

**Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky**

Softvér na odhad trhovej ceny nehnuteľností

Bakalárska práca

2022

Jakub Magáč

**Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky**

Softvér na odhad trhovej ceny nehnuteľností

Bakalárska práca

Študijný program: Informatika
Študijný odbor: 9.2.1. Informatika
Školiace pracovisko: Katedra počítačov a informatiky (KPI)
Školiteľ: Ing. Marcel Vološin, PhD.
Konzultant:

Košice 2022

Jakub Magáč

Abstrakt v SJ

Táto práca sa venuje predpovedi cien nehnuteľnosti. Jej cieľom je vytvoriť čo najpresnejší softvér pre odhadovanie cien s pomocou strojového učenia. Jej obsahom je taktiež vytvorenie softvéru pre zber dát z internetu a tvorba datasetu pre trénovanie modelu strojového učenia. Ďalej zahŕňa analýzu vytvoreného datasetu pomocou grafov, trénovanie modelov a analýza ich presnosti. V neposlednom rade zahŕňa vytvorenie Systémovej a Používateľskej príručky pre obsluhu softvéru.

Kľúčové slová v SJ

strojové učenie, regresia, ceny, nehnuteľnosti

Abstrakt v AJ

This thesis is about predicting real estate prices. The goal of this thesis is to create the most accurate software for predicting prices with help of machine learning. The part of the thesis is also creating software for collection of datas from internet and creation of dataset, that will be used for training the machine learning model. It also consist of analysis of created dataset with help of graphs, training a model and analysis of model accuracy. Last but not least it consist of creating System and User guide for software service.

Kľúčové slová v AJ

machine learning, regression, prices, real estate

Bibliografická citácia

MAGÁČ, Jakub. *Softvér na odhad trhovej ceny nehnuteľností*. Košice: Technická univerzita v Košiciach, Fakulta elektrotechniky a informatiky, 2022. 40s. Vedúci práce: Ing. Marcel Vološin, PhD.

TECHNICKÁ UNIVERZITA V KOŠICIACH
FAKULTA ELEKTROTECHNIKY A INFORMATIKY
Katedra počítačov a informatiky

ZADANIE
BAKALÁRSKEJ PRÁCE

Študijný odbor: **Informatika**
Študijný program: **Informatika**

Názov práce:

Softvér na odhad trhovej ceny nehnuteľností
Real estate market price estimation software

Študent: **Jakub Magáč**
Školiteľ: **Ing. Marcel Vološin, PhD.**
Školiace pracovisko: **Katedra počítačov a informatiky**
Konzultant práce:
Pracovisko konzultanta:

Pokyny na vypracovanie bakalárskej práce:

1. Vytvoriť nástroj na tvorbu datasetu vhodného na odhad trhovej ceny nehnuteľností.
2. Analyzovať a formou grafov znázorniť vlastnosti získaných dátových vzoriek.
3. Implementovať softvérové riešenie využívajúce strojové učenie na odhad trhovej ceny nehnuteľností.
4. Vhodnými metódami analyzovať presnosť vytvoreného riešenia.
5. V rámci zadania vypracovať používateľskú a systémovú príručku.

Jazyk, v ktorom sa práca vypracuje: slovenský
Termín pre odovzdanie práce: 31.08.2022
Dátum zadania bakalárskej práce: 29.10.2021



prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty

Čestné vyhlásenie

Vyhlasujem, že som záverečnú prácu vypracoval(a) samostatne s použitím uvedenej odbornej literatúry.

Košice, 31.8.2022

.....

Vlastnoručný podpis

Podakovanie

Na tomto mieste by som rád poďakoval svojmu vedúcemu práce za jeho čas a odborné vedenie počas riešenia mojej záverečnej práce. Rovnako by som sa rád poďakoval svojim rodičom a priateľom za ich podporu a povzbudzovanie počas celého môjho štúdia. V neposlednom rade by som sa rád poďakoval pánom *Donaldovi E. Knuthovi* a *Leslie Lamportovi* za typografický systém \LaTeX , s ktorým som strávil množstvo nezabudnuteľných večerov.

Obsah

Úvod	1
1 Ceny nehnuteľností	3
2 Strojové učenie	5
2.1 Typy strojového učenia	5
2.1.1 Učenie s učiteľom	5
2.1.2 Učenie bez učiteľa	6
2.1.3 Učenie s posilňovaním	6
2.2 Proces strojového učenia	7
2.2.1 Zber dát	7
2.2.2 Oboznámenie sa s dátami	7
2.2.3 Predspracovanie dát	7
2.2.4 Výber modelu	8
2.2.5 Validácia modelu	11
2.2.6 Nasadenie na produkciu	13
3 Regresné modely	15
3.1 Lineárna regresia	15
3.2 Hrebeňová regresia	16
3.3 LASSO	16
3.4 Elastická sieť	17
3.5 Náhodné lesy	17
3.6 Metódy podporných vektorov	18
4 Implementácia riešenia	20
4.1 Nástroj na tvorbu datasetu	20
4.1.1 Beautiful Soup	20
4.1.2 Tvorba datasetov	21
4.1.3 Dátové sady - predstavenie vlastností	21

4.2	Filtrácia dátových vzoriek	22
4.2.1	Nepotrebné atribúty	23
4.2.2	Vzorky nepochádzajúce zo Slovenska	23
4.2.3	Odláhlé hodnoty	24
4.3	Vizualizácia dát	25
4.4	Softvér na odhad trhovej ceny nehnuteľností	28
4.4.1	Predspracovanie dát	28
4.4.2	Trénovanie modelov	30
4.4.3	Ladenie hyperparametrov	30
4.4.4	Predikcia nových hodnot	31
5	Analýza riešenia	32
5.1	Porovnanie datasetov pred a po filtrácii dát	32
5.2	Analýza datasetu	33
5.2.1	Typy nehnuteľností	33
5.2.2	Stavy nehnuteľností	33
5.2.3	Materiály nehnuteľností	33
5.3	Modely bez ladenia hyperparametrov	34
5.4	Ladenie hyperparametrov	36
5.4.1	Dataset apartmans	36
5.4.2	Dataset houses	37
5.4.3	Výsledný model	37
6	Záver	38
	Literatúra	39
	Zoznam príloh	41
A	Používateľská príručka	42
A.1	Skript create_dataset	42
A.2	Skript get_all_cities	42
A.3	Skript main	42
B	Systémová príručka	44
B.1	Skript create_dataset	44
B.2	Skript main	44
B.2.1	Prvá bunka	44
B.2.2	Druhá bunka	45

B.2.3	Tretia bunka	45
B.2.4	Štvrtá bunka	46
B.2.5	Piata bunka	46
B.2.6	Šiesta bunka	47
B.2.7	Siedma bunka	48
B.2.8	Osma bunka	48
B.2.9	Deviata bunka	48
B.2.10	Desiata bunka	49
B.2.11	Jedenásta bunka	49
B.2.12	Dvanásta bunka	50
B.2.13	Trinásta bunka	52
B.2.14	Štrnásta bunka	52
B.2.15	Petnásta bunka	53
B.2.16	Šetstnásta bunka	54
B.2.17	Sedemnásta bunka	54
B.2.18	Osemnásta bunka	55
B.2.19	Devetnásta bunka	55
B.2.20	Dvadsať bunka	56

Zoznam obrázkov

1.1	Vývoj cien nehnuteľností na Slovensku za posledných 53 týždňov[3]	3
2.1	Rozdiel medzi klasifikáciou a regresiou[6]	6
2.2	Príklad použitia metódy One Hot Encoding[7]	8
2.3	Rozdiel medzi pretrénovaným, podtrénovaným a dobre natrénovaným modelom[9]	9
2.4	Rozdelenie dátovej sady pri k-zložkovej krížovej validácii[10]	11
3.1	Lineárna regresia[13]	16
3.2	Príklad náhodných lesov[14]	18
3.3	Metóda podporných vektorov[15]	18
4.1	Výsledný dataset	21
4.2	Histogram cien nehnuteľností	25
4.3	Porovnanie cien a typov nehnuteľností	26
4.4	Porovnanie cien a stavov nehnuteľností	27
4.5	Porovnanie cien a materiálov nehnuteľností	27
4.6	Korelačná matica atribútov datasetu byty	28
4.7	Pipeline s regresorom ElasticNet	29
5.1	Vizualizácia zastúpenia typov nehnuteľností	33
5.2	Vizualizácia zastúpenia stavov v ktorých sa nachádzali nehnuteľnosti z datasetov	34
5.3	Vizualizácia zastúpenia materiálov z ktorých boli nehnuteľnosti postavené	34

Zoznam tabuliek

5.1	Dáta pred filtráciou	32
5.2	Dáta po filtrácii	32
5.3	Výsledky základných modelov pre dataset apartmans	35
5.4	Výsledky základných modelov pre dataset houses	35
5.5	Výsledky základných modelov pre dataset combined	35
5.6	Výsledky ladenia hyperparametrov na datasete apartmans	36
5.7	Výsledky ladenia hyperparametrov na datasete houses	37

Úvod

Priemerná cena nehnuteľností v roku 2002 na m^2 bola 592€, pričom v roku 2022 to je 2510€. Čo je viac ako 4-násobný nárast ceny za 20 rokov[1]. Hypotekárny úver je pre väčšinu ľudí ich najväčší dlh, ktorý na seba dobrovoľne berú. Ide o investíciu, ktorá značne ovplyvní ich život na niekoľko rokov. Preto by mal byť výber nehnuteľnosti dobre zvážený.

Strojové učenie je oblasť informatiky, ktorá sa špecializuje na učenie počítačov myslieť viac ako ľudia. Medzi problémy, ktoré vďaka strojovému učeniu môžeme riešiť, patrí napríklad rozpoznávanie reči a obrazov. Firmy, ako napríklad Amazon, ho používajú pre odporúčanie tovarov, ktoré by mohli ich zákazníka zaujímať. Využitie má strojové učenie pri autonómnych vozidlách, v robotike alebo pre predikcie a zmeny cien.

Práve pre posledný bod je strojové učenie skvelý nástroj, vďaka ktorému vieme riešiť tento problém, bez toho, aby sme si zháňali odborníka na predaj nehnuteľností. Strojové učenie dokáže nájsť spojitosti medzi vlastnosťami domov a ich cenou omnoho lepšie, ako by sme to spravili my prezeraním a porovnávaním ponúk a zároveň to dokáže spraviť s lepšou presnosťou.

Cieľom tejto práce je vytvoriť čo najpresnejší model a porovnať jeho presnosť so skutočnými hodnotami.

Formulácia úlohy

Hlavnou náplňou tejto práce bude vytvoriť softvér pre predikciu cien nehnuteľností. Popri tejto hlavnej úlohe budeme riešiť viacero podúloh.

1. Tvorba datasetu.

Predtým, než začneme s tvorbou softvérového riešenia pre predikciu cien, potrebujeme najprv dataset s hodnotami, na ktorých budeme náš model trénovať.

2. Analyzovať a znázorniť vlastnosti získaných dátových vzoriek.

Dataset, ktorý sme vytvorili, najprv analyzujeme, pozrieme sa na jednotlivé vzorky, pokúsime sa nájsť nejaké spojitosti medzi nimi a znázorníme ich pomocou grafov.

3. Implementovať softvérové riešenie využívajúce strojové učenie na odhad trhovej ceny nehnuteľnosti.

Pokúsime sa nájsť čo najpresnejší model pre náš problém, porovnáme výkony viacerých modelov a ich hyperparametrov.

4. Analyzujte presnosť vytvoreného riešenia.

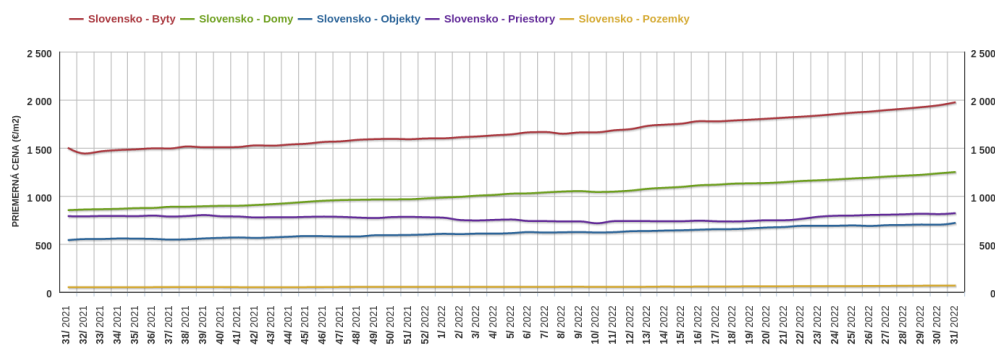
Pomocou viacerých metrík porovnáme výkon nášho modelu, ako je na tom s presnosťou voči skutočným cenám bytov a domov.

5. Vypracovanie používateľskej a systémovej príručky.

Používateľská príručka bude obsahovať pokyny a možnosti spúšťania skriptov s modelom strojovho učnia. Systémová príručka bude určená programátorom a bude obsahovať zoznam použitých funkcií a algoritmov pre prípad, ak by sa nejaký programátor rozhodol pokračovať v ďalšom vývoji. Táto príručka bude slúžiť ako dokumentácia projektu.

1 Ceny nehnuteľností

Určite ne jeden z nás v poslednej dobe postrehol správy o tom, aká je vysoká inflácia. Zdražovanie je najmä vidieť na cene pohonných palív, stavebných materiálov a nehnuteľností. Vývoj cien nehnuteľností má na Slovensku stúpajúci trend už od roku 2014, kedy bol naposledy v jednotlivých kvartáloch roka zaznamenaný pokles[2].



Obr. 1.1: Vývoj cien nehnuteľností na Slovensku za posledných 53 týždňov[3]

Jeden z dôvodov zvýšeného dopytu po nehnuteľnostiach je, že za posledných pár rokov výrazne klesali úrokové sadzby hypoték na slovenskom trhu[4]. Aktuálne sa začína pomaly zvyšovať, avšak len budúcnosť ukáže, či to bude mať pozitívny alebo negatívny dopad na dopyt po nehnuteľnostiach. Taktiež musíme vziať do úvahy aktuálnu situáciu na Ukrajine, ktorá určite bude mať za následok zvýšenie dopytu po nehnuteľnostiach a tým pádom aj zvýšenie ich ceny.

Pre rýchlu predikciu cien nehnuteľností existuje aktuálne na trhu niekoľko kalkulačiek. Príkladom môžu byť webové aplikácie ako Aktualna cena bytu¹, ďalšiu kalkulačku vlastní Inštitút finančnej politiky,² tretiu vlastní realitná kancelária HALO reality³ a štvrtý portál nehnuteľnosti.sk⁴. Čo majú tieto kalkulačky spoločné je, že každá z nich pozerá pri výpočte na podobné vlastnosti. Základom všet-

¹<https://www.aktualnacenabytu.sk/predatbyt/kalkulacka>

²<http://www.institutfinancnejpolitiky.sk/kalkulacky/nehnutelnosti/>

³<https://www.haloreality.sk/vypocet-ceny-nehnutelnosti-ocenovanie-kalkulacka>

⁴<https://www.nehnuteľnosti.sk/ocenovanie-nehnutelnosti/>

kých kalkulačiek je mesto, v ktorom sa nehnuteľnosť nachádza, ulica, počet izieb, stav nehnuteľnosti, plocha nehnuteľnosti a plocha pozemku. Všetky spomenuté kalkulačky sú voľne dostupné. Takáto predikcia však nemusí zodpovedať realite.

