Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Rozhodovací algoritmy na notebooku Jupyter

Bc. Michael Mandík

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Michael Mandík

Systémové inženýrství a informatika
Informatika

Název práce

Rozhodovací algoritmy na notebooku Jupyter

Název anglicky

Decision algorithms on Jupyter notebook

Cíle práce

Cílem práce je vypracovat metodiku učení a testování moderních metod strojového učení na notebooku Jupyter. Student tuto metodiku navrhne a ověří na úloze stanovení kreditního rizika.

Metodika

Student vysvětlí architekturu a fungování notebooků Jupyter a popíše jejich použití pro zpracování datových souborů. Dále podá přehled moderních rozhodovacích algoritmů založených na strojovém učení a navrhne obecný způsob jejich učení a testování na notebooku Jupyter. Navrženou metodiku ověří na souboru, který obsahuje data umožňující odhad kreditního rizika. Výsledný soubor Jupyteru (soubor .ipynb) bude obsahovat kromě programové realizace rozhodovacího algoritmu také analýzu datového souboru, výsledky získané v procesu učení a výsledky testování naučeného algoritmu. Výsledky, pro které to bude vhodné, budou prezentovány v grafické formě.

Doporučený rozsah práce

60 stránek

Klíčová slova

rozhodovací algoritmy, strojové učení, notebook Jupyter

Doporučené zdroje informací

Galea A.: Beginning Data Science with Python and Jupyter, Packt, 2018
SUMMERFIELD, M. *Python 3 : výukový kurz.* Brno: Computer Press, 2010. ISBN 978-80-251-2737-7.
WITTEN, I H. – FRANK, E. *Data mining : practical machine learning tools and techniques.* San Francisco, Calif.: Elsevier Science [distributor], 2005. ISBN 978-0-12-088407-0.

Předběžný termín obhajoby

2020/21 LS – PEF

Vedoucí práce

doc. Ing. Arnošt Veselý, CSc.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 7. 3. 2021

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 7. 3. 2021

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 19. 03. 2021

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Rozhodovací algoritmy na notebooku Jupyter" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31.3.2021 _____

Poděkování

Rád bych touto cestou poděkoval doc. Ing. Arnošt Veselý, CSc. za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování diplomové práce.

Rozhodovací algoritmy na notebooku Jupyter

Abstrakt

Cílem diplomové práce je předvést vybrané rozhodovací algoritmy v platformě Jupyter Notebook. Jako úloha pro představení vybraných algoritmů bylo vybráno stanovení kreditního rizika v bankovnictví. V první části práce budou představena populární data science platforma Jupyter Notebook. Vybrány budou také rozhodovací algoritmy od klasických statistických až po algoritmy založené na strojovém učení. Praktická část práce nejdříve zanalyzuje vybraná data a následně budou vytvořeny modely pro jednotlivé rozhodovací algoritmy. Na závěr práce bude přesnost jednotlivých modelů porovnána.

Klíčová slova: rozhodovací algoritmy, strojové učení, Jupyter Notebook, kreditní riziko

Decision algorithms on Jupyter notebook

Abstract

This thesis aims to demonstrate selected decision-making algorithms on the Jupyter Notebook platform. The task chosen for demonstrating selected algorithms is the assessment of the banking credit risk for protentional customers. The first part of the thesis will describe the popular data science platform Jupyter Notebook. Then we will choose several decisions making algorithms from classical statistical algorithms to more complex machine learning algorithms. The practical part of the thesis will firstly analyze chosen data followed by modeling of each chosen algorithm. The last part of this thesis will compare the prediction accuracy of chosen algorithms.

Key words: decision-making algorithms, machine learning, Jupyter Notebook, credit risk.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Diplomová práce se zabývá rozhodovacími algoritmy v kreditním riziku, což je jedno z nejstarších rizik v bankovnictví. Kreditní riziko spočívá v pravděpodobnosti ztráty způsobené neschopností klienta splatit svůj závazek. Tato ztráta pro banku znamená, že nedostane dlužnou částku a úrok, což může zapříčinit narušení cash flow a způsobit budoucí náklady na vymáhání. Banky a jiné finanční instituce se proti tomuto riziku chrání analýzou úvěruschopnosti klienta, kdy se snaží odhadnout, jak moc bude klient spolehlivý ve splacení poskytnutého úvěru.

Klasická metoda odhadu úvěruschopnosti klienta je pomocí úvěrových schvalovatelů, kteří na základě zkušeností, intuice a předem stanovených interních směrnic rozhodnou, zda úvěr schválí, nebo ne.

S vývojem moderních technologií a statistických algoritmů jsou nyní i přesnější a efektivnější způsoby k predikci dobrých a špatných klientů. Tyto nové metody a algoritmy dokážou automaticky rozhodnout o úvěruschopnosti klienta na základě poskytnutých dat.

V této práci se budeme zabývat statistickými algoritmy pro rozhodování v kreditním riziku. Kromě základních algoritmů pro modelování v kreditním riziku budeme testovat a modelovat i pokročilejší algoritmy založené na strojovém učení.

Vyhodnocování kreditního rizika pomocí vybraných rozhodovacích algoritmů bude provedeno v populární data science platformě Jupyter Notebook.

2 Cíl práce a metodika

Cílem práce je vypracovat metodiku učení a testování moderních metod strojového učení na notebooku Jupyter. Navržená a vypracovaná metodika bude ověřená na úloze stanovení kreditního rizika.

V části teoretických východisek práce bude vysvětlena architektura a fungování notebooků Jupyter a bude popsáno jejich použití pro zpracování datových souborů. Dále bude formou literárních rešerší podán přehled moderních rozhodovacích algoritmů založených na strojovém učení a navržen bude obecný způsob jejich učení a testování na notebooku Jupyter.

V praktické části práce bude navržená metodika ověřena na souboru, který obsahuje data umožňující odhad kreditního rizika. Výsledná analýza bude obsahovat kromě programové realizace rozhodovacího algoritmu také analýzu datového souboru, výsledky získané v procesu učení a výsledky testování naučeného algoritmu. V diskusi práce budou porovnané výsledky použitých algoritmů.

3 Teoretická východiska

3.1 Jupyter Notebook

Projekt Jupyter (anglicky Project Jupyter) je mezinárodní nezisková organizace vytvořená za účelem vývoje open-source softwaru, open-standardů a služeb pro interaktivní zpracování napříč desítkami programovacích jazyků, přičemž název projektu je odvozený ze tří základních jazyků podporovaných Jupyterem: Julia, Python a jazyk R. Mezi nejčastěji používané produkty tohoto projektu patří Jupyter Nootebook, JupyterHub a JupyterLab (Shirokov, 2015) (Project Jupyter, 2021).

V roce 2014 představil zakladatel projektu IPython (Interactive Python) Fernando Pérez pokračování tohoto interaktivního shellu. IPython pokračuje jako Python interpret příkazů a kernel, zatímco části jako notebook se přesunuly pod název Jupyter.

Filozofie tohoto projektu je zaměřena na podporu interaktivní práce s daty a vědeckého zpracování dat napříč všemi programovacími jazyky (Project Jupyter, 2021).

3.1.1 Architektura a principy fungování Jupyter notebooku

Jupyter Notebook, dříve IPython Notebooks, je interaktivní webové prostředí pro vytváření Jupyter notebook dokumentů.

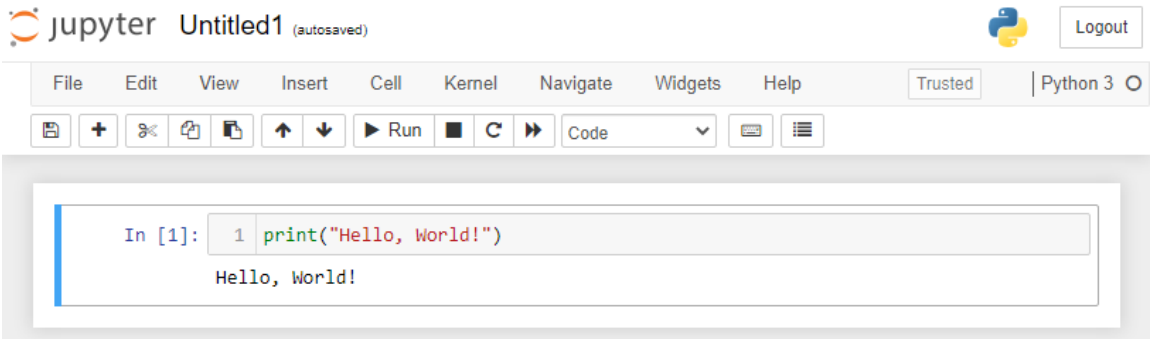
Jupyter Notebooky jsou uložené jako soubory typu IPYNB a běží lokálně jako webová aplikace, která obsahuje programovací kód, různé výpočty, rovnice, data, interaktivní aplikace a tzv. Markdown text.

Standardním jazykem a jazykem použitým v této práci je jazyk Python. Nicméně Jupyter podporuje širokou škálu i jiných jazyků jako například populární matematicky jazyk R či dokonce dotazovací jazyky jako je SQL (Jupyter Team, 2015) (Galea, 2018).

Základem Jupyter Notebooku jsou buňky (cell), které lze formátovat dvěma základními způsoby: Code a Markdown.

První formát **Code** funguje jako klasický interpreter zvoleného jazyka. V input části buňky může být libovolný kód vybraného programovacího jazyka a po spuštění buňky pomocí tlačítka Run nebo klávesové zkratky se zobrazí výstup (output).

Obrázek 1 Příklad code buňky v Jupyteru

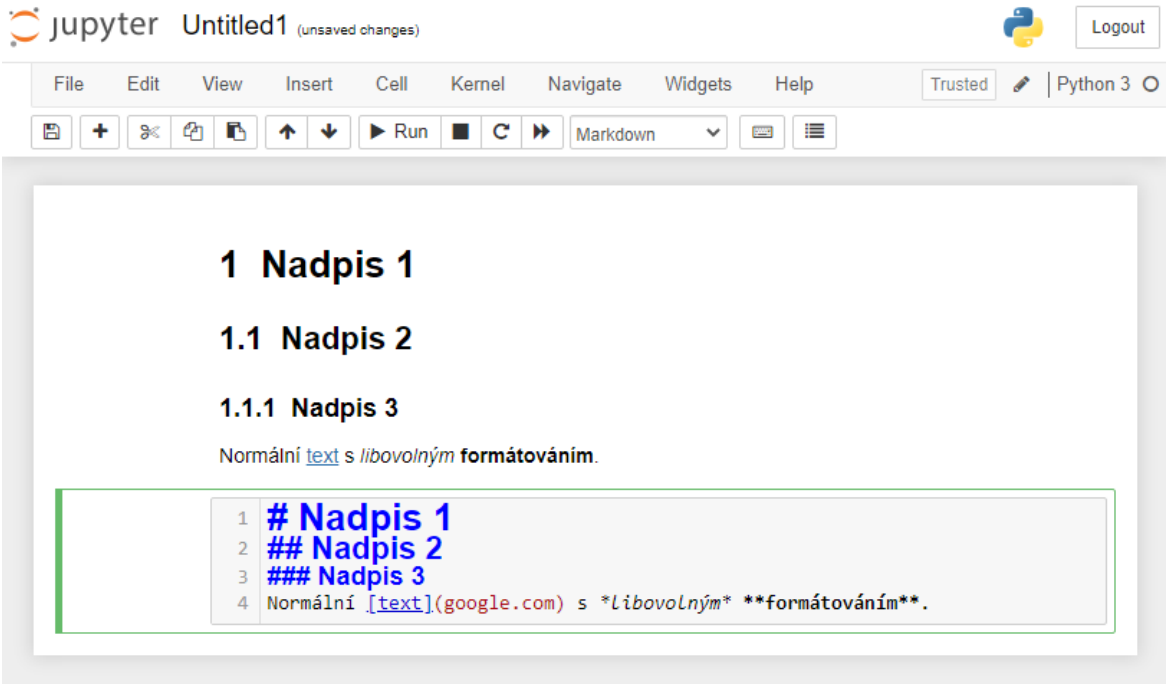


Zdroj: Vlastní zpracování

Markdown buňky slouží jako jednoduchý způsob komentování a dokumentace použitého kódu. Markdown umožňuje jednoduché formátování textu. Způsob formátování se podobá HTML jazyku, ale poskytuje mnohem méně způsobů přizpůsobení.

Základní symboly pro formátování textu v buňkách Markdown jsou mřížky „#“ (hash) pro vytvoření nadpisů různých úrovní, hranaté a oblé závorky pro vložení hypertextového odkazu a hvězdička „*“ pro formátování textu tučně nebo kurzívou (Galea, 2018).

Obrázek 2 Příklad markdown buňky v Jupyteru.



Zdroj: Vlastní zpracování

3.1.2 Motivace použití Jupyter notebooku

Jupyter Notebooky se postupně stávají jedním z nejdůležitějších nástrojů pro vědeckou práci s daty, a to nejen v programovacím jazyku Python. Notebook Jupyter tvoří ideální prostředí pro vývoj přesně opakovatelných postupů datové analýzy. Data mohou být nahrána, transformována a modelována najednou v jednom souvislém notebooku. Toto umožňuje rychlé a jednoduché testování kódu a aplikaci případných nápadů. Dále mohou být všechny zmíněné kroky dokumentovány přímo v notebooku pomocí formátovaného textu v tzv. markdown buňkách (Galea, 2018).

Možnost vidět veškeré ústřižky kódu s jejich outputy v jednom souvislém okně a být schopen se ihned podívat zpět na předchozí části analýzy pomáhá k efektivnějšímu vývoji modelů. Veškeré tyto výhody pomohly projektu Jupyter stát se nejpopulárnějším nástrojem při vývoji vědeckých algoritmů či datových analýzách v programovacím jazyku Python (Galea, 2018).

3.2 Kreditní riziko v bankovnictví

V průběhu své činnosti se banky nevyhnutelně setkávají s různými druhy rizik, která mohou mít potenciální negativní efekt na jejich provoz a obchod. Banky jsou nuceny vytvořit komplexní a spolehlivý systém risk managmentu, který dokáže řídit riziko ve všech obchodních i neobchodních činnostech banky (Bank for International Settlements, 2000).

Vůbec nejstarším typem rizika bankovního podnikání a zároveň i nejvýznamnějším rizikem je kreditní nebo také úvěrové riziko. Kreditní riziko je riziko defaultu (neschopnosti klienta splácet) na dluhu, které může vzniknout, pokud nebude dlužník schopen splácet svůj dluh. Neschopnost splatit dluh může způsobit ztrátu celého, nebo části poskytnutého úvěru, úroků, narušení cash flow banky a může potenciálně vytvořit další náklady na vymáhání (Labarre, 2021).

Pro snížení tohoto rizika banky provádí analýzy tzv. uvěrové důvěryhodnosti klientů. Klienti banky jsou na základě této analýzy ohodnoceni pomocí tzv. skóre, na jehož základě jim je nabídnuta výše úvěru a cena (úrok). Čím nižší skóre klient má, tím horší podmínky úvěru dostane a naopak. Tato individuální nabídka produktů na základě úvěrové důvěryhodnosti klientů slouží jako jistá pojistka pro banky a pomáhá jim v rozhodovacím procesu (Labarre, 2021) (Bank of America, 2020).

Pro zajištění správného zjištění úvěrové důvěryhodnosti klientů, zejména fyzický osob, potřebuje banka analyzovat velká množství dat a provádět tyto analýzy a profilování rychle a automaticky (Bank for International Settlements, 2000).

3.2.1 Modely kreditního skórování

Průkopníkem modelů kreditního rizika byl americký businessman Henry Wells, spoluzakladatel jedné z největších amerických bank Wells Fargo & Company. Spousta zkušených analytiků byla během druhé světové války odvedena a Wells se setkal s nedostatkem kvalifikovaných pracovníků pro vyhodnocování kreditního rizika. Řešením bylo vyvinutí nástrojů, které by pomohly nezkušeným analytikům provádět kreditní vyhodnocování (Lewis, 1994).

V americkém bankovnictví došlo během padesátých let dvacátého století k velkému rozšíření skórovacích modelů. První modely byly založeny na předem stanovených vahách pro určité charakteristiky klienta. Výsledkem byl součet těchto vah k dosazení klasifikačního skóre. K ještě většímu rozšíření došlo v šedesátých letech, kdy kreditní modely transformovaly americký trh, a to nejen u finančních institucí (Credit Risk Analysis Applying Logistic Regression, Neural Networks and Genetic, 2007).

3.2.2 Identifikace dobrých a špatných klientů

Vstupní sada klientů vstupujících do modelu skórování se skládá z klientů s informacemi během určitého období, o kterých lze rozhodnout –„dobrý”, nebo „špatný" (anglicky Good/Bad). Definice dobrých a špatných klientů má tendenci se lišit v různých modelech a je velmi důležité, vzít tuto definici v úvahu a důkladně ji přezkoumat při konstrukci modelů (Vitali, 2018) (Analyst Prep, 2019).

Příliš přísná definice špatného klienta vede k příliš velkému počtu klientů zahrnutých na hranici, což snižuje efektivitu skórovacího modelu a znamená to, že špatně využíváme poskytnuté nástroje a data. Zobecněním celé skupiny klientů vyřazujeme „špatné” klienty příliš brzy. Na druhou stranu nedostatečně přísná klasifikace znamená zařazení hraničních klientů do kategorie dobrých (dobré skoré), což nese riziko uzavírání obchodů s těmito potenciálně špatnými klienty (Vitali, 2018).

Můžeme samozřejmě kombinovat definice hranice dobrých a špatných klientů. Můžeme použít přísnější definici pro vývoj funkcí a méně přísné hranice pro následující úvahy. V každém případě je důležité řádně prozkoumat použité hranice, zejména z úhlu případných možností. Běžně používaná hranice k určení dobrého a špatného klienta je neschopnost platit bankovní splátky v intervalu třiceti až devadesáti dnů (Vitali, 2018).

3.2.3 **Vyhodnocování kreditního rizika**

Špatně vyhodnocené riziko zcela jistě způsobí bance ztrátu. Ať už kvůli přijetí klientů, kteří způsobí bance ztrátu, nebo zamítnutí dobrých klientů, kteří by naopak vytvořili pro banku zisk. Banky, které mají lepší způsoby vyhodnocování rizika než jejich konkurence, mají nad ostatními výhodu, jelikož jsou méně zranitelné vůči potenciálním následkům způsobeným špatným rozhodnutím při poskytování služeb.

Vyhodnocování kreditního rizika u potenciálních klientů se provádí dvěma způsoby:

- Posouzení – subjektivní analýza schvalovatele založená spíše na kvalitativních znacích klienta.
- Statisticky – klasifikací pomocí vyhodnocovacích modelů, založených spíše na kvantitativní analýze klienta.

V současné době všechny instituce pracující s kreditním rizikem klientů (banky, pojišťovny nebo nebankovní instituce) používají pro vyhodnocování klientů kombinaci obou způsobů vyhodnocování rizika (Credit Risk Analysis Applying Logistic Regression, Neural Networks and Genetic, 2007).

Porovnání jednotlivých aspektů subjektivního a statistického vyhodnocování kreditního rizika klientů lze podle Schreinera shrnout následovně:

Tabulka 1 Porovnání subjektivního a statistického vyhodnocování

Aspekt	Subjektivní vyhodnocování	Statistické vyhodnocování
Zdroj znalostí	Zkušenosti schvalovatele a instituce	Kvantifikované portfolio v databázi
Konzistence procesu	Odlišná v závislosti na schvalovateli	Identická pro každý studovaný případ
Přísnost procesu	V závislosti na směrnicích a intuici schvalovatele	Stanovena matematická pravidla
Proces a produkt	Kvalitativní klasifikace, jelikož schvalovatel pozná jednotlivé klienty zvlášť	Kvantitativní pravděpodobnost, jelikož skórkarty využívají kvantitativní charakteristiky rizika
Proces implementace	Dlouhý trénink a kontrolování schvalovatelů	Dlouhá příprava modelů
Možnosti zneužití	Osobní názory a předsudky, denní nálady nebo běžné lidské chyby	Špatně použitá data
Flexibilita	Široké použití, díky možnosti úprav procesu manažery	Jedno použití. K zahrnutí nového aspektu rizika je potřeba vytvořit nové skórkarty

Zdroj: (Schreiner, 2003)

3.2.4 Default rate

Podepsáním smlouvy o úvěru, ať už se jedná o hypotéky, osobní úvěr nebo kreditní kartu, se klient zavazuje k pravidelným měsíčním splátkám. V případě nezaplacení splátky dojde k začátku delikvence klienta. Z delikvence klienta se odvádí základní metrika pro sledování kvality portfolia banky z hlediska kreditního rizika default rate (česky míra selhání) (Hamilton, Cantor, 2006).

Default rate je definovaná jako procento úvěrů poskytnutých finanční institucí, které nebyly v delším časovém horizontu splaceny. O takových úvěrech se říká, že jsou delikventní a při jejich sledování a analýze je potřeba vzít v potaz hned několik hledisek.

Časový horizont

Perioda definovaná pro neschopnost splácet. Běžně se v praxi používají měsíční periody. Klient může být 30, 60 nebo 90 dní delikventní. Úvěry s 180+ denní delikvencí se většinou postupují dál (Open Risk, 2006).

Pokud úvěr selže, instituce může tuto pohledávku vymáhat na vlastni náklady nebo odepsat jako nevymahatelnou. Pohledávku může také instituce postoupit. Postoupení pohledávky je upraveno v ustanoveních § 1879 a. n. občanského zákoníku a týká se změny v osobě věřitele, kdy věřitel (postupitel) postupuje celou pohledávku nebo její část smlouvou třetí osobě (postupníkovi), což může učinit i bez souhlasu dlužníka.

Objem nebo počet

Poměr selhání úvěrů lze vypočítat z hlediska počtu nebo objemu. Počtem se může rozumět počet jednotlivých úvěrů ve sledovaném období nebo počet klientů. U objemu se sleduje výše úvěru nebo expozice, kterou banka riskuje v případě defaultu (Open Risk, 2006).

Současná a posunutá delikvence

Současná delikvence (angl. coincidental delinquency) poskytuje informaci o delikvenci portfolia ve vybraném okamžiku, nejčastěji ke konci měsíce. Tento způsob měření je ovlivněn růstem velikosti portfolia. Úvěry z různých delikvencí vznikly v různých časových obdobích. V případě rychlého růstu portfolia dojde k „naředění“ delikvencí, jelikož se delikvence sledují v měsíčních rozmezích (30, 60 a 90 dnů) (ICRA, 2016).

Posunutá delikvence (angl. lagged delinquency) tento problém řeší díky sledování pouze úvěrů vzniklých ve stejném období. Například delikvence 90 dní se sleduje pouze u úvěrů, které jsou alespoň 90 dní staré (ICRA, 2016) (Hamilton, Cantor, 2006).

