SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

KLASIFIKÁCIA DÁT METÓDOU RANDOM DECISION FOREST

SEMESTRÁLNA PRÁCA

2025 Bc. Lukáš Patrnčiak, Bc. Andrej Tomčík

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

KLASIFIKÁCIA DÁT METÓDOU RANDOM DECISION FOREST

SEMESTRÁLNA PRÁCA

Študijný program:	Multimediálne informačné a komunikačné technológie
Študijný odbor:	Informatika
Školiace pracovisko:	Ústav informačných a komunikačných technologií
Vedúci práce:	doc. Ing. Juraj Kačur, PhD.

Bratislava 2025 **Bc. Lukáš Patrnčiak, Bc. Andrej Tomčík**

SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program:	Multimediálne informačné a komunikačné technológie
Autor:	Bc. Lukáš Patrnčiak, Bc. Andrej Tomčík
Semestrálna práca:	Klasifikácia dát metódou Random Decision Forest
Vedúci práce:	doc. Ing. Juraj Kačur, PhD.
Miesto a rok predloženia práce:	Bratislava 2025

Táto práca sa zameriava na problematiku klasifikácie dát pomocou metód rozhodovacích stromov a náhodných lesov (Random Forests). V práci je popísaná teoretická podstata rozhodovacích stromov, entropie a informačného zisku ako základných princípov klasifikácie. Následne je implementovaný vlastný klasifikátor na báze rozhodovacích stromov, ako aj ensemble model Random Forest, ktorý zvyšuje robustnosť a presnosť pomocou baggingu. Úlohou modelov bolo predikovať cenovú kategóriu leteniek na základe rôznych atribútov. Výsledky sú vyhodnotené pomocou presnosti a konfúznych matíc, pričom Random Forest preukázal vyššiu klasifikačnú presnosť oproti samostatnému rozhodovaciemu stromu.

Kľúčové slová: klasifikácia dát, random forest, strojové učenie, rozhodovacie stromy

Obsah

Zoznam obrázkov a tabuliek

Zoznam skratiek

CSV	Comma-separated values
G	Graf
id3	Iterative Dichotomiser 3
T	Koreňový strom

Zoznam algoritmov

Úvod

Rozhodovacie stromy a ensemble metódy, ako napríklad Random Forest, patria medzi najčastejšie používané klasifikačné algoritmy v oblasti strojového učenia. Ich popularita spočíva v jednoduchej interpretovateľnosti rozhodovania, schopnosti pracovať s rôznorodými údajmi a vysokej presnosti v praxi.

Táto práca sa zameriava na úlohu klasifikácie cenových kategórií leteniek na základe vstupných údajov, ako sú typ letenky, počet prestupov, doba do odletu a iné relevantné atribúty. Hlavným cieľom bolo vytvoriť a implementovať vlastné verzie rozhodovacieho stromu a metódy Random Forest bez použitia predpripravených modelov knižníc.

V práci sa nachádza podrobný opis teórie rozhodovacích stromov, výpočtu entropie a informačného zisku, ako aj metódy baggingu, ktorá tvorí základ algoritmu Random Forest.

Po spracovaní dátového súboru boli modely trénované na kategorizovaných údajoch a ich výkonnosť bola vyhodnotená pomocou presnosti a analýzy konfúznych matíc. Výsledky ukazujú, že ensemble prístup Random Forest dosahuje vyššiu presnosť v porovnaní s jednotlivým rozhodovacím stromom, čo potvrdzuje jeho výhody v klasifikačných úlohách s rôznorodými dátami.

1 Základné pojmy

V tejto kapitole budú popísané základné matematické pojmy a definície, ktoré sú základom celej práce, pričom terória a príklady vychádzajú z [1, 2, 3].

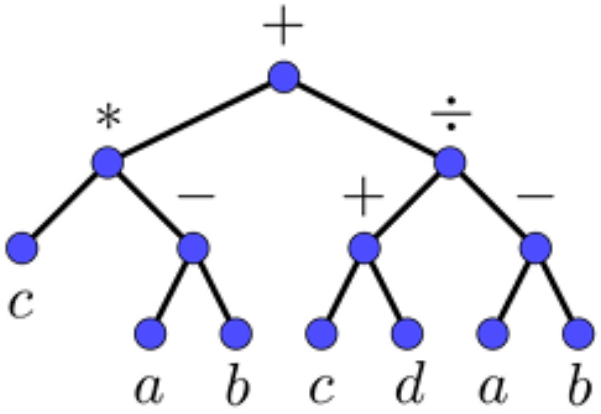
Definícia 1.1 (Graf) *Usporiadaná dvojica $G = (V, E)$ sa nazýva graf. Prvky množiny V sa nazývajú vrcholy grafu G. Prvky množiny E sa nazývajú hrany grafu G a zodpovedajú jedno a dvojprvkovým podmnožinám množiny V.*

Definícia 1.2 (Orientovaný graf) *Majme graf $G = (V, E)$. Ak $u, v \in V : h = (u, v)$, kde $u \neq v$, čiže orientovaná hrana h spája vrcholy u a v, pričom vrchol u je počiatočný a vrchol v je koncový vrchol hrany h. Ak $h = (u,u)$, tak hovoríme, že orientovaná hrana h je slučka.*

Definícia 1.3 (Strom) *Súvislý a acyklický graf (medzi ľubovoľnými dvoma vrcholmi $u, v \in V$ existuje práve jedna cesta) sa nazýva strom. Hrany tohto grafu sa nazývajú vetvy.*

Definícia 1.4 (Koreňový strom, koreň) *Koreňový strom T je strom s pevne vybraným vrcholom u, ktorý sa nazýva koreň, pričom ostatné vrcholy sa nazývajú listy (alebo uzly) stromu.*

Príklad 1.1 *Reprezentácia algebraického výrazu $c*(a-b)+(c+d)/(a-b)$ prostredníctvom koreňového stromu (obr. 1.1).*



Obrázok 1.1: Reprezentácia uvedeného algebraického výrazu koreňovým stromom

1.1 Rozhodovací strom

Ide o metódu aproximácie diskrétnych cieľových funkcií, v ktorej je naučená funkcia reprezentovaná rozhodovacím stromom. Naučené stromy je možné tiež preformulovať ako súbory pravidiel ak-potom, aby sa zlepšila čitateľnosť pre ľudí. Tieto metódy učenia patria medzi najpopulárnejšie algoritmy induktívnej inferencie a boli úspešne aplikované na širokú škálu úloh od učenia sa diagnostikovať zdravotné prípady až po učenie sa posudzovať kreditné riziko žiadateľov o úver. Štruktúru rozhodovacieho stromu môžeme popísať orientovaným grafom vo forme stromu. Funguje na princípe postupného 'riešenia otázok':

- Koreň stromu: východiskový (počiatočný) bod
- Uzol stromu: otázka/podmienka
- Vetvy stromu: možná odpoveď na danú otázku / podmienku, prípadne ďalšia otázka
- Listy stromu: ukazujú konečné rozhodnutie/výstup.

