VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

PREDIKTIVNÍ MODELOVÁNÍ NAD DATY O VYHLEDÁVÁNÍ LETENEK

PREDICTIVE MODELLING ON FLIGHT SEARCH DATA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **VILIAM PODHAJECKÝ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. IVANA BURGETOVÁ, Ph.D.**
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

156781

Ústav: Ústav informačních systémů (UIFS)
Student: **Podhajecký Viliam**
Program: Informační technologie
Název: **Prediktivní modelování nad daty o vyhledávání letenek**
Kategorie: Data mining
Akademický rok: 2023/24

Zadání:

1. Prostudujte problematiku analýzy dat a prediktivního modelování.
2. Seznamte se s dostupnými daty o vyhledávání letenek a navrhněte analytické úlohy, které by mohly přinést zajímavé informace.
3. Po dohodě s vedoucí vyberte analytickou úlohu/úlohy, kterou se budete dále zabývat. Pro tuto úlohu navrhněte vhodné rysy a natrénujte s nimi alespoň dva vybrané predikční modely.
4. Navrhněte vhodnou webovou aplikaci, která umožní uživateli zadat aktuální data a zobrazí informace o provedených predikcích.
5. Implementujte navrženou aplikaci.
6. Zhodnoťte dosažené výsledky.

Literatura:

- Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Third Edition. Morgan Kaufmann Publishers, 2012, 703 p., ISBN 978-0-12-381479-1
- Nielsen, F.A.: Data Mining with Python. 2015. 101 p. [cit. 2023-10-26]. Dostupné na http://www.freetechbooks.com/data-mining-with-python-working-draft-t1159.html.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 4.

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

Vedoucí práce: **Burgetová Ivana, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1.11.2023
Termín pro odevzdání: 9.5.2024
Datum schválení: 30.10.2023

Abstrakt

Táto bakalárska práca sa zaoberá vývojom webovej aplikácie zameranej na prediktívne modelovanie dát o vyhľadávaní leteniek. Hlavným cieľom je poskytnúť užívateľom nástroje pre informovanejšie rozhodovanie pri kúpe leteniek. Práca kombinuje metódy dolovania dát a prediktívneho modelovania s pokročilým webovým vývojom.

Abstract

This bachelor's thesis focuses on the development of a web application aimed at predictive modeling of flight search data. The main goal is to provide users with tools for more informed decision-making when purchasing airline tickets. The work combines data mining methods and predictive modeling with advanced web development.

Kľúčové slová

prediktívne modelovanie, dolovanie dát, letecká doprava, ceny leteniek, neurónové siete, strojové učenie

Keywords

predictive modeling, data mining, air travel, flight prices, neural networks, machine learning

Citácia

PODHAJECKÝ, Viliam. *Prediktivní modelování nad daty o vyhledávání letenek*. Brno, 2024. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Ivana Burgetová, Ph.D.

Prediktivní modelování nad daty o vyhledávání letenek

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Ivany Burgetovej, Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Viliam Podhajecký
9. 5. 2024

Poďakovanie

Na tomto mieste by som chcel poďakovať pani Ing. Ivaně Burgetovej, Ph.D., za jej vedenie, cenné rady a odbornú pomoc, ktoré mi poskytovala počas celej prípravy bakalárskej prace. Ďalej by som rád poďakoval spoločnosti Kiwi.com za poskytnutie dát, ktoré umožnili realizáciu tejto práce.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Kapitola 1

Úvod

V súčasnom digitálnom veku sa objem generovaných dát neustále zvyšuje, čo predstavuje výzvy aj príležitosti pre podniky a organizácie naprieč rôznymi odvetviami. Získavanie znalostí z dát, často označované ako dolovanie dát, sa stáva neoddeliteľnou súčasťou pri rozhodovaní, pretože poskytuje hlboké pohľady do komplexných vzorcov a trendov. Využitie týchto znalostí môže viesť k významným zlepšeniam v prediktívnej analýze, personalizácii služieb, zlepšení zákazníckej skúsenosti a optimalizácii procesov. Tieto možnosti zvyšujú konkurenčnú výhodu organizácií, ktoré sú schopné efektívne analyzovať a aplikovať získané dáta.

Táto bakalárska práca sa zaoberá vývojom webovej aplikácie, ktorá slúži ako prediktívny nástroj pre odhad vývoju cien leteniek. Práca využíva najnovšie technológie v oblasti webového vývoja a dátového inžinierstva, ktoré umožňujú nielen spracovanie veľkých objemov dát, ale aj ich rýchlu a efektívnu analýzu.

Jadro aplikácie je postavené na modernom webovom frameworku FastAPI, ktorý je známy svojou vysokou výkonnosťou a podporou asynchrónneho programovania. FastAPI umožňuje rýchly vývoj robustných API s automatickou validáciou dát a dokumentáciou, čo zvyšuje rýchlosť vývoja a zlepšuje spoluprácu v tímoch. Na strane servera, Uvicorn poskytuje ľahký a rýchly ASGI server, ktorý zabezpečuje, že aplikácia môže efektívne spracovať súbežné dopyty bez výrazného zaťaženia systémových zdrojov.

Frontend aplikácie je vyvinutý pomocou knižnice React, ktorá umožňuje vývoj dynamických a interaktívnych užívateľských rozhraní. React je integrovaný s Material-UI, ktorý poskytuje súbor komponentov založených na princípoch Material Design. Toto umožňuje rýchlu a konzistentnú implementáciu atraktívneho a intuitívneho dizajnu. Kľúčové prvky ako grafy a vizualizácie sú realizované s využitím Chart.js, čo používateľom poskytuje hmatateľné grafické reprezentácie dátových analýz a predikcií.

Samotná predikcia cien leteniek v aplikácii je implementovaná pomocou metód strojového učenia, zahŕňajúcich výber a trénovanie modelov na základe rozsiahlych historických dát. Tieto modely sú vyvíjané tak, aby zohľadňovali sezónne trendy a ďalšie relevantné faktory, ktoré môžu ovplyvniť ceny leteniek. Aplikácia dokáže predpovedať budúci vývoj cien s presnosťou, ktorá umožňuje užívateľom robiť informovanejšie rozhodnutia pri kúpe leteniek.

Celková architektúra aplikácie je založená na princípe kontajnerizácie s využitím Dockeru, ktorý zabezpečuje jednoduchosť nasadenia a prenosnosť prostredia. Toto je obzvlášť cenné pri zabezpečení konzistentnosti medzi vývojovými, testovacími a produkčnými prostrediami.

Táto bakalárska práca nie len demonštruje implementáciu moderných technológií pre získavanie znalostí z dát, ale tiež poskytuje praktické aplikácie týchto technológií v reálnom svete. Cieľom je poskytnúť užitočný nástroj, ktorý by mohol slúžiť analytikom a spotrebiteľom na získavanie predikcií a zlepšenie rozhodovacích procesov. Využitím pokročilých technológií a metodológií v oblasti dátovej analýzy a softvérového inžinierstva táto práca prispieva k lepšiemu pochopeniu možností digitálnej transformácie a automatizácie v rôznych priemyselných odvetviach.

Práca sa začína teoretickou časťou opisujúcou proces získavania znalostí, kde sú uvedené najznámejšie metódy dolovania dát a ich praktické využitie. Tretia kapitola je venovaná neurónovým sieťam, ich štruktúre a rôznym typom používaným v analýze dát. Vo štvrtej kapitole je podrobne opísaný návrh a architektúra webovej aplikácie, vrátane technológií použitých na frontend a backend. Piatá kapitola sa zameriava na implementáciu prediktívneho modelu, jeho trénovanie a testovanie. Šiesta kapitola podrobne rozoberá implementáciu webovej aplikácie, vrátane detailov vývoja, použitých frameworkov a nástrojov. Sedmá kapitola sa sústredí na experimentálne výsledky a vyhodnocovanie efektívnosti systému.

Kapitola 2

Získavanie znalostí z dát

Získavanie znalosti z dát alebo dolovanie dát (angl. Data mining) je proces objavovania vzorov vo veľkých súboroch dát, ktoré zahŕňa metódy strojového učenia, štatistiky a databázových systémov. Ide o multidisciplinárnu oblasť, ktorá využíva umelú inteligenciu, strojové učenie, štatistiku a databázové systémy na získavanie vedomostí a poznatkov zo štruktúrovaných a neštruktúrovaných dát [9].

Techniky dolovania dát možno použiť na identifikáciu trendov, vzorov a vzťahov v dátach a na extrahovanie užitočných informácií, ktoré možno použiť na zlepšenie rozhodnutí, optimalizáciu procesov, a zefektívnenie prevádzky. Niektoré bežné aplikácie dolovania dát zahŕňajú detekciu podvodov, analýzu trhu a správanie sa užívateľa na internete (napr. cielené reklamy).

Získavanie znalosti z dát zahŕňa použitie algoritmov a štatistických modelov na analýzu a extrahovanie užitočných informácií z dát. Proces zvyčajne zahŕňa nasledujúce kroky [10]:

1. **Zber a príprava dát:** Dáta sa zvyčajne zhromažďujú z viacerých zdrojov, ako sú databázy, textové súbory, webové logy a podobne. Dáta sa potom vyčistia a pripravia na analýzu.

2. **Skúmanie a vizualizácia dát:** Dáta sa skúmajú a vizualizujú, aby sa lepšie porozumeli vzorom a vzťahom v dátach.

3. **Modelovanie a testovanie:** Štatistické modely a algoritmy strojového učenia sa používajú na hľadanie vzorov a vzťahov v dátach. Tieto modely sa potom testujú, aby sa zistilo, ako dobre zodpovedajú a ako presne dokážu predpovedať budúce hodnoty.

4. **Vyhodnotenie výsledkov:** Výsledky procesu dolovania dát sa vyhodnocujú, aby sa určila ich užitočnosť a identifikovali sa potenciálne problémy alebo obmedzenia.

Získavanie znalostí z dát môže byť účinným nástrojom na odhaľovanie poznatkov a trendov, ale pri interpretácii výsledkov je dôležité postupovať opatrne, pretože nemusia byť vždy presné alebo reprezentatívne pre väčšiu populáciu.

2.1 Zber a príprava dát

Zber a príprava dát na analýzu je krokom v procese získavania znalostí z dát. Zahŕňa to zhromažďovanie dát z rôznych zdrojov ako sú databázy, textové súbory a webové denníky.

Po zhromaždení dát nasleduje čistenie a organizovanie dát tak, aby ich bolo možné efektívne analyzovať.

Zhromažďovanie a príprava dát na analýzu zahŕňa niekoľko krokov [2] a [17]:

1. **Identifikácia zdrojov dát:** Prvým krokom je identifikácia zdrojov dát, ktoré sa použijú na analýzu. To môže zahŕňať databázy, textové súbory, webové logy alebo iné zdroje dát.
2. **Extrahovanie dát:** Dáta sú potom extrahované zo zdrojov a usporiadané do použiteľného formátu, ako je tabuľka alebo databáza.
3. **Čistenie a predbežné spracovanie dát:** Dáta sa potom vyčistia, aby sa odstránili všetky chyby alebo nezrovnalosti, a odstránia sa všetky nepotrebné alebo nepodstatné dáta.
4. **Transformácia dát:** Možno bude potrebné dáta transformovať alebo agregovať, aby boli vhodnejšie na analýzu. Môže to zahŕňať vytváranie nových premenných, kombinovanie premenných alebo sumarizáciu dát.
5. **Rozdelenie dát:** Dáta sa zvyčajne delia na trénovaciu a testovaciu množinu. Tréningová množina sa používa na zostavenie modelov a testovacia množina sa používa na vyhodnotenie presnosti modelov.

Zhromažďovanie a príprava dát na analýzu je dôležitým krokom v procese získavania znalostí z dát, pretože kvalita a relevantnosť dát ovplyvňuje presnosť a spoľahlivosť výsledkov.

2.1.1 Identifikácia zdrojov dát

Identifikácia zdrojov dát pri získavaní znalosti z dát sa týka procesu určovania, odkiaľ budú pochádzať dáta pre konkrétnu analýzu. Je to dôležitý krok v procese, pretože kvalita a relevantnosť dát výrazne ovplyvní výsledky analýzy.

Zdroje dát možno klasifikovať podľa ich štruktúry a pôvodu. Toto rozdelenie pomáha zlepšiť pochopenie a výber vhodných dátových zdrojov pre analýzu:

Podľa štruktúry dát

- **Štruktúrované dáta:** Dáta uložené vo formátovanom spôsobe, ako sú tabuľky alebo databázy.
- **Neštruktúrované dáta:** Dáta bez pevného formátu, ako sú textové dokumenty alebo e-maily.

Podľa pôvodu dát

- **Interné dáta:** Dáta zbierané a uchovávané v rámci organizácie, ako sú napríklad záznamy o vyhľadávaní leteniek alebo zakúpený tovar.
- **Externé dáta:** Dáta zbierané zo zdrojov mimo organizácie, ako sú verejné databázy a pod.

Pri identifikácií zdrojov dát je dôležité zvážiť kvalitu a relevantnosť dát, ako aj právne alebo etické úvahy súvisiace so zberom a používaním dát.

2.1.2 Extrahovanie dát

Extrahovanie dát je proces, kde sa z veľkého množstva dátových zdrojov vyberajú a získavajú len tie údaje, ktoré sú nevyhnutné pre špecifickú analýzu. Tento krok zahŕňa identifikáciu a izoláciu relevantných dátových bodov z rôznych zdrojov, ako sú databázy, textové súbory alebo online zdroje.

V praxi zahŕňa extrahovanie dát viacero technických krokov: vyhľadávanie a selekcia dát podľa špecifických kritérií, ich export z jedného prostredia do druhého, a často aj prvé základné usporiadanie, ako je usporadúvanie dát v chronologickom poradí alebo podľa kategórií. Výzvou v tomto procese je zabezpečiť, aby boli dáta extrahované presne a kompletné, bez straty informácií.

Pri extrahovaní dát je tiež dôležité dbať na udržanie ich integrity a bezpečnosti. Prístup k dátam a ich spracovanie musí byť v súlade s príslušnými právnymi a etickými normami, najmä pokiaľ ide o osobné údaje. Z tohto dôvodu sa často používajú nástroje a protokoly, ktoré zabezpečujú, že dáta sú spracovávané transparentne a v súlade s predpismi o ochrane údajov.

Výsledkom efektívneho procesu extrakcie dát je dobre usporiadaná a relevantná dátová sada, ktorá je pripravená na ďalšie analytické fázy, ako je čistenie dát a ich hlbšia analýza. Správne vykonané extrahovanie značne uľahčuje a zefektívňuje celý proces získavania poznatkov z dát.

2.1.3 Čistenie a predbežné spracovanie dát

Čistenie a predbežné spracovanie zabezpečujú, že dáta sú presné a vhodne na spracovanie. Proces čistenia dát zahŕňa riešenie rôznych bežných problémov, ktoré môžu vzniknúť z rôznych zdrojov alebo počas akvizície dát. Bežné problémy pri čistení dát zahŕňajú [8]:

- **Chýbajúce hodnoty:** Jedným z najčastejších problémov v dátových súboroch sú chýbajúce dáta. Tieto hodnoty môžu byť riešené rôznymi technikami, ako je ich nahradenie priemernou hodnotou, mediánom, modusom alebo použitím pokročilejších metód, ako sú prediktívne modelovanie alebo interpolácia, ktoré odhadujú chýbajúce hodnoty na základe ostatných dát.

- **Duplicitné záznamy:** Duplicitné dáta môžu vznikať z viacerých dôvodov, vrátane chýb pri zbere dát alebo pri integrácii dát z rôznych zdrojov. Odstránenie duplikátov je nevyhnutné, aby sa predišlo skresleniu v analytických výsledkoch.

- **Nekonzistentné dáta:** Dáta pochádzajúce z rôznych zdrojov môžu obsahovať rozdiely vo formátoch alebo jednotkách merania, čo vyžaduje štandardizáciu. Napríklad dátumy a časy môžu byť zaznamenané v rôznych formátoch, ktoré je potrebné konvertovať do jednotného štandardu.

- **Chybné údaje a vychýlené hodnoty:** Chyby vo vstupných dátach alebo odľahlé hodnoty môžu viesť k nesprávnym analýzam a záverom. Identifikácia a oprava takýchto hodnôt, buď ich opravou alebo odstránením, je kľúčová.

- **Neplatné dáta:** Neplatné dáta môžu zahŕňať logicky nemožné alebo nezmyselné hodnoty, ako napríklad záporné časy trvania alebo vekové kategórie mimo možného rozsahu. Identifikácia a oprava takýchto dát je nevyhnutná pre spoľahlivosť analýzy.

Tieto kroky pomáhajú zabezpečiť, že dáta budú správne interpretované analytickými nástrojmi a že analytické modely budú schopné efektívne pracovať s dátami bez technických prekážok.

Efektívne vykonané čistenie a predbežné spracovanie zvyšujú pravdepodobnosť získania presných a spoľahlivých výsledkov z analytických modelov, čo je základom úspešného získavania poznatkov z dát.