Vzorec default rate pro celkový počet případů N_t a počet defaultních případů N_t^D v období t je následující:

$$DR_t = \frac{N_t^D}{N_t} \tag{1}$$

3.2.5 Rozhodování v úvěrovém riziku

Spousta lidí se v dnešní době potýká s problémem získat úvěr z důvěryhodných a regulovaných zdrojů, jako jsou banky. Tento problém je zapříčiněn nedostatečnou kreditní historií klienta. Může se jednat o studenty, ale i nezaměstnané jedince, kteří nemusí mít dostatečné znalosti, aby dokázali odhadnout spolehlivého poskytovatele úvěru. Někteří

nedůvěryhodní poskytovatelé úvěru tohoto zneužívají a poskytují úvěry s příliš vysokými úroky nebo se skrytými podmínkami, a často se proto mohou lidé dostat do finančních problémů.

Místo hodnocení klienta pouze na základě kreditní historie lze ohodnotit klienta a stanovit pravděpodobnost správné platební morálky i jinými způsoby. Kupříkladu zaměstnání klienta má z hlediska schopnosti splácet velkou vypovídací hodnotu, protože zaměstnaní lidé mají stabilní příjem, a jsou tedy schopni splácet. Další faktory mohou být například vlastněné nemovitosti, rodinný stav, místo bydliště, mohou také být užitečné při studování klientovy schopnosti splácet.

Moderní technologie a algoritmy umožňují použití velkého množství faktorů, které mohou pomoci při výběru klientů zcela, nebo alespoň částečně automaticky. Takovémuto automatickému vyhodnocování bonity klientů na základě předem vybraných faktorů se říká kreditní skórování (CGAP/World Bank, 2019).

3.2.6 Kreditní skórování

Kreditní skórování může bankám a jiným finančním institucím pomoci zvětšit portfolio schopností rychle zanalyzovat i klienty s nižšími příjmy, ale i celkovým zvýšením kvality poskytovaných služeb a spokojenosti zákazníků.

Model kreditního skórování je nástroj risk managementu, který vyhodnotí úvěruschopnost žadatele o úvěr odhadnutím pravděpodobnosti defaultu na základě historických dat.

Tyto modely jsou užitečné zejména v případech, kdy musí instituce zanalyzovat velké množství žádostí o malé úvěry, ale obecně i pro běžné úvěry a úvěry pro malé a střední podniky.

I přestože nejčastější využití kreditního skórování je zjištění úvěruschopnosti klienta při podání žádosti, finanční instituce využívají tento nástroj i při rozhodování v dalších fázích životního cyklu zákazníka. Každá fáze obsahuje různé druhy aplikace kreditního skórování. Může se jednat už o skórování potenciálních klientů, kdy instituce zjistí, kdo ji vůbec může oslovit, nebo ke zjištění, koho chce instituce vůbec z hlediska profitability oslovit. Dále může instituce skórovat již existující klienty na základě jejich chování (platební morálka, transakce atd.) a případně může klientovi nabídnout další produkty, upravené na míru (CGAP/World Bank, 2019).

Kreditní skórování žádosti se soustředí na vybrání klientů ke schválení z velkého množství podaných žádostí. Použití automatického systému na vyhodnocení kreditního skórování žádostí má mnoho výhod, jež lze vymezit následovně:

Operační efektivita

Zavedení automatizovaného systému může znamenat přínosy na operativní úrovni. Toho lze dosáhnout pomocí snížení finančních a časových nákladů na manuální vyhodnocování rizika žádosti, snížení počtu návštěv zákazníka na pobočce osobně, snížení administrativních nákladů.

Zvýšení přesnosti při rozhodování

Cíleným půjčováním, založeným na pravděpodobnosti defaultu se minimalizuje odmítnutí úvěruschopných zákazníků a zároveň se maximalizuje odmítnutí zákazníků s horší spolehlivostí.

Vytvoření objektivního a standardizovaného systému rozhodování založeného na datech.

Objektivní systém podložený daty minimalizuje možnost chyby způsobené člověkem.

Spokojenost zákazníků

Rychlejší a zároveň cílené úvěry mají větší pravděpodobnost přejití žádosti k podepsání smlouvy (CGAP/World Bank, 2019).

Zavedení kreditního skóringu může být pro instituci velká výzva. Implementace nového systému může být komplikovaná a bude požadovat vysokou investici. V závislosti na nastavení systému bude požadovat nové technické pozice, které nemusí být zatím v instituci dostupné. Protože modely používají historická data, je potřeba si v neposlední řadě uvědomit, že budou předpokládat, že se trh bude v budoucnu chovat stejně jako v minulosti.

3.2.7 Metody klasifikace klientů

Existuje mnoho systémů podporujících rozhodování, založených na modelech kreditního skórování. Tyto systémy pomáhají při klasifikaci klientů banky a pomáhají bankám při rozhodnutí, komu poskytnout úvěr.

Literatura vymezuje dvě skupiny běžně používaných prediktivních modelů pro kreditní skórování, a to metody statistické a metody založené na umělé inteligenci nebo strojovém učení (Anahita et al., 2018).

Mezi statistické metody se řadí lineární diskriminační analýza (LDA) a logistická regrese (Anahita et al., 2018; Baesens et al., 2003).

Metody obsahující prvky umělé inteligence a strojového učení obsahují komplexnější algoritmy, mezi něž patří decison trees, random forest, gradient boosting, support vector machine, neural networks (Baesens et al., 2003) (Anahita et al., 2018).

Navzdory velkému pokroku v oblasti umělé inteligence a strojového učení se stále v oblasti vyhodnocování kreditního rizika používají spíše jednodušší statistické algoritmy, jako je logistická regrese, a to díky její jednoduché implementaci a vysoké přesnosti (Anahita et al., 2018).

3.2.8 Problematika automatického rozhodování a umělé inteligence v kreditním riziku

Automatické rozhodování a profilování se využívá ve stále rozsáhlejším počtu odvětví, a to jak v soukromých, tak i veřejných sférách. Pojišťovnictví, bankovnictví, finance, zdravotní péče, marketing a reklama jsou jenom některé z mnoha oblastí, které profilování provádí pravidelně za účelem rozhodování (ÚOOÚ, 2017).

Moderní technologie, pokrok ve schopnosti analyzovat velké množství dat nebo také umělá inteligence a strojové učení usnadňují profilování a provádění automatického rozhodování, přitom mohou velmi ovlivnit práva a svobody jednotlivců. Široká přístupnost osobních údajů na internetu a ze zařízení internetu věcí a možnost nalézt vzájemné souvislosti a vytvářet mezi nimi vazby může vést k určení, analýze a odhadu hledisek týkajících se osobnosti, chování, zájmů a zvyklostí jednotlivců (ÚOOÚ, 2017).

Profilování a automatické rozhodování může být pro jednotlivce a organizace užitečné, neboť vede k vyšší efektivnosti a úsporám nákladů. Využití pro komerční účely je mnoho. Může být využito například k lepší segmentaci trhů a přizpůsobení produktů a služeb, aby byly v souladu s potřebami jednotlivce. Tyto procesy jsou užitečné i pro zdravotnictví, vzdělávání a dopravu. Automatické rozhodování a profilace však můžou znamenat vysoké riziko pro práva a svobody jednotlivých osob, což vyžaduje vhodné záruky (ÚOOÚ, 2017).

Procesy profilování a automatického rozhodování mohou být nejasné. Jednotlivci si nemusí být vědomi toho, že se vytváří jejich profil. Někdy nemusí vůbec chápat, o co jde. Profilování může negativně přispívat k zachování již existujících stereotypů a sociální segregace. Může rovněž uzavřít jednotlivce do jisté specifické skupiny a omezit ji na určité

doporučené preference. Toto může narušit například svobodu jednotlivce ve výběru určitých produktů nebo služeb. V dalších případech může profilování vést k nepřesným odhadům. V jiných případech může profilování vést k případnému neposkytnutí služeb a zboží a k neodůvodněné diskriminaci (ÚOOÚ, 2017).

Ochranou jedince z hlediska automatického rozhodování a profilování se mimo jiné zabývá Obecné nařízení o ochraně osobních údajů, zkráceně ONOOÚ (anglicky GDPR, General Data Protection Regulation)

Nařízení ONOOÚ zavádí ustanovení, jež zajistí, že profilování a automatizované individuální rozhodování (ať už zahrnuje profilování, či nikoli) nebude použito způsobem, který má neodůvodněný dopad na práva jednotlivců. V případě vyhodnocování kreditního rizika se může jednat například o specifické požadavky na transparentnost a spravedlnost (ÚOOÚ, 2017).

Právě transparentnost a spravedlivost jsou témata, kterých se modernější algoritmy založené na umělé inteligenci mohou dotknout. Většina současných algoritmů založených na strojovém učení nebo umělé inteligenci jsou neprůhledné. Pokud je umělá inteligence černá skřínka, udělá tedy rozhodnutí a predikce podobné jako člověk, ale bez schopnosti vysvětlit, proč se tak rozhodla. Rozhodovací proces umělé inteligence může být založen na vyhledávání vzorů chování, kterým člověk nedokáže porozumět (Bathaee, 2018).

Podle Evropské bankovní federace přináší umělá inteligence příležitosti ke zvýšení prosperity a růstu. Pro bankovní sektor poskytuje umělá inteligence příležitost ke zlepšení spokojenosti zákazníků, demokratizaci finančních služeb, zvýšení informační bezpečnosti včetně bezpečnosti informací svých klientů, ale také posílení risk managementu (European Banking Federation, 2019).

Nicméně k využití výše zmíněných příležitostí je podle Evropské bankoví federace důležité zajistit transparentnost a vysvětlitelnost. Kvůli statistické povaze této technologie jsou právě tyto dva aspekty nejdůležitější pro zachování důvěry v umělou inteligenci (European Banking Federation, 2019).

Přínosy moderních algoritmů založených na umělé inteligenci v kreditním skórování jsou nepochybné. Problém ale nastává právě ve výše zmíněné vysvětlitelnosti a transparentnosti. Instituce musí být schopna říct jednotlivci, na základě, jakého rozhodnutí byla jeho žádost odmítnuta. Před zaváděním umělé inteligence, v kreditním rozhodování, je tedy třeba zajistit, aby byla v souladu s regulárním prostředí; daného státu (European Banking Federation, 2019) (ÚOOÚ, 2017).

3.3 Strojové učení

Strojové učení (angl. machine learning) je oblast matematické informatiky, která studuje výpočetní algoritmy, jež se automaticky zlepšují a učí získáváním zkušeností na základě využití poskytnutých dat. Jedná se o podoblast umělé inteligence. Algoritmy strojového učení vytvoří model podle poskytnutých vzorových dat, kterým se říká „trénovací data“. Na těchto datech se model naučí vytvářet odhady, nebo rozhodnutí, aniž by se to programu explicitně řeklo. Strojové učení má široké možnosti využití, ať už se jedná o filtrování e-mailu, nebo schopnost počítače rozpoznávat obrázky. Využití je populární zejména v oblastech, kde by vytvoření konvekčního programu pro danou potřebu bylo příliš náročné (Mitchell, 1992).

Účelem strojového učení je, aby byl počítač schopen sám přijít na to, jak splnit svůj úkol, aniž by k tomu byl přímo naprogramován. Toho počítač dosáhne učením z poskytnutých dat. Pro jednodušší úkoly je možné naprogramovat algoritmy, které řeknou přístroji, jak přesně spustit jednotlivé kroky programu, aby dokázal splnit zadaný úkol. Není tedy na straně počítače nutné jakékoliv učení. Pro složitější úkoly může být pro člověka obtížnější manuálně vytvořit potřebné algoritmy. Proto je praktičtější nechat počítat, aby si algoritmy vytvořil sám, než aby programátor specifikoval každý krok programu (Alpaydin, 2020).

Disciplína strojového učení využívá mnoho přístupů k naučení počítače zvládnout zadaný úkol. V případě, že existuje velké množství řešení zadaného úkolu, jeden z přístupů k učení programu je označení správných odpovědí. Toto může být použito jako trénovací datová sada, na které se počítač postupně naučí zvolit správnou odpověď na zadaný problém. Kupříkladu při řešení problému rozpoznávání znaků byla pro trénování algoritmu použita sada 60 tisíc obrázků ručně psaných číslic (Alpaydin, 2020).

Oblast strojového učení zahrnuje učení pomocí vyhledávání vzorců v datech. Obecně se strojovým učením myslí změny v systémech, které provádí úkony spojené s umělou inteligencí. Mezi tyto úkoly patří rozpoznávání, analýza, plánování, ovládání robotů, předpovědi a jiné. Tento obor prozkoumává možnosti konstrukce algoritmu, který dokáže předpovídat data. Strojové učení je použito k vytváření programu s měnitelnými parametry, které se postupně adaptují, aby zvýšily svoji schopnost adaptace na již dříve použitá data (Accuracy Prediction for Loan Risk Using Machine Learning Models, 2016).

Strojové učení lze rozdělit do dvou kategorií: Supervised learning (česky učení s učitelem) a Unsupervised learning (česky učení bez učitele).

3.3.1 Supervised learning

Supervised learning (česky učení s učitelem) je metoda strojového učení pro učení funkce z trénovacích dat. Poskytnutá datová sada obsahuje jak features (měřitelná vlastnost, nebo charakteristika pozorovaného jevu), tak lables (cílová proměnná, požadované výstupy). Úkolem je vytvořit algoritmus, který dokáže předpovědět cílovou proměnnou na základě změřených vlastností.

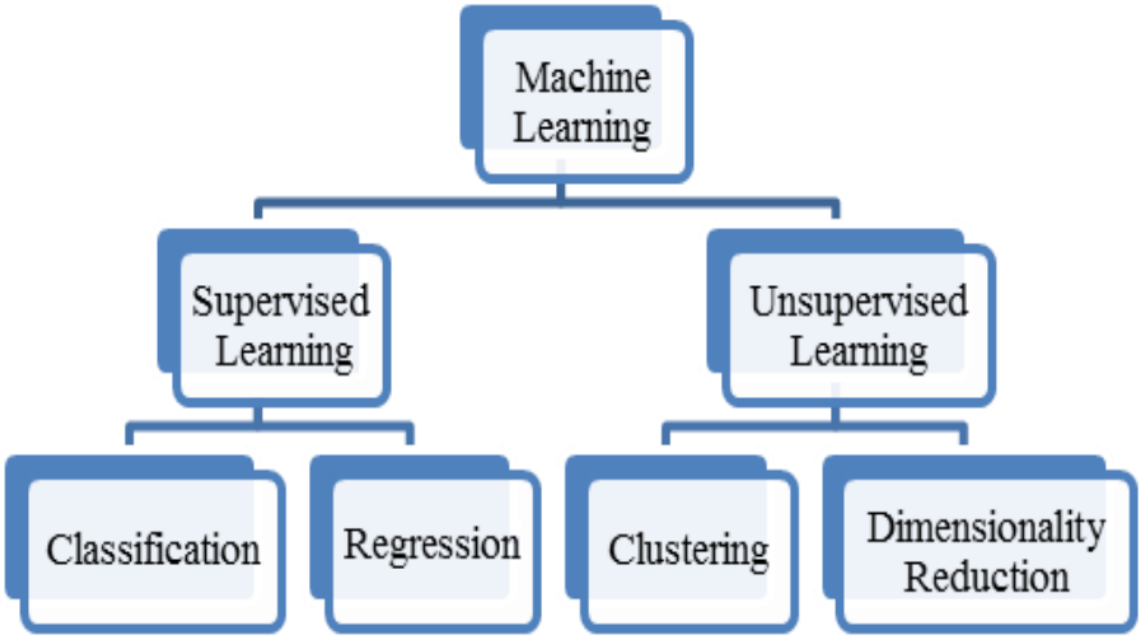
Supervised learning se dál dělí na klasifikaci a regresi. Klasifikační úkoly předpovídají hodnotu kategorické proměnné, zatímco regresní úkoly předpovídají hodnotu kontinuální proměnné, měnící se v čase (například cena, teplota atd.) (Alpaydin, 2020).

3.3.2 Unsupervised learning

Unsupervised learning (česky učení bez učitele) jsou algoritmy, které na rozdíl od supervised learning algoritmů nemají na vstupu data provázaná s cílovou proměnnou. Tato technika se používá k nejvhodnějšímu seřazení dat. Cílem je co nejvhodněji a zpravidla co nejjednodušeji reprezentovat vstupní data.

Unsepervised learning lze rozdělit na kompresi vstupních dat, například jako: snížení dimenzionality dat (aqngl. dimensionality reduction), vytvoření nových charakteristik, kterých je méně než v originální sadě, nebo také jejich redukce na konečný počet diskrétních bodů – shlukováním (angl. clustering), kdy jsou podobné vzorky seřazeny do skupin podle podobných znaků.

Obrázek 3 Kategorie strojového učení



Zdroj: (Alpaydin, 2020)

3.4 Vybrané rozhodovací algoritmy

Pro účely diplomové práce na téma představení rozhodovacích algoritmů v Jupyter Notebook byly vybrány algoritmy podle komplexnosti od jednodušších statistických metod, jako je logistická regrese, přes komplexnější metody založené na strojovém učení, jako je random forest a Support vector machines (SVM) neboli metoda podpůrných vektorů. Dále byly algoritmy vybrány podle popularity využití v praxi.

3.4.1 Logistická regrese

Logistická regrese je v praxi nejpoužívanější technika při vývoji kreditních skórovacích modelů (Credit Risk Analysis Applying Logistic Regression, Neural Networks and Genetic, 2007) (CGAP/World Bank, 2019).

Jedná se o klasifikační algoritmus, který se používá pro odhad pravděpodobnosti binární odezvy v závislosti na několika proměnných. V modelu logistické regrese je závislá proměnná zpravidla binární, tedy nabývá hodnot 1 nebo 0. Nezávislé proměnné jsou kategorické, nebo spojité.

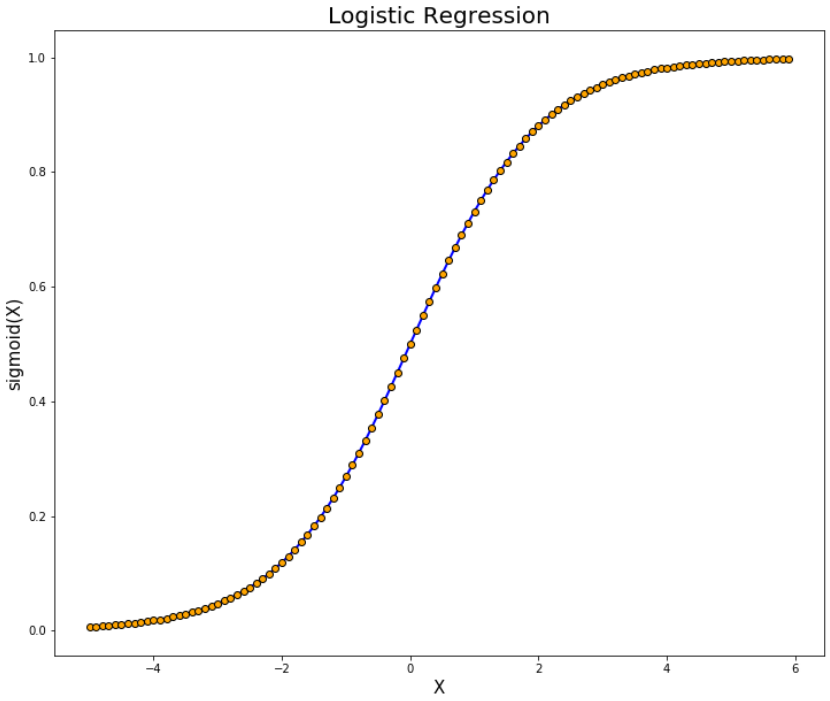
Ve statistice se tento algoritmus používá k modelování pravděpodobností, nějakého jevu na základě určitých známých skutečností, které mohou výskyt jevu ovlivnit. Stav jevu je modelován hodnotou 0, pokud nenastal, a hodnotou 1, pokud nastal. Může se jednat například o úspěchy/selhání, výhru/prohru, zdravý/nemocný, nebo v kreditním riziku dobrý/špatný klient.

Logistická regrese je statistický model, který ve své základní formě používá logistickou funkci k modelování binární cílové proměnné. Logistická funkce nebo logistická křivka je tvaru S (tvaru sigmoidy) s rovnicí:

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}} \tag{2}$$

Kde x_0 je hodnota x ve středu sigmoidy, L je maximální hodnota křivky a k je rychlost růstu křivky. Funkce nabývá hodnot <0;1>.

Obrázek 4 Logistická funkce



Zdroj: (Z_ai, 2020)

V regresní analýze logistická regrese (někdy zkráceně logit) odhaduje parametry logistického modelu.

Jedná se o techniku zobecněného lineárního modelu (angl. generalized linear model GLM), která pomůže odhadnout diskrétní výsledek. Výsledná proměnná logistické regrese je alternativní (Bernoulliho) rozdělení a dosáhne hodnoty 1 s pravděpodobností θ, nebo hodnoty 0 s pravděpodobností *1*- θ.

V analýze kreditního rizika můžeme definovat náhodnou proměnnou D, která nabývá hodnot 1 a 0, kde hodnota 1 (D = 1) znamená, že úvěr selhal, a 0 znamená, že úvěr neselhal. Pravděpodobnost defaultu je definována jako pravděpodobnost úspěchu pro náhodnou proměnnou, tedy θ = P(D = 1).

Vztah mezi odezvou a nezávislými proměnnými je vyjádřen logit transformací p následovně:

$$\theta = \frac{e^{(\alpha + -\beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n)}}{1 + e^{(\alpha + -\beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n)}} \quad (3)$$

Kde α je intercept (výchozí hodnota) rovnice, β jsou koeficienty nezávislých proměnných a n je počet nezávislých proměnných.

Alternativní forma logistické regrese je následující:

$$Logit\,[\theta(x)] = \log\left[\frac{\theta(x)}{1-\theta(x)}\right] = \alpha + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n \quad (4)$$

Nebo v kreditním riziku jako příklad:

$$Logit\,[P(D=1)] = \log\left[\frac{P(D=1)}{1-P(D=1)}\right] = \alpha + \beta_1 * MOB + \beta_2 * FICO + \beta_3 * LTV \quad (5)$$

Kde MOB představuje Month of Booking (počet měsíců od vzniku úvěru), FICO představuje obecně používané kreditní skóre (zejména ve spojených státech) a LTV (loan to value) je poměr mezi úvěrem a zástavní hodnotou. Logistická regrese v tomto případě vypočítá pravděpodobnost defaultu.