Pre odhad ceny pri žiadosti o hypotekárny úver sa preto používa znalecký posudok. Znalecký posudok je jeden z najdôležitejších dokumentov, ktoré potrebujeme, ak žiadame o hypotéku. Práve cena uvedená na znaleckom posudku je cena, z ktorej vám bude banka vypočítavať výšku poskytnutého úveru, aj keď kúpna cena nehnuteľnosti sa s ňou nemusí zhodovať. Cena znaleckého posudku sa môže pohybovať od 100 do 350€. Jeho cenu ovplyvňuje typ nehnuteľnosti, lokalita aj čas, za aký má byť vypracovaný. Takýto posudok môže vypracovať iba znalec, zapísaný v zozname znalcov Ministerstva spravodlivosti Slovenskej republiky, v potrebnom odbore.

Ceny nehnuteľností môžeme taktiež sledovať pomocou portálov nehnuteľnosti.sk,⁵ poprípade na webe Národnej banky Slovenska⁶. V oboch prípadoch sú ceny uvedené na m^2 . Portál nehnuteľnosti.sk nám ponúka porovnanie vývoja cien podľa typu nehnuteľnosti a zároveň ponúka filtráciu podľa oblasti Slovenska (na webe sú uvedené aj iné krajiny, avšak pre väčšinu z nich nedokáže aplikácia zobrazíť údaje). Všetky ceny a ich rast sú vizualizované pomocou grafov. Na webe Národnej banky Slovenska zase vieme odsledovať priemerné hodnoty cien rozdelené podľa kvartálu roka a okresu Slovenska. Ceny sú však uvádzané iba v tabuľke.

⁵<https://www.nehnuteľnosti.sk/ceny/>

⁶<https://nbs.sk/statisticke-udaje/vybrane-makroekonomicke-ukazovatele/ceny-nehnuteľnosti-na-byvanie/ceny-nehnuteľnosti-na-byvanie-podla-krajov/>

2 Strojové učenie

Pre riešenie nášho problému odhadu cien budeme používať strojové učenie. Strojové učenie sa v dnešnej dobe využíva vo veľkom množstvo, odvetví. Podľa definície priekopníka strojového učenia Toma Mitchela je strojové učenie oblasťou štúdia počítačových algoritmov, ktoré sa zlepšujú automaticky na základe skúseností[5].

Strojové učenie je použitie algoritmu, ktorý sa snaží nájsť vzory v dátach a potom použitie takto naučeného algoritmu na podobných dátach s cieľom vytvoriť predikciu pre nové dáta.

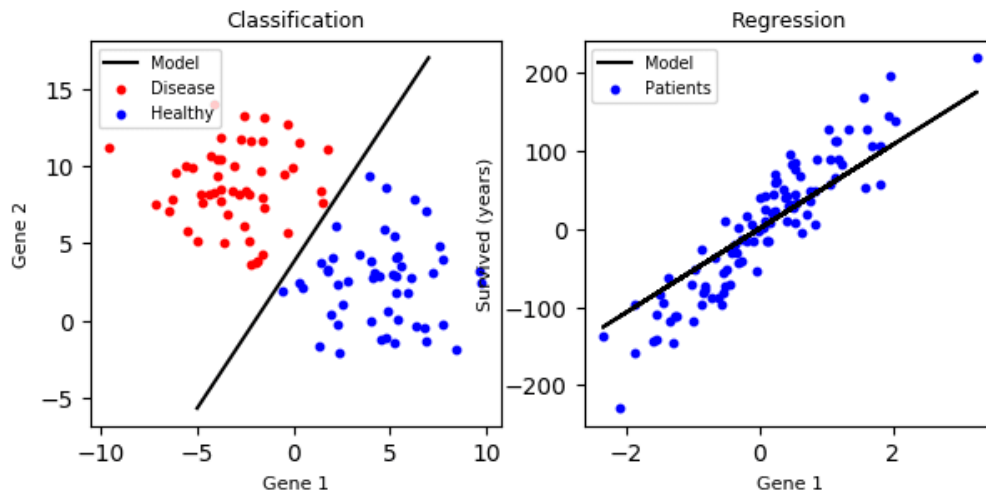
2.1 Typy strojového učenia

Strojové učenie môžeme rozdeliť na tri základné časti: učenie s učiteľom, učenie bez učiteľa a učenie posilňovaním.

2.1.1 Učenie s učiteľom

Forma strojového učenia, pri ktorom sú naše dáta označené. Naše namerané X hodnoty majú svoje y cieľové hodnoty. Napríklad klasický prípad môže byť rozpoznávanie číslíc. Majme sadu obrázkov s ručne písanými číslcami a každá vzorka má svoju triedu, do ktorej patrí (jednotka, dvojka...). Takto rozdelené dáta môžeme použiť pre naučenie modelu rozpoznávať čísllice.

Učenie s učiteľom môžeme rozdeliť na klasifikáciu (odhad triedy, do ktorej má výsledok patriť) a regresiu (odhad súvislej číselnej hodnoty). Príkladom klasifikácie by mohol byť problém, kde by sme odhadovali na vzorke pacientov pravdepodobnosť, že pacient trpí srdcovo cievnyými chorobami a príkladom pre regresiu je napríklad naša téma, predikcia cien nehnuteľností.



Obr. 2.1: Rozdiel medzi klasifikáciou a regresiou[6]

2.1.2 Učenie bez učiteľa

Prípad, kedy naše dáta nemajú určenú triedu, ktorú potrebujeme predikovať. Algoritmus musí najstť triedy sám, hľadá medzi dátami súvislosti. Tento typ učenia najčastejšie predstavuje klastrovanie a redukcia dimenzionality.

Klastrovanie je zhlukovanie vzoriek do skupín-klastrov. Príkladom algoritmu využívajúceho klastrovanie môže byť napríklad algoritmus K-najbližších susedov. Využitie takéhoto algoritmu môže byť pre model, ktorý sa bude snažiť odhadovať o aký tovar by mal záujem návštevník obchodu, na základe jeho predošlých nákupov a prezretých tovarov, pri odhadovaní či pacient trpí srdcovo-cievnyimi ochoreniami alebo pre mailový spam filter.

Redukciu dimenzionality by sme zase použili, ak by sme mali v datasete veľa vlastností. Metóda vytvára nízkorozmernú sadu z vysokorozmernej, komprimáciou pôvodnej sady. Príkladom algoritmu využívajúceho redukciu dimenzionality je napríklad PCA algoritmus.

2.1.3 Učenie s posilňovaním

V prípade učenia s posilňovaním je učenie vykonávané formou spätnej väzby, kde v prípade správnej predikcie je algoritmus odmenený a v prípade nesprávnej potrestaný. Príkladom problému, ktorý môžeme riešiť pomocou učenia posilňovaním, je bludisko, kde sa hráč má dostať do cieľa a počas toho sa vyhýbať prekážkam. Každý správny krok hráča bude odmenený bodom a každý nesprávny bude potrestaný strhnutím bodu. Následne sa vezme celková hodnota každého pokusu a vyberie sa tá s najlepšou odmenou.

2.2 Proces strojového učenia

Strojové učenie vieme rozdeliť na viacero častí. Jeho postup však nie je lineárny. Častokrát sa stane, že preskakujeme medzi jednotlivými krokmi, napríklad spravíme experiment s jedným typom modelu, následne odskúšame ako dobre model funguje a následne skúsime znova iný model.

2.2.1 Zber dát

V praxi častokrát dostane inžinier strojového učenia už pripravené dáta, ak firma zbierala dáta od zákazníkov alebo dostala výsledky prieskumu. Ak však dáta nevlastníme, musíme si ich zohnať. Existuje veľa voľne dostupných datasetov, ktoré vieme použiť. Ak však nenájdeme taký, ktorý by nám práve vyhovoval, vieme dáta zozbierať pomocou web scrappera, ktorý nám zozbiera dáta z webu. Na trhu existuje viacero produktov, ktoré ponúkajú web scraping, avšak ich použitie môže byť spoplatnené. Príkladom môže byť produkt Web Scrapper¹, ktorý taktiež ponúka verziu zadarmo vo forme rozšírenia do webového prehliadača. Vytvorenie web scraperu však nemusí byť komplikovaná záležitosť a s trochou znalostí programovania si vieme pomocou knižníc ako BeautifulSoup pre Python alebo Cheerio pre JavaScript naprogramovať vlastnú implementáciu web scraperu.

2.2.2 Oboznámenie sa s dátami

Keď máme dáta zozbierané, mali by sme sa s nimi oboznámiť. Dáta vieme preskúmať pred ich spracovaním pomocou knižníc pandas a matplotlib. Pričom nás zaujíma hlavne koľko a aké vlastnosti majú naše dáta, koľko dát máme, či sú všetky z nich použiteľné. Zaujímajú nás vzťahy medzi jednotlivými vlastnosťami. Táto časť procesu strojového učenia slúži hlavne pre nás na pochopenie súvislosti, aby sme sa oboznámili s dátami, s ktorými budeme pracovať. Pre jednoduchšie preskúmanie dát môžeme dáta vizualizovať pomocou grafov.

2.2.3 Predspracovanie dát

Ďalšou časťou procesu je predspracovanie dát. Zo zozbieraných dát musíme vybrať dáta, ktoré sú pre nás relevantné. Dáta, ktoré sú pre nás irelevantné odstránime. Taktiež sa musíme uistiť, že nemáme v datasete chýbajúce hodnoty, pretože s takýmito dátmi nevie model pracovať. Takto chýbajúce hodnoty vieme buď

¹<https://webscraper.io/>

zmazať, poprípade ich doplniť inými (imputácia), napríklad priemernou hodnotou vlastnosti všetkých vzoriek.

Ďalej musíme premeniť znakové hodnoty na číselné, pričom musíme zohľadniť rozdiely medzi ordinálnymi a nominálnymi vlastnosťami. Ordinálne vlastnosti sú merateľné hodnoty, ktoré vieme zobraziť na nejakej škále. Medzi takéto vlastnosti môže patriť hodnotenie študentov. Študent, ktorý dostal z predmetu A bude lepší ako študent, ktorý z neho dostal B. Naopak vlastnosť, ako napríklad farba auta, nedokážeme zobraziť na škále. Červené auto nie je nijako nadradené zelenému auto. Takéto vlastnosti sa taktiež nazývajú kategorické a pre ich spracovanie môžeme použiť One Hot Encoder z knižnice scikit-learn alebo dummies pomocou knižnice pandas.

Label Encoding			One Hot Encoding			
Food Name	Categorical #	Calories				
Apple	1	95				
Chicken	2	231				
Broccoli	3	50				

→

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

Obr. 2.2: Príklad použitia metódy One Hot Encoding[7]

Ak máme veľký nepomer vo veľkosti číselných atribútov, musíme využiť škálovanie. Takýto nepomer môže spôsobiť, že výsledok bude ovplyvnený vlastnosťami, ktoré sú iného rozsahu ako tie ostatné. Pre nápravu tohto problému môžeme využiť napríklad StandardScaler, MinMaxScaler alebo MaxAbsScaler z knižnice scikit learn.

A v neposlednom rade musíme rozdeliť naše dáta na trénovacie, testovacie a validačné. Trénovacie dáta budú slúžiť na trénovanie modelu, testovacie budeme používať pre jeho otestovanie a validačné dáta budú použité pre korekciu hyperparametrov. Nesmie nastať situácia, pri ktorej použijeme časť jednej skupiny dát na inú skupinu. Ak by sme použili testovacie dáta na trénovanie, náš model by mal vyššiu presnosť pri testovaní, avšak ak by sme mu dali dáta, ktoré ešte nevidel, jeho presnosť by bola nižšia ako tá, ktorú sme namerali pri testovaní.

2.2.4 Výber modelu

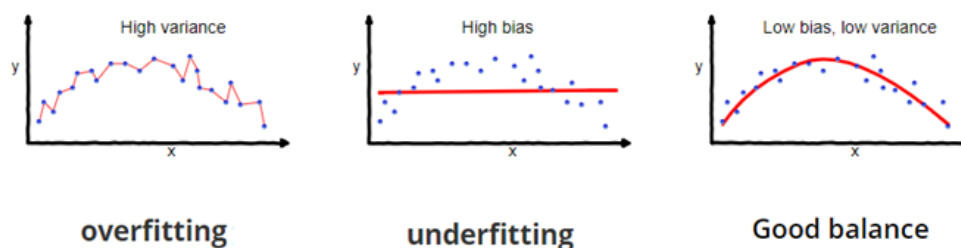
Keď máme spravené predspracovanie dát, môžeme sa pustiť do výberu modelu. Je ideálne si vyskúšať viacero možných algoritmov a zároveň aj hyperparametrov.

Pre výber modelu si môžeme pozrieť dokumentáciu a na základe nášho problému vybrať vhodný algoritmus. Pre zjednodušenie výberu algoritmu vytvorili v scikit-learn knižnici cheat-sheet². Po výbere vhodného algoritmu je čas natrénovať model na tréningových dátach. Trénovanie modelu môže trvať dlhú dobu a jeho čas sa bude zvyšovať v závislosti na koľkých dátach model trénujeme. Preto je vhodné trénovať model na menšom objeme dát, odskúšať viacero algoritmov a zistiť, ktorý z nich je najefektívnejší. Ak už máme ten najefektívnejší nájdený, môžeme model následne natrénovať na väčšom objeme dát.

Trénovanie modelu je činnosť, pri ktorej sa vezmu inicializačné hodnoty algoritmu a následne prebiehajú iterácie, počas ktorých sa presnosť algoritmu zlepšuje.

Pretrénovanie a podtrénovanie sú dva z najčastejších príčin slabej presnosti modelu. Podtrénovanie nastane, ak model nedokáže nájsť súvislosti medzi atribútmi. Tento problém nastáva, ak je model príliš jednoduchý.

Dôvody podtrénovania môžu byť, že máme nedostatok dát pre predikciu, náš model potrebuje menej regularizácie alebo potrebuje viac času na tréning[8]. Či už je to nedostatok vzoriek alebo málo atribútov. Pretrénovanie nastane naopak, ak model predikuje dáta až príliš dobre, nesprávne zovšeobecňuje. Tento problém môže nastať, ak nemáme správne rozdelené dáta na tréningové a testovacie a časť našich testovacích dát sa dostane do styku s tréningovými.



Obr. 2.3: Rozdiel medzi pretrénovaným, podtrénovaným a dobre natrénovaným modelom[9]

Čo sa týka výberu hyperparametrov, neexistuje tam osvedčený spôsob ako vybrať najlepšie. Jediný spôsob pre nás je otestovať jednotlivé parametre a otestovať, ktoré nastavenia boli pre algoritmus najvhodnejšie. Vieme to dosiahnuť viacerými spôsobmi.