2.1.4 Transformácia dát

Transformácia dát pri dolovaní dat je proces, ktorý zahŕňa zmeny na úrovni dátových štruktúr, formátov a hodnôt, aby boli dáta vhodnejšie na spracovanie a analýzu. Tento krok je nevyhnutný pre úpravu dát do formy, ktorá najlepšie podporuje analytické nástroje a techniky, a zároveň umožňuje efektívnejšie a presnejšie získavanie poznatkov. Kľúčové aspekty tranformácie dát zahŕňajú [8]:

- **Normalizácia a škálovanie:** Tieto techniky sa používajú na zmenšenie rozdielov v rozsahu hodnôt medzi rôznymi premennými. Normalizácia obvykle zahŕňa úpravu dát tak, aby ich rozsah bol medzi 0 a 1, alebo tak, aby mali nulový priemer a štandardnú odchýlku 1. Škálovanie pomáha pri zabezpečení, že žiadna premenná nebude dominovať ostatným pri výpočtoch, čo je kritické pre algoritmy strojového učenia, ktoré sú citlivé na rozsah dát.

- **Kodifikácia premenných:** Mnohé analytické modely vyžadujú, aby všetky vstupné premenné boli číselné. Kodifikácia kategorických dát (napríklad pohlavie alebo národnosť) do číselných formátov, ako je one-hot encoding alebo label encoding, je nevyhnutná. Tieto techniky transformujú kategorické atribúty na binárne alebo numerické reprezentácie, ktoré môžu byť efektívne spracované algoritmami.

- **Vytváranie a odvodzovanie premenných (Feature Engineering):** V tejto fáze sa nové premenné generujú z existujúcich dát. Napríklad z dátumu môže byť odvodený deň v týždni alebo ročné obdobie, čo môže mať prediktívnu hodnotu pre určité typy analýz. Feature engineering je zásadný pre zlepšenie výkonnosti modelov tým, že sa zvyšuje množstvo a kvalita informácií, ktoré sú modelom k dispozícii.

- **Redukcia rozmerov:** Táto technika sa používa na zníženie počtu premenných v datasete, často bez významnej straty informácií. Techniky ako analýza hlavných komponentov (PCA) alebo lineárne diskriminačné analýzy (LDA) sú bežné metódy na zníženie dimenzií dát, čo pomáha zjednodušiť modely a zrýchliť výpočty.

- **Diskretizácia:** Niektoré modely vyžadujú, aby boli spojité premenné spracované ako diskrétne kategórie. Diskretizácia transformuje spojité premenné na kategorické, čo môže pomôcť zjednodušiť analýzu a zlepšiť interpretáciu modelov.

Transformácia dát je nevyhnutná nielen pre prispôsobenie dát potrebám špecifických analytických nástrojov, ale aj pre maximalizáciu presnosti a efektivity získavania poznatkov. Správne transformované dáta zvyšujú relevanciu a spoľahlivosť analytických modelov, čo je rozhodujúce pre úspešné využitie dát v rozhodnutiach.

2.1.5 Rozdelenie dát

Rozdelenie dát v procese dolovania dát zabezpečuje, že modely strojového učenia sú trénované a validované na adekvátne oddelených množinách údajov. Tento proces umožňuje

objektívne hodnotenie výkonnosti modelu pred jeho nasadením v reálnych podmienkach. Rozdelenie dát sa zvyčajne vykonáva podľa špecifických pravidiel alebo algoritmov, aby sa zabezpečila spravodlivá a reprezentatívna distribúcia údajov. Dôležité aspekty rozdelenia dát zahŕňajú:

- **Trénovacie a testovacie sady:** Najbežnejším prístupom je rozdelenie dát na trénovaciu a testovaciu sadu. Trénovacia sada sa používa na budovanie a učenie modelu, zatiaľ čo testovacia sada slúži na overenie modelu v podmienkach, ktoré simulujú jeho reálne použitie. Typicky sa odporúča, aby trénovacia sada predstavovala väčší podiel dát (napríklad 70-80%).
- **Validačná sada:** Pre zložitejšie modely alebo pri ladení hyper-parametrov je vhodné použiť trojité rozdelenie, kde dáta sú rozdelené na trénovaciu, validačnú a testovaciu sadu. Validačná sada umožňuje ladenie modelu bez rizika "pretrénovania"na testovacej sade.
- **Krížová validácia:** Tento prístup zahŕňa rozdelenie dát na K podmnožín (alebo "skladov") a postupne používa každú skládku ako testovaciu sadu s ostatnými skladbami použitými ako trénovacie dáta. Táto metóda zvyšuje spoľahlivosť hodnotenia modelu tým, že zabezpečuje, že každý údaj je použitý v trénovacej aj testovacej sade.
- **Časovo závislé rozdelenie:** V prípadoch, kedy dáta obsahujú časové značky (napr. finančné dáta), je dôležité rozdeliť dáta tak, aby testovacie dáta chronologicky nasledovali po trénovacích dátach. Toto zabraňuje "presaku informácií"do modelu, ktorý by mohol skresliť jeho výkonnosť v reálnom svete.
- **Rozdelenie podľa skupín:** V situáciách, kedy sú dáta zoskupené (napr. užívatelia služby), je potrebné zabezpečiť, aby celé skupiny boli priradené buď k trénovacej alebo testovacej sade. Toto zabraňuje modelom v učení sa špecifických vzorcov, ktoré sú unikátne pre konkrétne skupiny, čo by mohlo viesť k nadmernému učeniu.

2.2 Skúmanie a vizualizácia dát

Skúmanie a vizualizácia dát sa robí s cieľom lepšie pochopiť vzory a vzťahy v dátach. Tento krok pomáha identifikovať trendy, vzory a anomálie v dátach a môže tiež pomôcť identifikovať potenciálne problémy alebo obmedzenia v dátach.

Existuje niekoľko techník, ktoré možno použiť na preskúmanie a vizualizáciu dát:

- **Deskriptívna štatistika:** Deskriptívna štatistika poskytuje súhrnné štatistiky, ako je priemer, medián a štandardná odchýlka, ktoré pomáhajú opísať celkové charakteristiky dát.
- **Vizualizácia dát:** Vizualizácia dát zahŕňa vytváranie grafov, tabuliek a grafov, ktoré pomáhajú vizualizovať dáta a uľahčujú ich pochopenie.
- **Prieskum dát:** Prieskum dát zahŕňa použitie rôznych techník na hĺbkové analyzovanie a pochopenie jednotlivých atribútov a ich vzájomných vzťahov.

Skúmanie a vizualizácia dát pomáha identifikovať trendy a vzory v dát a môže tiež pomôcť identifikovať akékoľvek potenciálne problémy alebo obmedzenia v dátach. Je dôležité preskúmať a vizualizovať dáta, aby ste sme sa uistili, že výsledky procesu dolovania dát sú presné a zmysluplné.

2.3 Modelovanie

Modelovanie zahŕňa výber vhodného algoritmu na analýzu dát a jeho nastavenie na optimalizáciu výkonu. Zahrnuje to výber medzi regresnými modelmi, klasifikačnými modelmi alebo zhlukovacími algoritmami v závislosti od typu a povahy údajov a špecifického cieľa analýzy. Napríklad:

- **Regresné modely:** Používajú sa na predpovedanie nepretržitých výstupných hodnôt založených na jednej alebo viacerých nezávislých premenných. Sú ideálne pre úlohy, kde je cieľom odhadnúť kvantitatívnu hodnotu, ako sú cenové predikcie alebo predpovede spotreby energie.
- **Klasifikačné modely:** Tieto modely predpovedajú kategorické výstupné štítky a sú často používané na rozhodovanie alebo rozpoznávanie vzorcov, ako je identifikácia spamu v e-mailoch alebo diagnostika chorôb.
- **Zhlukovacie modely:** Zhlukovanie je forma nenávodného učenia, ktorá sa používa na skupinové rozdeľovanie dát bez preddefinovaných štítkov. Modely identifikujú prirodzené skupiny alebo klastre v dátach, čo je užitočné pre segmentáciu trhu alebo identifikáciu nových vzorcov správania.

2.3.1 Regresné modely

Regresné modely sú základnou metódou používanou v štatistike a strojovom učení, kde sa uplatňujú na modelovanie vzťahov a predpovedanie hodnôt. Tieto modely skúmajú ako závislé premenné reagujú na zmeny v jednej alebo viacerých nezávislých premenných.

Existujú rôzne formy regresných modelov, vrátane lineárnej regresie, ktorá je vhodná pre údaje s lineárnou vzájomnou závislosťou medzi premennými. Pre komplexnejšie vzťahy, kde závislosť nie je lineárna, sa využíva polynomiálna regresia. Logistická regresia, napriek svojmu názvu, je často využívaná pre klasifikačné úlohy, kde výstupné hodnoty sú kategorické, a nie kontinuálne.

Na overenie účinnosti modelu sa často používajú techniky ako krížová validácia, ktorá testuje model na rôznych podmnožinách dát, aby sa zabezpečilo, že výsledky sú konzistentné a že model je schopný generalizovať na nevidené dáta.

Lineárna regresia

Lineárna regresia je základný regresný model, ktorý sa používa na modelovanie lineárneho vzťahu medzi jednou alebo viacerými nezávislými premennými a závislou premennou. Tento model hľadá lineárnu funkciu, ktorá najlepšie vyjadruje tento vzťah [7]:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon \tag{2.1}$$

kde y je závislá premenná, $\beta_0, \beta_1, \ldots, \beta_n$ sú koeficienty modelu, $x_1, x_2, \ldots, x_n$ sú nezávislé premenné, a ϵ je chybový člen predstavujúci náhodné odchýlky.

Predpoklady lineárnej regresie patrí normalita rezíduí, ktoré by mali byť rozdelené normálne okolo nuly, konzistentnosť rozptylu rezíduí pre rôzne hodnoty prediktívnych premenných, nezávislosť rezíduí, čo znamená, že chyby modelu by mali byť nezávislé, a absencia multikolinearity, čo znamená, že vysvetľujúce premenné by nemali byť príliš silne korelované medzi sebou.

Polynomiálna regresia

Polynomiálna regresia je rozšírením lineárnej regresie, ktoré umožňuje modelovanie vzťahov, kde závislosť medzi premennými nie je lineárna. Tento model zahrňuje polynomiálne členy, ako sú kvadratické alebo kubické členy, umožňujúc lepšie zachytiť zakrivené vzťahy medzi premennými [11]:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_n x^n + \epsilon \tag{2.2}$$

kde stupne x (napr. $x, x^2, \ldots, x^n$) umožňujú modelu zachytiť nelineárne vzťahy medzi premennými.

Aj keď predpoklady sú podobné ako pri lineárnej regresii, pri použití polynomiálnej regresie je dôležité byť obozretným ohľadom overfittingu, keďže vyššie stupne polynómu môžu viesť k prílišnému prispôsobeniu na trénovacie dáta.

Logistická regresia

Logistická regresia je špeciálne zameraná na modelovanie pravdepodobnosti výskytu určitej kategórie alebo udalosti, často sa používa v prípadoch, kde je výstupná premenná binárna. Model je založený na logitovej funkcií, ktorá prevádza lineárnu kombináciu vstupných premenných na pravdepodobnosť. Logistická regresia umožňuje modelovanie pravdepodobností medzi 0 a 1 a závisí na predpoklade, že rozdiel logaritmov pravdepodobností daného výsledku a jeho komplementu je lineárny vo vzťahu k prediktívnym premenným [12].

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n \tag{2.3}$$

kde p je pravdepodobnosť príslušnosti k určitej kategórii. Pravdepodobnosť p sa potom získava transformáciou logitov späť na pravdepodobnosť pomocou logistickej funkcie:

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n)}} \tag{2.4}$$

2.3.2 Klasifikačné modely

Klasifikačné modely sa používajú na priraďovanie jednotlivých príkladov do preddefinovaných kategórií. Tieto modely identifikujú a učia sa vzorce z trénovacích dát, aby mohli správne kategorizovať nové dáta.

V praxi sa klasifikačné modely využívajú v mnohých aplikáciách. V medicíne pomáhajú diagnostikovať choroby, v bankovníctve môžu predpovedať úverovú spoľahlivosť klientov, a v oblasti maloobchodu môžu predpovedať správanie zákazníkov.

Existuje niekoľko populárnych typov klasifikačných modelov:

- **Logistická regresia** aj keď sme ju spomenuli v kontexte regresných modelov 2.3.1, logistická regresia sa často používa aj pre klasifikačné úlohy, najmä pre binárne klasifikácie. Modeluje pravdepodobnosť príslušnosti do jednej z tried na základe lineárnej kombinácie vstupných premenných.
- **Rozhodovacie stromy** klasifikujú dáta tým, že ich postupne delia podľa rozhodovacích pravidiel, až kým každý prvok nespadne do určitej kategórie.
- **Support Vector Machines (SVM)** efektívne rozdeľujú dátové body do kategórií tým, že nachádzajú optimálnu rozdelovaciu hyperrovinu.

- **Náhodné lesy** je rozšírením rozhodovacích stromov, pričom sa trénuje viacero stromov na rôznych podmnožinách dát a často aj na podmnožinách atribútov. Ich predpovede sa potom kombinujú. Tento prístup zvyšuje presnosť a robustnosť modelu oproti jednoduchým rozhodovacím stromom.
- **Neurónové siete** sú flexibilné modely, ktoré sa skladajú z vrstiev neurónov a môžu modelovať zložité vzťahy v dátach. Sú široko používané pre hlboké učenie a vynikajú v úlohách, kde je veľa vstupných premenných a vzťahy medzi nimi sú komplexné.
- **K-najbližších susedov (K-NN)** klasifikuje dáta na základe najbližších susedov každého bodu, priraďujúc nové prvky do najčastejšie zastúpenej triedy medzi ich K susedmi. Táto metóda je efektívna pri modelovaní komplexných alebo nejasných vzťahov v dátach.

Aj keď sú mnohé klasifikačné modely primárne určené na kategorizovanie, je dôležité pripomenúť, že niektoré z nich môžu byť efektívne využité aj na predpovedanie spojitých hodnôt. Táto schopnosť je obzvlášť užitočná v prípadoch, kde tradičné regresné modely nemusia byť dostatočne presné alebo flexibilné.

Rozhodovacie stromy

Rozhodovacie stromy sú modely, ktoré vytvárajú stromovú štruktúru s rozhodovacími pravidlami založenými na atribútoch dát. Každý uzol v strome predstavuje test na vlastnosti (atribúty) a každá vetva predstavuje výsledok tohto testu, vedúci k listom, ktoré predstavujú klasifikačné rozhodnutia alebo výsledky. Tento model je obzvlášť efektívny v situáciách, kde je možné dáta jasne segmentovať na základe logických pravidiel. Je vysoko interpretovateľný, čo umožňuje ľahké pochopenie rozhodnutí modelu [5].

Náhodné lesy

Náhodné lesy rozširujú koncept rozhodovacích stromov tým, že kombinujú predpovede z mnohých stromových modelov na základe rôznych podmnožín tréningových dát. Tento prístup znižuje riziko pretrénovania, ktoré je bežné pri jednoduchých rozhodovacích stromoch, a zvyšuje presnosť modelu prostredníctvom agregácie výsledkov (bagging). Náhodné lesy sú vhodné pre aplikácie, kde je potrebná vysoká presnosť a robustnosť predpovedí [5].

Support Vector Machines (SVM)

Support Vector Machines (SVM) slúži na klasifikáciu, kde sa hľadá najlepšia hyperrovina, ktorá efektívne oddeľuje rôzne triedy v dátovom priestore. SVM sú zvlášť užitočné pri klasifikácii dát s vysokou dimenzionalitou, kde iné metódy môžu zlyhať. Tento model optimalizuje margin, čiže vzdialenosť medzi najbližšími bodmi rôznych tried, čo zvyšuje generalizačnú schopnosť modelu [4].

Neurónové siete

Neurónové siete sú zložité modely inšpirované fungovaním ľudského mozgu, ktoré sa skladajú z vrstiev neurónov s váhovanými spojeniami. Sú vynikajúce v modelovaní komplexných vzorov v dátach vďaka ich schopnosti učiť sa a modelovať nelineárne vzťahy. Neurónové siete sú obzvlášť účinné v oblastiach ako obrazová a zvuková analýza, rozpoznávanie reči, a ďalšie

aplikácie, kde je potrebné spracovávať veľké množstvá dát s vysokým stupňom abstrakcie. (viac v samostatnej kapitole 3)

K-najbližších susedov (K-NN)

K-najbližších susedov (K-NN) pracuje na princípe najbližšej blízkosti medzi dátovými bodmi. Model určuje kategórie nových príkladov podľa hlasovania ich K najbližších susedov, pričom na meranie blízkosti často využíva euklidovskú vzdialenosť. K-NN je neparametrický a lenivý algoritmus, čo znamená, že nevytvára explicitný model a vykonáva výpočty v reálnom čase počas klasifikácie [15].

