**Česká zemědělská univerzita v Praze**

**Provozně ekonomická fakulta**

**Katedra informačního inženýrství**

# Diplomová práce

## Testování rozhodovacích algoritmů na notebooku Jupyter

**Radek Fajgl**

**© 2022 ČZU v Praze**

# ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

# ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Radek Fajgl

Systémové inženýrství a informatika
Informatika

Název práce

**Testování rozhodovacích algoritmů na notebooku Jupyter**

Název anglicky

**Decision algorithms testing using Jupyter notebook**

---

**Cíle práce**

Cílem práce je demonstrovat možnosti využití notebooku Jupyter pro testování moderních rozhodovacích algoritmů na vybraném datovém souboru.

**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 popíše způsob jejich realizace a testování na notebooku Jupyter

V praktické části práce student názorně předvede, jakým způsobem lze využít platformu notebooku Jupyter pro testování rozhodovacích algoritmů. Vybere si vhodný datový soubor, navrhne algoritmy vhodné pro jeho řešení a metodiku jejich testování. Z realizovaných vybraných řešení poté vybere nejvhodnější. Výsledný soubor Jupyteru (soubor .ipynb) bude obsahovat kromě programové realizace rozhodovacích algoritmů také analýzu datového souboru a výsledky získané v procesu učení a testování. Výsledky, pro které to bude vhodné, budou prezentovány v grafické formě.

**Doporučený rozsah práce**

60 pages

**Klíčová slova**

strojové učení, konvoluční sítě, hluboké učení, deep learning

---

**Doporučené zdroje informací**

GÉRON, A. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems.* Beijing ; Boston ; Farnham ; Sevastopol ; Tokyo: O'Reilly, 2019. ISBN 978-1-492-03264-9.

---

**Předběžný termín obhajoby**

2021/22 ZS – PEF

**Vedoucí práce**

doc. Ing. Arnošt Veselý, CSc.

**Garantující pracoviště**

Katedra informačního inženýrství

Elektronicky schváleno dne 1. 11. 2021

**Ing. Martin Pelikán, Ph.D.**

**Vedoucí katedry**

Elektronicky schváleno dne 23. 11. 2021

**Ing. Martin Pelikán, Ph.D.**

**Děkan**

V Praze dne 27. 02. 2022

**Čestné prohlášení**

Prohlašuji, že svou diplomovou práci "Testování rozhodovacích algoritmů 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 30.3.2022 _____

**Poděkování**

Rád bych touto cestou poděkoval docentu Arnoštovi Veselému za odborné vedení a nedocenitelné rady k této diplomové práci.

# Testování rozhodovacích algoritmů na notebooku Jupyter


## Abstrakt

Tato diplomová práce se zabývá testováním rozhodovacích algoritmů, které je prováděno na notebooku Jupyter. Hlavním cílem práce je demonstrovat možnosti využití notebooku Jupyter pro řešení rozhodovacích problémů na vybraném datovém souboru. Práce obsahuje dvě hlavní části. První část je teoretická a je v ní nejprve vysvětlena architektura notebooku Jupyter a práce v jeho prostředí. Dále jsou v ní vysvětleny základní pojmy týkající se problematiky umělé inteligence a strojového učení. Jsou v ní též popsány algoritmy strojového učení, které budou v praktické části práce využívány. Praktická část práce se zabývá vytvořením kompletní metodiky pro řešení rozhodovacích problémů. Nejprve jsou v ní uvedeny možnosti získání vhodných dat. Následně je vybrán datový soubor. Poté je na notebooku Jupyter provedena analýza zvoleného datového souboru, která je následovaná sestavováním předpovědních modelů pomocí algoritmů strojového učení. Modely jsou otestovány a zhodnoceny jejich výsledky.