Hoci bolo vyvinutých množstvo metód učenia sa rozhodovacích stromov s trochu odlišnými možnosťami a požiadavkami, učenie sa rozhodovacích stromov je vo všeobecnosti najvhodnejšie pre problémy s nasledujúcimi charakteristikami:

- Inštancie sú reprezentované pármi atribút-hodnota. Inštancie sú opísané pevnou sadou atribútov a ich hodnôt. Najjednoduchšia situácia pre učenie sa rozhodovacích stromov je, keď každý atribút nadobúda malý počet disjunktných možných hodnôt.
- Metódy rozhodovacích stromov sa ľahko rozšíria na učiace sa funkcie s viac ako dvoma možnými výstupnými hodnotami. Výraznejšie rozšírenie umožňuje učenie cieľových funkcií s výstupmi s reálnymi hodnotami, hoci použitie rozhodovacích stromov v tomto prostredí je menej bežné.
- Môžu byť potrebné disjunktívne popisy.
- Trénovacie dáta môžu obsahovať chyby. Metódy učenia sa rozhodovacích stromov sú odolné voči chybám, a to ako chybám v klasifikáciách trénovacích príkladov, tak aj chybám v hodnotách atribútov, ktoré tieto príklady opisujú.
- Trénovacie dáta môžu obsahovať chýbajúce hodnoty atribútov. Metódy rozhodovacích stromov je možné použiť aj vtedy, keď niektoré trénovacie príklady majú neznáme hodnoty.

Dôležitým pojmom v kontexte rozhodovacích stromov je predovšetkým **entropia**, ktorá predstavuje mieru neurčitosti (nečistoty) množiny príkladov. Používa sa pri rozhodovacích stromoch na vyjadrenie, ako 'pomiešaná' je množina vzhľadom na cieľové triedy.

Definícia 1.5 (Entrópia) *Nech S je množina príkladov rozdelená do n tried, pričom p_i je pravdepodobnosť výskytu príkladov triedy i v množine S. Entropia množiny S sa definuje:*

$$Entropy(S) = H(S) = -\sum_{i=1}^{n} p_i \log_2 p_i,$$

kde $0 \log_2 0$ sa chápe ako 0.

Príklad 1.2 *Jednoduché znázorňujúce príklady definície entropie:*

- *Ak všetky príklady patria do jednej triedy: $H(S) = 0$*
- *Ak sú príklady rovnomerne rozdelené medzi dve triedy: $H(S) = 1$*

Informačný zisk udáva, o koľko sa zníži entropia po rozdelení množiny príkladov S podľa nejakého atribútu A. Atribút s najvyšším informačným ziskom je považovaný za najlepší pre rozdelenie v danom uzle stromu.

Definícia 1.6 (Informačný zisk) *Nech atribút A nadobúda hodnoty $v_1, v_2, \ldots, v_k$. Označme S_v podmnožinu príkladov, ktoré majú hodnotu v pre atribút A. Potom informačný zisk pri rozdelení podľa A je:*

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \cdot Entropy(S_v) \tag{1.1}$$

kde:

- *$Values(A)$ je množina všetkých možných hodnôt pre atribút A*
- *S_v je podmnožina S, pre ktorú atribút A má hodnotu v ($S_v = s \in S | A(s) = v$)*

Prvý člen v rovnici 1.1 je iba entropia pôvodnej množiny S a druhý člen je očakávaná hodnota entropie po rozdelení S pomocou atribútu A. Očakávaná entropia opísaná týmto druhým členom je súčet entropií každej podmnožiny S_v, váhovaným zlomkom príkladov, ktoré patria do množiny S_v. Gain(S, A) je teda očakávané zníženie entropie spôsobené znalosťou hodnoty atribútu A, čiže je to informácia poskytnutá o cieľovej hodnote funkcie, vzhľadom na hodnotu nejakého iného atribútu A. Hodnota Gain(S, A) je počet bitov uložených pri kódovaní cieľovej hodnoty ľubovoľného člena S, pričom je známa hodnota atribútu A.

Rozhodovací strom pre klasifikáciu bol vytvorený pomocou nižšie uvedeného základného algoritmu, ktorý sa nazýva aj Iterative Dichotomiser 3 (id3):

Algoritmus 1 ID3(Examples, TargetAttribute, Attributes)

Require: Examples sú trénovacie dáta. TargetAttribute je atribút, ktorého hodnotu má strom predpovedať. Attributes je zoznam ďalších atribútov, ktoré môže testovať naučený rozhodovací strom.

Ensure: Rozhodovací strom klasifikujúci zvolené dáta

1: Vytvor uzol `Root` s atribútom A
2: **if** všetky príklady v Examples majú rovnakú hodnotu TargetAttribute **then**
3: **return** list s touto hodnotou
4: **end if**
5: **if** Attributes je prázdna množina **then**
6: **return** list s najčastejšou hodnotou TargetAttribute v Examples
7: **end if**
8: Vyber atribút $A \in$ Attributes, ktorý najlepšie klasifikuje Examples
9: Rozhodovací atribút pre koreň Root $\leftarrow$ A
10: **for** každú hodnotu v_i atribútu A **do**
11: Pridať novú vetvu stromu pod koreň Root, zodpovedajúcu testu $A = v_i$
12: Nech Examples_{v_i} je podmnožina príkladov, ktorá má hodnotu v_i pre A
13: **if** Examples_{v_i} je prázdna **then**
14: Potom pod túto novú vetvu pridať koncový uzol s označením = najbežnejšia hodnota TargetAttribute v Examples
15: **else**
16: Pod túto novú vetvu pridať podstrom:
$\text{ID3}(\text{Examples}_{v_i}, \text{TargetAttribute}, \text{Attributes} - \{A\})$
17: **end if**
18: **end for**
19: **return** `Root`

Pojem najlepšia klasifikácia Examples zahŕňa atribút s najvyšším informačným ziskom.

1.2 Random Forest

Random Forest je jedným z najpopulárnejších algoritmov pre ensemble learning, ktorý patrí medzi metódy zlepšovania výkonu modelu kombinovaním viacerých jednoduchých modelov. Tento prístup sa nazýva bagging (Bootstrap Aggregating).

Bagging je metóda, ktorá kombinuje viacero modelov (zvyčajne rovnakého typu), pričom každý model je natrénovaný na náhodne bootstrapovanej (so spätným výberom) podmnožine trénovacích dát. Výsledná predikcia sa určí agregáciou výstupov všetkých modelov (hlasovaním alebo priemerom).

Definícia 1.7 (Bagging) *Nech $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ je trénovacia množina, a nech B je počet klasifikátorov v ensemble. Pre každý $b = 1, \ldots, B$:*

- *Vytvoríme bootstrap vzorku $\mathcal{D}^{*b}$ výberom s opakovaním z $\mathcal{D}$,*
- *Natrénujeme klasifikátor $\hat{f}^{*b}$ (rozhodovací strom) na $\mathcal{D}^{*b}$,*
- *Získame predikciu $\hat{f}^{*b}(x)$ pre nový vstup x.*

Výsledná baggingová predikcia je určená väčšinovým hlasovaním:

$$\hat{f}_{bag}(x) = majority\_vote\left(\hat{f}^{*1}(x), \ldots, \hat{f}^{*B}(x)\right)$$

Táto technika znižuje rozptyl a zvyšuje robustnosť modelu, čím zlepšuje generalizáciu.