2.3.3 Zhlukovacie modely

Zhlukovacie modely alebo klastrovacie modely sú modely v dolovaní dát, ktoré sa používajú k rozdeleniu dát do skupín (zhlukov) podľa ich podobnosti. Na rozdiel od klasifikačných modelov, ktoré priraďujú príklady do preddefinovaných kategórií, zhlukovanie identifikuje skryté vzorce v dátach, ktoré umožňujú dátové body automaticky zoskupiť do zhlukov alebo skupín [18].

V praxi sa zhlukovanie využíva v rôznych oblastiach, vrátane marketingu pre segmentáciu zákazníkov, v bioinformatike pre analýzu genetických dát, alebo v sociálnych sieťach pre identifikáciu komunít. Tieto modely sú dôležité aj pre zníženie dimenzionality dát a objavovanie základných štruktúr.

2.4 Vyhodnotnocovanie výsledkov

Vyhodnotenie výsledkov pri získavaní znalosti z dát sa vzťahuje na proces hodnotenia výkonnosti modelu alebo analýzy dolovania dát s cieľom pochopiť poznatky a informácie získané z dát. To môže zahŕňať použitie rôznych metrík a metód na meranie správnosti, presnosti, zapamätania a iných aspektov výkonnosti modelu. Existuje niekoľko spôsobov, ako sa dá vyhodnocovať výsledky:

- **Použitie metrík:** Je niekoľko metrík, ktoré je možné použiť k vyhodnocovaniu výsledkov z dolovania dát. Napr. F1 skóre, AUC-ROC, Matica zámien, RMSE, MSE a ďalšie.
- **Porovnanie s inými modelmi:** Porovnaním viacerých modelov je možné zistiť, ktorý model je vhodnejší na riešenie danej úlohy.
- **Vizualizácia:** Grafy a iné vizualizačné metódy vedia pomôcť pochopiť výsledky modelu a poprípade najsť vzorec alebo trend v dátach.
- **Porovnanie s očakávaniami:** Pokiaľ existujú očakávania ohľadom výsledkov modelu, tak je ho možné porovnať s týmito očakávaniami a zistiť, či predpoklady boli správne.

Vyhodnocovanie výsledkov je iteratívny proces, ktorý vyžaduje opakované skúmanie a úpravy modelu, pokiaľ sa nedosiahne požadovanej účinnosti alebo presnosti.

2.4.1 MSE

Stredná kvadratická chyba (MSE - Mean Squared Error) je štatistický ukazovateľ, ktorý meria priemernú veľkosť chýb v predikciách modelov, pričom chyby sú merané ako kvadráty rozdielov medzi predpovedanými a skutočnými hodnotami. MSE je široko používaná ako stratová funkcia v rôznych modeloch strojového učenia, najmä v regresných úlohách. Čím je nižšia výsledná hodnota, tým je model presnejši. Výsledok MSE je vo štvorcových jednotkách premennej, ktorú model predikuje. Výpočet MSE:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.5}$$

kde n je počet vzoriek v dátovom súbore, y_i je skutočná hodnota cieľa pre i-tú vzorku a $\hat{y}_i$ je predikovaná hodnota modelom. Suma kvadrátov rozdielov medzi skutočnými a predikovanými hodnotami poskytuje agregátny ukazovateľ chyby modelu [6].

2.4.2 RMSE

RMSE (Root Mean Square Error) je metrika, ktorá sa používa pre meranie presnosti predikčných modelov v dolovaní dát a v strojovom učení. Metrika je velmi podobná MSE 2.4.1. RMSE vyjadruje priemernú chybu medzi skutočnými hodnotami a predikovanými hodnotami. Je vyjadrený v jednotkách premennej, ktorú model predikuje. Čím nižšia je hodnota RMSE, tým presnejší je model. Výpočet RMSE:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.6}$$

kde n je počet vzoriek v dátovom súbore, y_i je skutočná hodnota cieľa pre i-tú vzorku a $\hat{y}_i$ je predikovaná hodnota modelom. Suma kvadrátov rozdielov medzi skutočnými a predikovanými hodnotami je podľa tohto vzorca normalizovaná a následne od nej je vypočítaný koreň [1].

Kapitola 3

Neurónové siete

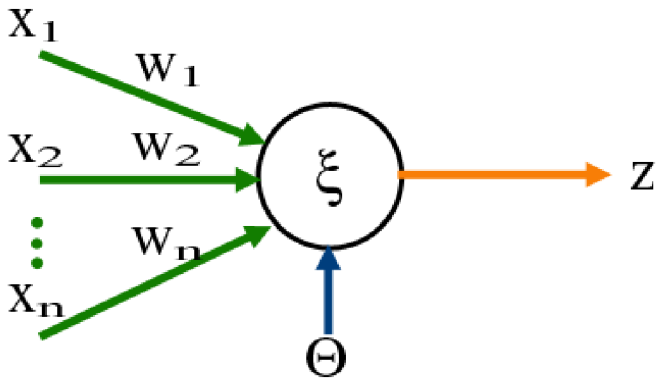
Neurónové siete sú výpočtové modely inšpirované biologickými neurónovými sieťami, ktoré sa skladajú z veľkého počtu prepojených neurónov pracujúcich spoločne na spracovaní informácií. Tento koncept sa začal rozvíjať už v polovici 20. storočia a dnes je základom mnohých moderných prístupov k strojovému učeniu, predovšetkým v oblastiach ako rozpoznávanie obrazu, spracovanie prirodzeného jazyka a autonómne riadenie[16] [19].

Využitie neurónových sietí na predikciu cien leteniek je motivované ich schopnosťou efektívne analyzovať a modelovať komplexné vzory v dátach, ktoré sú charakteristické pre letecký priemysel. Predikcia cien leteniek je zložitá úloha ovplyvnená mnohými faktormi, ako sú sezónne vplyvy, udalosti, zmeny v ponuke letov od konkurenčných leteckých spoločností, ako aj kolísanie cien palív a ekonomických podmienok.

Neurónové siete, zvlášť hlboké učenie, poskytujú robustnú platformu pre analýzu týchto dynamických faktorov tým, že učia z historických dát a neustále aktualizujú svoje predikcie v reálnom čase. Tento prístup umožňuje klientom nie len vidieť aktuálne ceny, ale poskytuje aj prognózy budúcich cenových trendov, čím im umožňuje urobiť informované rozhodnutia o najlepšom čase na kúpu leteniek.

3.1 Štruktúra neurónovej siete

Štruktúra neurónovej siete je navrhnutá tak, aby napodobňovala funkčnosť ľudského mozgu, a zahŕňa súbor prepojených uzlov alebo *neurónov*, ktoré sú organizované do rôznych vrstiev. Každý neurón v neurónovej sieti funguje ako základný spracovateľský prvok, ktorý prijíma vstupy označené ako $X_1, X_2, \ldots, X_n$, kombinuje ich (zvyčajne ich sčíta) s príslušnými váhami $W_1, W_2, \ldots, W_n$ a prahovou hodnotou Θ, aplikuje na ne aktivačnú funkciu ξ a vysiela výstup Z ďalej do siete. Tento proces je znázornený na Obrázku 3.1, ktorý ilustruje typickú štruktúru perceptrónu. Každá vrstva neurónov má špecifickú úlohu a spôsob spracovania informácií, čo umožňuje sieti učiť sa a extrahovať zložité vzorce z dát [4].



Obr. 3.1: Perceptrón (prevzaté z [3])

Aj keď jednoduché perceptróny, ako ten, ktorý je znázornený na obrázku 3.1, predstavujú základné nástroje pre spracovanie vstupov, obvykle sa nepoužívajú samostatne. V súčasnej praxi sa tieto základné jednotky zapájajú do zložitejších konštrukcií nazývaných viacvrstvové neurónové siete. Tieto siete zahrnujú viacero perceptrónov rozvrstvených do rôznych úrovní, čo umožňuje efektívnejšie spracovávanie zložitejších vzorcov a funkcií.

- **Vstupná vrstva:** Táto vrstva prijíma vstupné dáta, ktoré sú spravidla transformované do formátu, ktorý môže neurónová sieť efektívne spracovať.
- **Skryté vrstvy:** Medzi vstupnou a výstupnou vrstvou sa nachádzajú jedna alebo viacero skrytých vrstiev. Tieto vrstvy sú srdcom neurónovej siete, kde prebieha väčšina výpočtového spracovania cez váhovo upravené spojenia.
- **Výstupná vrstva:** Posledná vrstva neurónovej siete, ktorá vydáva výsledok, napríklad klasifikáciu alebo predikciu.

3.1.1 Vstupná vrstva

Vstupná vrstva je prvým bodom, kde neurónová sieť prijíma surové dáta (väčšinou už upravené a normalizované). Táto vrstva obvykle obsahuje toľko neurónov, koľko je vstupných atribútov v dátach. Napríklad pri analýze obrazu by každý neurón mohol reprezentovať jednotlivý pixel obrázka. Úlohou vstupnej vrstvy je prijať tieto vstupy a efektívne ich distribuovať do skrytých vrstiev na ďalšie spracovanie.

3.1.2 Skryté vrstvy

Skryté vrstvy tvoria jadro neurónovej siete, kde dochádza k najdôležitejšiemu spracovaniu dát. V neurónových sieťach sa môže nachádzať jedna alebo viacero skrytých vrstiev, v závislosti od zložitosti úlohy, ktorú sieť rieši. Tieto vrstvy sú zodpovedné za extrahovanie a transformáciu vlastností zadaných na vstupnej vrstve do formy, ktorá je užitočná pre predikcie alebo klasifikácie na výstupnej vrstve.

Každý neurón v skrytých vrstvách prijíma vstup z neurónov predchádzajúcej vrstvy, a tento vstup je vážený – to znamená, že každé spojenie má priradenú váhu, ktorá určuje silu a znamenie signálu, ktorý prechádza týmto spojením. Neuróny tiež používajú aktivačné funkcie, ako sú ReLU (Rectified Linear Unit), sigmoid alebo tanh (hyperbolický tangent),

ktoré rozhodujú o tom, či a ako silný signál bude odoslaný ďalej. Aktivačné funkcie pomáhajú neurónovým sieťam učiť sa nelineárne vzorce a komplexné funkcie.

Funkcie skrytých vrstiev môžu zahŕňať:

- **Identifikácia vzorcov**: Neuróny v týchto vrstvách môžu identifikovať užitočné vzorce alebo funkcie, ktoré prispievajú k celkovému cieľu úlohy, ako je rozpoznávanie objektov vo vizuálnych dátach alebo rozpoznávanie štruktúr v jazykových dátach.
- **Zníženie dimenzionality**: Skryté vrstvy môžu pomôcť zredukovať množstvo dát na spracovateľnejší počet dimenzií, ktoré zachovávajú dôležité informácie potrebné na vykonanie úloh, ako sú klasifikácia alebo predikcia.

Výsledok, ktorý je poslaný z poslednej skrytej vrstvy do výstupnej vrstvy, je kompiláciou transformovaných a integrovaných vlastností, ktoré sú pripravené na finálne rozhodnutia alebo predikcie neurónovej siete.

3.1.3 Výstupná vrstva

Výstupná vrstva je konečnou fázou neurónovej siete, kde sa generujú výsledky na základe informácií spracovaných v skrytých vrstvách. Úlohou výstupnej vrstvy je transformovať aktivácie zo skrytých vrstiev do formy, ktorá zodpovedá požiadavkám špecifického účelu siete. Toto môže zahŕňať klasifikáciu, regresiu, alebo iný typ výstupu.

Počet neurónov vo výstupnej vrstve závisí od typu úlohy, ktorú má sieť riešiť. Pre klasifikačné úlohy počet neurónov obvykle zodpovedá počtu klasifikačných tried, pričom pre binárnu klasifikáciu je často prítomný len jeden neurón vydávajúci pravdepodobnosť príslušnosti k jednej triede. V prípade regresných úloh výstupná vrstva typicky obsahuje jeden neurón, ktorý poskytuje kontinuálne výstupy.

Výstupná vrstva definuje, ako bude výkon siete interpretovaný a využitý. Integrálne zhrňuje naučené reprezentácie zo skrytých vrstiev a premieňa ich na predikcie alebo rozhodnutia.

3.2 Učenie neurónových sietí

Proces učenia neurónových sietí je základom ich schopnosti vykonávať špecifické úlohy, ako je klasifikácia, regresia, alebo rozpoznávanie vzorov. Tento proces zahŕňa viacero krokov, ktoré umožňujú sieti postupne sa zlepšovať v spracovaní a interpretácii dát.

Učenie neurónových sietí prebieha typicky prostredníctvom metódy zvané “spätné šírenie chyby“ (backpropagation), ktorá je spojená s optimalizačným algoritmom, ako je napríklad Adam. Cieľom je minimalizovať rozdiel medzi predikovanými výstupmi siete a skutočnými hodnotami (často nazývaný ako “chyba“ alebo “strata“) prostredníctvom systematickej úpravy váh a prahových hodnôt v neurónoch.

Fázy učenia [4]:

1. **Inicializácia váh:** Pred začiatkom učenia sa váhy a prahové hodnoty (bias) neurónov inicializujú, často s náhodnými hodnotami malého rozsahu. Tento krok je dôležitý, pretože odlišná inicializácia môže viesť k rôznym výsledkom učenia.
2. **Propagácia vpred:** V každej iterácii učenia sa vstupné dáta posielajú cez neurónovú sieť, od vstupnej vrstvy až po výstupnú. V každej vrstve sú dáta transformované pomocou váh a prahových hodnôt spojených s aktiváciou neurónov, a výsledok sa prenáša do nasledujúcej vrstvy.

3. **Výpočet chyby:** Po dosiahnutí výstupnej vrstvy sa vypočíta chyba (strata) porovnaním výstupov siete s očakávanými výsledkami. Táto chyba vyjadruje, ako dobre alebo zle sieť vykonáva zadanú úlohu.

4. **Spätné šírenie chyby:** Chyba sa potom použije na postupné úpravy váh a prahových hodnôt, šíriac sa späť smerom k vstupnej vrstve. Tento proces zahŕňa použitie derivácií chyby vo vzťahu k váham, čo umožňuje efektívne prispôsobenie modelu.

5. **Aktualizácia váh:** Váhy sa aktualizujú na základe vypočítaných gradientov a rýchlosti učenia, ktorá určuje, ako rýchlo sa majú váhy meniť. Menšia rýchlosť učenia môže znamenať presnejšie učenie, ale za cenu dlhšieho trvania procesu.

Učenie neurónových sietí je iteratívny proces, ktorý sa uskutočňuje opakovaným prechádzaním trénovacích dát v dávkach (batches) a epochách, a to až do momentu, kým sieť nedosiahne stanovené ciele týkajúce sa presnosti alebo iných výkonnostných metrík. Každá epocha zahrňuje prechod cez celý trénovací dataset, kde sieť aktualizuje svoje váhy na základe akumulovaných chýb v predikciách.

Počas procesu tréningu je dôležité pravidelne monitorovať výkonnosť modelu nielen na trénovacích dátach, ale aj na validačnej sade, ktorá poskytuje nezávislé hodnotenie výkonnosti siete. Tento krok zabraňuje nadmernému prispôsobeniu sa, známemu ako overfitting, kde model perfektne predikuje trénovacie dáta, ale zlyháva pri generalizácii na nové, nevidené dáta

Aby sme zabezpečili, že model zostane účinný a schopný generalizovať, je nutné pravidelne upravovať parametre učenia, ako sú rýchlosť učenia a veľkosť dávok, a prípadne implementovať techniky, ako je skoré zastavenie (early stopping), kedy tréning končí, ak sa validačná chyba prestane zlepšovať. Tieto kroky pomáhajú zabezpečiť, že neurónová sieť dosiahne optimálnu rovnováhu medzi učením a generalizačnou schopnosťou, vďaka týmto krokom sme schopný úspešne nasadiť model v reálnych aplikáciach.

3.3 Funkcia aktivácie

Funkcia aktivácie v neurónovej sieti je mechanizmom, ktorý rozhoduje, či a ako neurón má byť aktivovaný v reakcii na vstup, ktorý dostáva. Táto funkcia transformuje vstupné signály neurónu na výstupný signál, ktorý je ďalej posielaný do ďalších neurónov v sieti.

Pár príkladov aktivačných funkcií:

- **Lineárna funkcia alebo identity funkcia:** Je to najjednoduchšia forma aktivačnej funkcie, kde výstup je priamo úmerný vstupu. Táto funkcia sa často používa v poslednej vrstve regresných modelov, ale nie je vhodná pre skryté vrstvy, pretože neumožňuje modelom učiť sa komplexné vzory v dátach.

- **Sigmoid alebo logistická funkcia:** Táto funkcia transformuje každý vstupný signál na výstup v rozmedzí medzi 0 a 1, čo ju robí ideálnou pre úlohy binárnej klasifikácie. Sigmoid je užitočný, keď potrebujeme pravdepodobnosti na výstupe, ale trpí problémami ako sú zmiznuté gradienty (vanishing gradients) pri veľmi vysokých alebo nízkych hodnotách vstupu

- **Softmax:** Táto funkcia je rozšírením sigmoidu pre prípady, kde máme viac ako dve klasifikačné kategórie. Softmax vypočíta exponenciálne hodnoty pre každý vstupný

neurón a normalizuje tieto hodnoty tak, aby ich súčet bol rovný jednej. Výstup softmaxu je teda distribúcia pravdepodobnosti medzi viacerými triedami, čo je užitočné v mnohotriednej klasifikácii.

