

EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

**POROVNANIE NÁSTROJOV MACHINE LEARNING NA
KONKRÉTNOM PRÍKLADE**

Diplomová práca

Bratislava 2021

Bc. Jicchág Šoltés

EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY

**POROVNANIE NÁSTROJOV MACHINE LEARNING NA
KONKRÉTNOM PRÍKLADE**

Diplomová práca

Študijný program: Informačný manažment

Študijný odbor: Ekológia a manažment

Školiace pracovisko: KAI FHI – Katedra aplikovanej informatiky FHI

Vedúci záverečnej práce: RNDr. Eva Rakovská, PhD.

Bratislava 2021

Bc. Jicchág Šoltés



Ekonomická univerzita v Bratislave
Fakulta hospodárskej informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Jicchág Šoltés
Študijný program: Informačný manažment (Jednoodborové štúdium, inžiniersky II. st., denná forma)
Študijný odbor: 8. - ekonómia a manažment
Typ záverečnej práce: Inžinierska záverečná práca
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Porovnanie nástrojov machine learning na konkrétnom príklade.

Anotácia: Autor práce vyhľadá voľne dostupnú databázu (napr. futbalových výsledkov) a aplikuje rôzne voľne dostupné nástroje pre strojové učenie. Vyberie vhodný typ úlohy ako klasifikácia, clustering, regresiu a pod.), ktorými sa bude zaoberať a na nich bude aplikovať a porovnávať efektivitu zvolených nástrojov

Vedúci: RNDr. Eva Rakovská, PhD.
Katedra: KAI FHI - Katedra aplikovanej informatiky FHI
Vedúci katedry: Ing. Mgr. Peter Schmidt, PhD.
Dátum zadania: 23.10.2019

Dátum schválenia: 29.10.2019
Ing. Mgr. Peter Schmidt, PhD.
vedúci katedry

ČESTNÉ VYHLÁSENIE

Čestne vyhlasujem, že záverečnú prácu som vypracoval samostatne a že som uviedol použitú literatúru.

Dátum: 11.5.2021



.....
(podpis študenta)

ABSTRAKT

ŠOLTÉS, Jicchág: *Porovnanie nástrojov machine learning na konkrétnom príklade*. [Diplomová práca] / Jicchág Šoltés. – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra aplikovanej informatiky. – Vedúca: RNDr. Eva Rakovská, PhD. – Stupeň odbornej kvalifikácie: Inžinier. – Bratislava: Fakulta hospodárskej informatiky, 2021. 140 strán.

Cieľom práce je vysvetliť, čo je to machine learning a prediktívne modelovanie, v čom spočíva jeho význam, a na konkrétnom príklade porovnať vybrané nástroje machine learning. Práca je rozdelená do troch kapitol. Obsahuje 86 obrázkov a 34 tabuliek. Prvá kapitola je venovaná súčasnému stavu a výskumu aplikačnej praxe v oblastiach machine learning a prediktívneho modelovania. V ďalšej časti je charakterizovaný cieľ tejto práce, čiastkové ciele, ktoré viedli k jeho naplneniu, a metódy, akými bola písaná. Záverečná kapitola sa zaoberá praktickou časťou v podobe niekoľkých modelov vytvorených pomocou troch open source machine learning nástrojov. Výsledkom riešenia danej problematiky je klasifikácia, analýza aplikačného využitia týchto nástrojov a porovnanie vytvorených modelov na dvoch konkrétnych príkladoch.

Kľúčové slová: strojové učenie, dataset, datamining, prediktívne modelovanie, nástroje machine learning, klasifikácia, H2O.ai, Weka, Jupyter, Scikit-learn

ABSTRACT

ŠOLTÉS, Jicchág: *Comparison of machine learning tools on a specific example*. [Diploma thesis] / Jicchág Šoltés. – University of economics in Bratislava. Faculty of business informatics; Department of applied informatics. – Supervisor: RNDr. Eva Rakovská, PhD. – Qualification degree: Master. – Bratislava: Faculty of business informatics, 2021. 140 pages.

The goal of the diploma thesis is to explain what machine learning and predictive modeling is, what is its significance and to compare selected machine learning tools on a specific example. The work is divided into three chapters. It contains 86 pictures and 34 tables. The first chapter is devoted to the current state and research of application practice in the areas of machine learning and predictive modeling. The next part characterizes the goal of this work, the partial goals that led to its fulfillment and the methods by which it was written. The final chapter deals with the practical part in the form of several models created using three open source machine learning tools. The result of solving the problem is classification, analysis of application use of these tools and comparison of created models on two specific examples.

Keywords: machine learning, dataset, datamining, predictive modeling, machine learning tools, classification, H2O.ai, Weka, Jupyter, Scikit-learn

OBSAH

ÚVOD.....	15
1. Súčasný stav	17
1.1. Strojové učenie.....	18
1.2. Supervised ML.....	19
1.3. Klasifikácia.....	20
1.4. Klasifikácia a metóda podporných vektorov	21
1.4.1. Rozdelenie dát vo vektorovom priestore	22
1.4.2. Kernal.....	24
1.4.3. Klasifikátor podporných vektorov	25
1.5. Klasifikácia a Naive Bayes.....	26
1.5.1. Naivný Bayesovský klasifikátor.....	27
1.6. Hodnotiace metriky a výkon modelu	29
1.6.1. Confusion matrix.....	29
1.6.2. Accuracy.....	32
1.6.3. Precision a Recall	32
1.6.4. Skóre F1.....	33
1.7. Nástroje ML.....	33
1.8. Scikit-learn + Jupyter.....	35
1.9. Weka.....	36
1.10. H2O	37
2. Ciele a metodika práce	39
2.1. Formulácia úlohy.....	39
2.2. Prediktívne modelovanie.....	39
2.3. Metodika CRISP DM.....	40

3. Výsledky práce.....	42
3.1. Dataset.....	42
3.2. Dataset futbalových zápasov.....	43
3.3. Predikcia výsledkov zápasov anglickej Premier League	45
3.4. Scikit-learn+Jupyter a dataset futbalových zápasov.....	45
3.4.1. Moduly	48
3.4.2. Nahratie datasetu a jeho vlastnosti.....	49
3.4.3. Vizualizácia datasetu	51
3.4.4. Preprocessing dát	52
3.4.5. Štandardizácia dát	54
3.4.6. Rozdelenie dát na testovacie a trénovacie	56
3.4.7. Trénovanie klasifikátora.....	56
3.4.8. Testovanie klasifikátora.....	57
3.5. Weka a dataset futbalových zápasov.....	59
3.5.1. Attribute Relation File Format	60
3.5.2. Konverzia CSV na ARFF	62
3.5.3. Nahratie a vizualizácia datasetu vo Weka	64
3.5.4. Selekcia atribútov vo Weka	69
3.5.5. Selekcia atribútov na základe korelácií.....	72
3.5.6. Selekcia atribútov na základe informačného zisku	74
3.5.7. Selekcia atribútov na základe algoritmov strojového učenia	75
3.5.8. Rozdelenie dát na testovacie a trénovacie	78
3.5.9. Trénovanie a testovanie.....	79
3.6. H2O a dataset futbalových zápasov.....	85
3.6.1. H2O klient server architektúra	85

3.6.2. H2O Flow	86
3.6.3. Inštalácia a spustenie H2O Flow.....	87
3.6.4. Interface H2O Flow	88
3.6.5. Načítanie datasetu v H2O.....	91
3.6.6. Konverzia CSV na HEX	93
3.6.7. H2O Frame	94
3.6.8. Rozdelenie dát na testovacie a trénovacie v H2O	96
3.6.9. Vytvorenie modelu	98
3.6.10. Trénovanie a testovanie.....	100
3.6.11. H2O a SVM	104
3.6.12. AutoML: Automatic Machine Learning.....	109
3.6.13. Konfigurácia Auto ML.....	110
3.6.14. Požadované parametre	110
3.6.15. Spustenie AutoML.....	112
3.6.16. AutoML Leaderboard	113
3.6.17. Detail modelu AutoML.....	114
3.7. Dataset s údajmi o kreditných kartách.....	119
3.7.1. Predikcia podvodov vo finančných platobných službách	120
3.7.2. Import údajov datasetu s finančnými údajmi.....	121
3.7.3. Analýza údajov.....	121
3.7.4. Preprocessing údajov o kreditných kartách	125
3.7.5. Trénovanie a testovanie	125
3.8. Výsledky porovnania nástrojov	126
3.8.1. Metriky výkonu predikcie futbalových zápasov	126
3.8.2. Vyhodnotenie metrík výkonu predikcie finančných podvodov.....	129

Záver a diskusia.....	134
Zoznam bibliografických odkazov.....	136

ZOZNAM ILUSTRÁCIÍ A TABULIEK

Obrázok 1. Fázy strojového učenia	18
Obrázok 2. Hyperrovina	21
Obrázok 3. Mechanizmus SVM.....	23
Obrázok 4. Hyperplocha s neoddeliteľnými dátami	24
Obrázok 5. Kemel trik.....	25
Obrázok 6. Weka GUI.....	37
Obrázok 7. Zobrazenie datasetu v Exceli.....	43
Obrázok 8. Rozhranie Jupyter Notebook.....	46
Obrázok 9. Terminálový výpis Jupyter Notebook.....	46
Obrázok 10. Bunka Jupyter Notebook.....	47
Obrázok 11. Výpis reťazca v Jupyter Notebook.....	47
Obrázok 12. Import knižníc v Pythone.....	48
Obrázok 13. Nahratie datasetu futbalových zápasov do Pandas Data Frame.....	49
Obrázok 14. Výpis funkcie count()	50
Obrázok 15. Výpis hodnôt atribútu FTR a počet ich záznamov v datasete	50
Obrázok 16. Scatter matrix pre atribúty datasetu futbalových zápasov.....	52
Obrázok 17. Data Frame X_all, ktorý už neobsahuje FTR	53
Obrázok 18. Data Frame y_all s jediným atribútom FTR	53
Obrázok 19. Výsledok transformácie X_all funkciou scale().....	54
Obrázok 20. Atribúty v dátovom typom reťazec.....	55
Obrázok 21. Transformácia kategorických dát na číselné hodnoty.....	56
Obrázok 22. Report pre svc_classifier.....	58
Obrázok 23 Report pre gnb_classifier	59
Obrázok 24. Dáta vo formáte CSV súboru	60
Obrázok 25. Dáta vo formáte ARFF súboru	61
Obrázok 26. Okno Weka Chooser.....	62
Obrázok 27. Okno ARFF-Viewer.....	63
Obrázok 28. CSV súbor v ARFF-Viewer.....	63
Obrázok 29. Okno Weka Explorer	65
Obrázok 30. Vizualizácia dát vo Weka Explorer	66

Obrázok 31. Visualize all vo Weka Explorer.....	67
Obrázok 32. Scatter matrix vo Weka Visualize	68
Obrázok 33. Vizualizácia dát pre x: AST a y: FTHG.....	69
Obrázok 34. Okno Select attributes vo Weka	70
Obrázok 35. Hlásenie Alert vo Weka.....	71
Obrázok 36. Podrobnosti o konfigurácii Attribute Evaluator.....	71
Obrázok 37. Podrobný popis o konfigurácii Attribute Evaluator	72
Obrázok 38. Výsledok selekcie atribútov na základe korelácií	73
Obrázok 39. Výsledok selekcie atribútov na základe informačného zisku.....	75
Obrázok 40. WrapperSubsetEval vo Weka.....	76
Obrázok 41. Výsledok selekcie atribútov na základe algoritmov ML	77
Obrázok 42. Test options a rozdelenie dát pravidlom 80:20 vo Weka.....	78
Obrázok 43. Výber učiaceho algoritmu vo Weka	79
Obrázok 44. Štart tvorby modelu vo Weka.....	80
Obrázok 45. Report Classifier Output.....	81
Obrázok 46. Otvorenie Package Managera vo Weka	82
Obrázok 47. Package Manager vo Weka.....	83
Obrázok 48. Report Classifier Output pre SVM vo Weka.....	84
Obrázok 49. Confusion matrix pre kernel radial basis.....	85
Obrázok 50. Chybové hlásenie 503 pri inštalácii H2O	87
Obrázok 51. Prostredie H2O Flow	88
Obrázok 52. H2O príkaz assist	89
Obrázok 53. Bunka v H2O Flow	89
Obrázok 54.H2O Flow panel s tlačidlami.....	90
Obrázok 55.H2O Flow a upload súboru	91
Obrázok 56. H2O Flow Setup parse configuration	92
Obrázok 57. H2O Flow Setup parse – upraviť mená a typy atribútov.....	93
Obrázok 58.H2O Flow príkaz parseFiles	94
Obrázok 59.H2O Flow Key Frame	95
Obrázok 60.H2O Flow changeColumnTypes report.....	96
Obrázok 61. H2O Flow príkaz getFrameSummary	97

Obrázok 62. Rozdelenie dát na tréningovú a validačnú časť v H2O Flow	97
Obrázok 63. Zoznam všetkých H2O Frames v pamäti H2O Flow.....	98
Obrázok 64. H2O Flow a vytvorenie modelu	99
Obrázok 65. H2O Flow Setup Parse.....	100
Obrázok 66. H2O Flow konfigurácia prediktívneho modelu	101
Obrázok 67. H2O Flow príkaz buildModel	101
Obrázok 68. H2O Flow informácie o vytvorenom prediktívnom modeli.....	102
Obrázok 69. H2O Flow výstup Prediction.....	103
Obrázok 70. H2O Flow výstup Prediction – confusion matrix a hit_ratio	104
Obrázok 71. SVM v Jupyter+H2O.....	106
Obrázok 72. Informácie o spojení s klastrom H2O	107
Obrázok 73. Výpis funkcie model_performance()	108
Obrázok 74. H2O Run AutoML.....	110
Obrázok 75. AutoML parametre	111
Obrázok 76. H2O AutoML Job.....	112
Obrázok 77. H2O AutoML Leaderboard.....	113
Obrázok 78. Zobrazenie metriky Logloss v H2O AutoML.....	115
Obrázok 79. H2O AutoML – selekcia atribútov	116
Obrázok 80. Príklad Confusioin matrix v H2O AutoML	117
Obrázok 81. H2O AutoML validation metrics.....	118
Obrázok 82. H2O AutoML cross-validation metrics.....	119
Obrázok 83. Výpis funkcie head v Jupyter – dataset údajov o kreditných kartách	122
Obrázok 84. Zobrazenie vo Weka Explorer – dataset údajov o kreditných kartách.....	123
Obrázok 85. Zobrazenie dát v H2O Flow – dataset údajov o kreditných kartách	123
Obrázok 86. Analýza datasetu v Jupyter – dataset údajov o kreditných kartách.....	124
 Tabuľka 1. Confusion matrix.....	 30
Tabuľka 2. Confusion matrix starý mladý	31
Tabuľka 3. Vysvetlivky k datasetu futbalových zápasov	44
Tabuľka 4. Vysvetlivky datasetu údajov o kreditných kartách.....	120

Tabuľka 5. Confusion matrix pre Naive Bayes v Jupyter + sklearn	126
Tabuľka 6. Confusion matrix pre Naive Bayes vo Weka Explorer	126
Tabuľka 7. Confusion matrix pre Naive Bayes v H2O Flow	127
Tabuľka 8. Accuracy score pre Naive Bayes.....	127
Tabuľka 9. Precision score pre Naive Bayes.....	127
Tabuľka 10. Recall pre Naive Bayes.....	127
Tabuľka 11. F1 score pre Naive Bayes.....	127
Tabuľka 12. Confusion matrix pre Support Vector Machines v Jupyter + sklearn	128
Tabuľka 13. Confusion matrix pre Support Vector Machines vo Weka Explorer	128
Tabuľka 14. Confusion matrix pre Support Vector Machines v H2O Flow.....	128
Tabuľka 15. Accuracy pre Support Vector Machines	128
Tabuľka 16. Precision pre Support Vector Machines	129
Tabuľka 17. Recall pre Support Vector Machines	129
Tabuľka 18. F1 score pre Support Vector Machines	129
Tabuľka 19. Confusion matrix pre Naive Bayes v Jupyter + sklearn	129
Tabuľka 20. Confusion matrix pre Naive Bayes vo Weka Explorer.....	130
Tabuľka 21. Confusion matrix pre Naive Bayes v H2O Flow	130
Tabuľka 22. Accuracy score pre Naive Bayes	130
Tabuľka 23. Precision score pre Naive Bayes	130
Tabuľka 24. Recall pre Naive Bayes.....	130
Tabuľka 25. F1 score pre Naive Bayes.....	131
Tabuľka 26. Confusion matrix pre Support Vector Machines v Jupyter + sklearn.....	131
Tabuľka 27. Confusion matrix pre Support Vector Machines vo Weka Explorer	131
Tabuľka 28. Confusion matrix pre Support Vector Machines v H2O Flow	131
Tabuľka 29. Accuracy pre Support Vector Machines	131
Tabuľka 30. Precision pre Support Vector Machines	132
Tabuľka 31. Recall pre Support Vector Machines	132
Tabuľka 32. F1 score pre Support Vector Machines	132
Tabuľka 33. Poradie výsledkov metrík nástrojov.....	133
Tabuľka 34. Cca čas potrebný na dokončenie úloh.....	133

ZOZNAM SKRATIEK A ZNAČIEK

ML	Machine Learning
CRISP-DM	Cross-industry standard process for data mining
SVM	Support Vector Machines
AI	Artificial Intelligence
API	Application Program Interface
DF	Data Frame
CSV	Comma-Separated Values
ARFF	Attribute-Relation File Format
GUI	Graphical User Interface

ÚVOD

Machine learning a AI sú pojmy, ktoré keď sa snažíme vysvetliť ľuďom mimo sveta IT, väčšinou to pochopia ako jednu a tú istú vec. Umelá inteligencia a strojové učenie sú časti počítačovej vedy, ktoré navzájom korelujú. Tieto dve technológie sú najpopulárnejšími technológiami, ktoré sa používajú na vytváranie inteligentných systémov súčasnosti a ide síce o dve súvisiace technológie a ľudia ich niekedy používajú ako vzájomné synonymá, no v oboch prípadoch však ide o dva rôzne výrazy. Strojové učenie umožňuje počítačovému systému robiť predpovede alebo prijímať určité rozhodnutia pomocou historických údajov bez toho, aby boli výslovne naprogramované. Na to využíva obrovské množstvo údajov, aby model strojového učenia mohol generovať presný výsledok, alebo na základe týchto údajov poskytovať predpovede. Celý koncept machine learning funguje na algoritme, ktorý sa sám učí pomocou historických údajov a používa na rôznych miestach, napríklad pre online odporúčací systém, pre vyhľadávacie algoritmy Google, e-mailový spamový filter, návrh automatického označovania priateľov na Facebooku atď. Predstavme si, že chceme predpovedať vývoj správania sa zákazníkov v danom biznise. Máme k dispozícii dáta, máme k dispozícii tím. Aký nástroj strojového učenia ale použiť v tomto konkrétnom prípade? V diplomovej práci sa budeme snažiť opísať a porovnať tri nástroje na machine learning, ktorými budeme riešiť rovnakú úlohu. Každý z nástrojov bude v niečom odlišný, ale jednu vlastnosť budú mať všetky nástroje rovnakú, a to, že pôjde o voľne dostupný softvér (open source). V prvej časti práce vysvetlíme, ako fungujú vybrané algoritmy strojového učenia, ako vytvoriť model strojového učenia a ako vyhodnotiť a overiť, či model produkuje želané výstupy. Poslednou časťou prvej kapitoly je rebríček vybraných machine learning nástrojov. V rebríčku nájdeme riešenia na všetky dostupné platformy, no samozrejme nie všetky sú zadarmo. Vytvoriť dostatočne efektívny model vyžaduje pri veľkom objeme dát aj veľkú výpočtovú silu, ktorú osobný počítač veľakrát neposkytuje. Dva nástroje Scikit-learn+Jupyter a Weka mohli pracovať s výkonom, ktorý bol dostupný na zariadení, na ktorom sme tvorili túto diplomovú prácu. Už na začiatok by sme radi uviedli že tretí nástroj H2O sa pomocou API pripájal na klaster v Prahe, čo určite ovplyvnilo finálne výsledky metrík výkonu vytvorených modelov. Druhá kapitola sa venuje cieľom diplomovej práce a týka sa úloh, ktoré budeme riešiť v tretej kapitole. Popíšeme si pojem prediktívne modelovanie na

konkrétnom príklade zo športu a finančnej sféry. Pri praktickej časti sme pracovali metodikou CRISP DM, ktorá vznikla v rámci Európskeho výskumného projektu, ktorého cieľom bolo navrhnúť univerzálny postup použiteľný v najrôznejších komerčných aplikáciách. V tretej kapitole uvedieme postupy tvorby prediktívnych modelov pre vybrané nástroje. Na dvoch úlohách sa budeme snažiť porovnať plusy a mínusy vybraných nástrojov a prejdeme úlohy ako analýza datasetu, preprocessing dát, vytvorenie supervised machine learning modelu, tréning a testovanie. Pri úlohách, ktoré vyžadujú programovanie, sme použili populárny programovací jazyk Python. Úlohou číslo jedna je klasifikácia futbalových výsledkov na základe výstupného parametra, ktorým je výsledok zápasu hry. Pri riešení tejto úlohy si podrobne popíšeme každý z nástrojov. Druhá úloha sa zaoberá detekciou finančných podvodov pri bankových transakciách, a pri tejto úlohe už nebudeme dopodrobna vysvetľovať, ako sme sa dopracovali k výsledku, ale porovnáme kroky, ktoré musíme prejsť pri riešení úlohy vo vybraných nástrojoch. Na konci tretej kapitoly zhrnieme výsledky nášho skúmania, na čo nám poslúžia metriky výkonu vytvorených prediktívnych modelov. Uvedieme aj možno subjektívny pohľad na jednotlivé nástroje a zhrnieme výhody a nevýhody jednotlivých riešení. Machine learning je určite budúcnosť a vyskytuje sa už aj v odvetviach, kde by to nikto nečakal. Automatizácia procesov v biznise pomaly otvára novú dobu a napríklad vo veľkých korporáciách je stále väčší hlad po ľuďoch, ktorí sa vedia pohybovať v oblasti dátovej vedy a machine learning. V práci uvidíme, že vytvoriť machine learning model už v dnešnej dobe nie je extrémne náročné hoci aj pre úplných začiatočníkov. V práci vychádzame z viacerých zdrojov, nájdeme medzi nimi vedecké články, rôzne štatistické údaje a dokumentácie vybraných nástrojov. Inšpiráciu k úlohám sme čerpali z vedeckých publikácií, videí a príspevkov dostupných na portáloch o problematike data science.

1. Súčasný stav

Je ťažšie ako by sa zdalo, prísť s operatívnou definíciou strojového učenia (ML). Tento výraz môže môžeme chápať ako podoblasť počítačovej vedy, ale aj ako súbor tém, ktoré sa rozvíjajú a používajú v oblasti informatiky, strojárstva, štatistiky a čoraz viac v spoločenských vedách. Na začiatok by sme mohli definovať strojové učenie ako oblasť informatiky, ktorá vyvíja algoritmy zamerané na predikciu (regresiu), klasifikáciu a zhľukovanie [1]. Pomocou algoritmov dokážeme vytvárať ML modely, ktoré majú kognitívne vlastnosti. Medzi tieto vlastnosti vieme zaradiť napríklad rozpoznávanie objektov, predikciu nejakej situácie, analýzu textu, analýzu finančných dát a podobne.

Supervised ML vychádza z predpokladu, že model učíme pomocou historických dát tak, aby bol schopný predpovedať budúce hodnoty nami zvolenej vlastnosti. Príkladom môže byť klasifikácia zvierat, kde ML model rozoznáva druhy zvierat z fotografií. Systém je prezentovaný s príkladmi vstupov a ich požadovanými výstupmi a všeobecným pravidlom, ktoré mapuje vstupy na výstupy. Supervised ML je založené na tréňovaní modelu vzorkou dát zo zdroja údajov s už priradenou správnou klasifikáciou [2].

Unsupervised ML sa používa pri problémoch rozpoznávania vzorov v dátach. Cvičné dáta sa skladajú zo sady vektorov bez akýchkoľvek zodpovedajúcich cieľových hodnôt. Používa sa napríklad pri hľadaní skupín podobných dát, čo sa nazýva zhľukovanie. Tu hovoríme o hľadaní súvislostí v nejakom dátovom celku a následné utriedenie dátových bodov na základe podobných vlastností do zhľukov [3]. Ďalej môžeme využitie nájsť pri detekcii distribúcie dát konkrétneho dátového zdroja.

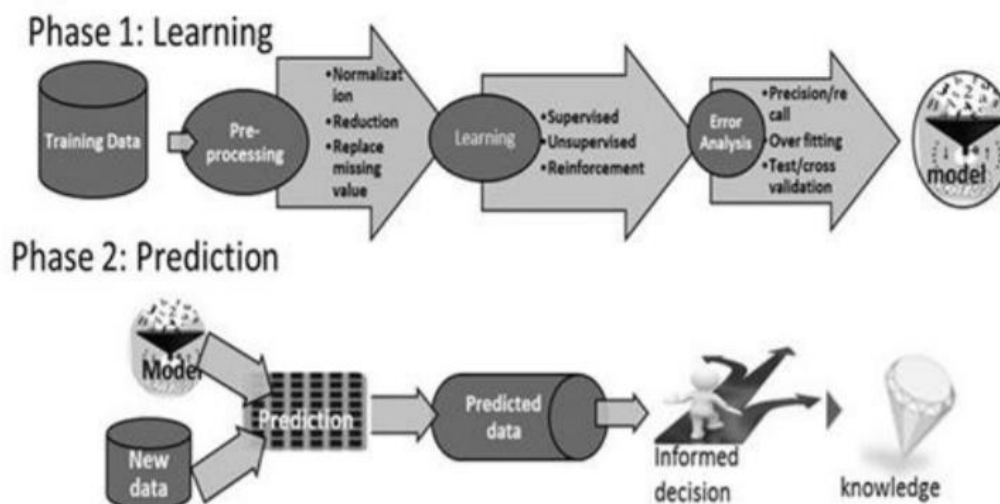
V posledných rokoch nepribúdajú nové metódy strojového učenia, ktoré by boli založené na celkom nových princípoch. Skôr je tu snaha využiť existujúce techniky na riešenie stále nových úloh v rozličných oblastiach. Možno práve pre veľký objem dát, ktoré je potrebné spracovať, sa aj v strojovom učení presadzujú metódy, ktoré sú charakteristické štatistickým prístupom [4].

1.1. Strojové učenie

Strojové učenie je odvetvie umelej inteligencie, ktoré sa zaoberá budovaním systémov, ktoré vyžadujú minimálny ľudský zásah a pomáhajú skúmať údaje a robiť presné predikcie. Na rozdiel od mnohých štatistických prístupov, ktoré môžu hodnotiť inferenciu predikcie, strojové učenie sa zameriava na presnosť predikcie. Strojové učenie pomáha eliminovať statický, pevný a prísny prístup k dobre štruktúrovanému programovaniu, ktorý zvyčajne poskytuje buď zlú optimalizáciu, alebo neefektívne využitie pamäťového priestoru a času [5].

Strojové učenie sa skladá z dvoch fáz, a to fázy učenia a fázy predikcie ako zobrazené na Obrázku 1. Fáza učenia zahŕňa nasledujúce:

- predspracovanie,
- učenie,
- analýza chýb,
- stavba modelu.



Obrázok 1. Fázy strojového učenia [5]

Fáza predikcie preberá výstup fázy učenia, ktorá je modelom predikcie nových údajov datasetu. Predpovedané údaje pomáhajú manažérom alebo rozhodovacím orgánom prijímať rozhodnutia, z ktorých ďalej môžu vybudovať napríklad databázu znalostí [5].

Strojové učenie je podoblasť umelej inteligencie, zaoberajúca sa metódami a algoritmami, ktoré umožňujú programu učiť sa a následne adekvátne reagovať na rôzne vstupné hodnoty bez toho, aby bol na to explicitne naprogramovaný, iba na základe informácií, ktoré sa naučil. Algoritmy strojového učenia využívajú prvky matematickej štatistiky, metódy štatistickej analýzy a data miningu [6].

Algoritmy môžu vykonávať tri základné problémy (úlohy), a to klasifikáciu, regresiu alebo zhlukovanie. Tieto algoritmy sa delia do troch kategórií podľa spôsobu učenia:

- Supervised learning
- Unsupervised learning
- Reinforcement learning

1.2. Supervised ML

Algoritmy supervised ML sú navrhnuté tak, aby sa učili na príkladoch. Názov supervised ML vychádza z predstavy, že tréningovanie tohto typu algoritmu je založené na dohľade učiteľa (supervisor) nad celým procesom. Pri tréningovaní algoritmu budú tréningové dáta pozostávať zo vstupov spárovaných so správnymi výstupmi. Počas tréningu algoritmus vyhľadáva vzory v dátach, ktoré korelujú s požadovanými výstupmi. Po tréningu algoritmus preberie nové vstupy a na základe údajov z predchádzajúceho tréningu určí, ktoré výstupné označenie budú mať nové klasifikované vstupy. Cieľom supervised ML modelu je správne predpovedať výstupnú triedu pre novo prezentované vstupné údaje. Vo svojej základnej podobe je možné supervised výučbový algoritmus zapísať jednoducho ako

$$Y = f(x)$$

kde Y je predikovaný výstup, ktorý je určený funkciou, ktorá priradí triedu vstupnej hodnoty x . Funkcia používaná na pripojenie vstupných funkcií k predpovedanému výstupu je vytvorená modelom strojového učenia počas tréningu [7].

Supervised ML sa dá rozdeliť do dvoch podkategórií a to klasifikácia a regresia. Regresia, rozhodovacie stromy, neurónové siete, metóda podporných vektorov (SVM), Naive Bayes sú príkladmi supervised ML [5]. Supervised ML je najjednoduchšou podkategóriou strojového učenia a slúži ako úvod do strojového učenia. Ide o najbežnejšie používanú formu strojového učenia a ukázalo sa, že je vynikajúcim nástrojom v mnohých oblastiach.

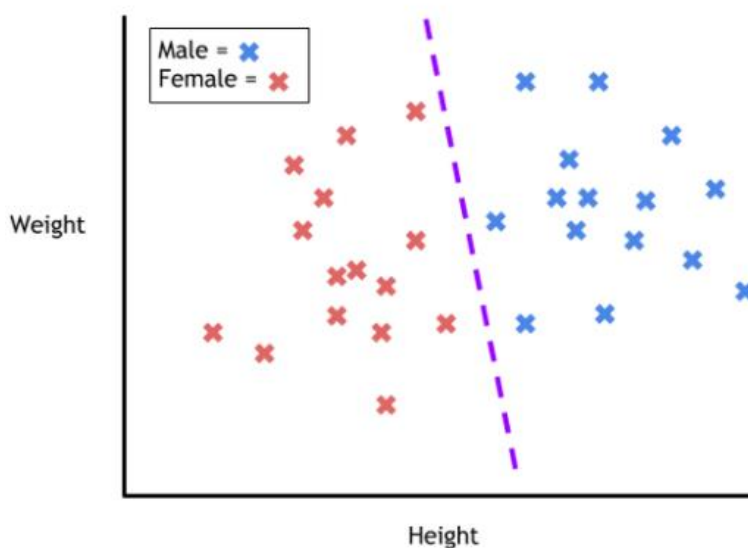
1.3. Klasifikácia

Úlohou klasifikačného algoritmu je vziať vstupnú hodnotu a priradiť jej triedu alebo kategóriu, do ktorej zapadá na základe poskytnutých údajov o tréningu [7]. Najbežnejším príkladom klasifikácie je určenie, či je e-mail spam alebo nie. Ak uvažujeme, že máme na výber z dvoch tried (spam, alebo nie spam), tento problém sa nazýva problém s binárnou klasifikáciou. Algoritmus dostane údaje o tréningu s e-mailmi, z ktorých jedna skupina spamom je a druhá nie. Model nájde vlastnosti v dátach, ktoré korelujú s ktoroukoľvek triedou, a vytvorí funkciu uvedenú vyššie: $Y = f(x)$. Potom, keď bude model vystavený novým e-mailom, model použije túto funkciu na zistenie, či je e-mail spam [7].

Problémy s klasifikáciou je možné vyriešiť pomocou veľkého množstva algoritmov. Akýkoľvek algoritmus, ktorý sa rozhodneme použiť, závisí od údajov a situácie. Tu je niekoľko populárnych klasifikačných algoritmov:

- Linear Classifiers,
- Support Vector Machines,
- Decision Trees,
- K-Nearest Neighbor,
- Random Forest.

Obrázok 2 vyjadruje grafické znázornenie klasifikácie. Hyperrovina (ang. hyperplane) klasifikovala súbory dát do svojich príslušných tried Male a Female. Hyperrovina v p dimenzionálnom priestore je podobný útvar, ale v $p-1$ dimenzionálnom priestore. Hyperrovina 2-dimenzionálneho priestoru musí byť 1-dimenzionálna a teda je to priamka [4].



Obrázok 2. Hyperrovina

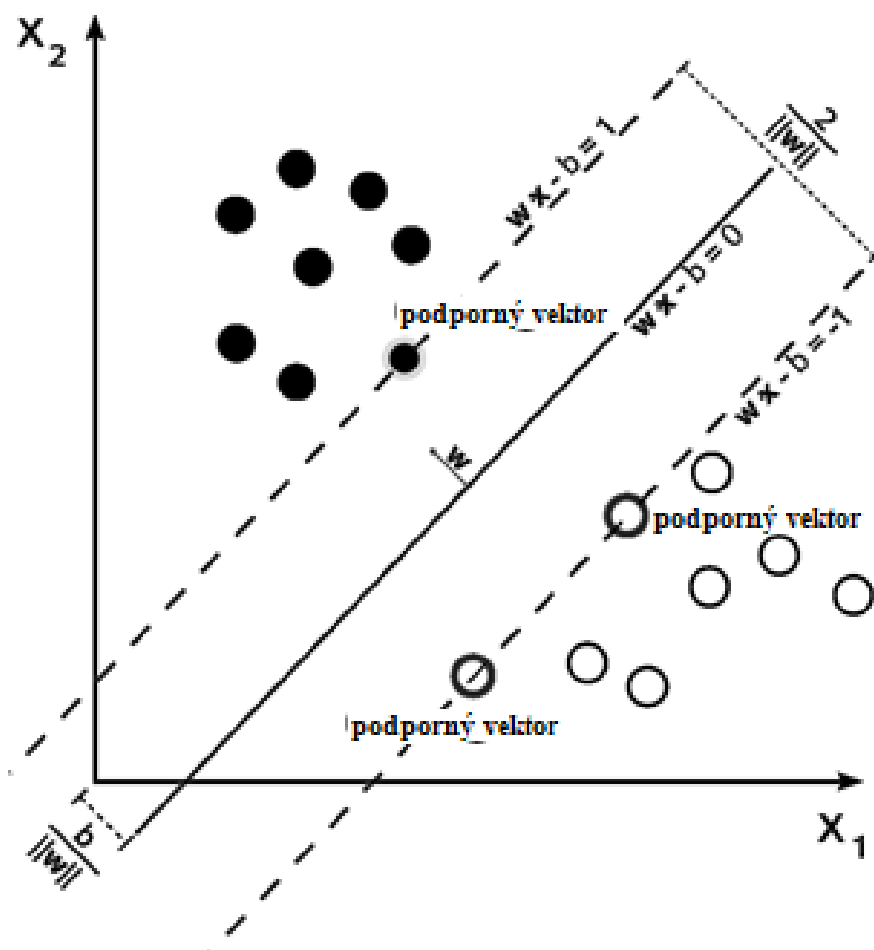
1.4. Klasifikácia a metóda podporných vektorov

Metóda podporných vektorov, po anglicky Support Vector Machines, slúži najmä na klasifikáciu, ale aj regresnú analýzu. Ide o klasifikačný algoritmus s cieľom rozdeliť dáta do dvoch tried. V prípade, že sú triedy lineárne separovateľné, hovoríme o lineárnej SVM, v prípade, že nie sú, pojem sa mení na nelineárna SVM alebo kernelová SVM. Je to jeden z najlepších a najpopulárnejších algoritmov strojového učenia. SVM vzniklo ako zovšeobecnenie jednoduchého a veľmi intuitívneho klasifikátora maximálneho rozpätia, tzv. Maximal Margin Classifier [8].