Prvý, najjednoduchší, avšak zároveň nejneefektívnejší spôsob by bol ručné

²https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

dosadzovanie parametrov a následné testovanie modelu. Tento proces je však zdĺhavý, zároveň si musíme niekam zapisovať výsledky a použité nastavenia.

Ak by sme chceli ešte lepšie otestovanie parametrov, môžeme použiť funkciu `GridSearchCV` knižnice `scikit-learn`. Táto funkcia knižnice `scikit-learn` nám umožňuje vytvoriť Pythonovský slovník, do ktorého uložíme parametre, ktorých kombináciu chceme skúsiť na našom klasifikátore. Funkcia následne bude prechádzať celým slovníkom a skúšať všetky kombinácie parametrov (brute force metóda). Aj keď `GridSearchCV` nám dá najpresnejší výsledok, proces hľadania parametru je veľmi zdĺhavý. Ak potrebujeme rýchlejšie vyhľadávanie ako `Grid Search` a nevadí nám, že nenájde najpresnejší parameter, môžeme použiť funkciu `RandomizedSearchCV`.

Oficiálna dokumentácia `scikit-learn` popisuje funkciu `RandomizedSearchCV` následovne. V kontraste k `GridSearchCV`, nie všetky hodnoty parametrov sú vyskúšané, namiesto toho sa vyberie fixný počet nastavení parametrov zo špecifikovanej distribúcie nastavení. Počet nastavení parametrov, ktoré sú vyskúšané je udaný parametrom `n_iter`[10].

Zdrojový kód 2.1: Príklad použitia funkcie `RandomizedSearchCV`

```
from sklearn.model_selection import RandomizedSearchCV

grid = { "n_estimators": [10, 100, 1200],
         "max_depth": [None, 5, 30],
         "max_features": ["auto", "sqrt"],
         "min_samples_leaf": [1, 2, 4] }

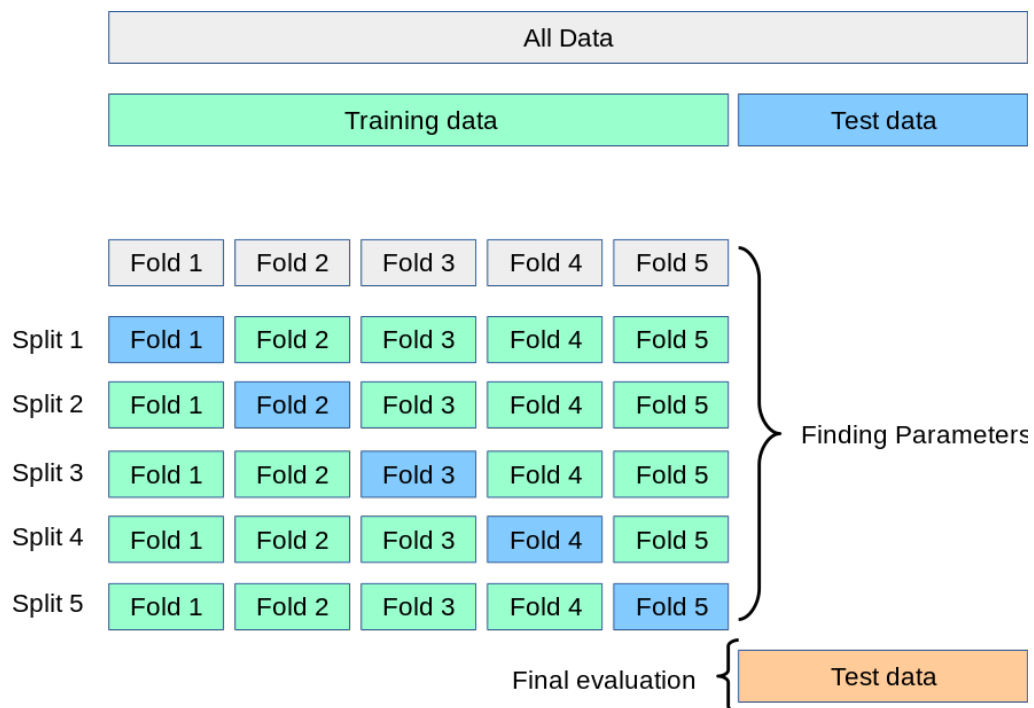
clf = RandomForestRegressor(n_jobs=1)
rs_clf = RandomizedSearchCV(
    estimator=clf,
    param_distributions=grid,
    n_iter=10,
    cv=5,
    verbose=2)
rs_clf.fit(X_train, y_train)
```

Výsledkom oboch funkcií je objekt, ktorý obsahuje parameter `best_estimator_` ktorého hodnota obsahuje najlepšie nájdené nastavenie hyperparametrov z testovaných kombinácií.

2.2.5 Validácia modelu

Predposlednou časťou je validácia modelu. Validácia by nám mala ozrejiť, ako kvalitný náš model je a mali by sme z nej získať dáta, vďaka ktorým ho vieme porovnať s inými modelmi.

Obe metódy (GridSearchCV a RandomizedSearchCV), ktoré sme si predstavili v predchádzajúcej sekcii v sebe implementujú krížovú validáciu. Existuje viacero typov krížovej validácie, avšak základná, ktorú taktiež implementuje knižnica scikit-learn sa nazýva k-zložková krížová validácia. Jej princíp fungovania je založený na tom, že našu dátovú sadu rozdelíme na k-zložiek. Následne je vždy jedna zložka použitá ako testovacia a zvyšné su použité na tréning modelu. Tréning je vykonaný toľkokrát, na koľko častí sme si rozdelili dátovú sadu. Následne nám funkcia vráti priemer všetkých nameraných presností modelu. Týmto zaistíme, že model bude trénovaný a testovaný na celom datasete.



Obr. 2.4: Rozdelenie dátovej sady pri k-zložkovej krížovej validácii[10]

Presnosť modelu môžeme porovnávať viacerými typmi metrík. Použité metriky sa budú líšiť v závislosti od toho, či ideme hodnotiť model, ktorý rieši klasifikačný problém alebo regresný problém.

Klasifikačné metriky Pri binárnych klasifikačných problémoch môžeme naše výsledky zaradiť do štyroch kategórii:

- **pravdivo pozitívne** (True Positive)

- **pravdivo negatívne** (True Negative)
- **nepravdivo pozitívne** (False Positive)
- **nepravdivo negatívne** (False Negative)

Ak by sme napríklad predikovali z datasetu či je nejaká žena tehotná, pravdivo pozitívne prípady by boli prípady, v ktorých by náš model predpovedal že je, a zároveň by aj skutočne bola. Ak by náš model predikoval, že nie je, a bola by to pravda, prípad by sme kategorizovali ako pravdivo negatívny, atď. . .

V závislosti od toho, aký model vytvárame, vieme sa zamerať na potrebnú kategóriu. Ak napríklad robíme model na odhad ľudí, ktorí majú sklony k samovražde, nie je až taký problém, ak máme väčšie množstvo nepravdivo pozitívnych prípadov. V najhoršom prípade odporúčime danej osobe, aby navštívila odborníka, aj keď to nemusí potrebovať. Naopak, ak robíme model pre predikciu choroby, napríklad či má pacient formu rakoviny, tak nepravdivo negatívny prípad môže spôsobiť, že daný pacient to nebude ďalej riešiť a choroba sa môže rozvinúť do ťažšieho štádia a zároveň nepravdivo pozitívny prípad môže stať pacienta veľke množstvo peňazí za liečenie, ktoré ani nepotrebuje.

Pomocou týchto štyroch kategórií vieme vytvoriť maticu zámien. Takáto matica by obsahovala štyri časti a do každej časti by sme napísali počet hodnôt pre danú kategóriu. Vďaka matici zámien vieme ďalej určiť tri základné klasifikačné metriky.

- **Presnosť** (Accuracy) - slúži na meranie koľko skutočne negatívnych a skutočne pozitívnych prípadov bolo správne zatriedených

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Správnosť** (Precision) - slúži na meranie koľko z predikovaných pozitívnych bolo skutočne pozitívnych

$$Precision = \frac{TP}{TP + FP}$$

- **Odvolanie** (Recall) - meria koľko pozorovaní zo všetkých pozitívnych bolo klasifikované ako ako pozitívne

$$Recall = \frac{TP}{TP + FN}$$

- **F1 skóre** - harmonický priemer správnosti a odvolania. V ideálnom prípade chceme mať model, v ktorom je správnosť aj odvolanie 1 a preto potrebujeme metriku, ktorá dokáže zohľadniť tieto dve merania[11]

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Multitriedová klasifikácia používa rovnaké vzorce pre výpočet metriky, avšak nedokážeme vypočítať celkovú metriku modelu. Preto počítame metriky pre každú triedu individuálne.

Regresné metriky Pre regresiu potrebujeme iné typy metrík ako pre klasifikáciu. Pri klasifikačných problémoch predpovedáme určitý počet tried, avšak pri regresných problémoch je výsledkom kontinuálna hodnota. Ak naša klasifikácia predpovedá nepravdivo pozitívnu hodnotu, vieme sa jasne zhodnúť, že takáto hodnota je nesprávna. Predstavme si však napríklad, že náš model predikuje cenu nehnuteľnosti a pomýli sa v predpovedanej hodnote o 1 000€. Takýto model sa dá považovať za celkom presný, aj keď jeho hodnota nebola úplne presná ako skutočná hodnota nehnuteľnosti.

- **Stredná absolútna chyba (MAE)** - je to priemerná chyba, ktorú náš model spraví pri predikcii

$$mae = \frac{\sum_{i=1}^n abs(y_i - \lambda(x_i))}{n}$$

- **Stredná kvadratická odchýlka (MSE)** - je podobná metrika ako stredná absolútna hodnota, avšak jej výhodou je 'trestanie' vysokých rozdielov pri predikcii

$$mse = \frac{\sum_{i=1}^n (y_i - \lambda(x_i))^2}{n}$$

- **R2 skóre** - je podobná metrika ako presnosť. Dáva rýchlu indikáciu ako dobrý môže byť model. Čím bližšie je hodnota k 1, tým presnejší model. R2 skóre však nevraví, ako nepresné predikované hodnoty modelu sú.

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_{pred} - y_{mean})^2}{\sum_{i=1}^n (y_{actual} - y_{mean})^2}$$

2.2.6 Nasadenie na produkciu

Posledná časť procesu je nasadenie modelu do aplikácie. Táto časť procesu nie je súčasťou zadania, avšak ak by sme robili model do praxe, tak by sme napríklad chceli, aby náš model na predikciu cien nehnuteľností bol súčasťou väčšej

aplikácie, napríklad kalkulačky ceny nehnuteľností. V takom prípade by sme museli prepojiť náš model so serverom a poprípadе dorobiť grafické prostredie pre používateľa.

3 Regresné modely

Regresné modely používame pre predikciu premenných na súvislej mierke. Preto pri implementácii nášho modelu využijeme regresné algoritmy z knižnice `sickit-learn`. V tejto časti práce sa pozrieme na teoretickú časť týchto algoritmov a potom v neskoršej časti sa pozrieme na porovnania presnosti modelov daných algoritmov.

3.1 Lineárna regresia

Cieľom jednoduchej (jednorozmernej) lineárnej regresie je modelovať vzťah medzi jednou vlastnosťou (premennou x) a súvislou hodnotou (cieľová premenná y)[12].

Grafom takejto funkcie je lineárne rastúca/ klesajúca priamka. Pre výpočet vzťahu použijeme vzorec:

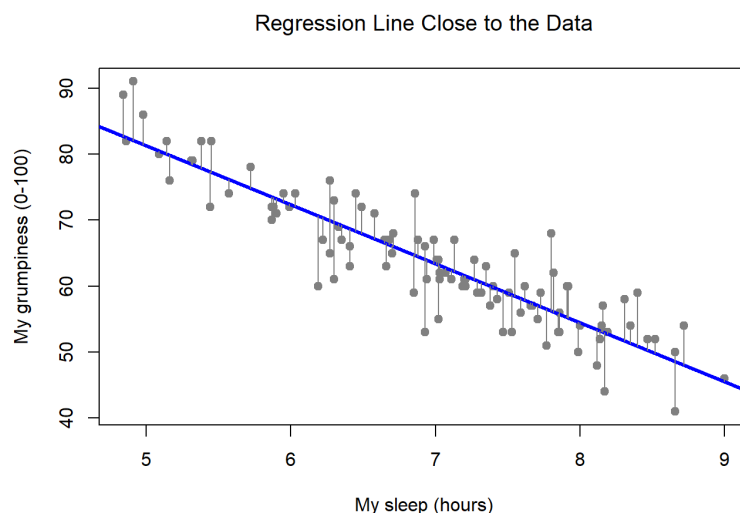
$$y = w_0 + w_1x$$

čo predstavuje jednoduchú lineárnu regresiu. Ak by sme mali viacero príznakov vo vzorke, šlo by o viacnásobnú lineárnu regresiu a pre jej výpočet by sme použili súčet všetkých príznakov.

$$y = w_0x_0 + w_1x_1 \cdots w_mx_m$$

Cieľom modelu, ktorý využíva regresnú analýzu, je potom naučiť sa jednotlivé váhy w_m pre vlastnosti dát, aby mohol predikovať novú hodnotu.

Na obrázku 3.1. môžeme vidieť, ako by mohol vyzeráť príklad použitia lineárnej regresie. Modrá čiara značí natrénovaný regresný model a určuje hodnoty, ktoré bude predikovať. Sivé body na grafe predstavujú skutočné namerané hodnoty a sivé čiary predstavujú veľkosť odchýlky, ktorú bude mať náš model od skutočných hodnôt.



