs ujson*. [Online; navštívene 01.03.2019].
URL https://dataweave.com/blog/json-vs-simplejson-vs-ujson-2887b2c128b2

[7] Fette, I.; Melnikov, A.: The WebSocket Protocol. RFC 6455, December 2011.
URL https://tools.ietf.org/html/rfc6455

[8] FuzzyWuzzy contributors: *FuzzyWuzzy*. [Online; navštívene 01.03.2019].
URL https://github.com/seatgeek/fuzzywuzzy

[9] Han, J.; Kamber, M.; Pei, J.: *Data Mining: Concepts and Techniques*. ITPro collection.; Morgan Kaufmann series in data management systems, Morgan Kaufmann Publishers, tretie vydanie, 2012, ISBN 978-0-12-381479-1.

[10] James M. Jensen II: *Damerau-Levenshtein Edit Distance Explained*. [Online; navštívene 01.03.2019].
URL https://www.lemoda.net/text-fuzzy/damerau-levenshtein/index.html

[11] Krylysov, A.: *Benchmark of Python JSON libraries*. [Online; navštívene 01.03.2019].
URL https://artem.krylysov.com/blog/2015/09/29/benchmark-python-json-libraries/

[12] Maimon, O.; Rokach, L.: Introduction to Knowledge Discovery and Data Mining. In *Data Mining and Knowledge Discovery Handbook*, editácia O. Maimon; L. Rokach, Springer US, druhé vydanie, 2010, ISBN 978-0-387-09823-4, s. 1–15.

[13] Marc-André Lemburg: *PEP 249 – Python Database API Specification v2.0*. [Online; navštívene 01.03.2019].
URL https://www.python.org/dev/peps/pep-0249/

[14] Nikhil Babar: *The Levenshtein Distance Algorithm*. [Online; navštívene 01.03.2019].
URL https://dzone.com/articles/the-levenshtein-algorithm-1

[15] NumPy developers: *NumPy*. [Online; navštívene 01.03.2019].
URL https://www.numpy.org/

[16] Octoparse authors: *Octoparse: Webscraping Introduction*. [Online; navštívene 01.03.2019].
URL https://www.octoparse.com/blog/web-scraping-introduction

[17] Pandas developers: *Pandas Ecosystem*. [Online; navštívene 01.03.2019].
URL https://pandas.pydata.org/pandas-docs/stable/ecosystem.html

[18] Requests authors: *Requests documentation*. [Online; navštívene 01.03.2019].
URL https://2.python-requests.org/en/master/

[19] Schwab, D.: *Different Types of Sports Bets*. [Online; navštívene 01.03.2019].
URL https://www.worldcasinodirectory.com/articles/different-types-of-sports-bets

[20] Scrapy authors: *Scrapy Framework Documentation*. [Online; navštívene 01.03.2019].
URL https://docs.scrapy.org/en/latest/

[21] Seif, G.: *An Introduction to Scikit Learn: The Gold Standard of Python Machine Learning*. [Online; navštívene 01.03.2019].
URL https://towardsdatascience.com/an-introduction-to-scikit-learn-the-gold-standard-of-python-machine-learning-e2b9238a98ab

[22] Selenium authors: *Selenium documentation*. [Online; navštívene 01.03.2019].
URL https://www.seleniumhq.org/docs/

[23] Spiegel, J.; Robie, J.; Dyck, M.: XML Path Language (XPath) 3.1. W3C recommendation, W3C, Marec 2017,
https://www.w3.org/TR/2017/REC-xpath-31-20170321/.

[24] *How do bookmakers make money*. [Online; navštívene 01.03.2019].
URL https://help.smarkets.com/hc/en-gb/articles/214180825-How-do-bookmakers-make-money

[25] *How to Bet on Sports – The Different Types of Wagers*. [Online; navštívene 01.03.2019].
URL
https://www.gamblingsites.com/sports-betting/introduction/bets-wagers/

[26] *Matplotlib*. [Online; navštívene 01.03.2019].
URL https://matplotlib.org/

[27] *Odds formats explained*. [Online; navštívene 01.03.2019].
URL https://metabetting.com/odds-formats-explained-n64

[28] *Puppet 6 documentation*. [Online; navštívene 01.03.2019].
URL https://puppet.com/docs/puppet/6.4/puppet_index.html

[29] *reCAPTCHA v3 The new way to stop bots*. [Online; navštívene 01.03.2019].
URL https://www.google.com/recaptcha/intro/v3.html

[30] *runit - a UNIX init scheme with service supervision*. [Online; navštívene 01.03.2019].
URL http://smarden.org/runit/

[31] *SQLite: Most Widely Deployed and Used Database Engine*. [Online; navštívene 01.03.2019].
URL https://www.sqlite.org/mostdeployed.html

[32] *Supervisor: A Process Control System*. [Online; navštívene 01.03.2019].
URL http://supervisord.org/

[33] *systemd - system and service manager*. [Online; navštívene 01.03.2019].
URL https://wiki.debian.org/systemd

[34] *The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things*. [Online; navštívene 01.03.2019].
URL https://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm

[35] *Twisted*. [Online; navštívene 01.03.2019].
URL https://twistedmatrix.com/trac/

[36] *What are the different betting odds formats*. [Online; navštívene 01.03.2019].
URL https://help.smarkets.com/hc/en-gb/articles/214071849-What-are-the-different-betting-odds-formats-

[37] Zeifman, I.: *Bot traffic report 2016*. [Online; navštívene 01.12.2018].
URL https://www.imperva.com/blog/bot-traffic-report-2016/?utm_campaign=Incapsula-moved

Príloha A

Obsah priloženého média

```
CD
├─ IBT2 (obsahuje zdrojové súbory pre ziskavanie dat z webu a ich analyzu.)
│  ├─ analysis
│  ├─ data_retrieval
│  │  ├─ event_resolution
│  │  ├─ odds
│  │  │  └─ books
│  │  ├─ scripts
│  │  └─ systemd_services
│  └─ db
├─ doc
│  └─ latex (obsahuje zdrojove texty z ktorych bol vytvoreny tento dokument)
└─ xjurko02-BP.pdf
```