- **ReLU (Rectified Linear Unit)** ReLU je dnes jednou z najpopulárnejších aktivačných funkcií v hlbokých neurónových sietiach, kvôli svojej jednoduchosti a efektivite. Výstup je 0 pre všetky záporné vstupy a lineárne rovný vstupu pre všetky kladné vstupy. ReLU pomáha riešiť problémy zmiznutých gradientov v hlbokých sieťach a zvyšuje rýchlosť konvergencie tréningu.

- **Tanh (hyperbolický tangens):** Tanh je podobná sigmoidu, ale výstup je normalizovaný na rozsah medzi -1 a 1. Toto zlepšuje tréningové vlastnosti siete tým, že normalizuje výstup neurónov, čo vedie k efektívnejšiemu šíreniu gradientov.

Aktivačné funkcie umožňujú modelom učiť sa a reprezentovať nelineárne vzťahy. Bez týchto funkcií by siete mohli vykonávať len lineárne transformácie, čo obmedzuje ich schopnosť spracovávať komplexnejšie údaje ako sú obrázky, zvuky alebo texty. Tým, že poskytujú nelineárnosti, umožňujú neurónovým sieťam stávať sa hlbšími a efektívnejšie modelovať široký rozsah vzorcov a dynamík v dátach.

3.4 Typy neurónových sietí

Neurónové siete sa môžu líšiť vo svojej štruktúre, komplexnosti a účelu, čo umožňuje ich široké využitie v rôznych oblastiach strojového učenia. Rozdielne typy neurónových sietí sú navrhnuté na riešenie špecifických typov problémov, od jednoduchého rozpoznávania vzorcov až po komplexné úlohy ako je rozpoznávanie reči alebo generovanie textu. Nasledujú popisy niekoľkých základných typov neurónových sietí, ktoré sú často využívané v praxi.

3.4.1 Dopredné neurónové siete (Feedforward Neural Networks)

Dopredné neurónové siete sú najjednoduchší a najrozšírenejší typ neurónových sietí. V týchto sieťach informácie postupujú iba jedným smerom, vpred, od vstupných vrstiev, cez skryté vrstvy, až po výstupnú vrstvu. Tento typ siete je často používaný pre klasifikačné a regresné úlohy a je základom pre mnohé zložitejšie siete.

3.4.2 Konvolučné neurónové siete (Convolutional Neural Networks, CNN)

Konvolučné neurónové siete sú špecializované na spracovanie dát s viac dimenzionálnou štruktúrou, ako sú obrázky a videá. Vďaka svojej architektúre, ktorá využíva konvolučné vrstvy na efektívne učenie sa priestorových hierarchií vstupných dát, sú ideálne pre úlohy, ako je rozpoznávanie obrazov a video analýza. Konvolučné vrstvy zachytávajú dôležité vlastnosti ako hrany, farby a textúry.

3.4.3 Rekurentné neurónové siete (Recurrent Neural Networks, RNN)

Rekurentné neurónové siete sú navrhnuté na efektívne spracovanie sekvenčných dát, ako sú časové rady alebo prirodzený jazyk. Na rozdiel od doprednych sietí, RNN majú "pamäť", ktorá im umožňuje uchovať informácie z predchádzajúcich vstupov a použiť ich na ovplyvnenie výstupu pre aktuálny vstup. Táto vlastnosť robí RNN mimoriadne užitočné pre úlohy, ako je predikcia časových radov, rozpoznávanie reči alebo generovanie textu.

3.4.4 Generatívne konkurenčné siete (Generative Adversarial Networks, GAN)

Generatívne konkurenčné siete sú zložené z dvoch súťažiacich sietí: generátora, ktorý vyrába dáta, a diskriminátora, ktorý hodnotí, či sú dáta skutočné alebo vygenerované. Tento model sa používa pre úlohy, kde je cieľom generovať nové dáta, ktoré sú nerozoznateľné od skutočných. GANs sú často využívané v oblastiach ako generovanie umelého obrazu, video hry a vylepšovanie fotografii.

3.5 Typy neurónov v neurónových sieťach

V rámci neurónových sietí existuje niekoľko špecifických typov neurónov, ktoré sa používajú na rôzne účely, v závislosti od štruktúry a funkčnosti modelu. Každý typ neurónu je optimalizovaný pre určité operácie alebo spracovanie dát.

3.5.1 Dense neuróny (Plne prepojené neuróny)

Dense neuróny alebo plne prepojené neuróny, sú základom mnohých neurónových sietí, pričom každý neurón v danej vrstve je prepojený so všetkými aktiváciami z predchádzajúcej vrstvy. Tento typ neurónov je často používaný na integráciu a syntézu informácií z celého dátového setu do finálneho rozhodnutia alebo predikcie, čo umožňuje efektívne spracovanie komplexných úloh ako sú klasifikácia a regresia. Výstupy z týchto neurónov sú vypočítané ako váhovaný súčet vstupov s pridaným biasom, ktorý je následne transformovaný cez aktivačnú funkciu, ako je ReLU alebo sigmoid, na zavedenie potrebnej nelinearity do modelu. Napriek svojej flexibilite a schopnosti integrácie sú dense neuróny výpočtovo náročné a môžu byť náchylné na overfitting, čo vyžaduje použitie techník ako regularizácia alebo dropout na zabezpečenie správneho generalizačného výkonu.

3.5.2 LSTM - neuróny s dlhou krátkodobou pamäťou

LSTM - Long Short Temporary Memory alebo neuróny s dlhou krátkodobou pamäťou, sú typ rekurentných neurónových sietí (RNN) navrhnuté na riešenie problémov s "krátkodobou pamäťou" v štandardných RNN. Tieto neuróny sú schopné uchovať informácie na dlhé časové obdobia a sú efektívne v úlohách, kde je dôležitá sekvenčná závislosť v dátach, ako je spracovanie prirodzeného jazyka, predikcia časových radov alebo rozpoznávanie reči. Na rozdiel od tradičných RNN, ktoré majú tendenciu trpieť problémami so zmiznutím alebo explóziou gradientov pri dlhých sekvenciách, LSTM neuróny používajú štruktúru obsahujúcu tzv. brány - zabudované mechanizmy, ktoré regulujú tok informácií. Tieto brány (vstupná a výstupná) umožňujú LSTM neurónom selektívne zapamätať alebo zabudnúť informácie, čo zlepšuje ich schopnosť učiť sa a udržiavať dôležité informácie počas dlhého časového obdobia bez zbytočného šumu.

Kapitola 4

Návrh aplikácie

Táto kapitola podrobne popisuje návrh webovej aplikácie a predikčného modelu. Cieľom je poskytnúť ucelený pohľad na architektúru aplikácie, využité technológie a spôsob ich integrácie do jednotlivých komponentov aplikácie. Prehľadne sa zameriavame na funkcie aplikácie, jej modulárnu štruktúru a výber použitých technológií.

4.1 Ciele aplikácie

Hlavným cieľom aplikácie je poskytnúť užívateľom nástroj pre predpovedanie cien leteniek a vizualizáciu vývoja týchto cien. Aplikácia je navrhnutá tak, aby splnila nasledujúce špecifické úlohy:

- **Predpovedanie cien leteniek:** Aplikácia umožňuje užívateľom získať predpovede cien leteniek na základe historických údajov a trendov pomocou algoritmov strojového učenia. Tento nástroj pomôže užívateľovi pri kúpe letenky tým, že im poskytne časový vývoj cien daného letu.
- **Vizualizácia vývoja cien:** Kľúčovou funkcionalitou je zobrazovanie grafu, ktorý ukazuje vývoj cien až do dátumu odletu. Tento graf bude umožňovať užívateľom lepšie pozorovať trendy a rozhodnúť sa podľa nich.
- **Užívateľské rozhranie:** Aplikácia bude navrhnutá s dôrazom na jednoduchosť a intuitívnosť, čo zabezpečuje, že aj užívatelia bez technického zázemia môžu ľahko využívať všetky funkcie. Rozhranie zahrňuje jednoduché formuláre pre zadávanie dátumov cesty a výber destinácií.

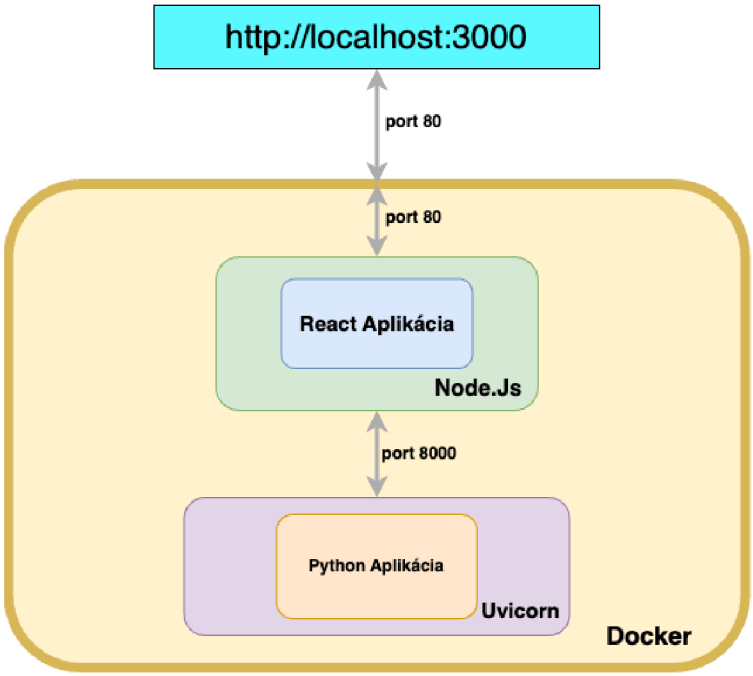
Cieľom týchto funkcií je poskytnúť užívateľom komplexný pohľad na ceny leteniek, umožniť im efektívne rozhodovanie pri kúpe leteniek a napomôcť k optimalizácii ich cestovného rozpočtu. Aplikácia teda slúži ako nástroj pre zlepšenie plánovania cestovania vďaka predpovediam a prehľadným vizualizáciám.

4.2 Architektúra aplikácie

Táto časť podrobne popisuje architektúru aplikácie na predpovedanie cien leteniek, ktorá je založená na REST architektúre. Tento prístup umožňuje efektívnu a flexibilnú komunikáciu medzi rôznymi časťami systému a zaisťuje, že aplikácia je škálovateľná a ľahko prispôsobiteľná budúcim potrebám.

Ako je znázornené na Obrázku 4.1, architektúra obsahuje niekoľko kľúčových vrstiev a komponentov, ktoré spolupracujú na spracovaní požiadaviek:

1. **Vstupný bod (http://localhost:3000)**:
   - Schéma začína adresou `http://localhost:3000`, ktorá je lokálna adresa, na ktorej je systém dostupný. Toto je vstupný bod pre užívateľské požiadavky.
2. **React Aplikácia (port 80)**:
   - Požiadavky smerujúce na túto adresu sú presmerované na React aplikáciu bežiacu na porte 80. React aplikácia je postavená na Node.js.
3. **Komunikácia medzi React a Python aplikáciou**:
   - React aplikácia následne komunikuje s Python aplikáciou prostredníctvom portu 8000. Node.js v tomto kontexte poskytuje serverové prostredie pre React aplikáciu.
4. **Python Aplikácia (Uvicorn, v Dockeri)**:
   - Python aplikácia beží na Uvicorn serveri, ktorý prijíma požiadavky od React Aplikácie. Uvicorn ako ASGI server optimalizuje asynchrónnu komunikáciu.
5. **Docker**:
   - Celá aplikácia je zabalená do Docker kontajneru, čo zjednodušuje nasadenie a zabezpečuje konzistentné prostredie pre všetky komponenty.



Obr. 4.1: Schéma aplikácie

4.2.1 Frontend

Implementované pomocou: React.js na Node.js platforme
Úloha: Frontend poskytuje dynamické užívateľské rozhranie, cez ktoré môžu užívatelia interagovať s aplikáciou. React.js bol vybraný kvôli jeho schopnosti efektívne spracovávať dynamické aktualizácie a udržiavať aplikáciu reaktívnu bez potreby opätovného načítania. Node.js poskytuje serverové prostredie pre spustenie React.js, čo umožňuje vyššiu efektivitu pri spracovaní požiadaviek a zjednodušuje vývoj jednostránkových aplikácií (SPA).

4.2.2 Backend

Implementované pomocou: FastAPI, Uvicorn
Úloha: Backend spracováva logiku aplikácie, manipuláciu s dátami a poskytuje predpovede cien leteniek frontendu. FastAPI je využívaný pre jeho rýchlosť a podporu asynchrónnych operácií, zatiaľ čo Uvicorn ako ľahký ASGI server zaisťuje efektívne spracovanie požiadaviek.

4.2.3 Docker

Úloha: Docker je využívaný na kontajnerizáciu celej aplikácie, čo zjednodušuje nasadenie a správu prostredí od vývoja po produkciu. Kontajnery Docker zabezpečujú konzistentnosť prostredia a minimalizujú problémy s kompatibilitou.

4.3 Podrobnejší popis použitých technológií

Táto sekcia podrobne rozoberá súbor technológií a nástrojov používaných na vývoj, nasadenie a prevádzku našej aplikácie na predpovedanie cien leteniek. Výber technológií bol motivovaný potrebou efektívnosti, bezpečnosti a snahou o minimalizáciu komplexnosti pri udržiavaní systému.

4.3.1 Frontend Technológie

React.js[1]

Jedná sa o deklaratívnu, efektívnu a flexibilnú JavaScript knižnicu pre stavbu užívateľských rozhraní. Je ideálna pre vytvorenie dynamických a responzívnych SPA (Single Page Applications), ktoré vyžadujú neustálu interakciu s užívateľom a rýchle aktualizácie bez nutnosti obnovovania celej stránky.

Node.js[2]

Node.js je platforma postavená na Chrome V8 JavaScript engine. Používa sa ako serverové prostredie pre beh React.js aplikácií.

[1] https://react.dev/
[2] https://nodejs.org/

4.3.2 Backend Technológie

FastAPI[3]

Moderný, rýchly webový framework pre Python, navrhnutý špeciálne pre vytvorenie API s vysokým výkonom. Podporuje asynchrónne operácie a automatickú generáciu dokumentácie API.

Uvicorn[4]

Vysoko výkonný ASGI server, ktorý je optimalizovaný pre rýchle a efektívne spracovávanie HTTP požiadaviek v asynchrónnom režime.

4.3.3 Kontajnerizácia a Orchestration

Docker a Docker Compose[5]

Docker poskytuje robustné riešenie pre kontajnerizáciu aplikácií, umožňujúc im behať v izolovaných prostrediach, ktoré zabezpečujú konzistenciu medzi vývojovými a produkčnými prostrediami. Docker Compose je nástroj, ktorý zjednodušuje proces definície, správy a spúšťania viacerých Docker kontajnerov. Umožňuje konfiguráciu viacerých kontajnerových služieb, sietí a úložísk v jednotnom súbore, čo zjednodušuje nasadzovanie komplexných aplikácií. Toto spojenie Docker technológií tak vytvára efektívne a škálovateľné riešenie pre nasadenie a správu našej aplikácie.

4.4 Využité knižnice a frameworky

V tejto časti sa podrobne venujeme knižniciam a frameworkom, ktoré sú kľúčové pre funkčnosť našej aplikácie na predpovedanie cien leteniek. Každá z týchto technológií prispieva k rôznym aspektom aplikácie, od spracovania dát až po užívateľské rozhranie.

4.4.1 Dátová analýza a spracovanie

Polars[6]

Polars je knižnica pre manipuláciu s dátami, navrhnutá pre vysokú rýchlosť a efektívne paralelné spracovanie. Na rozdiel od Pandas, ktorý je vhodný pre menšie až stredné dátové sady, Polars exceluje pri práci s veľkými dátovými množinami, vďaka svojej optimalizácii pre výkon a nízku pamäťovú stopu. Táto knižnica poskytuje funkcie podobné tým v Pandas, ale s lepšou škálovateľnosťou a rýchlosťou.

NumPy[7]

Základná knižnica pre numerické výpočty v Python. NumPy podporuje operácie s veľkými, viacrozmernými poliami a maticami, poskytujúc rozsiahlu kolekciu matematických funkcií potrebných pre analýzu a modelovanie dát.

[3] https://fastapi.tiangolo.com/
[4] https://www.uvicorn.org/
[5] https://www.docker.com/
[6] https://pola.rs/
[7] https://numpy.org/

4.4.2 Prediktívne modelovanie

Scikit-lear[8]

Scikit-learn je knižnica pre jazyk Python, ktorý sa používa pre strojové učenie. Poskytuje širokú škálu populárnych algoritmov pre klasifikáciu, regresiu a klastrovanie dát. Je navrhnutý tak aby bol vysoko výkonný pri spracovaní veľkých dát [14].