Obr. 3.1: Lineárna regresia[13]

3.2 Hrebeňová regresia

Používa sa na dáta, ktoré trpia multikolinearitou. Multikolinearita nastáva, ak máme v matici príznakov (X) determinant rovný alebo veľmi blízky 0. Algoritmus využíva lineárnu regresiu s regularizáciou L2. Komplexný parameter $\lambda \geq 0$ kontroluje množstvo zmenšenia: čím je väčšia hodnota λ , tým väčšia je hodnota zmenšenia a tým sa koeficienty stanú robustnejšie pre kolinearitu.

$$L2 : \lambda ||w||_2^2 = \lambda \sum_{j=1}^m w_j^2$$

3.3 LASSO

Je skratka pre Least Absolute Shrinkage and Selection Operator. Vykonáva L1 reguláciu - pripočítava penalizáciu rovnú absolútnej hodnote veľkosti koeficientov. Dokáže odhadovať riedke koeficienty. Je užitočný v niektorých kontextoch vzhľadom na jeho tendenciu uprednostňovať riešenia s menším počtom nenulových koeficientov, čím sa efektívne znižuje počet vlastností, na ktorých je dané riešenie závislé.

$$L1 : \lambda ||w||_1 = \lambda \sum_{j=1}^m |w_j|$$

3.4 Elastická sieť

Je algoritmus, ktorý využíva kombináciu LASSO a Ridge regresie, L1 a L2 regularizáciu. Elastic Net vylepšuje LASSO limitácie. Vďaka Ridge regresii, Elastic Net estimator dokáže spracovať korelácie medzi predikátormi lepšie ako LASSO a vďaka L1 regularizácii získame riedkosť.

$$J(w)_{ElasticNet} = \sum_{i=1}^n (y^{(i)} - \hat{y}^{(i)})^2 + \lambda_1 \sum_{j=1}^m w_j^2 + \lambda_2 \sum_{j=1}^m |w_j|$$

3.5 Náhodné lesy

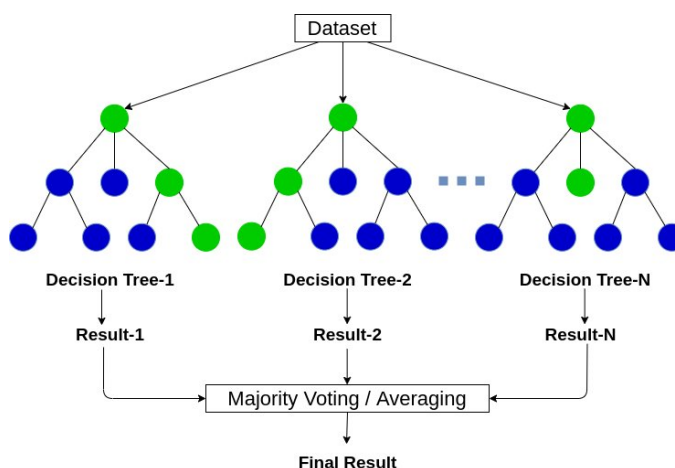
Na to, aby sme si vysvetlili, čo sú nahodné lesy, musíme najprv ozrejmiť čo sú to rozhodovacie stromy. Rozhodovacie stromy získavajú výsledok tak, že, kladú otázky na dáta (na ktoré model odpovedá áno/ nie). Každá otázka zúžuje výber, kým dokáže model spraviť svoju predikciu. Poradie si vyberá model. Každý strom obsahuje vetvy, uzly a listy. Strom prechádza od koreňa (príznak, ktorý dokáže najlepšie separovať vstupné dáta) cez svoje uzly (ďalšie príznaky) po vetvách až k predikovanej hodnote (listu). Rozhodovacie stromy sa dajú uplatniť v regresných aj klasifikačných problémoch. V knižnici scikit-learn ich dokážeme nájsť pod DecisionTreeClassifier.

Výhody rozhodovacích stromov:

- jednoduché na vysvetlenie pre laika
- jednoduchá vizualizácia
- nie je potreba dáta normalizovať
- podporujú numerické aj kategorické hodnoty (implementácia v scikit-learn knižnici však nie)
- funguje pre regresné aj klasifikačné problémy

Nevýhody rozhodovacích stromov:

- môžu vytvárať komplikované modely, ktoré trpia pretrénovaním
- malá zmena v dátach môže vygenerovať úplne iný strom
- nižšia efektivita v prípade predpovede kontinuálnej premennej

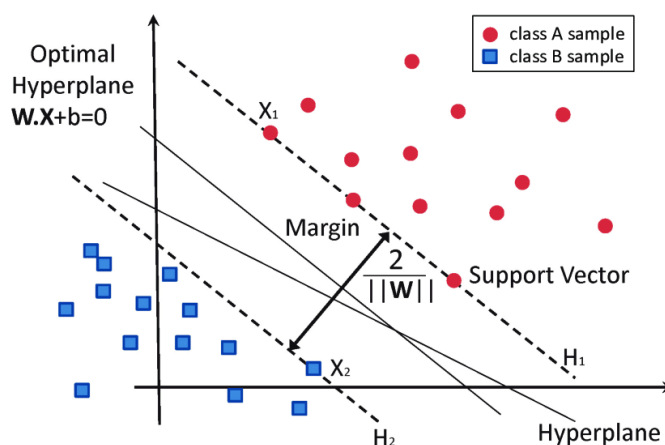


Obr. 3.2: Príklad náhodných lesov[14]

Algoritmus náhodných lesov je vlastne kombinácia viacerých rozhodovacích stromov. Vyberie sa priemerná predpovedaná hodnota alebo trieda, na ktorej sa zhodlo najviac stromov. Náhodný les odstraňuje nedostatok algoritmu rozhodovacích stromov, redukuje pretrénovanie modelu a zvyšuje precíznosť. Knižnica scikit-learn implementuje algoritmus náhodných lesov pomocou `RandomForestRegressor` klasifikátora.

3.6 Metódy podporných vektorov

Metódy podporných vektorov (Support Vector Machines) sú metódy učenia s učiteľom vhodné pre klasifikáciu (SVC) a regresiu (SVR). Cieľom algoritmu je nájsť nadrovinu, ktorá rozdeľuje triedy. Ideálna nadrovina je taká, ktorá má čo najväčší margin od podporných vektorov. Podporné vektory sú potom vzorky, ktoré sú najviac hraničné prípady pre danú triedu.



Obr. 3.3: Metóda podporných vektorov[15]

SVR používa rovnaký princíp ako SVM. Hlavná myšlienka je nájsť najlepšiu priamku. V SVR je najlapešia priamka nadrovina, ktorá má na sebe maximálny počet vzoriek. V SVR je regresia vykonávaná vo vyšších rozmeroch. Na vykonanie tohto kroku preto existuje funkcia, ktorá namapuje dáta do vyššej dimenzie. Funkcia sa nazýva kernel funkcia a je jedným z hyperparametrov modelu.

4 Implementácia riešenia

Implementáciu nášho riešenia predstavuje dvojica skriptov a súbor typu Jupyter notebook¹. Skripty slúžia pre tvorbu datasetu a filtráciu vzoriek. Jupyter notebook sme si zvolili, pretože nám umožňuje kombinovať Pythonovský kód a Markup language. Vďaka nemu vieme pridať nášmu textu, štruktúru a formátovať ho. V Jupyter notebook taktiež dokážeme vizualizovať grafy, ktoré zobrazujeme pomocou knižnice matplotlib.

4.1 Nástroj na tvorbu datasetu

Prvou časťou riešenia nášho problému bolo vytvoriť nástroj pre tvorbu datasetu, na ktorom sa bude model trénovať. V tejto časti si popíšeme ako a ktoré dáta sme zbierali a ako sme následne delili datasety.

4.1.1 BeautifulSoup

Skripty pre vytváranie datasetov sme riešili pomocou knižnice BeautifulSoup. Podľa oficiálnej dokumentácie je jej definícia ako Python knižnica pre vyťahovanie dát z HTML a XML súborov[16]. Knižnicu sme vybrali z dôvodu, aby sme nemuseli používať viacero programovacích jazykov. A keďže už riešime implementáciu modelov v Pythone, tak sme zvolili práve túto knižnicu. Funkcie v knižnici nám umožňujú vyhľadávať triedy, idečka a HTML tagy v texte odpovede, ktorú dostaneme po požiadavke na stránku.

Zdrojový kód 4.1: Príklad použitia funkcie BeautifulSoup, ktorá vyhľadá ul atribúty s triedou list-group-flush

```
soup = BeautifulSoup(requests.get(link_to_scrap).text, 'lxml')
table = soup.find('ul', class_='list-group-flush')
```

¹<https://jupyter.org/>

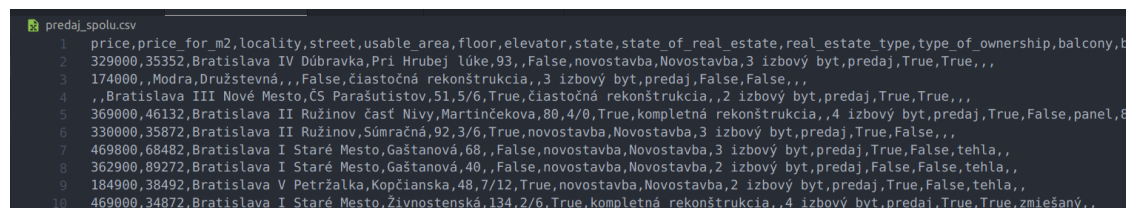
4.1.2 Tvorba datasetov

Pre tvorbu datasetov sme sa rozhodli použiť slovenský portál s realitami www.topreality.sk. Datasets ukladáme do súborov typu CSV. Podľa oficiálnej Python dokumentácie je CSV najpoužívanější importovací a exportovací formát pre tabuľky a databázy [17]. Dáta ukladáme po riadkoch a oddelujeme ich čiarkou (odkiaľ pochádza aj názov - Comma Separated Values).

Tvorba datasetu prebiehala prehádzaním všetkých podstránok a zbieraním údajov, ktoré sa nachádzajú v tabuľke každého inzerátu.

Posielanie veľkého množstva požiadaviek príliš rýchlo môže mať za následok spadnutie servera a vytvoriť zlý user experience pre návštevníkov webu. Toto je však hraničný prípad, avšak posielanie veľkého množstva požiadaviek na server môže spôsobiť znížený výkon webovej stránky. Preto sme pridali v skripte pauzu za každou požiadavkou na podstránku s inzerátom, a trochu dlhšiu pauzu, keď sa menila podstránka, na ktorej boli vylistované viaceré inzeráty. Taktiež sme pridali minútovú pauzu pre prípad, že sa požiadavka na server vráti s chybovou hláškou. Toto opatrenie sme pridali pre prípad, že by nám spadlo internetové pripojenie. Tieto opatrenia však nesú so sebou isté negatívum, proces scrapovania informácií s týmito pauzami na 29 372 vzorkách trval približne 18 hodín.

Datasets sme následne ukladali do troch súborov. Prvý bol pre dáta bytov, druhý pre dáta domov a tretí bol kombináciou oboch dohromady.



Obr. 4.1: Výsledný dataset

4.1.3 Dátové sady - predstavenie vlastností

Všetky tri datasets obsahovali rovnaké vlastnosti vzoriek, líšilo sa iba ich zastúpenie. Nie všetky vlastnosti boli neskôr použité pre tréning modelu. Ich spracovaniu sa budeme venovať v ďalších častiach práce.

- **price** - cena, výsledná hodnota, ktorú predikujeme
- **price_for_m2** - cena za m^2
- **locality** - mesto, v ktorom sa stavba nachádza.

- **street** - ulica
- **usable_area** - užitková plocha, dvor
- **floor** - ak ide o byt, na akom poschodí
- **elevator** - výťah
- **state** - v akom stave je stavba (novostavba, kompletne zrekonštruovaná...)
- **state_of_real_estate** - duplikát atribútu state
- **real_estate_type** - typ nehnuteľnosti (garsónka, 1 izbový byt, mezonet...)
- **type_of_ownership** - typ vlastníctva
- **balcony** - boolean či má nehnuteľnosť balkón alebo loggiu
- **basement** - boolean či má nehnuteľnosť pivnicu
- **material** - typ materiálu, z akého je nehnuteľnosť postavená
- **land** - veľkosť pozemku
- **build_up_area** - veľkosť zastavanej plochy

4.2 Filtrácia dátových vzoriek

Pre manipuláciu s dátami sme použili knižnicu pandas² a pre vizualizáciu matplotlib³

Wes McKinney definoval Pandas ako knižnicu, ktorá chce uľahčiť prácu s dátovými setmi, ktoré sú bežné pre finančné, štatistické a podobné odbory a ponúka súbor základných stavebných blokov pre implementovanie štatistických modelov[18]. Ponúkajú nám dve základné dátové štruktúry, Series a DataFrame, ktoré sa používajú na manipuláciu s jednorozmernými a dvojrozmernými typmi dát.

Matplotlib je pythonovská knižnica pre vytváranie statických, animovaných a interaktívnych vizualizácií v Pythone[19]. Ponúka nám funkcie pre tvorbu grafov a zároveň knižnica pandas má s ňou už veľa integrácií implementovaných.

Našu analýzu dát sme začali vykonávať volaním funkcie `.info()` nad datasetmi, aby sme dostali podrobnejšie informácie o dátach. Z výpisu sme zistili, že nie všetky hodnoty ceny, ktorú sme chceli predikovať boli nenulové a preto

²<https://pandas.pydata.org/>

³<https://matplotlib.org/>

boli neskôr odstránené. Taktiež sme zistili, že veľa atribútov je null hodnoty, preto ich trvalo imputovať, poprípade odstrániť.

Následne sme volali funkciu `.value_counts()` nad atribútami nášho objektu. Pri testovaní sme si všimli, že niektoré vlastnosti sú pre náš model zbytočné a ďalšie bude treba upraviť.

4.2.1 Nepotrebné atribúty

- **price_for_m2** - tento atribút sme ponechali len pre prípady vizualizácie grafov, keď sme však začali trénovať model, bol odstránený, pretože náš model mal predpovedať práve cenu a vďaka tomuto atribútu v kombinácii s atribútom **build_up_area** by ho bolo jednoduché vypočítať
- **state_of_real_estate** - tento atribút nadobúdala len hodnotu 'Novostavba' preto sme ho odstránili a pridali sme hodnotu 'novostavba' vlastnosti **state** pre každú vzorku, ktorá tento atribút vlastnila
- **type_of_ownership** - tento atribút bol taktiež nepodstatný a odstránený, pretože nadobúdala iba hodnotu 'predaj', keďže náš dataset bol vytvorený iba z inzerátov, ktoré nehnuteľnosti predávali

Zdrojový kód 4.2: Úprava atribútov

```
data_combined.loc[
    data_combined["state_of_real_estate"] == 'Novostavba',
    "state"
] = 'novostavba'
data_combined.drop(
    [
        'state_of_real_estate',
        'type_of_ownership',
        'price_for_m2'
    ], axis=1, inplace=True
)
```

4.2.2 Vzorky nepochádzajúce zo Slovenska

Následne bolo potrebné odstrániť z datasetov vzorky, ktoré nepochádzali z územia Slovenska. Problém bol, že vzorky neboli označené podľa štátu ale len podľa mesta v atribúte `locality`. Preto sme si vytvorili pomocný skript

`get_all_cities`, v ktorom sme pomocou knižnice BeautifulSoup zozbierali názvy obcí a miest na Slovensku z wikipedie. Následne bol tento zoznam použitý na filtráciu lokalít v datasete. Taktiež boli pozmenené lokácie v datasete, ak napríklad bolo väčšie mesto rozdelené na mestské časti, ponechali sme len názov mesta.

Zdrojový kód 4.3: Funkcia pre odfiltrovanie mestských častí z názvu mesta

```
with open('city.txt', 'r') as f:
    cities = f.read().splitlines()

for city in cities:
    data_apartmans.loc[
        data_apartmans['locality'].str.contains('{} '.format(city)
        ),
        ['locality']
    ] = city
```

Našli sme pár vzoriek, kde náš skript nefungoval najpresnejšie a zanechal v zozname zopár českých lokalít. Avšak takýchto vzoriek bolo zanedbateľné množstvo v porovnaní s celým datasetom, preto sme takéto hodnoty odignorovali.

4.2.3 Odľahlé hodnoty

Max Kuhn a Kjell Johnson definujú vo svojej knihe odľahlé dáta ako dáta, ktoré sú výnimočne ďaleko od hlavného prúdu údajov [20]. Za ich výskyt môže chyba merania, poškodené údaje alebo reálne výnimky, ktoré sa môžu z času na čas objaviť v reálnom svete. V našom prípade datasetu s cenami sa mohlo stať, že niekto napísal príliš nízku cenu nehnuteľností (100€ napríklad) a neskôr v inzeráte do popisu pridal skutočnú hodnotu, ktorú už ale náš webscraper nezachytil.