Logistická regrese se stala běžně používanou metodikou v modelování kreditního rizika. (Logistic regression and probability of default of developing, 1990) přestavil model logistické regrese ke zjištění pravděpodobnosti defaultu dluhů rozvojových zemí. Studie zahrnovala dluhy 79 zemí v 19letém období. Model předpověděl default státního dluhu Mexika, Brazílie a Argentiny dva roky dopředu.

3.4.2 Decision tree

Decision tree (česky rozhodovací strom) je struktura připomínající flowchart graf, kde každý uzel představuje rozhodování podle jedné (vybrané) vlastnosti objektu (například výsledek hodu mincí může být hlava, nebo orel). Každá větev stromu představuje výsledek rozhodování a každý konečný list (nebo uzel) stromu zobrazuje výslednou klasifikaci.

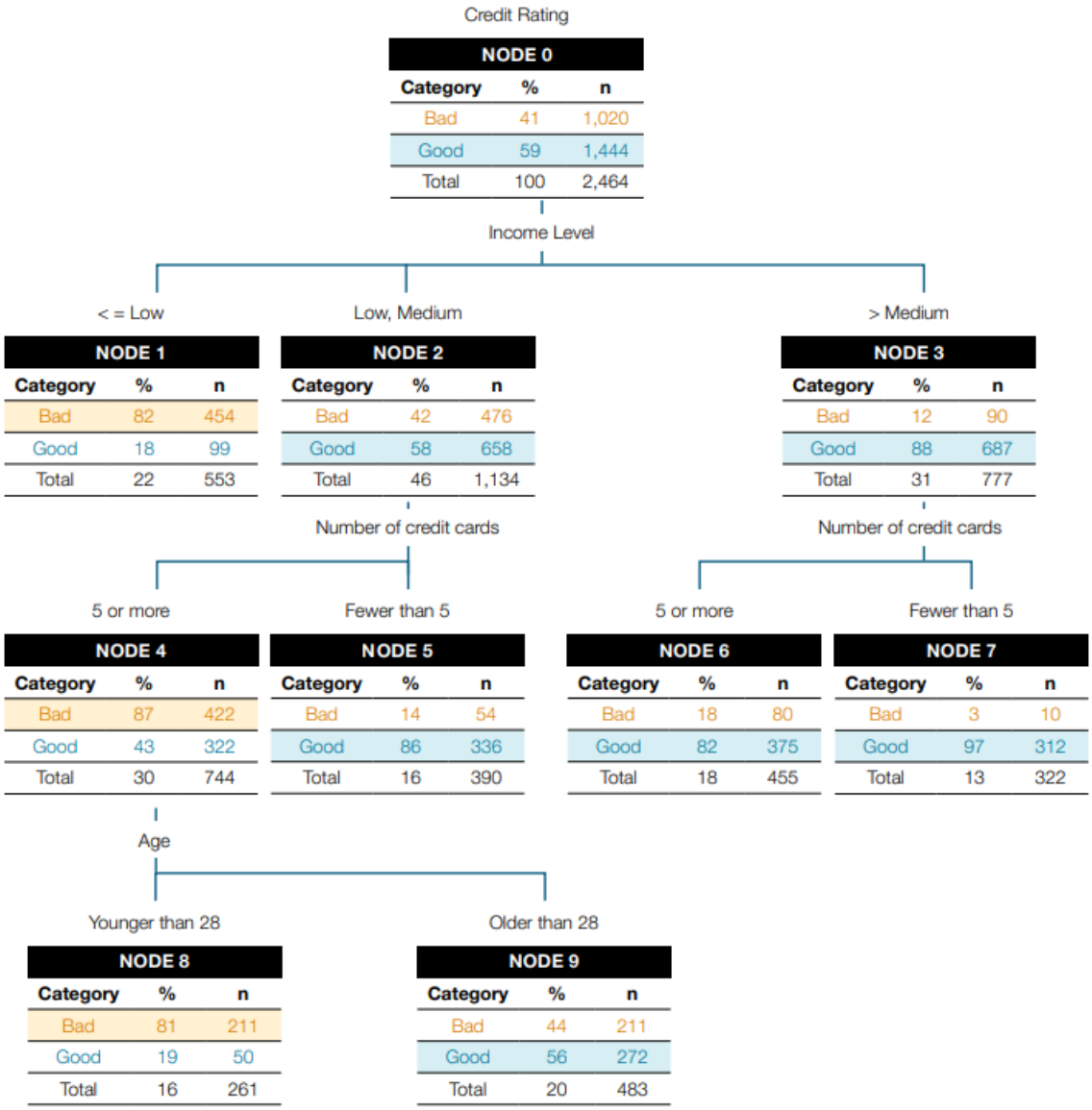
Rozhodovací stromy pomáhají lépe identifikovat skupiny, identifikovat vztahy mezi nimi a odhadnout budoucí stavy na základě cílové proměnné. Stromy dokážou pomoci při porozumění vztahu mezi proměnnými, a jak proměnné v kreditním riziku klasifikují dobré a špatné klienty.

Principem stromu je rozdělování dat do specifických skupin, které vůči ostatním skupinám kontrastují svojí vysokým výskytem cílové proměnné. Stromy testují možné kombinace výstupních proměnných k identifikaci, jaká nejlepší kombinace proměnných nejlépe vysvětluje výsledky a zobrazí tyto zjištěné stavy graficky pomocí uzlů (angl. nodes).

Základem je kořen na vrcholu stromu. Strom se dále otevírá v jednotlivých úrovních pomocí větví, dokud následující větev už dál nic nevysvětluje. Uzly na konci těchto větví jsou konečné a například v případě kreditního rizika vyjadřují hodnocení rizikovosti. Na rozdíl od logistické regrese nepožadují rozhodovací stromy normální rozdělení dat. Jelikož dokážou zpracovat data jakéhokoliv typu, jedná se o relativně jednoduchý způsob, jak zpracovat najednou všechna data a vyhodnotit různé faktory nebo kombinace faktorů, které mohou v kreditním riziku způsobovat selhaní úvěru klienta.

Následující příklad rozhodovacího stromu ukazuje výsledné kreditní hodnocení „Credit Rating“ na základě tří proměnných: Income Level (výška příjmu), Number of credit cards (počet kreditních karet) a Age (věk).

Obrázek 5 Příklad rozhodovacího stromu v kreditním riziku



Zdroj: (CGAP/World Bank, 2019)

Jak lze vidět, tento model identifikuje vztahy mezi proměnnými a vyhodnocuje pravděpodobnosti špatných klientů.

Výsledkem klasifikace stromu jsou konečné uzly, které zobrazuji jednotlivé koncentrace špatných klientů a vyhodnocení rizikovosti.

Obrázek 6 Vyhodnocení rizikovosti

Terminal Node Identifier	Bad	Good	Total	Bad (%)	% of customers	cumulative % of customers
Node 1	454	99	553	82	22	22
Node 8	211	50	261	81	11	33
Node 9	211	272	483	44	20	53
Node 6	80	375	455	18	18	71
Node 5	54	336	390	14	16	87
Node 7	10	312	322	3	13	100

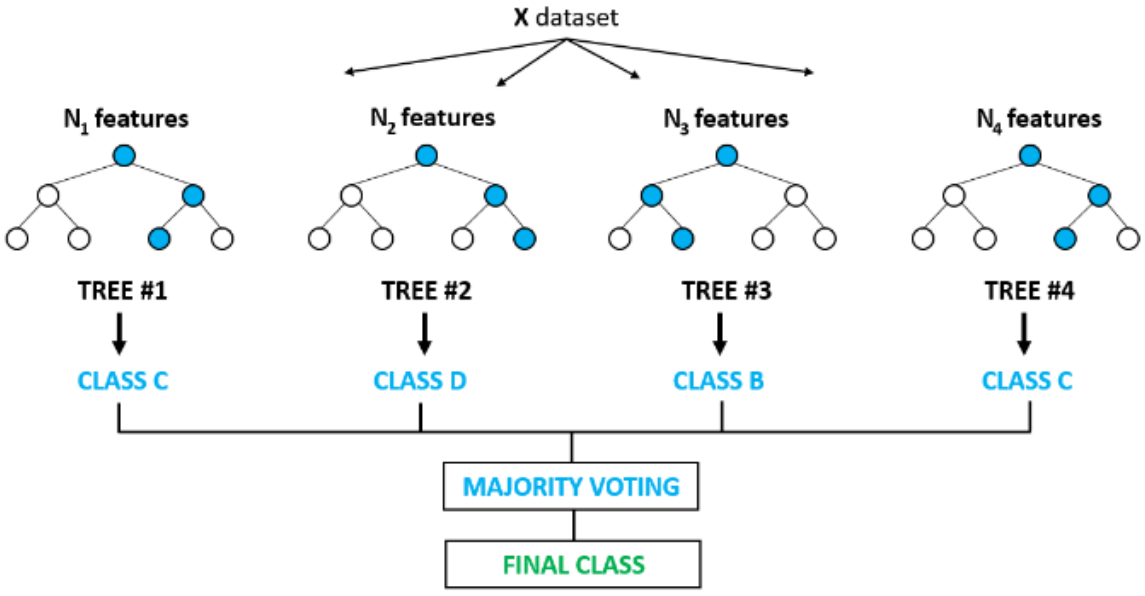
Zdroj: (CGAP/World Bank, 2019)

3.4.3 Random forest

Random forest (náhodný les) je kombinovaná učící metoda pro klasifikaci, regresi a další typy rozhodovacích úkolů. Jak již název napovídá, jedná se o velké množství rozhodovacích stromů, které operují kombinovaně, a výsledkem je u klasifikačních úloh modus jejich výsledku a u regresních úloh průměr výsledku.

Metoda random forest vezme v úvahu výsledné třídy jednotlivých individuálních rozhodovacích stromů a třída s nevíce „hlasy“ se zvolí jako výsledný odhad.

Obrázek 7 Random forest



Zdroj: (Abilash, 2018)

Random forest metoda je obecně přesnější než rozhodovací stromy, jelikož kombinace vysokého množství nekorelovaných stromů pomáhá opravit tendenci jednotlivého stromu overfitting (přeučení) na trénovací datové sadě.

Vyšší přesnost metody random forest má ale i své nevýhody. Rozhodovací strom je jeden z mála modelů strojového učení, který je jednoduše interpretovatelný podobně, jako je tomu u lineárních modelů nebo modelů založených na pravidlech. Právě možnost interpretace výsledků je jedna z nejžádanějších vlastností rozhodovacích stromů. Umožňuje vývojářům modelu potvrdit, že se model naučil realistické informace a také, co je nejdůležitější, umožňuje konečným uživatelům zachovat si důvěru v rozhodnutí učiněná tímto modelem. Sledovat cestu rozhodnutí jednoho rozhodovacího stromu je zcela triviální, naopak sledovat tyto cesty u desítek nebo stovek stromů je už obtížné. Některé modely proto používají kompresní techniky, které dokážou zachovat jak výkon modelu, tak možnost interpretace. Docílí toho transformováním náhodného lesa do jednoho „minimálního“ rozhodovacího stromu, který věrně zkopíruje stejnou rozhodovací funkci (Using Machine Learning to Examine Impact of Type of Performance Indicator on Flexible Pavement Deterioration Modeling, 2021).

3.4.4 Gradient boosting

Oproti metodě náhodného lesa jsou metody gradient boosting založené na boosting algoritmu. Základní myšlenkou boosting algoritmu je vytvoření velkého množství slabých modelů (anglicky weak learner), tedy modelu, jehož přesnost je pouze o málo lepší než náhoda. Velké množství slabých modelů je vytvořeno, aby se modely nevytvořily nezávisle na sobě a nerozdělily trénovací datovou sadu. Nově vytvořený model je napasovaný za využití informací o chybách předchozího modelu. Výsledný prediktor je kombinací všech modelů. Tedy každý nově vzniklý strom je napasován na upravenou verzi originálních dat (Harshdeep, 2018).

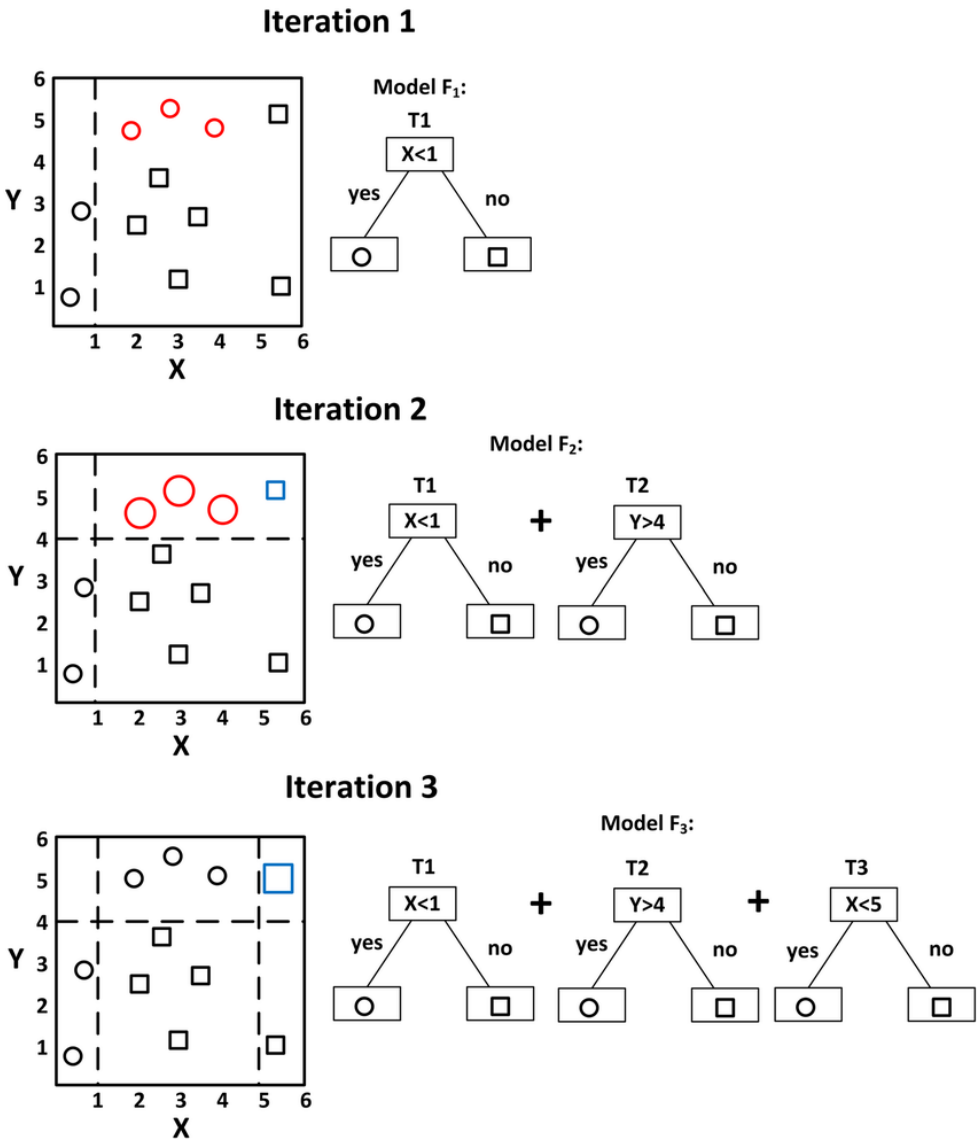
Klasický boosting algoritmus přiřadí na začátku každému pozorování stejnou váhu. Po vyhodnocení prvního stromu se zvýší váhy pro pozorování, které jsou obtížné na roztřídění a sníží váhy pozorování, které jsou jednoduché na roztřídění. Druhý strom je poté postaven na upravených datech s vahami a dojde ke zlepšení oproti prvnímu stromu (Harshdeep, 2018).

Gradient boosting trénuje několik modelů stupňovitě. Hlavní rozdíl mezi klasickým a gradient boosting je identifikace slabých stromů. Klasický boosting používá již zmíněné váhy, zatímco gradient boosting používá ztrátovou funkci (anglicky loss function). Ztrátová funkce je měřítko, které ukazuje, jak dobře dokážou koeficienty modelu vysvětlit originální data. Pro logické pochopení ztrátové funkce záleží na tom, co se snažíme modelem

optimalizovat. Například pokud se snažíme predikovat ceny na trhu pomocí regrese, ztrátová funkce bude rozdíl mezi skutečnou a odhadovanou tržní cenou. V případě kreditního rizika nám ztrátová funkce říká, jak dobrý je náš model v odhadování špatných úvěrů (Harshdeep, 2018) (Browlee, 2016).

Gradient boosting algoritmy se dají široce přizpůsobit. Algoritmus gradient boosting skončí, pokud dosáhne předem specifikovaného maximálního počtu iterací, nebo jiné definované podmínky. Počet iterací je běžně omezen úspěšností předchozí iterace. Pokud několik iterací po sobě nezaznamenalo další zlepšení, dojde k ukončení učení. Další omezení gradient boosting je hloubka stromu. Pokud není algoritmus nijak omezen, vznikají zejména u počátečních iterací příliš velké stromy. Zpravidla se doporučuje vybrat pouze malé stromy, jelikož modely s velkým počtem malých stromů (například pouze hloubka 2) bývají přesnější. Další možná omezení v gradient boosting mohou být pocty uzlu stromů nebo počty listů stromu (Hastie, a další, 2008).

Obrázek 8 Vizualizace kombinování iterací gradient boostingu



Zdroj: (Exploring the clinical features of narcolepsy type 1 versus narcolepsy type 2 from European Narcolepsy Network database with machine learning, 2018)

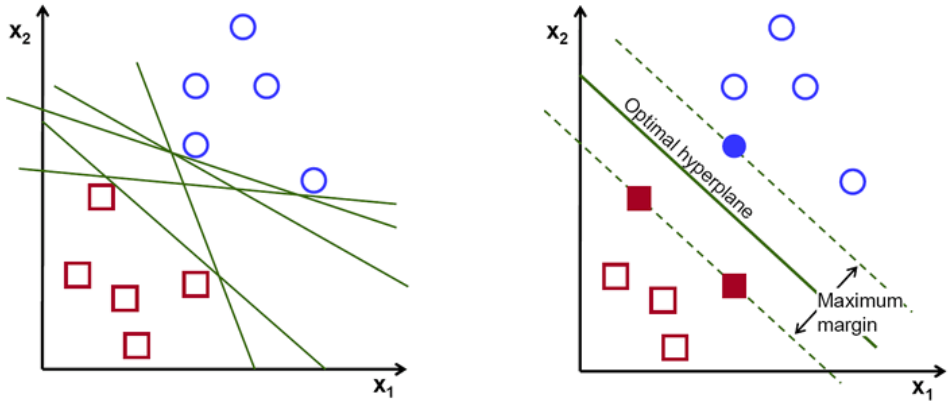
3.4.5 Support vector machine

Support vector machines (SVM) neboli metoda podpůrných vektorů je metoda strojového učení s učitelem používaná jak pro regresní, tak pro klasifikační úkoly, nicméně je preferovaná spíše pro klasifikační úkoly. Tato metoda je často preferovaná pro svoji přesnost a nízkou náročnost na výpočetní výkon (Gandhi, 2018).

Cílem algoritmu Support Vector Machine je nalezení nadroviny v n-dimenzionálním prostoru (n je počet proměnných), která zřetelně definuje trénovací data.

K rozděleni skupin tříd existuje mnoho možných nadrovin. Cílem Support Vector Machine je najít rovinu, u které lze na obě strany nalézt co širší pruh, který neobsahuje žádné body. Tomuto pruhu se říká, že má tzv. maximální odstup (angl. maximal margin) (Gandhi, 2018).

Obrázek 9 Nalezení optimální nadroviny



Zdroj: (Gandhi, 2018)

Nadroviny jsou hranice rozhodování, které pomáhají při klasifikaci dat. Data spadající na jakoukoliv stranu nadroviny jsou roztříděná mezi odlišné třídy. Počet dimenzí nadroviny je definován počtem proměnných. Pokud máme dvě proměnné, nadrovina bude pouze přímka. V případě tří proměnných bude nadrovina dvoudimenzionální rovina. U tří a více proměnných je podoba nadroviny obtížná na představu.

Podpůrné vektory (angl. suport vector) jsou body nejbližší k nadrovině a ovlivňují její pozici a orientaci. Jedná se o body, které nám pomáhají při vytváření modelu support vector machine. Použitím těchto bodů maximalizujeme odstup klasifikátoru.

4 Vlastní práce

4.1 Příprava systému

V následující sekci práce se budeme zabývat přípravou systému pro čtení a úpravu Jupyter Notebooku.

Projekt Jupyter Notebook dokáže podporovat širokou skálu programovacích jazyků. Pro cíl práce testování rozhodovacích algoritmů byl vybrán programovací jazyk Python. Konkrétně byla vybraná populární data science distribuce Anaconda verze 2020.11, který používá verzi Python 3.8.5.

4.1.1 Anaconda

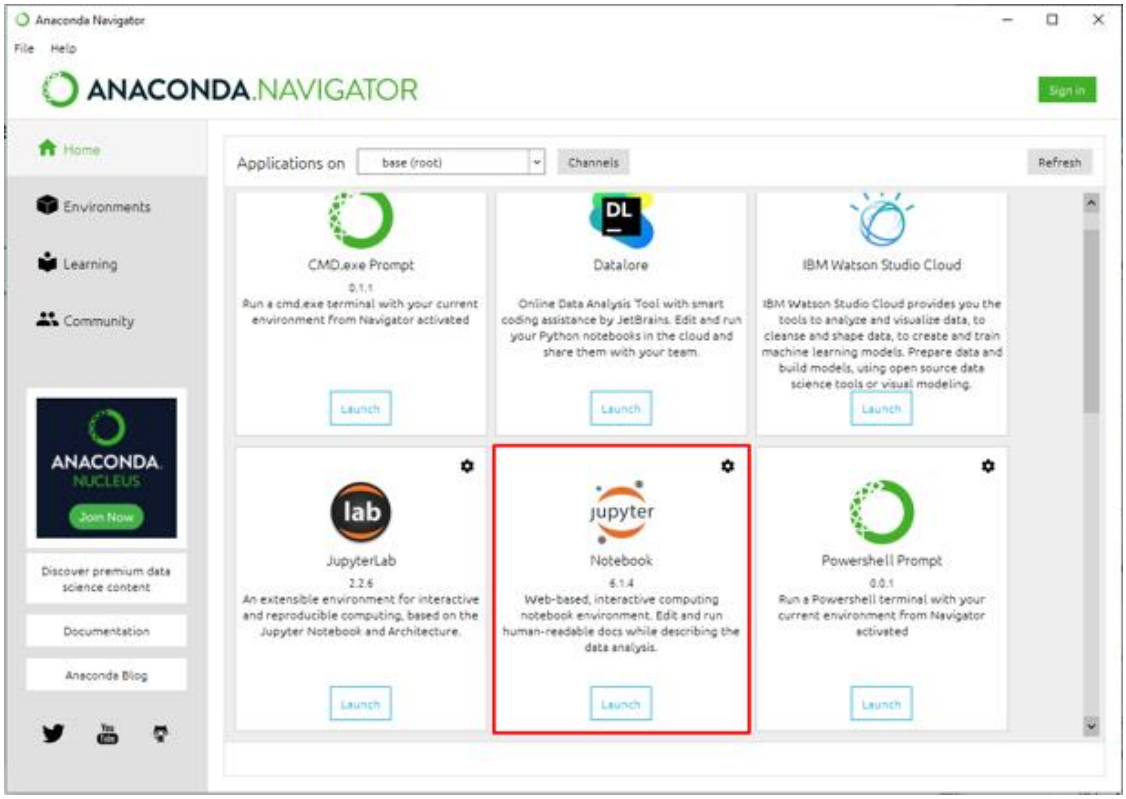
Anaconda je nejpoužívanější distribuce jazyků Python a R určená zejména pro vědecké výpočetní projekty. Ať už se jedná o klasická data science, strojové učení nebo umělou inteligenci, distribuce Anaconda je poskytována zdarma a po jednoduché instalaci, poskytuje správu programových knihoven. Python distribuce obsahuje přes 1500 populárních open source knihoven, které se automaticky aktualizují (Anaconda, 2021).