1.4.1. Rozdelenie dát vo vektorovom priestore

Pre lepšiu predstavu uvidíme jednoduchý príklad. Trénovacie dáta sú reprezentované dvojicou príznačkov x_1 , x_2 (príznačky sú napr. výška a šírka objektov snímaných kamerou alebo údaje pri chemickom rozbere krvi pri lekárskej diagnostike) rozdelených do dvoch kategórií. Tieto dáta môžeme znázorniť ako body v dvojrozmernom vektorovom priestore, zobrazené na Obrázku 3. Vo všeobecnosti na oddelenie tried môže existovať nekonečne veľa oddeľovacích hyperplôch, avšak pri metóde SVM hľadáme takú hranicu, ktorá vymedzuje po celej dĺžke čo najširšiu vzdialenosť medzi hraničnými bodmi jednotlivých tried, ako je vidno na obrázku nižšie. Hyperplocha tiež môže byť označená ako tzv. rozhodovacia hranica (angl. Decision boundary), ktorá oddeľuje na bodovom diagrame dve triedy a určuje, ktoré body patria do ktorej triedy [8]. Pri lineárne separovateľných dátach môžeme body rozdeliť pomocou priamky (resp. hyperplochy v prípade viacrozmerných dát). Triedy, ktoré sa nachádzajú na vymedzenej hranici, nazývame podporné vektory a šírka hranice reprezentuje kritériálnu funkciu.

Mechanizmus podporných vektorov (SVM) sa snaží nájsť hyperplochu čo najviac vzdialenú od trénovacích vzoriek dvoch tried. Podpornými vektormi sú potom tie vzorky každej triedy, ktoré sú najbližšie k oddeľujúcej hyperploche. Vzdialenosť medzi podpornými vektormi rôznych tried sa nazýva rozpätie [9]. Pre takúto hyperplochu platí: $w^T x + b = \pm 1$ a rozpätie $\frac{2}{||w||}$.

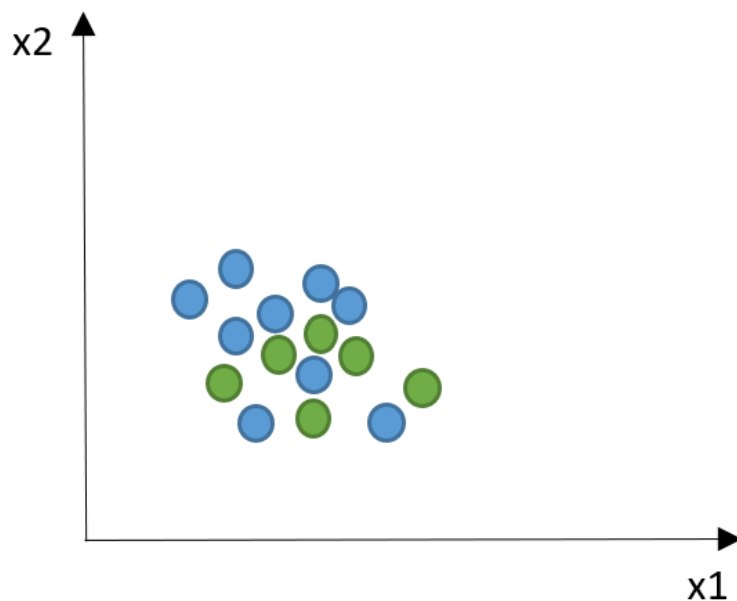


Obrázok 3. Mechanizmus SVM [8]

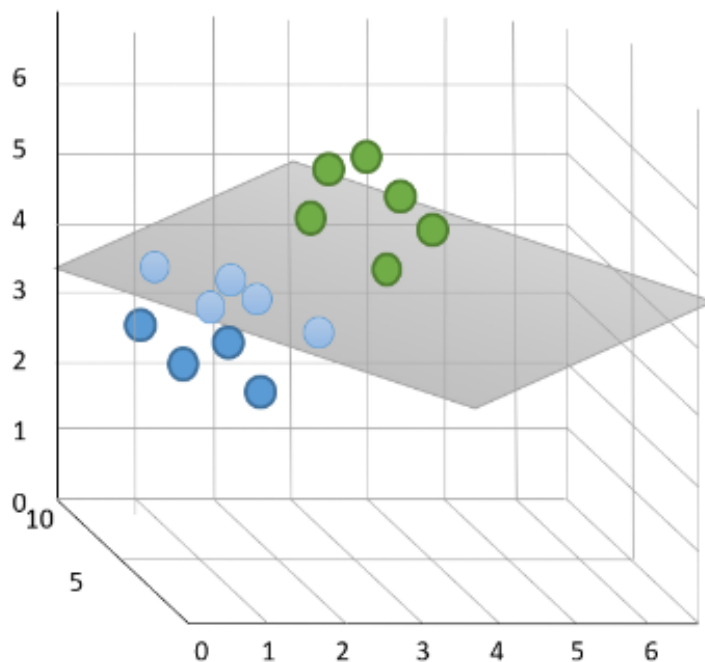
Na Obrázku 3 je znázornený ideálny stav, ktorý nastane málokedy. Častejšie sa stáva, že časť dát jednotlivých tried zasahuje do opačnej triedy. V takomto prípade pri trénovaní hľadáme maximum kriteriálnej funkcie, ktorej hodnoty sú penalizované v závislosti od počtu nesprávne klasifikovaných dát.

1.4.2. Kernel

V prípade, keď dáta vo vektorovom priestore nie je možné lineárne separovať (Obrázok 4), použije sa tzv. kernel trik. Ide o metódu, kde sa pomocou nelineárnej kernel funkcie dáta transformujú do nového priestoru s vyššou dimenziou, v ktorom už je možné nájsť deliacu hyperplochu, ako je znázornené na Obrázku 5.



Obrázok 4. Hyperplocha s neoddeliteľnými dátami [10]



Obrázok 5. Kernel trik [10]

Existuje niekoľko druhov kernelu a výber danej metódy závisí od usporiadania tried. Medzi základné patria:

- Lineárny kernel
- RBF (Gaussian Radial Basis function)
- Polynomiálny kernel
- Sigmoidálny kernel

1.4.3. Klasifikátor podporných vektorov

Ak sú trénovacie príklady oboch tried v priestore premiešané, a teda priestor nie je separabilný, potom sa klasifikátor maximálneho rozpätia nemôže použiť. Ale je možné rozšíriť tento koncept použitím takzvaných mäkkých okrajov (ang. soft margin), na ktorých je vybudovaný klasifikátor podporných vektorov. Klasifikátor podporných vektorov (ang.

Support Vector Classifier) je zariadenie vykonávajúce klasifikáciu. Klasifikátor podporných vektorov vznikol ako rozšírenie klasifikátora maximálneho rozpätia a je možné ho aplikovať na väčšinu dát [4].

Klasifikátor nastavujeme (trénujeme) pomocou trénovacej množiny – nastavujú sa vnútorné parametre klasifikátora. Tento proces sa nazýva učenie a jeho cieľom je čo najmenšia chyba na trénovacej množine. Klasifikátor môže mať aj hyperparametre, ktoré nastavuje pred učením používateľ, a tie sa testujú na validačnej množine. Validačná množina je časť trénovacej množiny, obyčajne v pomere trénovacia:validačná 80:20. Kvalita klasifikátora sa testuje na testovacej množine, v ktorej sú dáta, ktoré klasifikátor nemal k dispozícii. Chyba dosiahnutá na testovacej množine sa nazýva generalizačná chyba a cieľom je, aby bola čo najmenšia [9].

1.5. Klasifikácia a Naive Bayes

Bayesová teoréma nesie meno po svojom autorovi Thomasovi Bayesovi, anglickom duchovnom žijúcom v 18. storočí. Bayes nikdy nepublikoval svoju teorému. Vydaná a pomenovaná bola až po jeho smrti jeho priateľom Richardom Priceom [11]. Bayesovské klasifikátory sú štatistické klasifikátory, ktoré predikujú pravdepodobnosti, s ktorými daný príklad patrí do tej-ktorej triedy. Vychádzajú pritom z určenia podmienených pravdepodobností jednotlivých hodnôt atribútov pre rôzne triedy. Naivný Bayesovský klasifikátor vychádza z predpokladu nezávislosti atribútov medzi sebou. To znamená, že efekt, ktorý má hodnota každého atribútu na danú triedu, nie je ovplyvnený hodnotami ostatných atribútov. Kvôli tomuto zjednodušeniu je tento klasifikátor nazývaný ako naivný [12].

Nech X je príklad s neznámym zaradením do triedy C . Nech H je hypotéza, že X patrí do určitej triedy C_i . Cieľom klasifikácie je určenie podmienenej pravdepodobnosti $P(X|H)$. Ide o posteriórnu pravdepodobnosť hypotézy H za podmienky X . V tomto prípade pravdepodobnosti toho, že X patrí do triedy C_i .

Predpokladajme, že máme súbor dát o futbalových hráčoch, ktoré sú popisované atribútmi vek a počet strelených gólov, napr. že hráč X má 35 rokov a v sezóne nastrielal 15 gólov. Predpokladajme, že H je hypotéza tvrdiaca, že jeho tím vyhrá turnaj. Potom $P(X|H)$

predstavuje pravdepodobnosť, že tím hráča X vyhrá na základe toho, že vieme hráčov vek a počet strelených gólov. Naopak, $P(H)$ je apriórna pravdepodobnosť H . V našom prípade ide o pravdepodobnosť, že tím vyhrá bez uvažovania veku hráča, počtu strelených gólov, alebo inej vlastnosti. $P(X|H)$ je posteriórna pravdepodobnosť X podmienenej H . Ide o pravdepodobnosť, že hráč X má 35 rokov a strelil 15 gólov, pričom vieme, že jeho tím vyhrá turnaj. $P(X)$ je apriórna pravdepodobnosť X . V našom príklade by išlo o pravdepodobnosť, že osoba z nášho súboru hráčov má 35 rokov a strelila 15 gólov. Podľa Bayesovskej teóremy môžeme vypočítať žiadanú posteriórnu pravdepodobnosť $P(H|X)$ na základe pravdepodobností $P(H)$, $P(X|H)$ a $P(X)$ ako

$$P(H|X) = \frac{P(X|H) * P(H)}{P(X)}$$

1.5.1. Naivný Bayesovský klasifikátor

1. Nech D je trénovacia množina príkladov a ich zaradení do tried. Každý príklad je reprezentovaný n -rozmerným vektorom vlastností $X=(x_1, x_2, \dots, x_n)$, patriacim n -atribútom A_1, A_2, \dots, A_n .

2. Majme m tried C_1, C_2, \dots, C_m . Neznámy príklad X bude klasifikovaný do triedy s najväčšou posteriornou pravdepodobnosťou

$$P(C_i|X) > P(C_j|X) \quad \text{pre } 1 \leq j < m, j \neq i.$$

3. Podľa Bayesovej teóremy $P(C_i|X) = \frac{P(X|C_i) * P(C_i)}{P(X)}$. Keďže $P(X)$ je konštantná pre všetky triedy C_i , stačí maximalizovať výraz $P(X|C_i) * P(C_i)$. $P(C_i)$, ktorý vyjadruje pravdepodobnosť, s akou ľubovoľne zvolený prvok patrí do triedy C_i . Používame pri tom vzťah $P(C_i) = \frac{s_i}{S}$, kde s_i predstavuje počet príkladov triedy C_i a S je počet všetkých príkladov. Maximalizujeme teda len člen $P(X|C_i)$.

4. Vzhľadom na to, že pri veľkom počte atribútov by bolo výpočtovo náročné určovať $P(X|C_i)$, vychádzame z predpokladu, že nezávislosti atribútov pri príslušnosti k danej cieľovej triede, teda

$$P(X|C_i) = \prod_{k=1}^n P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \dots \times P(x_n|C_i)$$

Pravdepodobnosť $P(x_1|C_i) \times P(x_2|C_i) \dots \times P(x_n|C_i)$ vieme určiť z trénovacej množiny, pričom:

a) Ak A_k je kategorický atribút, potom $P(x_k|C_i) = \frac{s_{ik}}{S_i}$, kde s_{ik} je počet príkladov z trénovacej množiny patriacich triede C_i , pre ktoré $A_k=x_k$. S_i je počet vzoriek z trénovacej množiny patriacich triede C_i [13].

b) Pre spojité atribúty A_k sa zvyčajne predpokladáme Gaussovo normálne rozdelenie hodnôt, a potom

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}) = \frac{1}{\sqrt{2\pi}\sigma_{C_i}} \cdot e^{-\frac{(x-\mu_{C_i})^2}{2\sigma_{C_i}^2}}$$

kde μ_{C_i} je stredná hodnota a σ_{C_i} rozptyl hodnôt atribútu A_k trénovacích vzoriek z triedy C_i .

5. Pri klasifikácii neznámeho príkladu X je vypočítaný člen $P(X|C_i)P(C_i)$ pre každú triedu C_i . Príkladu X je priradená trieda C_i , vtedy a len vtedy ak

$$P(C_i|X)P(C_i) > P(X|C_j)P(C_j) \quad \text{pre } 1 \leq j < m, j \neq i.$$

Inými slovami, príklad X je zaradený do tej triedy, pre ktorú je $P(X|C_i)P(C_i)$ maximálne [13].

1.6. Hodnotiace metriky a výkon modelu

Hodnotiace metriky (ang. Evaluation metrics) sú také hodnoty, ktoré vyjadrujú ako dobre funguje model strojového učenia. Pod slovíčkom funguje myslíme, nakoľko presný náš model je, akú má chybovosť pri predikcii či klasifikácii. Model je len natoľko dobrý, ako si ho spravíme a všetko závisí od kvality dát a od krokov, ktoré zahrnieme do vytvárania modelu. Medzi ukazovatele môže patriť rýchlosť modelu pri predikcii, počet úspešne klasifikovaných objektov, ale neexistuje univerzálna definícia alebo metrika výkonu modelu strojového učenia. Metriky výkonu veľmi závisia od domény a konečného účelu budovaného modelu.

Výkonnosť machine learning modelu je taká dobrá, aká je len pri konkrétnej úlohe, ale definícia dobrého môže mať veľa podôb. Dobrým modelom môže byť model, ktorý dobre predpovedá, model, ktorý sa trénuje rýchlo, model, ktorý nájde spoľahlivé riešenie, alebo akákoľvek kombinácia vyššie uvedených.

1.6.1. Confusion matrix

Confusion matrix alebo matica chýb je tabuľka, ktorá sa často používa na popis výkonu klasifikačného modelu (alebo klasifikátora) na súbore testovacích údajov, pre ktoré sú známe skutočné hodnoty. Ide o koreláciu medzi predikciami modelu a aktuálnym označením triedy dátových bodov [14]. V Tabuľke 1 vidíme model confusion matrix, kde máme triedy starý a mladý.

	Trieda mladý	Trieda starý
Predikovaný mladý	Počet správne predikovaných	Počet nesprávne predikovaných
Predikovaný starý	Počet nesprávne predikovaných	Počet správne predikovaných

Tabuľka 1. Confusion matrix [14]

Uvažujme, že máme model na detekciu mladých a starých zákazníkov. Dataset rozdelíme na testovaciu a trénovaciu časť a testovacie dáta majú dĺžku 100 dátových bodov. Z týchto 100, 70 dátových bodov je označených ako mladý a zvyšných 30 ako starý. Na obrázku vidíme confusion matrix pre tento problém binárnej klasifikácie.

	Trieda mladý	Trieda starý
Predikovaný mladý	64	3
Predikovaný starý	6	27

Tabuľka 2. Confusion matrix starý mladý [14]

Confusion matrix znázornený v Tabuľke 2 vyššie interpretujeme nasledovne:

- Zo 70 údajových bodov, patriacich triede mladý, klasifikoval model správne 64 bodov ako mladý a nesprávne 6 ako starý.
- Z 30 údajových bodov, patriacich triede starý, klasifikoval model nesprávne 3 ako mladý a 27 správne ako starý.

Ak chceme, aby bol náš model inteligentný, musí model predikovať správne. To znamená, že počty správne predikovaných by mali byť čo najvyššie čísla a zároveň počet nesprávne predikovaných čo najnižšie. Pre náš prípad modelu detekcie mladý/starý môžeme vypočítať tieto pomery:

- Úspešnosť pri detekcii mladý = 91,4% (správne klasifikovaní ako mladý/mladý).
- Úspešnosť pri detekcii starý = 90% (správne klasifikovaní ako starý/starý).
- Chyba pri detekcii mladý = 10% (nesprávne klasifikovaní mladý/starý).
- Chyba pri detekcii starý = 8,6% (nesprávne klasifikovaný starý/mladý).

1.6.2. Accuracy

Doslovný význam hovorí o Accuracy ako o miere presnosti nášho modelu.

$$Accuracy = \frac{\text{počet správne predikovaných}}{\text{počet všetkých predikcií}}$$

Accuracy je jednou z najjednoduchších metrík výkonu, ktorú môžeme použiť. Accuracy môže niekedy viesť k falošným ilúziám o našom modeli, a preto by sme mali najskôr poznať použitý súbor údajov a použitý algoritmus, a až potom rozhodnúť, či presnosť použijeme alebo nie [14]. Predtým, ako prejdeme k prípadom zlyhania presnosti, predstavíme dva typy množín údajov:

- Vyvážené: množina údajov, ktorá obsahuje takmer rovnaké položky pre všetky triedy. Napríklad z 1000 dátových bodov o zákazníkoch je 600 mladých a 400 starých.
- Nevyvážené: množina údajov, ktorá obsahuje neobjektívne rozdelenie záznamov smerom ku konkrétnej triede. Napríklad z 1000 záznamov je 990 mladých zákazníkov, 10 starých zákazníkov. Accuracy sa nikdy nepoužíva ako opatrenie pri práci s nevyváženou testovacou súpravou [14].

1.6.3. Precision a Recall

Precision (ang. positive predictive value) je hodnota, ktorá tiež udáva presnosť machine learning modelov. Je to pomer všetkých záznamov jednej triedy a celkových predikcií tejto triedy. V zásade nám hovorí, koľkokrát bola naša predpoveď konkrétnej triedy skutočne správna. Uvažujme že máme dve triedy C1 a C2.

$$Precision = \frac{\text{počet záznamov triedy } C1}{\text{počet všetkých predikcií triedy } C1}$$

$$Recall = \frac{\text{počet záznamov triedy } C1}{\text{počet správnych predikcií } C1 + \text{počet nesprávnych predikcií } C2}$$

1.6.4. Skóre F1

Skóre F1 (F1 score) poskytuje mieru toho, ako dobre dokáže klasifikátor klasifikovať pozitívne prípady. Skóre F1 sa počíta z harmonického priemeru precision a recall. Skóre F1=1 znamená, že precision aj recall sú dokonalé a model správne identifikoval všetky C1 prípady a neoznačil C2 prípad ako C1 prípad. Ak je precision alebo recall veľmi nízka hodnota, prejaví sa tak, že F1 sa bude blížiť k 0.

$$F1 = 2 \frac{precision * recall}{precision + recall}$$

Aby sme to pochopili, pozrime sa na tento príklad. Vyhľadávanie v službe Google nám vráti 40 stránok, ale iba 30 bolo relevantných. Náš známy, ktorý je zamestnancom spoločnosti Google, nám však povedal, že pre tento dopyt bolo celkovo 100 relevantných stránok. Takže jeho precision je $30/40 = 3/4 = 75\%$, zatiaľ čo recall je $30/100 = 30\%$. V tomto prípade precision odpovedá na otázku: Ako nápomocné sú výsledky vyhľadávania? Recall vyjadruje: Ako úplné sú výsledky [14]?

1.7. Nástroje ML

V súčasnosti sa takmer v každej sfére nesmierne zvýšilo prijatie AI alebo strojového učenia, rovnako ako rovnakým spôsobom vzrástol aj počet softvérových nástrojov pre vývojárov. Očakáva sa, že do roku 2025 bude svetový trh s AI takmer 60 miliárd dolárov.

V roku 2016 to bolo 1,4 miliardy dolárov [15]. Programové nástroje, ku ktorým môžeme pristupovať pri riešení úloh strojového učenia:

- cloudové riešenia (SaaS),
- desktopové riešenia (programy nainštalované na osobnom PC),
- knižnice programovacích jazykov (zdroj programových funkcií).

Machine learning sa viac a viac používa či už v zdravotníctve, ekonómii, športe a každodenne pomáha ľuďom po celom svete. Globálny HDP porastie do roku 2030 vďaka AI o 15,7 bilióna dolárov [16]. Aj kvôli tomuto faktu veľké spoločnosti ako Google, Amazon, Microsoft investujú nemalé peniaze do vývoja svojich ML platforiem. V tabuľke si uvedieme najpopulárnejšie nástroje v súčasnosti.

Tensorflow	Open source knižnica strojového učenia, ktorá pomáha vyvíjať modely ML. Vyvinul ju tím Google. Má flexibilnú schému nástrojov, knižníc a zdrojov, ktorá umožňuje vedcom a vývojárom vytvárať aplikácie strojového učenia.
Google Cloud ML engine	Je to cloudová platforma. Cloudové platformy sú primárne určené na sofistikované algoritmy a zložitejšie modely.
Amazon Machine Learning	Cloudová platforma od Amazonu
Apache Mahout	Distribuovaný framework určený na lineárnu algebru
Shogun	Knižnica ML napísaná v jazyku C++
Scikit-learn	Otvorený softvér, knižnica pod licenciou BSD
Weka	Programové riešenie z univerzity Wakaito
H2O	Projekt, zaoberajúci sa ML. Obsahuje knižnicu h2o, desktopovú verziu H2O flow, cloudové riešenie DriverlessAI

1.8. Scikit-learn + Jupyter

Scikit-learn je modul programovacieho jazyka Python pre strojové učenie postavený na vrchole SciPy a je distribuovaný pod licenciou BSD s 3 klauzulami. Projekt začal v roku 2007 David Cournapeau ako projekt Google Summer of Code a odvtedy sa na ňom podieľalo veľa dobrovoľníkov. Na stránke About Us nájdeme zoznam hlavných prispievateľov. V súčasnosti ju udržiava tím dobrovoľníkov [17]. Scikit-learn je open-source knižnica pre strojové učenie. Jej knižnica pre programovací jazyk Python, obsahuje veľa efektívnych nástrojov pre strojové učenie a štatistické modelovanie, napríklad klasifikáciu, regresiu, klastrovanie.

Jupyter nám umožňuje programovať vo webovom prehliadači. Je to mix inštrukcií kódu a výstupu a všetky tieto informácie sú zobrazené na webovej stránke, čím je veľmi užitočným nástrojom na písanie kódu. Je používaný hlavne vedcami a výskumnými pracovníkmi. Jupyter je webová aplikácia s otvoreným zdrojovým kódom, ktorá umožňuje vytvárať a zdieľať dokumenty, ktoré obsahujú živý kód, rovnice, vizualizácie a naratívny text. Medzi použitia patrí:

- čistenie a transformácia údajov,
- numerická simulácia,
- štatistické modelovanie,
- vizualizácia údajov,
- strojové učenie.

Celkovo projekt Jupyter existuje s cieľom vyvinúť softvér s otvoreným zdrojovým kódom, otvorené štandardy a služby pre interaktívne výpočty v desiatkach programovacích jazykov.

1.9. Weka

Waikato Environment for Knowledge Analysis (Weka) je softvér strojového učenia s otvoreným zdrojovým kódom, ku ktorému je možné pristupovať prostredníctvom grafického používateľského rozhrania a štandardných terminálových aplikácií alebo Java API. Je široko používaný na výučbu, výskum a priemyselné aplikácie, obsahuje množstvo zabudovaných nástrojov pre štandardné úlohy strojového učenia a navyše poskytuje transparentný prístup k známym balíkom nástrojov, ako sú scikit-learn, R a Deeplearning⁴. Obsahuje zbierku vizualizačných nástrojov a algoritmov pre analýzu dát a prediktívne modelovanie spolu s grafickými užívateľskými rozhraniami pre ľahký prístup k týmto funkciám.

Weka by sme mohli charakterizovať ako nástroj ML, kde nepotrebujeme programátorské schopnosti na to, aby sme dokázali vytvoriť žiadaný model. Tento nástroj z dielne novozélandskej univerzity Waikato podporuje úlohy data miningu, dátový preprocessing, clustering, klasifikáciu, regresiu, sekvenčné modelovanie a rôzne vizualizácie dát. Všetky tieto techniky predpokladajú, že údaje sú k dispozícii vo forme súboru alebo databázovej relácie, kde je každý údajový bod opísaný pevným počtom atribútov (zvyčajne číselné alebo nominálne atribúty). Weka poskytuje prístup k databázam SQL pomocou Java Database Connectivity [18]. Keď sa bavíme o dátach vo forme súboru, Weka podporuje svoj vlastný formát súborov s koncovkou .arff. Weka je:

- open source nástroj GNU General Public License,
- portabilná, pretože je plne implementovaná v programovacom jazyku Java, a teda beží na takmer akejkolvek modernej výpočtovej platforme (Windows, Linux, OSX),
- komplexná zbierka techník spracovania a modelovania údajov,
- ľahko použiteľná vďaka grafickým užívateľským rozhraniam.



Obrázok 6. Weka GUI

1.10. H2O

H2O je rýchla, škálovateľná platforma určená na strojové učenie s otvoreným zdrojovým kódom a pre smart aplikácie. Využívajú ju podniky ako PayPal, Nielsen Catalina, Cisco pre jej schopnosť spracovávať obrovské množstvo dát a pre jej výpočtovú silu. Pre vývojárov aplikácií sú ľahko dostupné pokročilé algoritmy, ako sú Deep Learning, Boosting a Bagging Ensembles, ktoré umožňujú vytvárať smart aplikácie pomocou elegantných API rozhraní. H2O spolu s ich zákazníkmi vytvorili prediktívne mechanizmy špecifické pre jednotlivé domény ako odporúčania, obrat zákazníkov, sklon k nákupu, dynamické ceny a zisťovanie podvodov pre poisťovacie, zdravotné, telekomunikačné, reklamné, maloobchodné a platobné systémy [19].

Pomocou techník kompresie v pamäti dokáže H2O zvládnuť miliardy dátových riadkov v pamäti, dokonca aj s pomerne malým klastrom. Platforma obsahuje rozhrania pre R, Python, Scala, Java, JSON a JavaScript, spolu so zabudovaným webovým rozhraním

Flow, ktoré uľahčuje spájanie kompletných analytických pracovných postupov nie iba pre inžinierov. H2O Flow je v podstate projekt H2O pre neprogramátorov.

H2O implementuje takmer všetky bežné algoritmy strojového učenia, ako napríklad generalizované lineárne modelovanie (lineárna regresia, logistická regresia atď.), Naive Bayes, analýza hlavných komponentov, časové rady, k-means klastrovanie a ďalšie. H2O tiež implementuje najlepšie algoritmy vo svojej triede, ako sú Random Forest, Gradient Boosting a Deep Learning. Používatelia môžu zostaviť tisíce modelov a porovnať ich, aby dosiahli najlepšie výsledky predikcií [19].

H2O je navrhnuté pre fyzikov, matematikov, počítačových a dátových vedcov. Aj obri zo Stanfordskej univerzity Stephen Boyd, Trevor Hastie a Rob Tibshirani spolupracujú s H2O na rozvoji strojového učenia a umelej inteligencie. Z H2O sa stal fenomén, ktorý rastie medzi dátovou komunitou stokrát a v súčasnosti ho využíva viac ako 12 000 používateľov nasadených v 2000 plus korporáciách [19].

Základný kód H2O je napísaný v jazyku Java. Algoritmy sú implementované nad distribuovaným rámcom a využívajú rámec Java Fork/Join pre viacvláknové spracovanie. Dáta sa čítajú paralelne a distribuujú sa po klastri a komprimovaným spôsobom sa ukladajú do pamäte v stĺpcovom formáte. H2O má zabudovanú inteligenciu na odhad schémy prichádzajúcej množiny údajov a podporuje čítanie dát z viacerých zdrojov v rôznych formátoch.

Rozhranie REST API umožňuje prístup ku všetkým možnostiam H2O z externého programu alebo skriptu vo formáte JSON cez HTTP. Rest API používa webové rozhranie H2O (Flow UI), väzbu R (H2O-R) a väzbu Python (H2O-Python). Rýchlosť, kvalita, jednoduché použitie a nasadenie modelu pre rôzne špičkové algoritmy ako sú Deep Learning, Tree Ensembles a GLRM robia z H2O veľmi vyhľadávané API pre vedu o big data.

2. Ciele a metodika práce

Zadaním diplomovej práce je porovnanie nástrojov machine learning na konkrétnom príklade. Cieľom práce je porovnanie efektivity voľne dostupných ML nástrojov. Vyhladáme voľne dostupnú databázu (napr. futbalových výsledkov) a aplikujeme rôzne voľne dostupné nástroje pre strojové učenie. Vyberieme vhodný typ úlohy (klasifikáciu, regresiu, clustering a pod.), ktorou sa budeme zaoberať, a budeme porovnávať efektívnosť zvolených nástrojov.

Pri práci budeme používať metodiku CRISP DM, ktorá poskytuje silné usmernenie aj pre najpokročilejšie súčasné aktivity v oblasti data science. Ide metodiku dolovania dát, známu ako Cross-industry standard process for data mining. Je to otvorený a štandardný procesný model, ktorý popisuje bežné prístupy používané expertmi na dolovanie dát. Je to najbežnejšie používaný analytický model .

2.1. Formulácia úlohy

Našou úlohou v tejto práci bude porovnanie 3 voľne dostupných nástrojov určených na machine learning na konkrétnom príklade. Zvolili sme nástroje Weka, H2O a Scikit-learn+Jupyter, v ktorých vytvoríme modely ML. Ich výkon overíme na 2 úlohách pomocou metrík výkonu vytvorených modelov. Vo všetkých modeloch využijeme princípy supervised learning.

Na prvom príklade budeme ilustrovať proces tvorby modelu ML, ktorý bude klasifikovať zápasy anglickej Premier League. V spomenutých nástrojoch budeme klasifikovať hodnotu výsledku na konci zápasu jednotlivých zápasov. V tomto prípade pôjde o klasifikáciu viacerých tried. Druhá úloha bude spočívať v tvorbe modelu schopného klasifikovať podvod vo finančných záznamoch. Budeme vytvárať model binárnej klasifikácie.

2.2. Prediktívne modelovanie

Prediktívne modelovanie využíva metódy matematickej štatistiky na predpovedanie nejakej udalosti. Najčastejšie udalosť (alebo aj vlastnosť), ktorú chceme predpovedať, je v

budúcnosti, ale prediktívne modelovanie sa môže aplikovať na akýkoľvek typ neznámej udalosti bez ohľadu na to, kedy nastala. Prediktívne modely sa často používajú na predpovedanie výkyvov trhu, stavu zásob, cien, správania sa zákazníkov, výsledkov, počasia a pomaly všetkého, čo poznáme, či už hovoríme o ekonómii, biológii, športe alebo o tom, či zajtra bude pršať.

Modely môžu používať jeden alebo viac klasifikátorov v snahe určiť pravdepodobnosť množiny údajov patriacich do iného súboru údajov. Napríklad model môže byť použitý na určenie, či je e-mail spam alebo „šunka“ (nie spam). V závislosti od definícií hraníc je prediktívne modelovanie synonymom, alebo sa do značnej miery prekrýva s oblasťou strojového vzdelávania, pretože sa bežne označuje v kontextoch akademického výskumu a vývoja. Keď je komerčne nasadené, prediktívne modelovanie sa často označuje ako prediktívna analytika [20].

Prediktívne modelovanie sa vo veľkej miere používa v analytickom riadení vzťahov so zákazníkmi a ťažbou dát na vytvorenie modelov na úrovni zákazníkov, ktoré opisujú pravdepodobnosť, že zákazník vykoná konkrétnu akciu [20]. Rozdelenie na Investopedia.sk hovorí, že existujú dve triedy ML modelov:

- Parametrické modely robia špecifické predpoklady týkajúce sa jedného alebo viacerých parametrov, ktoré charakterizujú základné distribúcie.
- Neparametrické modely zvyčajne zahŕňajú menej predpokladov štruktúry a distribučnej formy (ako parametrické modely), ale zvyčajne obsahujú silné predpoklady o nezávislosti.

2.3. Metodika CRISP DM

Proces alebo metodika CRISP-DM je popísaná v týchto šiestich hlavných krokoch:

- Obchodné porozumenie. Zameriavame sa na pochopenie cieľov a požiadaviek projektu (úlohy) z obchodného hľadiska. Tieto vedomosti formulujeme ako problém s dolovaním dát a v biznise sa vypracúva predbežný plán.

- Porozumenie údajom. Počnúc počiatocným zhromažďovaním údajov pokračujeme v činnostiach zameraných na oboznámenie sa s údajmi, identifikáciu problémov s kvalitou údajov a zisťovanie prvých poznatkov o údajoch. V tejto fáze môžeme objaviť aj zaujímavé podmnožiny a vytvoriť hypotézy o skrytých informáciách.
- Príprava údajov. Fáza prípravy údajov pokrýva všetky činnosti na zostavenie konečného súboru údajov z počiatocných nespracovaných údajov
- Modelovanie. Vyhodnotíme, vyberieme a použijeme príslušné techniky modelovania. Pretože niektoré techniky ako neurónové siete majú špecifické požiadavky týkajúce sa formy údajov, môže tu nastať krok späť k príprave údajov.
- Vyhodnotenie. Zostavíme a vyberieme modely, ktoré majú vysokú kvalitu, na základe vybraných funkcií. V tomto kroku otestujeme vytvorené modely. Následne tiež overíme, či modely dostatočne pokrývajú všetky kľúčové problémy. Konečným výsledkom je výber šampiónov spomedzi modelov.
- Nasadenie. Spravidla to bude znamenať nasadenie kódovej reprezentácie modelu do operačného systému. Patria sem aj mechanizmy na skórovanie alebo kategorizáciu nových údajov, keď vzniknú. Mechanizmus by mal využívať nové informácie pri riešení pôvodného problému. Dôležité je, že reprezentácia kódu musí obsahovať aj všetky kroky prípravy údajov vedúce k modelovaniu. Tým zaistíme, že model bude s novými nespracovanými údajmi zaobchádzať rovnakým spôsobom ako počas vývoja modelu.

Nejde o lineárny proces, ktorý začína jednou fázou a v každom kroku úhladne funguje v presnom poradí. Tieto fázy sú súčasťou prebiehajúceho cyklu analytickej činnosti a analytický tím môže medzi týmito fázami často pracovať tam a späť. Avšak zhruba povedané, proces začína konkrétnym obchodným problémom a vedie k vytvoreniu modelov a ich integrácii do bežných obchodných operácií [21].