**Klíčová slova:** strojové učení, konvoluční sítě, hluboké učení, notebook jupyter

# Decision algorithms testing using Jupyter notebook


## Abstract

This diploma thesis deals with testing decision algorithms, which is performed on a Jupyter Notebook. The main goal of this thesis is to demonstrate the possibilities of using a Jupyter Notebook to solve decision problems on a selected data file. This thesis contains two main parts. The first part is theoretical and first explains the architecture of the Jupyter Notebook and the work in its environment. It also explains the basic concepts related to artificial intelligence and machine learning. It also describes machine learning algorithms that will be used in the practical part of this thesis. The practical part of this thesis deals with the creation of a complete methodology for solving decision problems. First, it presents the possibilities of obtaining suitable data. Then a data file is selected. Then, the analysis of the selected data file is performed on the Jupyter Notebook, which is followed by the compilation of prediction models using machine learning algorithms. The models are tested and their results are evaluated.



**Keywords**: machine learning, convolutional networks, deep learning, notebook jupyter

# Obsah

# Seznam obrázků

# Seznam tabulek

# 1 Úvod

V dnešní době se umělá inteligence a strojové učení využívá v mnoha oborech lidské činnosti. Ať už jde například o předpovědi kurzů měn, rozpoznávání obrazů, nebo doporučování produktů na základě předchozích nákupů. Pro řešení těchto úloh využíváme algoritmů, pomocí nichž sestavujeme vhodné modely. Na jejich základě dokážeme s určitou přesností předpovědět budoucí vývoj, nebo rozhodování v dané situaci. Užitečným nástrojem pro řešení těchto typů úloh je notebook Jupyter.

Hlavním cílem této práce je navrhnout postup pro řešení úloh strojového učení a s využitím Jupyter Notebooku demonstrovat na konkrétním příkladu řešení vybrané úlohy. Práce je rozdělena na teoretickou a praktickou část.

V teoretické části se nejprve seznámíme s notebookem Jupyter, dozvíme se něco o jeho architektuře a práci v jeho prostředí. Poté se budeme zabývat umělou inteligencí a strojovým učením. Vysvětlíme si základní pojmy týkající se této problematiky a též i základní rozdělení algoritmů. V poslední kapitole teoretické části popíšeme algoritmy, které budeme prakticky využívat pro tvorbu modelů.

V praktické části práce je za použití teoretických znalostí popsaných v teoretické části práce zpracován kompletní postup, který lze použít pro řešení úloh strojového učení. Ten je demonstrován na konkrétním příkladu.

# 2 Cíl práce a metodika

## 2.1 Cíl práce

Tato diplomová práce se zabývá testovacími algoritmy a jejich využitím pro řešení rozhodovacích problémů. Hlavním cílem práce je vypracovat metodiku pro testování rozhodovacích algoritmů na notebooku Jupyter. Bude k tomu využit vhodný datový soubor, který bude vybrán v praktické části, na němž bude názorně demonstrováno, jak využít Jupyter Notebook k řešení rozhodovacích úloh.

## 2.2 Metodika

První kapitola teoretické části práce bude věnována notebooku Jupyter, bude v ní vysvětlena jeho architektura, způsob instalace a práce v jeho prostředí. Další kapitola teoretické části se bude zabývat umělou inteligencí a strojovým učením. Bude obsahovat vysvětlení základních pojmů, které s touto problematikou souvisejí. V poslední kapitole teoretické části budou vysvětleny metody strojového učení, které budeme používat v praktické části pro tvorbu modelů.