Pro účely diplomové práce byla zvolena verze Anaconda3-2020.11 64-Bit dostupná na webu anaconda.com.

4.1.2 Jupyter Notebook

Jak již bylo zmíněno, distribuce Anaconda obsahuje velké množství populárních knihoven se zaměřením na data science. Jupyter notebook mezi tyto knihovny patří a je předinstalovaný. Není tedy potřeba dále instalovat.

Obrázek 10 Jupyter Notebook v distribuci Anaconda

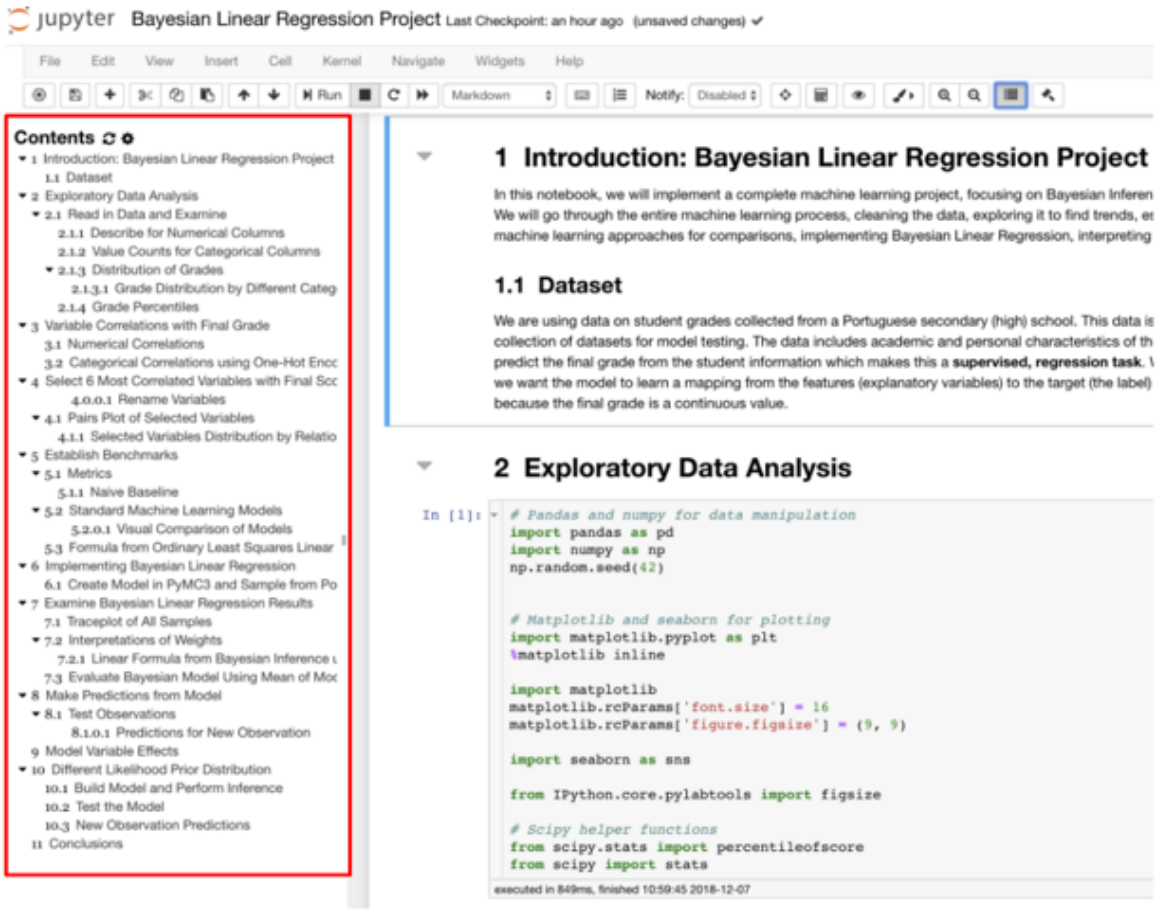


Zdroj: Vlastní zpracování

Instalace Jupyter widgetu Jupyter Notebook Extensions

Pro snazší navigaci v Jupyter notebooku byl zvolen neoficiální Jupyter doplněk **Jupyter Notebook Extensions**. Jedná se o widget, který vytvoří interaktivní okno navigace po nadpisech v celém Jupyter Notebooku.

Obrázek 11 Okno navigace v Jupyter Nootebooku – označeno červeně



Zdroj: (Koehrsen, 2021)

Instalace widgetu se provádí přes terminál spuštěný v Anaconda. Stažení a instalace proběhne automaticky po zadání následujícího příkazu:

Obrázek 12 Instalace widget nbextensions

```
(base) C:\Users\mmand>conda install -c conda-forge jupyter_contrib_nbextensions
```

Zdroj: (Vlastní zpracování, 2021)

Otevření Jupyter Notebooku

Po správné instalaci Jupyter Notebooku můžeme Jupyter otevřít buď přes Anacondu, kliknutím na ikonku Jupyter, nebo zadáním adresy http://localhost:8888/ v libovolném webovém prohlížeči.

4.1.3 Python a potřebné software knihovny

Vytvoření a testování rozhodovacích algoritmů bude prováděno v programovacím jazyku Python. Vybraná distribuce Anaconda obsahuje verzi Python 3.8.5. K dosažení cíle práce budou potřeba následující Python knihovny:

Pandas – knihovna používaná zejména pro datovou analýzu. Pandas umožňuje import dat z různých zdrojových formátů, jako jsou: CSV, JSON, SQL a Microsoft Excel. Data jsou ukládána v tabulárních strukturách, kde sloupce představují jednotlivé proměnné a řádky sledování. Kromě možnosti manipulace s daty umožňuje dále tato knihovna různé operace pro čistění dat jako například vyplnění chybějících hodnot, nebo také výpočty statistických ukazatelů (Galea, 2018) (pandas, 2021).

NumPy – knihovna NumPy poskytuje pokročilejší matematické funkce, ale také možnost manipulace s vícerozměrnými datovými sadami. Operace provedené knihovnou NumPy probíhají na pozadí pomocí programovacího jazyka C. Toto umožňuje mnohonásobně rychlejší řešení operací, než kdyby se provedly klasickou Python strukturou (Galea, 2018) (NumPy, 2021).

Matplotlib – knihovna slouží k vizualizace informativních statistických grafik. Je inspirovaná populární matematickou platformou MATLAB. Jedná se o jednu z nejpopulárnějších Python knihoven pro vizualizaci hodnot (Matplotlib, 2021) (Galea, 2018).

Seaborn – seaborn slouží jako rozšíření knihovny matplotlib a je úzce integrován s knihovnou pandas. Jsou zde přidány různorodé nástroje pro vizualizaci, data science a obecně umožňuje některé problémy vyřešit rychleji, než kdyby se řešily manuálně jen pomocí knihoven, jako jsou matplotlib nebo scikit-learn (seaborn, 2021) (Galea, 2018).

XGBoost – open-source knihovna poskytující algoritmus strojového učení založený na gradientních rozhodovacích stromech. Knihovna je navržena, aby byla co nejefektivnější a flexibilní. Tento algoritmus získal v současné době velkou popularitu. Stala se z něj první volba v soutěžích zabývajících se strojovým učením (xgboost developers, 2020) (dmlc, 2020).

Většina výše zmíněných Python knihoven je již obsazena v distribuci Anaconda, a není je tedy potřeba instalovat. Jediná potřebná software knihovna, která není součástí základní Anaconda instalace, je knihovna XGBoost. Knihovnu stáhneme a nainstalujeme následujícím příkazem v Anaconda terminálu:

Obrázek 13 Instalace XGBoost knihovny

```
(base) C:\Users\mmand>conda install -c anaconda py-xgboost
```

Zdroj: (Vlastní, 2021)

Po správné instalaci knihoven, můžeme v novém Jupyter Notebooku knihovny importovat, aby byly připravené k použití:

Obrázek 15 Import potřebých knihoven

```
1 import pandas as pd
2 import numpy as np
3 import seaborn as sns
4 import matplotlib.pyplot as plt
5 %matplotlib inline
6 import warnings
7 import xgboost
8 warnings.filterwarnings('ignore')
```

Zdroj: Vlastní (2021)

4.2 Použitá data

Poskytování úvěru je základním obchodem banky. Hlavní zdroje příjmu přichází ve formě úroků z úvěru. Ačkoli jsou úvěry poskytovány na základě intenzivního procesu schvalování, který obnáší verifikaci a validaci klienta, neznamená to, že bude mít banka jistotu řádného splácení ze strany klienta.

Banky si čím dál tím víc uvědomují přenosnost optimalizace rozhodovacího systému a díky tomu se z toho stala populární činnost v různých komunitách zabývajících se strojovým učením. Ve zmíněných komunitách vznikají soutěže, kde se týmy z celého světa snaží o vymýšlení nejoptimálnějšího algoritmu pro zadaný problém. Některé soutěže jsou dokonce organizovány ze strany bank a nejúspěšnější tým může vyhrát peněžní cenu v řádech desetitisíc dolarů.

Pro účely diplomové práce byla vybrána vzorová data soutěže „**Loan Prediction Practice Problem“** z komunity **Analytics Vydhya**. Pro lepší porozumění byla data přeložena do češtiny (Analytics Vidhya, 2016).

4.2.1 Datová sada

K praktickému předvedení rozhodovacích algoritmů v Jupyter Notebooku bude použita datová sada ve formátu CSV (Comma-separated values, hodnoty oddělené čárkami).

Soubor **train.csv** je datová sada tabulárních dat určený pro „trénování“ modelu. Na tomto souboru budou předvedeny vybrané rozhodovací algoritmy a náš finální model se na tomto souboru naučí rozpoznávat kreditní riziko. Datová sada obsahuje všechny nezávisle proměnné a cílovou proměnnou. Celkem se jedná o 13 sloupců a 614 řádků vzorových dat.

4.2.2 Čtení dat

Pro použití zmíněných datových sad je potřeba sady načíst. Načtení dat provedeme funkcí „read_csv“ z knihovny pandas. CSV data používají jako oddělovač středník, proto je potřeba použít argument sep=‘;‘

Vzorová data lze zobrazit funkcí head:

Obrázek 16 Načtení a zobrazení dat

```
1 train = pd.read_csv('Dataset/train.csv', sep=';')
2 train.head()
```

	Cislo_Zadosti	Pohlavi	Zenaty_Vdana	Clenove_Rodiny	Vzdelani	OSVC	Prijem_Zadatele	Prijem_Spoluzadatele	Vyse_Uveru	Maturita	Kreditni_Historie	Mira_Urbanizace	Cilova_Promenna
0	LP001002	Muz	Ne	0	Magisterske	Ne	5849	0.0	NaN	360.0	1.0	Mestska	A
1	LP001003	Muz	Ano	1	Magisterske	Ne	4583	1508.0	128.0	360.0	1.0	Venkovska	N
2	LP001005	Muz	Ano	0	Magisterske	Ano	3000	0.0	66.0	360.0	1.0	Mestska	A
3	LP001006	Muz	Ano	0	Bakalarske	Ne	2583	2358.0	120.0	360.0	1.0	Mestska	A
4	LP001008	Muz	Ne	0	Magisterske	Ne	6000	0.0	141.0	360.0	1.0	Mestska	A

Zdroj: Vlastní zpracování

Jak již bylo zmíněno, v datové sadě máme celkem 12 nezávislých proměnných a jednu cílovou proměnnou.

Obrázek 17 Sloupce datové sady

```
1 train.columns
```

```
Index(['Cislo_Zadosti', 'Pohlavi', 'Zenaty_Vdana', 'Clenove_Rodiny',
       'Vzdelani', 'OSVC', 'Prijem_Zadatele', 'Prijem_Spoluzadatele',
       'Vyse_Uveru', 'Maturita', 'Kreditni_Historie', 'Mira_Urbanizace',
       'Cilova_Promenna'],
      dtype='object')
```

Zdroj: Vlastní zpracování

Datové formáty jednotlivých sloupců v datové sadě můžeme dále zobrazit pomocí funkce dtypes.

Obrázek 18 Zobrazení datových typů

```
1 train.dtypes
```

```
Cislo_Zadosti           object
Pohlavi                 object
Zenaty_Vdana            object
Clenove_Rodiny          object
Vzdelani                object
OSVC                    object
Prijem_Zadatele          int64
Prijem_Spoluzadatele    float64
Vyse_Uveru              float64
Maturita                float64
Kreditni_Historie       float64
Mira_Urbanizace          object
Cilova_Promenna          object
dtype: object
```

Zdroj: Vlastní zpracování

V našich datech můžeme vidět tři datové formáty:

Object – představuje textové nebo smíšené textové a číselné hodnoty. Jedná se tedy zejména o kategorické proměnné.

Float64 – číselný formát. Hodnoty jsou číselné a mají desetinné hodnoty.

Int64 – číselný formát bez desetinných hodnot.

4.3 Explorační analýza dat

Ke správnému sestavení a pochopení modelu je potřeba správně porozumět použitým datům. Průzkum dostupných dat provedeme explorační analýzou, kdy prozkoumáme jednotlivé proměnné a vazby mezi nimi.

Základní popis jednotlivých proměnných je dostupný z dokumentace datové sady:

Proměnná	**Popis**
Cislo_Zadosti	Unikátní identifikátor úvěrové žádosti
Pohlavi	Pohlaví žadatele
Zenaty_Vdana	Rodinný stav žadatele
Clenove_Rodiny	Počet členů rodiny závislých na žadateli
Vzdelani	Vzdělání žadatele
OSVC	Příznak, zda je žadatel osoba samostatně výdělečně činná (OSVČ)
Prijem_Zadatele	Celkový příjem žadatele
Prijem_Spoluzadatele	Celkový příjem spolužadatele
Vyse_Uveru	Výše požadovaného úvěru v tisících
Maturita	Délka úvěru v měsících
Kreditni_Historie	Příznak pozitivní kreditní historie žadatele
Mira_Urbanizace	Míra urbanizace bydliště žadatele
Cilova_Promenna	Cílová proměnná určující dobré a špatné žádosti

Trénovací datová sada obsahuje celkem 422 dobrých a 192 špatných žádostí. Proporce proměnné Cilova_Promenna jsou tedy 68,73 % dobré žádosti a 31,27 % špatné žádosti:

Obrázek 19 Poměr výskytu cílové proměnné

```
train['Cilova_Promenna'].value_counts(normalize=True)
```

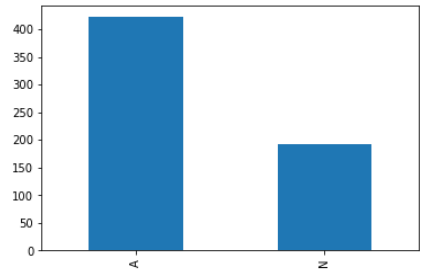
```
A    0.687296
N    0.312704
Name: Cilova_Promenna, dtype: float64
```

Zdroj: Vlastní zpracování

Obrázek 20 Graf četnosti výskytu cílové proměnné

```
train['Cilova_Promenna'].value_counts().plot.bar()
```

```
<AxesSubplot:>
```



Zdroj: Vlastní zpracování

Ostatní proměnné lze roztřídit do tří kategorií podle druhu:

- kategorické,
- ordinální,
- kvantitativní.

4.3.1 Kategorické proměnné

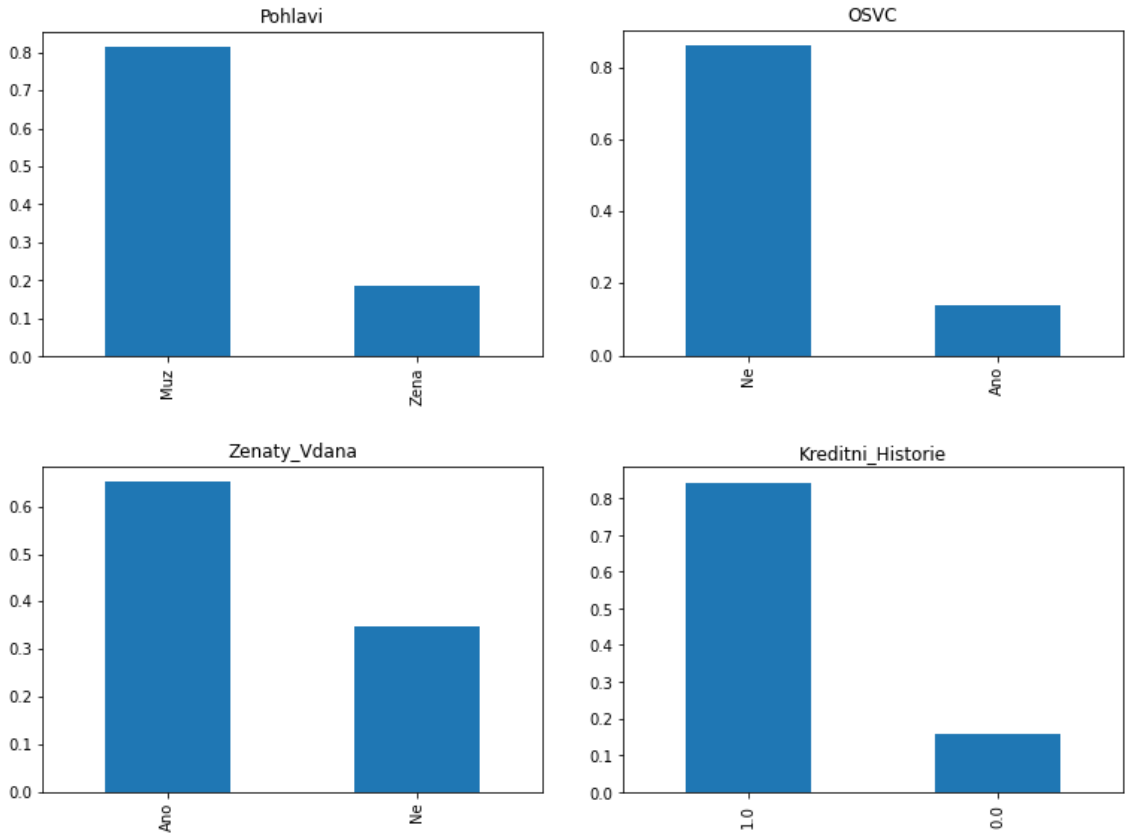
Kategorické proměnné, které představují jisté kategorie. Kategorie jsou bez přirozeného uspořádání a o každých dvou variantách lze pouze říct, zda jsou stejné, nebo různé. Mezi tyto proměnné patří: Pohlavi, Zenaty_Vdana, OSVC, Kreditni_Historie.

Obrázek 21 Zobrazení poměru kategorických proměnných

```
train['Pohlavi'].value_counts(normalize=True).plot.bar(title='Pohlavi')
plt.show()
train['Zenaty_Vdana'].value_counts(normalize=True).plot.bar(title='Zenaty_Vdana')
plt.show()
train['OSVC'].value_counts(normalize=True).plot.bar(title='OSVC')
plt.show()
train['Kreditni_Historie'].value_counts(normalize=True).plot.bar(title='Kreditni_Historie')
plt.show()
```

Zdroj: Vlastní zpracování

Obrázek 22 Grafy poměrů kategorických proměnných



Zdroj: Vlastní zpracování

Jak lze z grafů vidět, pouze 20 % žadatelů jsou ženy, 85 % žadatelů nejsou OSVC, více jak polovina žadatelů jsou ženatý/vdaná a více jak 85 % žadatelů má kreditní historii.

4.3.2 Ordinální proměnné

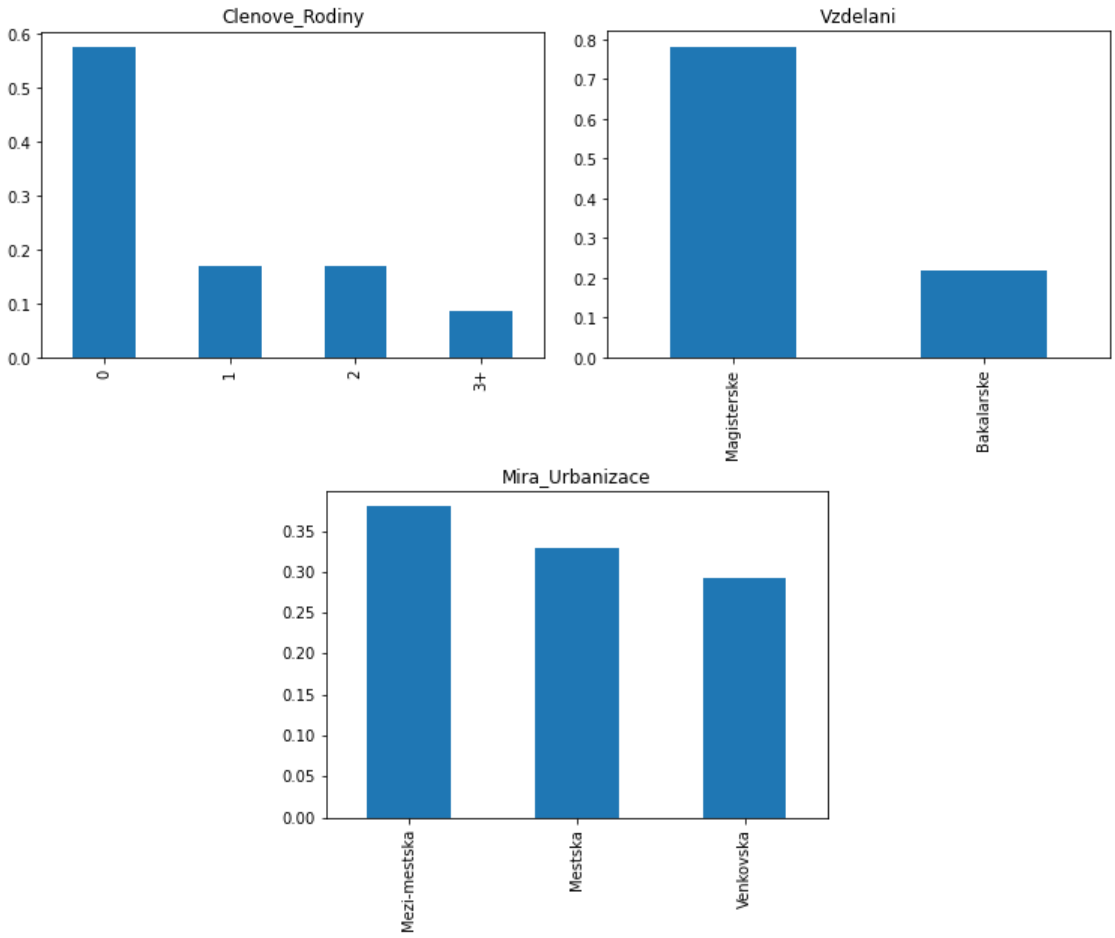
Ordinální proměnné jsou kategorie s přirozeným uspořádáním. Všechny varianty těchto proměnných lze určitým způsobem seřadit. Ordinální proměnné naší datové sady jsou proměnné: Clenove_Rodiny, Vzdelani a Mira_Urbanizace.