3. Výsledky práce

Cieľom tejto práce je porovnať nástroje Sklearn+Jupyter, Weka a H2O. Pre každý z nástrojov vytvoríme klasifikačné modely z datasetov o futbalových zápasoch a transakciách kreditnými kartami. Tretia kapitola bude praktickou časťou tejto diplomovej práce. Vytvoríme modely a porovnáme metriky výkonu vytvorených modelov. Pri úlohe s futbalovým datasetom budeme klasifikovať hodnotu výsledku zápasu. Modely budú predikovať, či zápas skončí výhrou jedného z tímov alebo remízou. Pri tejto úlohe budeme podrobne rozoberať aj jednotlivé nástroje. V úlohe s datasetom údajov o kreditných kartách budeme klasifikovať podvodné transakcie. Nebudeme sa už podrobne venovať každému kroku ako sme to robili pri klasifikácii futbalových zápasov, ale budeme sa snažiť porovnávať nástroje v každom z nasledujúcich krokov:

- Import datasetu
- Analýza údajov
- Predspracovanie
- Trénovanie a testovanie
- Vyhodnotenie metrík výkonu

3.1. Dataset

Dataset (alebo množina údajov) je súbor údajov, obvykle uvádzaný v tabuľkovej forme. Každý stĺpec predstavuje konkrétnu premennú (atribút). Každý riadok zodpovedá danému členovi príslušnej množiny údajov. Dataset môže mať rôznu podobu, napríklad ako tabuľka, databáza, JSON objekt, XML súbor a podobne.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Div	Date	Time	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	HS	AS	HST	AST
2	E0	12/9/2020	12:30	Fulham	Arsenal	0	3	A	0	1	A	C Kavanag	5	13	2	6
3	E0	12/9/2020	15:00	Crystal Pal	Southamp	1	0	H	1	0	H	J Moss	5	9	3	5
4	E0	12/9/2020	17:30	Liverpool	Leeds	4	3	H	3	2	H	M Oliver	22	6	6	3
5	E0	12/9/2020	20:00	West Ham	Newcastle	0	2	A	0	0	D	S Attwell	15	15	3	2
6	E0	13/09/2020	14:00	West Bron	Leicester	0	3	A	0	0	D	A Taylor	7	13	1	7
7	E0	13/09/2020	16:30	Tottenham	Everton	0	1	A	0	0	D	M Atkinson	9	15	5	4
8	E0	14/09/2020	20:15	Brighton	Chelsea	1	3	A	0	1	A	C Pawson	13	10	3	5
9	E0	14/09/2020	18:00	Sheffield U	Wolves	0	2	A	0	2	A	M Dean	9	11	2	4
10	E0	19/09/2020	12:30	Everton	West Bron	5	2	H	2	1	H	M Dean	17	6	7	4
11	E0	19/09/2020	15:00	Leeds	Fulham	4	3	H	2	1	H	A Taylor	10	14	7	6
12	E0	19/09/2020	17:30	Man Unite	Crystal Pal	1	3	A	0	1	A	M Atkinson	13	14	4	5
13	E0	19/09/2020	20:00	Arsenal	West Ham	2	1	H	1	1	D	M Oliver	7	14	3	3
14	E0	20/09/2020	12:00	Southamp	Tottenham	2	5	A	1	1	D	D Coote	14	9	7	6
15	E0	20/09/2020	14:00	Newcastle	Brighton	0	3	A	0	2	A	K Friend	6	13	0	6
16	E0	20/09/2020	16:30	Chelsea	Liverpool	0	2	A	0	0	D	P Tierney	5	18	3	6
17	E0	20/09/2020	19:00	Leicester	Burnley	4	2	H	1	1	D	L Mason	14	16	6	5
18	E0	21/09/2020	18:00	Aston Villa	Sheffield U	1	0	H	0	0	D	G Scott	18	4	2	1
19	E0	21/09/2020	20:15	Wolves	Man City	1	3	A	0	2	A	A Marriner	10	14	1	9
20	E0	26/09/2020	12:30	Brighton	Man Unite	2	3	A	1	1	D	C Kavanag	18	7	5	3
21	E0	26/09/2020	15:00	Crystal Pal	Everton	1	2	A	1	2	A	K Friend	8	10	1	5
22	E0	26/09/2020	17:30	West Bron	Chelsea	3	3	D	3	0	H	J Moss	9	22	3	10
23	E0	26/09/2020	20:00	Burnley	Southamp	0	1	A	0	1	A	A Marriner	10	5	2	1
24	E0	27/09/2020	12:00	Sheffield U	Leeds	0	1	A	0	0	D	P Tierney	14	17	4	9
25	E0	27/09/2020	14:00	Tottenham	Newcastle	1	1	D	1	0	H	P Bankes	23	6	12	1
26	E0	27/09/2020	16:30	Man City	Leicester	2	5	A	1	1	D	M Oliver	16	7	5	7
27	E0	27/09/2020	19:00	West Ham	Wolves	4	0	H	1	0	H	M Atkinson	15	11	7	2

Obrázok 7. Zobrazenie datasetu v Exceli

3.2. Dataset futbalových zápasov

Na ukážku funkcionality neskôr popísaných nástrojov sme zvolili dataset s údajmi o futbalových zápasoch, na ktorom budeme demonštrovať použitie klasifikácie v jednotlivých nástrojoch. Zo stránky <https://www.football-data.co.uk/englandm.php> sme stiahli súbory v CSV formáte a zlepením dát dokopy sme vytvorili dataset s údajmi o viac ako 7000 zápasoch, ktoré sa odohrali v rokoch 2002 až 2020. Náhľad datasetu môžeme vidieť na Obrázku 7. Vidíme, že v prvom riadku sa nachádzajú mená atribútov alebo parametrov a od druhého riadka začínajú hodnoty týchto parametrov. Pri práci s dátami musíme dáta preskúmať a skontrolovať či sú dáta jednotné, aké hodnoty nadobúdajú jednotlivé atribúty, či nám niekde údaje nechýbajú. Taktiež musíme preskúmať, čo znamenajú jednotlivé údaje. V Obrázku 7 vyššie vidíme mená parametrov ako FTR, FTAG a musíme prísť na to čo tieto parametre vyjadrujú aby sme správne nastavili náš model. Na stránke <https://www.football-data.co.uk/englandm.php> sme našli textový súbor s poznámkami ku dátam. Najdôležitejšie parametre sú vypísané v tabuľke nižšie.

Kľúč	Hodnota
Div	Divízia
HomeTeam	Názov domáceho tímu
AwayTeam	Názov hostujúceho tímu
FTHG	Počet gólov domáceho tímu na konci zápasu
FTAG	Počet gólov hostujúceho tímu na konci zápasu
FTR	Konečný výsledok zápasu (H = výhra domáci, D = remíza, A = výhra hostia)
HTHG	Počet gólov domáceho tímu v polčase
HTAG	Počet gólov hostujúceho tímu v polčase
HTR	Výsledok v polčase zápasu (H = výhra domáci, D = remíza, A = výhra hostia)
Referee	Meno rozhodcu
HS	Počet striel domáceho tímu
AS	Počet striel hostujúceho tímu
HST	Počet striel na bránu domáceho tímu
AST	Počet striel na bránu hostujúceho tímu
HF	Počet faulov domáceho tímu
AF	Počet faulov hostujúceho tímu
HC	Počet rohových kopov domáceho tímu
AC	Počet rohových kopov hostujúceho tímu
HY	Počet žltých kariet domáceho tímu
AY	Počet žltých kariet hostujúceho tímu
HR	Počet červených kariet domáceho tímu
AR	Počet červených kariet hostujúceho tímu

Tabuľka 3. Vysvetlivky k datasetu futbalových zápasov

Náš dataset bude obsahovať tieto údaje o zápasoch. Chceme však dodať, že pri zbieraní dát sme narazili aj na atribúty ako dátum, čas a údaje o stávkových kurzoch, no

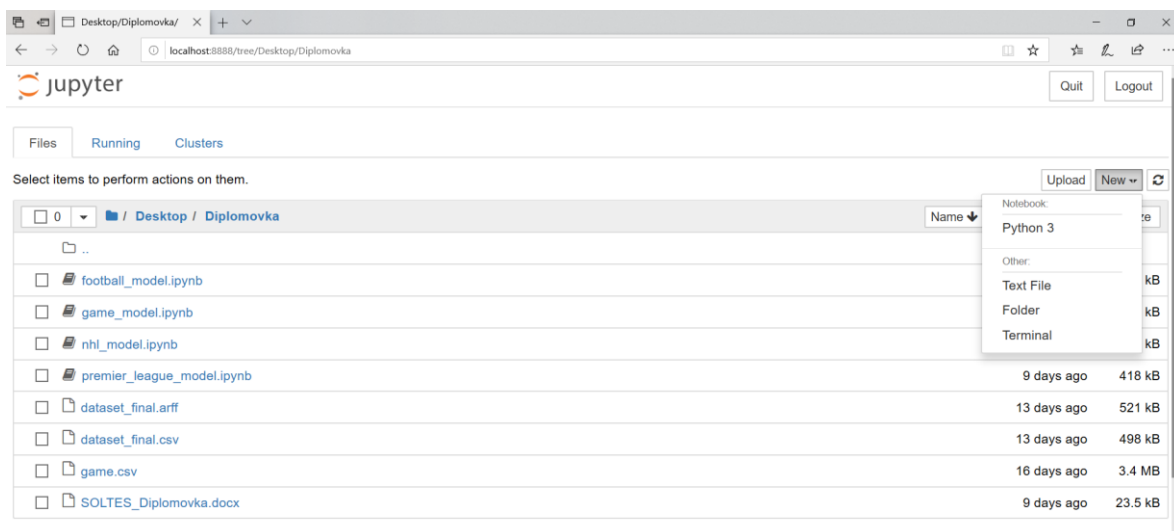
nepoužili sme ich v našom datasete z dôvodu neprítomnosti týchto dát v niektorých súboroch.

3.3. Predikcia výsledkov zápasov anglickej Premier League

Cieľom úlohy je vytvoriť model, ktorý bude schopný predikovať hodnotu výsledkov zápasov futbalovej Premier League, čo je najvyššia divízia najstaršej futbalovej ligy na svete. Pôjde o klasifikáciu viacerých tried, na ktorej chceme demonštrovať použitie ML vo vybraných nástrojoch. Multiclass alebo multinomial klasifikácia je klasifikácia inštancií do jednej z troch alebo viacerých tried. Futbalový zápas sa môže skončiť výhrou, prehrou alebo remízou a preto ide o problém klasifikácie 3 tried. Budeme sa snažiť použiť nástroje ML na riešenie tohto problému a z existujúcich techník klasifikácie viacerých tried si vyberieme tie, ktoré spadajú do skupiny rozšírení existujúcich binárnych klasifikátorov. Konkrétny problém budeme riešiť metódou podporných vektorov a algoritmom Naive Bayes.

3.4. Scikit-learn+Jupyter a dataset futbalových zápasov

V prvom kroku stiahneme inštalačný súbor Jupyter z oficiálnej stránky. Po inštalácii Jupyter spustíme pomocou príkazového riadka príkazom `jupyter notebook`. Na spustenie môžeme použiť napríklad program Power Shell a po zadaní príkazu sa v našom systéme spustí webový server a na URL <http://localhost:8888>. Túto URL otvoríme v ľubovoľnom webovom prehliadači, kde sa nám zobrazí grafické prostredie na Obrázku 8. V karte Files sa dokážeme pohybovať po súborovom systéme a tlačidlom New vytvoríme nový súbor. Súbory sú uložené vo formáte ipython notebook file .ipynb a do týchto súborov píšeme kód.



Obrázok 8. Rozhranie Jupyter Notebook

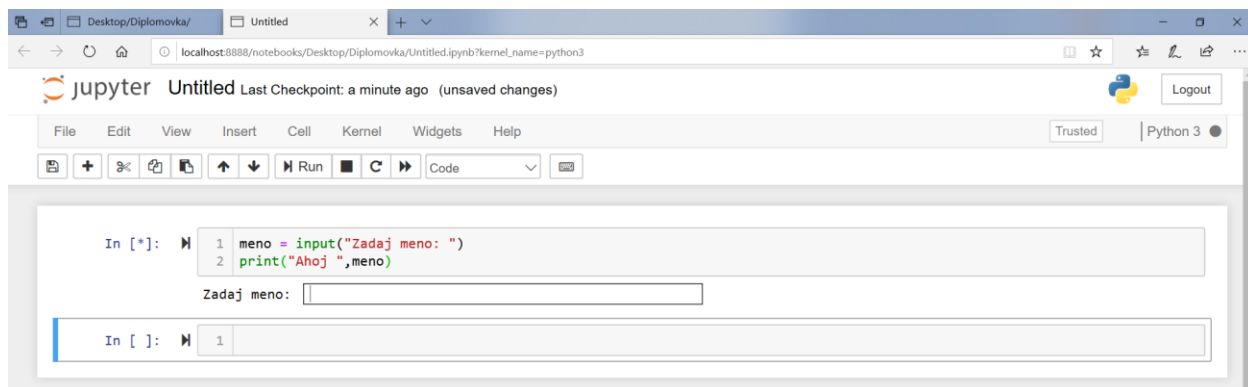
Keď otvoríme takýto súbor, v konzole sa nám zobrazí riadok ktorý ukazuje, že Jupyter spustil kernel. Kernel alebo jadro definuje programovacie prostredie. Jupyter je vyvinutý tak, aby mohol spúšťať programy v Pythone, Jave a Scale. My máme nastavený Jupyter tak, aby spúšťal programy v Pythone.

```
To access the notebook, open this file in a browser:
  file:///C:/Users/JicchagSoltes/AppData/Roaming/jupyter/runtime/nbserver-68020-open.html
Or copy and paste one of these URLs:
  http://localhost:8888/?token=d6275aa18a10a0794e6019c79d465e7f7ed1b81f77f90b81
[I 17:37:40.440 NotebookApp] 302 GET / (:::1) 0.00ms
[I 17:37:43.446 NotebookApp] 302 GET / (:::1) 0.00ms
[I 18:15:52.049 NotebookApp] Creating new notebook in /Desktop/Diplomovka
[I 18:15:54.242 NotebookApp] Kernel started: c7a8a9d9-61f1-4ac7-bbb1-6be22603a1c3
[I 18:15:55.647 NotebookApp] Adapting to protocol v5.1 for kernel c7a8a9d9-61f1-4ac7-bbb1-6be22603a1c3
[I 18:17:54.171 NotebookApp] Saving file at /Desktop/Diplomovka/Untitled.ipynb
```

Obrázok 9. Terminálový výpis Jupyter Notebook

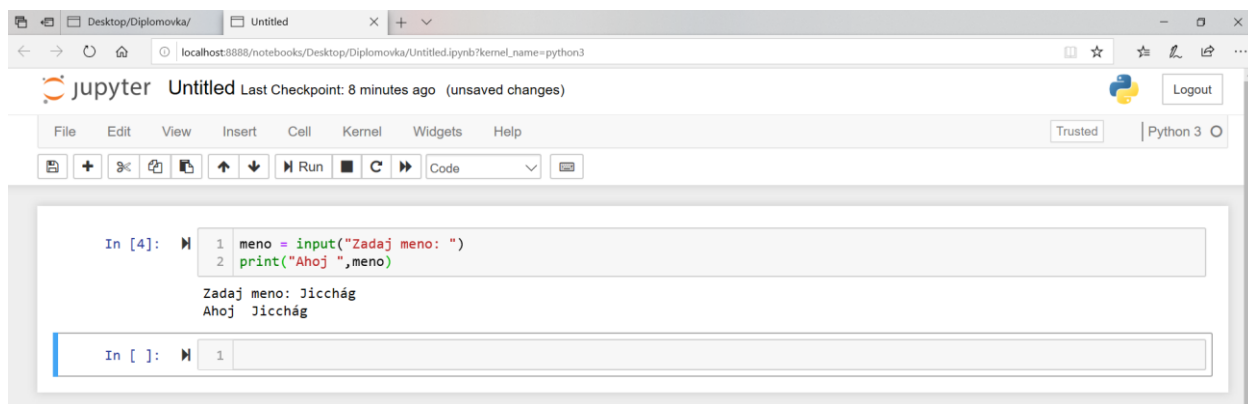
Jupyter je veľmi jednoduché prostredie, ktorého používanie je veľmi intuitívne. Keď klikneme do poľa ako na Obrázku 10, pole sa zmení na zelenú čo označuje mód úprav a užívateľ môže písať kód programu. Výhodou Jupyter Notebook je schopnosť spúšťať kód po jednotlivých poliach alebo inak povedané bunkách. Bunky spúšťame tlačidlom Run alebo

klávesovou skratkou SHIFT+ENTER. Ako príklad uvidíme dva riadky kódu, kde do premennej meno načítame reťazec a na vstupe ho vypíšeme. Po spustení bunky sa začne vykonávať kód čo značí hviezdička vedľa bunky.



Obrázok 10. Bunka Jupyter Notebook

Vidíme, že bunka čaká na používateľský vstup a po zadaní reťazca do premennej meno, prostredie vypíše zadaný reťazec, čo môžeme vidieť na Obrázku 11.



Obrázok 11. Výpis reťazca v Jupyter Notebook

Polia vieme spúšťať, pridávať, mazať a editovať. Táto vlastnosť robí podľa nášho názoru projekt Jupyter najlepším nástrojom na učenie sa programovacieho jazyka Python a konkrétne pri data science sa s ním dnes stretne každý developer. Keby sa nám stalo, že náš program z nejakého dôvodu prestane fungovať, môžeme reštartovať kernel. V konzole sa nám vypíše hláška Kernel restarted a vykonávanie programu sa zastaví. Po reštarte kernelu sa z pamäte vymaže hodnota každej premennej.

3.4.1. Moduly

V nasledujúcich kapitolách si ukážeme, ako vytvoriť klasifikátor, ktorý bude schopný predikovať výsledky zápasov. Prejdeme pojmy ako nahranie datasetu, overenie správnosti dát, vizualizácia distribúcie dát, štandardizovanie dát, trénovanie a testovanie modelu.

Ak chceme vytvoriť model pomocou knižnice scikit-learn, budeme potrebovať nasledujúce moduly:

- Numpy - Knižnica používaná na prácu s poliami. Má tiež funkcie pre prácu v doméne lineárnej algebry, Fourierovej transformácie a matíc.
- Pandas - Open-source nástroj na manipuláciu a analýzu dát. Často sa používa spoločne s knižnicou Numpy, pretože zdieľa mnoho jej funkcií.
- Matplotlib - Komplexná knižnica na vytváranie statických, animovaných a interaktívnych vizualizácií v Pythone.
- Sklearn - Knižnica Scikit-learn v Pythone, ktorá sa používa na machine learning. Poskytuje veľa učiacich sa algoritmov, slúži na prediktívnu analýzu údajov. Je postavená na niektorých technológiách, ako sú NumPy, Pandas a Matplotlib.

Moduly jednoducho importujeme príkazom import ako vidíme na Obrázku 12.

```
1 import pandas as pd
2 import numpy as np
```

Obrázok 12. Import knižníc v Pythone

3.4.2. Nahrание datasetu a jeho vlastnosti

Na čítanie a uloženie dát z datasetu do pamäte použijeme knižnicu Pandas, ku ktorej metódam budeme pristupovať skratkou pd. Na prečítanie dát z CSV súboru použijeme metódu pd.read_csv(). Dáta vložíme do vstupnej premennej, ktorú nazveme game_data. Premenná game_data je objekt typu Pandas DataFrame. Či sa dáta správne uložili do objektu DataFrame, môžeme overiť príkazom game_data.head(), ktorý vypíše prvých pár riadkov s dátami. Výstup vidíme na Obrázku 13.

```
In [2]: 1 game_data = pd.read_csv('dataset_final.csv')
        2 game_data.head()
```

Out[2]:

	Div	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	...	HST
0	E0	Fulham	Arsenal	0	3	A	0	1	A	C Kavanagh	...	2
1	E0	Crystal Palace	Southampton	1	0	H	1	0	H	Jj Moss	...	3
2	E0	Liverpool	Leeds	4	3	H	3	2	H	M Oliver	...	6
3	E0	West Ham	Newcastle	0	2	A	0	0	D	S Attwell	...	3
4	E0	West Brom	Leicester	0	3	A	0	0	D	A Taylor	...	1

5 rows × 22 columns

Obrázok 13. Nahrание datasetu futbalových zápasov do Pandas Data Frame

Pre preskúmanie správnosti dát a lepšie pochopenie datasetu sme ďalej použili funkcie:

- shape, ktorá vracia dvojicu počet záznamov a počet atribútov v datasete (počet riadkov a počet stĺpcov),
- size (vracia integer s počtom prvkov v Pandas Data Frame objekte),
- count(), ktorá vracia pole dvojíc atribút a počet záznamov. Pomocou tejto funkcie overujeme, či nám nechýba nejaký zo záznamov.

```
In [5]: 1 game_data.count()

Out[5]: Div      7025
        HomeTeam  7025
        AwayTeam  7025
        FTHG      7025
        FTAG      7025
        FTR       7025
        HTHG      7025
        HTAG      7025
        HTR       7025
        Referee   7025
        HS        7025
        AS        7025
        HST       7025
        AST       7025
        HF        7025
        AF        7025
        HC        7025
        AC        7025
        HY        7025
        AY        7025
        HR        7025
        AR        7025
        dtype: int64
```

Obrázok 14. Výpis funkcie count()

Zadaním našej úlohy je vytvoriť model, ktorý bude schopný predpovedať výsledky futbalových zápasov. V našom prípade sa jedná o hodnoty atribútu FTR (vysvetlenie v Tabuľke 3). Pre predstavu o tom, aké hodnoty nadobúda FTR, sme použili príkaz `game_data['FTR'].value_counts()`.

```
In [6]: 1 game_data['FTR'].value_counts()

Out[6]: H      3251
        A      2019
        D      1755
        Name: FTR, dtype: int64
```

Obrázok 15. Výpis hodnôt atribútu FTR a počet ich záznamov v datasete

Na výstupe v Obrázku 15 vidíme, že atribút FTR vyjadrujúci výsledok zápasu, nadobúda hodnoty H (výhra domáci), A (výhra hostia) a D (remíza). Na výstupe vidíme aj zastúpenie hodnôt FTR v našom datasete, ktoré ukazuje, že najviac zápasov končí výhrou domáceho mužstva.

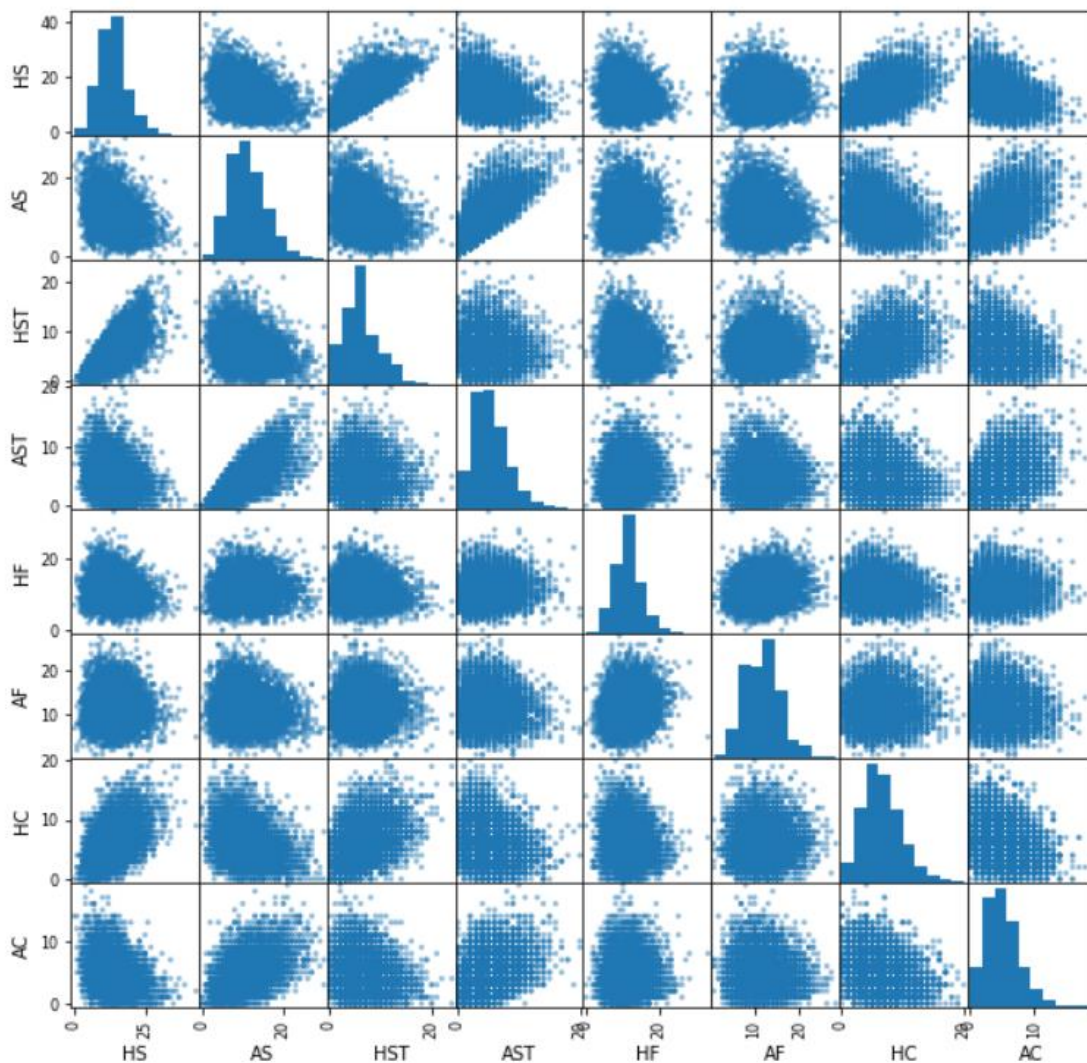
3.4.3. Vizualizácia datasetu

Medzi úlohami v oblasti dátovej vedy je bežné pochopiť vzťah medzi dvoma premennými. Na pochopenie vzťahu medzi dvoma premennými najčastejšie používame koreláciu. Existuje však scatter matrix (súbor bodových grafov) a kovariancia. Keď kovarianciu nemožno vypočítať alebo je nákladné ju vypočítať, scatter matrix môže slúžiť ako odhad kovariančnej matice. Ak existuje k atribútov, scatter matrix bude mať k riadkov a k stĺpcov.

Scatter matrix sa skladá sa z niekoľkých párových bodových grafov atribútov prezentovaných vo formáte matice. Môže sa použiť na určenie, či premenné korelujú a či je korelácia pozitívna alebo negatívna. Knižnica Pandas obsahuje funkciu, ktorá nám dokáže jednoducho vizualizovať distribúciu zadaných dát. Zadáme príkaz:

- `scatter_matrix(game_data[['HS', 'AS', 'HST', 'AST', 'HF', 'AF', 'HC', 'AC']],
figsize=(20,20)).`

Funkcia `scatter_matrix` má parametre názov objektu Data Frame s poľom atribútov, ktorých dvojice chceme vykresliť a `figsize`, ktorý v jednoduchosti vyjadruje mierku vykreslenej matice. Výstupom predošlého príkazu sa vykreslí matica rozptylu pre atribúty HS, AS, HST, AST, HF, AF, HC, AC. Výstup na Obrázku 16 nižšie nám môže opticky pomôcť napríklad vybrať relevantné atribúty pri vytváraní prediktívneho modelu. Na vizualizácii vidíme, že niektoré dvojice vlastností majú kladnú koreláciu, niektoré zápornú, čo nám môže slúžiť ako indikátor ako medzi sebou vlastnosti súvisia a ktoré sú najlepšie pre náš model.



Obrázok 16. Scatter matrix pre atribúty datasetu futbalových zápasov

3.4.4. Preprocessing dát

V momente, keď sme už dáta preskúmali a preštudovali, musíme ich pripraviť na spracovanie. Dáta musíme dostať to takej formy, aby sme mohli využiť knižnicu scikit-learn a pomocou nej mohli prediktívny model natrénovať a otestovať. Musíme si určiť cieľovú premennú, jeden atribút, ktorý chceme predikovať, v našom prípade FTR.

V ďalšom kroku vyberieme množinu atribútov, pomocou ktorých chceme predikovať FTR. Preto rozdelíme Data Frame na vlastnosť FTR a na všetky ostatné. Do premennej `X_all` uložíme množinu vlastností, ktorou chceme predikovať FTR tak, že z pôvodného objektu Data Frame vymažeme stĺpce s atribútmi, ktoré nechceme použiť na predikciu. Použijeme na to príkaz `X_all = game_data.drop(['FTR'], 1)`. Po vykonaní funkcie na Obrázku 17 vidíme, čo sa uloží do novej premennej `X_all`.

```
In [37]: 1 X_all.head()
```

Out[37]:

	FTHG	FTAG	HTHG	HTAG	HTR	HS	AS	HST	AST	HF	AF	HC	AC	HY	AY	HR	AR
0	0	3	0	1	A	5	13	2	6	12	12	2	3	2	2	0	0
1	1	0	1	0	H	5	9	3	5	14	11	7	3	2	1	0	0
2	4	3	3	2	H	22	6	6	3	9	6	9	0	1	0	0	0
3	0	2	0	0	D	15	15	3	2	13	7	8	7	2	2	0	0
4	0	3	0	0	D	7	13	1	7	12	9	2	5	1	1	0	0

Obrázok 17. Data Frame `X_all`, ktorý už neobsahuje FTR

Našu cieľovú premennú, ktorú chceme predikovať, uložíme do premennej `y_all` príkazom `y_all = game_data['FTR']`. V premennej `y_all` sa nachádza iba jeden stĺpec s hodnotami FTR, ako je ukázané na Obrázku 18.

```
In [48]: 1 y_all.head()
```

Out[48]:

```
0    A
1    H
2    H
3    A
4    A
Name: FTR, dtype: object
```

Obrázok 18. Data Frame `y_all` s jediným atribútom FTR

3.4.5. Štandardizácia dát

Keď máme dataset rozdelený na FTR a množinu ostatných vlastností, dáta môžeme a nemusíme štandardizovať. Štandardizácia dát znamená vyčistenie dátovej sady z rôznych zdrojov alebo formátov do jedného štandardného formátu s použitím jednotných pojmov a formátov [22]. To znamená, že číselné hodnoty v datasete transformujeme na rovnakú škálu a ostatné hodnoty zmeníme na dátový typ integer. Robíme to preto, lebo nie je dobré, keď jeden atribút nadobúda hodnoty v stovkách tisíc a druhý hodnoty od 1 do 10. Ak jedna vlastnosť nadobúda napríklad hodnoty (-1,1) chceme aby aj hodnoty ostatných vlastností boli v tomto rozpätí. Tento krok zlepšuje schnopnosť predikcie modelu. V našom prípade použijeme knižnicu sklearn a použijeme jej funkciu `scale(X_all)`, čím transformujeme hodnoty na rovnakú škálu.

Out[59]:

	FTHG	FTAG	HTHG	HTAG	HTR	HS	AS	HST	AST
0	-1.176674	1.622540	-0.821022	0.693658	A	-1.629135	0.495096	-1.250289	0.402499
1	-0.406298	-1.005825	0.380077	-0.697817	H	-1.629135	-0.373873	-0.955360	0.052770
2	1.904830	1.622540	2.782275	2.085133	H	1.587893	-1.025600	-0.070573	-0.646686
3	-1.176674	0.746418	-0.821022	-0.697817	D	0.263235	0.929580	-0.955360	-0.996415
4	-1.176674	1.622540	-0.821022	-0.697817	D	-1.250661	0.495096	-1.545218	0.752227

Obrázok 19. Výsledok transformácie `X_all` funkciou `scale()`

Keď sa pozrieme na náš dataset, vidíme, že niektoré vlastnosti nadobúdajú iné hodnoty ako čísla. Atribút HTR nadobúda hodnoty A, D, H. AwayTeam, HomeTeam a Referee obsahujú reťazce znakov. Aké hodnoty tieto atribúty nadobúdajú, je ukázané na Obrázku 20.

	Div	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	...
0	E0	Fulham	Arsenal	0	3	A	0	1	A	C Kavanagh	...
1	E0	Crystal Palace	Southampton	1	0	H	1	0	H	Jj Moss	...
2	E0	Liverpool	Leeds	4	3	H	3	2	H	M Oliver	...
3	E0	West Ham	Newcastle	0	2	A	0	0	D	S Attwell	...
4	E0	West Brom	Leicester	0	3	A	0	0	D	A Taylor	...

Obrázok 20. Atribúty v dátovom typom reťazec

Tieto reťazce znakov transformujeme na dátový typ integer pretože model chceme trénovať a testovať číselnými dátami. Počítač je zariadenie alebo stroj na realizáciu výpočtov alebo riadenie operácií vyjadriteľných číselnými alebo logickými výrazmi. Preto aj v tomto prípade sa uplatňuje princíp, že ak je to možné, transformujeme všetky údaje na číselné hodnoty ešte pred samotným vytváraním modelu. Postup transformácie je nasledovný:

- Vytvor nový Pandas DataFrame z pôvodného.
- Nájdi tie stĺpce, ktoré neobsahujú číselné hodnoty. Ak stĺpec obsahuje typ objekt namiesto typu číslo, potom transformuj tieto dáta na integer.

Týmto sa zbavíme reťazcov alebo inak povedané kategorických dát v `X_all` a ostane nám iba jedna kategorická premenná `FTR`, ktorú chceme predikovať. Ostatné atribúty transformujeme na spojité premenné. Vo výsledku to vyzerá tak, že sme nahradili vlastnosť `HTR`, ktorá nadobúda hodnoty z množiny $\{A, H, D\}$, novými stĺpcami `HTR_A`, `HTR_D`, `HTR_H`, ktoré obsahujú hodnoty 0 alebo 1. Rovnakým spôsobom transformujeme stĺpce s atribútmi `Referee`, `HomeTeam`, `AwayTeam`, vytvoríme nové stĺpce (22 pôvodne, 173 teraz). Na Obrázku 21 je zobrazený novovytvorený Data Frame, ktorý už obsahuje iba číselné hodnoty. Čo sme spravili, je viditeľné ihneď, keď porovnáme Obrázok 20 a 21 a všimneme si, čo sa udialo s atribútom `HTR`.

	FTHG	FTAG	HTHG	HTAG	HTR_A	HTR_D	HTR_H	HS	AS	HST	...	Referee_I Mason	Referee_A D Urso	Referee_A Wiley	Referee_C Foy
0	-1.176674	1.622540	-0.821022	0.693658	1	0	0	-1.629135	0.495096	-1.250289	...	0	0	0	0
1	-0.406298	-1.005825	0.380077	-0.697817	0	0	1	-1.629135	-0.373873	-0.955360	...	0	0	0	0
2	1.904830	1.622540	2.782275	2.085133	0	0	1	1.587893	-1.025600	-0.070573	...	0	0	0	0
3	-1.176674	0.746418	-0.821022	-0.697817	0	1	0	0.263235	0.929580	-0.955360	...	0	0	0	0
4	-1.176674	1.622540	-0.821022	-0.697817	0	1	0	-1.250661	0.495096	-1.545218	...	0	0	0	0

5 rows × 173 columns

Obrázok 21. Transformácia kategorických dát na číselné hodnoty

3.4.6. Rozdelenie dát na testovacie a tréningové

V momente, keď sme transformovali dáta do číselnej podoby, môžeme dataset rozdeliť na tréningovú a testovaciu sadu. Dáta rozdelíme príkazom

- `X_train, X_test, y_train, y_test = train_test_split(X_all, y_all, test_size = 0.2, random_state = 4, stratify = y_all).`

Do premenných `X_train` a `y_train` sa uložia dáta, ktorými budeme modely trénovať. `X_test` a `y_test` sú dáta, na ktorých budeme vytvorené modely testovať. Rozdelenie sme nastavili parametrom `test_size`, ktorý nadobúda hodnoty $0 < \text{test_size} < 1$. V celej diplomovej práci budeme rozdeľovanie dát robiť v pomere 80:20.

3.4.7. Trénovanie klasifikátora

Jeden z modelov, ktoré budeme tvoriť, je Support Vector Classifier (klasifikátor podporných vektorov) a druhý je Gaussian Naive Bayes Classifier (naivný bayesovský klasifikátor s Gaussovým normálnym rozdelením). Z knižnice `sklearn` importujeme `svm` a `GaussianNB`. V tomto kroku natrénujeme naše klasifikátory.

Pre Support Vector Classifier použijeme príkazy:

- `svc_classifier = svm.SVC(kernel='poly', gamma='auto', C=10)`
- `svc_classifier.fit(X_train, y_train)`

Pre Naive Bayes:

- `gnb_classifier = GaussianNB()`
- `gnb.fit(X_train, y_train)`

V prvom kroku do premennej uložíme konfiguráciu klasifikátora a v druhom kroku metódou `fit` trénujeme vytvorený model s dátami uloženými v `X_train` a `y_train`.