Príklad 1.3 *Predstavme si úlohu klasifikácie, kde chceme na základe údajov (napr. trieda, počet dní do odletu, počet prestupov) predikovať, aký typ letenky si cestujúci kúpi:* ***Low****,* ***Medium*** *alebo* ***High****. Trénovali sme 5 rozhodovacích stromov, každý na inej bootstrap vzorke. Pre nového cestujúceho s vlastnosťami:*

- *Trieda: Business*
- *Počet dní do odletu: 3*
- *Prestupy: 1*

sú výstupy jednotlivých stromov nasledovné:

Strom	**Predikcia**
$f^{*1}(x)$	Low
$f^{*2}(x)$	Medium
$f^{*3}(x)$	Low
$f^{*4}(x)$	High
$f^{*5}(x)$	Low

Tabuľka 1.1: Predikcie jednotlivých stromov pre vstup x

Majoritné hlasovanie:

- *Low: 3-krát*
- *Medium: 1-krát*
- *High: 1-krát*

Baggingová predikcia *pre vstup x je teda:*

$$\hat{f}_{bag}(x) = Low$$

Aj keď niektoré stromy poskytli rôzne výstupy, väčšinové hlasovanie zabezpečilo stabilnú a robustnú predikciu.

Random Forest je tvorený množstvom rozhodovacích stromov, ktoré pracujú spoločne a vytvárajú silný model.

Algoritmus 2 Random Forest pre klasifikáciu

Require: Trénovacia množina $\mathcal{Z} = \{(x_i, y_i)\}_{i=1}^{N}$, počet stromov B, počet atribútov m (pre split), minimálna veľkosť uzla $n_{\min}$

Ensure: Ensemble stromov $\{T_b\}_{b=1}^{B}$

1: **for** $b = 1$ to B **do**
2: Vytiahni bootstrap vzorku $\mathcal{Z}^*$ veľkosti N z trénovacích dát
3: Trénuj náhodný strom T_b na $\mathcal{Z}^*$ nasledovne:
4: **while** aktuálny uzol nie je terminálny a obsahuje aspoň $n_{\min}$ vzoriek **do**
5: Náhodne vyber m atribútov z celkových p
6: Vyber najlepší split medzi týmito m atribútmi (napr. podľa entropie alebo Gini indexu)
7: Rozdeľ uzol na dve dcérske vetvy
8: **end while**
9: **end for**
10: **return** Ensemble stromov $\{T_b\}_{b=1}^{B}$

Predikcia pre nový vstup x**:** Nech $\hat{C}_b(x)$ je predikcia b-tého stromu pre x. Potom baggingová predikcia Random Forestu je:

$$\hat{C}_{\text{rf}}^{B}(x) = \text{majority vote}\left\{\hat{C}_b(x)\right\}_{b=1}^{B}$$

Poznámka 1 *V algoritme označujeme každý strom ako T_b, čo predstavuje trénovaný rozhodovací strom (funkciu). Pri predikcii nového vstupu x sa často zapisuje predikcia tohto stromu ako $\hat{C}_b(x)$, pričom platí:*

$$\hat{C}_b(x) = T_b(x)$$

Teda trieda predikovaná stromom b-tej iterácie je výstupom samotného modelu T_b pre vstup x.

Príklad 1.4 *Predpokladajme, že máme $B = 5$ stromov a každý predikuje pre vstup x nasledovné triedy:*

Strom	**Predikcia** $\hat{C}_b(x)$
1	Low
2	Medium
3	Low
4	High
5	Low

Tabuľka 1.2: Predikcie jednotlivých stromov pre vstup x

Spočítanie výskytu tried:

- *Low: 3-krát*
- *Medium: 1-krát*
- *High: 1-krát*

Výsledná predikcia pomocou majority vote:

$$\hat{C}_{rf}(x) = Low$$

2 Spracovanie a analýza dát

V tejto kapitole bude popísaný spôsob načítania a prípavy dát, s ktorými budeme pracovať neskôr v našich vytvorených modelov. Súbor s dátami (vo formáte CSV[1]), pričom dáta v tomto súbore obsahujú nasledovné stĺpce:

- *ID* - identifikátor zoznamu.
- *Airline* - názov leteckej spoločnosti
- *Flight* - identifikátor letu
- *Source city* - mesto, z ktorého lietadlo odlieta
- *Departure time* - časový rámec odletu lietadla
- *Stops* - počet zastávok letu
- *Arrival time* - časový rámec príletu lietadla
- *Destination city* - mesto, kde lietadlo pristane.
- *Class* - trieda cestovania
- *Duration* - dĺžka letu
- *Days left* - počet dní do odletu
- *Price* - cena letenky

Taktiež je potrebné spomenúť, že aplikácia vybraných algoritmov pre rozhodovací strom a Random Forest metódu bola realizovaná prostredníctvom programovacieho jazyka Python s použitím nasledovných knižníc: [4]

- **NumPy**[2]: Pre matematické výpočty.
- **Pandas**[3]: Manipulácia s tabuľkovými dátami a ich analýza.
- **Seaborn**[4]: Vysokoúrovňová vizualizácia dát so zameraním na štatistiku.
- **Matplotlib.pyplot**[5]: Základná knižnica na tvorbu grafov s nízkoúrovňovou kontrolou nad vizualizáciami.
- **skLearn**[6]: Knižnica určená pre klasické metódy strojového učenia. V tomto prípade bola použitá pre rozdelenie dát do množín a vykreslenie konfúznej matice
- **collections**[7], textbfdataclasses[8]

[1] jednoduchý súborový formát, ktorý sa skladá z riadkov, v ktorých sú jednotlivé položky oddelené znakom. Takýto formát slúži na výmenu tabuľkových dát.