Obrázek 23 Zobrazení poměrů ordinálních proměnných

```
train['Clenove_Rodiny'].value_counts(normalize=True).plot.bar(title='Clenove_Rodiny')
plt.show()
train['Vzdelani'].value_counts(normalize=True).plot.bar(title='Vzdelani')
plt.show()
train['Mira_Urbanizace'].value_counts(normalize=True).plot.bar(title='Mira_Urbanizace')
plt.show()
```

Zdroj: Vlastní zpracování

Obrázek 24 Grafy poměrů ordinálních proměnných



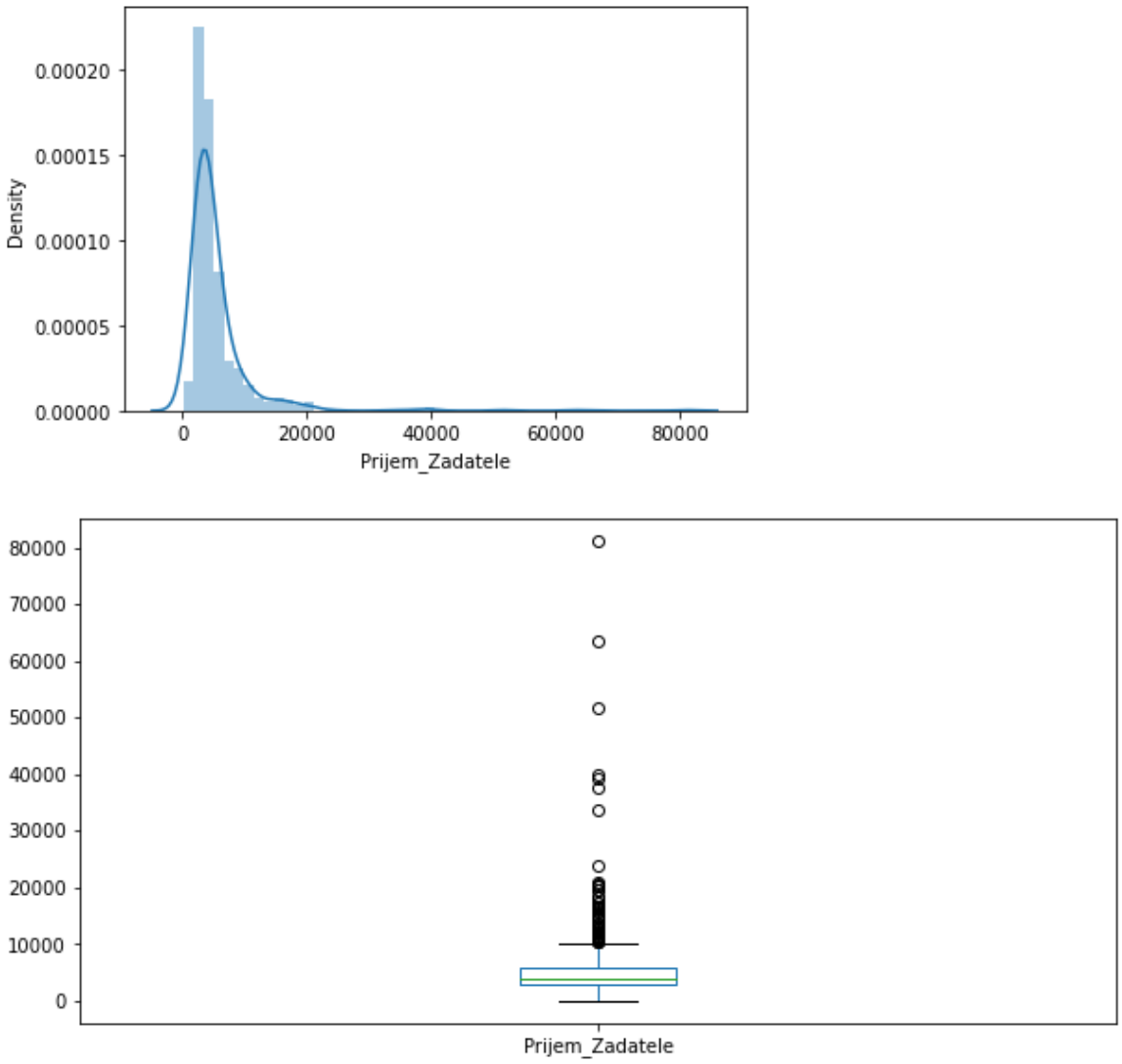
Zdroj: Vlastní zpracování

Z grafů lze konstatovat, že většina žadatelů nemá člena rodiny, který je na něm závislý. Kolem 80 % žadatelů má alespoň magisterské vzdělání a 40 % žadatelů má bydliště v místě s meziměstskou urbanizací.

4.3.3 Kvantitativní proměnné

Kvantitativní proměnné nabývají číselných hodnot. Nejprve se podíváme na distribuci příjmu žadatele:

Obrázek 25 Graf distribuce proměnné Prijem_Zadatele



Zdroj: Vlastní zpracování

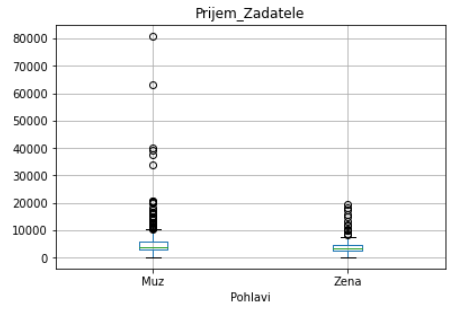
Jak lze z grafů vidět, příjmy žadatelů se chýlí k levé straně grafu, což znamená, že data nemají normální rozdělní. Odchylky v příjmech žadatelů (80000+) proto budeme muset dále ošetřit, jelikož algoritmy, které použijeme, lépe fungují na data s normálním rozdělením.

Hypotézu o normálním rozdělení potvrzuje i tzv. box plot graf, kde jsou extrémní hodnoty vidět.

Obrázek 26 Příjem žadatele podle pohlaví

```
train.boxplot(column='Prijem_Zadatele', by = 'Pohlavi')
plt.suptitle('')
```

```
Text(0.5, 0.98, '')
```

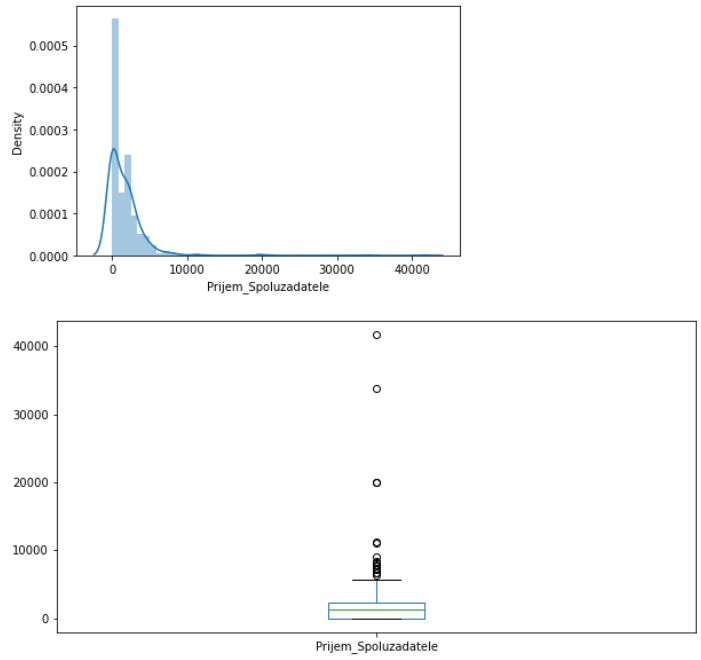


Zdroj: Vlastní zpracování

Při rozdělení podle pohlaví lze vidět, že extrémní hodnoty se vyskytují zejména u mužů.

Obrázek 27 Distribuce proměnné Prijem Spoluzadatele

```
sns.distplot(train['Prijem_Spoluzadatele'])
plt.show()
train['Prijem_Spoluzadatele'].plot.box(figsize=(10,5))
plt.show()
```

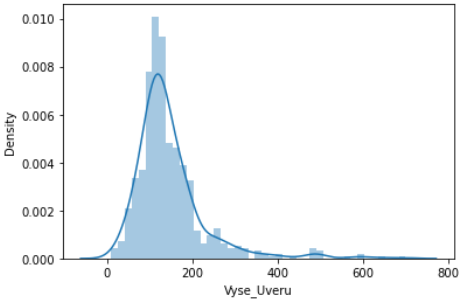


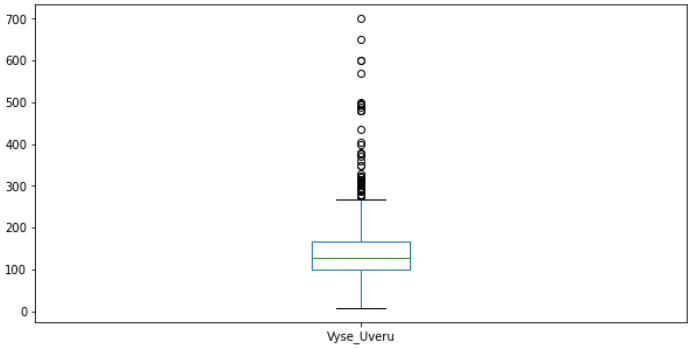
Zdroj: Vlastní zpracování

Podobnou distribuci jako u příjmu žadatele lze vidět i u příjmu spolužadatele. Většina spolužadatelů má příjem mezi 0 a 5500, nicméně jsou zde také extrémy s příjmem přes 40000, které budeme muset v dalších kapitolách ošetřit.

Obrázek 28 Distribuce výše příjmu

```
train.notna()
sns.distplot(train['Vyse_Uveru'])
plt.show()
train['Vyse_Uveru'].plot.box(figsize=(10,5))
plt.show()
```





Zdroj: Vlastní zpracování

Proměnná výše úvěru také nabývá extrémních hodnot (600+). Presto lze konstatovat, že je rozdělení normální.

4.3.4 Vztahy kategorické a cílová proměnná

Pomocí sloupcových grafů můžeme porovnat vztahy mezi kategorickými proměnnými a cílovou proměnnou.

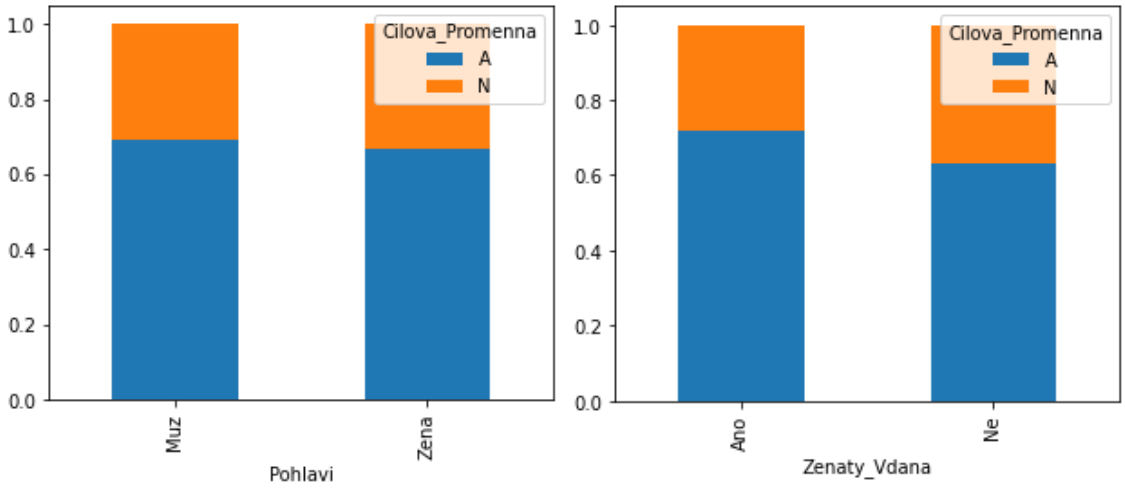
Obrázek 29 Kombinování proměnných pomocí pandas a jejich zobrazení

```
Zenaty_Vdana=pd.crosstab(train['Zenaty_Vdana'],train['Cilova_Promenna'])
Clenove_Rodiny=pd.crosstab(train['Clenove_Rodiny'],train['Cilova_Promenna'])
Vzdelani=pd.crosstab(train['Vzdelani'],train['Cilova_Promenna'])
OSVC=pd.crosstab(train['OSVC'],train['Cilova_Promenna'])
Kreditni_Historie=pd.crosstab(train['Kreditni_Historie'],train['Cilova_Promenna'])
Mira_Urbanizace=pd.crosstab(train['Mira_Urbanizace'],train['Cilova_Promenna'])

Pohlavi.div(Pohlavi.sum(1).astype(float), axis=0).plot(kind='bar',stacked=True,figsize=(5,4))
plt.show()
Zenaty_Vdana.div(Zenaty_Vdana.sum(1).astype(float), axis=0).plot(kind='bar',stacked=True,figsize=(5,4))
plt.show()
Clenove_Rodiny.div(Clenove_Rodiny.sum(1).astype(float), axis=0).plot(kind='bar',stacked=True,figsize=(5,4))
plt.show()
Vzdelani.div(Vzdelani.sum(1).astype(float), axis=0).plot(kind='bar',stacked=True,figsize=(5,4))
plt.show()
OSVC.div(OSVC.sum(1).astype(float), axis=0).plot(kind='bar',stacked=True,figsize=(5,4))
plt.show()
Kreditni_Historie.div(Kreditni_Historie.sum(1).astype(float), axis=0).plot(kind='bar',stacked=True,figsize=(5,4))
plt.show()
Mira_Urbanizace.div(Mira_Urbanizace.sum(1).astype(float), axis=0).plot(kind='bar',stacked=True,figsize=(5,4))
plt.show()
```

Zdroj Vlastní zpracování

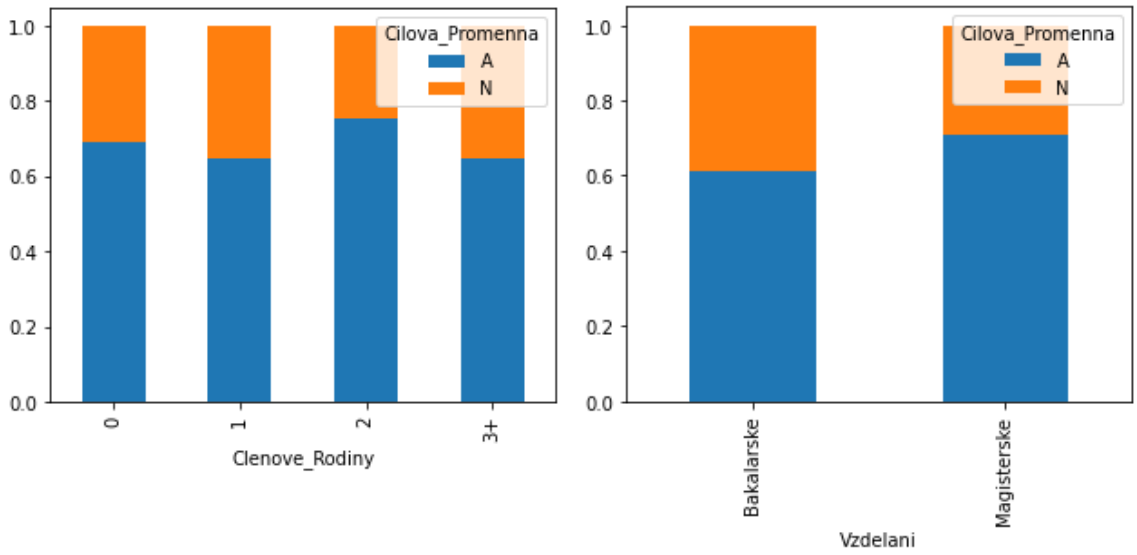
Obrázek 30 Poměry cílové proměnné Pohlaví a Rodinný stav



Zdroj: Vlastní zpracování

Proporce počtu žadatelů mužů a žen je u dobrých a špatných žádostí stejná. Pohlaví žadatele tedy nemá vliv na kvalitu žádosti. U svobodných lidí lze očekávat lehce horší kvalitu žádosti.

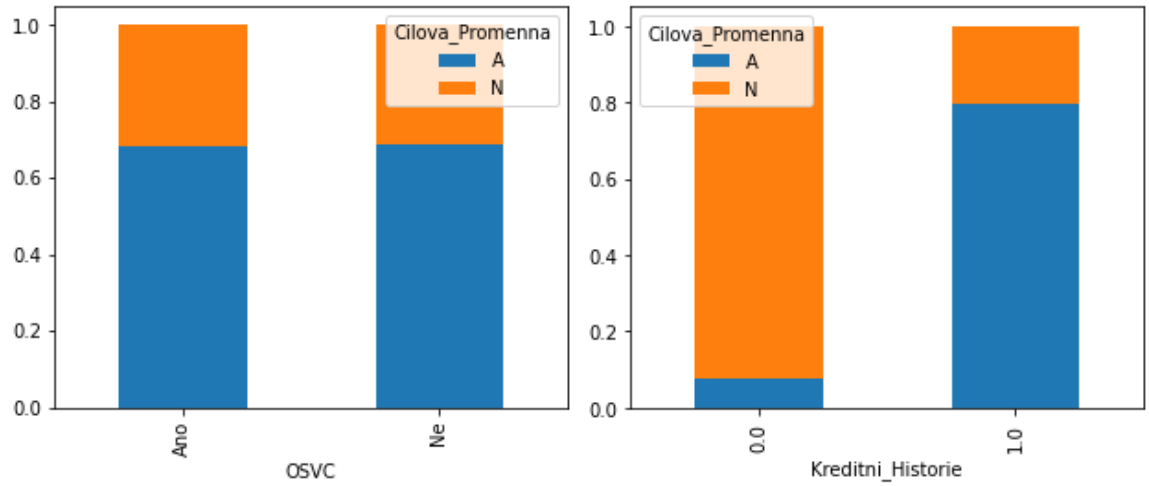
Obrázek 31 Poměry cílové proměnné a proměnných Clenove Rodiny a Vzdělání



Zdroj: Vlastní zpracování

Distribuce špatných žádostí je ve všech kategoriích počtu závislých členů rodiny stejná. U lidí s bakalářským vzděláním lze očekávat lehce horší kvalitu žádostí.

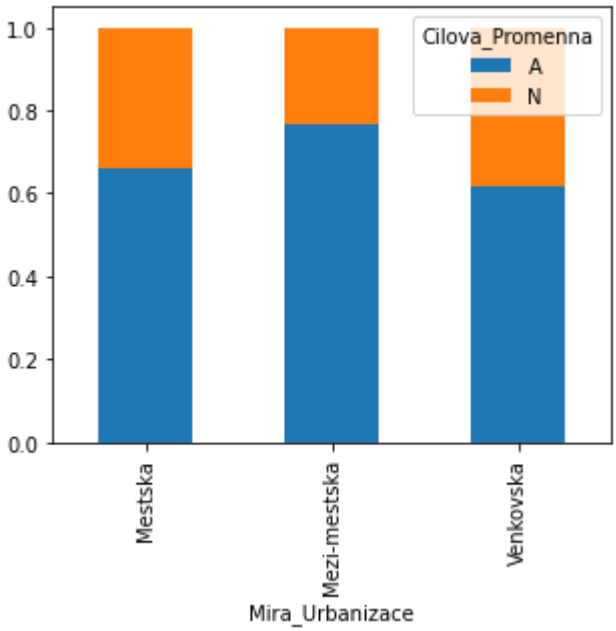
Obrázek 32 Poměry cílové proměnné a OSVC/Kreditní historie



Zdroj: Vlastní zpracování

Fakt, zda je osoba samostatně výdělečně činná, nemá žádný vliv na žádost. Informace o pozitivní kreditní historii klienta má značný na kvalitu žádosti.

Obrázek 33 Poměr cílové proměnné a míry urbanizace



Zdroj: Vlastní zpracování

Z posledního grafu můžeme konstatovat, že u žadatelů z meziměstských oblastí je větší pravděpodobnost kvalitní žádosti.

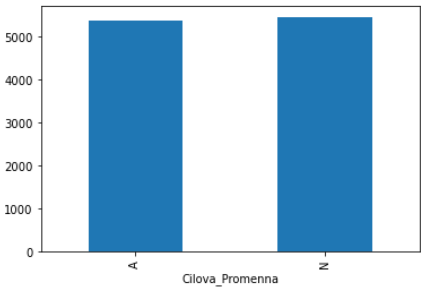
4.3.5 Vztahy kvantitativní a cílová proměnná

Prvním krokem je zjištění průměrného příjmu žadatele u dobrých a špatných žádostí. Toho docílíme vytvořením sloupcového grafu příjmů pro hodnoty cílové proměnné.