3.4.8. Testovanie klasifikátora

Natrénované modely chceme otestovať a zistiť, akú presnosť dosahujú na testovacích dátach. Aby bolo testovanie úspešné, testovacie dáta sú rozdielne od trénovacích. Ľahko by sa mohlo stať že testovanie ukáže, že náš model je 100%. V takomto prípade sme asi niektorý z predchádzajúcich krokov urobili zle. Testovaním sa myslí predikovať odpovede modelu na testovacích dátach. Na to nám slúži funkcia `predict`. Do premennej `y_pred` uložíme odpovede modelu.

Support Vector Machine:

- `y_predict = svc_classifier.predict(X_test)`

Naive Bayes:

- `y_predict = gnb_classifier.predict(X_test)`

Keď máme dáta klasifikované našimi modelmi, využijeme metódu `classification_report` z knižnice `sklearn`, ktorá porovná `y_predict` s `y_test` a vytvorí report s hodnotami ako sú presnosť klasifikátora pre hľadané triedy (W,A,D), F1 score, recall. Tieto parametre sú indikátory presnosti klasifikátora.

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_predict))
```

	precision	recall	f1-score	support
A	1.00	1.00	1.00	404
D	0.99	1.00	1.00	351
H	1.00	1.00	1.00	650
micro avg	1.00	1.00	1.00	1405
macro avg	1.00	1.00	1.00	1405
weighted avg	1.00	1.00	1.00	1405

```
from sklearn import metrics
metrics.confusion_matrix(y_test, y_predict, labels=['A', 'H', 'D'])
```

```
array([[402,  0,  2],
       [ 0, 650,  0],
       [ 1,  0, 350]], dtype=int64)
```

```
metrics.accuracy_score(y_test, y_predict)
```

```
0.997864768683274
```

Obrázok 22. Report pre svc_classifier

```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
```

```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Accuracy: 0.7088967971530249

```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
A	0.76	0.72	0.74	404
D	0.47	0.59	0.53	351
H	0.85	0.77	0.81	650
micro avg	0.71	0.71	0.71	1405
macro avg	0.69	0.69	0.69	1405
weighted avg	0.73	0.71	0.72	1405

```
from sklearn import metrics
metrics.confusion_matrix(y_test, y_pred, labels=['A', 'H', 'D'])
```

```
array([[291, 17, 96],
       [ 18, 498, 134],
       [ 75, 69, 207]], dtype=int64)
```

Obrázok 23 Report pre gnb_classifier

3.5. Weka a dataset futbalových zápasov

V nasledujúcich kapitolách si ukážeme, ako vytvoriť model, ktorý bude schopný klasifikovať výsledky futbalových zápasov v nástroji Weka. Nahráme dataset, overíme správnosť dát, ukážeme si vizualizácie distribúcie dát, štandardizovanie dát, tréning a testovanie modelu.

3.5.1. Attribute Relation File Format

Ako už vieme, dataset s údajmi o futbalových zápasoch anglickej Premier League máme vo formáte CSV súboru. Weka však podporuje iba formát ARFF a preto musíme vykonať niekoľko krokov a konvertovať CSV súbor do žiadaného formátu. ARFF je skratka, ktorá znamená Attribute Relation File Format. Jedná sa o rozšírenie formátu súboru CSV, kde sa používa hlavička, ktorá poskytuje metadáta o dátových typoch v stĺpcoch. Napríklad prvých pár riadkov súboru údajov vo formáte CSV je ilustrovaných na Obrázku 24. Ten istý súbor vo formáte ARFF je ukázaný na Obrázku 25.

```
Div,HomeTeam,AwayTeam,FTHG,FTAG,FTR,HTHG,HTAG,HTR,Referee,HS,AS,HST,AST,HF,AF,HC,AC,HY,AY,HR,AR  
E0,Fulham,Arsenal,0,3,A,0,1,A,C Kavanagh,5,13,2,6,12,12,2,3,2,2,0,0  
E0,Crystal Palace,Southampton,1,0,H,1,0,H,Jj Moss,5,9,3,5,14,11,7,3,2,1,0,0  
E0,Liverpool,Leeds,4,3,H,3,2,H,M Oliver,22,6,6,3,9,6,9,0,1,0,0,0  
E0,West Ham,Newcastle,0,2,A,0,0,D,S Attwell,15,15,3,2,13,7,8,7,2,2,0,0  
E0,West Brom,Leicester,0,3,A,0,0,D,A Taylor,7,13,1,7,12,9,2,5,1,1,0,0  
E0,Tottenham,Everton,0,1,A,0,0,D,M Atkinson,9,15,5,4,15,7,5,3,1,0,0,0
```

Obrázok 24. Dáta vo formáte CSV súboru

```

@relation dataset_final

@attribute Div {E0}
@attribute HomeTeam {Fulham,'Crystal Palace',Liverpool,'West Ham','West Brom',Totten
United',Arsenal,Southampton,Newcastle,Chelsea,Leicester,'Aston Villa',Wolves,Burnley
City',Bournemouth,Watford,Norwich,Huddersfield,Cardiff,Stoke,Swansea,Hull,Middlesbro
@attribute AwayTeam {Arsenal,Southampton,Leeds,Newcastle,Leicester,Everton,Chelsea,W
United','Man City','Man United','Aston
Villa',Norwich,Bournemouth,Watford,Cardiff,Huddersfield,Stoke,Swansea,Sunderland,Hul
@attribute FTHG numeric
@attribute FTAG numeric
@attribute FTR {A,H,D}
@attribute HTHG numeric
@attribute HTAG numeric
@attribute HTR {A,H,D}
@attribute Referee {'C Kavanagh','Jj Moss','M Oliver','S Attwell','A Taylor','M Atki
Bankes','S Hooper','A Madley','D England','R Jones','O Langford','T Robinson','S Sco
Foy','P Dowd','H Webb','M Halsey','P Walton','A Wiley','S Bennett','St Bennett','Mn
Knight','R Beeby','A D Urso','I Williamson','M Messias','N Barry','S Dunn','P Crossl
Dunn','J Winter','P Durkin','G Barber','R Martin','Graham Barber','D Elleray','C Wil
@attribute HS numeric
@attribute AS numeric
@attribute HST numeric
@attribute AST numeric
@attribute HF numeric
@attribute AF numeric
@attribute HC numeric
@attribute AC numeric
@attribute HY numeric
@attribute AY numeric
@attribute HR numeric
@attribute AR numeric