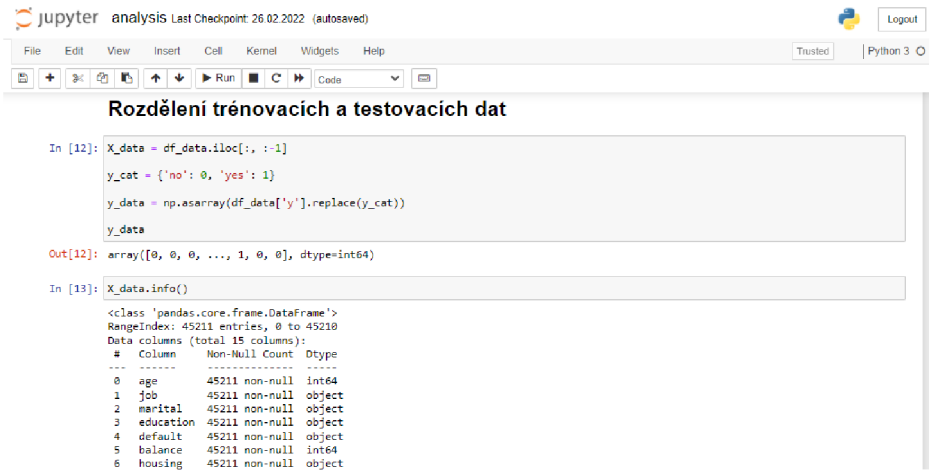
Praktická část této práce bude věnována vypracování metodiky pro testování rozhodovacích algoritmů na notebooku Jupyter. Nejprve vybereme vhodný datový soubor a provedeme jeho analýzu. Poté rozdělíme datový soubor na trénovací a testovací část. Následně budeme pomocí vybraných algoritmů strojového učení sestavovat modely, které budou trénovány na trénovacích datech. Poté bude provedeno jejich otestování na testovacích datech, které bude následně vyhodnoceno.

# 3 Teoretická část

## 3.1 Jupyter Notebook

Projektem Jupyter je nazývána skupina aplikací, které umožňují programování prostřednictvím webového rozhraní. Tyto aplikace jsou vyvíjeny neziskovou organizací Project Jupyter. Název byl zvolen podle tří hlavních podporovaných jazyků, kterými jsou Julia, Python, a R, a také jako pocta Galileovým notebookům obsahujících objev měsíců Jupyteru. Autory projektu Jupyter jsou Fernando Pérez a Brian Granger, kteří ho založili v roce 2014. Stal se nástupcem čistě pythonového projektu iPython, jehož programový kód je v projektu nadále vyvíjený a stal se jeho základní součástí. [1, 2]

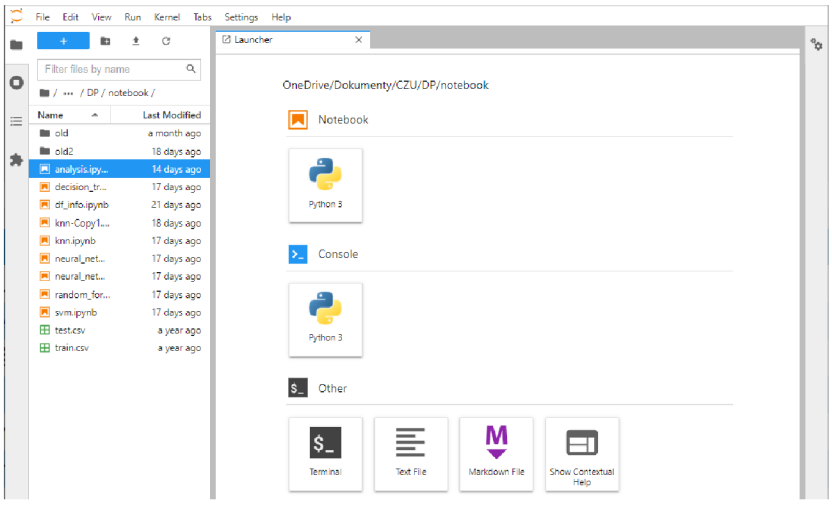
Nejpopulárnější a také i nejvíce používanou součástí projektu Jupyter je Jupyter Notebook. Je to webová aplikace s otevřeným zdrojovým kódem běžící přímo v internetovém prohlížeči. Název je odvozen od jeho uživatelského rozhraní podobnému diáři (notebooku) do kterého uživatel zadává obsah. Jeho předchůdcem byl projekt iPython Notebooks. který umožňoval prostřednictvím webového rozhraní interaktivně vyhodnocovat jednotlivé vstupy uživatele. Uživatelské rozhraní Jupyter Notebooku umožňuje jeho uživatelům vytvářet a sdílet dokumenty, které kromě zdrojového kódu mohou obsahovat i další obsah, což mohou být například tabulky, grafy, obrázky. Obsah lze do vytvářeného dokumentu vkládat přímo, nebo nechat vytvořit pomocí zadání funkce, jejíž výstup je ihned interaktivně zobrazen. Je též podporována řada dalších funkcí, jako zadávání vzorců psaných v TeXu nebo LaTeXu, tvorba slajdů, sdílení zdrojového kódu. V textových buňkách je podporován Markdown pro jednoduché formátování textu. Dokumenty z notebooku Jupyter jsou uloženy v souboru s příponou ipynb a lze je též vyexportovat do mnoha formátů jakými jsou PDF, HTML, LaTeX, Markdown a další. [1, 2]