Obrázek 34 Průměrný příjem žadatele pro cílovou proměnnou

```
1 train.groupby('Cilova_Promenna')['Prijem_Zadatele'].mean().plot.bar()
```

```
<AxesSubplot:xlabel='Cilova_Promenna'>
```

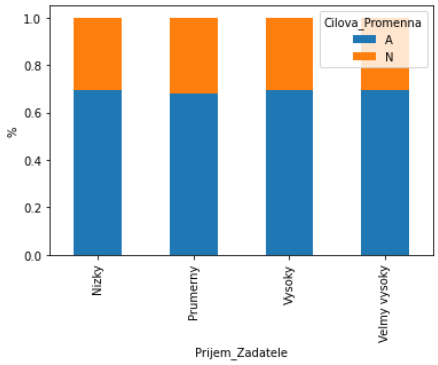


Zdroj: Vlastní zpracování

Z grafu lze vidět, že průměrný příjem je pro obě kategorie stejný. Dále můžeme příjmy žadatelů seskupit do binů podle velikosti: nízký, průměrný, vysoký a velmi vysoký.

Obrázek 35 Rozdělení příjmu žadatele do binů

```
bins=[0,3000,5000,8000,80000]
group=['Nizky','Prumerny','Vysoky','Velmy vysoky']
train['Prijem_bin']=pd.cut(train['Prijem_Zadatele'],bins,labels=group)
Prijem_bin=pd.crosstab(train['Prijem_bin'],train['Cilova_Promenna'])
Prijem_bin.div(Prijem_bin.sum(1).astype(float), axis=0).plot(kind='bar',stacked=True)
plt.xlabel('Prijem_Zadatele')
P=plt.ylabel('%')
```

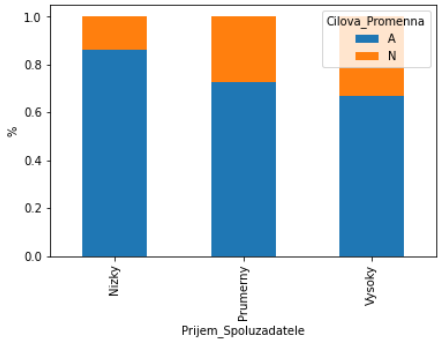


Zdroj: Vlastní zpracování

Lze vidět, že velikost příjmu žadatele nemá vliv na úvěruschopnost klienta. Stejný postup můžeme provést pro příjem spolužadatele.

Obrázek 36 Rozdělení příjmu spolužadatele do binů

```
bins=[0,1000,3000,42000]
group=['Nizky','Prumerny','Vysoky']
train['Prijem_Spoluzadatele_bin']=pd.cut(train['Prijem_Spoluzadatele'],bins,labels=group)
Prijem_Spoluzadatele_bin=pd.crosstab(train['Prijem_Spoluzadatele_bin'],train['Cilova_Promenna'])
Prijem_Spoluzadatele_bin.div(Prijem_Spoluzadatele_bin.sum(1).astype(float), axis=0).plot(kind='bar',stacked=True)
plt.xlabel('Prijem_Spoluzadatele')
P=plt.ylabel('%')
```

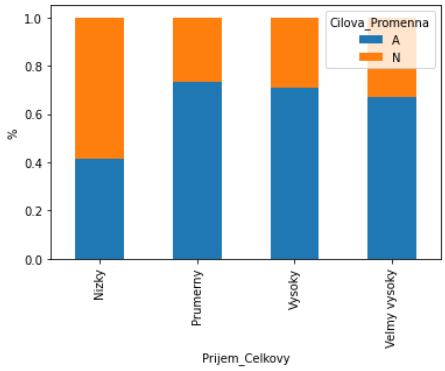


Zdroj: Vlastní zpracování

Graf ukazuje, že nižší příjem spolužadatele zvyšuje šanci na kvalitní žádost. Nicméně nulový příjem spolužadatele nemusí být automaticky špatný. Může to znamenat, že na žádosti je pouze žadatel, nikoliv dva žadatelé. Způsob, jak se vyhnout špatné interpretaci, může být sečtení všech příjmů na žádosti do nové proměnné Prijem_Celkovy.

Obrázek 37 Rozdělení celkového příjmu do binů

```
train['Prijem_Celkovy']=train['Prijem_Zadatele']+train['Prijem_Spoluzadatele']
bins=[0,2500,4000,6000,81000]
group=['Nizky','Prumerny','Vysoky','Velmy vysoky']
train['Prijem_Celkovy_bin']=pd.cut(train['Prijem_Celkovy'],bins,labels=group)
Prijem_Celkovy_bin=pd.crosstab(train['Prijem_Celkovy_bin'],train['Cilova_Promenna'])
Prijem_Celkovy_bin.div(Prijem_Celkovy_bin.sum(1).astype(float), axis=0).plot(kind='bar',stacked=True)
plt.xlabel('Prijem_Celkovy')
P=plt.ylabel('%')
```



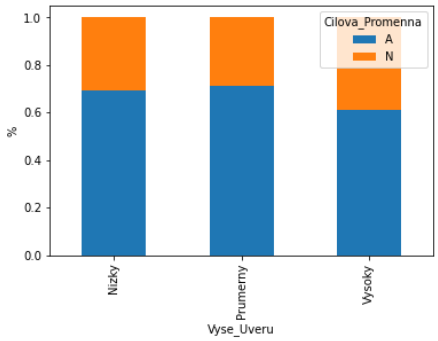
Zdroj: Vlastní zpracování

Jak lze vidět proporce kvalitních žádostí u nízkého celkového příjmu je velmi odlišná od předchozího grafu.

Dále budeme pozorovat vliv výše úvěru na cílovou proměnnou.

Obrázek 38 Rozdělení výše úvěrů do binů

```
bins=[0,100,200,700]
group=['Nizky','Prumerny','Vysoky']
train['Vyse_Uveru_bin']=pd.cut(train['Vyse_Uveru'],bins,labels=group)
Vyse_Uveru_bin=pd.crosstab(train['Vyse_Uveru_bin'],train['Cilova_Promenna'])
Vyse_Uveru_bin.div(Vyse_Uveru_bin.sum(1).astype(float), axis=0).plot(kind='bar',stacked=True)
plt.xlabel('Vyse_Uveru')
P=plt.ylabel('%')
```



Zdroj: Vlastní zpracování

Po rozdělení proměnné výše úvěru do binů nízký, průměrný a vysoký, lze vidět, že procento dobrých žádostí je u vysokých úvěrů pouze mírně vyšší.

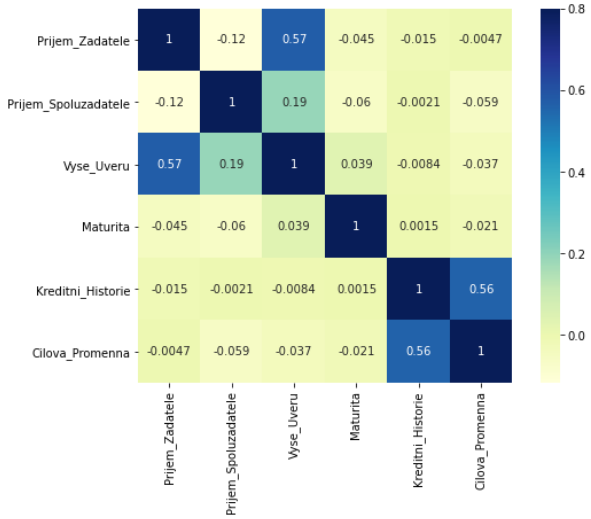
4.3.6 Korelační matice

K zobrazení vzájemných vztahů mezi číselnými proměnnými nám pomůže sestrojení korelační matice. Vizualizaci provedeme pomocí heat mapy. Heat mapy zobrazují data pomocí odstínování různých barev. V Python nám s tímto pomůže knihovna seaborn s funkcí „heatmap“. Parametr cmap upraví základní barvy podle volby. V našem případě volíme Žlutou – zelenou – Modrou heat mapu. Hodnoty s tmavší barvou ukazují vyšší korelaci.

Obrázek 39 Korelační matice

```
matrix = train.corr()
f, ax = plt.subplots(figsize=(9,6))
sns.heatmap(matrix,vmax=.8,square=True,cmap='YlGnBu', annot = True)
```

```
<AxesSubplot:>
```



Zdroj: Vlastní zpracování

Z heat mapy korelační matice lze vyčíst, že nejvíce korelují kombinace proměnných příjmů žadatele, výše úvěrů, ale také kreditní historie s cílovou proměnnou. Výše úvěru dále lehce koreluje s příjmem spolužadatele.

4.4 Čištění dat

Dostupná data nemusí být pokaždé dokonalá. Je proto potřeba data pročistit. Čištění může zahrnout identifikaci neúplných, nebo také nepřesných dat. V našem případě se budeme zabývat nahrazením chybějících null hodnot v datech, odchylkami v datech a přetvořením kategorických proměnných na dummy proměnné.

Nejprve v našem notebooku smažeme naše uměle vytvořené proměnné.

Obrázek 40 Smazání sloupců z datasetu pomocí funkce drop

```
train=train.drop(['Prijem_bin','Prijem_Spoluzadatele_bin','Prijem_Celkovy','Prijem_Celkovy_bin','Vyse_Uveru_bin'], axis=1)
```

Zdroj: Vlastní zpracování

Algoritmy jako logistická regrese dokážou pracovat pouze s číselnými hodnotami. Z tohoto důvodu budeme muset některé kategorické proměnné přetvořit na číselné.

Proměnná členové rodiny nabývá hodnot „0“, „1“, „2“, „3+“. Hodnoty „3+“ nahradíme hodnotou 3, abychom mohli s proměnnou dále pracovat jako s numerickou.

Obrázek 41 Nahrazení hodnot v proměnné Clenove_Rodiny

```
train['Clenove_Rodiny'].replace('3+', 3,inplace=True)
```

Zdroj: Vlastní zpracování

V dalším kroku přeměníme hodnoty A/N cílové proměnné na binární hodnoty 1/0

Obrázek 42 Nahrazení textových hodnot za číselné

```
train['Cilova_Promenna'].replace('N', 0,inplace=True)
train['Cilova_Promenna'].replace('A', 1,inplace=True)
```

Zdroj: Vlastní zpracování

4.4.1 Chybějící hodnoty

Jak již bylo zmíněno, poskytnutá data mohou být neúplná a některé proměnné mohou nabývat null hodnot. Funkcí isnull() zjistíme jejich četnosti.

Obrázek 43 Zobrazení chybějících hodnot v datech

```
train.isnull().sum()
```

```
Cislo_Zadosti           0
Pohlavi                13
Zenaty_Vdana            3
Clenove_Rodiny         15
Vzdelani                0
OSVC                   32
Prijem_Zadatele         0
Prijem_Spoluzadatele    0
Vyse_Uveru             22
Maturita               14
Kreditni_Historie      50
Mira_Urbanizace         0
Cilova_Promenna         0
dtype: int64
```

Zdroj: Vlastní zpracování

V našich datech se vyskytují prázdné hodnoty u proměnných Pohlavi, Clenove_Rodiny, OSVC, Vyse_Uveru, Maturita a Kreditni_Historie.

Prázdné hodnoty u proměnných číselného typu nahradíme průměrem. U kategorických proměnných tyto hodnoty nahradíme hodnotou modus.

Obrázek 44 Výskyt hodnot proměnné maturita

```
train['Maturita'].value_counts()
```

```
360.0    512
180.0     44
480.0     15
300.0     13
84.0       4
240.0      4
120.0      3
36.0       2
60.0       2
12.0       1
Name: Maturita, dtype: int64
```

Zdroj: Vlastní zpracování

Pokud se podíváme detailněji na hodnoty vyskytující se u proměnné Maturita, lze vidět, že hodnota 360 se vyskytuje suverénně nejčastěji. Můžeme tedy chybějící hodnoty nahradit touto proměnnou pomocí funkce modus. Stejné provedeme i u ostatních kategorických proměnných.

Kvantitativní proměnná výše úvěru chybí v datech v 22 případech. Jak již bylo zmíněno, u numerických hodnot lze chybějící hodnoty nahradit průměrem nebo mediánem. Z podrobnější analýzy této proměnné v předchozích kapitolách víme, že proměnná obsahuje velké odchylky hodnot a zvolení průměru jako náhrady chybějících hodnot by nebyla správná volba. Proto zvolíme medián.

Obrázek 45 Nahrazení hodnot u chybějících proměnných maturita a výše úvěru

```
train['Maturita'].fillna(train['Maturita'].mode()[0], inplace=True)
```

```
train['Vyse_Uveru'].fillna(train['Vyse_Uveru'].median(), inplace=True)
```

Zdroj: Vlastní zpracování

Stejný postup opakujeme u ostatních proměnných s chybějícími hodnotami:

Obrázek 46 Doplnění zbývajících hodnot

```
train['Pohlavi'].fillna(train['Pohlavi'].mode()[0], inplace=True)
train['Zenaty_Vdana'].fillna(train['Zenaty_Vdana'].mode()[0], inplace=True)
train['Clenove_Rodiny'].fillna(train['Clenove_Rodiny'].mode()[0], inplace=True)
train['OSVC'].fillna(train['OSVC'].mode()[0], inplace=True)
train['Kreditni_Historie'].fillna(train['Kreditni_Historie'].mode()[0], inplace=True)
```

Zdroj: Vlastní zpracování

Funkcí isnull můžeme pro kontrolu ještě jednou zjistit četnosti chybějících hodnot.

Obrázek 47 Finální kontrola chybějících hodnot

```
1 train.isnull().sum()
```

```
Cislo_Zadosti          0
Pohlavi                0
Zenaty_Vdana           0
Clenove_Rodiny         0
Vzdelani               0
OSVC                   0
Prijem_Zadatele        0
Prijem_Spoluzadatele   0
Vyse_Uveru             0
Maturita               0
Kreditni_Historie      0
Mira_Urbanizace        0
Cilova_Promenna        0
dtype: int64
```

Zdroj: Vlastní zpracování

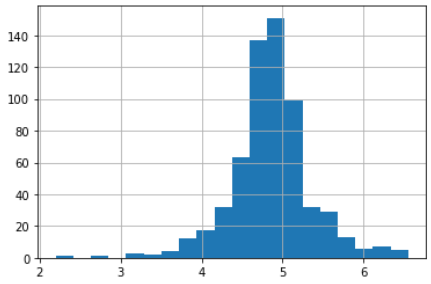
4.4.2 Odchylky hodnot

Z analýzy proměnné výše úvěru bylo vidět, že hodnoty této proměnné nemají normální rozdělení. Seskupení hodnot výše úvěru v levé části grafu se říká šikmost. Jeden ze způsobů, jak ošetřit šikmost proměnné, je logaritmická transformace. Logaritmus příliš neovlivní malé hodnoty, ale velmi zredukuje vysoké hodnoty. Proto se výsledná distribuce bude více podobat normálnímu rozdělení. Proměnnou Vyse_Uveru zlogaritmujeme funkcí log z knihovny numpy. Dále proměnnou vizualizujeme v histogramu.

Obrázek 48 Opravení distribuce výše úvěru

```
1 train['Vyse_Uveru_log']=np.log(train['Vyse_Uveru'])
2 train['Vyse_Uveru_log'].hist(bins=20)
```

```
<AxesSubplot:>
```



Zdroj: Vlastní zpracování

4.4.3 Dummy proměnné

Dalším potřebným krokem při přípravě dat je vytvoření dummy proměnných pro kategorické proměnné. Pomocí dummy proměnných přeměníme kategorické proměnné do proměnných obsahujících 1 a 0. Tato transformace na binární proměnné nám pomůže k jejich kvantifikaci a porovnání.

Jako příklad vezmeme proměnou Pohlaví, která nabývá hodnot muž, či žena. Algoritmy jako logistická regrese dokážou pracovat pouze s numerickými hodnotami, musíme tedy tuto proměnnou transformovat. Díky dummy proměnné můžeme vytvořit dvě nové proměnné „Pohlavi_Muz“ a „Pohlavi_Zena“.

Proměnná Pohlavi_Muz nabude hodnot 1, pokud pohlaví bude muž a hodnotu 0 pokud bude pohlaví žena.

K automatickému transformování kategorických proměnných na binární dummy proměnné lze použít funkci „get_dummies“ z knihovny pandas.

Obrázek 49 Vytvoření dummy proměnných

```
X = pd.get_dummies(X)
train=pd.get_dummies(train)
```

```
train.head()
```

	Prijem_Zadatele	Prijem_Spoluzadatele	Vyse_Uveru	Maturita	Kreditni_Historie	Cilova_Promenna	Vyse_Uveru_log	Pohlavi_Muz	Pohlavi_Zena	Zenaty_Vdana
0	5849	0.0	128.0	360.0	1.0	1	4.852030	1	0	
1	4583	1508.0	128.0	360.0	1.0	0	4.852030	1	0	
2	3000	0.0	66.0	360.0	1.0	1	4.189655	1	0	
3	2583	2358.0	120.0	360.0	1.0	1	4.787492	1	0	
4	6000	0.0	141.0	360.0	1.0	1	4.948760	1	0	

5 rows × 22 columns

Zdroj: Vlastní zpracování

4.5 Vytvoření modelu

K vytvoření modelů budeme používat open source Python knihovnu scikit-learn (sklearn). Knihovna obsahuje spoustu funkcí a modelů. Jedná se o jednu z nejlepších knihoven pro modelování v Python.

Před začátkem modelování je potřeba odstranit proměnnou Cislo_Zadosti, jelikož pro vytváření modelů nemá žádnou přidanou hodnotu.

Obrázek 50 Odstranění čísla žádosti

```
train=train.drop('Cislo_Zadosti',axis=1)
```

Zdroj: Vlastní zpracování

Knihovna sklearn požaduje, aby byla cílová proměnná oddělená v odlišné datové sadě. Vytvoříme proto dvě nové sady x a y. V datové sadě x budeme mít všechny nezávislé proměnné a v sadě y budeme mít pouze cílovou proměnnou.

Obrázek 51 Vytvoření nových datových sad

```
X = train.drop('Cilova_Promenna',1)
y = train.Cilova_Promenna
```

Zdroj: Vlastní zpracování

Nyní můžeme pokračovat ve vytváření modelu.

Po trénování modelu je potřeba predikce modelu validovat. Tohoto dosáhneme vytvořením splitu datové sady na dvě části: trénovací a validační.

Ke splitu naší datové sady použijeme funkci train_test_split z knihovny sklearn. Parametrem test_size určíme, že velikost validačního datasetu bude 30 % originálních dat.

Obrázek 52 Split dat

```
from sklearn.model_selection import train_test_split
x_train, x_cv, y_train, y_cv = train_test_split(X,y, test_size=0.3)
```

Zdroj: Vlastní zpracování

Po úpravě proměnných a přípravě trénovací a validační datové sady můžeme vytvořit první model.

Jako první model byla zvolena logistická regrese. Algoritmus logistické regrese je obsažen v knihovně sklearn. Dále jsme použili funkci accuracy_score, která nám popisuje přesnost našeho modelu.

Obrázek 53 První model logistické regrese

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
model = LogisticRegression()
model.fit(x_train, y_train)

pred_cv = model.predict(x_cv)
accuracy_score(y_cv,pred_cv)
acc_lr_1 = (accuracy_score(y_cv,pred_cv) * 100.0)

print("Přesnost: %.2f%%" % acc_lr_1)
```

```
Přesnost: 68.11%
```

Zdroj: Vlastní zpracování

Přesnost našeho prvního modelu je 68,11 %. Znamená to tedy, že náš model správně identifikoval 68 % cílových proměnných.

4.6 Logistická regrese za použití stratifikované křížové validace

K zajištění robustnosti modelu je potřeba provést validaci. Validace je technika, při níž se část datové sady rezervuje a nepoužívá se k trénování modelu. Po vytvoření modelu se tato rezervovaná část použije pro testování přesnosti modelu.

V předchozí kapitole jsme použili metodu validačního setu pomocí funkce train_test_split z knihovny sklearn. Tato prostá metoda vybere x % náhodných dat, která rezervuje pro validaci.

Metod pro validaci dat existuje mnoho. Pro účely této práce byla vybrána metoda stratifikované křížové validace k-fold (angl. stratified k-folds cross-validation).

Při stratifikované křížové validaci k-fold jsou oddíly vybrány tak, aby střední hodnota odezvy byla přibližně stejná ve všech oddílech. V případě naší binární klasifikace (cílová proměnná nabývá hodnot 1/0) to znamená, že každý oddíl obsahuje zhruba stejné proporce cílové proměnné.

V opakované křížové validaci jsou data náhodně rozdělena do k-oddílů několikrát a výkon modelu tak lze průměrovat. Pro účely práce bylo vybráno k = 5, tedy model použije 5 náhodných oddílů validačních dat.

K stratifikované křížové validaci k-fold použijeme funkci StratifiedKFold z knihovny sklearn.

Obrázek 54 Import funkce StratifiedKfold

```
1 from sklearn.model_selection import StratifiedKFold
```

Zdroj: Vlastní zpracování

Kód pro logistickou regresi upravíme pomocí klasického „for cyklu“, aby se opakoval pro každý oddíl stratifikace zvlášť:

Obrázek 55 Logistická regrese za použití stratifikace

```
i=1
mean = 0
kf = StratifiedKFold(n_splits=5,random_state=1)
for train_index,test_index in kf.split(X,y):
    print('[Fold %d/%d]' % (i, kf.n_splits))
    xtr,xvl = X.loc[train_index],X.loc[test_index]
    ytr,yvl = y[train_index],y[test_index]
    model = LogisticRegression(random_state=1)
    model.fit(xtr,ytr)
    pred_test=model.predict(xvl)
    score=accuracy_score(yvl,pred_test)
    mean += score
    print("Přesnost: %.2f%%" % (score * 100.0))
    i+=1
    pred = model.predict_proba(xvl)[:,1]
print("\n Průměrná přesnost: %.2f%%" % (mean/(i-1) * 100.0))
acc_lr_2 = (mean/(i-1) * 100.0)
```

```
[Fold 1/5]
Přesnost: 69.11%
[Fold 2/5]
Přesnost: 78.05%
[Fold 3/5]
Přesnost: 78.05%
[Fold 4/5]
Přesnost: 69.11%
[Fold 5/5]
Přesnost: 81.15%