@data
E0,Bournemouth,'Man City',0,1,A,0,0,D,'K Friend',0,23,0,7,7,7,0,14,1,2,0,0
E0,'Crystal Palace',Brighton,1,1,D,1,0,H,'S Attwell',1,20,1,3,13,12,2,5,3,2,0,1
E0,'West Brom','Aston Villa',0,3,A,0,1,A,'M Atkinson',1,19,1,10,14,8,1,7,1,2,1,0
E0,Newcastle,Arsenal,0,1,A,0,0,D,'A Marriner',1,22,0,9,15,8,0,9,6,1,1,0
E0,Birmingham,Chelsea,1,0,H,1,0,H,'M Halsey',1,24,1,9,9,2,4,14,2,0,0,0
E0,Middlesbrough,Liverpool,0,0,D,0,0,D,'L Mason',1,17,1,10,14,3,2,12,0,0,0,0
E0,Watford,'Man City',0,4,A,0,2,A,'M Oliver',2,26,0,10,14,11,0,8,2,0,0,0
E0,Cardiff,'Man City',0,5,A,0,3,A,'M Oliver',2,21,2,10,6,4,1,9,1,1,0,0
E0,Brighton,Wolves,1,0,H,0,0,D,'A Taylor',2,25,1,7,11,8,1,10,3,0,0,0

```

Obrázok 25. Dáta vo formáte ARFF súboru

Všimnime si na Obrázku 25, že názov množiny údajov je definovaný príkazom @relation. Na definovanie názvu a dátového typu atribútu používame príkaz @attribute a príkaz @data označuje začiatok nespracovaných dát. Riadky v súbore ARFF, ktoré začínajú symbolom percenta (%), označujú komentár. Hodnoty v sekcii nespracovaných dát, ktoré majú symbol otáznika (?), označujú neznámu alebo chýbajúcu hodnotu. Formát

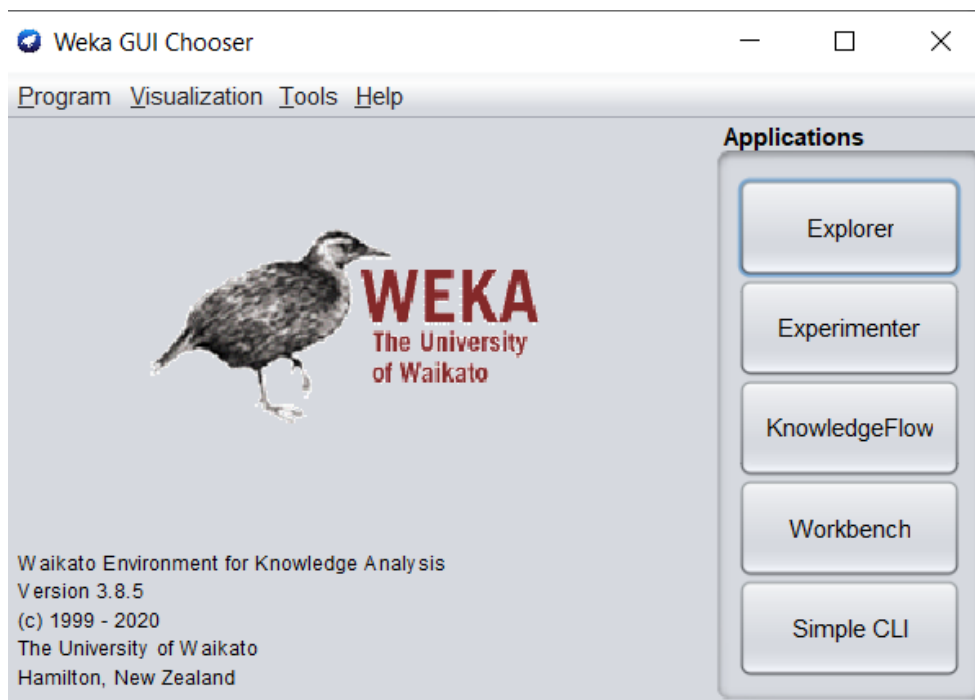
podporuje číselné a kategorické hodnoty ako v príklade vyššie a podporuje aj dátumy a hodnoty reťazcov.

3.5.2. Konverzia CSV na ARFF

Naše dáta máme vo formáte CSV, a tým pádom musíme prejsť postupnosťou krokov, aby sme ich konvertovali do formátu ARFF. CSV (Comma Separated Value) je jednoduchý formát, v ktorom sú údaje usporiadané do tabuľky riadkov a stĺpcov, a na oddelenie hodnôt v riadku sa používa čiarka. Na oddelenie hodnôt možno použiť aj úvodzovky, najmä ak údaje obsahujú reťazce textu s medzerami.

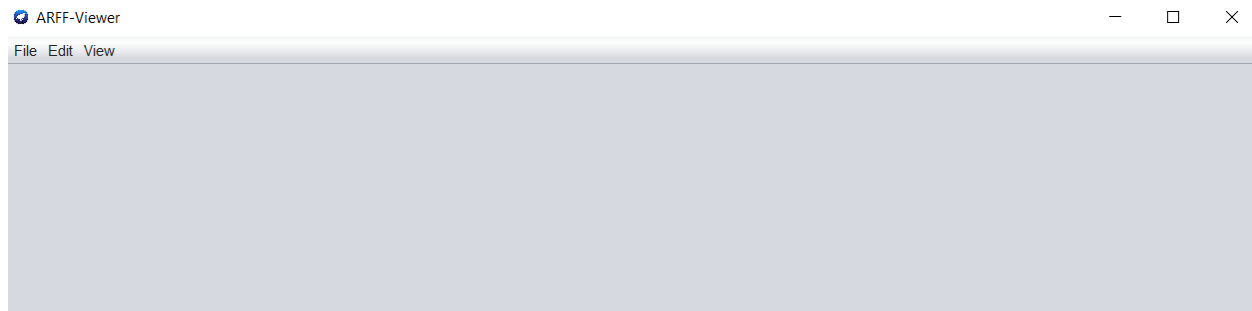
Weka poskytuje užitočný nástroj na načítanie súborov CSV a ich ukladanie do ARFF. V podstate úlohou je vytvoriť súbor v žiadanom formáte, čo spravíme raz a ďalej používame novovytvorený súbor.

- Otvoríme Weka Chooser.



Obrázok 26. Okno Weka Chooser

- Otvoríme ARFF Viewer kliknutím na Tools v ponuke a vyberieme ArffViewer. Zobrazí sa nám prázdne okno ARFF-Viewer.



Obrázok 27. Okno ARFF-Viewer

- Klikneme na File a Open. Prejdeme do svojho aktuálneho pracovného adresára a zvolíme náš CSV súbor, z ktorého chceme vytvoriť súbor ARFF.

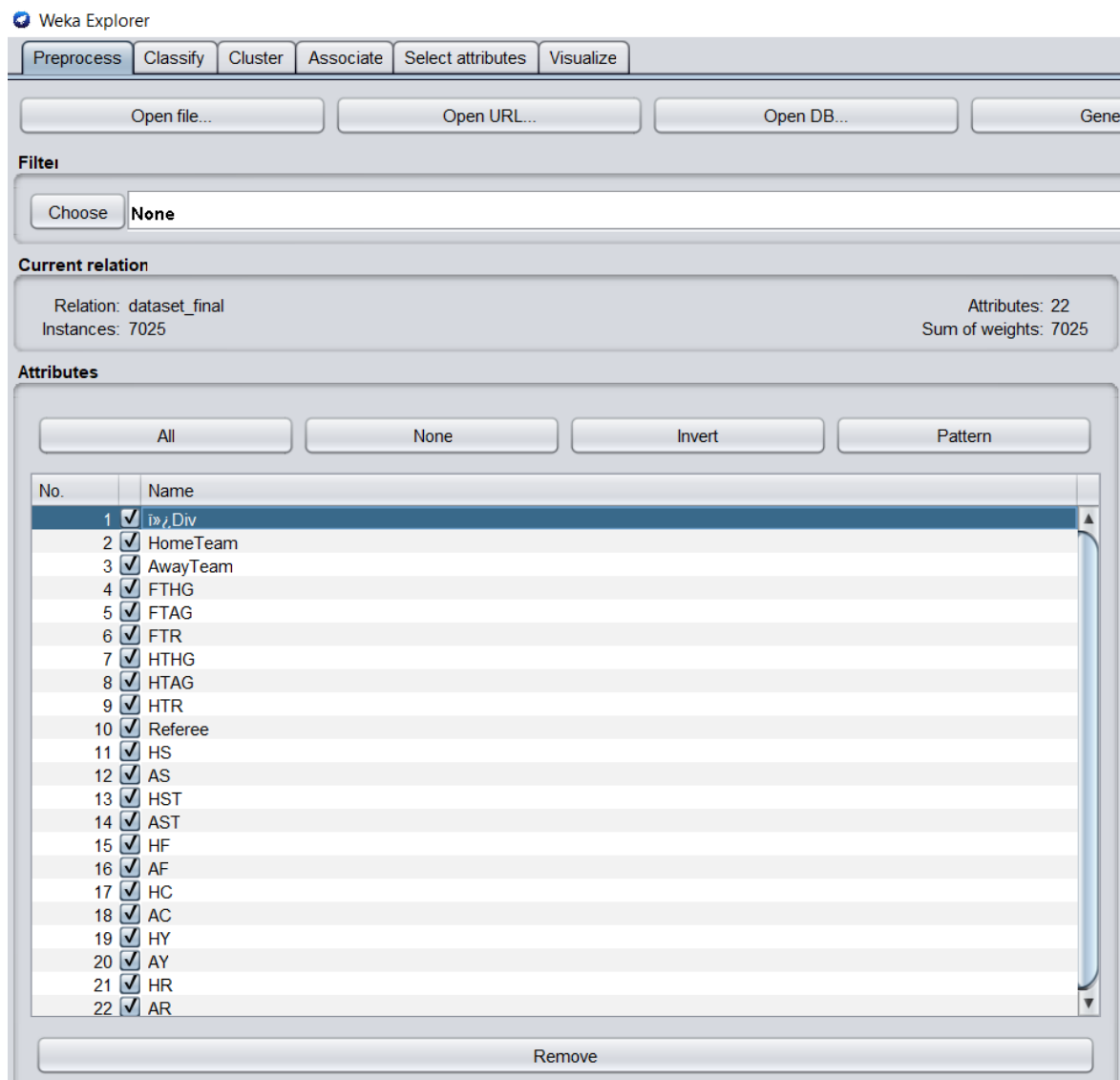
No.	1: Div	2: HomeTeam	3: AwayTeam	4: FTHG	5: FTAG	6: FTR	7: HTHG	8: HTAG	9: HTR	10: Referee	11: HS	12: AS	13: HST	14: AST	15: HF	16: AF	17: HC	18: AC	19: I
	Nominal	Nominal	Nominal	Numeric	Numeric	Nominal	Numeric	Numeric	Nominal	Nominal	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Nominal
1	E0	Fulham	Arsenal	0.0	3.0	A	0.0	1.0	A	C Kavan...	5.0	13.0	2.0	6.0	12.0	12.0	2.0	3.0	2
2	E0	Crystal Pala...	Southampt...	1.0	0.0	H	1.0	0.0	H	Jj Moss	5.0	9.0	3.0	5.0	14.0	11.0	7.0	3.0	2
3	E0	Liverpool	Leeds	4.0	3.0	H	3.0	2.0	H	M Oliver	22.0	6.0	6.0	3.0	9.0	6.0	9.0	0.0	2
4	E0	West Ham	Newcastle	0.0	2.0	A	0.0	0.0	D	S Attwell	15.0	15.0	3.0	2.0	13.0	7.0	8.0	7.0	2
5	E0	West Brom	Leicester	0.0	3.0	A	0.0	0.0	D	A Taylor	7.0	13.0	1.0	7.0	12.0	9.0	2.0	5.0	2
6	E0	Tottenham	Everton	0.0	1.0	A	0.0	0.0	D	M Atkinson	9.0	15.0	5.0	4.0	15.0	7.0	5.0	3.0	2
7	E0	Brighton	Chelsea	1.0	3.0	A	0.0	1.0	A	C Pawson	13.0	10.0	3.0	5.0	8.0	13.0	4.0	3.0	2
8	E0	Sheffield Unit...	Wolves	0.0	2.0	A	0.0	2.0	A	M Dean	9.0	11.0	2.0	4.0	13.0	7.0	12.0	5.0	2

Obrázok 28. CSV súbor v ARFF-Viewer

- Vidíme CSV súbor nahratý v ARFF-Viewer. Takto nahratý súbor uložíme kliknutím na File a Save as. Zadáme názov súboru s koncovkou .arff a klikneme na tlačidlo Save.

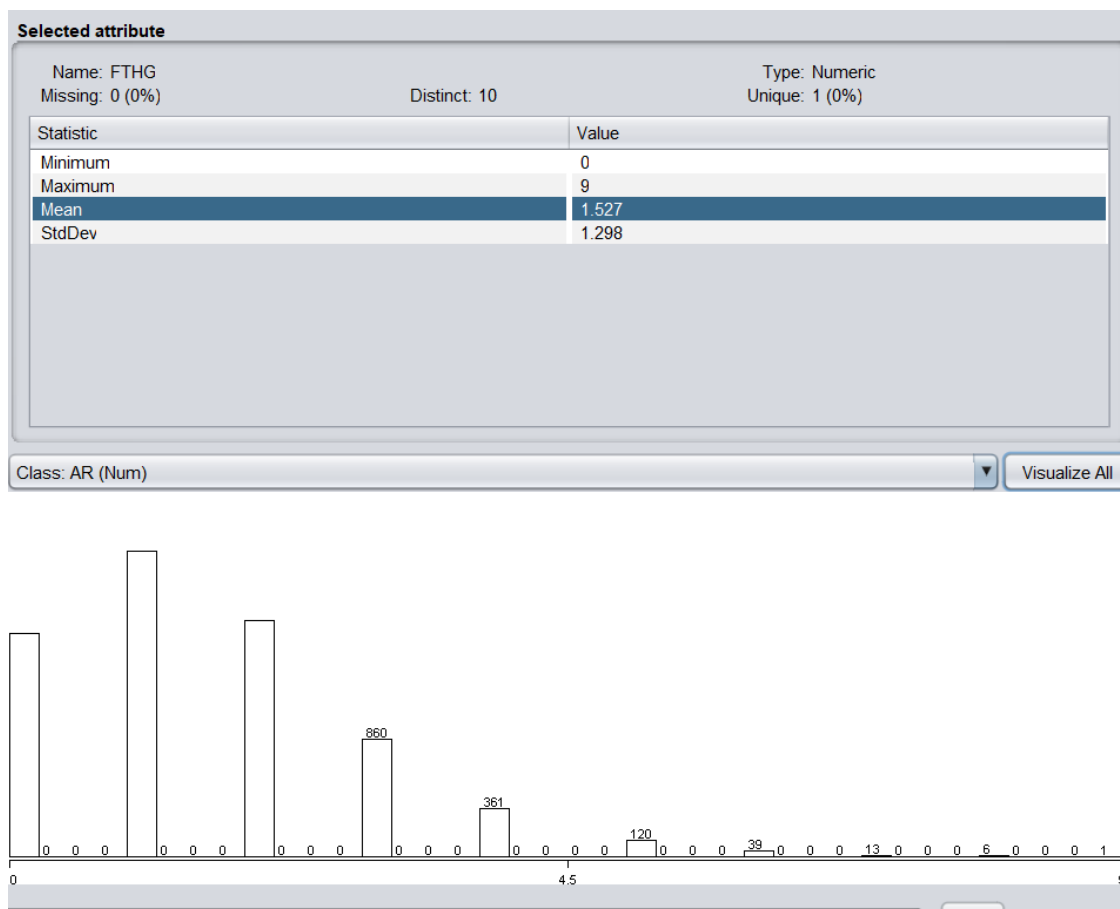
3.5.3. Nahratie a vizualizácia datasetu vo Weka

Úspešne sme vytvorili súbor vo formáte ARFF a dáta môžeme nahráť do prostredia Weka. Vo Weka Chooser klikneme na tlačidlo Explorer a zobrazí sa nám okno ako na Obrázku 29, v ktorom môžeme začať modelovať úlohu. Weka Explorer obsahuje jednotlivé záložky alebo taby, z ktorých my použijeme Preprocess. Pomocou tlačidla Open File vyberieme súbor vo formáte ARFF. Po nahratí súboru sa nám zobrazia vlastnosti nášho datasetu. V časti Current relation vidíme napríklad počet atribútov a počet inštancií (záznamov). V časti Attributes môžeme označiť konkrétny atribút a tlačidlom Remove ho vymazať a nepočítať s ním v ďalších krokoch.



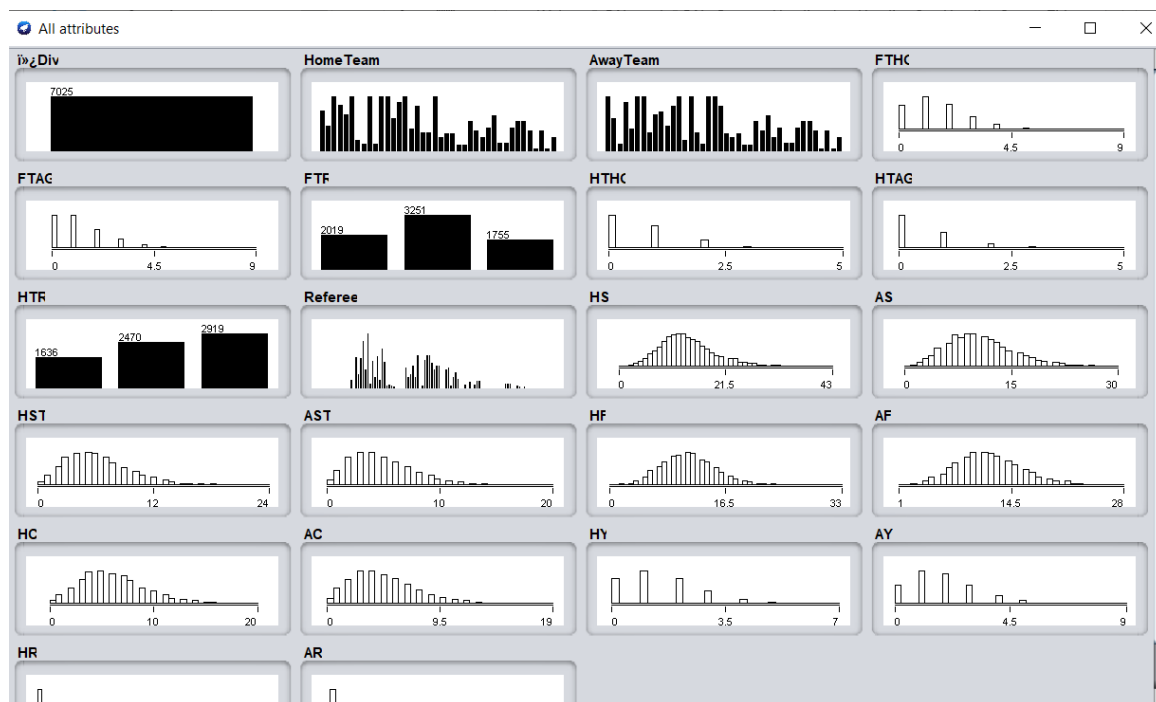
Obrázok 29. Okno Weka Explorer

Po kliknutí na konkrétny atribút sa v pravej časti okna zobrazujú informácie o zvolenom atribúte v závislosti od toho, o aký atribút sa jedná. Okrem toho sa nám v spodnej časti zobrazí aj vizualizácia distribúcie dát konkrétneho atribútu. Ak ide o atribút obsahujúci číselné hodnoty, v pravej časti okna sa vypíše maximálna hodnota, minimálna hodnota, priemer a štandardná odchýlka. Pod spomenutými informáciami o atribúte sa nachádza vizualizácia hodnôt atribútu v podobe stĺpcového grafu.



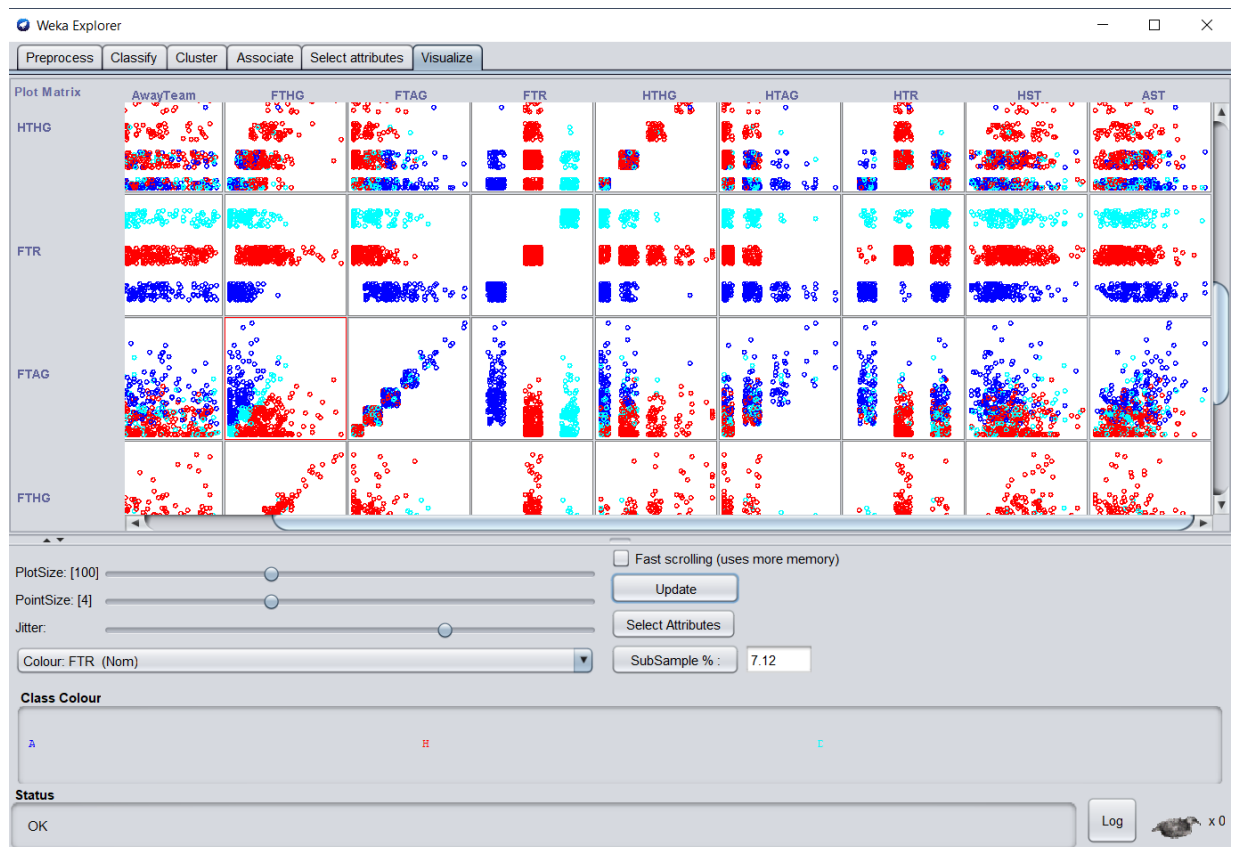
Obrázok 30. Vizualizácia dát vo Weka Explorer

Za zmienku určite stojí aj tlačidlo Visualise all, ktoré otvorí okno s vizualizáciou hodnôt všetkých atribútov v jednom celku. Výsledok vidíme na Obrázku 31.



Obrázok 31. Visualize all vo Weka Explorer

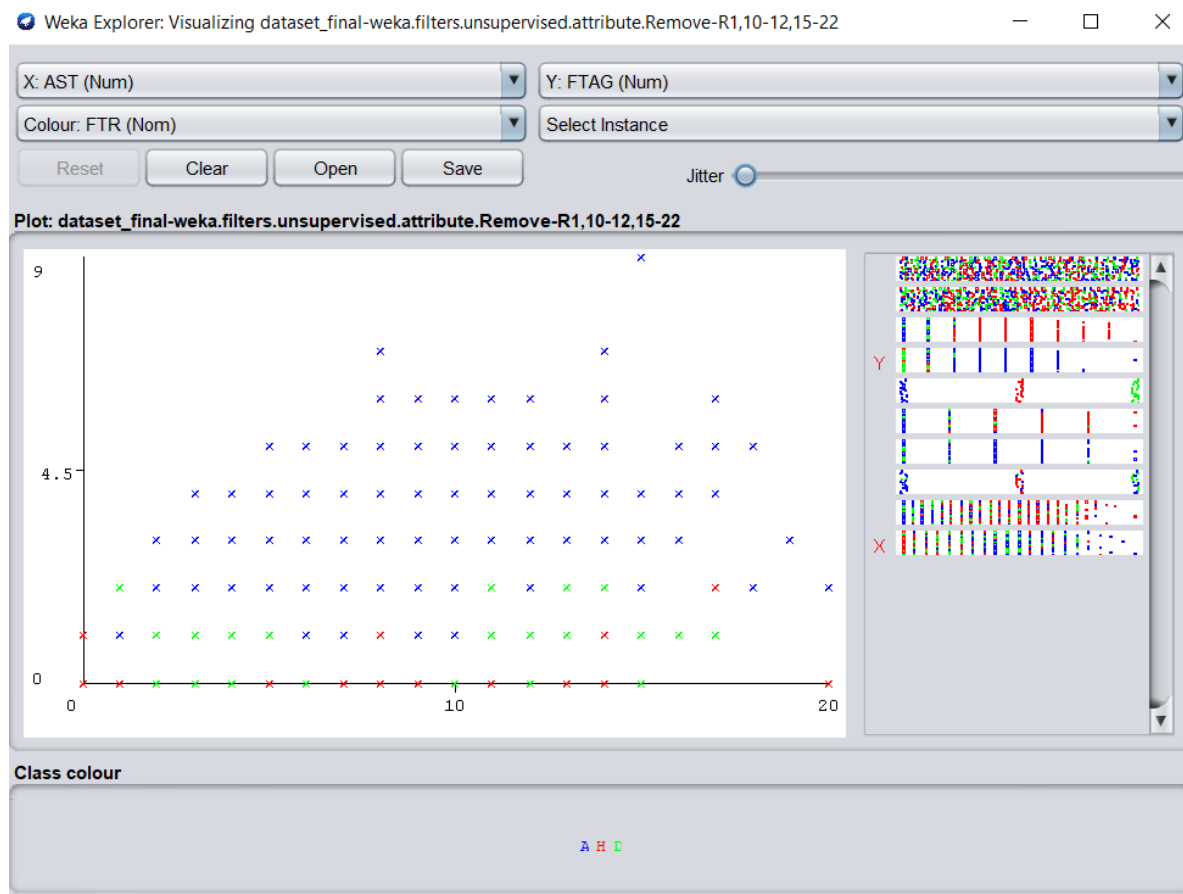
Na vizualizáciu dát ďalej môžeme použiť tab Visualise. Tab ukazuje niekoľko párových bodových grafov atribútov prezentovaných vo formáte matice. Atribúty sú vyznačené na osi x a y. V podstate tu hovoríme o scatter matrix ako v kapitolách o Jupyter a Scikit-learn.



Obrázok 32. Scattermatrix vo Weka Visualize

V rozhraní klikneme na jeden zo štvorcov s grafom pre zväčšenie. Napríklad x: AST a y: FTHG. Názvy triedy sú znázornené v rôznych farbách.

- trieda A: modrá farba
- trieda H: červená
- trieda D: zelená

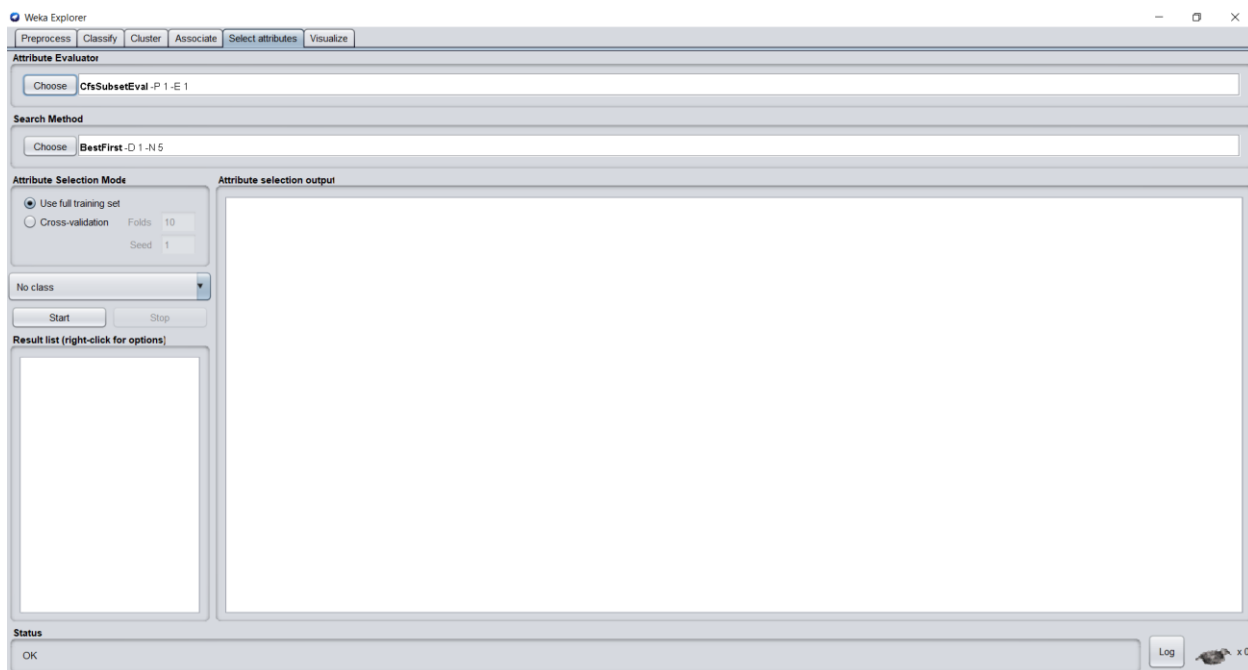


Obrázok 33. Vizualizácia dát pre x: AST a y: FTHG.

3.5.4. Selekcia atribútov vo Weka

Údaje pre strojové učenie obsahujú zmes atribútov, z ktorých niektoré sú relevantné pre tvorbu predpovedí. Ako vieme, ktoré atribúty máme použiť a ktoré odstrániť? Proces výberu atribútov v našich dátach na modelovanie nášho problému sa nazýva selekcia atribútov, po anglicky feature selection. V tejto kapitole si povieme o tom, ako vykonávať výber atribútov s našimi údajmi vo Weke. Povieme si o dôležitosti selekcie atribútov pri riešení problému so strojovým učením, a ako je podporovaný výber atribútov na platforme Weka.

Weka podporuje veľa techník selekcie atribútov. Ak máme dataset nahratý v pamäti, klikneme na sekciu Select attributes.



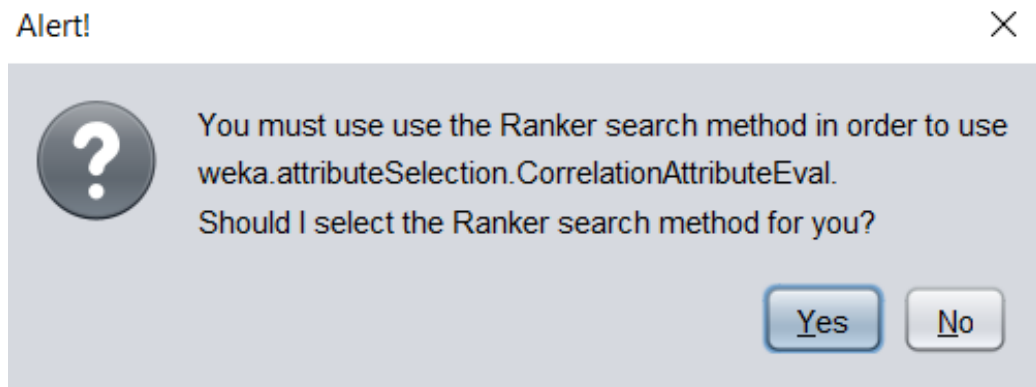
Obrázok 34. Okno Select attributes vo Weka

Selekcia atribútov je rozdelená do dvoch častí:

- Evaluácia atribútov Attribute Evaluator
- Metóda vyhľadávania Search Method

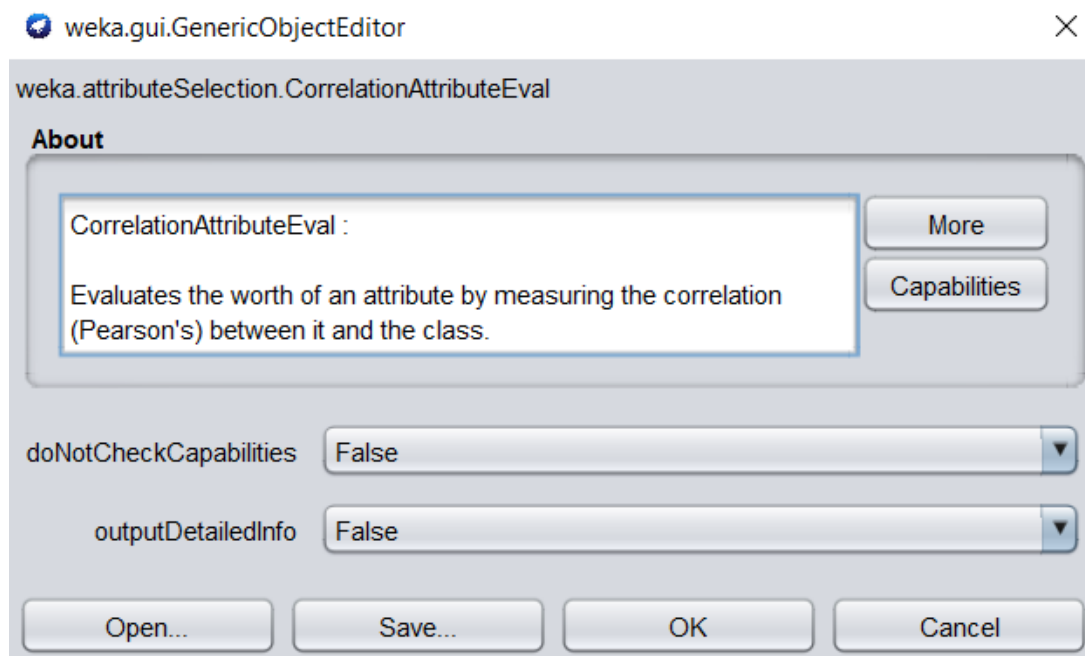
Attribute Evaluator je technika, pomocou ktorej sa každý atribút v našej množine údajov (tiež nazývaný stĺpec alebo vlastnosť) hodnotí v kontexte výstupnej premennej (napr. triedy). Metóda vyhľadávania je technika, pomocou ktorej môžeme vyskúšať alebo prechádzať rôznymi kombináciami atribútov v množine údajov, aby sme dostali krátky zoznam vybraných atribútov [23].

Niektoré techniky Attribute Evaluator vyžadujú použitie špecifických metód vyhľadávania. Napríklad techniku CorrelationAttributeEval použitú v nasledujúcej časti je možné použiť iba s metódou Ranker Search, ktorá hodnotí každý atribút a uvádza výsledky za sebou. Pri výbere rôznych Attribute Evaluator nás rozhranie môže požiadať, aby sme zmenili metódu vyhľadávania na niečo kompatibilné so zvolenou technikou [23].



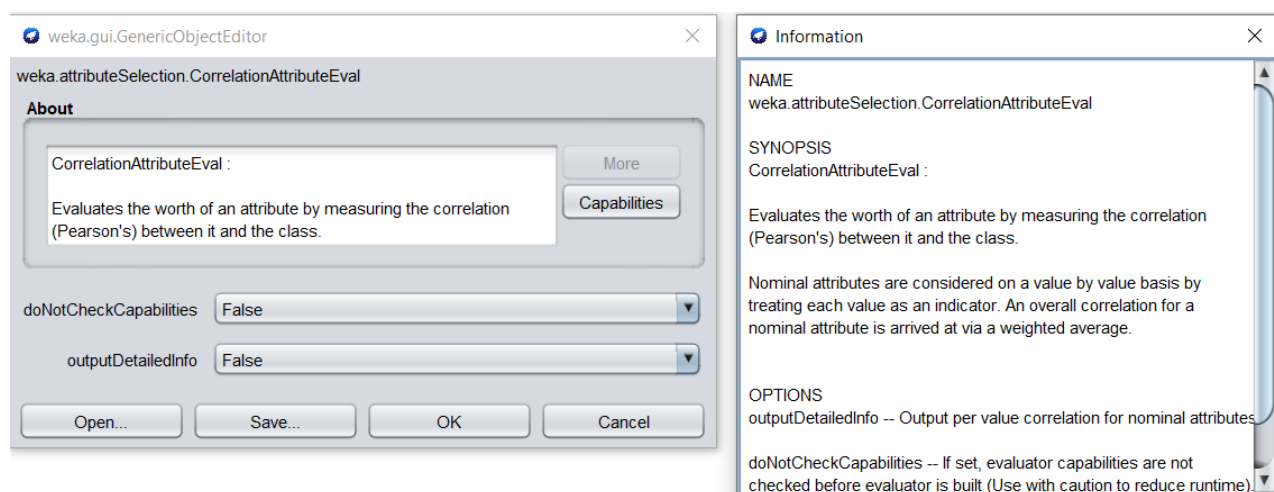
Obrázok 35. Hlásenie Alert vo Weka

Attribute Evaluator aj Search Method sa dajú nakonfigurovať. Po výbere klikneme na názov techniky, aby sme získali prístup k jej podrobnostiam o konfigurácii.



Obrázok 36. Podrobnosti o konfigurácii Attribute Evaluator

Kliknutím na tlačidlo More získame ďalšiu dokumentáciu o technike a konfiguračných parametroch. Umiestnením kurzora myši na konfiguračný parameter sa nám zobrazí popis s ďalšími podrobnosťami.



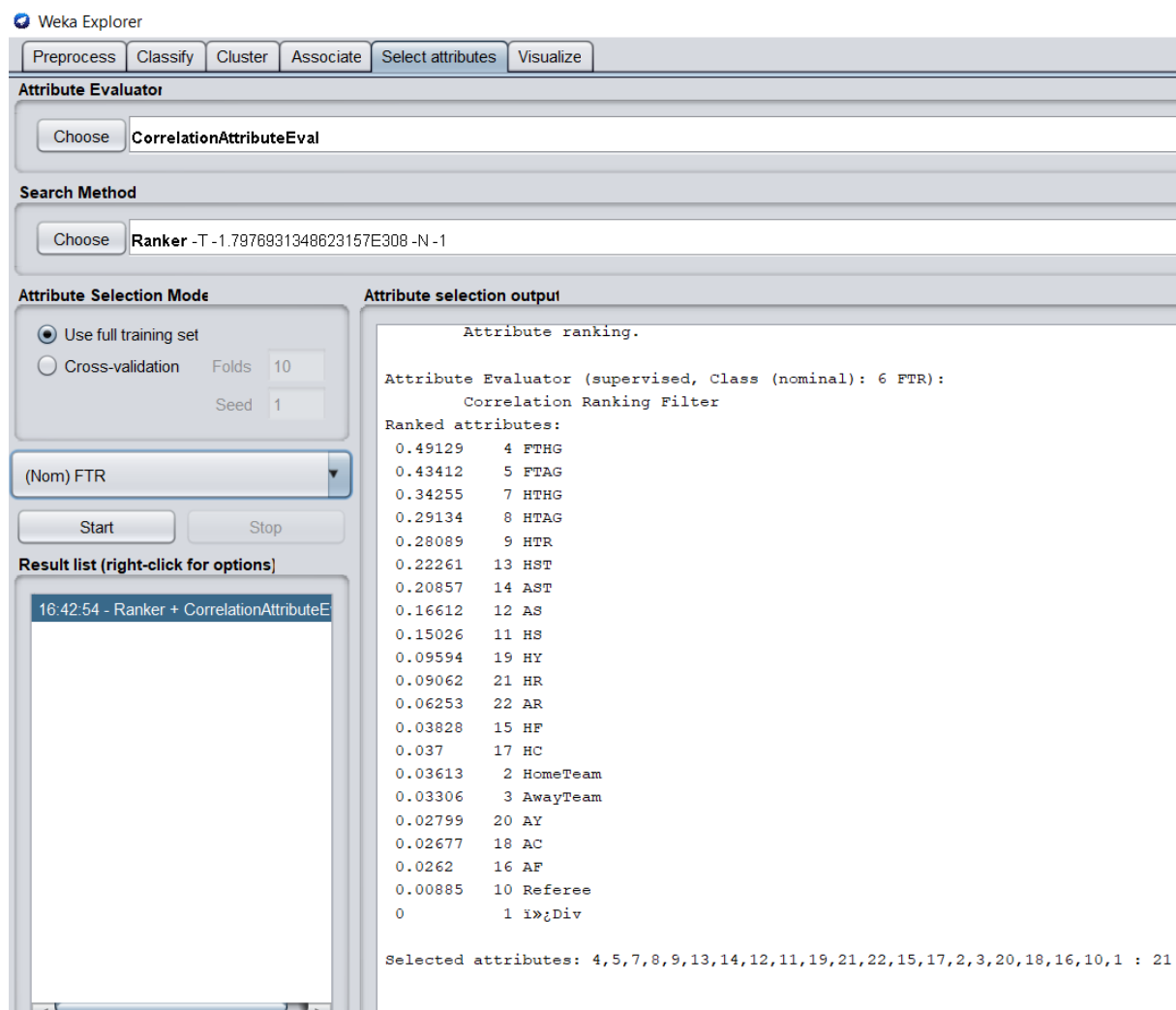
Obrázok 37. Podrobný popis o konfigurácii Attribute Evaluator

Nemôžeme vedieť, ktoré zobrazenia údajov vytvoria najpresnejšie modely. Preto je dobré vyskúšať na údajoch množstvo rôznych techník selekcie a následne vytvoriť mnoho rôznych zobrazení svojich údajov. To nám pomôže porovnať výsledky a získať predstavu o tom, ktoré zobrazenie údajov vedie k najlepšiemu výkonu. Získame tak predstavu o pohľade alebo konkrétnejšie o vlastnostiach, ktoré najlepšie vystavia problém učiacim sa algoritmom všeobecne.

3.5.5. Selekcia atribútov na základe korelácií

Populárnou technikou výberu najrelevantnejších atribútov v súbore údajov je použitie korelácie. Korelácia sa v štatistike formálne označuje ako Pearsonov korelačný koeficient. Môžeme vypočítať koreláciu medzi každým atribútom a výstupnou premennou a vybrať iba tie atribúty, ktoré majú strednú až vysokú pozitívnu alebo negatívnu koreláciu (blízku -1 alebo 1) a atribúty s nízkou koreláciou nezahrnúť do modelovania (hodnota blízka nule).

Weka podporuje výber funkcií založený na korelácii technikou CorrelationAttributeEval, ktorá si vyžaduje použitie vyhľadávacej metódy Ranker [23]. Spustenie selekcie v našom datasete futbalových zápasov naznačuje, že jeden atribút (FTHG) má najvyššiu koreláciu s výstupnou triedou FTR. Rozhranie tiež navrhuje celý rad atribútov s miernou koreláciou. Ak použijeme 0.2 ako náš limit pre príslušné atribúty, potom by sme mohli prípadne odstrániť zvyšné atribúty (AS, HS, HY, HR, AR, HF, HC, HomeTeam, AwayTeam, AY, AC, AF, Referee).

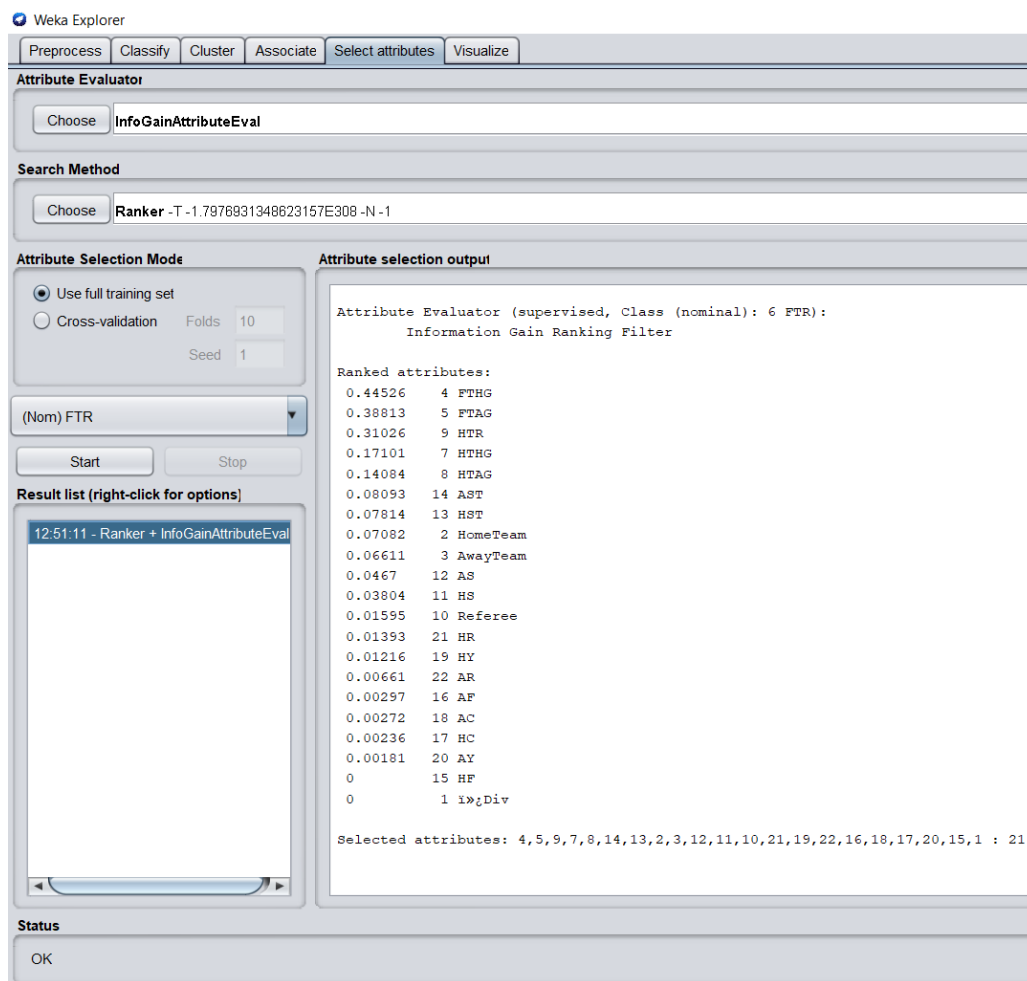


Obrázok 38. Výsledok selekcie atribútov na základe korelácií

3.5.6. Selekcia atribútov na základe informačného zisku

Ďalšou populárnou technikou selekcie atribútov je výpočet informačného zisku. Pre každý atribút výstupnej premennej môžeme vypočítať informačný zisk (tiež nazývaný entropia). Vstupné hodnoty sa pohybujú od 0 (žiadne informácie) do 1 (maximálne informácie). Tie atribúty, ktoré prispievajú viac informáciami, budú mať vyššiu hodnotu prínosu informácií a je možné ich zvoliť, zatiaľ čo tie, ktoré nepridajú veľa informácií, budú mať nižšie skóre a môžu byť odstránené [23].

Weka podporuje selekciu atribútov na základe informačného zisku pomocou nástroja InfoGainAttributeEval. Rovnako ako vyššie uvedená korelačná technika, musí byť použitá metóda Ranker Search. Pri použití tejto techniky na našom futbalovom datasete vidíme, že jeden atribút prispieva viac informáciami ako všetky ostatné (FTHG). Ak použijeme ľubovoľný medzný limit 0.05, potom by sme tiež vybrali atribúty AwayTeam, HomeTeam, HST, AST, HTAG, HTHG, HTR, FTAG a zvyšok by sme vynechali z našej množiny údajov.



Obrázok 39. Výsledok selekcie atribútov na základe informačného zisku

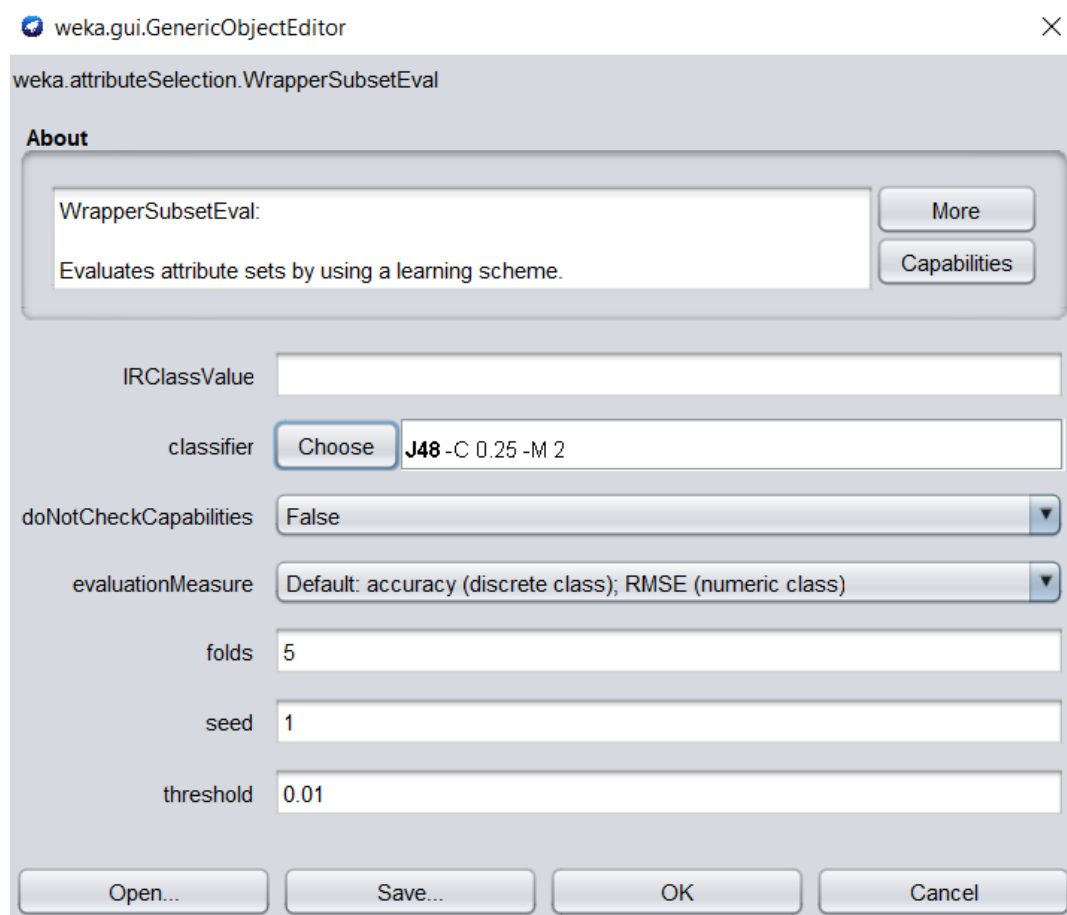
3.5.7. Selekcia atribútov na základe algoritmov strojového učenia

Populárnou technikou selekcie atribútov je použitie všeobecného, ale výkonného algoritmu učenia a vyhodnotenie výkonu algoritmu v množine údajov s rôznymi vybranými podmnožinami vybraných atribútov. Podmnožina, ktorá vedie k najlepšiemu výkonu, sa považuje za vybranú podmnožinu. Algoritmus použitý na vyhodnotenie podmnožín nemusí byť algoritmus, ktorý chceme použiť na modelovanie nášho problému, ale mal by byť

všeobecne rýchlo trénovateľný a výkonný, ako napríklad metóda rozhodovacieho stromu [23].

Vo Weke je tento typ selekcie atribútov podporovaný technikou WrapperSubsetEval a musí používať metódu vyhľadávania GreedyStepwise alebo BestFirst. Druhá možnosť, BestFirst, je preferovaná, ak nemusíme šetriť výpočtový čas.

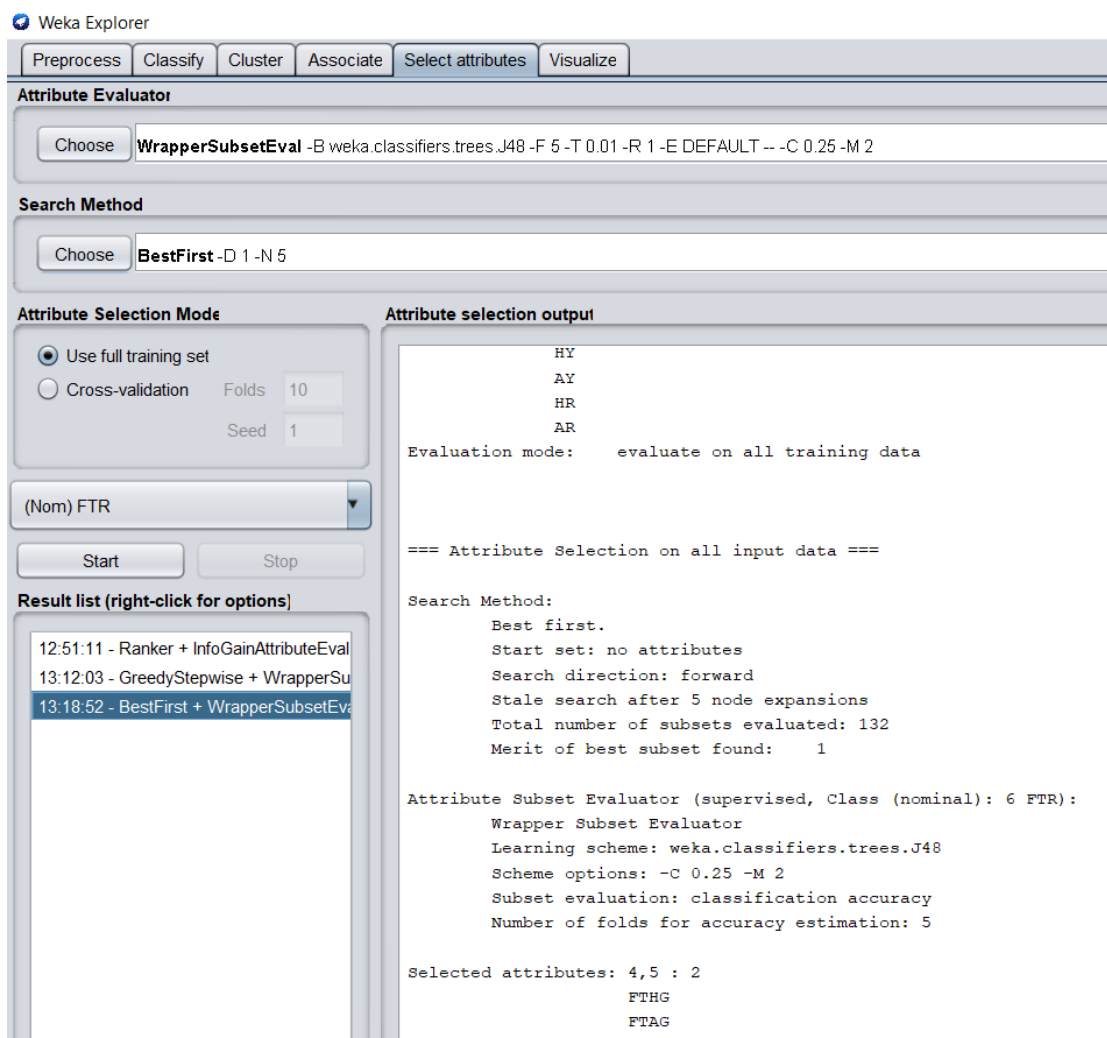
- Najskôr zvolíme techniku WrapperSubsetEval.
- Kliknutím na názov WrapperSubsetEval otvoríme konfiguráciu metódy.
- Kliknutím na tlačidlo Choose vyberieme klasifikátor.



Obrázok 40. WrapperSubsetEval vo Weka

- Klikneme na OK, čím uložíme konfiguráciu.
- Zmeníme Search Method na BestFirst.
- Klikneme na tlačidlo Start.

Spustenie tejto techniky selekcie atribútov na futbalovom datasete vybralo 2 z 22 vstupných premenných. Na Obrázku 41 vidíme výsledky tejto selekcie atribútov.



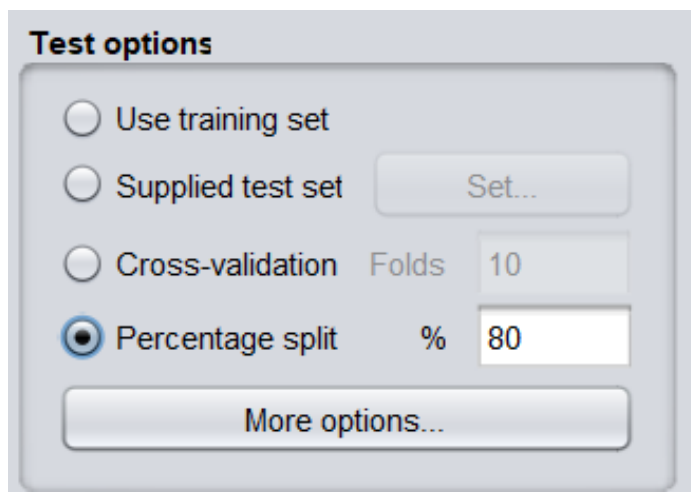
Obrázok 41. Výsledok selekcie atribútov na základe algoritmov ML

3.5.8. Rozdelenie dát na testovacie a trénovacie

V momente, keď máme dataset nahraný a predspracovaný, dáta musíme rozdeliť na testovaciu (validačnú) časť a trénovacie dáta, aby sme mohli overiť správnosť modelu. Vo Weka Explorer v časti Classify nájdeme sekciu Test Options a môžeme si zvoliť zo 4 možností:

- Use Training Set - testovanie sa uskutoční na trénovacích dátach.
- Supplied Test Set - testovanie na užívateľom zadanych dátach.
- Cross validation - vykoná sa n-násobná cross-validácia.
- Percentage split – trénovanie na užívateľom zadanom percentuálnom podiele a na zvyšku testovanie.

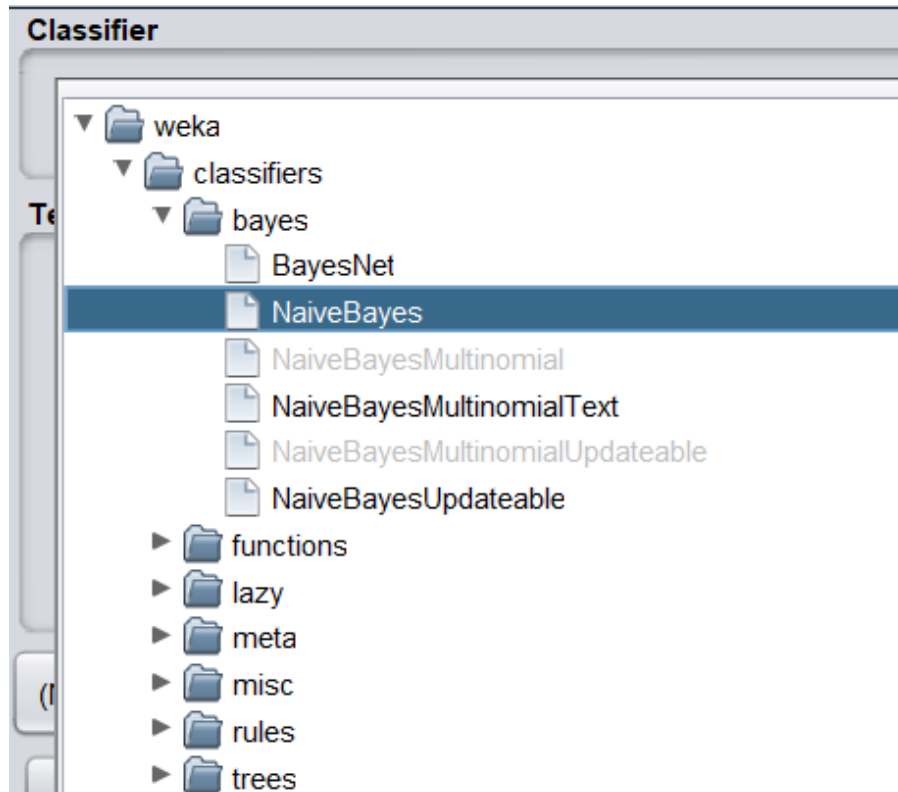
Na rozdelenie dát použijeme Percentage Split, v našom prípade už spomínané pravidlo 80:20.



Obrázok 42. Test options a rozdelenie dát pravidlom 80:20 vo Weka

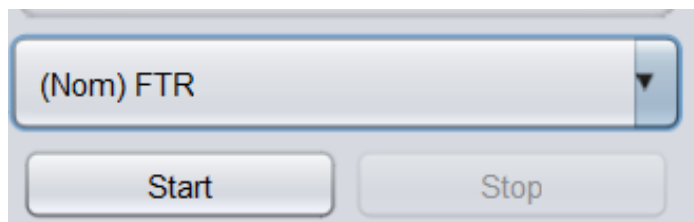
3.5.9. Trénovanie a testovanie

Trénovanie a testovanie je vo Weka spojené v jednom kroku. Stále sa nachádzame v časti Classify. Pozrieme sa do sekcie Classifier a vyberieme algoritmus. Ako prvým kandidátom bude Naive Bayes.



Obrázok 43. Výber učiaceho algoritmu vo Weka

Pod časťou Test Option sa nachádza okienko, v ktorom zvolíme atribút, ktorý chceme klasifikovať, a klikneme na tlačidlo Start. To spustí tvorbu modelu, trénovanie, testovanie a vytvorí report Classifier Output.



Obrázok 44. Štart tvorby modelu vo Weka

V časti Classifier Output vidíme výstupný model a jeho vlastnosti. Najskôr si všimnime presnosť klasifikácie. Pri použití algoritmu Naive Bayes vidíme, že model dosiahol výsledok 1037/368 správnych alebo 74%. Po druhé, pozrime sa na confusion matrix. Môžeme vidieť tabuľku skutočných tried v porovnaní s predpovedanými triedami. Došlo k 12 chybám, keď bola v zápase výhra hostujúceho tímu klasifikovaná ako výhra domáceho tímu a 93 krát ako remíza. Model pri predikcii, že vyhrá domáci tím, 7 krát klasifikoval výsledok ako výhru hostujúceho tímu 123 krát ako remízu. Pri remíze model nesprávne klasifikoval 57 krát výhru hostí a 76 krát výhru domácich. Táto tabuľka interpretuje presnosť dosiahnutú algoritmom.