*Obrázek 1 - Jupyter Notebook - okno*

## JupyterLab

JupyterLab je další generaci notebooku Jupyter. Rozšiřuje možnosti jeho použití a poskytuje vylepšené uživatelské rozhraní, které má modulární strukturu. Lze v něm otevřít několik notebooků, nebo dalších aplikací (např. editory, widgety, mapy) jako záložek v jednom okně. Zachovává si kompatibilitu s formáty klasického Jupyter Notebooku. Je vhodnější pro pokročilé uživatele, kterým již funkce klasického Jupyter Notebooku nedostačují. [31]

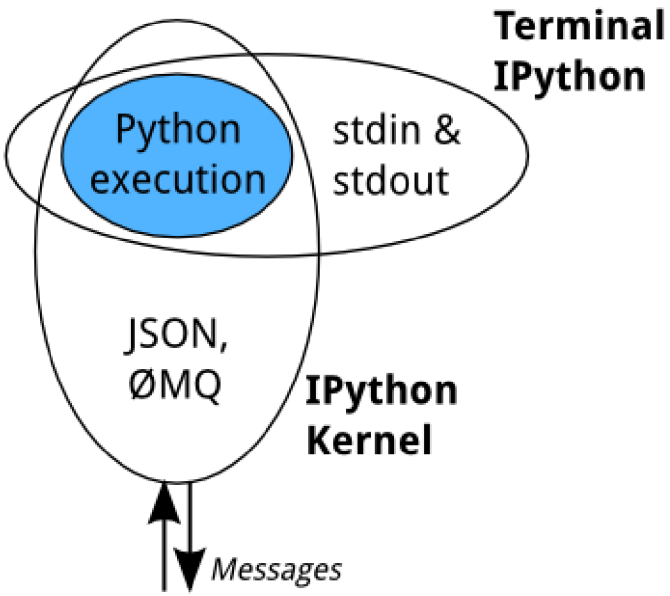



*Obrázek 2 - JupyterLab - okno*

### 3.1.1 Architektura Jupyter Notebooku

Jupyter Notebook je založen na síťové architektuře klient-server. Klientskou aplikaci má spuštěnou uživatel ve svém webovém prohlížeči. Serverem je Jupyter, který obsahuje výpočetní moduly (kernely) pro zvolené programovací jazyky. Uživatel zadává ve webovém prohlížeči příkazy, které jsou po odeslání na server zpracovány a jejich výsledky ihned poslány zpět do prohlížeče uživateli. [2]

Základem Jupyter Notebooku je iPython Kernel, který byl převzat z původního projektu iPython Notebooks. Pro umožnění podpory dalších programovacích jazyků a kernelů však musela být architektura původního kernelu upravena. [2, 3]