[2] https://numpy.org/

[3] https://pandas.pydata.org/

[4] https://seaborn.pydata.org/

[5] https://matplotlib.org/stable/api/pyplot_summary.html

[6] https://scikit-learn.org/stable/

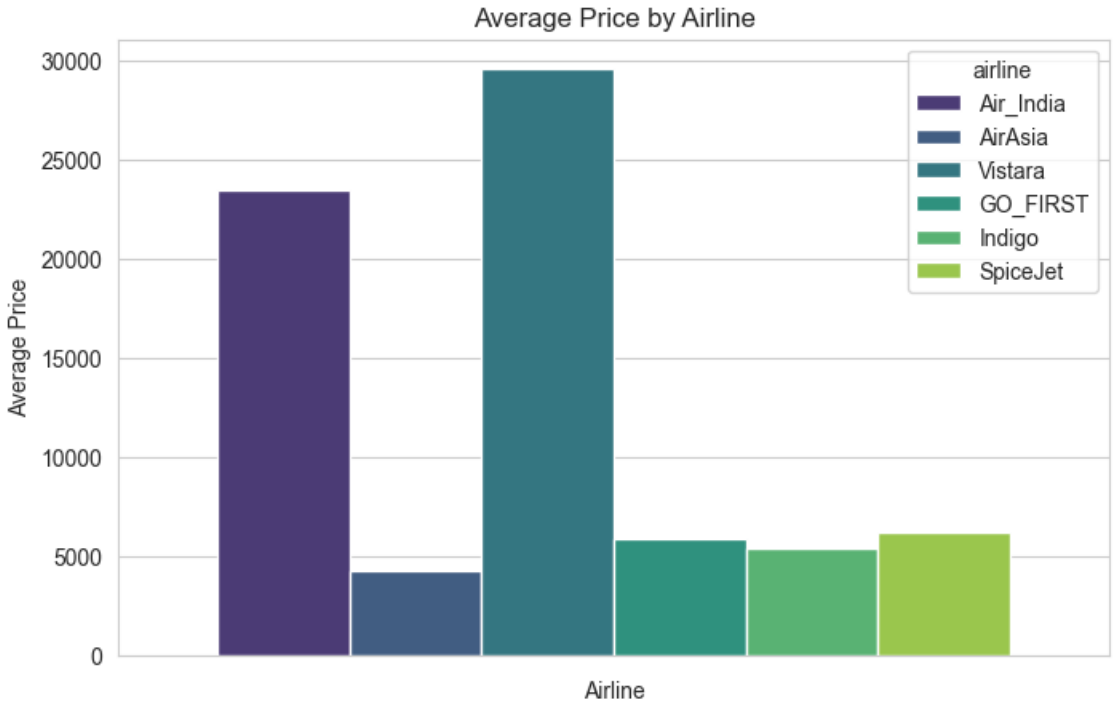
[7] https://docs.python.org/3/library/collections.html

[8] https://docs.python.org/3/library/dataclasses.html

Po načítaní dát z daného súboru nasledoval ich výpis, aby sme zistili informácie o nich - koľko chýbajúcich hodnôt a duplicít obsahujú a ich celkový počet. Na začiatku bolo celkovo k dispozícii 50000 riadkov dát v súbore, pričom po pretriedení ich ostalo 36638. Taktiež boli odstránené stĺpce 'ID' a 'Flight', ktoré sú pre účel tejto práce zbytočné.

Keďže nami implementovaný algoritmus rozhodovacieho stromu dokáže pracovať len s diskrétnymi a nie spojitými hodnotami, bolo potrebné číselné stĺpce kategorizovať do troch kvantilov nasledovným spôsobom:

- **Cena ('price'):** kategorizovaná ako low - malá, medium - stredná a hight - veľká
- **Dĺžka cesty ('duration'):** kategorizovaná ako short - krátka, medium - stredná a long - dlhá
- **Počet dní do odletu ('days_left'):** kategorizované ako last_minute - posledná chvíľa, soon - čoskoro a flexible - flexibilné



Obrázok 2.1: Graf znázorňujúci závislosť priemernej ceny a leteckej spoločnosti

Dáta boli následne rozdelené na vstupnú množinu X (obsahujúcu všetky množiny okrem 'price') a cieľovú / výstupnú množinu Y (obsahujúcu cieľovú premennú 'price'). Následne boli dáta rozdelené na trénovaciu a testovaciu množinu, pričom pre testovaciu množinu bolo vyhradených 30 % dát.

Cieľom tohto rozdelenia je trénovať model na jednej množine a vyhodnotiť ho na nezávislej testovacej množine, čím sa overí, ako dobre model generalizuje. Taktiež bolo nastavené, aby sa dáta rovnako rozdelili pri každom spustení.

3 Aplikácia algoritmov

V tejto kapitole budú popísané jednotlivé parametre vytvorených modelov a taktiež aj výstup z nich.

3.1 Rozhodovací strom

V prípade rozhodovacieho stromu bola vytvorená základná trieda `TreeNode`, ktorá predstavuje uzol v našom rozhodovacom strome. Táto trieda obsahuje

- `atribút (attribute)`: Textový názov atribútu (napr. 'airline', 'duration') podľa ktorého sa strom vetví v tomto uzle. Používa sa len vo vnútorných (rozhodovacích) uzloch. Ak je uzol listový (koncový), atribút zostáva prázdny - pretože sa podľa ničoho už nerozhoduje.
- `potomka (child)`: ide o štruktúru v tvare kľúč: hodnota, ktorá mapuje hodnotu atribútu na ďalší uzol stromu (TreeNode). Ide o reprezentáciu vetvy stromu, ktoré vznikajú na základe rôznych hodnôt atribútu. Ak je uzol list, potomok je prázdna hodnota - pretože nemá žiadne potomky, žiadne vetvy.
- `hodnotu (value)`: Ide o predikovanú hodnotu triedy pre daný listový uzol (napr. 'low' ak uzol predikuje, že cena je nízka).

Následne bola vytvorená hlavná trieda `DecisionTree`, ktorá obsahuje konkrétnu implementáciu nášho rozhodovacieho stromu. Na začiatok je potrebné popísať premenné, ktoré inicializuje konštruktor:

- `max_depth`: maximálna povolená hĺbka stromu.
- `random_state`: náhodné semienko (napr. pre výber atribútov).
- `max_features`: koľko atribútov maximálne uvažovať pri rozhodovaní.

Ďalej konštruktor nastaví:

- `self.majority_class` – predvolená trieda pre prípady, keď strom nedokáže rozhodnúť.
- `self.tree` – bude uchovávať koreň stromu.
- `self.rng` – generátor náhodných čísel pre replikovateľnosť (spustenie toho istého kódu s rovnakými vstupmi vždy vygeneruje rovnaký výsledok).

V tejto triede je taktiež implementovaných niekoľko metód, ktoré budú popísané v podkapitolách nižšie.

3.1.1 Výpočet entropie a informačného zisku

Výpočet entropie a informačného zisku prebieha v pomocných metódach `_entropy()` a `_information_gain()`, pričom tieto výpočty boli navrhnuté podľa kapitoly 1.1. **Výpočet entropie prebieha nasledovným spôsobom**:

1. Spočítanie, koľkokrát sa vyskytuje každá trieda.
2. Výpočet relatívnej frekvencie triedy pravdepodobnosť výskytu triedy)

$$\text{Relatívna frekvencia triedy} = \frac{\text{Počet výskytov triedy}}{\text{Celkový počet vzoriek}}$$

3. Výpočet entropie podľa definície 1.5

Výpočet informačného zisku je zas realizovaný v nasledovných krokoch:

1. Spočítanie entropie pred rozdelením
2. Pre každú hodnotu atribútu:
   - Zobrať podmnožinu množiny Y, kde sa X zhoduje na tejto hodnote (z Y vybrať tie hodnoty, ktoré zodpovedajú riadkom v X, kde má daný atribút určitú hodnotu).
   - Spočítať entropiu tejto podmnožiny.
   - Vypočítať vážený priemer entropií podľa veľkosti podmnožiny.
3. *Výsledok:* Rozdiel medzi entropiou pred rozdelením a váženou entropiou po rozdelení = informačný zisk.