```

Classifier output

Time taken to build model: 0.03 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.02 seconds

=== Summary ===

Correctly Classified Instances      1037           73.8078 %
Incorrectly Classified Instances    368           26.1922 %
Kappa statistic                    0.5978
Mean absolute error                 0.2028
Root mean squared error             0.3381
Relative absolute error             47.3253 %
Root relative squared error         72.867 %
Total Number of Instances          1405

=== Detailed Accuracy By Class ===

            TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
            0.734   0.063   0.819     0.734   0.774     0.695   0.948    0.876    A
            0.796   0.114   0.852     0.796   0.823     0.686   0.943    0.933    H
            0.644   0.210   0.527     0.644   0.580     0.410   0.818    0.595    D
Weighted Avg.   0.738   0.125   0.756     0.738   0.745     0.615   0.911    0.827

=== Confusion Matrix ===

  a  b  c  <-- classified as
290 12 93 |  a = A
  7 506 123 |  b = H
  57 76 241 |  c = D

```

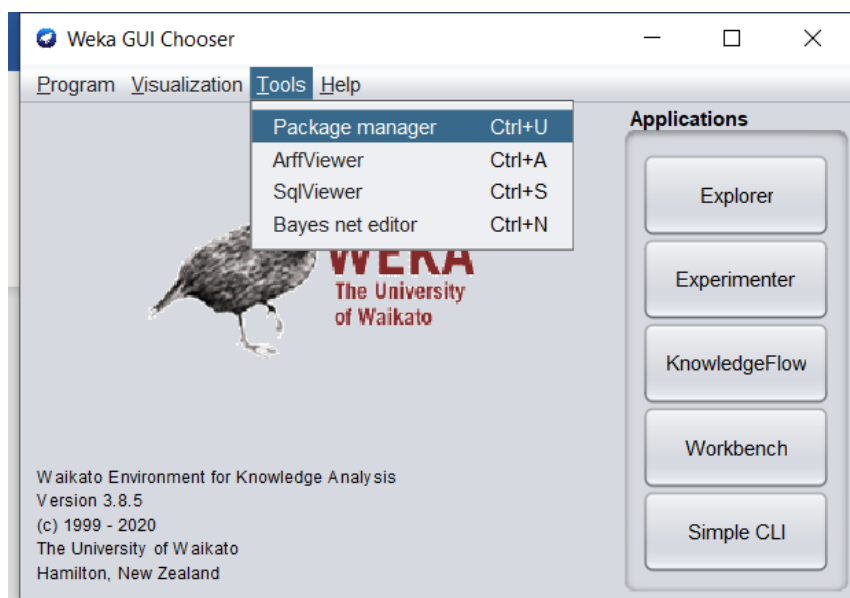
Obrázok 45. Report Classifier Output

Pre metódu podporných vektorov vo Weka máme dve možnosti. Jednou je použitie algoritmu LibSVM, druhou použitie algoritmu SMO. Povieme si o oboch, ale implementujeme prvý spomínaný, pretože je rýchlejší [24].

SMO (sequential minimal optimization) sa nachádza v základnej výbave Weka a implementuje sekvenčný algoritmus minimálnej optimalizácie Johna Platta na tréning klasifikátora podporných vektorov. Táto implementácia globálne nahrádza všetky chýbajúce hodnoty a transformuje nominálne atribúty na binárne. Predvolene tiež normalizuje všetky atribúty (v takom prípade sú koeficienty na výstupe založené na normalizovaných údajoch, nie na pôvodných údajoch - to je dôležité pre interpretáciu klasifikátora).

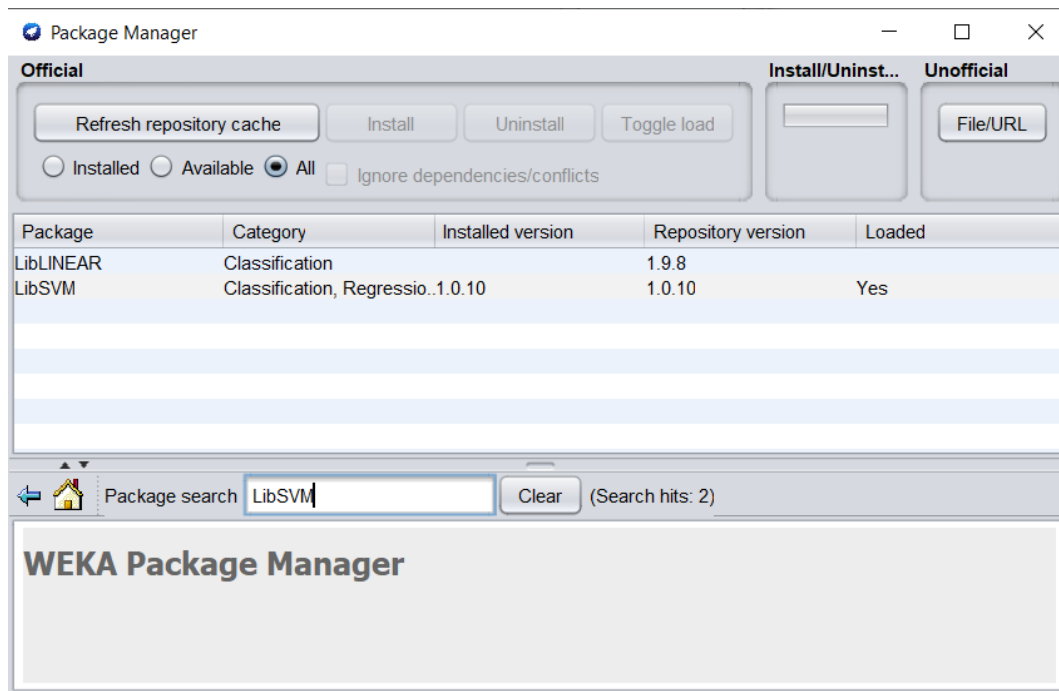
Problémy viacerých tried sa riešia pomocou párovej klasifikácie. Ak chceme získať správne odhady pravdepodobností, použijeme voľbu, ktorá prispôsobí modely logistickej regresie výstupom SVM. V prípade viacerých tried sú predpovedané pravdepodobnosti spojené pomocou metódy párovej väzby Hastie a Tibshiraniho [25].

LibSVM je wrapper pre knižnicu libsvm. Wrapper v tomto prípade zgrupuje funkcionality libsvm pre Weka. Libsvm a liblinear sú populárne open source knižnice určené na machine learning. V libsvm je implementovaný SMO pre SVM (Support Vector Machines s kernelom). Libsvm bola vytvorená Chih-Chung Changom a Chih-Jen Linom v 2011. Knižnica je napísaná v jazykoch C++ a JAVA [26]. LibSVM musíme najskôr importovať z Package Managera. V rozhraní Weka Chooser klikneme na záložku Tools a vyberieme Package Manager. Package Manager je rozhranie kde môžeme nainštalovať knižnice, ktoré nie sú v základnej výbave knižníc Weka.



Obrázok 46. Otvorenie Package Managera vo Weka

Po kliknutí na Package manager sa nám zobrazí okno Package Managera. Do Package Search zadáme názov knižnice a stlačíme ENTER. Klikneme na LibSVM a spustí sa inštalácia.



Obrázok 47. Package Manager vo Weka

Po inštalácii sa vrátíme v rozhraní do Weka Explorer a časti Classify. V sekcii Classifier zvolíme LibSVM a klikneme na Start. Po zbehnutí tréningu a testovania modelu algoritmom LibSVM s použitím polynomiálneho kernelu vyzerá Classifier Output nasledovne.

```

Classifier output

Time taken to test model on test split: 0.37 seconds

=== Summary ===

Correctly Classified Instances      1321           94.0214 %
Incorrectly Classified Instances     84            5.9786 %
Kappa statistic                     0.9082
Mean absolute error                  0.0399
Root mean squared error              0.1996
Relative absolute error              9.3019 %
Root relative squared error          43.0286 %
Total Number of Instances           1405

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0.891    0.000    1.000     0.891    0.942      0.925    0.946     0.922     A
                0.936    0.000    1.000     0.936    0.967      0.942    0.968     0.965     H
                1.000    0.081    0.817     1.000    0.899      0.866    0.959     0.817     D
Weighted Avg.   0.940    0.022    0.951     0.940    0.942      0.917    0.959     0.913

=== Confusion Matrix ===

  a  b  c  <-- classified as
352  0  43 |  a = A
  0 595  41 |  b = H
  0  0 374 |  c = D

```

Obrázok 48. Report Classifier Output pre SVM vo Weka

Pri použití algoritmu LibSVM vidíme, že model dosiahol výsledok 1321/84 správnych alebo 94%. Po druhé, pozrime sa na confusion matrix. Môžeme vidieť tabuľku aktuálnych tried v porovnaní s predpovedanými triedami a môžeme vidieť, že došlo k 43 chybám, keď bola trieda A klasifikovaná ako remíza. Model pri predikcii, že vyhrá domáci tím, 41 krát klasifikoval ako remízu. Pri remíze model nespravil žiadnu chybu.

Pre zaujímavosť sme zmenili konfiguráciu kernela z polynomial na radial basis. V takomto prípade dostaneme na rovnakom datasete 99% presnosť, kde iba 1 inštancia z 1405 bola nesprávne klasifikovaná. Confusion matrix hovorí, že chyba nastala, keď model nesprávne vyhodnotil výhru hostí v jednom prípade ako remízu.

```

=== Confusion Matrix ===

      a    b    c  <-- classified as
394    0    1 |    a = A
  0 636    0 |    b = H
  0    0 374 |    c = D

```

Obrázok 49. Confusion matrix pre kernel radial basis

3.6. H2O a dataset futbalových zápasov

H2O môžeme stiahnuť a inštalovať:

- stiahnutím súboru z oficiálnej stránky H2O,
- pomocou príkazového riadka,
- pomocou package manažéra programovacích jazykov ako Python, R.

H2O platforma je integrovaná ako:

- funkcie/metódy, ktoré importujeme priamo do kódu napríklad v Pythone,
- webové rozhranie na H2O hybridnom cloude, nazývané aj Driverless AI,
- webové rozhranie na lokálnom serveri nazývané aj Flow,
- integrácie na cloude Amazon AWS, Microsoft Azure, Databricks, IBM DSX, Nimbix Cloud, Kubernetes [27].

3.6.1. H2O klient server architektúra

Platforma H2O funguje na architektúre klient/server. Tento model rozlišuje systémy klienta od systémov servera, ktoré komunikujú cez počítačovú sieť. Ako používatelia pristupujeme ku zdrojom H2O cez klientsku aplikáciu/program, ktorá nás spája a komunikuje so serverom H2O. Medzi klientov H2O patrí:

- H2O Flow, webové používateľské rozhranie (GUI).

- R klient, knižnica H2O-R, ktorá interne používa H2O REST API na pripojenie k serveru H2O a umožňuje používateľom spustiť H2O cez prostredie programovacieho jazyka R.
- Python klient, knižnica H2O, ktorá interne používa H2O REST API na pripojenie k serveru H2O a umožňuje používateľom spustiť H2O cez prostredie programovacieho jazyka Python [28].

3.6.2. H2O Flow

H2O Flow je open source webové užívateľské rozhranie pre H2O. Jedná sa o webové interaktívne prostredie, ktoré umožňuje kombinovať vykonávanie kódu, text, matematiku, grafy a multimediálne súbory v jednom dokumente [29]. Tento proces sa tu nazýva workflow.

Pomocou H2O Flow môžeme zachytiť, znova spustiť, anotovať, prezentovať a zdieľať svoj workflow. H2O Flow nám umožňuje interaktívne používať H2O na import súborov, vytváranie modelov a ich opakované vylepšovanie v prostredí prehliadača.

Hybridné užívateľské rozhranie Flow plynulo kombinuje výpočty príkazového riadku s moderným grafickým užívateľským rozhraním. Namiesto zobrazenia výstupu ako obyčajného textu však Flow poskytuje užívateľské rozhranie typu zamier-a-klikni pre každú operáciu H2O. Umožňuje nám prístup k ľubovoľnému objektu H2O vo forme usporiadaných tabuľkových údajov.

H2O Flow odosiela príkazy H2O ako sekvenciu spustiteľných buniek. Bunky je možné upraviť, zmeniť ich usporiadanie alebo uložiť do knižnice. Každá bunka obsahuje vstupné pole, ktoré nám umožňuje zadávať príkazy, definovať funkcie, volať ďalšie funkcie a pristupovať k ďalším bunkám alebo objektom na webovej stránke. Po spustení bunky je výstupom grafický objekt, ktorý je možné skontrolovať a zobrazit ďalšie podrobnosti.

Zatiaľ čo H2O Flow podporuje REST API, R skripty a CoffeeScript, na spustenie H2O Flow nie sú potrebné žiadne skúsenosti s programovaním. Môžeme sa preklikať cez ktorúkoľvek operáciu H2O bez toho, aby sme napísali jediný riadok kódu. Môžeme dokonca zakázať vstupné bunky, aby sa tok H2O spustil iba pomocou grafického užívateľského

rozhrania. H2O Flow je navrhnutý tak, aby nás sprevádzal na každom kroku poskytovaním vstupných výziev, interaktívnej pomoci a príkladných tokov [29].

3.6.3. Inštalácia a spustenie H2O Flow

Ako prvé musíme stiahnuť H2O a na to sme chceli použiť príkaz curl v príkazovom riadku. Žiaľ, po pár neúspešných pokusoch, keď Windows Terminal vypísal chybu 503 Server Unavailable, sme sa rozhodli stiahnuť inštalačný súbor z oficiálnej stránky. Na prvý pohľad nám príde divné, že nefunguje niečo, čo autori H2O uviedli v online dokumentácii, a že ide o dosť zásadnú vec z pohľadu používateľa, ale našťastie uviedli odkaz na stránku, z ktorej sa dá ich produkt stiahnuť klasicky cez webový prehliadač.

```
PS C:\Users\JicchagSoltes> curl -o h2o.zip http://download.h2o.ai/versions/h2o-3.32.1.1.zip
curl : The remote server returned an error: (503) Server Unavailable.
At line:1 char:1
+ curl -o h2o.zip http://download.h2o.ai/versions/h2o-3.32.1.1.zip
+ ~~~~~
```

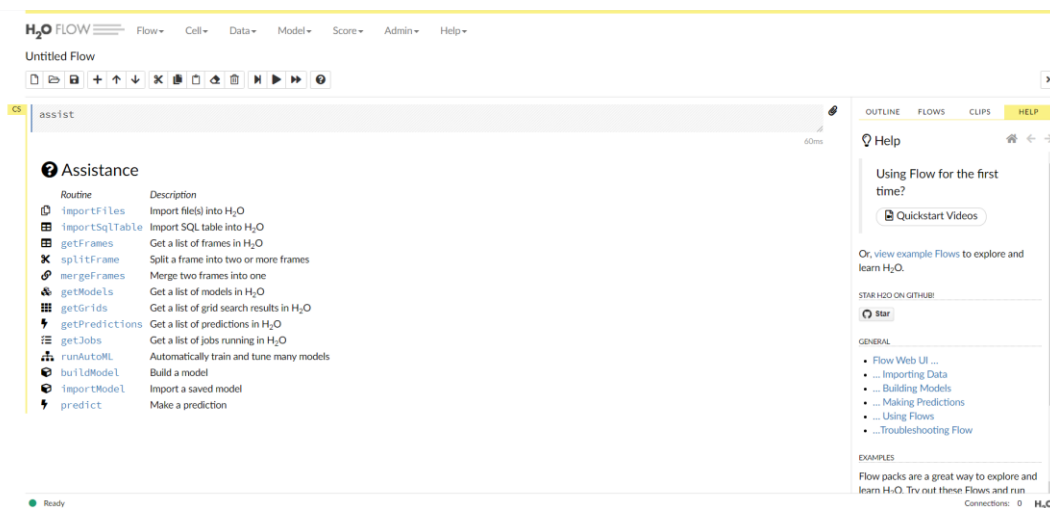
Obrázok 50. Chybové hlásenie 503 pri inštalácii H2O

Po stiahnutí inštalačného súboru vo svojom príkazovom riadku zadáme nasledujúce riadky príkazov po jednom.

- `cd ~/Downloads`
- `unzip h2o.zip` (alternatíva je `Expand-Archive h2o.zip`)
- `cd h2o`
- `java -jar h2o.jar`

Prvý riadok nás presmeruje do priečinka Stiahnuté súbory, druhý riadok rozbalí náš súbor zip, tretí riadok nás presmeruje na priečinok h2o a štvrtý riadok spustí JAVA súbor h2o.jar, ktorý spustí lokálny server.

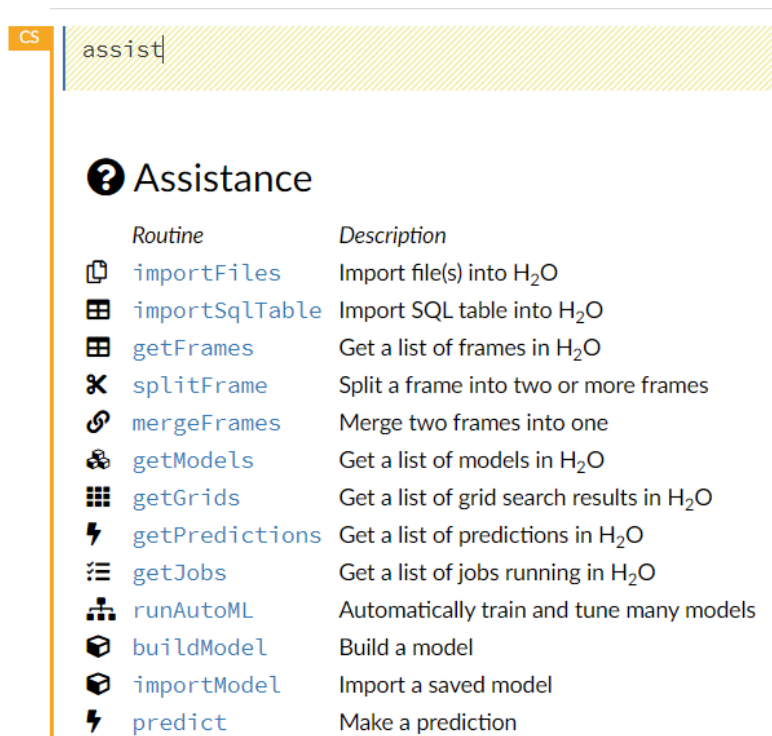
Posledným príkazom v príkazovom riadku nájdeme v prehliadači na adrese <http://localhost:54321> používateľské rozhranie H2O Flow. H2O Flow je dizajnovaný pre dátových vedcov tak aby vytváranie modelov bolo rýchle a jednoduché. Ďalej v ňom môžeme importovať súbory, pracovať so súbormi ako s Data Frame objektami a robiť veci, na ktorých by sme sa v iných prostrediach natrápili programovaním.



Obrázok 51. Prostredie H2O Flow

3.6.4. Interface H2O Flow

Inteface H2O Flow sa skladá z rôznych tlačidiel a buniek. Na prvý pohľad môžeme vidieť nejaké podobnosti s Jupyter Notebook. Avšak do buniek nepíšeme kód v Pythone, ale používame špecifickú syntax pre H2O. Do prázdnej bunky môžeme napísať príkaz `assist` a stlačiť `Ctrl + Enter`. Zobrazí sa zoznam bežných úloh, ktoré nám pomôžu nájsť správny príkaz.



Obrázok 52. H2O príkaz assist

V Jupyter sme vytvárali tzv. Notebooky a tu nazývame súbor buniek Flow. Flow môžeme uložiť, stiahnuť, kopírovať a načítať. Flow sa predvolene ukladá ako súbor vo formáte nazov.flow do adresára h2oflows pod domovským adresárom. Podporované sú iba exportované Flows s predvoleným typom súboru .flow. Ostatné typy súborov sa neotvoria.

Pre bunky existujú dva režimy a to Command a Edit. V Edit móde (režim úprav) je žltá bunka s blikajúcim pruhom, ktorý označuje, kde je možné zadať text, a naľavo od bunky je oranžová vlajka [29].

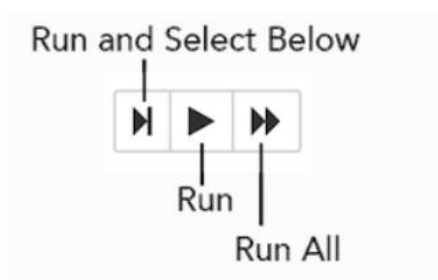


Obrázok 53. Bunka v H2O Flow

V Command režime (príkazový režim) je vľajka žltá. Vľajka tiež označuje formátovanie bunky:

- CS: Code (default) – do takejto bunky môžeme vkladať príkazy ako na Obrázku 53.
- MD: Markdown – slúži na popis alebo komentár a obsahuje text.
- RAW: Raw format – komentáre ku kódu
- H[1-6]: Headers – nadpisy
- Error: Chybové hlásenie - ak je v bunke chyba, vľajka je červená

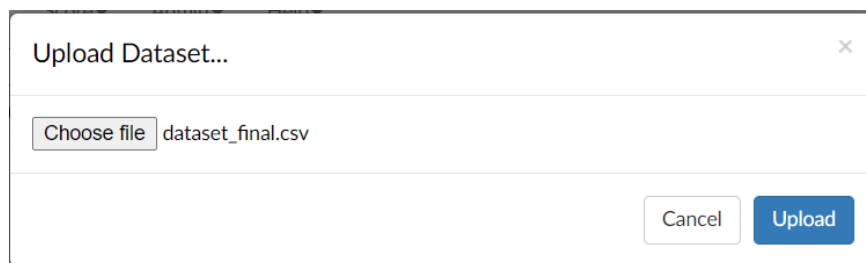
Ak chceme spustiť všetky bunky v našom workflow, klikneme na ponuku Flow a potom na možnosť Run All Cells. Ak chceme spustiť aktuálnu bunku a všetky nasledujúce bunky, klikneme na ponuku Flow a potom na možnosť Run All Cells Below. Ak chceme spustiť jednotlivú bunku, potvrdíme, že je bunka v režime úprav a potom stlačíme Ctrl+Enter. Všetky spomenuté varianty môžeme vykonať aj cez panel s tlačidlami.



Obrázok 54. H2O Flow panel s tlačidlami

3.6.5. Načítanie datasetu v H2O

Existuje niekoľko spôsobov, ako importovať údaje do H2O Flow. Ak chceme nahráť lokálny súbor, klikneme na ponuku Data a vyberíme možnosť Upload file. Klikneme na tlačidlo Choose file, vyberíme súbor, klikneme na tlačidlo Choose a potom na tlačidlo Upload.



Obrázok 55.H2O Flow a upload súboru


Ako overenie, či sme súbor nahrali, nám poslúži výpis Setup parse, v ktorom môžeme náš dataset predspracovať. Pole Sources zobrazuje cestu k súboru pre importované údaje vybrané na analýzu. ID obsahuje automaticky generovaný názov pre analyzované údaje (v predvolenom nastavení používa názov súboru importovaného súboru príponu .hex). Môžeme použiť predvolený názov alebo do tohto poľa zadať vlastný názov.

CS

```
setupParse source_frames: [ "dataset_final_h2o.csv"]
```

Setup Parse

PARSE CONFIGURATION

Sources  dataset_final_h2o.csv

ID Key_Frame__dataset_final_h2o.hex

Parser

Separator

Column Headers ☐ Auto

☒ First row contains column names

☐ First row contains data

Options ☐ Enable single quotes as a field quotation character

☒ Delete on done

Obrázok 56. H2O Flow Setup parse configuration

Nižšie, ako vidíme na Obrázku 57, môžeme editovať názvy stĺpcov (atribútov) a ich dátový typ. Ak chceme zmeniť dátový typ stĺpca, klikneme na drop-down napravo od poľa na zadanie názvu stĺpca a vyberieme typ údajov. Možnosti sú:

- Unknown,
- Numeric,
- Enum,
- Time,
- UUID,
- String,
- Invalid.

EDIT COLUMN NAMES AND TYPES

Search by column name...

1	HomeTeam	Enum	Fulham	Crystal Palace	Liverpool	West Ham	West Brom	Tottenham
2	AwayTeam	Enum	Arsenal	Southampton	Leeds	Newcastle	Leicester	Everton
3	FTHG	Numeric	0	1	4	0	0	0
4	FTAG	Numeric	3	0	3	2	3	1
5	FTR	Enum	A	H	H	A	A	A
6	HTHG	Numeric	0	1	3	0	0	0
7	HTAG	Numeric	1	0	2	0	0	0
8	HTR	Enum	A	H	H	D	D	D
9	Referee	Enum	C Kavanagh	Jj Moss	M Oliver	S Attwell	A Taylor	M Atkinson
10	HS	Numeric	5	5	22	15	7	9
11	AS	Numeric	13	9	6	15	13	15
12	HST	Numeric	2	3	6	3	1	5
13	AST	Numeric	6	5	3	2	7	4
14	HF	Numeric	12	14	9	13	12	15
15	AF	Numeric	12	11	6	7	9	7

← Previous page

→ Next page

Parse

Obrázok 57. H2O Flow Setup parse – upraviť mená a typy atribútov

3.6.6. Konverzia CSV na HEX

Pri práci s dátami v H2O Flow musíme naše dáta predspracovať a dostať do formátu, s ktorým vie H2O pracovať. Musíme vytvoriť tzv. Key Frame (niečo ako Data Frame objekt v sklearn) s koncovkou .hex. V bunke Setup Parse vyberieme typ parsera (ak je to potrebné) zo zoznamu Parser. Pri väčšine syntaktických analýz H2O automaticky rozpozná dátový typ, takže predvolené nastavenia zvyčajne nie je potrebné meniť. K dispozícii sú nasledujúce možnosti: AUTO, ARFF, XLS, XLSX, CSV a iné.

Po vykonaní potrebných zmien v nastaveniach klikneme na tlačidlo Parse, ktoré vykoná príkaz parseFiles, spracuje náš dataset a zobrazí sa výpis tejto úlohy.

CS

```

parseFiles
  source_frames: ["dataset_final_h2o.csv"]
  destination_frame: "Key_Frame__dataset_final_h2o1.hex"
  parse_type: "CSV"
  separator: 44
  number_columns: 21
  single_quotes: false
  column_names:
["HomeTeam", "AwayTeam", "FTHG", "FTAG", "FTR", "HTHG", "HTAG", "HTR", "Referee", "HS", "
  column_types:
["Enum", "Enum", "Numeric", "Numeric", "Enum", "Numeric", "Numeric", "Enum", "Enum", "Nu
eric", "Numeric", "Numeric", "Numeric", "Numeric"]
  delete_on_done: true
  check_header: 1
  chunk_size: 4194304

```

Job

Run Time 00:00:34.826

Remaining Time 00:00:00.0

Type Frame

Key  Key_Frame__dataset_final_h2o1.hex

Description Parse

Status DONE

Progress 100%

Done.

Actions  View

 Ready

Obrázok 58. H2O Flow príkaz `parseFiles`

Po dokončení `parseFiles`, klikneme na tlačidlo `View` a vidíme, čo nám vrátilo parsovanie. H2O Flow konvertoval náš súbor vo formáte CSV a vytvoril tzv. H2O Key Frame.

3.6.7. H2O Frame

H2O Frame môžeme chápať ako niečo podobné Data Frame objektu z knižnice Pandas, o ktorom sme hovorili v kapitolách vyššie. V bunke na Obrázku 59 vidíme dôležité

informácie o H2O Frame atribútoch ako label (názov), typ, kardinalita, maximálna hodnota, minimálna hodnota, stredná hodnota a iné. V stĺpci Actions môžeme konvertovať hodnoty v jednotlivých stĺpcoch.

Key_Frame__dataset_final_h2o1.hex

Actions: [View Data](#) [Split](#) [Build Model](#) [Run AutoML](#) [Predict](#) [Download](#) [Export](#)

Rows	Columns	Compressed Size
7026	21	151KB

▼ COLUMN SUMMARIES

label	type	Missing	Zeros	+Inf	-Inf	min	max	mean	sigma	cardinality	Actions
HomeTeam	enum	0	352	0	0	0	40.0	·	·	41	Convert to numeric
AwayTeam	enum	0	351	0	0	0	40.0	·	·	41	Convert to numeric
FTHG	int	1	1641	0	0	0	9.0	1.5274	1.2982	·	Convert to enum
FTAG	int	1	2403	0	0	0	9.0	1.1480	1.1415	·	Convert to enum
FTR	enum	0	2019	0	0	0	3.0	·	·	4	Convert to numeric
HTHG	int	1	3575	0	0	0	5.0	0.6836	0.8326	·	Convert to enum
HTAG	int	1	4280	0	0	0	5.0	0.5015	0.7187	·	Convert to enum
HTR	enum	0	1636	0	0	0	3.0	·	·	4	Convert to numeric
Referee	enum	0	56	0	0	0	74.0	·	·	75	Convert to numeric
HS	int	1	1	0	0	0	43.0	13.6090	5.2848	·	Convert to enum
AS	int	1	5	0	0	0	30.0	10.7210	4.6035	·	Convert to enum
HST	int	1	68	0	0	0	24.0	6.2393	3.3909	·	Convert to enum
AST	int	1	161	0	0	0	20.0	4.8491	2.8596	·	Convert to enum
HF	int	1	2	0	0	0	33.0	11.2750	3.7068	·	Convert to enum
AF	int	1	0	0	0	1.0	28.0	11.7748	3.8538	·	Convert to enum
HC	int	1	75	0	0	0	20.0	6.0902	3.1068	·	Convert to enum
AC	int	1	163	0	0	0	19.0	4.7997	2.7352	·	Convert to enum
HY	int	1	1749	0	0	0	7.0	1.4174	1.1849	·	Convert to enum
AY	int	1	1155	0	0	0	9.0	1.7495	1.2697	·	Convert to enum
HR	int	1	6610	0	0	0	3.0	0.0618	0.2523	·	Convert to enum

← Previous 20 Columns → Next 20 Columns

● Ready

Obrázok 59.H2O Flow Key Frame

Pri vytváraní modelu nechceme používať reťazce znakov, pretože počítače najlepšie pracujú s číslami, a tak po jednom klikneme na Convert to numeric. Pre ukážku si zvolíme atribút HomeTeam. Na obrázku vidíme, že kardinalita tohto atribútu je 41 a vieme, že ide o názvy tímov v tvare reťazca znakov. Po kliknutí na Convert to numeric sa v ďalšej bunke spustí príkaz `changeColumnType` s parametrami frame, atribút, výstupný typ a na obrazovku sa vypíše zhrnutie s príslušnými grafmi o charakteristike a distribúcii údajov atribútu HomeTeam. Tento proces konverzie dát spravíme aj pre ostatné nie číselné hodnoty

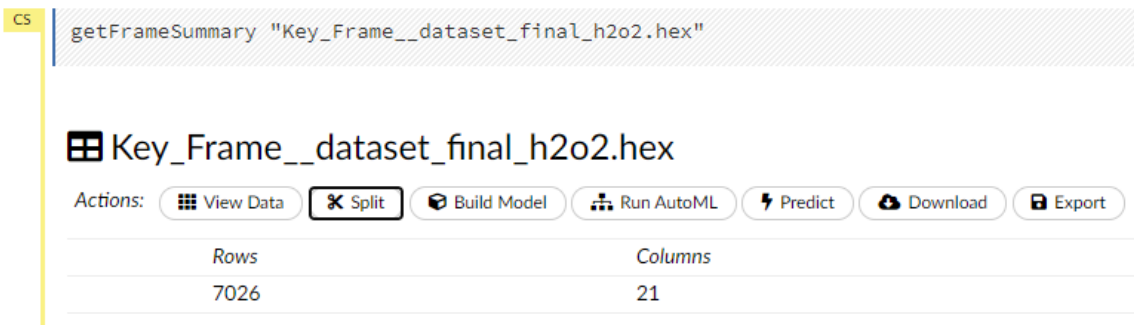
atribútov. Po dokončení príkazu `changeColumnType` sa nám na obrazovku vypíše report o konverzii ukázaný na Obrázku 60.



Obrázok 60.H2O Flow `changeColumnTypes` report

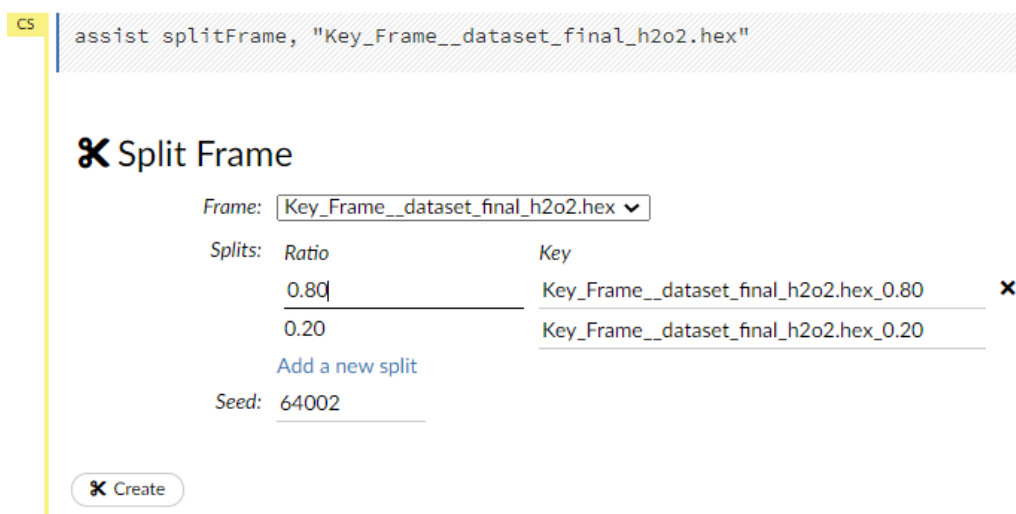
3.6.8. Rozdelenie dát na testovacie a trénovacie v H2O

V momente keď máme dataset nahratý a predspracovaný, dáta musíme rozdeliť na testovaciu (validačnú) časť a trénovacie dáta, aby sme mohli overiť správnosť modelu. V H2O spustíme príkaz `getFrameSummary` a v bunke klikneme na tlačidlo Split, čím spustíme príkaz `assist splitFrame`.



Obrázok 61. H2O Flow príkaz `getFrameSummary`

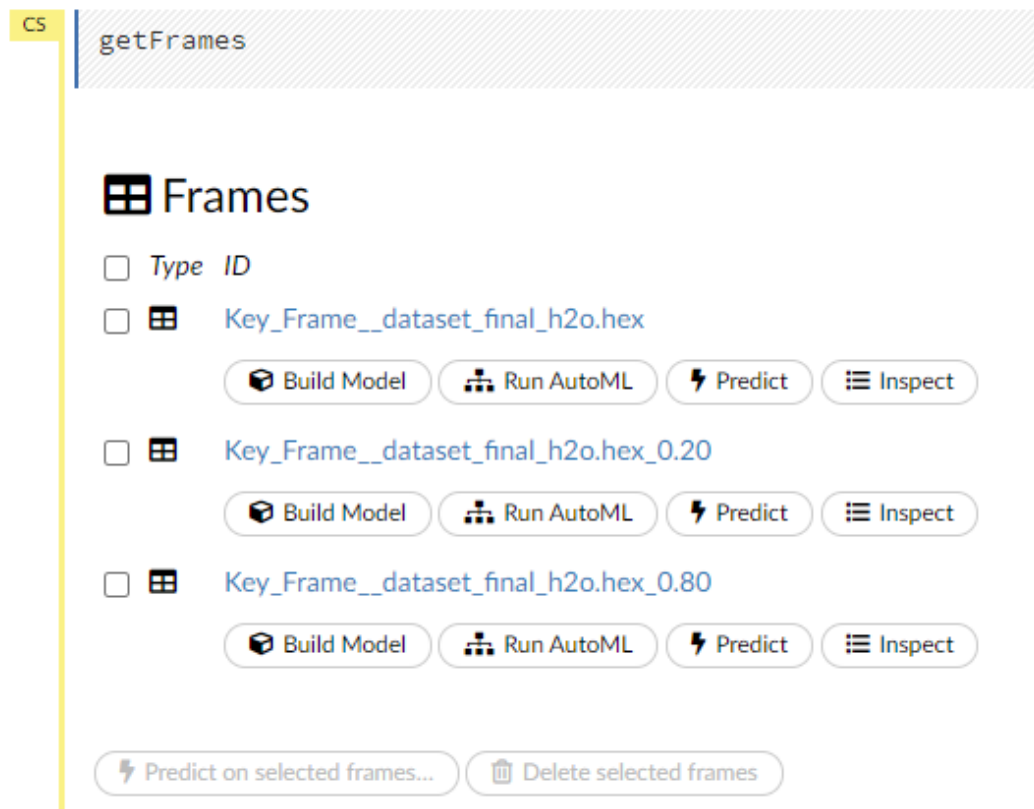
Príkaz `assist splitFrame` nás presmeruje do novej bunky, kde môžeme nastaviť konfiguráciu pre rozdelenie dát. Nastavíme pomer 80:20 a klikneme na tlačidlo Create.



Obrázok 62. Rozdelenie dát na tréningovú a validačnú časť v H2O Flow

Kliknutím na tlačidlo Create sa spustí procedúra `splitFrame` s parametrami [H2O Frame, 80, 20] a tá vráti dva nové H2O Frame objekty, z ktorých neskôr jeden použijeme ako tréningovú sadu dát a druhú ako validačnú sadu dát. Keď si chceme overiť či sa tieto sady dát vytvorili, spustíme príkaz `getFrames`. Jeho výpis obsahuje všetky H2O frame objekty,

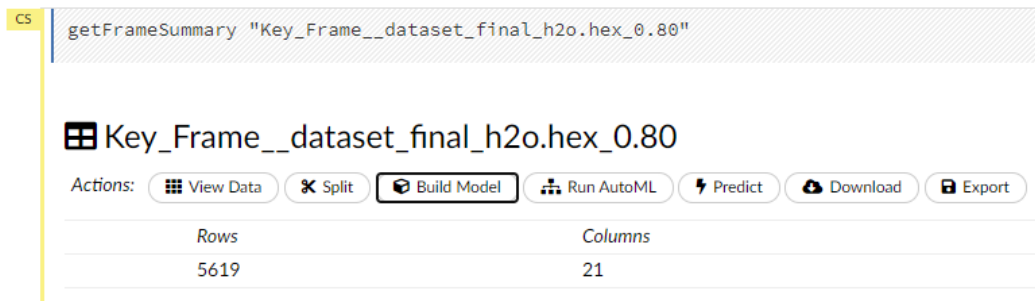
ktoré sme vytvorili. Ktoré H2O frame objekty sú práve uložené v pamäti, vidíme na Obrázku 63. V pamäti sa nachádza pôvodný frame a jeho rozdelenie na dva ďalšie.