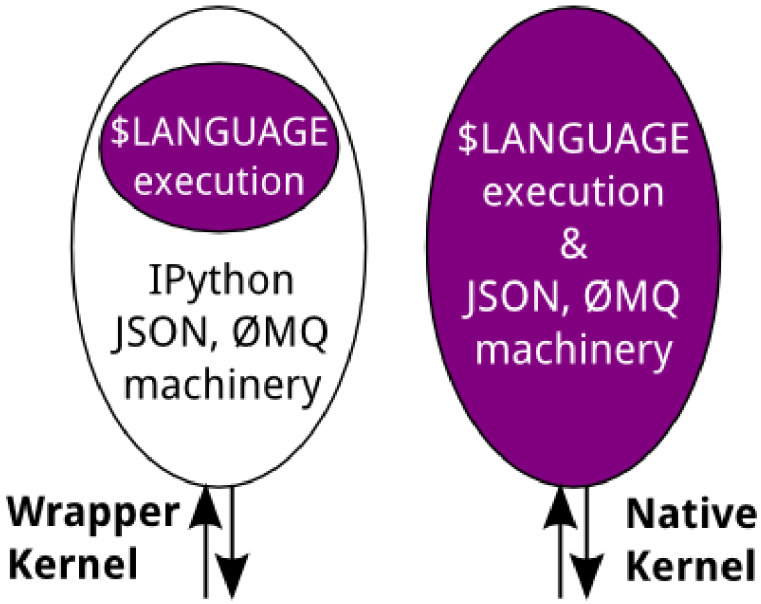

*Obrázek 3 - IPython Kernel [3]*

iPython kernel využívají všechna ostatní rozhraní. Je zodpovědný za spuštění uživatelského kódu. Komunikuje prostřednictvím rozhraní ØMQ, přes které přijímá pomocí JSON zpráv příkazy a zasílá zpět jejich výsledky. Kernel podporuje programovací jazyk Python. Architektura kernelu je však navržena tak, aby bylo možné Jupyter rozšířit o další jazyky. [2, 3]

**Rozšíření Jupyter Notebooku o další jazyky lze provést dvěma způsoby:**

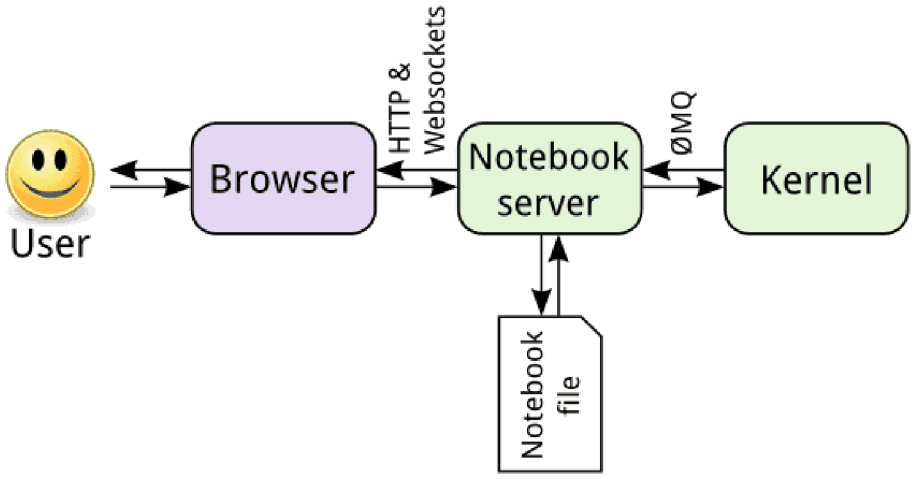
- Wrapper Kernel - Využít iPython kernelu, který bude volat příkazy interpreteru jiného programovacího jazyka.
- Native Kernel - Přidání dalšího nativního Kernelu jiného programovacího jazyka.