3.1.2 Trénovanie

Pre natrénovania rozhodovacieho stromu slúži metóda `fit(self, x, y)`, ktorá na základe vstupných dát x a výstupných hodnôt y. Účelom tejto metódy je naučiť model pravidlá rozhodovania zo vstupných dát (atribúty) a cieľových hodnôt (triedy). Táto metóda zahŕňa nasledovné kroky:

1. Kontrola a úprava vstupov x (atribúty) a konverzia výstupov y (triedy): V tejto časti sa vstupné dáta prevedú na zoznam slovníkov, kde každý riadok je ako ätribút": hodnota a výstupy y na obyčajný zoznam.

2. Zistenie najčastejšej triedy v trénovacích dátach: Spočíta výskyty jednotlivých tried. Najčastejšia trieda sa uloží ako záložná predikcia, ak strom nebude vedieť rozhodnúť.

3. Rekurzívne natrénovanie stromu: Spustí sa rekurzívna metóda `_build_tree`, ktorá vybuduje celý rozhodovací strom od koreňa. Začína sa s úplnými dátami a hĺbkou 0.

3.1.3 Budovanie stromu

Pre budovanie stromu slúži metóda `_build_tree(self, x, y, attributes, depth)`. Je to hlavná rekurzívna časť rozhodovacieho stromu, ktorá krok za krokom buduje štruktúru stromu – od koreňa (root node) až po listy (internal/leaf node). Účelom tejto metódy je vybudovať rozhodovací strom na základe trénovacích dát x (atribúty), y (triedy), dostupných atribútov a aktuálnej hĺbky v strome. Systematický popis tejto metódy je nasledovný:

1. Kontrola čistoty dát (zastavenie vetvenia): Ak sú všetky triedy rovnaké, strom už vie s istotou rozhodnúť - vytvorí sa listový uzol (leaf node) s touto triedou.

2. Zastavenie kvôli nedostatku atribútov alebo prekročeniu hĺbky: Ak už nemáme žiadne atribúty na vetvenie alebo sme dosiahli maximálnu hĺbku stromu, vráti sa listový uzol s najčastejšou triedou (zistí sa početnosť tried a vyberie sa názov triedy s najväčšou početnosťou).

3. Výber najlepšieho atribútu na rozdelenie: Na základe informačného zisku sa vyberie atribút, ktorý najlepšie rozdelí dáta (najviac zníži entropiu).

4. Vytvorenie vnútorného uzla stromu: Vytvorí sa nový uzol stromu, ktorý bude vetviť podľa `best_attr`.

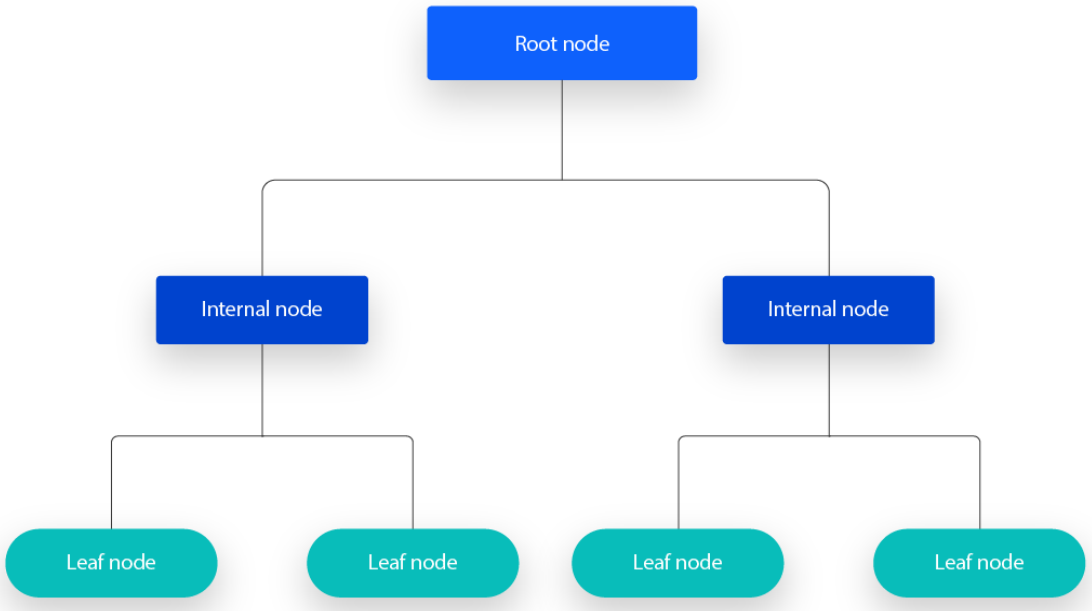
5. Zistenie unikátnych hodnôt atribútu: Vyberú sa všetky rôzne hodnoty vybraného atribútu (napr. `"low", "medium", "high"`).

6. Spracovanie každej hodnoty atribútu – vetvenie: Pre každú hodnotu sa vytvorí podmnožina dát `x_subset` a `y_subset`, kde má atribút hodnotu `val` a strom pre túto vetvu buď skončí listom (ak nie sú dáta) alebo pokračuje ďalej rekurzívne.

   - Výber podmnožiny dát: Vyberie sa časť dát, ktorá zodpovedá konkrétnej hodnote atribútu (napr. `duration == "short"`).
   - Vetva bez dát → záložná trieda: Ak pre danú hodnotu nie sú žiadne dáta (napr. chýbajúce kombinácie), použije sa najčastejšia trieda.
   - Rekurzívne vetvenie – podstrom: Pre danú vetvu sa odstráni sa už použitý atribút (`best_attr`) a strom sa rekurzívne stavia ďalej na podmnožine dát a so zvýšenou hĺbkou.

   Po spracovaní všetkých hodnôt atribútu sa vráti vytvorený uzol (so všetkými vetvami).

Vizualizáciu rozhodovacieho stromu, resp. koreňového uzla, interné uzly až po konečné (listové uzly) je možné vidieť na obrázku 3.1 [5].



Obrázok 3.1: Vizualizácia rozhodovacieho stromu

3.1.4 Výber najlepšieho atribútu

Metóda `_best_attribute(self, x, y, attributes)` má za úlohu nájsť najlepší atribút, podľa ktorého sa budú dáta rozdeliť vo vnútornom uzle rozhodovacieho stromu. Kritériom výberu je najvyšší informačný zisk, ktorý vyjadruje, o koľko sa zníži neusporiadanosť (entropia), keď rozdelíme dáta podľa daného atribútu.

Táto metóda prijíma vstupné dáta x (zoznam slovníkov), výstupné triedy y (zoznam hodnôt) a zoznam názvov atribútov, ktoré sú k dispozícii. Popis metódy je nasledovný:

1. Voliteľný výber náhodnej podmnožiny atribútov (pre Random Forest): Ak je zadané `max_features`, vyberie sa náhodná podmnožina atribútov. Znižuje sa tým korelácia medzi stromami v Random Foreste. Príklad: ak máme 10 atribútov a `max_features` = 3, zvolí náhodne 3.

2. Výpočet informačného zisku pre každý atribút: Pre každý atribút sa vypočíta: entropia pred rozdelením, vážená entropia po rozdelení, rozdiel (informačný zisk), pričom výsledok sa uloží ako dvojica (atribút, zisk).