TensorFlow[9]/Keras[10]

TensorFlow, spolu s jeho vysokoúrovňovým API Keras, tvorí základ našej infraštruktúry pre hlboké učenie. Táto kombinácia nám umožňuje efektívne vytvárať a trénovať pokročilé modely neurónových sietí.

4.4.3 Vizualizácia dát

Matplotlib[11]

Knižnica pre vizualizáciu dát v Python. Matplotlib umožňuje vytváranie širokej škály statických vizualizácií. Je nevyhnutná pre zobrazovanie výsledkov našich analýz a modelov v zrozumiteľnej forme.

[8] https://scikit-learn.org/
[9] https://www.tensorflow.org/
[10] https://keras.io/
[11] https://matplotlib.org/

Kapitola 5

Implementácia predikčného modelu

Táto kapitola detailne opisuje vývoj prediktívneho modelu na predpovedanie cien leteniek z dát poskytnutých spoločnosťou Kiwi.com. Pri predspracovaní dát je dôležité rozdelenie dátovej sady a výber vhodnej architektúry modelu, aby bola zabezpečená jeho efektivitu a presnosť.

5.1 Dátová sada

Táto analýza sa opiera o dátovú sadu získanú od spoločnosti Kiwi.com, ktorá poskytuje služby v oblasti vyhľadávania a rezervácie leteniek. Táto dátová sada obsahuje agregované dáta o vyhľadávaniach letov uskutočnených užívateľmi, kde každý záznam predstavuje agregáciu dát podľa hodiny vyhľadávania (*search_hour*) na hodinovej báze. V dátovej sade sa nachádza skoro 18 miliónov záznamov. Každý záznam v dátovej sade obsahuje informácie, ktoré sú podrobne popísané v Tabuľke 5.1.

Atribút	**Popis**
search_hour	Čas vyhľadávania, agregovaný po hodinách, vo formáte *YYYY-MM-DD HH:MM:SS* v UTC
itinerary_src	Kód letiska odletu
itinerary_dst	Kód letiska príletu
distance	Vzdialenosť medzi letiskami odletu a príletu v kilometroch
avg_basefare_per_pax	Priemerná základná cena letenky na cestujúceho, vypočítaná z agregovaných dát
outbound_dTimeUTC	Plánovaný čas odletu v UTC
outbound_aTimeUTC	Plánovaný čas príletu v UTC
flights	Identifikátor letu
n_search_results	Počet výsledkov vyhľadávania pre daný let počas agregovanej hodiny

Tabuľka 5.1: Popis atribútov v dátovej sade

5.1.1 Kiwi.com

Kiwi.com je online cestovacia agentúra, ktorá sa špecializuje na predaj cenovo dostupných leteniek. Spoločnosť bola založená v roku 2012 a sídli v Brne v Českej republike. Kiwi.com sa využíva unikátny algoritmus, ktorý umožňuje nájsť výhodné letenky kombináciou rôznych spojení od rozličných dopravcov, vrátane nízkonákladových a tradičných leteckých spoločností. Ponúka tiež širokú škálu ďalších služieb, ako sú poistenie, ubytovanie a transferové služby.

5.2 Predspracovanie dát

Na identifikáciu príznakov boli vybrané tie znaky, ktoré majú významný dopad na cieľovú premennú, teda cenu letenky. Zahrnuté boli informácie o odletových a príletových destináciách, ako aj časové údaje týkajúce sa vyhľadávania a odletu, vrátane dát o sviatkoch. Vzhľadom na to, že dáta neobsahujú informácie o tom, aký typ služby (napríklad ekonomická alebo biznis trieda) si cestujúci zvolil, boli vybraté dáta týkajúce sa len vybraných leteckých spoločností W6 (WizzAir) a FR (Ryanair).

Následne bolo vykonané čistenie dát, ktoré zahŕňalo overenie úplnosti a správnosti informácií v našej dátovej sade. Keďže neboli neidentifikované žiadne závažné chyby alebo neúplnosti, tento krok bol relatívne jednoduchý a rýchly.

Poslednou časťou predspracovania, bola transformácia dát. Bolo nevyhnutné premeniť kategorické premenné na číselné hodnoty vhodné pre strojové učenie. Tento proces bol realizovaný pomocou techniky One-Hot Encoding, ktorá premieňa kategorické atribúty na binárny formát. Metóda bola aplikovaná na premenné, ako sú deň v týždni, deň v mesiaci a mesiac v roku, čo umožňuje modelu rozpoznávať a naučiť sa dôležité časové vzorce.

Celý proces predspracovania bol realizovaný s použitím knižnice Polars v programovacom jazyku Python, čo nám umožnilo efektívne spracovať veľké objemy dát. Výsledné predspracované dáta boli uložené do súboru CSV, ktorý poskytuje univerzálny a jednoducho manipulovateľný formát pre načítanie a ďalšie spracovanie dát v rámci rôznych analytických platforiem.

5.3 Rozdelenie dát

Pred samotným vývojom prediktívneho modelu bolo potreba rozdeliť dátový súbor na trénovaciu, validačnú a testovaciu sadu. Týmto krokom pre zabezpečíme schopnosti modelu generalizovať naučené vzorce na dátach, ktoré neboli počas tréningu videné, a predchádza tak problému preučenia. Rozdelenie dát na tieto tri sady zabezpečuje, že proces vývoja modelu je robustný a výsledný model bude mať dobrú generalizačnú schopnosť. Implementácia rozdelenia bola realizovaná s použitím funkcie `train_test_split` z knižnice `sklearn.model_selection`, ktorá umožnila náhodne rozdeliť dáta v pomere 70% pre trénovaciu sadu a zvyšných 30% rovnomerne rozdeliť medzi validačnú a testovaciu sadu.

Náhodné rozdelenie eliminuje potenciálny problém so zoradenými dátami vo vzorkách. Predstavme si, že naše dáta pochádzajú z rôznych časových období a sú predbežne zoradené chronologicky. Ak by sme tieto dáta nerozdelili náhodne, mohli by sa všetky trénovacie dáta týkať starších prípadov, zatiaľ čo testovacie dáta by obsahovali len najnovšie prípady. Toto by mohlo spôsobiť, že model bude neefektívny pri predikcii aktuálnych alebo budúcich trendov, pretože nebol trénovaný na relevantných, súčasných dátach. Náhodné rozdelenie

zabezpečí, že každá sada obsahuje vzorky z rôznych časových období, čím sa zvyšuje pravdepodobnosť, že model bude robustný a spoľahlivý aj v reálnom prostredí.

5.4 Vývoj modelu

Táto časť sa venuje konkrétnym krokom a technikám, ktoré boli aplikované pri vývoji prediktívneho modelu. Cieľom bolo vytvoriť model, ktorý by bol schopný s vysokou presnosťou predpovedať priemernú cenu na cestujúceho na základe dostupných dát.

Je dôležité spomenúť, že pôvodná dátová sada bola zredukovaná z dôvodu jej veľkosti, ktorá predstavovala výzvu pri trénovaní neurónových sietí. Aj napriek disponovaniu serverom s grafickou kartou Nvidia RTX A5000, trénovanie modelu na kompletnej sade trvalo nadmieru dlho a výsledky neboli dostatočne presné. Tento fakt viedol k rozhodnutiu sústrediť sa na trénovanie neurónových sietí na okresanej sade, konkrétne na letovej trase PRG->FCO pre letecké spoločnosti W6 a FR. Tento prístup umožnil nielen zrýchlenie trénovania, ale aj optimalizáciu neurónových sietí pre linky, čo predstavuje výhodu pre zvýšenie presnosti modelov.

5.4.1 Architektúra modelu

Model bol postavený s využitím hlbokého učenia a konkrétne siete typu Dense 3.5.1, ktoré sú známe svojou schopnosťou efektívne spracúvať veľké objemy dát a identifikovať v nich komplexné vzorce. Architektúra modelu pozostávala z niekoľkých vrstiev:

- **Vstupná vrstva:** Prvá vrstva modelu, ktorá prijíma vstupné dáta. Veľkosť vstupnej vrstvy bola nastavená na počet vybraných príznakov po predspracovaní dát a to 179. Veľká časť týchto vstupov tvorí one-hot encoding pre časové atribúty a to čas vyhľadávania, čas odletu a čas príletu. Bližšie špecifikované atribúty sú popísané v kapitole 7.3.
- **Skryté vrstvy:** Model bol testovaný s viacerými nastaveniami skrytých vrstiev, tie sú bližšie popísané v kapitole 7.1.
- **Výstupná vrstva:** Posledná vrstva modelu, ktorá generuje predpoveď cieľovej premennej. Vzhľadom na to, že úlohou bolo predpovedať kontinuálne hodnoty (priemernú cenu), výstupná vrstva obsahovala jediný neurón s lineárnou aktivačnou funkciou.

5.4.2 Trénovanie modelu

Trénovanie modelu bolo realizované s použitím knižnice `TensorFlow` a jej vysokoúrovňového rozhrania `Keras`. Na optimalizáciu váh modelu bol použitý algoritmus *Adam*, ktorý je známy svojou efektivitou v rôznych typoch úloh hlbokého učenia. Ako funkciu straty bola zvolená *Mean Squared Error (MSE)*.

Počas trénovania modelu bolo kľúčové monitorovať výkonnosť nielen na trénovacej sade, ale aj na validačnej sade, aby sa predišlo preučeniu. Jednou z techník, ktoré boli aplikované, bolo *predčasné ukončenie* (*early stopping*), kde trénovanie modelu sa automaticky zastaví, ak sa výkonnosť na validačnej sade nezlepšuje.

Okrem vyššie uvedených stratégií, významnú časť procesu trénovania modelu tvorilo aj experimentovanie s rôznymi konfiguráciami skrytých vrstiev. Rozhodli sme sa preskúmať vplyv modelu na jeho výkonnosť, a preto sme vytvorili viaceré varianty modelu, ktoré sa líšili počtom skrytých vrstiev. Cieľom bolo zistiť, či zvýšenie počtu skrytých vrstiev prinesie

zlepšenie v presnosti predpovedí alebo, naopak, či to povedie k preučeniu modelu v dôsledku prílišnej komplexnosti.

V kontexte experimentovania s rôznymi konfiguráciami skrytých vrstiev bol kladený dôraz aj na testovanie rozličných počtov neurónov v jednotlivých vrstvách. Bolo analyzované, ako počet neurónov ovplyvňuje schopnosť modelu zachytiť nuansy v dátach a či existuje určitá optimalizovaná veľkosť pre našu konkrétnu úlohu.

Tieto experimenty nám poskytli cenné poznatky o tom, ako architektúra modelu ovplyvňuje jeho schopnosť učiť sa z dát a generalizovať. Bolo zistené, že existuje malá rovnováha medzi pridaním ďalších skrytých vrstiev na zvýšenie modelovej kapacity a rizikom preučenia, ktoré s tým môže byť spojené. Na základe týchto experimentov bolo možné identifikovať konfiguráciu modelu, ktorá ponúkla najlepšiu rovnováhu medzi výkonnosťou a generalizáciou.

Výsledkom týchto experimentov bolo lepšie porozumenie dôležitosti správneho navrhnutia architektúry modelu a toho, ako rôzne aspekty konfigurácie vrstiev ovplyvňujú výslednú presnosť modelu. Táto fáza vývoja bola zásadná pre optimalizáciu našeho prediktívneho modelu, umožňujúca nám dosiahnuť vyváženú kombináciu medzi dostatočnou modelovou kapacitou na učenie sa zložitých vzorcov v dátach a zároveň udržaním schopnosti modelu generalizovať na nevidených dátach.

Pri trénovaní rôznych konfigurácií modelu bol taktiež kladený dôraz na optimalizáciu procesu učenia prostredníctvom rôznych nastavení optimalizátorov. Experimenty boli prevádzané nielen s rôznymi typmi optimalizátorov, ako sú Adam, SGD, ale aj s ich špecifickými nastaveniami, vrátane veľkosti kroku (learning rate), aktivačných funkcií a batch size.

Veľkosť kroku (Learning Rate): Jedným z kľúčových parametrov pri nastavovaní optimalizátora bol výber veľkosti kroku. Experimentovali sme s rôznymi hodnotami (viď. 7.2), aby sme našli optimálnu rýchlosť učenia. Bolo zistené, že najlepší learning rate pre naše účely bol 0.001, čo umožnilo modelu efektívne konvergovať k minimu stratovej funkcie bez rizika "presprievodzenia"cez minimum

Batch Size: "Veľkosť dávky" je ďalším zásadným parametrom, ktorý ovplyvňuje proces trénovania neurónovej siete. Počas experimentovania sme testovali rôzne veľkosti dávok, aby sme zistili, ako veľkosť dávky ovplyvňuje rýchlosť a stabilitu konvergencie modelu (viď. 7.2).

Tieto experimenty s rôznymi nastaveniami optimalizátorov umožnili lepšie pochopiť dynamiku trénovacieho procesu a identifikovať kombináciu parametrov, ktorá najefektívnejšie viedla k naučeniu modelu. Bolo zistené, že správne nastavenie optimalizátora môže výrazne ovplyvniť schopnosť modelu naučiť sa komplexné vzorce v dátach a zároveň udržať stabilitu a efektivitu trénovania.

Tieto poznatky boli nesmierne cenné pre finálnu fázu vývoja modelu, kde sme hľadali najlepšiu rovnováhu medzi rýchlosťou učenia, presnosťou modelu a jeho schopnosťou generalizácie.

5.4.3 Vyhodnotenie modelu

Po dokončení trénovania bolo vykonané konečné vyhodnotenie modelu na testovacej sade, ktorá bola modelu doposiaľ neznáma. Výsledky vyhodnotenia poskytli dôležité informácie o výkonnosti modelu v reálnych podmienkach.

Kapitola 6

Implementácia webovej aplikácie

Táto kapitola podrobne rozoberá implementáciu webovej aplikácie na predpovedanie cien leteniek. Skúma technické aspekty vývoja a poskytuje detailný pohľad na vybrané technológie a metódy, ktoré boli použité pri vytváraní efektívneho, bezpečného a škálovateľného riešenia. Bližší návrh je popísaný v kapitole 4.2.

6.1 Prostredie

Docker je kľúčový v implementácii našej aplikácie, zabezpečujúci konzistenciu prostredí a zjednodušenie. Nižšie sú podrobnosti o tom, ako sme integrovali Docker do nášho vývojového procesu a aké výhody to prinieslo.

6.1.1 Dockerfiles

Backend: Naša backendová komponenta aplikácie využíva vlastný `Dockerfile`, ktorý umožňuje vytvorenie izolovaného prostredia pre Python aplikáciu. `Dockerfile` špecifikuje základný obraz Python, nainštaluje potrebné závislosti uvedené v súbore `requirements.txt` a nakopíruje zdrojové kódy aplikácie do kontajnera. Takto pripravený kontajner obsahuje všetky potrebné súbory a závislosti na spustenie backendu a je pripravený na interakciu s frontendom.

Frontend: Frontend používa samostatný Dockerfile, ktorý zahŕňa inštaláciu Node.js a kopírovanie všetkých potrebných súborov aplikácie do kontajnera. Tento prístup zabezpečuje, že všetky závislosti frontendu sú správne nainštalované a aplikácia je pripravená na spustenie v izolovanom a konzistentnom prostredí.

6.1.2 Docker Compose

Pre spojenie viacerých kontajnerov (backend, frontend) a zjednodušenie konfigurácie sme použili `docker-compose.yaml`. Tento súbor definuje služby, sieťové nastavenia a závislosti medzi kontajnermi. Docker Compose nám umožňuje s jedným príkazom spustiť celú aplikáciu, čo zjednodušuje vývoj a testovanie. Využitie Docker Compose tiež pomáha pri simulácii produkčného prostredia na lokálnom stroji, čím sa minimalizujú problémy s nasadením.

Výhody Dockeru

Integrácia Dockeru priniesla výrazné výhody, vrátane:

- **Konzistencia prostredia:** Zabezpečuje rovnaké prostredie na všetkých vývojových a na možných produkčných strojoch.
- **Izolácia:** Každá komponenta aplikácie beží izolovane, čo znižuje konflikty závislostí a problémy so závislosťami.
- **Škálovateľnosť:** Kontajnery sa dajú ľahko škálovať a spravovať v rámci väčších nasadení pomocou orchestračných nástrojov ako je napríklad Kubernetes.
- **Jednoduchosť nasadenia:** Nasadenie nových verzií aplikácie alebo aktualizácie závislostí je jednoduchšie a menej náchylné na chyby.

6.2 Implementácia Backendu

Táto sekcia poskytuje detailný pohľad na implementáciu backendu webovej aplikácie na predpovedanie cien leteniek. Rozoberá kľúčové komponenty, technológie a postupy, ktoré boli použité na vytvorenie robustného a efektívneho serverového riešenia.