Rozšíření pomocí Wrapper Kernelu je vhodné především pro programovací jazyky blízké Pythonu (např. octave_kernel). U nativních kernelů je důležité, aby komunita udržovala jejich vývoj (např. IJulia, IHaskel). [3]




*Obrázek 4 - Rozšíření Jupyteru o další jazyky [3]*

Architektura Jupyter Notebooku má dvě hlavní části. Jsou jimi webový prohlížeč a kernel(y). Tyto moduly jsou propojeny prostřednictvím Notebook serveru, přes který spolu komunikují. S prohlížečem server komunikuje prostřednictvím http a Web socketů, s kernely přes ØMQ. Dokument Jupyter Notebooku je soubor s příponou ipynb. Je v něm uložen v JSON formátu veškerý obsah zadaný ve webové aplikaci. Viz. následující obrázek. [2, 3]




*Obrázek 5 - Architektura Jupyter Notebooku [3]*

### 3.1.2 Jak začít s Jupyter Notebookem

Začít pracovat s Jupyter Notebookem lze dvěma základními způsoby:

- Nainstalovat si ho do počítače
- Využít webovou aplikaci, kde je již nainstalován (Kaggle)

Hlavní výhoda nainstalovaného Jupyter Notebooku v počítači je v rychlosti zpracování kódu, což se projeví především u spouštění složitějších algoritmů náročných na zdroje. Nevýhodou je nutnost projít procesem instalace nejen Jupyteru, ale i potřebných knihoven, pokud nepoužijeme již předpřipravenou distribuci s potřebnými knihovnami (Anaconda). Je zde též nízké riziko neoprávněného odcizení našich dat, jelikož je máme u sebe.

U použití předinstalované webové aplikace (např. Kaggle) jsou hlavními výhodami možnost okamžitě začít pracovat, dostupnost odkudkoliv z Internetu, možnosti sdílení notebooků v rámci komunity. Nevýhodami jsou pomalejší zpracování úloh a nutnost pracovat na serveru, kde nemáme plnou kontrolu nad bezpečností našich dat.

Zde jsou dva nejtypičtější způsoby instalace do počítače, použití webové aplikace Kaggle se budu věnovat v praktické části.

**Instalace do Pythonu:**

Nejprve potřebujeme nainstalovat Python. Ten lze získat buď z oficiálních stránek https://www.python.org/, nebo lze využít instalačních balíčků, které jsou součástí repozitářů podporovaných operačních systémů. Jupyter Notebook není standardní součástí Pythonu, doinstalovat lze pomocí nástroje pip, což je doporučený nástroj pro instalaci balíčků do Pythonu, který bývá jeho standardní součástí. Následně doinstalujeme potřebné knihovny pro práci.

**Instalace distribuce Anaconda**

Další možností instalace je použít distribuci Anaconda. Tato možnost je vhodná pokud budeme používat Jupyter Notebook pro datovou vědu. Součástí této distribuce je Python3 a další potřebné knihovny obsahující nástroje pro datovou vědu a strojové učení jakými jsou například Scikit-learn, Pytorch, Keras, TensorFlow a též i nástroje pro vizualizaci dat. [22]

Instalaci provedeme stažením instalátoru ze stránek https://www.anaconda.com/. Podporovány jsou operační systémy Windows, Linux, macOS.

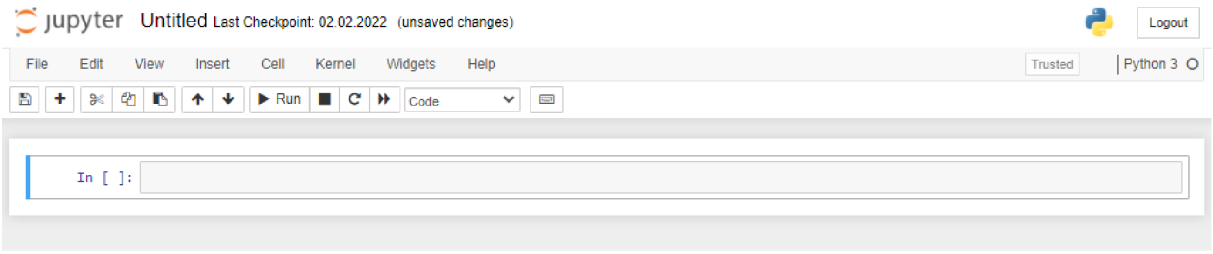
Součástí nainstalované Anacondy je program Anaconda Navigator, který slouží ke spouštění nainstalovaných aplikací, jež jsou součástí instalace a též i správě nainstalovaných balíčků. Jupyter Notebook lze spustit z Anaconda Navigátoru, nebo též samostatně.