Pre zachytenie týchto odľahlých hodnôt môžeme použiť jednoduchú vizualizáciu pomocou grafu a pozrieť sa, kde sa zhromažďuje najviac hodnôt. Avšak takéto rozhodovanie, či je hodnota odľahlá alebo nie, nie je vhodné.

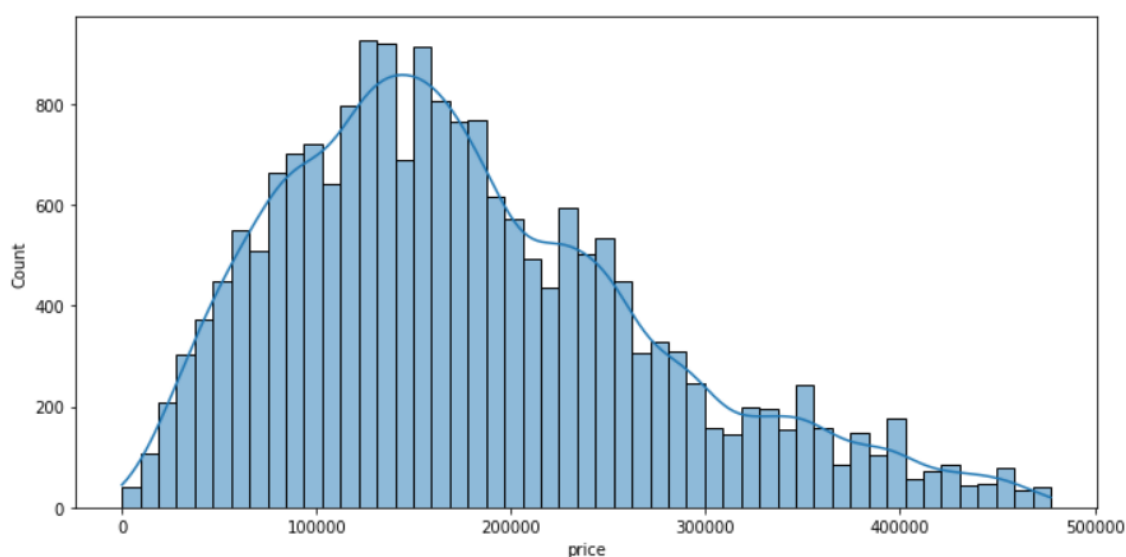
Jeden zo spôsobov ako môžeme určiť dáta, ktoré sú odľahlé, je pomôcť si madzikvartilovým rozsahom. Medzikvartilový rozsah je rozdiel vzdialenosti medzi prvým a tretím kvartilom. Do skupiny odľahlých hodnôt potom budeme brať vzorky, ktorých hodnota je nižšia ako $Q_1 - 1.5 * IQR$ alebo vyššia ako $Q_3 + 1.5 * IQR$. IQR potom vypočítame pomocou nasledujúceho vzorca:

$$IQR = Q_3 - Q_1$$

Hodnota Q_1 je hodnota, pod ktorou leží 25% vzoriek celého datasetu a hodnota Q_3 je hodnota, pod ktorou leží 75% všetkých hodnôt. Vďaka konštante 1.5 dostávame rozsah pokrytia pod 99.72% avšak nad 95.44% Gausovej krivky hodnôt. A preto môžeme považovať všetky hodnoty, ktoré nezapadajú do tohto rozpetia, za odľahlé hodnoty.

V skriptoch sme preto vytvorili funkciu `find_outliers`, ktorá využíva tento vzorec a vracia zoznam odľahlých hodnôt, ktoré sme potom odstánili z datasetu.

4.3 Vizualizácia dát



Obr. 4.2: Histogram cien nehnuteľností

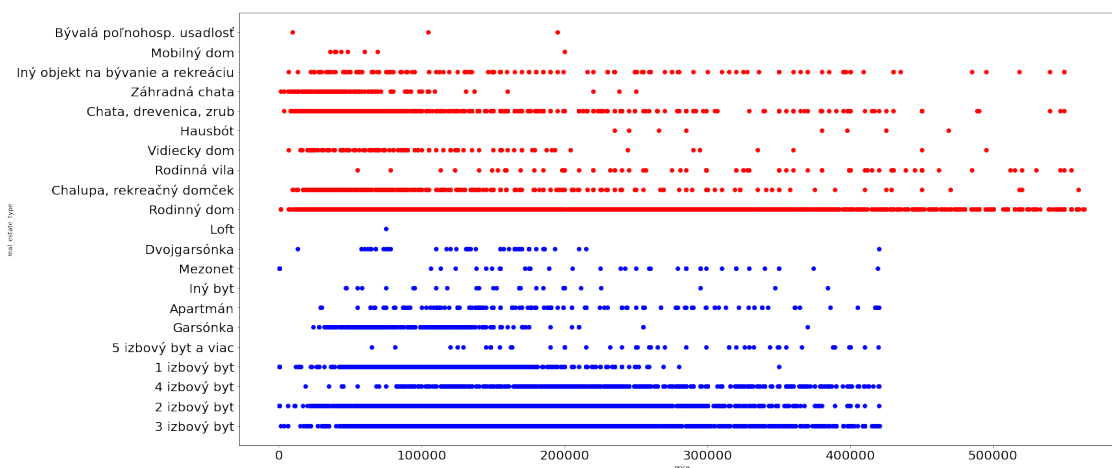
Ako prvý graf sme si vizualizovali histogram cien nahnuteľností oboch datasetov bez odľahlých hodnôt. Histogram je graf, ktorého osa x tvorí triedy hodnôt pozorovanej veličiny (v našom prípade cena nehnuteľnosti) a osa y, slúžiaca na znázornenie absolútnej hodnoty početnosti danej triedy. Z nášho grafu vieme vyčítať, že najčastejšie sa cena nahnuteľnosti pohybovala medzi 100 - 200 tisíc eur, a po odstránení odľahlých hodnôt už náš dataset neobsahoval hodnoty rovné alebo väčšie ako 500 000.

Tento graf však zobrazuje tretí dataset, ktorý obsahoval všetky vzorky zozbierané webscraperom a odľahlé hodnoty boli počítané práve na ňom. Z toho dôvodu je na tomto grafe vidieť, že nexistujú vzorky, ktoré by mali hodnotu ceny vyššiu alebo rovnú 500 000 aj keď v neskorších grafoch také vzorky budú existovať, pretože neškoršie grafy boli tvorené kombináciou datasetu pre byty a datasetu pre domy a odľahlé hodnoty boli prepočítavané jednotlivo na oboch

datasetoch.

Zdrojový kód 4.4: Funkcia pre vykreslenie histogramu cien

```
fig, ax = plt.subplots(figsize=(10,5))
sns.histplot(combined_without_outliers["price"], kde = True)
ax.ticklabel_format(useOffset=False, style='plain')
plt.tight_layout()
plt.show()
```



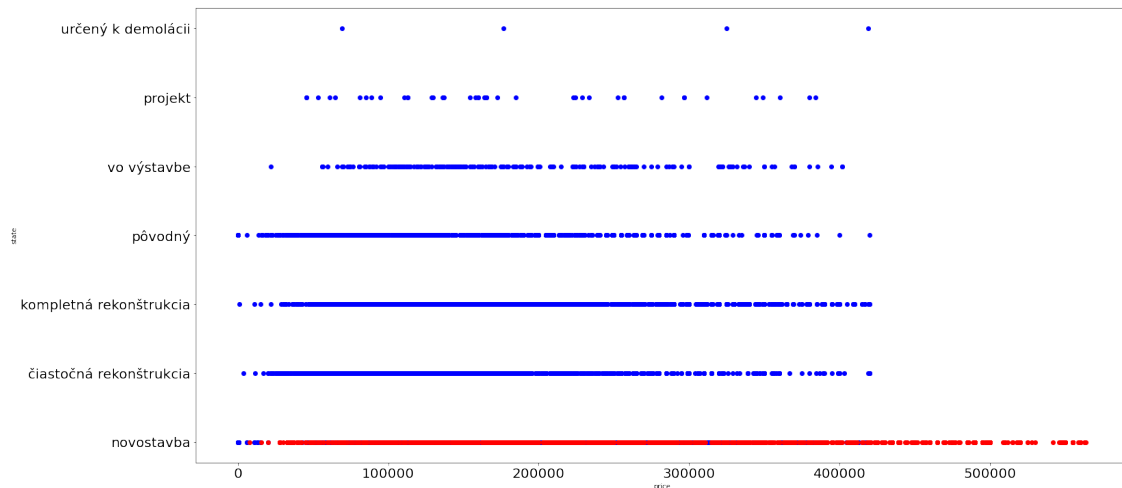
Obr. 4.3: Porovnanie cien a typov nehnuteľností

Ako ďalšie grafy sme si vykreslili korelačné diagramy. V týchto diagramoch boli farebne odlíšené naše datasety, kde modré vzorky predstavovali dataset bytov a červené vzorky dataset domov.

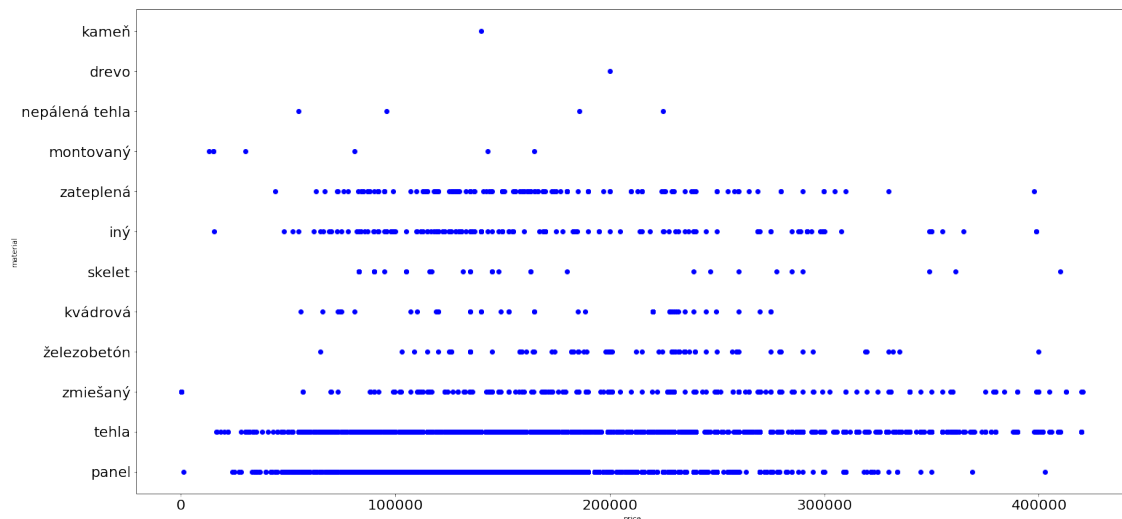
Prvý z nich predstavoval koreláciu medzi typom nehnuteľnosti a jej cenou. Z grafu vieme vypočítavať, že najväčšie zastúpenie v datasete domov predstavovala nehnuteľnosť typu 'Rodinný dom' a najmenšie zastúpenie mala 'Bývalá poľnohospodárska usadlosť'. Dataset bytov mal najväčšie zastúpenie práve v kategóriách 1, 2, 3 a 4 izbový byt a najmenej častý bol 'Loft', ktorý sa v datasete vyskytol len raz. Taktiež v datasete pre byty mali vzorky najväčšiu cenu 420 668€.

Ďalší graf slúžil pre vizualizáciu korelácie medzi cenou a stavom stavby. Z grafu môžeme určiť, že všetky vzorky z datasetu pre domy boli v stave novostavba a zvyšné hodnoty atribútov boli použité iba v datasete bytov. Novostavba bola taktiež najčastejším stavom nehnuteľnosti.

Náš posledný korelačný graf vizualizoval koreláciu medzi typom materiálu, použitého pri stavbe a ceny nehnuteľnosti. Najčastejším materiálom boli tehla a panel, z grafu taktiež vieme vyčítať, že dataset domov neobsahoval dáta o type materiálu použitého pri stavbe.



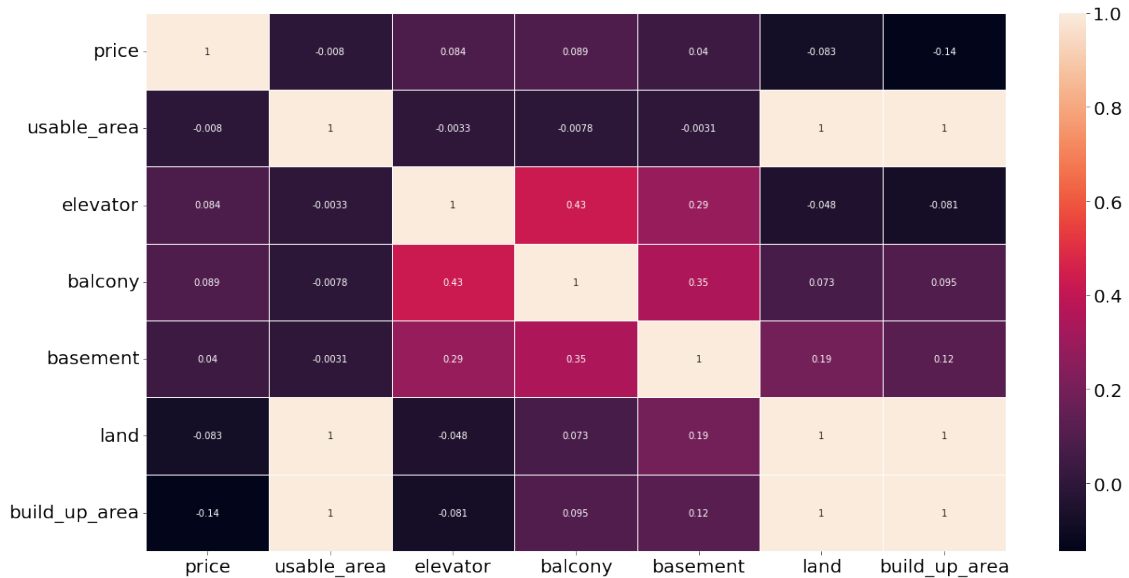
Obr. 4.4: Porovnanie cien a stavov nehnuteľností



Obr. 4.5: Porovnanie cien a materiálov nehnuteľností

Po analýze a vizualizácii dát sme zistili, že veľa dát, ktoré obsahoval dataset o bytoch, práve dataset o domoch neobsahoval. Na základe toho môžeme očakávať, že náš model pre odhad cien pre domy bude mať nižšiu presnosť ako ten pre byty.

Ako posledný graf sme si vykreslili korelačnú maticu. Korelačná matica je tabuľka, ktorá vizualizuje koreláciu medzi koeficientami jednotlivých atribútov. Zobrazuje však len korelácie pre číselné atribúty. Korelačná matica sa používa hlavne pri veľkých datasetoch pre zobrazenie dôležitosti atribútov. Nie všetky atribúty sú pri predikcii podstatné a vďaka korelačnej matici vieme takéto atribúty lepšie identifikovať. Z našej korelačnej matice vieme vypožorovať, že najväčší vplyv na zmenu ceny mal práve atribút **build_up_area**.