6.2.1 Kľúčové súbory a moduly

api.py: Tento modul slúži ako hlavný vstupný bod pre API aplikácie. Definuje cesty (routy) pre API, ktoré umožňujú užívateľom interagovať s aplikáciou prostredníctvom HTTP požiadaviek. Každá cesta je asociovaná s funkciou, ktorá spracováva príslušné požiadavky a vracia odpovede. Tento súbor tiež obsahuje konfiguráciu middleware, ktorý zabezpečuje správnu bezpečnosť API.

predictor.py: Modul obsahuje logiku potrebnú pre načítanie modelov strojového učenia, ich inicializáciu a vykonávanie predpovedí. Tento súbor zahŕňa funkcie pre manipuláciu s dátami, výpočet predpovedí a ich formátovanie pre ďalšie použitie v aplikácii. Umožňuje rýchle načítanie predtrénovaných modelov a zabezpečuje, že predpovede sú dostupné na požiadanie s minimálnym oneskorením.

6.3 Implementácia Frontendu

Táto sekcia popisuje implementačné detaily frontendu webovej aplikácie na predpovedanie cien leteniek, ktorý je postavený s použitím knižnice React.js. Rozoberá sa štruktúra, kľúčové komponenty a interakcia s backendom.

6.3.1 Hlavná štruktúra a komponenty

Frontend aplikácie sa skladá z niekoľkých hlavných súborov a komponentov, ktoré sú základom užívateľského rozhrania.

index.html: Základný HTML súbor, ktorý slúži ako vstupný bod pre React aplikáciu. Obsahuje len jednoduchý kontajner `div` s identifikátorom `root`, do ktorého React dynamicky vkladá celú aplikáciu.

App.js: Koreňový komponent Reactu, ktorý zahŕňa hlavnú logiku pre zobrazovanie užívateľského rozhrania. Tento komponent importuje a používa `FlightPricePredictor`, ktorý je hlavným komponentom pre predikciu cien leteniek.

FlightPricePredictor.js: Komplexný React komponent, ktorý implementuje logiku a vizualizáciu predikcie cien leteniek. Umožňuje užívateľom vyplniť požadované údaje, ako sú dátumy, letiská a dopravcovia, a zaslať tieto informácie na server prostredníctvom HTTP požiadaviek pomocou knižnice axios.

6.3.2 Interakcia s backendom

`FlightPricePredictor.js` zabezpečuje komunikáciu s backendovými API prostredníctvom HTTP požiadaviek, ktoré odosielajú a prijímajú dáta potrebné pre funkcionality predikcie. Užívateľské vstupy sú spracované pomocou formulárových elementov a odoslané na server, kde sú vyhodnotené a vrátené ako predpovede.

6.3.3 Vizualizácia údajov

Komponent obsahuje vizualizačné nástroje, ktoré zobrazujú predikcie cien v grafe. Použitá knižnica `react-chartjs-2` spolu s `Chart.js` umožňuje detailné zobrazenie predikčných dát v interaktívnom grafe, ktorý užívateľom poskytuje ľahko pochopiteľný a vizuálne prehľadný graf o cenových trendoch.

Kapitola 7

Experimentálne výsledky

V tejto kapitole je popísaná séria experimentov, ktoré boli vykonané na evaluáciu rôznych architektúr neurónových sietí použitých na predikciu cien leteniek. Každý experiment sa zameriava na rôzne aspekty modelovania, vrátane architektúr sietí, optimalizácie parametrov a porovnania výkonností. Všetky experimenty boli realizované na MacBooku s čipom M1 Pro.

7.1 Experiment 1: Hľadanie optimálneho nastavenia počtu vrstiev a jej počet neurónov

V experimente sa preskúmava, ako rôzne konfigurácie hĺbky (počet vrstiev) neurónovej siete ovplyvňujú jej schopnosť presne predpovedať ceny leteniek. Cieľom je identifikácia optimálnej hĺbky siete, ktorá umožňuje dosiahnuť najvyššiu presnosť predpovede pri zachovaní efektívnej konvergencie trénovacieho procesu, a tiež sa pozoruje, za aký čas sa neurónová sieť dokáže natrénovať.

Experiment začína s najjednoduchším modelom, ktorý má jedinú skrytú vrstvu, a postupne sa zvyšuje počet vrstiev. Každá nová konfigurácia modelu je podrobená dôkladnému testovaniu, pri ktorom sa sleduje jej presnosť a rýchlosť konvergencie. Presnosť modelov sa meria pomocou strednej kvadratickej chyby (MSE - Mean Squared Error), ktorá poskytuje kvantitatívny základ pre porovnanie účinnosti jednotlivých architektúr.

Pridávaním ďalších vrstiev je sledované, ako sa mení schopnosť modelu zachytiť zložitejšie vzťahy v dátach, pričom zároveň je brané do úvahy potenciálne riziko preučenia, ktoré môže viesť k horšej generalizácii na nových dátach. Týmto postupom je postupne odhaľované, ktorá konfigurácia poskytuje optimálnu rovnováhu medzi presnosťou a trénovateľnosťou.

Pre zlepšenie efektivity trénovania a prevenciu pretrénovania modelu je využívaná funkcia EarlyStopping z Keras. Táto funkcia monitoruje **val_loss** (stratu na validačnej sade) a má nastavenú trpezlivosť (patience) na 5 epoch. Trénovanie sa zastaví, ak sa počas týchto 5 epoch nezaznamená zlepšenie vo validačnej strate, čím sa zaisťuje, že model neztráca schopnosť generalizácie na nové dáta. Funkcia EarlyStopping je aktivovaná od 5. epochy, aby model mal dostatočný čas na adaptáciu pred spustením predčasného ukončenia.

Výsledky každej iterácie sú dokumentované a analyzované. Pre každý model je poskytnutá tabuľka s údajmi o konfigurácii vrstiev, počte neurónov a výsledných metrikách.

7.1.1 Experiment s jednou vrstvou

Táto časť práce je zameraná na porovnanie výkonnosti neurónových sietí s jednou vrstvou pri rôznom počte neurónov a na pochopenie, ako tieto faktory ovplyvňujú presnosť predikcií cien leteniek, ktoré sú podrobne špecifikované v tabuľke 7.1. Experimenty boli rozdelené do dvoch hlavných skupín na základe počtu neurónov: modely s nižším počtom neurónov (M1 až M8) a modely s vysokým počtom neurónov (M9 až M12).

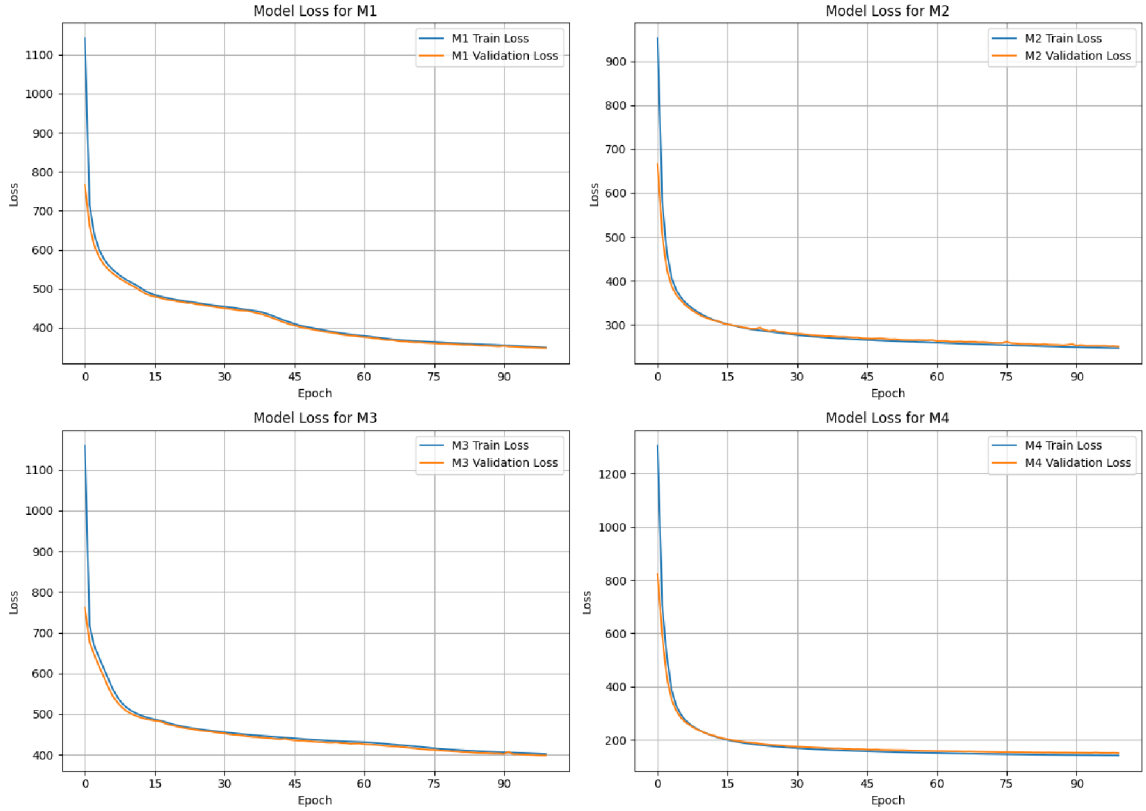
Model ID	Skryté vrstvy	Typ neuronov a aktivácie	Počet neurónov na vrstvu	Výsledky Výsledky MSE	Čas [s]
M1	1	Dense ReLU	10	333.970	139.059
M2	1	Dense ReLU	40	257.876	161.783
M3	1	Dense Tanh	10	278.930	141.622
M4	1	Dense Tanh	40	152.846	165.927
M5	1	Dense ReLU	80	190.163	164.675
M6	1	Dense ReLU	160	136.553	171.315
M7	1	Dense Tanh	80	100.561	170.497
M8	1	Dense Tanh	160	73.625	199.368
M9	1	Dense ReLU	300	111.539	124.416
M10	1	Dense ReLU	800	69.893	191.790
M11	1	Dense Tanh	300	61.771	178.059
M12	1	Dense Tanh	800	52.865	163.293

Tabuľka 7.1: Prehľad testovaných modelov neurónových sietí s detailmi o počte neurónov v jednotlivých vrstvách pre modely M1 až M12

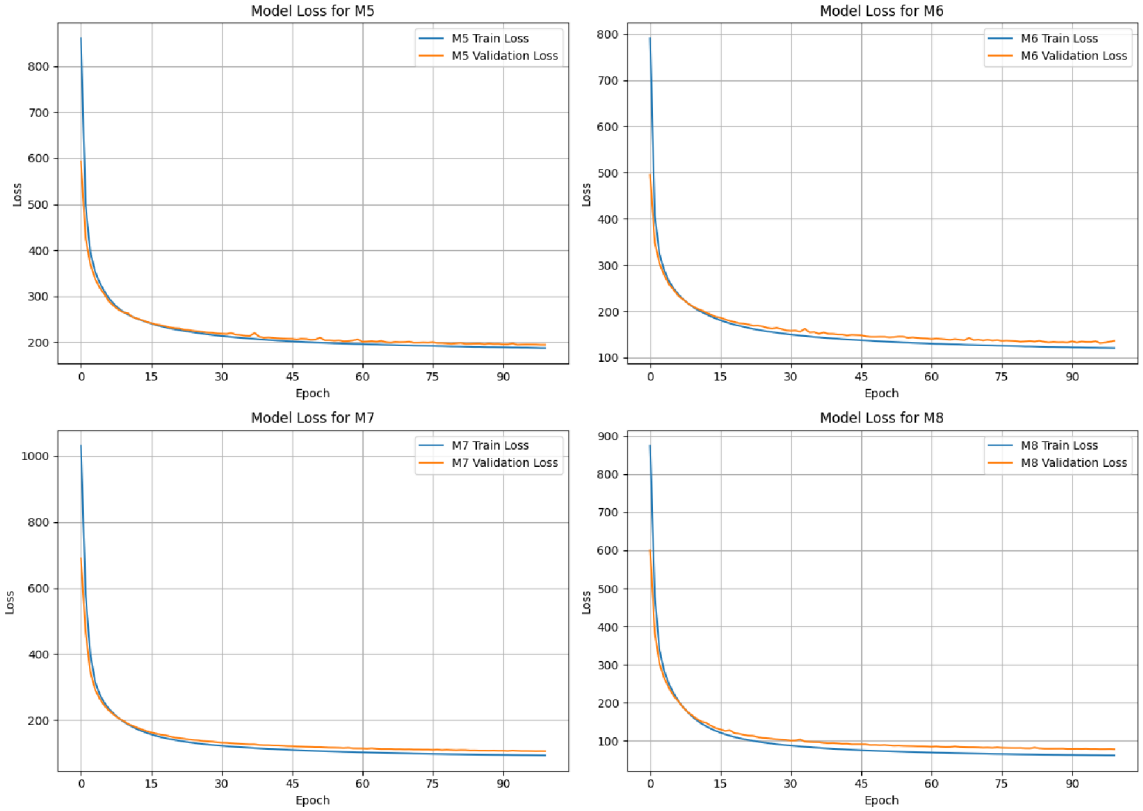
V prvej skupine modelov M1 až M8 boli analyzované neurónové siete s počtom neurónov od 10 do 160 a s aktivačnými funkciami ReLU a Tanh. Zistilo sa, že model M1 s 10 neurónmi a aktiváciou ReLU mal najvyššiu hodnotu MSE, čo naznačuje slabú predikčnú schopnosť. Naopak, modely s vyšším počtom neurónov vykazovali lepšie výsledky, pričom najlepšie výsledky dosiahol model M8 s 160 neurónmi a Tanh aktiváciou. Tento trend ukazuje, že zvýšenie počtu neurónov môže zlepšiť predikčnú schopnosť siete, najmä ak je použitá vhodná aktiváčná funkcia.

V druhej skupine experimentov s modelmi M9 až M12 boli skúmané efekty značného zvýšenia počtu neurónov, kde sme dosiahli počty od 300 do 800 neurónov. Aj tu boli použité aktivačné funkcie ReLU a Tanh. Modely najmä s Tanh aktiváciou (M11 a M12), ukázali výrazné zlepšenie v predikčných schopnostiach, pričom model M12 s 800 neurónmi a Tanh aktiváciou dosiahol najnižšie MSE. Zdá sa, že Tanh poskytuje lepšie výsledky v kombinácii s veľkým počtom neurónov, možno kvôli svojej schopnosti efektívne riešiť komplexnejšie predikčné úlohy.

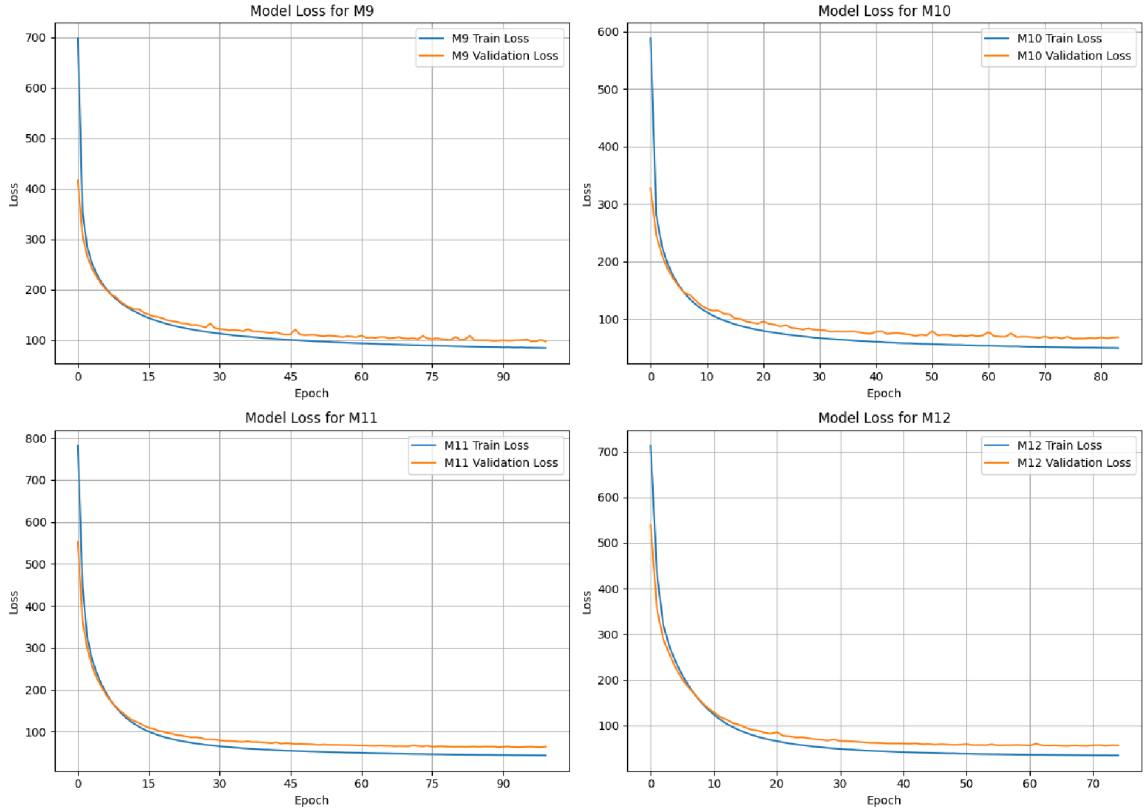
V grafe 7.1 a 7.2 pre jednotlivé modely je možné pozorovať, že validačné straty neprejavujú tendenciu pretrénovania, čo podčiarkuje ich schopnosť generalizovať a predpovedať ceny leteniek s vysokou spoľahlivosťou. Na základe týchto pozorovaní budú v ďalších experimentoch skúmané architektúry s vyšším počtom neurónov a zvýšeným počtom skrytých vrstiev.