### 3.1.3 Prostředí Jupyter Notebooku

Po spuštění Jupyter Notebooku se jeho webová aplikace otevře na jeho standardním lokálním portu 8888 (localhost:8888). Ve webovém rozhraní poté můžeme vybrat adresář, v něm vytvořit nový notebook, nebo otevřít stávající.


*Obrázek 6 - Jupyter Notebook - výběr*

Po vytvoření nového notebooku vstoupíme do hlavního rozhraní Jupyter Notebooku, ve kterém budeme pracovat. V záhlaví okno obsahuje název souboru, hlavní menu a nástrojovou lištu s tlačítky pro operace a navigaci při tvorbě notebooku. Pracovní plocha notebooku obsahuje buňky, do kterých zapisujeme programový kód, nebo text. [34]



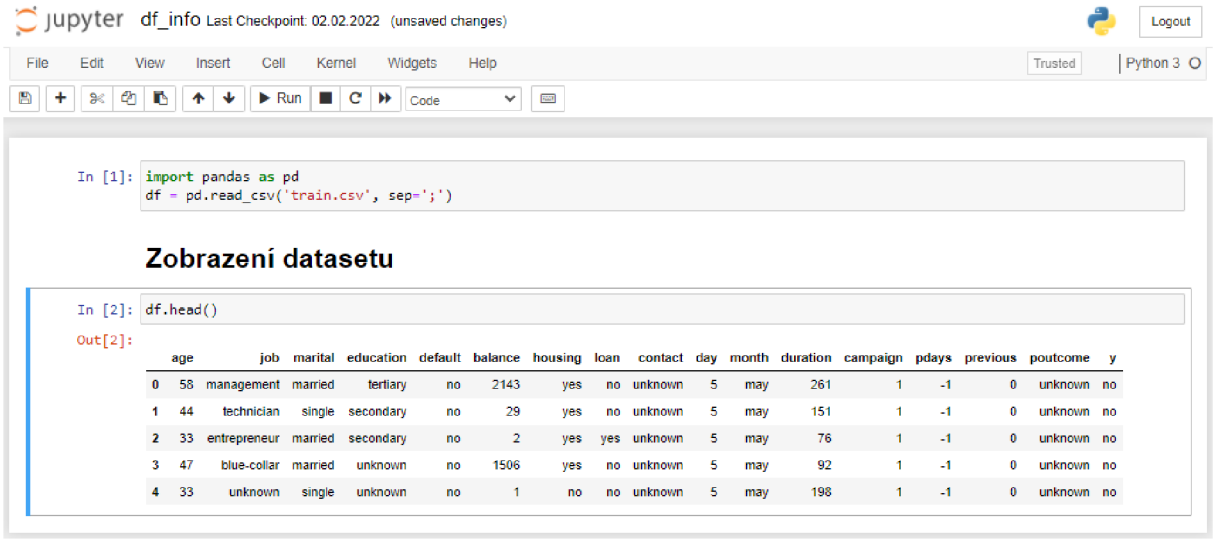

*Obrázek 7 - Jupyter Notebook - rozhraní*

Název nového notebooku je defaultně nastaven na Untitled. To lze jednoduše změnit kliknutím na název.