Obr. 4.6: Korelačná matica atribútov datasetu byty

4.4 Softvér na odhad trhovej ceny nehnuteľností

Hlavnou časťou tejto práce bolo vytvoriť čo najpresnejší model pre odhad trhovej ceny nehnuteľnosti. Pri jeho tvorbe sme postupovali nasledovne.

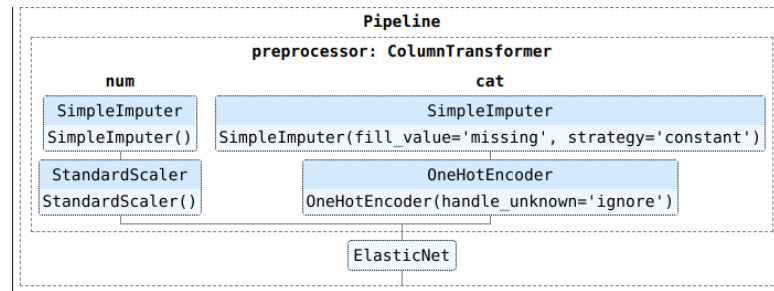
4.4.1 Predspracovanie dát

Dáta sme už mali v tomto kroku čiastočne predspracované, keďže sme odstraňovali odľahlé hodnoty a nepotrebné atribúty ešte pred ich vizualizáciou. Náš ďalší krok bolo rozdelenie dát. Dáta sme oddelili na atribúty a cieľovú predikovanú hodnotu. Ešte pred tým sme ich však premiešali, aby sa nestalo, že sú pohromade nejaké vzorky, ktoré súvisia spolu spôsobom, ktorý sme si nevšimli pri analýze dát a nekazili nám tým krížovo validačné skóre.

Zdrojový kód 4.5: Rozdelenie dát

```
data_apartmans = data_apartmans.sample(frac = 1)
X_apartmans = data_apartmans.drop("price", axis=1)
y_apartmans = data_apartmans["price"]
```

Potom sme potrebovali doplniť chýbajúce hodnoty do datasetu pomocou Imputera, pretransformovať kategorické hodnoty pomocou One Hot Encodera a škálovať numerické hodnoty. Aby sme si tento krok uľahčili použili sme pipelines, ktoré ponúka knižnica sci-kit learn. Dokumentácia sci-kit learn uvádza, že výhoda pipelines je zaobalenie viacerých krokov, ktoré môžu byť použité spoločne pre krížovú validáciu[10].



Obr. 4.7: Pipeline s regresorom ElasticNet

Ako prvý krok sme si rozdelili naše atribúty na numerické a kategorické.

- **numerické**

- usable_area
- land
- build_up_area

- **kategorické**

- locality
- street
- real_estate_type
- material
- floor
- elevator
- balcony
- basement

Pre numerické chýbajúce hodnoty sme sa rozhodli doplniť priemernú hodnotu daného atribútu, na čo sme využili funkciu SimpleImputer knižnice sci-kit learn. Kategorickým hodnotám, ktoré v datasete chýbali sme zase priradili stringovú hodnotu 'missing'. Takto sme spojili numerický transformer pomocou pipeline, ktorá pozostávala z objektov Simple Imputer a Standard Scaler.

Knižnica sci-kit learn ponuka viacero spôsobov ako využiť škálovanie. Najčastejšie z nich sa používajú StandardScaler, MinMaxScaler a RobustScaler. RobustScaler sme nepoužili, pretože odstraňuje medián a škáluje dáta podľa medzikvartilového rozptylu. My sme už však odľahlé hodnoty odstránili. Standard Scaler sa odporúča použiť v prípade, že dáta nasledujú normálnu distribúciu (ich grafom je Gausova krivka). MinMaxScaler zase funguje, že transformuje

všetky atribúty medzi hodnoty 0 a 1. My sme však aj napriek tomu použili práve StandardScaler. Dôvodom použitia StandardScalera bolo testovanie parametra 'R2 skóre' regresorov, kde hodnoty vychádzali presnejšie práve pri použití StandardScalera ako pri použití MinMaxScalera.

Kódovanie kategorických hodnôt sme riešili pomocou One Hot Encodera. Ako kategorické hodnoty sme zadefinovali všetky atribúty nášho datasetu, ktorých hodnota bola typu reťazec. Zvažovali sme aj použitie Ordinal Encodera pre atribút **real_estate_type**, avšak zostali sme pri One Hot Encoderi aj v tomto prípade. Aj keď sa dá určiť, že hodnota '3 izbový byt' by mala byť viac ako napríklad '1 izbový byt' je diskutabilné robiť takéto zoradenie napríklad pre hodnoty 'Chata, drevenica, zrub' a 'Chalupa, rekreačný domček'.

4.4.2 Trénovanie modelov

Po prespracovaní dát sme prešli k trénovaniu dát. Každý regresor sme najprv testovali bez hyperparametrov a výsledné predpovedané hodnoty poslali spolu s reálnymi hodnotami do našej funkcie **evaluate_preds()**. Táto funkcia vracia objekt s metrikou, ktorú sme použili pre hodnotenie modelov. Náš dataset sme delili pri trénovaní v pomere 80:20 v prospech trénovacích dát.

Zdrojový kód 4.6: Trénovanie modelu, predikcie a metrika

```
from sklearn.model_selection import train_test_split,
    cross_val_score

X_train, X_test, y_train, y_test = train_test_split(
    X_houses, y_houses, test_size=0.2, random_state=42
)

houses_pipe.fit(X_train, y_train)
y_preds = houses_pipe.predict(X_test)
baseline_metrics = evaluate_preds(y_test, y_preds)
```

4.4.3 Ladenie hyperparametrov

Hľadanie najvhodnejších hyperparametrov sme implementovali pomocou funkcie GridSearchCV. Pre každý regresor sme si spravili slovník so zoznamom parametrov, ktoré sme chceli testovať. Funkcia následne používa k-zložkovú krížovú validáciu pre vyhľadanie najlepších parametrov. Defaultne ide o 5 zložkovú validáciu.

Zdrojový kód 4.7: Slovník hyperparametrov regresora SVR

```
svr_params = {
    'model__kernel': ['rbf', 'sigmoid'],
    'model__C': [0.01, 0.1, 1.0, 10.0, 100.0, 1000.0],
    'model__epsilon': [0.1, 0.3, 0.7, 1.0, 1.3],
    'model__gamma': [1.0, 0.1, 0.01, 0.001, 0.0001, 0.00001]
}
```

Funkcia GridSearchCV po nájdení najlepších hyperparametrov z poskytnutého slovníka vracia objekt, ktorý obsahuje parametre:

- **best_estimator** - model spolu s nastaveniami hyperparametrov
- **best_score** - najlepšie dosiahnute skóre (R2 skóre)
- **best_params** - parametre, ktoré dosiahli najlepšiu presnosť

4.4.4 Predikcia nových hodnôt

Pre načítanie dát, pre ktoré chceme predikovať hodnotu, je potrebné najprv vytvoriť CSV súbor v koreňovom adresári skriptov. Podľa toho, aký model chceme použiť, volíme aj názov súboru.

- **input_apartmans** - model trénovaný na datasete apartmans
- **input_houses** - model trénovaný na datasete houses
- **input_combined** - model trénovaný na datasete combined

Nie je potrebné zadať všetky atribúty, stačí, ak sa v súbore bude nachádzať správny počet čiarok. Pre predikciu modelom trénovaným na datasete houses je potrebný súbor so siedmimi čiarkami a pre predikciu datasetom apartmans alebo datasetom combined je potrebných dvanásť čiarok.

Potom spustíme bunku, ktorá má markdown popis s rovnakým názvom. Výsledok predikovanej hodnoty bude uložený v premennej `y_pred`. Pre predikciu novej hodnoty sme použili model, ktorý dosahoval najpresnejšie výsledky zo všetkých testovaných regresorov.

5 Analýza riešenia

5.1 Porovnanie datasetov pred a po filtrácii dát

Naše datasety, ktoré vyvoril web scraper obsahovali atribúty, ktoré neboli potrebné alebo vhodné pre predikciu. Z toho dôvodu boli tieto atribúty z datasetov odstránené. Najviac filtráciou dát utrpel dataset houses, keďže ten neobsahoval žiadne informácie v atribútoch **floor**, **elevator**, **balcony**, **basement** a **material**. Preto sme tieto atribúty z neho zmazali.

Z všetkých datasetov sme taktiež zmazali atribúty **price_for_m2**, **state_of_real_estate** a **type_of_ownership**.

Po zmazaní nepotrebných atribútov, vzoriek nepochádzajúcich zo Slovenska a zmazaní odľahlých hodnôt a odstránení vzoriek, ktoré neobsahovali cieľovú cenu, naše datasety prišli o značnú časť dát. Celkovo sme prišli o 9 895 vzoriek z oboch datasetov.

Tabuľka 5.1: Dáta pred filtráciou

Typ Datasetu	Počet vzoriek	Počet atribútov
Byty	15 562	16
Domy	13 809	16
Spolu	29 371	16

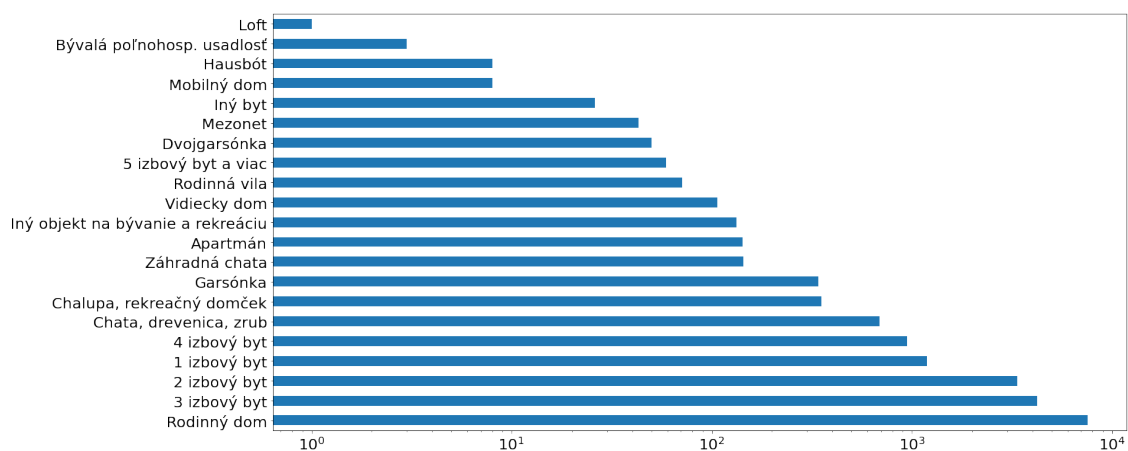
Tabuľka 5.2: Dáta po filtrácii

Typ Datasetu	Počet vzoriek	Počet atribútov
Byty	12 769	13
Domy	10 297	8
Spolu	19 476	13

5.2 Analýza datasetu

Dataset, sme si čiastočne predstavili už v časti o vizualizácii dát. Teraz, sa pozrieme podrobnejšie na to, aké hodnoty, jeho vlastnosti dosahovali. V časti o vizualizácii dát, sme robili korelačné grafy medzi jednotlivými atribútmi a cenou nehnuteľnosti. Avšak z takého grafu, sa ťažko odhaduje v akom množstve sa dané hodnoty atribútov nachádzajú v datasete.

5.2.1 Typy nehnuteľností



Obr. 5.1: Vizualizácia zastúpenia typov nehnuteľností

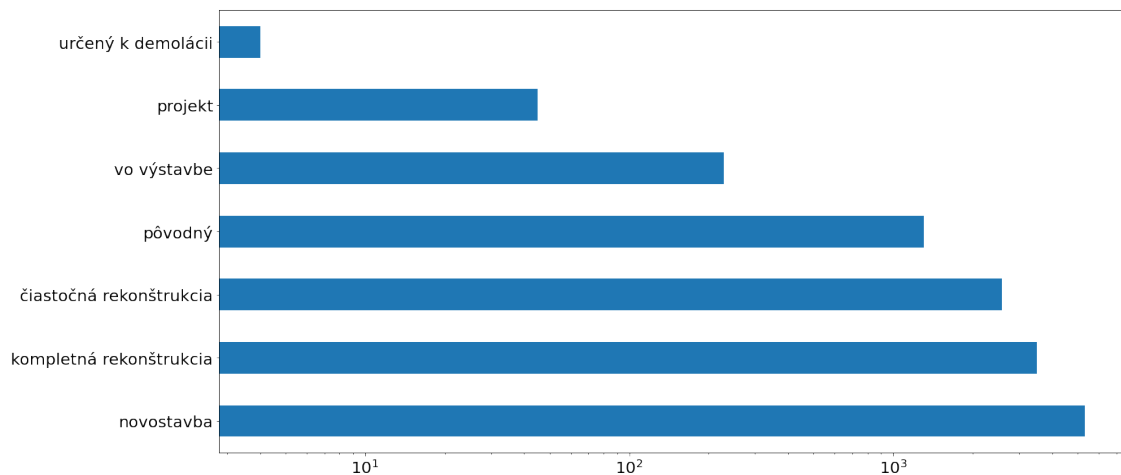
Najčastejšie zastúpeným typom nehnuteľností bol 'Rodinný dom'. Hodnota bola prítomná v datasetoch houses a combined. Hneď za ním bola hodnota '3 izbový byt', ktorá bola najčastejšou hodnotou v datasete apartmans a druhou najčastejšou v datasete combined.

5.2.2 Stavby nehnuteľností

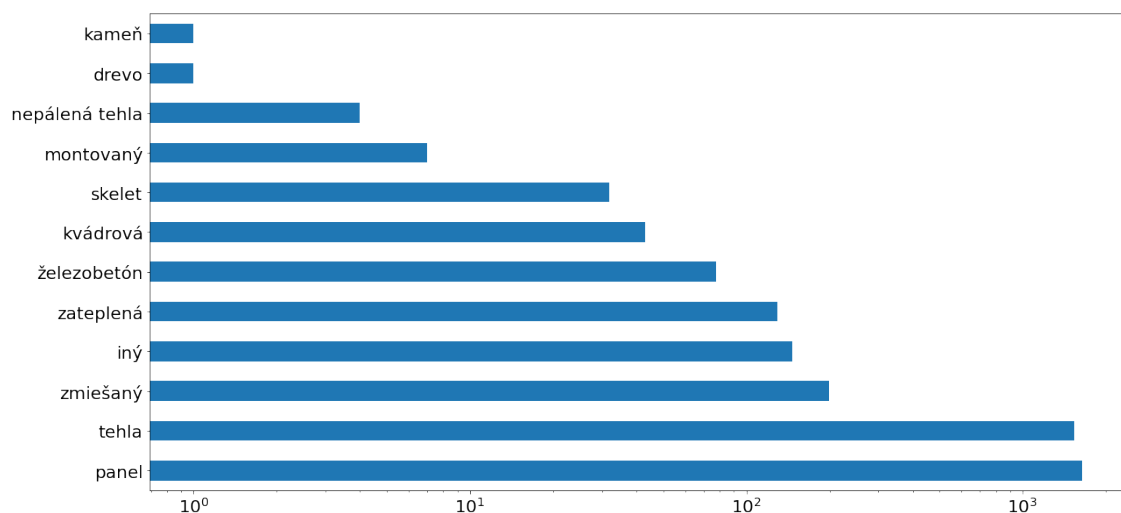
Najčastejším stavom nehnuteľnosti, bola hodnota 'novostavba'. Zároveň, to bola jediná hodnota atribútu state, ktorý nadobudali vzorky z datasetu houses. Druhou najpočetnejšou hodnotou bola 'kompletná rekonštrukcia'. Hodnota sa nachádzala v datasetoch houses a combined.