Obrázok 63. Zoznam všetkých H2O Frames v pamäti H2O Flow

3.6.9. Vytvorenie modelu

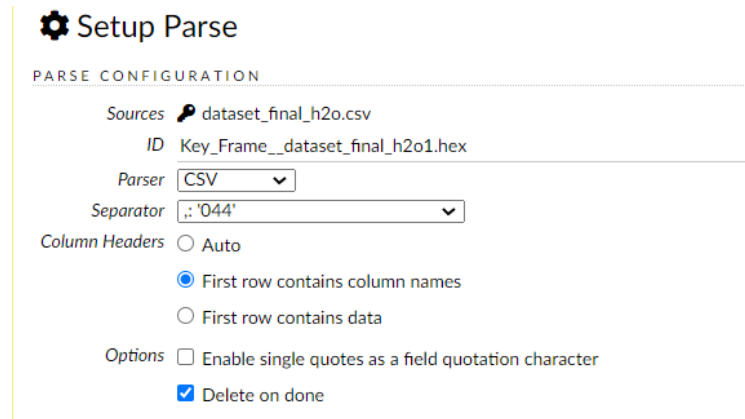
V predchádzajúcich kapitolách sme spracovali naše dáta v platforme H2O, vytvorili sme testovacie a validačné sety dát a teraz sa môžeme pustiť do tvorby klasifikačného modelu pre futbalový dataset. V H2O spustíme príkaz `getFrameSummary` s jedným parametrom, a to názov tréningového H2O Frame. V bunke na výstupe klikneme na tlačidlo `Build Model`. `Build Model` spustí v ďalšej bunke príkaz `assist buildModel` a rozhranie od nás vyžaduje vybrať si z ponúkaných algoritmov.



Obrázok 64. H2O Flow a vytvorenie modelu


Naive Bayes a SVM sú algoritmy, ktoré sme používali doteraz, a tu nastávajú prvé problémy. Keď z ponuky zvolíme Naive Bayes, nastane problém s dátami, ktoré nie sú kategorické a voľba nám vyhodila chybu. Algoritmus Naive Bayes v H2O je schopný spracovať iba kategorické dáta. S týmto problémom sme sa vysporiadali tak, že sme dáta konvertovali na typ ENUM v okne `getFrameSummary`. Ďalším problémom bolo, že po nastavení všetkých parametrov sme v sumári výsledkov našli jednu nezrovnalosť so spracovanými dátami. Naším cieľovým atribútom je FTR, ktorý nadobúda hodnoty H, A, D. Avšak v H2O sa stalo to, že do úvahy a do modelu započítal aj názov stĺpca ako hodnotu, čiže FTR nadobúdal hodnoty H, A, D a aj FTR čo nie je správne. Toto všetko sa udialo aj napriek tomu, že v bunke `Setup Parse` sme zaškrtnuli, aby prvý riadok CSV súboru brala platforma ako názov atribútu.

Tento problém sme vyriešili tak, že priamo vo vstupnom CSV súbore sme vymazali prvý riadok s názvami atribútov a pri `Setup Parse` sme zaškrtnuli `First row contains data`, čo v preklade znamená, že už prvý riadok v CSV súbore obsahuje dáta. Na Obrázku 65 by sme v časti `Column Headers` zaškrtnuli tretiu možnosť. Týmto krokom sme dosiahli, že vo výsledku dostávame správne výsledky a model pracuje len s relevantnými hodnotami atribútov.



Setup Parse

PARSE CONFIGURATION

Sources  dataset_final_h2o.csv

ID Key_Frame__dataset_final_h2o1.hex

Parser CSV

Separator ;;'044'

Column Headers

☐ Auto

☒ First row contains column names

☐ First row contains data

Options

☐ Enable single quotes as a field quotation character

☒ Delete on done

Obrázok 65. H2O Flow Setup Parse

Čo sa týka SVM metódy, tá nie je v H2O Flow podporovaná a celkovo v knižnici H2O ešte nie je implementovaná verzia metódy podporných vektorov, kde môžeme klasifikovať viac tried. Implementácia algoritmu stroja podporných vektorov H2O sa riadi špecifikáciou algoritmu PSVM: Parallelizing Support Vector Machines on Distributed Computers a je možné ju použiť iba na riešenie problémov s binárnou klasifikáciou.

3.6.10. Trénovanie a testovanie

Odbočíme od prvých problémov a pokúsime sa vytvoriť prvý model v H2O. Keďže dáta už máme v správnom tvare, zvolíme z ponuky Naive Bayes algoritmus.

CS

assist buildModel, null, training_frame: "Key_Frame__dataset_final_h2o.hex_0.80"

Build a Model

Select an algorithm: Naive Bayes

PARAMETERS

model_id	naivebayes-ab74eee7-1bf3-4efc-841c-3d8fa67d431d	Destination id for this
nfolds	0	Number of folds for K
training_frame	Key_Frame__dataset_final_h2o.hex_0.80	Id of the training data
validation_frame	Key_Frame__dataset_final_h2o.hex_0.20	Id of the validation da
response_column	C5	Response variable col

Obrázok 66. H2O Flow konfigurácia prediktívneho modelu

V konfigurácii nastavíme trénovaciu sadu v poli `training_frame`, validačnú sadu v poli `validation_frame` a `response column` nastavíme na `C5`, čo zodpovedá nášmu atribútu FTR. Bunka Build Model obsahuje množstvo parametrov a nastavení, ktoré sa dajú využiť pri pokročilejších modeloch. Pre naše skúmanie si vystačíme so základným nastavením. V momente, keď máme kompletnú konfiguráciu, ktorú chceme použiť, klikneme na tlačidlo Build Model nižšie.

CS

```
buildModel 'naivebayes', {"model_id":"naivebayes-ab74eee7-1bf3-4efc-841c-3d8fa67d431d","nfolds":0,"training_frame":"Key_Frame__dataset_final_h2o.hex_0.80","validation_frame":"Key_Frame__dataset_final_h2o.hex_0.20","response_column":"C5","ignore_const_cols":true,"laplace":0,"min_sdev":0.001,"eps_sdev":0,"min_protect":20,"max_runtime_secs":0,"gainslift_bins":-1,"auc_type":"AUTO","seed":-1}
```

Job

Run Time 00:00:00.88

Remaining Time 00:00:00.0

Type Model

Key naivebayes-ab74eee7-1bf3-4efc-841c-3d8fa67d431d

Description NaiveBayes

Status DONE

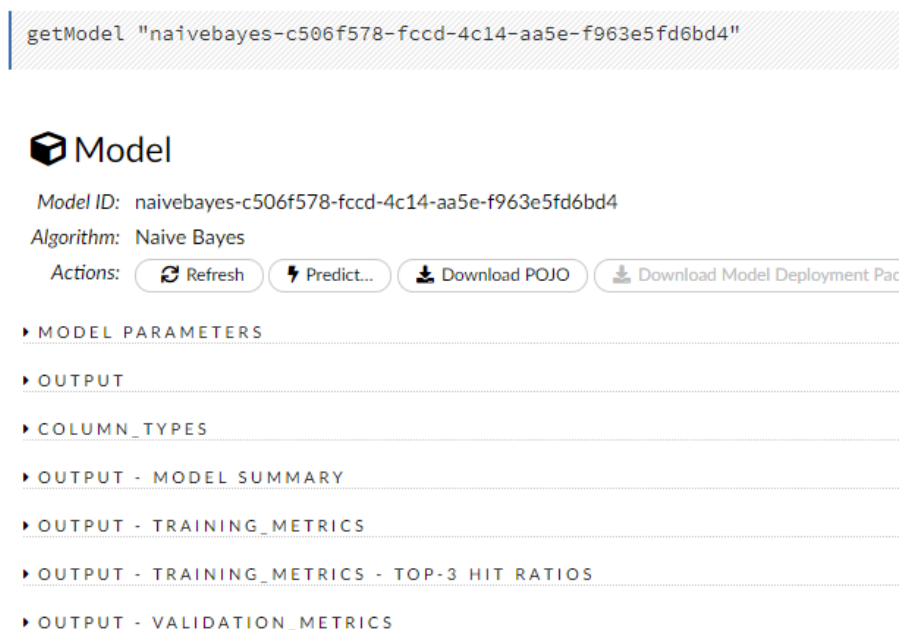
Progress 100%

Done.

Actions View

Obrázok 67. H2O Flow príkaz `buildModel`


Tlačidlo Build Model spustilo procedúru buildModel s parametrami, ktoré sme špecifikovali v bunke vyššie. Klikneme na tlačidlo View, spustí sa funkcia getModel s parametrom, ktorým je názov modelu, v bunke na výstupe nájdeme informácie o modeli.



Obrázok 68. H2O Flow informácie o vytvorenom prediktívnom modeli

Vytvorený model môžeme otestovať kliknutím na tlačidlo Predict. Spustí sa procedúra predict model, ktorej najdôležitejšie 2 parametre sú objekty model a H2O frame. Výstupom tejto procedúry je výpis s informáciami o vykonaní testu (predikcie). Výstup Prediction obsahuje štatistické údaje o presnosti nášho modelu, napríklad confusion matrix, MSE, RMSE, Precision, Recall. Parameter hit_ratio na Obrázku 70 vyjadruje accuracy (metrika výkonu) pre všetky hodnoty, ktoré môže nadobúdať atribút FTR.

Prediction

Actions:  Inspect

▼ PREDICTION

model	naivebayes-c506f578-fccd-4c14-aa5e-f963e5fd6bd4
model_checksum	-1987590577148264656
frame	Key_Frame__dataset_final_H2O.hex_0.20
frame_checksum	6773728234689247735
description	•
model_category	Multinomial
scoring_time	1617298052051
predictions	prediction-4ef7bc3d-accb-4306-aaf8-79af4140d3e9
MSE	0.177192
RMSE	0.420942
nobs	1387
custom_metric_name	•
custom_metric_value	0
r2	0.751204
logloss	0.539264
mean_per_class_error	0.259541
AUC	NaN
pr_auc	NaN
multinomial_auc_table	•
multinomial_aucpr_table	•

Obrázok 69. H2O Flow výstup Prediction

▼ PREDICTION - TOP-3 HIT RATIOS						
k	hit_ratio					
1	0.7599					
2	0.9668					
3	1.0					

▼ PREDICTION - CONFUSION MATRIX ROW LABELS: ACTUAL CLASS; COLUMN LABELS: PREDICTED CLASS						
	A	D	H	Error	Rate	Precision
A	290	71	12	0.2225	83 / 373	0.79
D	57	208	64	0.3678	121 / 329	0.54
H	20	109	556	0.1883	129 / 685	0.88
Total	367	388	632	0.2401	333 / 1,387	
Recall	0.78	0.63	0.81			

Obrázok 70. H2O Flow výstup Prediction – confusion matrix a hit_ratio

3.6.11. H2O a SVM

V predošlej kapitole sme si ukázali, ako vytvoriť model Naive Bayes v prostredí H2O Flow. Ak chceme vytvoriť model algoritmom SVM, H2O Flow nám nepostačí a musíme sa uchýliť k viac programátorskému riešeniu. Na to použijeme knižnicu H2O a programovací jazyk Python. Toto riešenie je oveľa kratšie ako použitie H2O Flow, no vyžaduje znalosti programovania. Procesy ako analýza, rozdelenie dát, vytvorenie modelu a zhrnutie výkonnosti nám zaberie pár riadkov kódu. Ako programátorské rozhranie zvolíme Jupyter a postupujeme tak isto ako pri knižnici sklearn, čítame dokumentáciu a krok po kroku sa pokúsime vytvoriť aj model pre SVM za pomoci H2O.

Tu však narazíme na problém. Implementácia SVM v H2O sa riadi špecifikáciou PSVM: Parallelizing Support Vector Machines on Distributed Computers a je možné ju použiť iba na riešenie problémov s binárnou klasifikáciou [30]. V prípade futbalového datasetu však ide o problém klasifikácie viacerých tried. Preto si ukážeme iba vzorový príklad a budeme dúfať, že ľudia z tímu H2O implementujú túto funkcionálnu činnosť čím skôr.

Na začiatku musíme nainštalovať knižnicu h2o pomocou package manažera pip. Tento krok nie je zobrazený na obrázku, ale pre jeho jednoduchosť si myslíme, že to nie je potrebné a každý kto programoval v Pythone, intuitívne príde na to, ako má príkaz na inštaláciu knižnice h2o vyzerieť. Do konzoly (alebo bunky v Jupyter) napíšeme nasledujúci príkaz a počkáme na jeho vykonanie:

- `pip install h2o.`

Ďalej budeme postupovať opisom obrázku, ktorý obsahuje kód. Príkazom `import` povieme programu, aby používal metódy h2o, ktoré potrebujeme. Pre použitie algoritmu SVM v H2O importujeme funkciu `H2OSupportVectorMachineEstimator`. Metódou z knižnice h2o `import_files()`, nahráme do pamäte dataset a príkazom `describe()` si dáme dataset vypísať na obrazovku. Funkcia `describe()` je v podstate veľmi podobná funkcii `head()` z knižnice Pandas. Ďalším krokom je rozdelenie datasetu na trénovaciu a validačnú sadu, čo urobíme funkciou `split_frame()`. Funkcia `split_frame()` obsahuje 2 parametre, a to pomer, v akom chceme dáta rozdeliť (v našom prípade 80:20). Hodnota `seed` je číslo, ktoré riadi, či generátor náhodných čísel vytvára novú sadu náhodných čísel alebo opakuje konkrétnu postupnosť náhodných čísel (pretože chceme, aby rozdelenie dát bolo náhodné). Ďalej použijeme funkcie:

- `H2OSupportVectorMachineEstimator()` na vytvorenie modelu,
- `train()` na trénovanie modelu (určíme výstupný atribút, trénovaciu a validačnú sadu),
- `model_performance()` na vyhodnotenie výkonu modelu na validačnej sade.

```

import h2o
from h2o.estimators import H2OSupportVectorMachineEstimator
h2o.init()

# Import the splICE dataset into H2O:
df = h2o.import_file("creditcard.csv")
df.describe()

train, valid = df.split_frame(ratios = [.8], seed = 1234)

# Build and train the model:
svm_model = H2OSupportVectorMachineEstimator(gamma=0.01,
                                              rank_ratio = 0.1,
                                              disable_training_metrics = False)

svm_model.train(y = "Fraud", training_frame = train, validation_frame = valid)
perf = svm_model.model_performance(valid)
|
# h2o.cluster().shutdown(prompt=True)

```

Obrázok 71. SVM v Jupyter+H2O

Po spustení kódu sa vytvorí webový server, ktorý sa snaží komunikovať s H2O serverom. Tento fakt je zrejmý z prvého výpisu, ktorý sa zobrazí po spustení. Webový server komunikuje s klastrom H2O a podrobnosti vidíme na Obrázku 72.

Checking whether there is an H2O instance running at http://localhost:54321 . connected.

H2O_cluster_uptime:	17 mins 37 secs
H2O_cluster_timezone:	Europe/Prague
H2O_data_parsing_timezone:	UTC
H2O_cluster_version:	3.32.1.1
H2O_cluster_version_age:	1 month and 3 days
H2O_cluster_name:	H2O_from_python_ZZ022U693_at90bt
H2O_cluster_total_nodes:	1
H2O_cluster_free_memory:	223.2 Mb
H2O_cluster_total_cores:	8
H2O_cluster_allowed_cores:	8
H2O_cluster_status:	locked, healthy
H2O_connection_url:	http://localhost:54321
H2O_connection_proxy:	{"http": null, "https": null}
H2O_internal_security:	False
H2O_API_Extensions:	Amazon S3, Algos, AutoML, Core V3, TargetEncoder, Core V4
Python_version:	3.7.1 final

Obrázok 72. Informácie o spojení s klastrom H2O

Po vytvorení modelu SVM vypíšeme na obrazovku indikátory výkonu modelu funkciou `model_performance()`. Je to podobný výpis ako pri spustení tlačidla Predict v H2O Flow. Metriky výkonu, ktoré táto funkcia poskytuje, vidíme na Obrázku 73.

[illegible]

```
ModelMetricsBinomial: psvm
** Reported on test data. **
```

```
MSE: 0.17676767676767677
RMSE: 0.42043748259126085
LogLoss: NaN
Mean Per-Class Error: 0.16826923076923084
AUC: NaN
AUCPR: NaN
Gini: NaN
```

Confusion Matrix (Act/Pred) for max f1 @ threshold = 1.0:

		False	True	Error	Rate
0	False	69.0	35.0	0.3365	(35.0/104.0)
1	True	0.0	94.0	0.0	(0.0/94.0)
2	Total	69.0	129.0	0.1768	(35.0/198.0)

Maximum Metrics: Maximum metrics at their respective thresholds

	metric	threshold	value	idx
0	max f1	1.0	0.843049	0.0
1	max f2	1.0	0.930693	0.0
2	max f0point5	1.0	0.770492	0.0
3	max accuracy	1.0	0.823232	0.0
4	max precision	1.0	0.728682	0.0
5	max recall	1.0	1.000000	0.0
6	max specificity	1.0	0.663462	0.0
7	max absolute_mcc	1.0	0.695308	0.0
8	max min_per_class_accuracy	1.0	0.663462	0.0
9	max mean_per_class_accuracy	1.0	0.831731	0.0
10	max tns	1.0	69.000000	0.0
11	max fns	1.0	0.000000	0.0
12	max fps	1.0	35.000000	0.0
13	max tps	1.0	94.000000	0.0
14	max fnr	1.0	0.663462	0.0

Obrázok 73. Výpis funkcie `model_performance()`

3.6.12. AutoML: Automatic Machine Learning

V kapitolách vyššie sme hovorili o nedostatkoch H2O Flow, a to konkrétne, že SVM algoritmus nie je podporovaný v takej miere, ako by sme potrebovali na riešenie našej úlohy. Avšak H2O ponúka na prvý dojem veľmi zaujímavý nástroj AutoML, ktorý slúži na správny výber učiaceho sa algoritmu, ladenie modelu a celkovo akú konfiguráciu použiť pri vytváraní modelu na užívateľom zadanom datasete.

V posledných rokoch dopyt po odborníkoch na strojové učenie predbehol ponuku, a to aj napriek nárastu počtu ľudí v odbore. Na vyriešenie tohto nedostatku došlo k veľkým pokrokom vo vývoji užívateľsky prívetivého softvéru pre strojové učenie, ktorý môžu používať aj neodborníci. Prvé kroky k zjednodušeniu strojového učenia spočívali vo vývoji jednoduchých a zjednotených rozhraní s rôznymi algoritmami strojového učenia [31].

Aj keď H2O uľahčilo užívateľom experimentovať so strojovým učením, stále existuje dostatok vedomostí a znalostí v oblasti dátovej vedy, ktoré sú potrebné na výrobu výkonných modelov strojového učenia. Najmä nakonfigurovať neurónové siete je pre neodborníka notoricky ťažké. Aby bol softvér pre strojové učenie skutočne prístupný neodborníkom, H2O navrhli ľahko použiteľné rozhranie, ktoré automatizuje proces trénovania veľkého množstva kandidátskych modelov. AutoML vo formáte H2O môže byť tiež užitočným nástrojom pre pokročilých používateľov tým, že poskytuje jednoduchú funkciu, ktorá vykonáva veľké množstvo úloh súvisiacich s modelovaním, ktoré by zvyčajne vyžadovali veľa riadkov kódu, AutoML šetrí čas zameraním sa na ďalšie aspekty datascience, ako je predbežné spracovanie údajov, inžinierstvo funkcií a nasadenie modelov [31].

Program H2O AutoML je možné použiť na automatizáciu strojového učenia, ktorý zahŕňa automatické učenie a konfiguráciu prediktívnych modelov v časovom limite stanovenom používateľom. H2O ponúka množstvo metód opisu modelu, ktoré sa vzťahujú na objekty AutoML (skupiny modelov), ako aj na jednotlivé modely. Opisy je možné generovať automaticky pomocou jedného volania funkcie, ktoré poskytuje jednoduché rozhranie na skúmanie a opis modelov AutoML.

3.6.13. Konfigurácia Auto ML


Rozhranie H2O AutoML je navrhnuté tak, aby malo čo najmenej parametrov, takže používateľ musí importovať svoju množinu údajov, identifikovať výstupný atribút a voliteľne určiť časové obmedzenie alebo limit počtu vycvičených modelov. V rozhraní API R aj Python používa AutoML rovnaké argumenty súvisiace s údajmi, x, y, training_frame, validation_frame, ako ostatné algoritmy H2O. Väčšinou stačí zadať argumenty údajov. Potom môžeme nakonfigurovať hodnoty pre max_runtime_secs a / alebo max_models, aby sme nastavili explicitné časové limity alebo limity počtu modelov.

3.6.14. Požadované parametre

Parameter y a training_frame sú povinné polia označené hviezdíčkou na Obrázku 74.

- y (response_column): Tento argument je názov (alebo index) výstupného stĺpca (atribútu).
- training_frame: Určuje tréningovú sadu.

```
assist runAutoML, training_frame: "Key_Frame__dataset_final_H2O.hex_0.80"
```

 Run AutoML

PARAMETERS

project_name

training_frame*

Key_Frame__dataset_final_H2O.hex_0.80

▼

response_column*

C5

▼

validation_frame

Key_Frame__dataset_final_H2O.hex_0.20

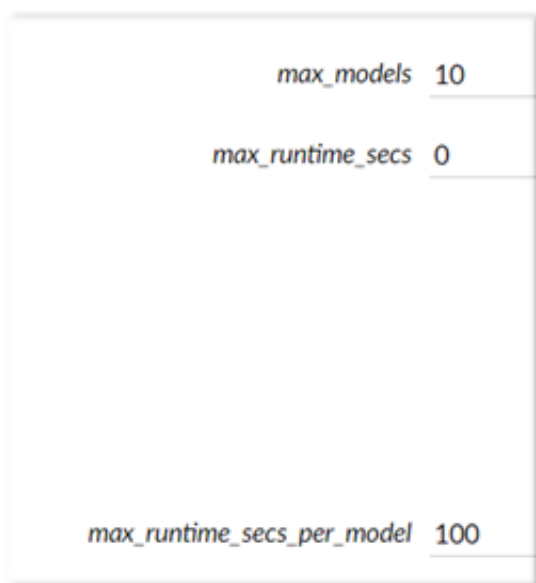
▼

Obrázok 74. H2O Run AutoML

V AutoML môžeme nastaviť, kedy má tento proces skončiť. Ak nenastavíme nasledujúce hodnoty parametrov AutoML, ľahko sa môže stať, že ukazovateľ dokončenia

procesu AutoML tzv. progress bar po určitom čase zastaví na určitej hodnote. Na vlastnej koži sme zažili, že AutoML bežal celú noc, vytváral nové a nové modely a stav dokončenia bol niekoľko hodín 48% a nepohol sa ďalej. Parametre `max_runtime_secs` a `max_models` sú polia povinné vtedy, keď požadujeme, aby sa proces Auto ML zastavil pri dosiahnutí hodnôt týchto polí. `Max_runtime_secs` je nastavený pôvodne na 3600 sekúnd (1 hodina). Tieto parametre sú žiadúce hlavne vtedy, keď spúšťame H2O na laptope, stolovom počítači alebo zariadení s nie vysokým výkonom.

Parameter `max_runtime_secs_per_model` vyjadruje maximálnu hodnotu času stráveného pri jednom modeli. Aby sme boli schopní dokončiť proces AutoML na našom systéme v normálnom čase, nastavíme túto hodnotu na 100 a `max_models` nastavíme na 20.



Obrázok 75. AutoML parametre

Ďalšou dôležitou informáciou je aj to, aké typy modelov AutoML vytvorí. V zozname algoritmov, ktoré sú použité v AutoML môžeme nájsť:

- GLM – Generalized Linear Model
- DRF – Distributed Random Forest

- GBM – Gradient Boosting Machine
- DeepLearning
- StackedEnsemble
- XGBoost

3.6.15. Spustenie AutoML

Po nastavení parametrov AutoML spustíme celý proces tlačidlom Build Models. Následne sa spustí príkaz runAutoML s parametrami, ktoré sme zadali v okne bunky Run AutoML. Po spustení vidíme v novej bunke Job, ako prebieha proces AutoML. Bunka Job nám ukazuje napríklad stav procesu AutoML, čas, progress bar. Vo výpise na Obrázku 76 vidíme aj práve vytváraný model.

```
runAutoML {"input_spec":{"training_frame":"frame_0.80","response_col":
[],"sort_metric":"AUTO"},"build_control":{"nfolds":0,"balance_classi
{"seed":-1,"max_models":10,"max_runtime_secs":0,"max_runtime_secs_pe
pping_tolerance":-1},"keep_cross_validation_predictions":false,"keep
signment":false},"build_models":{"exclude_algos":[],"exploitation_ra
```

Job

Run Time 00:01:09.972

Remaining Time 00:38:49.68

Type Auto Model

Key 🔍 [AutoML_20210411_123905711@@FTR](#)

Description AutoML build

Status RUNNING

Progress 3%

GLM_1_AutoML_20210411_123905 [GLM def_1]

Actions 🔍 View ⏹ Cancel Job

Obrázok 76. H2O AutoML Job

3.6.16. AutoML Leaderboard

Po skončení procesu AutoML sa môžeme pozrieť na výsledky príkazom `getLeaderboard`. Na obrázku vidíme Leaderboard, časť výpisu znázorňujúceho, ktoré modely boli vytvorené a v tabuľke sú uvedené hodnoty ukazovateľov výkonu alebo presnosti vytvorených modelov. Modely sú zoradené podľa predvolenej metriky na základe typu problému (druhý stĺpec výsledkovej tabuľky v Obrázku 77). Pri problémoch s binárnou klasifikáciou je touto metrikou AUC a pri problémoch s klasifikáciou viacerých tried je metrikou stredná chyba v každej triede. V prípade regresných problémov je predvolenou metrikou zoradenia odchýlka. Poskytované sú aj niektoré ďalšie metriky.

CS

getLeaderboard "AutoML_20210411_155232208@@FTR"

Leaderboard

Monitor Live

▼ MODELS

models sorted in order of mean_per_class_error, best first

	model_id	mean_per_class_error	logloss
0	GBM_grid__1_AutoML_20210411_155232_model_2	0	0.0012139987417048588
1	DeepLearning_1_AutoML_20210411_155232	0	0.00006566104605675581
2	DeepLearning_grid__2_AutoML_20210411_155232_model_1	0.002331002331002331	0.00421737332513661
3	GBM_5_AutoML_20210411_155232	0.004140786749482401	0.010410884200328806
4	GBM_1_AutoML_20210411_155232	0.004140786749482401	0.03217899526359269
5	GBM_2_AutoML_20210411_155232	0.004140786749482401	0.028179959233207333
6	DeepLearning_grid__1_AutoML_20210411_155232_model_1	0.008802791411487064	0.06950924791861403
7	DRF_1_AutoML_20210411_155232	0.015013971535710666	0.16203954885346392
8	XRT_1_AutoML_20210411_155232	0.1030594119838971	0.4114487076532983
9	GLM_1_AutoML_20210411_155232	0.4222794749110539	0.8704631133210595

Obrázok 77. H2O AutoML Leaderboard

Ukazovatele presnosti je možné meniť počas konfigurácie AutoML. Ich výber závisí aj od typu modelu. Nižšie sa nachádza tzv. Event Log, čo je v podstate terminálový výpis AutoML

do konzoly. Vo výpise v bunke sa nachádzajú dôležité informácie o akciách a objavoch AutoML. Medzi akcie patrí napríklad:

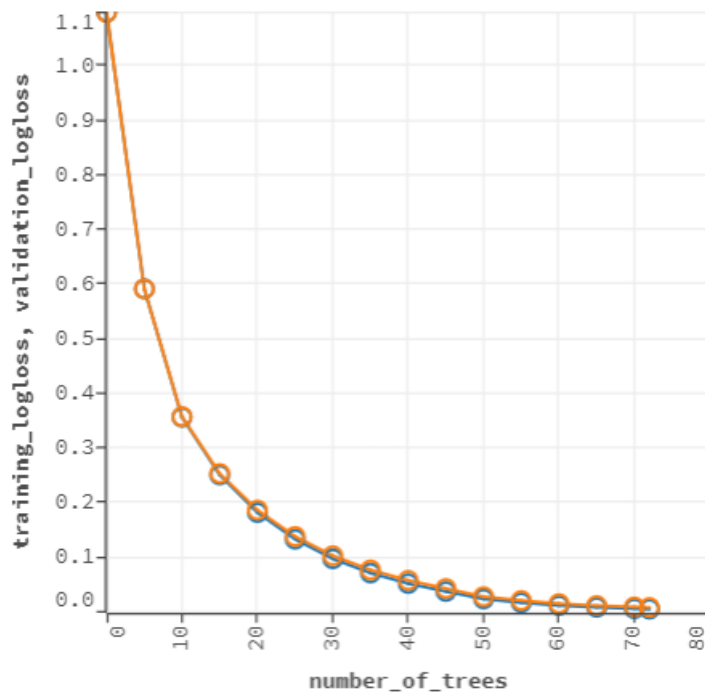
- DataImport – import dát
- Validation – validácia
- ModelTraining – tréno vanie modelu

3.6.17. Detail modelu AutoML

AutoML pre nás vytvoril niekoľko modelov, ktoré môžeme porovnať. Keď klikneme na jeden z vytvorených modelov, zobrazí sa bunka s rôznymi informáciami o výkonnosti modelu s niekoľkými grafmi. Prvým je funkcia logloss, vyjadrená ako pre tréno vaciu, tak aj validačnú sadu.

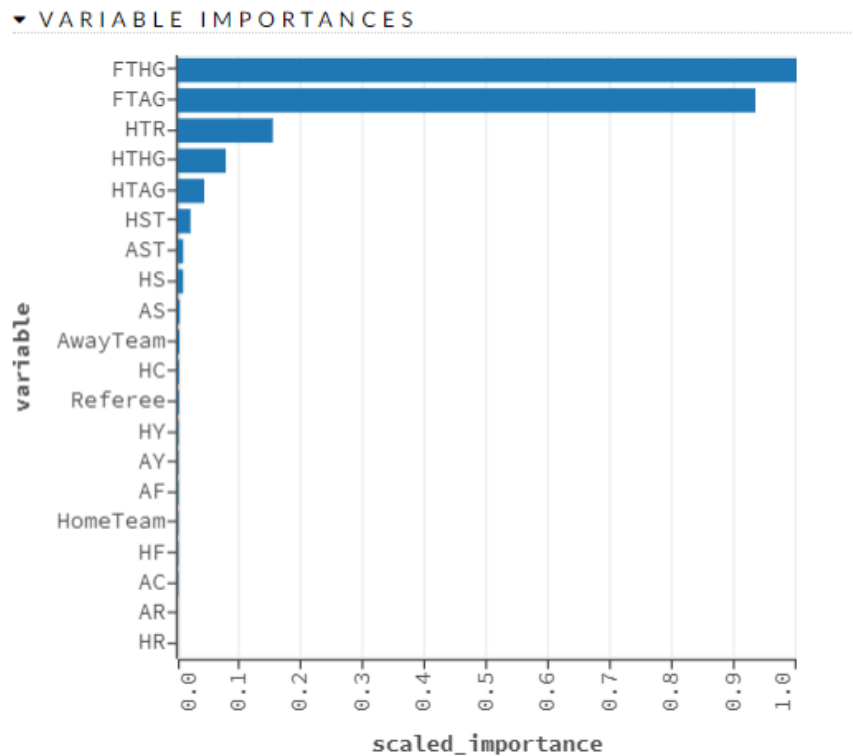
► MODEL PARAMETERS

▼ SCORING HISTORY - LOGLOSS



Obrázok 78. Zobrazenie metriky Logloss v H2O AutoML

Ďalší graf môžeme považovať za určitú formu selekcie atribútov. Graf vyjadruje dôležitosť atribútov pri vytváraní modelov na škále od 0 do 1.



Obrázok 79. H2O AutoML – selekcia atribútov

H2O AutoML nám poskytuje aj niekoľko tabuliek s confusion matrix, a to pre tréningovú, validačnú sadu. Na obrázku je zobrazená úspešnosť modelu GBM na validačnej sade dát.

▼ VALIDATION METRICS - CONFUSION MATRIX ROW LABELS

	A	D	H	Error	Rate	Precision
A	446	0	1	0.0022	1 / 447	1.0
D	0	321	0	0	0 / 321	1.0
H	0	0	613	0	0 / 613	1.0
Total	446	321	614	0.0007	1 / 1,381	
Recall	1.0	1.0	1.0			

Obrázok 80. Príklad Confusioin matrix v H2O AutoML

Ďalšie vlastnosti poskytuje výpis metrics. Na obrázku vidíme validation_metrics s parametrami ako MSE, RMSE, R2, Logloss.

▼ OUTPUT - VALIDATION_METRICS

<i>model</i>	GBM_5_AutoML_20210411_202257
<i>model_checksum</i>	8081888583103410304
<i>frame</i>	Key_Frame__dataset_final_H2O_all.hex_0.20
<i>frame_checksum</i>	193722888656
<i>description</i>	•
<i>model_category</i>	Multinomial
<i>scoring_time</i>	1618165660792
<i>predictions</i>	•
<i>MSE</i>	0.001989
<i>RMSE</i>	0.044602
<i>nobs</i>	1381
<i>custom_metric_name</i>	•
<i>custom_metric_value</i>	0
<i>r2</i>	0.997358
<i>logloss</i>	0.009681
<i>mean_per_class_error</i>	0.000746
<i>AUC</i>	NaN
<i>pr_auc</i>	NaN
<i>multinomial_auc_table</i>	•
<i>multinomial_aucpr_table</i>	•

Obrázok 81. H2O AutoML validation metrics

Ak v konfigurácii zvolíme možnosť cross-validácie modelu, potom na výstupe dostaneme aj cross-validation metrics summary, kde máme aj parameter accuracy (presnosť).

▼ OUTPUT - CROSS-VALIDATION METRICS SUMMARY

	mean	sd	cv_1_valid	cv_2_valid
accuracy	0.9916726	0.0087699285	0.9854713	0.99787384
auc	NaN	0.0	NaN	NaN
aucpr	NaN	0.0	NaN	NaN
err	0.008327427	0.0087699285	0.014528703	0.0021261517
err_count	23.5	24.748737	41.0	6.0
logloss	0.04637238	0.053980988	0.0845427	0.008202057
max_per_class_error	0.029677771	0.030347032	0.051136363	0.008219178
mean_per_class_accuracy	0.9892265	0.011361435	0.98119277	0.9972603
mean_per_class_error	0.010773473	0.011361435	0.018807221	0.002739726
mse	0.009136169	0.010156191	0.01631768	0.0019546575
pr_auc	NaN	0.0	NaN	NaN
r2	0.98721105	0.014165136	0.9771948	0.9972273
rmse	0.085976094	0.05906404	0.12774068	0.04421151

Obrázok 82. H2O AutoML cross-validation metrics

Ak chceme vytvorený model otestovať, klikneme na Predict, vložíme hodnotu do poľa testovacie dáta a aj takto môžeme dodatočne overiť správnosť a výkon vytvoreného modelu.