Obr. 7.1: Porovnanie trénovacej a validačnej straty pre modely M1 až M4



Obr. 7.2: Porovnanie trénovacej a validačnej straty pre modely M5 až M8



Obr. 7.3: Porovnanie trénovacej a validačnej straty pre modely M9 až M12

7.1.2 Experiment s viacerými vrstvami

V poslednej sérii experimentov bola zameraná pozornosť na analýzu výkonnosti neurónových sietí s dvoma skrytými vrstvami, pričom sa skúmalo, ako zmeny v architektúre ovplyvňujú schopnosť modelov presne predpovedať ceny leteniek, tieto nastavenia sú podrobne popísané v tabuľke 7.2. Testy boli rozdelené na dve hlavné skupiny: prvú skupinu modelov M13 až M16 s miernym počtom neurónov a druhú skupinu modelov M17 až M20 s výrazne zvýšeným počtom neurónov.

Prvá skupina modelov (M13 až M16) zahŕňala siete s relatívne vyšším počtom neurónov vo dvoch vrstvách, pričom boli použité aktivácie ReLU a Tanh. V druhej skupine (M17 až M20) bolo experimentované s ešte vyšším počtom neurónov. Pozorovania naznačujú, že zvyšovanie počtu neurónov a zmena typu aktivácie môžu prinášať lepšie predikčné výsledky. Zaujímavým zistením bolo, že dlhšie trvanie trénovania nie je nutne priamo spojené so zvýšeným počtom neurónov alebo komplexnejšími modelmi, ale s použitými aktivačnými funkciami.

Pri detailnejšom pohľade na časy trénovania jednotlivých modelov sa zistilo, že časy sú porovnateľné medzi modelmi s rôznymi konfiguráciami a počtami neurónov, čo naznačuje, že faktory ako typ aktivácie a špecifická implementácia trénovacej rutiny môžu mať takisto významný vplyv na výpočtové nároky.

Model ID	Skryté vrstvy	Typ neuronov a aktivácie	Počet neurónov na vrstvu	Výsledky MSE	Čas [s]
M13	2	Dense ReLU	120, 120	68.525	124.437
M14	2	Dense ReLU	200, 200	53.904	138.867
M15	2	Dense Tanh	120, 120	59.445	146.576
M16	2	Dense Tanh	200, 200	51.406	236.015
M17	2	Dense ReLU	300, 400	55.065	223.370
M18	2	Dense ReLU	600, 400	53.116	229.055
M19	2	Dense Tanh	300, 400	51.560	330.620
M20	2	Dense Tanh	600, 400	55.064	209.437

Tabuľka 7.2: Prehľad testovaných modelov neurónových sietí s detailmi o počte neurónov v jednotlivých vrstvách pre modely M13 až M20

7.1.3 Výsledky experimentu

Na základe zistení z experimentov 7.1.1 a 7.1.2 sa došlo k záveru, že ďalšie zvyšovanie počtu vrstiev alebo neurónov v rámci testovaných architektúr už neprináša signifikantné zlepšenia v predikčnej presnosti. Prístupy, ktoré boli aplikované, ukázali, že bol dosiahnutý bod, kde ďalšia komplexita modelu nevedie k lepším výsledkom.

V dôsledku týchto poznatkov sa je možné domnievať, že súčasné nastavenie neurónového modelu sú dostatočne optimalizované pre úlohy predikcie cien leteniek a nevyžadujú ďalšie modifikácie. Preto bol vybratý model **M14**, ktorý predstavuje najlepšie vyváženie medzi rýchlosťou trénovania a presnosťou predikcii.

Ďalej je dôležité sa zamerať na zabezpečenie, že nami zvolený model dosiahne najvyššiu možnú úroveň predikcie a efektivity, preto boli prevádzané ďalšie experimenty, ktoré sa budú zameriavať na optimálne nastavenie hyperparametrov, tie sú popísané v kapitole 7.2.

7.2 Experiment 2: Nastavenie hyper-parametrov

Tento experiment sa zameriava na optimalizáciu hyper-parametrov. Týmto experimentom je sledovaný vplyv rôznych kombinácií hyper-parametrov na presnosť a efektivitu modelu. Experiment začína s vytvorením základného modelu, na ktorom sú postupne menené rôzne hyper-parametre. V experimente bol testovaný vplyv rýchlosti učenia (learning rate), batch size a aktivačné funkcie, aby bolo zistené, ktoré nastavenie bude predikovať ceny leteniek čo najpresnejšie.

Na testovanie hyper-parametrov bol použitý model M14, ktorý bol identifikovaný ako najúčinnejší v predchádzajúcom experimente 7.1. Model bude mať nastavený EarlyStopping ako bol nastavený v 7.1 a limit 50 epoch.

Efektívnosť jednotlivých konfigurácií hyper-parametrov je hodnotená pomocou metriky strednej kvadratickej chyby (MSE). Výsledky sú podrobne dokumentované a zaznamenané v tabuľke 7.3.

Model ID	Optimalizator	Veľkosť kroku	Batch Size	MSE	Čas (s)	Graf
H1	adam	0.01	32	74.316	210.209	A.1
H2	adam	0.01	64	69.289	125.634	A.2
H3	adam	0.01	128	63.851	115.141	A.3
H4	adam	0.001	32	52.783	442.788	A.4
H5	adam	0.001	64	55.166	257.448	A.5
H6	adam	0.001	128	53.904	148.867	A.6
H7	adam	0.0001	32	83.864	452.604	A.7
H8	adam	0.0001	64	93.335	263.875	A.8
H9	adam	0.0001	128	103.144	163.231	A.9
H10	sgd	0.01	32	1003.578	85.609	A.10
H11	sgd	0.01	64	1003.759	59.331	A.11
H12	sgd	0.01	128	1003.313	48.981	A.12
H13	sgd	0.001	32	63.025	376.173	A.13
H14	sgd	0.001	64	56.261	237.258	A.14
H15	sgd	0.001	128	61.647	148.722	A.15
H16	sgd	0.0001	32	71.045	267.174	A.16
H17	sgd	0.0001	64	75.221	197.935	A.17
H18	sgd	0.0001	128	81.009	143.866	A.18

Tabuľka 7.3: MSE výsledky pre rôzne hyper-parametre

Na základe výsledkov experimentu je možné pozorovať, že model s najnižšou strednou kvadratickou chybou (MSE) a s prijateľným časom trénovania je model H6, ktorý bol používaný aj ako predvolený model v Experimente 2 7.1. Tento model má nasledujúce nastavenia:

- **Optimalizátor:** Adam
- **Rýchlosť učenia (learning rate):** 0.001
- **Batch Size:** 128

Tento model dosiahol MSE 53.904, čo predstavuje jedno z najpresnejších výsledkov z celého radu testovaných konfigurácií. Výsledok poukazuje na to, že nižšia rýchlosť učenia

v kombinácii s optimálnou veľkosťou dávky môže efektívne prispievať k minimalizácii chýb pri predikcii cien leteniek.

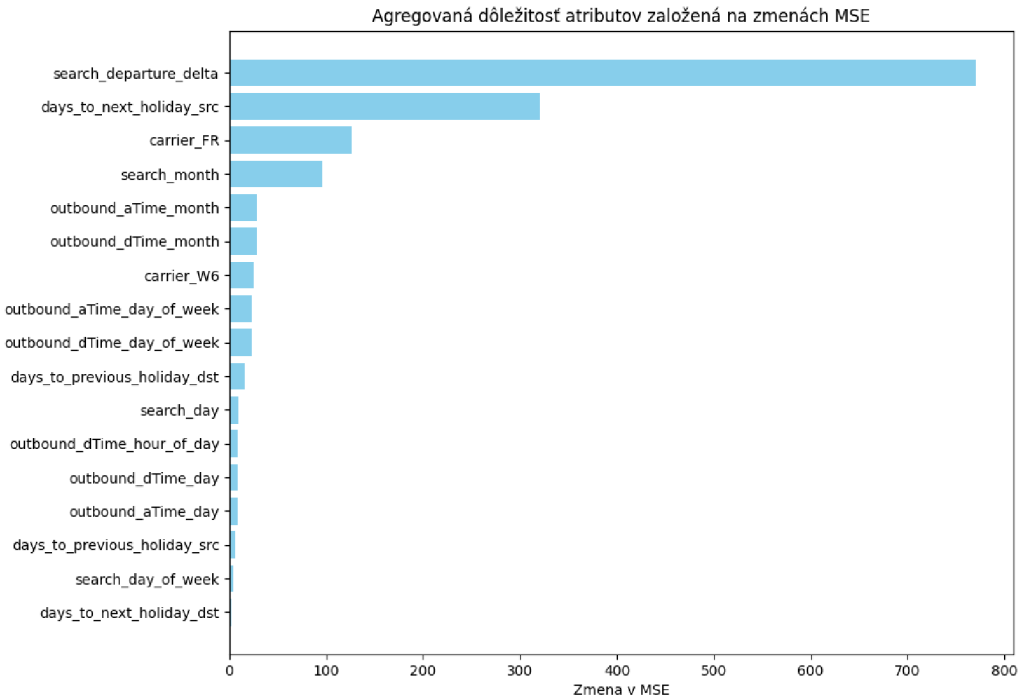
Výsledok, taktiež dokazuje, že optimalizácia hyper-parametrov môže mať významný vplyv na výkonnosť modelu. Pomalá rýchlosť učenia znižuje riziko pretrénovania modelu, zatiaľ čo optimálna veľkosť dávky zaisťuje, že model má dostatok, ale nie príliš veľa, informácií pre každý krok aktualizácie váh, čo v konečnom dôsledku vedie k stabilnejšiemu a presnejšiemu učeniu.

7.3 Experiment 3: Atribúty

Tento experiment sa zameriava na identifikáciu a analýzu atribútov, ktoré majú zásadný vplyv na predikciu cien leteniek. Cieľom bolo odhaliť, ktoré faktory prispievajú k najväčším variáciám v predikovaných cenách a je možné tieto informácie využiť na zlepšenie presnosti a efektivity predikčných modelov.

Na hodnotenie významnosti jednotlivých atribútov bol použitý model z Experimentu 2, ktorý vykázal najlepšie výsledky. Hlavný dôraz bol kladený na permutačnú dôležitosť [13], ktorá zisťuje, ako zmena (zamiešanie) hodnôt v jednotlivých atribútoch ovplyvňuje výkon modelu.

V experimente bola najprv určená základná stredná kvadratická chyba (MSE) modelu na testovacej sade. Následne boli pre každý atribút zamiešané jeho hodnoty v testovacej sade a vypočítaná nová MSE. Rozdiel v hodnotách MSE pred a po zamiešaní ukázal, aký významný je daný atribút pre predikčnú schopnosť modelu. Pri analýze výsledkov bolo potreba agregovať atribúty, ktoré boli zakódované pomocou one-hot encoding metódy. Agregácia bola vykonaná priemerom všetkých jednotlivých kategórií atribútu, aby bol zistený vplyv tohto atribútu na model. Celkové výsledky sa nachádzajú v grafe 7.4.



Obr. 7.4: Výsledný graf testu atribútov

Na základe vykonaného experimentu je možné vyvodiť niekoľko kľúčových záverov o atribútoch ovplyvňujúcich predikciu cien leteniek:

- **Vplyv časového horizontu rezervácie:** Atribút *search_departure_delta*, ktorý meria počet dní medzi vyhľadávaním a skutočným odletom, ukazuje, že má zásadný vplyv na predikciu ceny. To podporuje bežne odporúčanú stratégiu, že rezervácia leteniek s dostatočným predstihom pravdepodobne prinesie lepšie ceny.
- **Dni do nasledujúceho sviatku v krajine odletu:** Atribút *days_to_next_holiday_src* významne ovplyvňuje predikcie cien, čo naznačuje, že blížiace sa sviatky môžu byť spojené s vyšším dopytom a následne vyššími cenami leteniek z odletovej destinácie.
- **Sezónne trendy:** Atribút *search_month* sa ukazuje ako najvýznamnejší, čo naznačuje, že čas roka, kedy je letenka vyhľadávaná (teoreticky kupovaná), výrazne ovplyvňuje predpoveď ceny. Môže to byť spôsobené sezónnymi výkyvmi dopytu, ako sú sviatočné sezóny alebo typické dovolenkové obdobia.

Ďalšie pozorovania ukazujú, že atribúty s časom odletu a príletu (*outbound_dTime_day*, *outbound_aTime_day*, resp. *outbound_dTime_day_of_week* a *outbound_aTime_day_of_week*) majú menšiu, ale zreteľnú dôležitosť, čo by mohlo naznačovať, že dni odletov môžu mať mierny dosah na cenovú politiku. Zároveň je dôležité poznamenať, že aj keď sú na rovnakej trase dve nízkonákladové spoločnosti ako Ryanair a WizzAir, tak ich cenno tvorba sa môže líšiť.

7.4 Zhodnotenie experimentálnych výsledkov

V tejto kapitole sú predstavené experimentálne výsledky zamerané na evaluáciu rôznych architektúr neurónových sietí pre predikciu cien leteniek. Skúmané boli rôzne modely, zahrnujúce rozličný počet vrstiev a neurónov. Experimenty odhalili, ako konfigurácie hĺbky siete ovplyvňujú schopnosť modelov predpovedať ceny a zároveň ako rôzne nastavenia hyperparametrov a atribúty dát môžu významne ovplyvniť výkonnosť predikčných modelov. Hoci výsledky neukázali vysokú presnosť v predpovediach cien, poskytujú cenné informácie, ktoré môžu slúžiť ako doplnkový nástroj pri rozhodovaní o kúpe leteniek.

Kapitola 8

Záver

Hlavným cieľom tejto bakalárskej práce bolo vyvinúť webovú aplikáciu zameranú na prediktívne modelovanie dát o vyhľadávaní leteniek s cieľom maximalizovať užívateľský komfort pri kúpe leteniek.

V rámci práce boli navrhnuté a implementované viaceré analytické úlohy a modely, ktoré sú schopné analyzovať a predpovedať cenové trendy. Jednotlivé komponenty systému boli detailne popísané, vrátane ich funkcionalít a vzájomnej interakcie. Taktiež bol kladený veľký dôraz na používateľskú prívetivosť aplikácie, čo umožňuje jednoduché a intuitívne používanie aj pre používateľov bez technických znalostí.

Praktické testovanie aplikácie preukázalo, že prediktívne modely sú schopné efektívne využívať dostupné dáta a poskytovať užívateľom užitočné predikcie.

Na záver možno konštatovať, že táto bakalárska práca poskytla hodnotné poznatky a nástroje pre zlepšenie procesu rozhodovania o kúpe leteniek. Ďalší vývoj by sa mohol zamerať na rozšírenie dátových zdrojov, integráciu s inými informačnými systémami a zdokonalenie algoritmov pre ešte presnejšie predikcie. Tiež by bolo prínosné prieskum možností implementácie strojového učenia pre dynamickú adaptáciu modelov v reálnom čase na základe neustále sa meniacich trhových podmienok.

Táto práca tak otvára dvere k novým možnostiam v oblasti leteckých služieb a predstavuje pevný základ pre ďalší výskum a inovácie v oblasti prediktívneho modelovania a analýzy dát.

Literatúra

[1] *Root-mean-square deviation*. Wikimedia Foundation. Navštívené: 2024-05-01. Dostupné z: https://en.wikipedia.org/wiki/Root-mean-square_deviation.

[2] *Knowledge Discovery in Data-Mining: 5. Steps of the Knowledge Discovery in Databases Process*. 2024. Navštívené: 2024-02-08. Dostupné z: https://learn.saylor.org/mod/book/view.php?id=66656&chapterid=60129.

[3] *Perceptron*. Wikimedia Foundation, 2024. Navštívené: Máj 1, 2024. Dostupné z: https://cs.wikipedia.org/wiki/Perceptron#/media/Soubor:Perceptron.png.

[4] Aggarwal, C. C. *Neural Networks and Deep Learning: A Textbook*. Springer, 2018. Dostupné z: https://www.amazon.com/Neural-Networks-Deep-Learning-Textbook/dp/3319944622/.

[5] Builtin. Random Forest Algorithm. 2024. Navštívené: April 10, 2024. Dostupné z: https://builtin.com/data-science/random-forest-algorithm.

[6] Frost, J. *Mean Squared Error (MSE)*. 2024. Navštívené: 2024-05-01. Dostupné z: https://statisticsbyjim.com/regression/mean-squared-error-mse/.

[7] GeeksforGeeks. What is Linear Regression? 2024. Navštívené: April 10, 2024. Dostupné z: https://www.geeksforgeeks.org/ml-linear-regression/.