5.2.3 Materiály nehnuteľností

Následný graf platí len pre dataset combined a dataset apartmans. Dataset houses nenadobúdal žiadne hodnoty pre atribút material. Najčastejším typom materiálu



Obr. 5.2: Vizualizácia zastúpenia stavov v ktorých sa nachádzali nehnuteľnosti z datasetov



Obr. 5.3: Vizualizácia zastúpenia materiálov z ktorých boli nehnuteľnosti postavené

z ktorého boli nehnuteľnosti postavené bol panel. Hneď za ním, nasledovala tehla. Najmenej častým stavebným materiálom bol kameň.

5.3 Modely bez ladenia hyperparametrov

Najlepšie hodnoty dosahovali modely náhodných lesov, LASSO regresie a hrebeňovej regresie. Najhoršie hodnoty dosahoval regresor SVR, neskôr s ladením hyperparametrov sa tieto hodnoty zlepšili.

Najdlhšie tréningovanie trvalo modelom náhodných lesov a SVR regresoru. Naopak najrýchlejšie sa dokázali natréňovať modely hrebeňovej regresie a elastickej siete.

Model hrebeňovej regresie dokázal produkovať druhé najpresnejšie predikcie, hneď za modelom náhodných lesov. Na rozdiel od náhodných lesov bolo jeho tréňovanie o dosť rýchlejšie. Kde tréňovanie jedného modelu náhodného lesa trvalo zhruba 1 min, hrebeňová regresia to dokázala za pár sekúnd.

Najmenšiu strednú absolútnu chybu mal pri všetkých testoch model náhodných lesov. Najhoršie v tejto metrike skončil SVR regresor.

Najmenšiu strednú kvadratickú odchýlku dosahoval model náhodných lesov s výnimkou datasetu houses, kde sa lepšie umiestnil model hrebeňovej regresie.

Tabuľka 5.3: Výsledky základných modelov pre dataset apartmans

Regresor	R2 skóre	MAE	MSE
LASSO	0.74	26 256.54	1 497 492 914
Ridge	0.75	26 313.03	1 466 578 338.93
Elastic	0.21	52 124.61	4 591 534 159.72
SVR	-0.02	58 313.99	5 915 534 917.26
Forest	0.8	21 119.14	1 147 449 680.63

Tabuľka 5.4: Výsledky základných modelov pre dataset houses

Regresor	R2 skóre	MAE	MSE
LASSO	0.34	68 887.95	9 393 420 398.46
Ridge	0.45	66 080.45	7 768 232 386.7
Elastic	0.11	89 189.52	12 628 658 709.84
SVR	-0.02	94 883.18	14 518 414 731.23
Forest	0.44	62 574.9	7 991 123 738.54

Tabuľka 5.5: Výsledky základných modelov pre dataset combined

Regresor	R2 skóre	MAE	MSE
Lasso	0.54	45 642.74	4 357 098 125.63
Ridge	0.57	45 167.52	4 130 630 331.22
Elastic	0.1	73 047.28	8 591 828 711.62
SVR	-0.3	76 638.57	9 853 385 526.73
Forest	0.58	41 919.13	4 000 904 795.66

5.4 Ladenie hyperparametrov

Ladenie hyperparametrov sme implementovali funkciou GridSearchCV, ktorej sme poslali parametre vo forme Pythonovského slovníka. Skúšali sme ho na datasetoch `apartmans` a `houses`. Neskúšali sme ho na `datasete combined`, pretože jeho trénovanie dosahovalo veľmi vysoké hodnoty. Len jeden model s použitím `Random Forest Regresora` sa trénoval zhruba 30 min. Pričom bola použitá 5 zložková validácia, čiže otestovanie len jednej kombinácie parametrov trvalo na trénovať približne 4 a pol hodiny.

5.4.1 Dataset `apartmans`

Najlepšie zlepšenie sa prejavilo pri `SVR` regresore. Najlepšie sa umiestnil `Random Forest Regresor` s `R2 skóre 0.81`.

Tabuľka 5.6: Výsledky ladenia hyperparametrov na `datasete apartmans`

Regresor	R2 skóre
LASSO	0.67
Ridge	0.41
Elastic	-0.00
SVR	0.43
Forest	0.81

Zoznamy hyperparametrov pre jednotlivé regresory s najlepším skóre hodnotenia

- **LASSO** - `alpha: 100`
- **Ridge** - `alpha: 1000`
- **Elastic Net** - `alpha: 100, max_iter: 5, l1_ratio: 0`
- **SVR** - `kernel: sigmoid, C: 1 000, epsilon: 1.3, gamma: 0.1`
- **Random Forest** - `max_depth: None, max_features: auto, min_samples_split: 2, n_estimators: 1000, num_imputer_strategy: median`

Tabuľka 5.7: Výsledky ladenia hyperparametrov na datasete houses

Regresor	R2 skóre
LASSO	0.43
Ridge	0.45
Elastic	-0.11
SVR	0.48
Forest	0.82

5.4.2 Dataset houses

Zoznamy hyperparametrov pre jednotlivé regresory s najlepším skóre hodnotenia

- **LASSO** - alpha: 10
- **Ridge** - alpha: 1
- **Elastic Net** - alpha: 100, max_iter: 5, l1_ratio: 0
- **SVR** - kernel: rbf, C: 1 000, epsilon: 1.3, gamma: 0.1
- **Random Forest** - max_depth: None, max_features: auto, min_samples_split: 2, n_estimators: 1000, num_imputer_strategy: median

5.4.3 Výsledný model

Model, ktorý sme sa nakoniec rozhodli používať pre predikcie bol RandomForestRegressor. Tento model dokázal odhadovať cenu s najlepšou presnosťou zo všetkých testovaných. Jeho negatívum je dlhá doba tréovania.

6 Záver

Hlavnou úlohou bakalárskej práce bolo vytvorenie softvéru pre predikciu cien nehnuteľností. Podúlohami bolo vytvorenie softvéru pre tvorbu datasetu, vizualizácia vzoriek datasetu, analýza presnosti vytvoreného softvéru pre predikciu cien a vytvorenie používateľskej a systémovej príručky.

V úvode práce sa analyzoval vývin cien nehnuteľností v priebehu rokov. Pozreli sme sa na už vytvorené kalkulačky cien, ktoré na Slovensku existujú a taktiež ako sa posudzuje cena pri žiadosti o úver.

V ďalšej kapitole sme si predstavili strojové učenie, aké základné druhy rozlišujeme. Pozreli sme sa na proces riešenia problému za pomoci strojového učenia.

Následne sme si priblížili regresné modely a aké regresory sa najčastejšie používajú pri regresných problémoch.

V syntetickej časti sme si predstavili, ako sa tvoril dataset, ako sme filtrovali dáta, ako sme ich predspracovali a ako sme trénovali modely.

Spravili sme v nej taktiež analýzu datasetu, preskúmali sme z akých vzoriek dataset pozostáva a graficky sme ho zobrazovali za pomoci knižnice matplotlib.

Za prínos tejto práce sa dá pokladať nástroj pre tvorbu datasetu a vytvorený dataset, ktorý sa dá použiť na učebné účely v rámci predmetu Inteligentné systémy.

V budúcnosti sa môžu preskúmať iné typy modelov, či sa nenájde model, ktorý by mal lepšiu presnosť. Môžu sa na to použiť knižnice ako CatBoost, Tensorflow alebo PyTorch. Taktiež sa môže zlepšiť nástroj na tvorbu datasetu, kde sa môžu pridať vlastnosti vzoriek.

Taktiež sa môže spraviť grafické prostredie pre zadávanie hodnôt, ktoré chceme predpovedať, aby sme sa vyhli spúšťaniu zo skriptu a softvér by bol viac používateľsky prívetivý.

Literatúra

1. SLOVNESKA, Národná banka. *Vývoj cien nehnuteľností na bývanie v SR*. Dostupné tiež z: <https://nbs.sk/statisticke-udaje/vybrane-makroekonomicke-ukazovatele/ceny-nehnutelnosti-na-byvanie/vyvoj-cien-nehnutelnosti-na-byvanie-v-sr/>.
2. GRUSZECKÁ, Timea. *Ceny bytov a domov na Slovensku v roku 2022*. 2021. Dostupné tiež z: <https://www.amazonreality.sk/ceny-bytov-domov-na-slovensku/>.
3. *Vývoj cien nehnuteľností na Slovensku za posledných 53 týždňov*. 2022. Dostupné tiež z: <https://www.nehnutelnosti.sk/ceny/>.
4. MARTIŠKA, Andrej. *Ceny nehnuteľností naďalej rastú. Zastaví ich niečo?* 2022. Dostupné tiež z: <https://www.tatrabanka.sk/sk/blog/ekonomicke-analyzy/ceny-nehnutelnosti-rastu-zastavi-ich-nieco/>.
5. MITCHELL, Tom. *Machine Learning*. 1. vyd. McGraw Hill, 1997. ISBN 0070428077.
6. ELHAMRAOUI, Zahra. *Between Classification and Regression How to know which is which?* 2020. Dostupné tiež z: <https://laptrinhx.com/between-classification-and-regression-how-to-know-which-is-which-1045807665/>.
7. DELSOLE, Michael. *What is One Hot Encoding and How to Do It*. 2018. Dostupné tiež z: <https://medium.com/@michaeldelsole/what-is-one-hot-encoding-and-how-to-do-it-f0ae272f1179>.
8. EDUCATION, IBM Cloud. *Underfitting*. 2021. Dostupné tiež z: <https://www.ibm.com/cloud/learn/underfitting>.
9. KASIRAJAN, Aarthi. *UNDERFIT and OVERFIT Explained*. 2020. Dostupné tiež z: <https://medium.com/@minions.k/underfit-and-overfit-explained-8161559b37db>.

10. PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011, roč. 12, s. 2825–2830.
11. B, Harikrishnan N. *Confusion Matrix, Accuracy, Precision, Recall, F1 Score*. 2019. Dostupné tiež z: <https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd>.
12. RASCHKA, Sebastian. *Python Machine Learning*. Packt Publishing, 2015. ISBN 1783555130.
13. NAVARRO, Daniel. *Learning statistics with R: A tutorial for psychology students and other beginners (Version 0.6.1)*. 2019.
14. SHARMA, Abhishek. *Decision Tree vs. Random Forest – Which Algorithm Should you Use?* 2020. Dostupné tiež z: <https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/>.
15. GARCÍA-GONZALO, Esperanza; FERNÁNDEZ-MUÑIZ, Zulima; GARCIA NIETO, Paulino Jose; SÁNCHEZ, Antonio; MENÉNDEZ, Marta. Hard-Rock Stability Analysis for Span Design in Entry-Type Excavations with Learning Classifiers. *Materials*. 2016, roč. 9, s. 531. Dostupné z DOI: 10.3390/ma9070531.
16. RICHARDSON, Leonard. Beautiful soup documentation. *April*. 2007.
17. VAN ROSSUM, Guido; DRAKE, Fred L. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN 1441412697.
18. MCKINNEY, Wes. Data Structures for Statistical Computing in Python. In: WALT, Stéfan van der; MILLMAN, Jarrod (ed.). *Proceedings of the 9th Python in Science Conference*. 2010, s. 56–61. Dostupné z DOI: 10.25080/Majora-92bf1922-00a.
19. HUNTER, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*. 2007, roč. 9, č. 3, s. 90–95. Dostupné z DOI: 10.1109/MCSE.2007.55.
20. KUHN, Max; JOHNSON, Kjell. *Applied Predictive Modeling*. Applied predictive modeling. New York, NY: Springer, 2013. ISBN 9781461468493 1461468493 1461468485 9781461468486. Dostupné tiež z: <http://www.amazon.com/Applied-Predictive-Modeling-Max-Kuhn/dp/1461468485/>.

Zoznam príloh

Príloha A Používateľská príručka

Príloha B Systémová príručka

Príloha C CD médium – záverečná práca v elektronickej podobe

A Používateľská príručka

A.1 Skript `create_dataset`

Tento skript slúži pre tvorbu datasetov. Spúšťa sa príkazom `python3 create_dataset`. Skript následne vytvorí 3 csv súbory a naplní ich dátami z portálu `topreality.sk`. Skript môže bežať aj 18 hodín. Celkový čas bude závisieť od aktuálneho množstva inzerátov na portáli.

A.2 Skript `get_all_cities`

Skript sa spúšťa príkazom `python3 get_all_cities`. Jeho výstupom je textový súbor, ktorý obsahuje zoznam všetkých miest a obcí na Slovensku. V prípade, že nie je v koreňovom adresári prítomný súbor `cities.txt`, je spustenie tohoto skriptu nutnosťou pre správne fungovanie Skriptu `main`.

A.3 Skript `main`

Súbor je formátu Jupyter Notebook. Každá bunka s kódom obsahuje popis akú funkcionality spúšťa. Bunky treba pre správne fungovanie spúšťať v chronologickom poradí.

Pre načítanie dát, pre ktoré chceme predikovať hodnotu, je potrebné najprv vytvoriť CSV súbor. Podľa toho aký model chceme použiť volíme aj názov súboru. Nie je potrebné zadať všetky atribúty, stačí ak sa v súbore bude nachádzať správny počet čiarok. Pre predikciu modelom trénovaným na datasete `houses` je potrebný súbor so siedmimi čiarkami a pre predikciu datasetom `apartmans` alebo datasetom `combined` je potrebných dvanásť čiarok.

- **`input_apartmans`** - model trénovaný na datasete `apartmans`
- **`input_houses`** - model trénovaný na datasete `houses`

- **input_combined** - model trénovaný na datasete combined

Potom spustíme bunku, ktorá má markdown popis s rovnakým názvom. Výsledok predikovanej hodnoty bude uložený v premennej `y_pred`.

B Systémová príručka

B.1 Skript `create_dataset`

Skript využíva knižnicu BeautifulSoup pre vyťahovanie dát z HTML súborov.

Predtým ako budeme zapisovať do CSV súboru musíme zadať jeho hlavičku. Používame na to funkciu `writerow()` do ktorej posielame hlavičku. V objekte `header` sa nachádzajú stringové hodnoty hlavičky.

Funkcia `make_request()` posiela požiadavku na server, v prípade chybovej návratovej hodnoty minútú počká a svoj pokus zopakuje.

V zozname `webs_to_scrap` máme dve stringové hodnoty, prvá z nich predstavuje stránku portálu `topreality.sk` s vyfiltrovanými bytmi. Druhý reťazec je podstránka s domami portálu `topreality.sk`. V tejto adrese sa potom len menia parametre a vieme takto prechádzať podstránky so zoznamami nehnuteľností.