3. Výber atribútu s najväčším ziskom: Začneme s najnižšou možnou hodnotou a následne porovnávame všetky výpočty a nájdeme atribút, ktorý najviac znižuje neistotu v dátach (maximalizuje `gain`).

Metóda na záver vráti najlepší atribút.

3.1.5 Predikcia

Pre generovanie predikcie tried (výstupov) pre nové vstupné dáta pomocou už natrénovaného rozhodovacieho stromu slúži metóda `predict(self, x)`. Metóda dostáva jediný parameter a to x, čo sú vstupné dáta na predikciu vo forme `pandas.DataFrame`. Princíp metódy spočíva v nasledovných krokoch:

1. Inicializácia zoznamu predikcií: Ukladanie predikovaných hodnôt (triedy) pre jednotlivé vzorky.

2. Ak je vstup vo forme DataFrame, prevedie sa na zoznam slovníkov: Každý riadok sa zmení na slovník `"atribút": hodnota`.

3. Predikcia pre každý riadok vstupných dát: Pre každú vzorku sa zavolá rekurzívna metóda `_predict_row`, ktorá prejde stromom od koreňa až po list a vráti triedu, pričom výsledok sa uloží do zoznamu predictions.

Výstupom je zoznam predikovaných tried.

3.1.6 Predikcia pre jeden riadok

Metóda `_predict_row(self, row, node)` slúži na predikciu výstupnej triedy pre jednu konkrétnu vzorku (riadok vstupných dát) pomocou prechodu rozhodovacím stromom. Ide o rekurzívnu metódu, ktorá sa volá opakovane, až kým nedôjde do listového uzla. Metóda prijíma parametre `row` (jedna vzorka vstupných dát (napr. `{"class": "Economy", "duration": "short"}`)) a `node` (aktuálny uzol v rozhodovacom strome (začína sa koreňom)). Princíp tejto metódy spočíva v nasledovných krokoch:

1. Skontrolovať, či sme v listovom uzle: Ak má uzol hodnotu (`value`), znamená to, že je to koncový uzol stromu (list) → vrátime túto predikciu (triedu).

2. Získať hodnotu atribútu, podľa ktorého sa vetví aktuálny uzol: Napr. ak uzol vetví podľa `class`, získame hodnotu z riadku: `"Economy"`.

3. Pokús sa nájsť zodpovedajúceho potomka (vetvu - `child`): Podľa hodnoty atribútu (napr. `"Economy"`) sa pozrieme, či taká vetva existuje medzi `children`.

4. Ak vetva neexistuje (napr. neznáma hodnota alebo chýbajúce údaje): Ak hodnota z `row` sa nevyskytla počas trénovania, strom nevie, ako ďalej - použije sa najčastejšia trieda z trénovacích dát ako záložná predikcia. V opačnom prípade budeme rekurzívne pokračovať na ďalšej úrovni stromu - ak vetva existuje, volá sa metóda znova s podstromom, a proces sa opakuje, kým sa nedostaneme do listu.

3.1.7 Vizualizácia

Metóda `visualize(self, max_depth=None)` slúži na textovú (konzolovú) vizualizáciu rozhodovacieho stromu, ktorý bol vytvorený pomocou `fit()`. Umožňuje prehľadne zobraziť, ako strom vyzerá – ako sa vetví a aké rozhodnutia robí. Metóda dostane na vstup parameter `max_depth` (voliteľný parameter), ktorý udáva maximálnu hĺbku, ktorú chceme zobraziť. Ak je `None`, zobrazí sa celý strom. Táto metóda obsahuje pomocnú vnorenú funkciu `_print_node(node, depth)`, ktorá postupne prejde celý strom. Táto vnorená funkcia pozostáva zo vstupov `node` (aktuálny uzol stromu, ktorý sa má vypísať) a `depth` (úroveň hĺbky stromu (na odsadenie riadkov)). Princíp tejto vnorenej funkcie je nasledovný:

1. Kontrola maximálnej hĺbky: Ak je zadané obmedzenie hĺbky (`max_depth`) a sme už hlbšie, zastaví sa výpis tejto vetvy.

2. Príprava odsadenia: Každý uzol sa odsádza podľa hĺbky, aby štruktúra pripomínala strom.

3. Vypísanie uzla (zobrazí, podľa ktorého atribútu sa rozhoduje):
   - ak je uzol listový (obsahuje predikciu),
   - ak je uzol vnútorný (obsahuje atribút na vetvenie).

4. Rekurzívne vypísanie potomkov uzla: Pre každú hodnotu aktuálneho atribútu zobrazí podmienku (napr. `If duration == short:`) a následne zavolá `_print_node()` na zobrazenie ďalšej úrovne stromu.

5. Spustenie výpisu od koreňa stromu: Zobrazí nadpis a spustí rekurzívne prechádzanie a výpis od koreňa.

Na obrázku 3.2 je možné vidieť textovú vizualizáciu rozhodovacieho stromu, ktorá je realizovaná prostredníctvom funkcie `visualize()`. Každý uzol označený `[N] Attribute: ...` je rozhodovací uzol, ktorý sa pýta na hodnotu daného atribútu, každá vetva (napr. `If class == Business:`) predstavuje podmienku, ktorou sa rozhoduje. Uzly `Predict: high` sú listové uzly – tam strom končí a vráti predikciu (napr. cena letenky je 'high').

Na záver kapitoly je potrebné poznamenať, že výsledné predikcie dosiahnuté rozhodovacím stromom boli uložené do súboru vo formáte CSV, ktorý obsahuje 2 stĺpce - aktuálne dáta (skutočné hodnoty) a predikované dáta (výsledky rozhodovacieho stromu).

```
DECISION TREE VISUALIZATION
 Attribute: class
  └── If class == Business:
   Attribute: stops
    └── If stops == one:
     Prediction: high
    └── If stops == two_or_more:
     Prediction: high
    └── If stops == zero:
     Attribute: source_city
      └── If source_city == Mumbai:
       Prediction: high
      └── If source_city == Delhi:
       Prediction: high
      └── If source_city == Hyderabad:
       Attribute: destination_city
        └── If destination_city == Mumbai:
         Prediction: high
        └── If destination_city == Bangalore:
         Prediction: medium
        └── If destination_city == Delhi:
         Prediction: high
        └── If destination_city == Kolkata:
         Prediction: high
```

Obrázok 3.2: Textová ukážka výpisu rozhodovacieho stromu

3.2 Random Forest

Pre túto metódu bola vytvorená hlavná trieda `RandomForest`, ktorá obsahuje jej implementáciu. Táto metóda pozostáva z nasledovných krokov:

- Z pôvodného tréningového datasetu sa náhodne vyberú vzorky pre každý strom s opakovaním (tzv. bootstrap sampling). Niektoré vzorky sa tak môžu objaviť viackrát, iné vôbec nie.
- Každý strom (alebo dokonca každý uzol v strome) nevidí všetky vstupné atribúty, ale len náhodne vybranú podmnožinu.