[8] Han, J., Kamber, M. a Pei, J. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2011. ISBN 0123814790. Dostupné z: https://www.amazon.com/Data-Mining-Concepts-Techniques-Management/dp/0123814790.

[9] Han, J., Kamber, M. a Pei, J. *Data Mining: Concepts and Techniques*. 3. vyd. Amsterdam: Morgan Kaufmann, 2012. ISBN 978-0-12-381479-1. Dostupné z: http://www.sciencedirect.com/science/book/9780123814791.

[10] Hotz, N. *What is CRISP DM?* 2024. Navštívené: 2024-02-08. Dostupné z: https://www.datascience-pm.com/crisp-dm-2/.

[11] JavaTpoint. Machine Learning - Polynomial Regression. n.d. Navštívené: April 10, 2024. Dostupné z: https://www.javatpoint.com/machine-learning-polynomial-regression.

[12] Kushwaha, A. Logistic Regression: Important Questions. 2020. Navštívené: April 10, 2024. Dostupné z: https://agam-kushwaha.medium.com/logistic-regression-important-questions-5c950e4c59ce.

[13] Molnar, C. *Feature Importance* [https://christophm.github.io/interpretable-ml-book/feature-importance.html#feature-importance]. Navštívené: 2024-05-04.

[14] Nielsen, F. *Data Mining with Python*. 2017. Navštívené: September 10, 2023. Dostupné z: https://www2.imm.dtu.dk/pubdb/edoc/imm6814.pdf.

[15] Programmer, L. K-Nearest Neighbors (KNN). n.d. Navštívené: April 10, 2024. Dostupné z: https://lazyprogrammer.me/mlcompendium/supervised/knn.html.

[16] Roberts, E. *A Brief History of Neural Nets* [https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history1.html]. 1996. Navštívené: Máj 1, 2024.

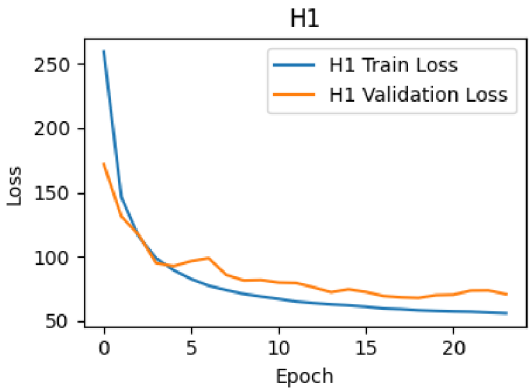
[17] Sharma, P. *Data Mining: The Knowledge Discovery of Data*. 2023. Navštívené: 2024-02-08. Dostupné z: https://www.analyticsvidhya.com/blog/2023/02/data-mining-the-knowledge-discovery-of-data/.

[18] Smith, J. D. a Doe, J. E. The Application of Network Science to Social Media. *Applied Network Science*. 2019, zv. 4, č. 88, s. 1–15. DOI: 10.1007/s41109-019-0248-7. Navštívené: April 10, 2024. Dostupné z: https://appliednetsci.springeropen.com/articles/10.1007/s41109-019-0248-7.

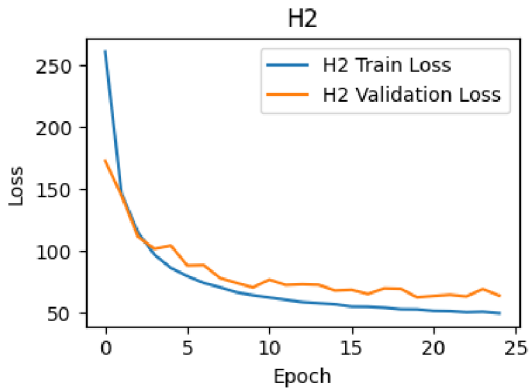
[19] Özer, A. A Brief History of Neural Networks. n.d. Navštívené: Máj 1, 2024. Dostupné z: https://www.kdnuggets.com/a-brief-history-of-the-neural-networks.

Príloha A

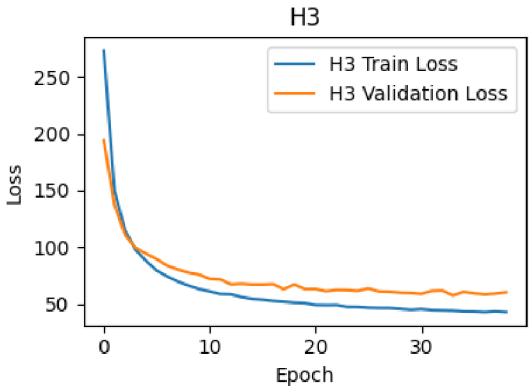
Detailné analýzy vplyvu hyper-parametrov



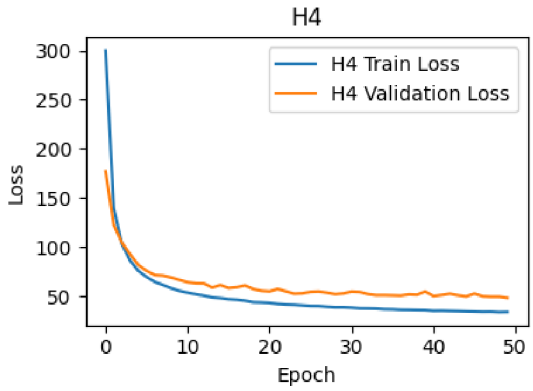
Obr. A.1: Porovnanie trénovacej a validačnej sady pre H1



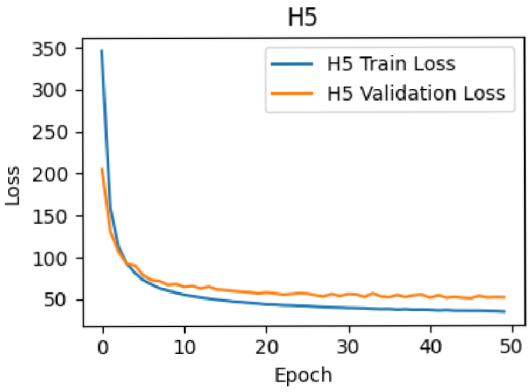
Obr. A.2: Porovnanie trénovacej a validačnej sady pre H2



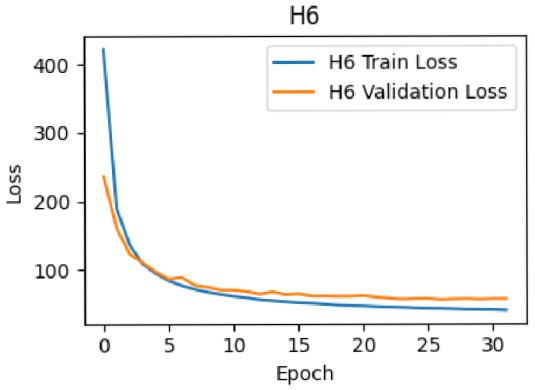
Obr. A.3: Porovnanie trénovacej a validačnej sady pre H3



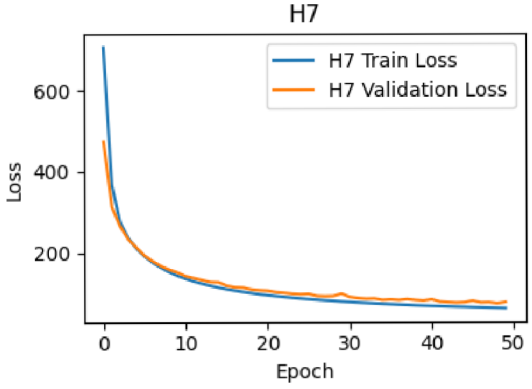
Obr. A.4: Porovnanie trénovacej a validačnej sady pre H4



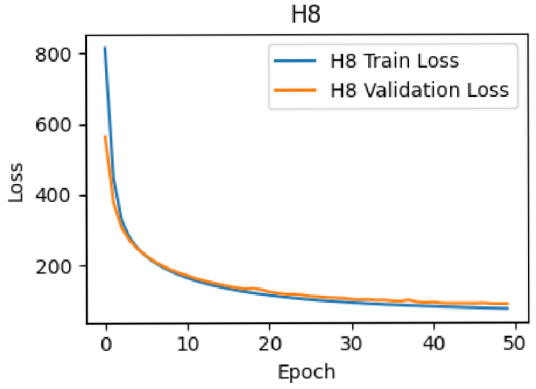
Obr. A.5: Porovnanie trénovacej a validačnej sady pre H5



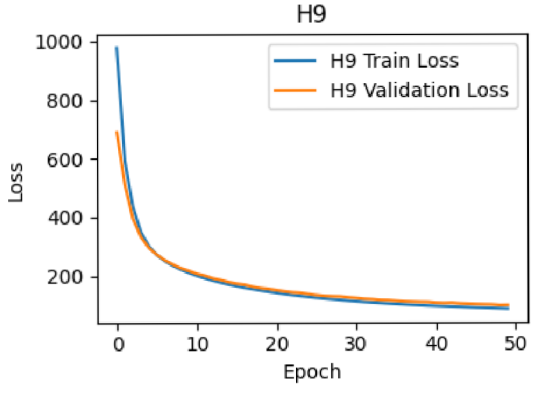
Obr. A.6: Porovnanie trénovacej a validačnej sady pre H6



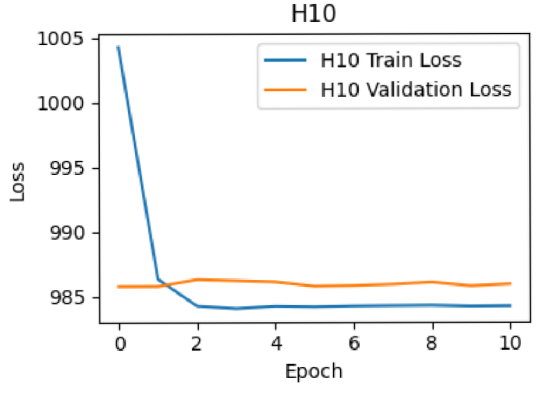
Obr. A.7: Porovnanie trénovacej a validačnej sady pre H7



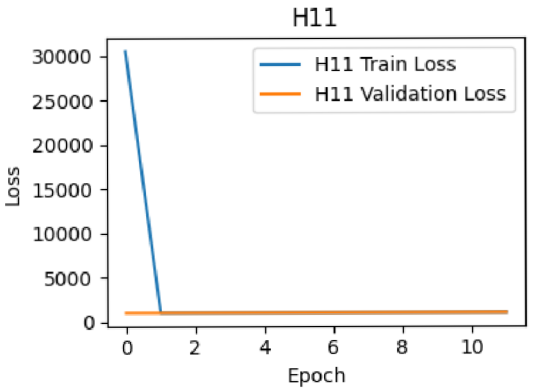
Obr. A.8: Porovnanie trénovacej a validačnej sady pre H8



Obr. A.9: Porovnanie trénovacej a validačnej sady pre H9

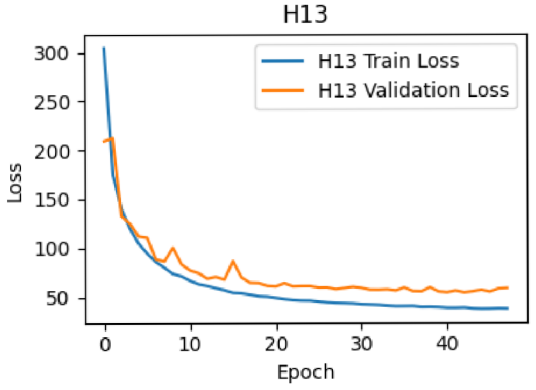


Obr. A.10: Porovnanie trénovacej a validačnej sady pre H10

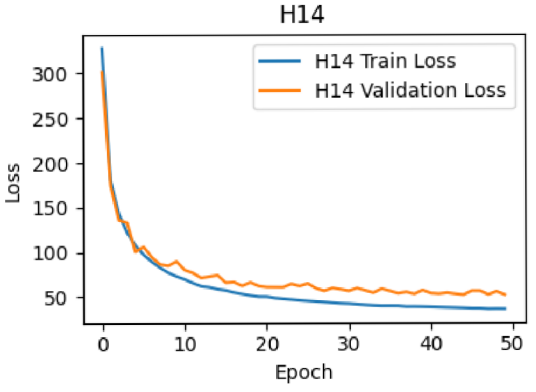


Obr. A.11: Porovnanie trénovacej a validačnej sady pre H11

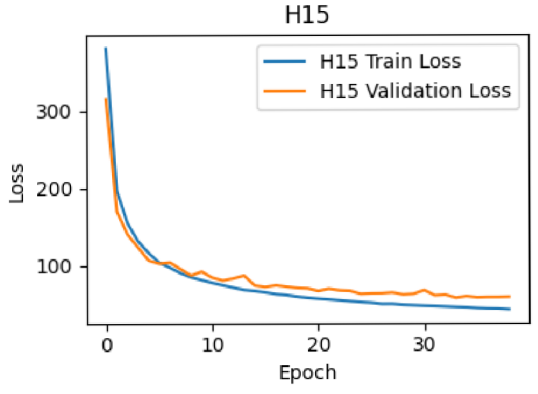
Obr. A.12: Porovnanie trénovacej a validačnej sady pre H12



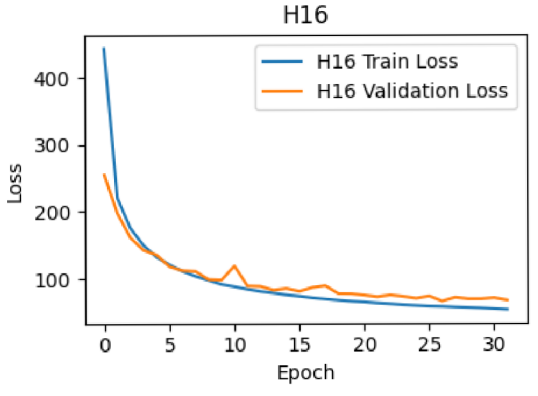
Obr. A.13: Porovnanie trénovacej a validačnej sady pre H13



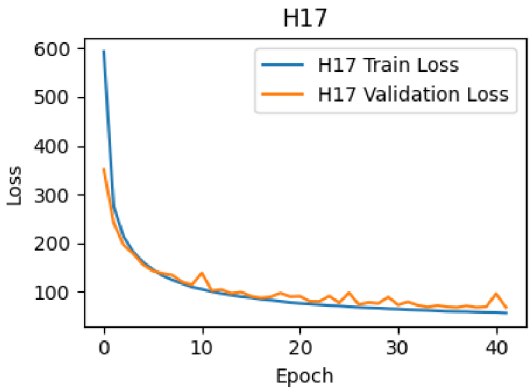
Obr. A.14: Porovnanie trénovacej a validačnej sady pre H14



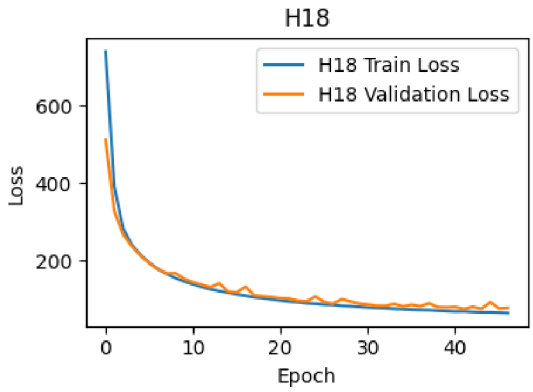
Obr. A.15: Porovnanie trénovacej a validačnej sady pre H15



Obr. A.16: Porovnanie trénovacej a validačnej sady pre H16



Obr. A.17: Porovnanie trénovacej a validačnej sady pre H17



Obr. A.18: Porovnanie trénovacej a validačnej sady pre H18

Príloha B

Manuál

Táto príloha obsahuje manuál k spusteniu webovej aplikácie tejto bakalárskej práce.

Pred spustením aplikácie je nevyhnutné mať nainštalovaný `Docker`, ktorý si môžete stiahnuť a nainštalovať podľa pokynov na oficiálnej stránke docker.com.

B.1 Príprava prostredia

Pred spustením aplikácie treba mať nainštalovaný `Docker CLI`. Ten si môžete nainštalovať podľa návodu na oficiálnej stránke docker.com.

B.2 Spustenie

Na spustenie aplikácie použite Docker Compose. V adresári, kde je uložený `docker-compose.yml` súbor, spustite nasledujúci príkaz:

```
docker compose up
```

Tento príkaz sťahuje potrebné Docker obrazy, vytvára kontajnery a spúšťa webovú aplikáciu podľa definície v `docker-compose.yml` súbore.

Po úspešnom spustení sa webová aplikácia bude nachádzať na URL adrese http://localhost:3000/, kde môžete vo svojom webovom prehliadači vidieť jej rozhranie.

Príloha C

Obsah priloženého pamäťového média

- Zdrojové súbory pre technickú správu v adresári **tex**
- Technická správa bakalárskej práce **xpodha01.pdf**
- Zdrojové kódy k backend a frontend časti sa nachádzajú v **flight_price_predictor**
- Zdrojové kódy pre modelovanie sa nachádzajú v **model**