 Průměrná přesnost: 75.09%
```

Zdroj: Vlastní zpracování

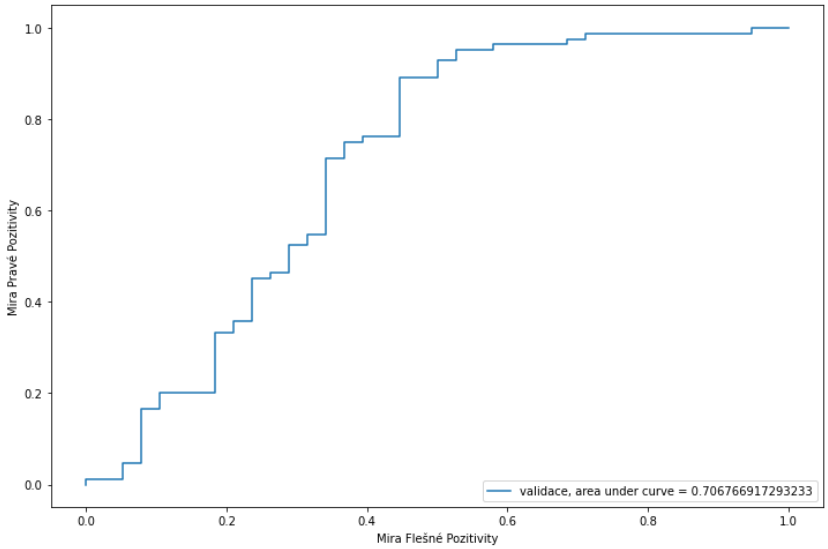
Model logistické regrese má po 5 opakováních průměrnou přesnost 75 %.

Kvalitu našeho modelu dál vyhodnotíme pomocí ROC křivky (z anglického Receiver Operating Characteristic, operační charakteristika přijímače). ROC křivka je graf, který popisuje kvalitu naší binární klasifikace. Na základě počtu případů pravé a falešné pozitivity vyhodnotí pravděpodobnost detekce (true positive rate) a pravděpodobnost falešného poplachu (flase positive rate).

Metriku ROC křivku naimportujeme z knihovny sklearn a aplikujeme na předchozí model. Výsledky vizualizujeme v grafu:

Obrázek 56 ROC křivka

```
from sklearn import metrics
fpr, tpr, _ = metrics.roc_curve(yvl, pred)
auc = metrics.roc_auc_score(yvl, pred)
plt.figure(figsize=(12,8))
plt.plot(fpr, tpr, label='validace, area under curve = '+str(auc))
plt.xlabel('Mira Flešné Pozitivity')
plt.ylabel('Mira Pravé Pozitivity')
plt.legend(loc=4)
plt.show()
```



Zdroj: Vlastní zpracování

Agregovaná metrika ROC křivky je hodnota AUC (Area under the ROC Curve, plocha pod křivkou ROC). AUC nabývá hodnot od 0 do 1, kdy 0 znamená, že má model 100 % predikcí špatných a 1 znamená, že má model 100 % predikcí správných.

4.7 Hledání a příprava proměnných

Anglicky feature engineering je proces využití znalosti k extrakci features (nových proměnných) z dat. Features by měly být jednodušší na pochopení v kontextu problému. Dále feature hraje roli v ovlivnění výsledků našeho prediktivního modelu.

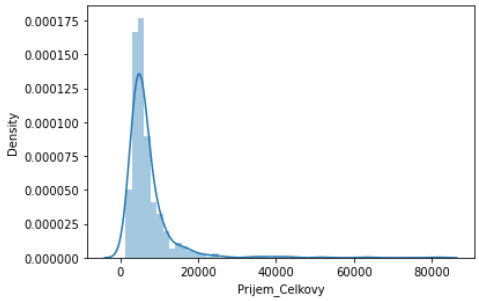
Jak již bylo zmíněno v analýze dat, první features proměnnou, kterou použijeme, bude proměnná Celkovy_Prijem. Jedná se o prostý součet příjmů žadatele a spolužadatele.

Obrázek 57 Proměnná celkový příjem

```
train['Prijem_Celkovy']=train['Prijem_Zadatele']+train['Prijem_Spoluzadatele']
```

```
sns.distplot(train['Prijem_Celkovy'])
```

```
<AxesSubplot:xlabel='Prijem_Celkovy', ylabel='Density'>
```



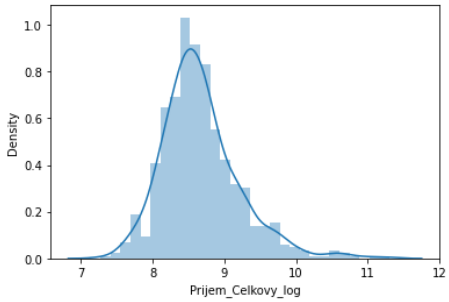
Zdroj: Vlastní zpracování

Z grafu lze vidět, že je distribuce naší nové proměnné šikmá doprava. Proměnnou proto přeměníme logaritmickou transformací, aby měla normální rozdělení.

Obrázek 58 Logaritmická transformace

```
train['Prijem_Celkovy_log'] = np.log(train['Prijem_Celkovy'])
sns.distplot(train['Prijem_Celkovy_log'])
```

```
<AxesSubplot:xlabel='Prijem_Celkovy_log', ylabel='Density'>
```



Zdroj: Vlastní zpracování

Logaritmická transformace výrazně snížila extrémní hodnoty a distribuce má nyní normální rozdělení.

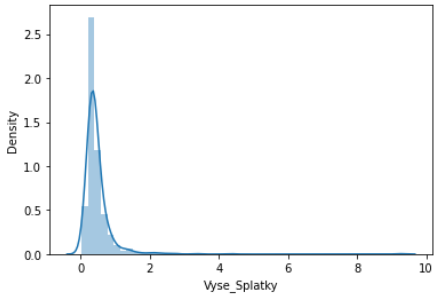
Další novou proměnnou je výška splátky. Výši splátky spočítáme pouze jako prostý poměr velikosti úvěru a počtu měsíců maturity.

Obrázek 59 Proměnná Vyse Splatky

```
train['Vyse_Splatky']=train['Vyse_Uveru']/train['Maturita']
```

```
sns.distplot(train['Vyse_Splatky'])
```

```
<AxesSubplot:xlabel='Vyse_Splatky', ylabel='Density'>
```



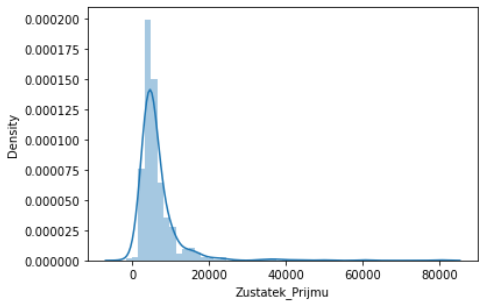
Zdroj: Vlastní zpracování

Poslední novou proměnnou je zůstatek příjmu. Po výpočtu výšky splátky můžeme tuto proměnnou odečíst od celkového příjmu. Důvodem vytvoření této proměnné je předpoklad, že pokud jedinci zůstane víc peněz každý měsíc, bude větší pravděpodobnost, že splátku znova splatí. Výše splátky byla v tisících, proto ji musíme vynásobit 1000, aby se pracovalo se stejnými jednotkami.

Obrázek 60 Proměnná Zustatek Prijmu

```
train['Zustatek_Prijmu'] = train['Prijem_Celkovy']-(train['Vyse_Splatky']*1000)
sns.distplot(train['Zustatek_Prijmu'])
```

```
<AxesSubplot:xlabel='Zustatek_Prijmu', ylabel='Density'>
```



Zdroj: Vlastní zpracování

Logistická regrese předpokládá, že korelace mezi proměnnými není příliš vysoká. Korelace mezi novými a starými proměnnými je vysoká. Z tohoto důvodu musíme staré proměnné, ze kterých jsme vytvořili nové, smazat z datové sady.

Obrázek 61 Smazání korelovaných proměnných

```
train=train.drop(['Prijem_Zadatele', 'Prijem_Spoluzadatele', 'Vyse_Uveru', 'Maturita'], axis=1)
```

Zdroj: Vlastní zpracování

4.8 Logistická regrese

Po vytvoření nových proměnných můžeme pokračovat ve vytváření modelů. Jak již bylo zmíněno v teoretické části práce, pro účely diplomové práce byly vybrány algoritmy od jednodušších, jako je logistická regrese a rozhodovací stromy, po komplexnější, jako je algoritmus náhodných lesů, gradient boosting a support vector machines.

Prvním algoritmem je logistická regrese. Logistickou regresi pustíme stejným způsobem jako v předchozí kapitole. Nyní nicméně pracujeme již s features proměnnými, a přesnost proto může vyjít jinak.

Obrázek 62 Logistická regrese

```
i=1
mean = 0
kf = StratifiedKFold(n_splits=5,random_state=1,shuffle=True)
for train_index,test_index in kf.split(X,y):
    print('[Fold %d/%d]' % (i, kf.n_splits))
    xtr,xvl = X.loc[train_index],X.loc[test_index]
    ytr,yvl = y[train_index],y[test_index]
    model = LogisticRegression(random_state=1)
    model.fit(xtr,ytr)
    pred_test=model.predict(xvl)
    score=accuracy_score(yvl,pred_test)
    mean += score
    print("Přesnost: %.2f%%" % (score * 100.0))
    i+=1
print("\n Průměrná přesnost: %.2f%%" % (mean/(i-1) * 100.0))
acc_lr_3 = (mean/(i-1) * 100.0)
```

```
[Fold 1/5]
Přesnost: 78.86%
[Fold 2/5]
Přesnost: 69.11%
[Fold 3/5]
Přesnost: 66.67%
[Fold 4/5]
Přesnost: 78.05%
[Fold 5/5]
Přesnost: 68.03%

 Průměrná přesnost: 72.14%
```

Zdroj: Vlastní zpracování

Průměrná přesnost našeho modelu logistické regrese při použití stratifikační křížové validace a za použití features proměnných je 72,14 %.

4.9 Decision Tree

Rozhodovací strom je metoda učení s učitelem, což znamená, že máme předem definovanou cílovou proměnnou. V tomto modelu se data rozdělí do dvou a více homogenních skupin dat v závislosti na rozdělení pomocí proměnné.

Decison tree algoritmus naimportujeme z knihovny sklearn a spustíme funkci DecisonTreeClassifier. Stejně jako u logistické regrese nadále použijeme k validaci stratifikaci k-fold, kde k = 5.

Obrázek 63 Decision tree

```
from sklearn import tree
i=1
mean = 0
kf = StratifiedKFold(n_splits=5,random_state=1,shuffle=True)
for train_index,test_index in kf.split(X,y):
    print('[Fold %d/%d]' % (i, kf.n_splits))
    xtr,xvl = X.loc[train_index],X.loc[test_index]
    ytr,yvl = y[train_index],y[test_index]
    model = tree.DecisionTreeClassifier(random_state=1)
    model.fit(xtr,ytr)
    pred_test=model.predict(xvl)
    score=accuracy_score(yvl,pred_test)
    mean += score
    print("Přesnost: %.2f%%" % (score * 100.0))
    i+=1
print("\n Průměrná přesnost: %.2f%%" % (mean/(i-1) * 100.0))
acc_dt = (mean/(i-1) * 100.0)
```

```
[Fold 1/5]
Přesnost: 74.80%
[Fold 2/5]
Přesnost: 71.54%
[Fold 3/5]
Přesnost: 72.36%
[Fold 4/5]
Přesnost: 71.54%
[Fold 5/5]
Přesnost: 67.21%

 Průměrná přesnost: 71.49%
```

Zdroj: Vlastní zpracování

Průměrná přesnost modelu decison tree je v našem případě 71,49 %.

4.10 Random Forest

Random forest je již komplexnější metoda, která používá množství několika rozhodovacích stromů k vytvoření jednoho prediktivního modelu. Výsledný prediktivní model je funkcí všech použitých rozhodovacích stromů.

Obrázek 64 Random Forest

```
from sklearn.ensemble import RandomForestClassifier
i=1
mean = 0
kf = StratifiedKFold(n_splits=5,random_state=1,shuffle=True)
for train_index,test_index in kf.split(X,y):
    print('[Fold %d/%d]' % (i, kf.n_splits))
    xtr,xvl = X.loc[train_index],X.loc[test_index]
    ytr,yvl = y[train_index],y[test_index]
    model = RandomForestClassifier(random_state=1, max_depth=10)
    model.fit(xtr,ytr)
    pred_test=model.predict(xvl)
    score=accuracy_score(yvl,pred_test)
    mean += score
    print("Přesnost: %.2f%%" % (score * 100.0))
    i+=1
print("\n Průměrná přesnost: %.2f%%" % (mean/(i-1) * 100.0))
acc_rf = (mean/(i-1) * 100.0)
```

```
[Fold 1/5]
Přesnost: 82.93%
[Fold 2/5]
Přesnost: 81.30%
[Fold 3/5]
Přesnost: 79.67%
[Fold 4/5]
Přesnost: 78.86%
[Fold 5/5]
Přesnost: 76.23%

 Průměrná přesnost: 79.80%
```

Zdroj: Vlastní zpracování

Průměrná přesnost modelu random forest je u našich dat 71,49 %.

Jak již bylo zmíněno, algoritmus random forest je komplexnější a je možné ho jistým způsobem ladit. Ladění algoritmu se provádí pomocí tzv. hyper parametrů.

V této práci se zaměříme pouze na parametry max_depth (maximální hloubka stromů) a n_estimators (počet stromů).

Ladění hyper parametrů můžeme provádět manuálně podle intuice, nebo automaticky pomocí grid search metody. Grid sreach metoda postupně vyzkouší zadané kombinace parametrů a vybere nejpřesnější kombinaci.

Nevýhodou této automatické metody je náročnost na výpočetní výkon. Kombinování několika možností parametrů je výpočetně náročné, a i při použití pouze cca 600 řádků dat může trvat několik minut k nalezení optimálních parametrů.

Obrázek 65 Grid search

```
from sklearn.model_selection import GridSearchCV
paramgrid = {'max_depth': list(range(1,20,2)), 'n_estimators': list(range(1,200,20))}
grid_search=GridSearchCV(RandomForestClassifier(random_state=1),paramgrid)
from sklearn.model_selection import train_test_split
x_train, x_cv, y_train, y_cv = train_test_split(X,y, test_size=0.3, random_state=1)
grid_search.fit(x_train,y_train)
GridSearchCV(estimator=RandomForestClassifier(random_state=1),
             param_grid={'max_depth': [1, 3, 5, 7, 9, 11, 13, 15, 17, 19],
                         'n_estimators': [1, 21, 41, 61, 81, 101, 121, 141, 161,
                                          181]})
grid_search.best_estimator_
```

```
RandomForestClassifier(max_depth=5, n_estimators=41, random_state=1)
```

Zdroj: Vlastní zpracování

Meto grid-search našla jako optimální kombinaci parametrů max_depth=5 a n_estimators=41. Doporučuje nám tedy pracovat pouze s maximální hloubkou stromů 5 a celkovým počtem stromů 41.

Parametry nyní aplikujeme do funkce RandomForestClassifier a pustíme algoritmus random tree znovu:

Obrázek 66 Random forest po ladění hyperparametrů

```
from sklearn.ensemble import RandomForestClassifier
i=1
mean = 0
kf = StratifiedKFold(n_splits=5,random_state=1,shuffle=True)
for train_index,test_index in kf.split(X,y):
    print('[Fold %d/%d]' % (i, kf.n_splits))
    xtr,xvl = X.loc[train_index],X.loc[test_index]
    ytr,yvl = y[train_index],y[test_index]
    model = RandomForestClassifier(max_depth=5, n_estimators=41, random_state=1)
    model.fit(xtr,ytr)
    pred_test=model.predict(xvl)
    score=accuracy_score(yvl,pred_test)
    mean += score
    print("Přesnost: %.2f%%" % (score * 100.0))
    i+=1
print("\n Průměrná přesnost: %.2f%%" % (mean/(i-1) * 100.0))
acc_rf2 = (mean/(i-1) * 100.0)
```

```
[Fold 1/5]
Přesnost: 82.11%
[Fold 2/5]
Přesnost: 83.74%
[Fold 3/5]
Přesnost: 78.05%
[Fold 4/5]
Přesnost: 78.86%
[Fold 5/5]
Přesnost: 79.51%

 Průměrná přesnost: 80.45%
```

Zdroj: Vlastní zpracování

Jak lze vidět, průměrná přesnost stoupla na 80,45 %.

4.11 XGBoost

XGBoost je algoritmus, který v poslední době dominuje na poli aplikovaného strojového učení a v soutěžích zaměřených na strukturovaná nebo tabulární data (Browlee, 2016).

XGBoost je implementace založena na rozhodovacích stromech gradient boostingu. Výhoda algoritmu je jeho rychlost, ale i přesnost výsledných modelů.

K validaci modelů bude nadále použita křížová validace k-fold. Algoritmus XGBoost naimportujeme z nainstalované knihovny XGBoost. Modelace proběhne pomocí funkce XGBClassifier.

Obrázek 67 XGBoost

```
from xgboost import XGBClassifier
i=1
mean = 0
kf = StratifiedKFold(n_splits=5,random_state=1,shuffle=True)
for train_index,test_index in kf.split(X,y):
    print('[Fold %d/%d]' % (i, kf.n_splits))
    xtr,xvl = X.loc[train_index],X.loc[test_index]
    ytr,yvl = y[train_index],y[test_index]
    xgb_params = {
    'objective': 'binary:logistic'
    }
    model = XGBClassifier(**xgb_params)
    model.fit(xtr, ytr, eval_metric='logloss')
    pred_test = model.predict(xvl)
    score = accuracy_score(yvl,pred_test)
    mean += score
    print("Přesnost: %.2f%%" % (score * 100.0))
    i+=1
print("\n Průměrná přesnost: %.2f%%" % (mean/(i-1) * 100.0))
acc_xgb_1 = (mean/(i-1) * 100.0)
```

```
[Fold 1/5]
Přesnost: 78.86%
[Fold 2/5]
Přesnost: 74.80%
[Fold 3/5]
Přesnost: 75.61%
[Fold 4/5]
Přesnost: 75.61%
[Fold 5/5]
Přesnost: 72.95%

 Průměrná přesnost: 75.57%
```

Zdroj: Vlastní zpracování

Průměrná přesnost našeho prvního modelu xgboost je 75,57 %.

Stejně jako u metody Random Forest, xgboost nabízí možnost ladění velkého množství hyperparametrů. Parametry mohou být opět optimalizované za účelem dosažení nejvyšší přesnosti, nicméně kvůli výpočetní náročnosti byly parametry optimalizované pouze manuálně. Pro účely práce bylo vybráno 5 hyper parametrů:

learning_rate: Kontrola vah u nově přidaných stromů do modelu. Výchozí hodnota parametru je 0,3, v následujícím modelu použita hodnota 0,01

n_estimators: Počet stromů v lese. Výchozí hodnota parametru je 100, v následujícím modelu bude použita hodnota 1000.

max_depth: Maximální hloubka stromů. Výchozí hodnota parametru není stanovená, v následujícím modelu použita hodnota 3.

gamma: Minimální redukce ztráty požadovaná pro další dělení stromu. Výchozí hodnota parametru je 0, v následujícím modelu bude použita hodnota 5.

min_child_weight: Minimální suma vah uzlů. Pokud rozvětvení stromu bude mít součet vah nižší než tento parametr, nebude se strom dál dělit. Výchozí hodnota parametru je 1, v následujícím modelu bude použita hodnota 2.

Po zadání vybraných parametrů do funkce XGBClassifer můžeme model pustit znovu:

Obrázek 68 Model XGBoost po přidání parametrů

```
from xgboost import XGBClassifier
i=1
mean = 0
kf = StratifiedKFold(n_splits=5,random_state=1,shuffle=True)
for train_index,test_index in kf.split(X,y):
    print('[Fold %d/%d]' % (i, kf.n_splits))
    xtr,xvl = X.loc[train_index],X.loc[test_index]
    ytr,yvl = y[train_index],y[test_index]
    xgb_params = {
    'objective': 'binary:logistic'
    ,'learning_rate': 0.01
    ,'n_estimators': 1000
    ,'max_depth': 3
    ,'gamma': 5
    ,'min_child_weight': 2
    }
    model = XGBClassifier(**xgb_params)
    model.fit(xtr, ytr, eval_metric='logloss',verbose=True)
    pred_test = model.predict(xvl)
    score = accuracy_score(yvl,pred_test)
    mean += score
    print("Přesnost: %.2f%%" % (score * 100.0))
    i+=1
print("\n Průměrná přesnost: %.2f%%" % (mean/(i-1) * 100.0))
acc_xgb_2 = (mean/(i-1) * 100.0)
```

```
[Fold 1/5]
Přesnost: 81.30%
[Fold 2/5]
Přesnost: 83.74%
[Fold 3/5]
Přesnost: 79.67%
[Fold 4/5]
Přesnost: 80.49%
[Fold 5/5]
Přesnost: 79.51%

 Průměrná přesnost: 80.94%
```

Zdroj: Vlastní zpracování

Průměrná přesnost modelu XGBoost je po úpravě vstupních parametrů modelu 80,94 %.

4.12 Support vector machines

Posledním vybraným modelem je Support vector machines (SVM) neboli metoda podpůrných vektorů. Validace bude nadále provedena křížovou validací k-fold (k = 5).

Tuto metodu strojového učení importujeme z knihovny sklearn a pustíme funkci SVC.

Obrázek 69 Model SVM

```
from sklearn import svm
i=1
mean = 0
kf = StratifiedKFold(n_splits=5,random_state=1,shuffle=True)
for train_index,test_index in kf.split(X,y):
    print('[Fold %d/%d]' % (i, kf.n_splits))
    xtr,xvl = X.loc[train_index],X.loc[test_index]
    ytr,yvl = y[train_index],y[test_index]
    model = svm.SVC()
    model.fit(xtr,ytr)
    pred_test=model.predict(xvl)
    score=accuracy_score(yvl,pred_test)
    mean += score
    print("Přesnost: %.2f%%" % (score * 100.0))
    i+=1
print("\n Průměrná přesnost: %.2f%%" % (mean/(i-1) * 100.0))
acc_svm = (mean/(i-1) * 100.0)
```

```
[Fold 1/5]
Přesnost: 69.11%
[Fold 2/5]
Přesnost: 69.11%
[Fold 3/5]
Přesnost: 68.29%
[Fold 4/5]
Přesnost: 68.29%
[Fold 5/5]
Přesnost: 67.21%