3.7. Dataset s údajmi o kreditných kartách

V predošlej kapitole sme si na ukážku vzali dataset o športe, no uvedieme si aj príklad, ktorý sa týka sveta ekonómie a financií. Je dôležité, aby spoločnosti vydávajúce kreditné karty dokázali rozpoznať podvodné transakcie kreditnou kartou, aby zákazníkom neboli účtované poplatky za položky, ktoré si nekúpili. Úlohou v tomto prípade bude binárna klasifikácia a vytvoríme machine learning model, ktorý bude schopný predikovať či peňažná transakcia je alebo nie je podvod. Preto sme hľadali vhodný dataset na splnenie tejto úlohy. Dataset, ktorý sme našli na Kaggle.com, obsahuje transakcie uskutočnené kreditnými kartami v septembri 2013 európskymi držiteľmi kariet. Tento súbor údajov predstavuje transakcie, ku ktorým došlo za dva dni, kedy bolo zaznamenaných 492 podvodov z 284 807 transakcií [32]. Súbor údajov je vysoko nevyvážený, kladná trieda (podvody) predstavuje 0,172% všetkých transakcií. Tento fakt nás prinútil dataset značne zmenšiť a optimalizovať na vyváženú vzorku súboru údajov kvôli technickému vybaveniu, na ktorom daný problém

chceme riešiť. Aby sme vytvorili vybalansovanú vzorku súboru údajov, vybrali sme 1000 záznamov, z ktorých takmer polovicu tvoria podvodné transakcie. Dataset obsahuje iba numerické vstupné premenné, ktoré sú výsledkom šandardizácie dát pomocou transformácie PCA [32]. Z dôvodu problémov s dôvernosťou neboli poskytnuté pôvodné atribúty a ďalšie základné informácie o údajoch. Funkcie V1, V2, ... V28 sú hlavné komponenty získané pomocou PCA, jediné atribúty, ktoré neboli transformované pomocou PCA, sú Time a Amount. Atribút Time obsahuje sekundy, ktoré uplynuli medzi každou transakciou a prvou transakciou v množine údajov. Objekt Class je atribút odpovede a v prípade podvodu má hodnotu 1, v opačnom prípade hodnotu 0. Tento atribút v zázname označuje, či tento záznam je podvod alebo nie. Atribúty aj s vysvetlivkami uvádzame v tabuľke.

Kľúč	Hodnota
V1-V28	Štandardizované dáta neznámych atribútov (number)
Amount	Suma transakcie (number)
Time	Sekundy od prvej transakcie pozorovania (number)
Class / Fraud	{0,1}

Tabuľka 4. Vysvetlivky datasetu údajov o kreditných kartách

3.7.1. Predikcia podvodov vo finančných platobných službách

Odvetvie finančných služieb a odvetvia, ktoré zahŕňajú finančné transakcie, trpia stratami a škodami spojenými s podvodmi. Napríklad v roku 2016 len v USA dosiahol počet zákazníkov, ktorí zažili podvod, rekordných 15,4 milióna ľudí, čo je o 16 percent viac ako v roku 2015. Podvodníci ukradli bankám v roku 2016 asi 6 miliárd dolárov. Prechod na digitálny priestor otvára nové kanály pre distribúciu finančných služieb [33].

Ak zločinci v minulosti museli sfalšovať identifikačné čísla klienta, na krádež peňazí môže byť teraz potrebné už len získanie hesla k účtu osoby. Vernosť zákazníka a konverzie sú ovplyvnené v oboch prostrediach, digitálnom aj fyzickom. Podľa agentúry Javelin Strategy & Research trvá odhalenie podvodov pre kamenné finančné inštitúcie viac ako 40 dní [33]. Podvod ovplyvňuje aj banky, ktoré poskytujú služby online platieb. Napríklad 20 percent zákazníkov zmení svoje banky po tom, čo zažili podvody.

Už len z dôvodov uvedených vyššie sme sa rozhodli pridať okrem úlohy o športe aj úlohu zo sveta financií. Cieľom našej úlohy bude predikovať hodnotu atribútu Fraud (podvod), čím myslíme binárnu klasifikáciu, pretože atribút Fraud nadobúda boolovské hodnoty True/False. V jednotlivých nástrojoch vytvoríme modely určené na predikciu podvodu.

3.7.2. Import údajov datasetu s finančnými údajmi

Import údajov do všetkých troch nástrojov sa nelíšil od úlohy na datasete futbalových zápasov. V rozhraní Jupyter a v H2O sme nahrali CSV súbor, vo Weka sme dáta museli konvertovať na formát ARFF. Môžeme povedať, že v tomto sú Jupyter s H2O Flow o niečo lepšími nástrojmi, pretože tu odpadol krok s konverziou dát do podporovaného formátu. Kroky importu dát:

- Jupyter: 1. load file
- Weka: 1. konverzia do ARFF 2. load file
- H2O Flow: 1. load file

3.7.3. Analýza údajov

Pri všetkých 3 nástrojoch sa vieme jedným krokom pozrieť, aké sú dáta, ktoré sme nahrali do pamäte, aké atribúty obsahuje a aké hodnoty tieto atribúty nadobúdajú. Výpis v Jupyter obsahuje dáta ako v tabuľkovom editore v Exceli.

```
[3]: print(df.head())
```

	Time	V1	V2	V3	V4	V5	V6	\
0	28755	-30.552380	16.713389	-31.103685	6.534984	-22.105532	-4.977692	
1	28726	-29.876366	16.434525	-30.558697	6.505862	-21.665654	-4.940356	
2	28692	-29.200329	16.155701	-30.013712	6.476731	-21.225810	-4.902997	
3	102572	-28.709229	22.057729	-27.855811	11.845013	-18.983813	6.474115	
4	28658	-28.524268	15.876923	-29.468732	6.447591	-20.786000	-4.865613	

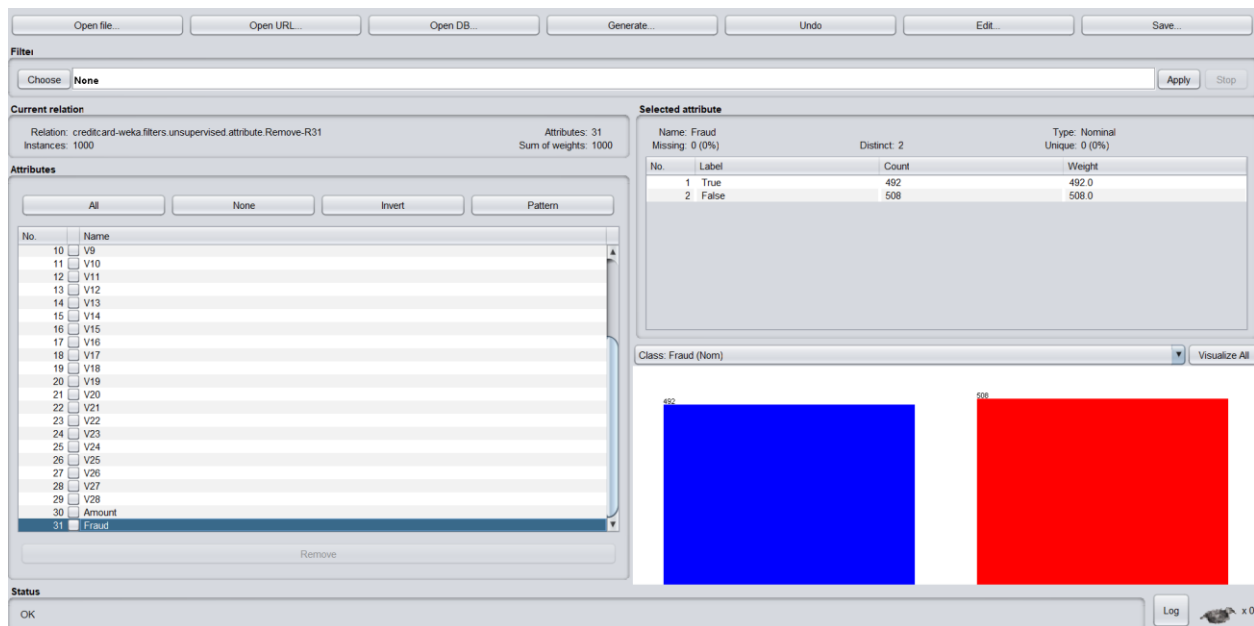
	V7	V8	V9	...	V22	V23	V24	\
0	-20.371514	20.007208	-3.565738	...	-2.288686	-1.460544	0.183179	
1	-20.081391	19.587773	-3.591491	...	-2.232252	-1.412803	0.178731	
2	-19.791248	19.168327	-3.617242	...	-2.175815	-1.365104	0.174286	
3	-43.557242	-41.044261	-13.320155	...	8.316275	5.466230	0.023854	
4	-19.501084	18.748872	-3.642990	...	-2.119376	-1.317450	0.169846	

	V25	V26	V27	V28	Amount	Class	Fraud
0	2.208209	-0.208824	1.232636	0.356660	99.99	1	True
1	2.156042	-0.209385	1.255649	0.364530	99.99	1	True
2	2.103868	-0.209944	1.278681	0.372393	99.99	1	True
3	-1.527145	-0.145225	-5.682338	-0.439134	0.01	1	True
4	2.051687	-0.210502	1.301734	0.380246	99.99	1	True

[5 rows x 32 columns]

Obrázok 83. Výpis funkcie head v Jupyter – dataset údajov o kreditných kartách

Vo Weka v okne Preprocess sa môžeme preklikávať jednotlivými atribútmi, zobrazujú sa tu aj vlastnosti atribútov, počet záznamov a v malom okne aj vizualizácia dát v podobe stĺpcového grafu. V okne Preprocess ešte môžeme vymazať atribúty, s ktorými nechceme nakítniť model. Toto zobrazenie považujeme za jednoduché na pochopenie, je veľmi intuitívne, no na druhej strane dizajn riešenia pôsobí na nás trochu lacno a študentsky.



Obrázok 84. Zobrazenie vo Weka Explorer – dataset údajov o kreditných kartách

V rozhraní H2O vidíme v podstate to isté, čo sme už videli vo Weka s tým rozdielom, že jedným klikom vieme zmeniť dátový typ atribútov.

Search by column name...

1	Time	Numeric ▼	28755	28726	28692	102572	28658
2	V1	Numeric ▼	-30.55238004	-29.87636551	-29.20032859	-28.70922925	-28.52426759
3	V2	Numeric ▼	16.71338924	16.43452455	16.15570143	22.05772899	15.87692299
4	V3	Numeric ▼	-31.10368482	-30.55869682	-30.01371249	-27.85581113	-29.46873209
5	V4	Numeric ▼	6.534983864	6.505861787	6.47673118	11.84501291	6.447591402
6	V5	Numeric ▼	-22.10553152	-21.6656543	-21.22580965	-18.98381258	-20.78600004
7	V6	Numeric ▼	-4.977691964	-4.940356329	-4.902997397	6.474114627	-4.865613418
8	V7	Numeric ▼	-20.37151359	-20.08139107	-19.79124841	-43.55724157	-19.50108408
9	V8	Numeric ▼	20.00720837	19.58777262	19.16832739	-41.04426092	18.74887195
10	V9	Numeric ▼	-3.565737791	-3.591491047	-3.617241786	-13.32015469	-3.642989819
11	V10	Numeric ▼	-7.731071441	-7.800598208	-7.870121943	-24.58826244	-7.939642419
12	V11	Numeric ▼	3.829427395	3.947839717	4.066255073	3.481951803	4.184673689
13	V12	Numeric ▼	-5.314332276	-5.487911486	-5.661492423	-9.128341286	-5.835075215
14	V13	Numeric ▼	1.44693028	1.369940349	1.292950145	-3.008255478	1.215959645
15	V14	Numeric ▼	-4.579263728	-4.829552443	-5.079845681	0.796579962	-5.330143782

← Previous page → Next page

Obrázok 85. Zobrazenie dát v H2O Flow – dataset údajov o kreditných kartách

V základnom zobrazení len ťažko dokážeme poriadne analyzovať zvolený dataset. Weka a H2O do určitej miery už automaticky analyzujú dáta namiesto užívateľa, ale všetko je pred užívateľom skryté. V rozhraní Jupyter vieme niektoré dôležité fakty o datasete získať metódami v Pythone. Zistili sme, že z piatich typov transakcií sa podvody vyskytujú iba v dvoch z nich, a to v TRANSFER, kde sa peniaze posielajú zákazníkovi/podvodníkovi, a CASH_OUT, kde sa peniaze posielajú obchodníkovi, ktorý platí zákazníkovi/podvodníkovi v hotovosti. Je pozoruhodné, že počet podvodných TRANSFER sa takmer rovná počtu podvodných CASH_OUT.

```
print('\n typy platieb, ktoré sú označované ako podvodné {}'.format(\
list(df.loc[df.isFraud == 1].type.drop_duplicates().values))) # iba 'CASH_OUT'
                                                                # & 'TRANSFER'

dfFraudTransfer = df.loc[(df.isFraud == 1) & (df.type == 'TRANSFER')]
dfFraudCashout = df.loc[(df.isFraud == 1) & (df.type == 'CASH_OUT')]

print ('\n počet podvodov typu TRANSFER = {}'.\
      format(len(dfFraudTransfer))) # 260

print ('\n počet podvodov typu CASH_OUT = {}'.\
      format(len(dfFraudCashout))) # 275
```

typy platieb, ktoré sú označované ako podvodné ['TRANSFER', 'CASH_OUT']

počet podvodov typu TRANSFER = 260

počet podvodov typu CASH_OUT = 275

Obrázok 86. Analýza datasetu v Jupyter – dataset údajov o kreditných kartách

Podobnými úvahami sme prišli na to, že pôvod atribútu isFlaggedFraud je nejasný, na rozdiel od uvedeného popisu. Zdá sa, že isFlaggedFraud, nekoreluje s inými atribútmi. Atribút isFlaggedFraud je vraj 1 pri pokuse o TRANSFER, ktorého hodnota je väčšie ako 200 000. V skutočnosti, parameter isFlaggedFraud môže zostať 0, aj keď je táto podmienka splnená. Takýchto úvah by sme mali urobiť čo najviac a Jupyter je v tomto najlepším z vybraných nástrojov, pretože si na naše otázky ohľadom vlastností datasetu vieme sami programovo odpovedať. Na druhej strane, tento typ analýzy vyžaduje od používateľa značnú

dávku skúseností s programovaním a všetky úvahy a fakty o datasete presahujú zadanie tejto práce. Avšak považujeme za dôležité spomenúť túto výhodu riešenia v Jupyter, keďže chceme vybrané nástroje porovnať a prísť na to, ktorý z nich je najlepší.

3.7.4. Preprocessing údajov o kreditných kartách

Aj čo sa týka predspracovania, Jupyter je ideálny nástroj. Avšak konkrétne v tomto prípade, keď si potrebujeme určiť cieľový atribút, rozdeliť si dáta na trénovaciu a testovaciu sadu, sú Weka a H2O oveľa viac užívateľsky priateľskejšie, pretože v rozhraní Jupyter musíme vykonať niekoľko krokov a celému systému okolo preprocessingu musíme správne rozumieť. V prípade, že sme začiatníkmi v oblasti data science, určite najskôr pochopíme, čo je potrebné urobiť vo Weka aj bez väčších znalostí teórie machine learning. V okne Classify nájdeme všetko, čo potrebujeme. V jedinom okne si vyberieme typ učiaceho sa algoritmu, vieme si vybrať, ako dáta rozdelíme na trénovacie a testovacie, vyberieme výstupný atribút, klikneme na štart, a na výstupe hneď vidíme metriky výkonu vytvoreného modelu. Určite najmenej času nám zabral tento jednoduchý preprocessing vo Weka. Čo sa týka H2O Flow, klikneme na tlačidlo Split a v bunke vyplníme rozdelenie datasetu na trénovacia:validačná 80:20.

Selekciu atribútov vykonáme vo všetkých troch nástrojoch ešte pred samotným importom dát, a to manuálne vymazaním stĺpcov (atribútov), ktoré obsahujú kategorické dáta.

3.7.5. Trénovanie a testovanie

Trénovanie a testovanie vo všetkých troch nástrojoch sa nebudeníjak líšiť od krokov, ktoré sme spravili pri datasete futbalových zápasov. Rozdelenie dát na testovaciu a trénovaciu sadu sme boli schopní vykonať vo všetkých spomínaných nástrojoch, takže nám nič nebráni v tom, vytvoriť model. Učiace algoritmy, ktoré sme zvolili, sú:

- Naive Bayes,
- Support Vector Machines.

Na akom princípe tieto algoritmy fungujú, sme si popísali v kapitolách vyššie, a my sa teda môžeme pustiť do tréovania. Na platforme Jupyter použijeme funkcie `fit` a `predict` z knižnice `sklearn`, vo Weka a H2O nám postačí kliknutie na konkrétne tlačidlo.

3.8. Výsledky porovnania nástrojov

Na vyhodnotenie výkonu a presnosti vytvorených modelov použijeme už spomenuté metriky na meranie výkonu machine learning modelov. V kapitolách 3.8.1 a 3.8.2 sme zhrnuli výsledky porovnania v tabuľkovej forme. Pri úlohách sme sa snažili o zachovanie rovnakých vstupných podmienok. Pri riešení úloh sme v každom modeli:

- používali rovnaký dataset,
- dáta rozdeľovali podľa pravidla 80:20,
- nepoužívali cross-validáciu pri testovaní,
- nepoužili vylepšenia nástrojov, ktoré by narúšali objektivnosť meraní.

3.8.1. Metriky výkonu predikcie futbalových zápasov

	Skutočne FTR = A	Skutočne FTR = H	Skutočne FTR = D
Predikované FTR = A	291	17	96
Predikované FTR = H	18	498	134
Predikované FTR = D	75	69	207

Tabuľka 5. Confusion matrix pre Naive Bayes v Jupyter + sklearn

	Skutočne FTR = A	Skutočne FTR = H	Skutočne FTR = D
Predikované FTR = A	290	12	93
Predikované FTR = H	7	506	123
Predikované FTR = D	57	76	241

Tabuľka 6. Confusion matrix pre Naive Bayes vo Weka Explorer

	Skutočne FTR = A	Skutočne FTR = H	Skutočne FTR = D
Predikované FTR = A	290	12	71
Predikované FTR = H	20	556	109
Predikované FTR = D	57	64	208

Tabuľka 7. Confusion matrix pre Naive Bayes v H2O Flow

Jupyter + sklearn	0.7088
Weka Explorer	0.7381
H2O Flow	0.9089

Tabuľka 8. Accuracy score pre Naive Bayes

Jupyter + sklearn	0.73
Weka Explorer	0.756
H2O Flow	0.7367

Tabuľka 9. Precision score pre Naive Bayes

Jupyter + sklearn	0.71
Weka Explorer	0.738
H2O Flow	0.7367

Tabuľka 10. Recall pre Naive Bayes

Jupyter + sklearn	0.72
Weka Explorer	0.745
H2O Flow	0.7367

Tabuľka 11. F1 score pre Naive Bayes

	Skutočne FTR = A	Skutočne FTR = H	Skutočne FTR = D
Predikované FTR = A	402	0	2
Predikované FTR = H	0	650	0
Predikované FTR = D	1	0	350

Tabuľka 12. Confusion matrix pre Support Vector Machines v Jupyter + sklearn

	Skutočne FTR = A	Skutočne FTR = H	Skutočne FTR = D
Predikované FTR = A	352	0	43
Predikované FTR = H	0	595	41
Predikované FTR = D	0	0	374

Tabuľka 13. Confusion matrix pre Support Vector Machines vo Weka Explorer

	Skutočne FTR = A	Skutočne FTR = H	Skutočne FTR = D
Predikované FTR = A	X	X	X
Predikované FTR = H	X	X	X
Predikované FTR = D	X	X	X

Tabuľka 14. Confusion matrix pre Support Vector Machines v H2O Flow

Jupyter + sklearn	0.9979
Weka Explorer	0.9402
H2O Flow	X

Tabuľka 15. Accuracy pre Support Vector Machines

Jupyter + sklearn	1
Weka Explorer	0.951
H2O Flow	X

Tabuľka 16. Precision pre Support Vector Machines

Jupyter + sklearn	1
Weka Explorer	0.940
H2O Flow	X

Tabuľka 17. Recall pre Support Vector Machines

Jupyter + sklearn	1
Weka Explorer	0.942
H2O Flow	X

Tabuľka 18. F1 score pre Support Vector Machines

3.8.2. Vyhodnotenie metrík výkonu predikcie finančných podvodov

	Skutočne Fraud == 1	Skutočne Fraud == 0
Predikované Fraud == 1	62	36
Predikované Fraud == 0	1	101

Tabuľka 19. Confusion matrix pre Naive Bayes v Jupyter + sklearn

	Skutočne Fraud == 1	Skutočne Fraud == 0
Predikované Fraud == 1	79	16
Predikované Fraud == 0	4	101

Tabuľka 20. Confusion matrix pre Naive Bayes vo Weka Explorer

	Skutočne Fraud == 1	Skutočne Fraud == 0
Predikované Fraud == 1	103	1
Predikované Fraud == 0	3	93

Tabuľka 21. Confusion matrix pre Naive Bayes v H2O Flow

Jupyter + sklearn	0.811
Weka Explorer	0.9
H2O Flow	0.965

Tabuľka 22. Accuracy score pre Naive Bayes

Jupyter + sklearn	0.86
Weka Explorer	0.905
H2O Flow	0.98

Tabuľka 23. Precision score pre Naive Bayes

Jupyter + sklearn	0.81
Weka Explorer	0.9
H2O Flow	0.98

Tabuľka 24. Recall pre Naive Bayes

Jupyter + sklearn	0.81
Weka Explorer	0.8999
H2O Flow	0.98

Tabuľka 25. F1 score pre Naive Bayes

	Skutočne Fraud == 1	Skutočne Fraud == 0
Predikované isFraud == 1	98	0
Predikované isFraud == 0	48	54

Tabuľka 26. Confusion matrix pre Support Vector Machines v Jupyter + sklearn

	Skutočne Fraud == 1	Skutočne Fraud == 0
Predikované Fraud == 1	72	23
Predikované Fraud == 0	16	89

Tabuľka 27. Confusion matrix pre Support Vector Machines vo Weka Explorer

	Skutočne Fraud == 1	Skutočne Fraud == 0
Predikované Fraud == 1	94	0
Predikované Fraud == 0	35	69

Tabuľka 28. Confusion matrix pre Support Vector Machines v H2O Flow

Jupyter + sklearn	0.7647
Weka Explorer	0.805
H2O Flow	0.8232

Tabuľka 29. Accuracy pre Support Vector Machines

Jupyter + sklearn	0.84
Weka Explorer	0.806
H2O Flow	0.7287

Tabulka 30. Precision pre Support Vector Machines

Jupyter + sklearn	0.76
Weka Explorer	0.805
H2O Flow	1

Tabulka 31. Recall pre Support Vector Machines

Jupyter + sklearn	0.75
Weka Explorer	0.804
H2O Flow	0.8431

Tabulka 32. F1 score pre Support Vector Machines

Metrika výkonu	Jupyter+Scikit-learn (%)	Weka (%)	H2O (%)
Accuracy/Naive Bayes/PL	70.88	73.81	90.89
Precision/Naive Bayes/PL	73	75.6	73.67
Recall/Naive Bayes/PL	71	73.8	73.67
F1/Naive Bayes/PL	72	74.5	73.67
Accuracy/SVM/PL	99.79	94.2	X
Precision/SVM/PL	100	95.1	X
Recall/SVM/PL	100	94	X
F1/SVM/PL	100	94.2	X
Accuracy/Naive Bayes/FP	81.1	90	96.5
Precision/Naive Bayes/FP	86	90.5	98
Recall/Naive Bayes/FP	81	90	98
F1/Naive Bayes/FP	81	89.99	98
Accuracy/SVM/FP	76.47	80.5	82.32
Precision/SVM/FP	84	80.6	72.87
Recall/SVM/FP	76	80.5	100
F1/SVM/FP	75	80.4	84.31

Tabuľka 33. Poradie výsledkov metrik nástrojov

*PL – Premier League (dataset futbalových zápasov)

*FP – Fraud Prediction (dataset bankových transakcií)

Jupyter+Scikit-learn	20h
Weka	4h
H2O	10h

Tabuľka 34. Cca čas potrebný na dokončenie úloh

Záver a diskusia

Je veľmi náročné povedať, ktorý z nástrojov je najlepší. Každý nástroj je v niečom lepší a v niečom inom horší a všetko závisí od okolností, zadania úlohy a skúseností riešiteľa. Takisto treba brať do úvahy, že sa jedná o open source nástroje, ktoré majú svoje limity a platené riešenia sú určite viac optimalizované a užívateľsky prívetivejšie. V tomto zhrnutí popíšeme niekoľko pohľadov na celé riešenie danej problematiky. V kapitolách 3.8.1 a 3.8.2 sme číselne vyjadrili správnosť vytvorených modelov. Pri vyhodnocovaní vybraných nástrojov sme nebrali do úvahy confusion matrix, pretože táto metrika má skôr informatívny charakter. Tabuľka 33 je pohľad na porovnanie, kde by sme ráтали, ktorý nástroj dosiahol najlepšie výsledky pri metrikách výkonu najviac krát. V tomto prípade by bol víťaz H2O, pretože dosahoval najvyššie čísla, a to aj napriek tomu, že jednu úlohu sme neboli schopní implementovať kvôli technickým nedostatkom tohto nástroja. Dôvodom, prečo H2O dosahoval najvyššie čísla je to, že modely neboli trénované a testované na rovnakom systéme ako Jupyter+Scikit a Weka. H2O používa REST API na pripojenie sa ku klastru, ktorý vykonáva beh programu a poskytuje oveľa väčší výkon ako stolový počítač. Čo sa týka času, ktorý sme museli venovať dokončeniu úloh, víťazom je Weka. Weka vyhráva časom pri inštalácii, naštudovaní, analýze, implementácii a aj testovaní. Čakali sme, že implementácia v H2O bude rýchla, no práve komunikácia so serverom bola problematická, často sa stávalo, že pre problémy so spojením sme museli aplikáciu reštartovať a začínať odznova. Samozrejme, že toto v platenej cloudovej verzii H2O nemusí užívateľ riešiť. Je ťažké porovnávať čas pri naprogramovanom riešení v Jupyter s nástrojmi, kde si užívateľ úlohu vyrieši „pár klikmi“ a to vidíme aj v Tabuľke 34. Myslíme si, že keby sme pred písaním práce nemali skúsenosti s programovaním, riešenie v Jupyter by trvalo oveľa dlhšie. Na druhej strane Jupyter vyniká v transparentnosti riešenia, pretože si ho užívateľ musí naprogramovať. Pri Weka a H2O nevieme, čo sa deje na pozadí, pretože celý proces je skrytý za grafickým užívateľským rozhraním. Weka a H2O sú „čierne skrinky“, v ktorých užívateľ dokáže vyklikať prediktívny model bez toho aby si niečo viac naštudoval o konfiguračných parametroch. Preto sa ľahko môže stať, že neskúsený užívateľ vytvorí nesprávny model, čo môže mať katastrofálne následky. H2O a Weka by sme odporučili začiatocníkom alebo študentom na rýchlejšie pochopenie konceptu machine learning a načerpanie vedomostí o krokoch, ktoré sú nevyhnutné pri vytváraní aplikácii machine learning. Čo sa

týka flexibility jednotlivých nástrojov, Scikit užívateľa nelimituje grafickým používateľským rozhraním. Užívateľ si dokáže vygenerovať funkciu z obrovského množstva metrík výkonu, vizualizácie môžu mať rôzne formy a tvary. V Scikit-learn sú implementované takmer všetky súčasné machine learning algoritmy, a to sa o zvyšných dvoch nástrojoch povedať nedá. Scikit-learn je určite jedným z najpoužívanejších nástrojov na machine learning a z vlastnej skúsenosti môžeme povedať, že pred písaním práce to bol jediný machine learning nástroj spolu s Tensorflow, ktorý sme poznali. Subjektívne teda môžeme povedať, že Scikit-learn je najpopulárnejší zo spomínaných nástrojov, hlavne medzi programátormi. Po napísaní tejto práce by sme nástroj Weka označili ako najlepší nástroj pre ľudí, ktorí napríklad robia nejaký výskum alebo štúdiu, pri ktorej potrebujú machine learning, ale nemajú skúsenosti s programovaním. H2O sa zdá byť najlepším nástrojom pre skupinu ľudí z predošlej vety, a ak užívateľ potrebuje spracovať obrovské množstvo dát v rozumnom čase. Duo Jupyter+Scikit je určený predovšetkým pre programátorov a pre situácie, kedy užívateľ/riešiteľ nechce byť obmedzený grafickým používateľským rozhraním, ale chce naplno využiť všetok potenciál teórie machine learning. Ktorý open source nástroj machine learning je najlepší, to nevie povedať určite nikto. Prínos tejto práce vidíme v tom, že opisuje výhody a nevýhody troch odlišných nástrojov nielen z pohľadu konkrétnej úlohy a algoritmov, ale aj „logistických“ krokov, ktoré sme museli prejsť k vyriešeniu daných úloh. Myslíme si, že táto práca môže poslúžiť aj ako univerzálny návod na vytvorenie klasifikačných modelov v opisovaných nástrojoch. Prešli sme všetky kroky od inštalácie až po výsledkové listiny pri testovaní modelov. Na záver by sme radi povedali, že pri každom nástroji sme sa stretli s pozitívami aj negatívami a pri výbere toho najlepšieho nástroja na machine learning by sme sa riadili predovšetkým zadaním úlohy a zručnosťami riešiteľa.

Zoznam bibliografických odkazov

- [1] ATHEY, S. The Impact of Machine Learning on Economics. National Bureau of Economic. University of Chicago Press. [Online]. Chicago, 2019. ISBN: 978-0-226-61347-5. Dostupné na internete: <http://www.nber.org/chapters/c14009>
- [2] SATHYA, R. - ABRAHAM, A. Comparison of Supervised and Unsupervised. *International Journal of Advanced Research in Artificial Intelligence*. [Online]. [s. 34-38]. Bangalore, 2013. Dostupné na internete: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.278.5274&rep=rep1&type=pdf#page=41>
- [3] SINČÁK, P. - Andrejková, G. Neurónové siete. Technická Univerzita Košice. Univerzita P.J. Šafárika, Košice. [Online]. [Cit. 1.3.2021]. Košice, 1996. Dostupné na internete: <http://kuzo.szm.com/neuronky1.pdf>
- [4] MACHOVÁ, K. Nové trendy v strojovom učení - Štatistický prístup. Katedra kybernetiky a umelej inteligencie Fakulta elektrotechniky a informatiky Technická univerzita v Košiciach. [Online]. [Cit. 2.3.2021]. Košice, 2016. ISBN 978-80-553-2602-3. Dostupné na internete: <http://people.tuke.sk/kristina.machova/pdf/noveSU.pdf>
- [5] IGIRI, C. P. Support Vector Machine–Based Prediction System for a Football. In *IOSR Journal of Computer Engineering*. [Online]. [s. 21-26]. 2015. e-ISSN: 2278-0661. Dostupné na internete: <http://www.iosrjournals.org/iosr-jce/papers/Vol17-issue3/Version-3/D017332126.pdf>
- [6] CIBULA, M. *Definícia*. [Online]. [Cit. 1.5.2021]. 2017. Dostupné na internete: https://smnd.sk/mcibula/zakl_info/definicia.html
- [7] WILSON, A. *A Brief Introduction to Supervised Learning*. [Online]. 2019. Dostupné na internete: <https://towardsdatascience.com/a-brief-introduction-to-supervised-learning-54a3e3932590>.
- [8] CIBULA, M. *Support Vector Machines*. [Online]. [Cit. 1.5.2021]. 2017. Dostupné na internete: <https://smnd.sk/mcibula/alg/SVM.html>.

- [9] FTÁČNIK, M. *Lineárny klasifikátor a SVM*. [Online]. [Cit. 1.5.2021]. 2019. Dostupné na internete: <http://www.sccg.sk/~ftacnik/RO6-linearny%20klasifikator-2019-20.pdf>
- [10] TINÁKOVÁ, B. *Základy Support Vector Machines*. [Online]. [Cit. 1.5.2021]. 2019. Dostupné na internete: <https://umelainteligencia.sk/zaklady-support-vector-machines/>.
- [11] *Thomas Bayes*. [Online]. 2011. Dostupné na internete: http://en.wikipedia.org/wiki/Thomas_Bayes.
- [12] HAN, J. - KAMBER, M. *Data Mining: Concepts and Techniques*. [Online]. San Francisco: Morgan Kaufmann Publisher, 2003. ISBN: 1-55860-901-6.
- [13] KAŠČÁK, P. *Metody pro klasifikaci dat: Bakalárska práca*. [Online]. Brno: FIT VUT v Brne, 2010. Dostupné na internete: <http://www.fit.vutbr.cz/study/DP/rpfile.php?id=10647>
- [14] RAJ, S. *How to Evaluate the Performance of Your Machine Learning Model*. [Online]. Indian Institute of Technology Jammu, 2020. Dostupné na internete: <https://www.kdnuggets.com/2020/09/performance-machine-learning-model.html>
- [15] LIU, S. *Artificial intelligence software market revenue worldwide 2018-2025*. [Online]. 2020. Dostupné na internete: <https://www.statista.com/statistics/607716/worldwide-artificial-intelligence-market-revenues/>
- [16] RAO, A. *PwC analysis*. [Online]. 2017. Dostupné na internete: <https://www.pwc.com/gx/en/issues/data-and-analytics/publications/artificial-intelligence-study.html>
- [17] CORNAPEAU, D. *scikit-learn*. [Online]. 2007. Dostupné na internete: <https://github.com/scikit-learn/scikit-learn>
- [18] *Weka (machine learning)*. [Online]. 2021. Dostupné na internete: [https://en.wikipedia.org/wiki/Weka_\(machine_learning\)](https://en.wikipedia.org/wiki/Weka_(machine_learning))

- [19] H2O, AI. *What is H2O?* [Online]. 2021. Dostupné na internete: <https://docs.h2o.ai/h2o-tutorials/latest-stable/WhatIsH2O.html>.
- [20] *Prediktívne modelovanie (Predictive modelling)*. [Online]. 2021. Dostupné na internete: <https://investopedia.sk/2021/03/05/prediktivne-modelovanie-predictive-modelling/>
- [21] BROWN, M. S. *What IT Needs To Know About The Data Mining Process*. [Online]. 2015. Dostupné na internete: <https://www.forbes.com/sites/metabrown/2015/07/29/what-it-needs-to-know-about-the-data-mining-process/?sh=5a971119515f>.
- [22] *Štandardizácia dát (Data Standardization)*. [Online]. 2018. Dostupné na internete: <https://managementmania.com/sk/standardizacia-dat-data-standardization>.
- [23] BROWNLEE, J. *How to Perform Feature Selection With Machine Learning Data in Weka*. [Online]. 2019. Dostupné na internete: <https://machinelearningmastery.com/perform-feature-selection-machine-learning-data-weka/>.
- [24] *Class LibSVM*. [Online]. 2021. Dostupné na internete: <https://weka.sourceforge.io/doc.stable/weka/classifiers/functions/LibSVM.html>.
- [25] *SMO*. [Online]. 2021. Dostupné na internete: <https://weka.sourceforge.io/doc.stable/weka/classifiers/functions/SMO.html>.
- [26] *LIBSVM*. [Online]. 2021. Dostupné na internete: <https://en.wikipedia.org/wiki/LIBSVM>.
- [27] H2O, AI, *Cloud Integration*. [Online]. 2021. Dostupné na internete: <https://docs.h2o.ai/h2o/latest-stable/h2o-docs/cloud-integration.html>.
- [28] H2O, AI. *H2O Clients*. [Online]. 2021. Dostupné na internete: <https://docs.h2o.ai/h2o/latest-stable/h2o-docs/h2o-client.html>.

- [29] H2O, AI. *Using Flow - H2O's Web UI*. [Online]. 2021. Dostupné na internete: <https://docs.h2o.ai/h2o/latest-stable/h2o-docs/flow.html>.
- [30] H2O, AI. *Support Vector Machine (SVM)*. [Online]. 2021. Dostupné na internete: <https://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/svm.html>.
- [31] H2O, AI. *AutoML*. [Online]. 2021. Dostupné na internete: <https://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html>.
- [32] M.L.G. - ULB. *Credit Card Fraud Detection*. [Online]. 2018. Dostupné na internete: <https://www.kaggle.com/mlg-ulb/creditcardfraud>.
- [33] JAVELIN. Identity Fraud Study. In *J. S. & Research*. [Online]. 2017. Dostupné na internete: <https://www.javelinstrategy.com/sites/default/files/17-1001J-2017-LL-Identity-Fraud-Hits-Record-Highs-Javelin.pdf>.