Prvý krok v implementácii spočíva v inicializácii všetkých potrebných parametrov a vnútorných premenných, ktoré bude metóda `RandomForest` využívať počas učenia (`fit`) a predikcie (`predict`). Ide o konštruktor celej metódy, pri ktorom je možné nastaviť nasledovné parametre:

1. počet rozhodovacích stromov (`n_estimators`), ktoré budú použité v náhodnom lese (predvolene 10),
2. maximálnu hĺbku jednotlivých stromov (`max_depth`); ak nie je zadaná, stromy môžu rásť neobmedzene až po listové uzly,
3. semienko generátora náhodných čísel (`random_state`), ktoré zabezpečuje reprodukovateľnosť — rovnaké vstupy vedú k rovnakému náhodnému výberu.

V tejto časti sa taktiež vykonáva:

- uloženie počtu stromov (`n_estimators`),
- uloženie maximálnej povolenej hĺbky stromov (`max_depth`),
- inicializácia prázdneho zoznamu `trees`, do ktorého sa budú počas trénovania ukladať všetky natrénované rozhodovacie stromy,
- inicializácia zoznamu `features_idx` (v prípade rozšírenia o náhodný výber atribútov pre každý strom),
- uloženie hodnoty `random_state` pre účely reprodukovateľnosti,
- vytvorenie vlastného generátora náhodných čísel `rng` pomocou knižnice NumPy.

3.2.1 Trénovanie

Tréning modelu prebieha prostredníctvom metódy `fit(self, x, y)` na zadaných vstupných dátach `x` (atribúty) a cieľových hodnotách `y` (triedy). Pre každý strom v lese:

- náhodne vyberie vzorky dát s opakovaním (bootstrap sampling),
- vytvorí a natrénuje samostatný rozhodovací strom (trieda `DecisionTree`),
- a uloží ho do zoznamu `trees`.

Tréningový cyklus: Tréning sa opakuje pre každý strom zvlášť, pričom počet iterácií zodpovedá hodnote `n_estimators`.

1. **Bootstrap sampling:** náhodne sa vyberú indexy vzoriek s opakovaním. To znamená, že niektoré vzorky môžu byť v trénovacej množine viackrát, iné vôbec — čím sa zabezpečí variabilita medzi stromami.
2. **Nastavenie počtu atribútov:** počet atribútov, ktoré sa majú vyberať je nastavený na $\lfloor \sqrt{p} \rfloor$ kde p je počet všetkých atribútov. Je to predovšetkým z dôvodu, aby mal model nižšiu varianciu, menšie riziko pretrénovania, aby bola pridaná určitá náhodnosť a zlepšil sa celkový výkon. Parameter `max_features` umožňuje, aby každý strom pracoval len s náhodnou podmnožinou atribútov.
3. **Výber trénovacej vzorky:** na základe bootstrap indexov sa vytvoria nové podmnožiny `x_sample` a `y_sample`, ktoré slúžia ako vstupy pre trénovanie stromu.
4. **Vytvorenie a trénovanie stromu:** vytvorí sa nová inštancia rozhodovacieho stromu s parametrami `max_depth` a `max_features`, a následne sa natrénuje na priradenej vzorke dát.
5. **Uloženie stromu do lesa:** natrénovaný strom sa pridá do zoznamu `self.trees`, ktorý reprezentuje celý náhodný les.

3.2.2 Predikcia

Predikcia výstupnej triedy prebieha prostredníctvom metódy `predict()`, ktorá vracia konečné predikcie modelu Random Forest pre vstupné dáta `x`. Každý rozhodovací strom v náhodnom lese predikuje samostatne, a výsledné predikcie sa následne kombinujú pomocou hlasovania väčšiny (majority voting). Priebeh predikcie je nasledovný:

1. **Zber predikcií od všetkých stromov:** Každý strom v náhodnom lese predikuje triedu pre všetky vstupné vzorky. Výstupom každého stromu je zoznam predikcií (napr. [`'low'`, `'high'`, `'low'`, ...]). Všetky tieto predikcie sa uložia do zoznamu `predictions`.

2. **Transformácia zoznamu na maticu:** Predikcie sa skonvertujú na maticu a transponujú, aby každému riadku zodpovedali všetky predikcie od rôznych stromov pre tú istú vzorku. Riadky matice teraz reprezentujú jednotlivé vstupné vzorky (napr. letenky) a stĺpce predikcie rôznych stromov.
3. **Hlasovanie väčšiny:** Pre každý riadok (t. j. jednu vzorku) sa zoberú všetky predikcie stromov a určí sa trieda, ktorá sa vyskytuje najčastejšie. Táto trieda sa potom stane finálnou predikciou pre danú vzorku.

Nakoniec sa vráti zoznam konečných predikcií — teda jedna predikovaná trieda pre každý riadok vo vstupnom `x`.
Na záver je vhodné poznamenať, že výsledné predikcie získané modelom Random Forest boli uložené do súboru vo formáte CSV. Tento súbor obsahuje dva stĺpce — skutočné hodnoty a predikované hodnoty (výsledky metódy `predict()`).

3.3 Vyhodnotenie výsledkov

Pre vyhodnotenie výsledkov dosiahnutých modelov boli implementované metódy `_accuracy_score()` a `score()`, ktoré sú implementované v triedach `DecisionTree` a `RandomForest` (avšak sú totožné - preto sú tieot metódy popísané v tejto kapitole). Samotné vyhodnotenie výsledkov prebieha prosterdníctvom prvej spomenutej metódy, pričom postup je nasledovný:

1. Porovnanie predikcií: spočíta sa, koľko predikovaných hodnôt sa zhoduje so skutočnými.
2. Výpočet presnosti a chyby:

$$\text{Presnosť} = \frac{\text{Počet správnych predikcii}}{\text{Všetky predikcie}},$$
$$\text{Chyba} = 1 - \text{Presnosť}$$

Druhá spomenutá metóda slúži len ako pomocná funkcia, ktorá získa potrebné dáta - predikované (pomocou `predict()`, ktorá ich získa na základe vstupných dát z množiny X) a skutočné (množina Y), ktoré pošle do metódy `_accuracy_score()`. Pokiaľ je `return_error` nastavené na 'False', metóda vráti presnosť, inak vráti chybu. Výsledné odhady predností a chýb sú nasledovné:

	Presnosť	Chyba
Rozhodovací strom	86.54%	13.46%
Random Forest	87.81%	12.19%

Tabuľka 3.1: Porovnanie presností a chýb rozhodovacieho stromu a Random Forest

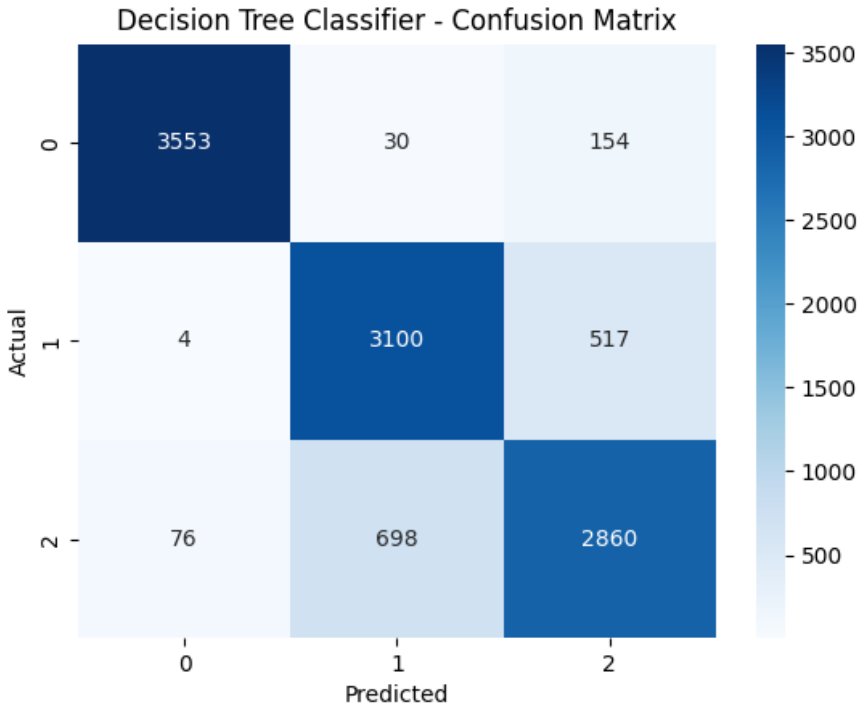
Random Forest prekonal rozhodovací strom o približne 1.27 %, čo potvrdzuje prínos ensemble prístupu a baggingu pri zvyšovaní robustnosti a generalizačnej schopnosti modelu.

Pre vizuálne vyhodnotenie výkonnosti týchto modelov bola použitá konfúzna matica, ktorá ukazuje, koľkokrát model správne alebo nesprávne predpovedal jednotlivé triedy. [6]

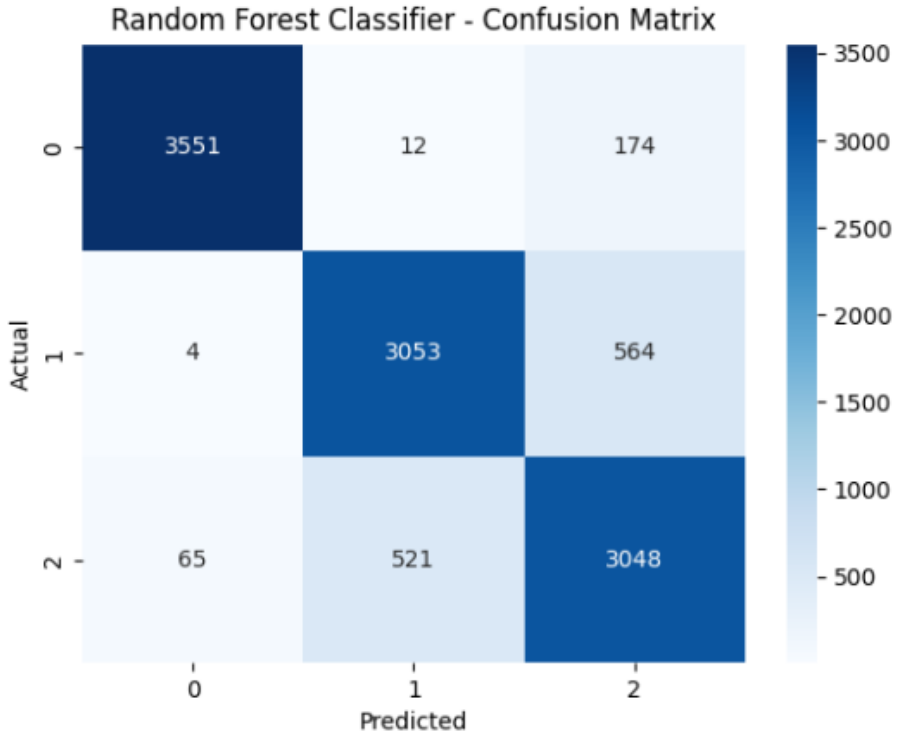
Každý riadok zodpovedá skutočnej triede (Actual) a každý stĺpec zodpovedá predikovanej triede (Predicted). Tieto matice môžeme vidieť na obrázkoch 3.3 a 3.4.

Označenie tried (ceny letenky)

- `0`: trieda "low"
- `1`: trieda "medium"
- `2`: trieda "high"



Obrázok 3.3: Konfúzna matica pre rozhodovací strom



Obrázok 3.4: Konfúzna matica pre Random Forest

Na hlavnej diagonále sa nachádzajú správne detekované predikcie a mimo diagonálny sa nachádzajú nesprávne predikcie (napríklad v prípade rozhodovacieho stromu bolo 698 prípadov triedy 2 (high) nesprávne klasifikovaných ako trieda 1 (medium)).

Z daných obrázkov je možné na základe výsledkov hlavnej diagonály poznamenať, že Random Forest mal lepšiu úspešnosť ako samostatný rozhodovací strom.

Záver

Na základe vykonanej analýzy a implementácie bolo potvrdené, že metóda Random Forest dosahuje vyššiu klasifikačnú presnosť ako samostatný rozhodovací strom. Random Forest vďaka využitiu viacerých nezávislých stromov a princípu baggingu vykazuje väčšiu robustnosť voči preučeniu a lepšiu generalizáciu na neznáme dáta.

Práca preukázala schopnosť implementácie základných algoritmov strojového učenia „od nuly“ a ich uplatnenie v praktickej úlohe predikcie cenovej kategórie leteniek. Výsledky poukazujú na potenciál týchto algoritmov pre riešenie klasifikačných úloh v rôznych oblastiach, kde je potrebné robiť automatizované rozhodnutia na základe historických dát.

Do budúcna je možné v práci použiť iné algoritmy rozhodovavích stromov, prípadne aplikovať modely na iné typy datasetov s cieľom overenia ich univerzálnosti a výkonnosti.

Zoznam použitej literatúry

1. POLAKOVIČ, M. a KOLLÁR, J. *Diskrétna matematika*. Bratislava: Penguin, 2019. Dostupné tiež z: https://www.math.sk/jmkollar/.
2. HASTIE, Trevor, TIBSHIRANI, Robert a FRIEDMAN, Jerome. *The Elements of Statistical Learning*. Springer. Dostupné tiež z: https://www.sas.upenn.edu/~fdiebold/NoHesitations/BookAdvanced.pdf.
3. MITCHELL, T.M. *Machine Learning*. New York: McGraw-Hill Science/Engineering/Math, 1997. ISBN 0070428077. Dostupné tiež z: https://www.cs.cmu.edu/~tom/files/MachineLearningTomMitchell.pdf.
4. HALVORSEN, H. P. *Python Programming*. Dostupné tiež z: https://www.halvorsen.blog/documents/programming/python/resources/Python%20Programming.pdf.
5. *What is a Decision Tree?* IBM. Dostupné tiež z: https://www.ibm.com/think/topics/decision-trees.
6. MUREL, J. a KAVLAKOGLU, E. *What is a confusion matrix?* IBM. Dostupné tiež z: https://www.ibm.com/think/topics/confusion-matrix.