 Průměrná přesnost: 68.40%
```

Zdroj: Vlastní zpracování

Průměrná přesnost našeho modelu SVM je 68,40 %.

5 Výsledky a diskuse

5.1 Významnost proměnných

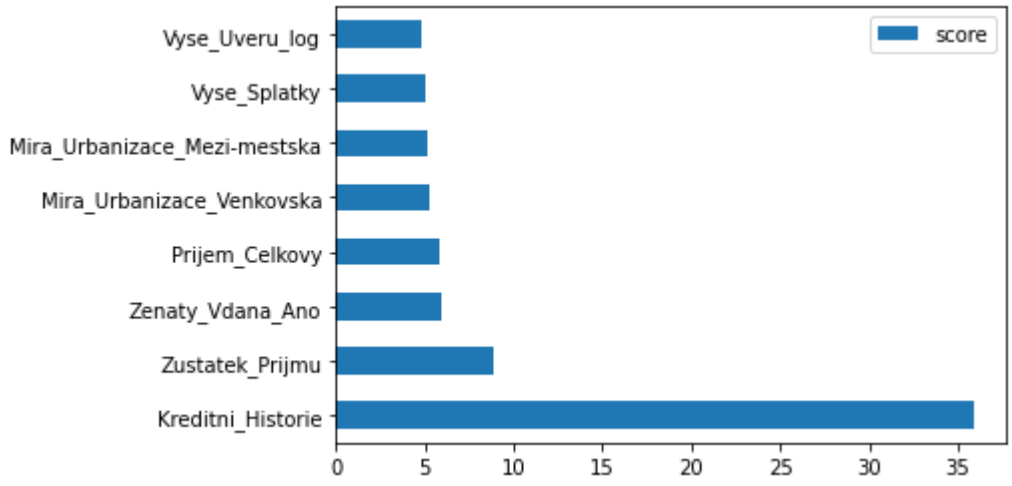
Největší výzvou při používání modelů strojového učení je jejich interpretovatelnost. Schopnost porozumět, jak modely fungují, je velmi důležitá a v případech, kdy modely dělají business rozhodnutí, je jejich interpretovatelnost black-box algoritmu kritická.

Modely strojového učení již v praxi předvedly svoji schopnost přinést přesnější možnosti vytváření predikcí. V současné době se při jejich vývoji klade větší důraz, aby se modely strojového učení daly zároveň i pochopit. Jedním takovým nástrojem pro interpretaci výsledků je nástroj Feature Importance**.**

Feature Importance je graf, který zobrazuje relativní důležitost proměnných v modelu. Nicméně existuje několik metod vypočtení „důležitosti“ proměnné. Pro metody založené na stromech se nejčastěji používá průměrné zlepšení, celkové zlepšení, počet štěpení stromů a Shapley Additive Explanations (SHAP).

Výchozí metoda pro hodnocení modelů XGBoost je metoda průměrného zlepšení. Tuto metodu aplikujeme na náš nejúspěšnější model XGBoost a získáme graf významnosti proměnných.

Obrázek 70 Feature importance gain



Zdroj: Vlastní zpracování

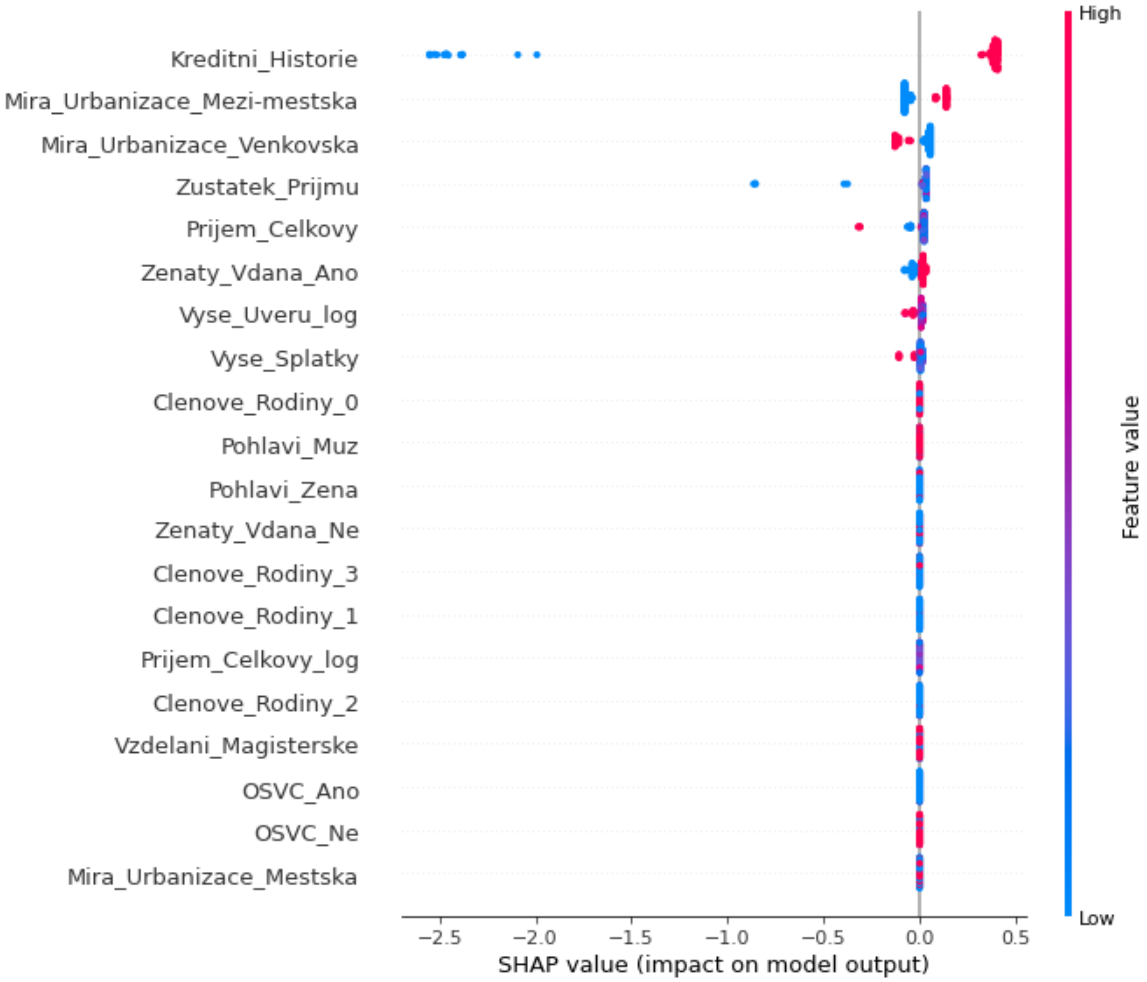
Z grafu lze vidět jednoznačná důležitost proměnné kreditní historie. Takto vysoká důležitost odpovídá očekávání, jelikož historická zkušenost instituce s klientem je velice cenná z hlediska kreditního rizika. Dále je vidět, že naše nové proměnné Zustatek_Prijmu, Prijem_Celkovy a Vyse_Splatky jsou z hlediska průměrného přínosu modelu také důležité.

Znamená to tedy, že nám část features engineering pomohla při odhadování cílové proměnné.

Problém metody features importance je, že přínosnost proměnných se u různých pohledů liší a je na nás odhadnout, zda použijeme zrovna pohled průměrného, nebo celkového zlepšení.

K lepšímu měření důležitosti proměnných se používá nástroj Shapley Additive Explanations (SHAP). SHAP se při vyhodnocování důležitosti proměnných zaměřuje zejména na jejich konzistenci a přesnost. Celkovou důležitost proměnné lze v grafu vyčíst díky šířce rozpětí proměnné. Jak proměnná ovlivňuje predikce modelu, lze poznat podle barvy. Červená barva znamená, že proměnná obecně zvyšuje přesnost predikce modelu.

Obrázek 71 Feature importance SHAP



Zdroj: Vlastní zpracování

Z grafu lze vidět, že kreditní historie klienta má vysoký a pozitivní vliv na schopnost klienta splácet.

5.2 Srovnání výsledků přesností algoritmů

V Jupyter Notebook jsme celkem použili 9 variací metod. Výsledná tabulka ukazuje průměrné přesnosti modelu predikovat cílové proměnné u každé metody.

Tabulka 2 Tabulka výsledných přesností

Metoda	**Přesnost**	**AUC**
Logistická regrese (stratified k-fold cross validation) + feature engineering	72.14 %	56.27 %
Decision Tree	71.49 %	61.78 %
Random Forest	79.80 %	63.82 %
Random Forest + Grid Search hyperparametr tuning	80.45 %	64.54 %
XGBoost	75.57 %	62.99 %
XGBoost + hyperparametr tuning	80.94 %	65.55 %
Support Vector Machines	68.40 %	58.21 %

Zdroj: Vlastní zpracování

Z tabulky výsledných přesností je zřejmé, že přesnost modelu stoupá s jeho komplexností. Nejnižší přesnosti v našem případě dosáhla metoda Support Vector Machines s přesností 68,4 %. Jednodušší algoritmy jako logistická regrese a decison tree u nás dosáhly pouze přesnosti kolem 72 %.

S vyšší přesností se můžeme setkat u náročnějších algoritmů, jako je Random Forest, která dosáhla přesnosti 79,80 %, po ladění hyperparametrů pomocí algoritmu grid-search se přesnost zvýšila o dalších 0,65 % na 80,45 %.

Poslední a nejkomplexnější použitá metoda byla metoda XGBoost. První model této metody dosáhl přesnosti 75,57 %, což je nižší než metoda Random Forest. Druhý model algoritmu XGBoost byl po ladění hyperparametrů suverénně nejúspěšnější, co se přesnosti modelu týče, s přesností 80,94 %.

Je potřeba zmínit, že ladění parametrů u modelu XGBoost bylo pouze manuální podle uvážení a že při dostatečném výpočetním výkonu a použití algoritmu grid-search optimalizace by se mohla přesnost modelu ještě zvýšit.

Diagnostickou sílu testů měříme pomocí hodnoty AUC, tedy plochou pod křivkou ROC. I podle této metriky, měřicí poměry skutečně pozitivních a skutečně negativních pozorování, jsou algoritmy Random Forest a XGBoost favority.

5.3 Přínosy realizace v Jupyteru

Jupyter Notebooky excelují v prezentaci vytvořené práce, jelikož ukazují jak komentáře pomocí Markdown buněk, tak kód a výstupy kódu. V případě sdílení modelů

a analýz je pro člověka mnohem jednodušší porozumět analýze, pokud je schopen si postupně pustit kód buňku po buňce.

Dalším přínosem je bezpečnost. Jupyter Notebooky lze ukládat na serveru, který může být mnohem lépe zabezpečen, nemusí být tedy žádná data uložená na lokálních přístrojích. Dále je také v případě server-side noteboocích možné využít výpočetního výkonu serveru.

6 Závěr

Statistické modely zvyšují přesnost a efektivitu rozhodování v kreditním riziku. Finanční instituce si často myslí, že zavedení kreditního skórování je příliš obtížné a nákladné, nebo že nemají dostatek dat pro jeho implementaci. Nicméně nejdůležitějším nejmenším vstupem pro modely kreditního skórování je již dostupná historie splácení klienta.

Ke kreditnímu skórování lze použít mnoho statistických modelů. Při výběru modelů je potřeba zohlednit náklady na implementaci. Náklady zahrnují náklady na výpočetní techniku (databáze a servery pro výpočet modelů) až po náklady na analytické pozice potřebné pro vývoj modelů, jako jsou například datoví analytici nebo matematici.

Základem všech modelů je správná datová základna z plně integrovaných primárních systémů, zachycujících data v reálném čase. Tato data je potřeba správně uchovávat v databázích a následně správně analyzovat.

Finanční instituce, které nemají příliš mnoho zkušeností s datovou analýzou mohou začít s jednoduššími modely a postupně je odladit k jejich potřebám. Větší instituce mohou pro dosažení přesnějších výsledků při kreditním skórování použít komplexnější algoritmy, jako je random forest nebo XGBoost. Správným laděním těchto algoritmů mohou dosáhnout opravdu dobrých výsledků, co se přesnosti týče.

Nicméně je potřeba zohlednit vysokou náročnost na výpočetní výkon. V práci použitá datová sada obsahovala pouze 600 řádků pozorování a při použití metody grid-search trvalo hledání optimálních parametrů několik minut. V praxi se při vytváření modelů používají datové sady v řádech stovek tisíc pozorování.

Dále je potřeba zohlednit náklady na personalistiku. K dosažení nejlepších výsledků nestačí pouze vypočtený výkon, ale i kvalitní team analytiků a matematiků. Důkazem náročnosti na personalistiku, ale i přínosnosti na přesnosti vytvořených modelů, je vysoká popularita soutěží zaměřených na hledání nejlepšího algoritmu v kreditním rozhodování. Jedna taková soutěž byla soutěž pořádána mezinárodní bankou Home Credit, kdy tato banka poskytla veřejně hned několik velice podrobných a obsáhlých anonymizovaných datových sad o svých klientech, žádostech a historii splácení (základní datová sada čítala 122 proměnných a téměř půl milionu pozorování). Nejúspěšnější 6členný tým získal jako výhru peněžní odměnu $70.000.

V poslední řadě je při použití modelů strojového učení potřeba, aby si každá instituce ověřila soulad s pravidly regulátorů. Složitá interpretovatelnost black-box modelů nemusí být pokaždé v souladu s požadovanou férovostí a srozumitelností, co se týče problematiky automatického rozhodování a profilování.

Ať už se instituce při vybírání modelů kreditního rozhodování rozhodne pro jakýkoliv algoritmus, vždy bude přínosné mít celkový vývoj analýzy a modelů dostupný v jednom souvislém dokumentu, proto je v současné době Jupyter Notebook „industry-standard“ v oblasti data science.

7 Seznam použitých zdrojů

ABILASH, R, 2018. APPLYING RANDOM FOREST (CLASSIFICATION) — MACHINE LEARNING ALGORITHM FROM SCRATCH WITH REAL DATASETS. In: *Medium.com* [online]. 31. 7. 2018 [cit. 2021-03-03]. Dostupné z: https://medium.com/@ar.ingenious/applying-random-forest-classification-machine-learning-algorithm-from-scratch-with-real-24ff198a1c57

ALPAYDIN, E., 2020. *Introduction to Machine Learning.* 4th ed. Cambridge: MIT Press. ISBN 9780262043793.

ANACONDA, 2021. Anaconda Documentation. *Anaconda.com* [online]. © 2019 [cit. 2021-03-03]. Dostupné z: https://docs.anaconda.com/

ANAHITA, N. et al., 2018. Credit risk prediction in an imbalanced social lending environment. In: *Arxiv.org* [online]. © 2018 [cit. 2021-03-03]. Dostupné z: https://arxiv.org/ftp/arxiv/papers/1805/1805.00801.pdf

ANALYST PREP, 2019. Credit Scoring and Retail Credit Risk Management. *Analystprep.com* [online]. © 2019 [cit. 2021-03-03]. Dostupné z: https://analystprep.com/study-notes/frm/part-2/credit-risk-measurement-and-management/credit-scoring-and-retail-credit-risk-management/.

ANALYTICS VIDHYA, 2016. Loan Prediction Practice Problem. *Datahack.analyticsvidhya.com* [online]. © 2016 [cit. 2021-03-03]. Dostupné z: https://datahack.analyticsvidhya.com/contest/practice-problem-loan-prediction-iii/#data_dictionary

BAESENS, B, et al., 2003. Benchmarking State-of-the-Art Classification Algorithms for Credit Scoring. *The Journal of the Operational Research Society.* Vol. 54, No. 6, p. 9.

BANK FOR INTERNATIONAL SETTLEMENTS, 2000. *Principles for the Management of Credit Risk* [online]. Basel: Basel Committee on Banking Supervision [cit. 2021-03-03]. Dostupné z: https://www.bis.org/publ/bcbs75.htm.

BANK OF AMERICA, 2020. 5 C's of Credit: What Are Banks Looking For?. *Bankofamerica.com* [online]. © 2020 [cit. 2021-03-03]. Dostupné z: https://www.bankofamerica.com/smallbusiness/business-financing/learn/5-cs-of-credit/

BATHAEE, Y., 2018. THE ARTIFICIAL INTELLIGENCE BLACK BOX AND THE FAILURE OF INTENT AND CAUSATION. *Harvard Journal of Law & Technology*. p. 900.

BROWLEE, J., 2016. A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning. In: *Machinelearningmastery.com* [online]. 17. 2. 2021 [cit. 2021-03-03]. Dostupné z: https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/.

Brownlee, Jason. 2021. A Gentle Introduction to XGBoost for Applied Machine Learning. [Online] 2021. https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

CGAP/WORLD BANK, 2019. *Credit Scoring In Financial Inclusion.* Washington DC: Consultative Group to Assist the Poor, 2019.

DMLC, 2020. Awsome XGBoost. In: *Github.com* [online]. © 2020 [cit. 2021-11-03]. Dostupné z: https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions

EUROPEAN BANKING FEDERATION, 2019. *EBF position pap on AI in banking industry* [online]. Brussel: EBF [cit. 2021-11-03]. Dostupné z: https://www.ebf.eu/wp-content/uploads/2020/03/EBF-AI-paper-_final-.pdf

GALEA, A., 2018. *Beginning Data Science with Python and Jupyter*. Birmingham : Packt. ISBN 978-1-87953-202-9.

GANDHI, R., 2018. Support Vector Machine — Introduction to Machine Learning Algorithms. In: *Towardsdatascience.com* [online]. 7. 6. 2018 [cit. 2021-11-03]. Dostupné z: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

GOUVÊA, M. A. a GONÇALVES, E. B., 2007. *Credit Risk Analysis Applying Logistic Regression, Neural Networks and Genetic.* Dallas: POMS 18th Annual Conference.

GOYAL, A. a KAUR, R., 2016. Accuracy Prediction for Loan Risk Using Machine Learning Models. *International Journal of Computer Science Trends and Technology*. Vol. 4, Iss. 1, p. 6.

HAMILTON, D. T. a CANTOR, R., 2006. *Measuring Corporate Default Rates* [online]. New York: Moodys [cit. 2021-03-11]. Dostupné z: https://www.moodys.com/sites/products/defaultresearch/2006200000425249.pdf

HARSHDEEP, S., 2018. Understanding Gradient Boosting Machines. In: *Towardsdatascience.com* [online]. 3. 11. 2018 [cit. 2021-11-03]. Dostupné z: https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab

HASTIE, T., TIBSHIRANI, R., FRIEDMAN, J., 2008. *The Elements of Statistical Learning.* 2nd ed. Stanford: Springer, 2008.

HOME CREDIT, 2018. Home Credit Default Risk. In: *Kaggle.com* [online]. © 2018 [cit. 2021-03-11]. Dostupné z: https://www.kaggle.com/c/home-credit-default-risk

ICRA, 2016. ICRA's Rating Methodology for Securitization Transactions [online]. New Delhi: ICRA [cit. 2021-03-11]. Dostupné z: https://www.icra.in/Files/Articles/ICRA%20Securitisation%20Rating%20Methodology.pdf

JUPYTER TEAM, 2015. The Jupyter Notebook. *Jupyter-notebook.readthedocs.io* [online]. © 2015 [cit. 2021-03-11]. Dostupné z: https://jupyter-notebook.readthedocs.io/en/stable/

KOEHRSEN, W., 2021. Jupyter Notebook Extensions. In: *Towardsdatascience.com* [online]. 7. 12. 2018 [cit. 2021-03-11]. Dostupné z: https://towardsdatascience.com/jupyter-notebook-extensions-517fa69d2231

KOPIČKA, O., 2019. Odemykáme potenciál AI & Machine Learning v e-commerce - řízení AI projektu (část 3). In: *Ecommerce-academy.cz* [online]. 8. 7. 2019 [cit. 2021-03-11]. Dostupné z: https://www.ecommerce-academy.cz/post/serial-uvod-do-ai-3

KUTTY, G., 1990. Logistic regression and probability of default of developing. *Applied Economics* [online]. Vol. 22, Iss. 12, p. 1649 [cit. 2021-03-11]. Dostupné z: https://www.tandfonline.com/doi/abs/10.1080/00036849000000071

LABARRE, O., 2021. Credit Risk. In: *Investopedia.com* [online]. 4. 3. 2021 [cit. 2021-03-11]. Dostupné z: https://www.investopedia.com/terms/c/creditrisk.asp

LEWIS, E. M., 1994. *Introduction to Credit Scoring.* San Rafael: Athena Press.

MATPLOTLIB, 2021. Matplotlib: Visualization with Python. [online]. © 2021 [cit. 2021-03-11]. Dostupné z: https://matplotlib.org/stable/index.html

MITCHELL, T., 1992. *Machine Learning.* New York: McGraw Hill. ISBN 0-07-042807-7.

NUMPY. 2021. About Us. *Numpy.org* [online]. © 2021 [cit. 2021-03-11]. Dostupné z: https://numpy.org/about/.

OPEN RISK, 2006. Open Risk Manual. *Openriskmanual.org* [online]. © 2006 [cit. 2021-03-11]. Dostupné z: https://www.openriskmanual.org/wiki/Default_Rate

PANDAS, 2021. Package overview. *Pandas.pydata.org* [online]. © 2021 [cit. 2021-03-11]. Dostupné z: https://pandas.pydata.org/docs/getting_started/overview.html

PIRYONESI, M. S. a EL-DIRABY, T. E. 2021. Using Machine Learning to Examine Impact of Type of Performance Indicator on Flexible Pavement Deterioration Modeling. *Journal of Infrastructure Systems* [online]. Vol. 27, No. 2 [cit. 2021-03-11]. Dostupné z: https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29IS.1943-555X.0000602

PROJECT JUPYTER, 2021. About Us. *Jupyter.org* [online]. © 2021 [cit. 2021-03-11]. Dostupné z: https://jupyter.org/about.

SEABORN, 2021. An introduction to seaborn. *Seaborn.pydata.org* [online]. © 2021 [cit. 2021-03-11]. Dostupné z: https://seaborn.pydata.org/introduction.html

SHIROKOV, S., 2015. GitHub + Jupyter Notebooks = <3. In: *Github.blog* [online]. 7. 5. 2015 [cit. 2021-03-11]. Dostupné z: https://github.blog/2015-05-07-github-jupyter-notebooks-3/.

SCHREINER, M. 2003. *A Cost-Effectiveness Analysis of the Grameen Bank of Bangladesh.* Center for Social Development. St. Louis: Washington University in St. Louis, 2003.

ÚOOÚ. 2017. Pokyny k automatizovanému individuálnímu rozhodování a profilování. [online]. Praha: ÚOOU [cit. 2021-03-11]. Dostupné z: https://www.uoou.cz/assets/File.ashx?id_org=200144&id_dokumenty=31893.

VITALI, S., 2018. *Credit Risk in Banking* [online]. Praha: MFF UK [cit. 2021-03-11]. Dostupné z: https://www2.karlin.mff.cuni.cz/~vitali/documentsCourses/CLIENT%20SCORING%20PROCEDURE.pdf

XGBOOST DEVELOPERS, 2020. XGBoost Documentation. *Xgboost.readthedocs.io* [online]. © 2020 [cit. 2021-03-11]. Dostupné z: https://xgboost.readthedocs.io/en/latest/

Z_AI, 2020. Logistic Regression Explained. In: *Towardsdatascience.com* [online]. 2. 8. 2020 [cit. 2021-03-11]. Dostupné z: https://towardsdatascience.com/logistic-regression-explained-9ee73cede081

ZHANG, Z. et al., 2018. Exploring the clinical features of narcolepsy type 1 versus narcolepsy type 2 from European Narcolepsy Network database with machine learning. *National Library of Medicine* [online]. Vol. 8, No. 1, p. 10628 [cit. 2021-11-03]. Dostupné z: https://pubmed.ncbi.nlm.nih.gov/30006563/