Ako ďalší v poradí je cyklus v `tele`, ktorého získavame informácie z podstránky už konkrétneho inzerátu. Postupne prechádzame objekty s triedou `card-title`, ktorá obsahuje link na podstránku inzerátu.

Na podstránke inzerátu vytiahneme z dát `html` tabuľku a z nej vyberieme informácie z ktorých následne tvoríme dataset.

Keď máme vytiahnuté informácie z inzerátu, dávame pauzu pred ďalšou požiadavkou na server, aby sme nespomaľovali stránku.

B.2 Skript `main`

Súbor je formátu Jupyter Notebook. Každá bunka s kódom obsahuje popis akú funkcionality spúšťa. Bunky treba pre správne fungovanie spúšťať v chronologickom poradí.

B.2.1 Prvá bunka

Prvá bunka skriptu slúži pre načítanie datasetov do pamäte.

Zdrojový kód B.1: Načítanie datasetu

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sys

data_apartmans = pd.read_csv("predaj_byty.csv")
data_houses = pd.read_csv("predaj_domy.csv")
data_combined = pd.read_csv("predaj_spolu.csv")
```

B.2.2 Druhá bunka

Druhá bunka slúži pre preskúmanie dátových sád, ich veľkosť, hodnoty.

Zdrojový kód B.2: Preskúmanie dát

```
data_apartmans.shape,
data_houses.shape,
data_combined.shape
data_combined.head()
data_combined.tail()
data_combined.info()
data_combined.describe()
data_combined.state.value_counts()
```

B.2.3 Tretia bunka

Tretia bunka slúži na zmazanie nepotrebných atribútov a zmenu atribútu state na hodnotu novostavba.

Zdrojový kód B.3: Mazanie atribútov

```
data_combined.loc[data_combined["state_of_real_estate"] == '
    Novostavba', "state"] = 'novostavba'
data_combined.drop(['state_of_real_estate', 'type_of_ownership',
    'price_for_m2'], axis=1, inplace=True)

data_houses.loc[data_houses["state_of_real_estate"] == '
    Novostavba', "state"] = 'novostavba'
data_houses.drop(['state_of_real_estate', 'type_of_ownership',
    'price_for_m2', 'floor', 'elevator', 'balcony', 'basement']
```



```
, 'material'], axis=1, inplace=True)

data_apartmans.loc[data_apartmans["state_of_real_estate"] == '
    Novostavba', "state"] = 'novostavba'
data_apartmans.drop(['state_of_real_estate', '
    type_of_ownership', 'price_for_m2'], axis=1, inplace=True)

data_combined.shape
```

B.2.4 Štvrtá bunka

Pred spustením štvrtej bunky je najprv potrebné spustiť script `get_all_cities`. Slúži na zmenu mestských častí na obyčajné mestá.

Zdrojový kód B.4: Zmena názvov miest

```
with open('city.txt', 'r') as f:
    cities = f.read().splitlines()

for city in cities:
    data_apartmans.loc[
        data_apartmans['locality'].str.contains('{} '.format(city)
        ),
        ['locality']
    ] = city
```

B.2.5 Piata bunka

Piata bunka slúži pre zmazanie všetkých českých miest.

Zdrojový kód B.5: Odstránenie českých miest

```
# get all non cz cities
np.set_printoptions(threshold=sys.maxsize)
not_sk_city = data_combined

with open('city.txt', 'r') as f:
    cities = f.read().splitlines()

for city in cities:
    not_sk_city = not_sk_city[not_sk_city["locality"].str.
        contains(city)==False]
```

```
# remove all czech cities
for city in not_sk_city["locality"].unique():
    data_combined = data_combined[data_combined["locality"].str.
        contains(city)==False]
    data_houses = data_houses[data_houses["locality"].str.
        contains(city)==False]
    data_apartmans = data_apartmans[data_apartmans["locality"].
        str.contains(city)==False]
```

B.2.6 Šiesta bunka

Šiesta bunka je pre odstánenie odľahlých hodnôt z datasetov. Obsahuje funkciu `find_outliers_IQR`, ktorá vyhľadáva odľahlé hodnoty pomocou medzikvartilového rozptylu.

Zdrojový kód B.6: Funkcia pre odstránenie odľahlých hodnôt

```
def find_outliers_IQR(df):
    q1=df.quantile(0.25)
    print(q1)
    q3=df.quantile(0.75)
    print(q3)
    IQR=q3-q1
    print(IQR)
    print(q1-1.5*IQR)
    print(q3+1.5*IQR)
    outliers = df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]
    return outliers

combined_outliers = find_outliers_IQR(data_combined["price"
    ])
houses_outliers = find_outliers_IQR(data_houses["price"])
apartmans_outliers = find_outliers_IQR(data_apartmans["price"
    ])

combined_without_oltliers = data_combined[~data_combined["
    price"].isin(combined_outliers)]
houses_without_oltliers = data_houses[~data_houses["price"].
    isin(houses_outliers)]
```

```
apartmans_without_oltliers = data_apartmans[~data_apartmans[
    "price"].isin(apartmans_outliers)]
```

B.2.7 Siedma bunka

Siedma bunka slúži pre zmazanie vzoriek, ktoré nemajú cieľovú hodnotu.

Zdrojový kód B.7: Zmena názvov miest

```
combined_without_oltliers.dropna(subset=['price'], inplace=
    True)
houses_without_oltliers.dropna(subset=['price'], inplace=True)
apartmans_without_oltliers.dropna(subset=['price'], inplace=
    True)
```

B.2.8 Osma bunka

Osma bunka slúži pre vykreslenie histogramu cien datasetu combined.

Zdrojový kód B.8: Histogram

```
fig, ax = plt.subplots(figsize=(10,5))
sns.histplot(combined_without_oltliers["price"], kde = True)
ax.ticklabel_format(useOffset=False, style='plain')
plt.tight_layout()
plt.show()
```

B.2.9 Deviata bunka

Deviata bunka je pre vykreslenie korelačných grafov

Zdrojový kód B.9: Korelačné grafy

```
list = ["real_estate_type", "state", "material"]

for parameter in list:
    non_nan_without_oltliers_apartmans =
        apartmans_without_oltliers[apartmans_without_oltliers[
            parameter].notna()]
    non_nan_without_oltliers_houses = houses_without_oltliers[
        houses_without_oltliers[parameter].notna()]

    fig, ax = plt.subplots(figsize=(20,10))
```

```
ax.ticklabel_format(useOffset=False, style='plain')
plt.tight_layout()

plt.scatter(non_nan_without_outliers_apartmans["price"],
            non_nan_without_outliers_apartmans[parameter], color = "
            blue")
plt.scatter(non_nan_without_outliers_houses["price"],
            non_nan_without_outliers_houses[parameter], color = "red"
            )

plt.xlabel('price')
plt.ylabel(parameter)
plt.rc('ytick', labels=20)
plt.rc('xtick', labels=20)
plt.show()
```

B.2.10 Desiata bunka

Desiata bunka je pre zobrazenie korelačnej matice datasetu apartmans.

Zdrojový kód B.10: korelačná matica

```
corr_matrix = apartmans_without_outliers.corr()
fig, ax = plt.subplots(figsize=(20,10))
sns.heatmap(corr_matrix, annot=True, linewidths=1, ax=ax)
plt.subplots_adjust(left=0.1, right=0.9, top=0.9, bottom=0.1)
plt.show()
```

B.2.11 Jedenásta bunka

Jedenásta bunka je pre premiešanie vzoriek datasetov a rozdelenie datasetov na X-ové a y-ové hodnoty.

Zdrojový kód B.11: Premiešanie vzoriek a rozdelenie dát

```
data_apartmans = apartmans_without_outliers
data_houses = houses_without_outliers
data_combined = combined_without_outliers

data_apartmans = data_apartmans.sample(frac = 1)
X_apartmans = data_apartmans.drop("price", axis=1)
y_apartmans = data_apartmans["price"]
```

```
data_houses = data_houses.sample(frac = 1)
X_houses = data_houses.drop("price", axis=1)
y_houses = data_houses["price"]

data_combined = data_combined.sample(frac = 1)
X_combined = data_combined.drop("price", axis=1)
y_combined = data_combined["price"]
```

B.2.12 Dvanásta bunka

Dvanásta bunka rozdeľuje atribúty na kategorické a numerické hodnoty. Ďalej vytvára pipelines, ktoré implementujú SimpleImputer, OneHotEncoder a StandardScaler.

Zdrojový kód B.12: Kódovanie numerických a kategorický atribútov

```
from math import remainder
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder,
    StandardScaler, MinMaxScaler
from sklearn.compose import make_column_transformer
from sklearn.pipeline import Pipeline

apartmans_num_features = [
    "usable_area",
    "land",
    "build_up_area"
]

apartmans_cat_features = [
    "locality",
    "street",
    "state",
    "real_estate_type",
    "material",
    "floor",
    "elevator",
    "balcony",
    "basement",
```

```
]

houses_num_features = [
    "usable_area",
    "land",
    "build_up_area"
]

houses_cat_features = [
    "locality",
    "street",
    "state",
    "real_estate_type",
]

numeric_transformer = Pipeline(
    steps=[
        ("imputer", SimpleImputer(strategy="mean")),
        ("scaler", MinMaxScaler())
    ]
)

categorical_transformer = Pipeline(
    steps=[
        ("imputer", SimpleImputer(strategy="constant",
            fill_value="missing")),
        ("ohe", OneHotEncoder(handle_unknown="ignore"))
    ]
)

houses_preprocessor = ColumnTransformer(
    transformers=[
        ("num", numeric_transformer, houses_num_features),
        ("cat", categorical_transformer, houses_cat_features),
    ]
)

apartmans_preprocessor = ColumnTransformer(
    transformers=[
        ("num", numeric_transformer, apartmans_num_features),
```

```

        ("cat", categorical_transformer,
         apartmans_cat_features),
    ]
)

combined_preprocessor = ColumnTransformer(
    transformers=[
        ("num", numeric_transformer, apartmans_num_features),
        ("cat", categorical_transformer,
         apartmans_cat_features),
    ]
)

```

B.2.13 Trinásta bunka

Trinásta bunka slúži pre importovanie a vytvorenie regresorov. Do premennej model ukládame regresor, ktorý chceme testovať.

Zdrojový kód B.13: Zadefinovanie regresorov

```

from sklearn.svm import SVR
from sklearn.linear_model import Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import ElasticNet
from sklearn.linear_model import Ridge

svr = SVR()
lasso = Lasso()
forest = RandomForestRegressor()
elastic = ElasticNet()
ridge = Ridge()

model = elastic

```

B.2.14 Štrnásta bunka

Štrnásta bunka vyvára pipelines ktoré prepájajú predspracovanie dát a model.

Zdrojový kód B.14: Pipelines

```

from sklearn import set_config

```

```
apartmans_pipe = Pipeline(steps=[
    ('preprocessor', apartmans_preprocessor),
    ('model', model)
])

houses_pipe = Pipeline(steps=[
    ('preprocessor', houses_preprocessor),
    ('model', model)
])

combined_pipe = Pipeline(steps=[
    ('preprocessor', combined_preprocessor),
    ('model', model)
])

set_config(display='diagram')
apartmans_pipe
```

B.2.15 Petnásta bunka

Petnásta bunka je pre zadefinovanie funkcie, ktorú sme používali pre výpočet presnosti modelu.

Zdrojový kód B.15: Funkcia pre výpočet metrík

```
from sklearn.metrics import mean_absolute_error,
    mean_squared_error, r2_score
from sklearn.metrics import accuracy_score, precision_score,
    recall_score, f1_score
from sklearn.model_selection import cross_val_score

def evaluate_preds(y_true, y_preds):
    r2 = r2_score(y_true, y_preds)
    mae = mean_absolute_error(y_true, y_preds)
    mse = mean_squared_error(y_true, y_preds)
    metric_dict = {"r2_socre": r2,
                   "mean_absolute_error": mae,
                   "mean_squared_error": mse,
    }
```



```

print(f"R2: {r2:.2f}")
print(f"MAE: {mae:.2f}")
print(f"MSE: {mse:.2f}")
return metric_dict

```

B.2.16 Šetstnásta bunka

Šetstnásta bunka je pre rozdelení dát na tréningové a testovacie, predpovedá hodnoty a vracia presnosť modelu.

Zdrojový kód B.16: Rozdelenie dát, tréning, predikcia, metrika

```

from sklearn.model_selection import train_test_split,
    cross_val_score

X = X_apartmans
y = y_apartmans

X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)
apartmans_pipe.fit(X_train, y_train)

y_preds = apartmans_pipe.predict(X_test)

baseline_metrics = evaluate_preds(y_test, y_preds)

```

B.2.17 Sedemnásť bunka

Sedemnásť bunka je pre ladenie hyperparametrov. Obsahuje slovníky parametrov, ktoré sú potom testované funkciou GridSearchCV.

Zdrojový kód B.17: Ladenie hyperparametrov

```

from sklearn.model_selection import GridSearchCV

forest_params = {
    "preprocessor__num__imputer__strategy": ["mean", "median"],
    "model__n_estimators": [100, 1000],
    "model__max_depth": [None, 5],
    "model__max_features": ["auto"],
    "model__min_samples_split": [2, 4]
}

```

```
}

lasso_params = {'model__alpha':[0.01, 0.1, 1, 10, 100, 1000,
    1500]}
ridge_params = {'model__alpha':[0.01, 0.1, 1, 10, 100, 1000,
    1500]}

elastic_params = {
    "model__max_iter": [1, 5, 10, 100, 1000],
    "model__alpha": [0.0001, 0.001, 0.01, 0.1, 1, 10, 100],
    "model__l1_ratio": np.arange(0.0, 1.0, 0.1)
}

svr_params = {
    'model__kernel': ['rbf', 'sigmoid'],
    'model__C': [0.01, 0.1, 1.0, 10.0, 100.0, 1000.0],
    'model__epsilon': [0.1, 0.3, 0.7, 1.0, 1.3],
    'model__gamma': [1.0, 0.1, 0.01, 0.001, 0.0001, 0.00001]
}

params = elastic_params

grid_search = GridSearchCV(apartmans_pipe, params, cv=5,
    verbose=2)

grid_search.fit(X_apartmans, y_apartmans)
```

B.2.18 Osemnásta bunka

Osemnásta bunka je pre výpis najlepšieho regresora, ktorý našiel GridSearchCV, jeho skóre a najlepšie hyperparametre.

Zdrojový kód B.18: Najlepšie parametre

```
print(grid_search.best_estimator_)
print(grid_search.best_score_)
print(grid_search.best_params_)
```

B.2.19 Devetnásta bunka

Devetnásta bunka je pre uloženie a načítanie modelu, pre ďalšie použitie.

Zdrojový kód B.19: Uloženie a načítanie modelu

```
from joblib import dump, load
dump(forest, filename="gs_random_forest_model_1.pkl")
loaded_pickle_model = load(filename="gs_random_forest_model_1.
   .pkl")
```

B.2.20 Dvadsať bŕnka

Dvadsať bŕnka je pre predikciu načítanej hodnoty z CSV súbora.

Zdrojový kód B.20: Uloženie a načítanie modelu

```
input = pd.read_csv("input_apartmans.csv")
y_pred = apartmans_pipe.predict(input)
y_pred
```