*Obrázek 8 - Jupyter Notebook - změna jména souboru*

Na pracovní plochu Notebooku vkládáme buňky, ty mohou obsahovat programový kód, nebo text. Typ buňky lze měnit v nástrojové liště. Další tlačítka pro práci s buňkami umožňují přidávání a odebírání buněk, kopírování buněk a změny jejich pořadí. Pomocí Markdownu lze textové buňky formátovat. [34]

V nástrojové liště též najdeme skupinu tlačítek pro ovládání kernelu. Jedná se o spouštění kódu v dané buňce, zastavení běhu kernelu, nebo jeho restart. Informace o použitém kernelu a jeho stavu vidíme vpravo nahoře.




| | age | job | marital | education | default | balance | housing | loan | contact | day | month | duration | campaign | pdays | previous | poutcome | y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 58 | management | married | tertiary | no | 2143 | yes | no | unknown | 5 | may | 261 | 1 | -1 | 0 | unknown | no |
| 1 | 44 | technician | single | secondary | no | 29 | yes | no | unknown | 5 | may | 151 | 1 | -1 | 0 | unknown | no |
| 2 | 33 | entrepreneur | married | secondary | no | 2 | yes | yes | unknown | 5 | may | 76 | 1 | -1 | 0 | unknown | no |
| 3 | 47 | blue-collar | married | unknown | no | 1506 | yes | no | unknown | 5 | may | 92 | 1 | -1 | 0 | unknown | no |
| 4 | 33 | unknown | single | unknown | no | 1 | no | no | unknown | 5 | may | 198 | 1 | -1 | 0 | unknown | no |

*Obrázek 9 - Jupyter Notebook - příklad*

V prostředí Jupyter Notebooku lze používat i klávesové zkratky, kterými lze práci značně urychlit. Seznam všech zkratek se dá zobrazit stisknutím klávesy „h“. Lze pomocí nich kompletně ovládat práci v jeho prostředí, což může být např. práce s buňkami, ukládání dokumentu, spouštění kódu a dalších užitečných funkcí.

## 3.2 Umělá inteligence a strojové učení

Umělá inteligence (Artificial Intelligence - AI) je obor informatiky, který se zabývá schopnostmi strojů přibližovat se inteligenci lidské. Měla by být schopna řešit různé komplexní úlohy a napodobovat lidské schopnosti jako je učení, uvažování, kreativita, plánování. [1, 29]

### Strojové učení (Machine Learning)

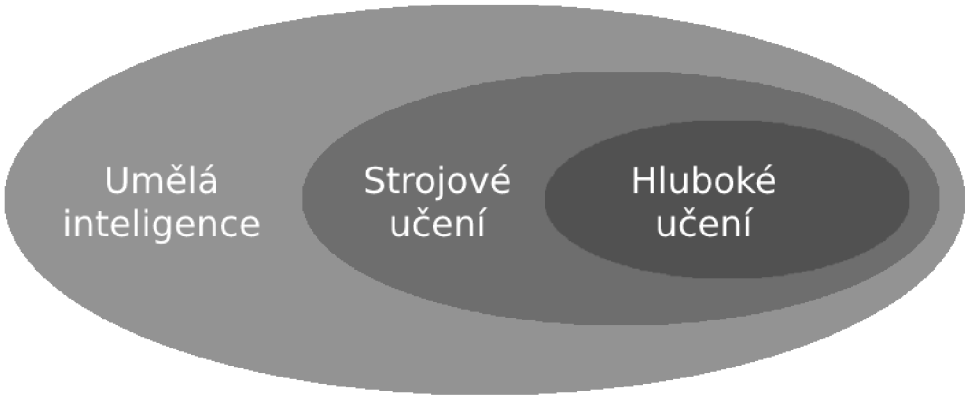
Strojové učení (Machine Learning - ML) je podoblastí umělé inteligence, která zkoumá schopnosti strojů učit se a přizpůsobovat se dynamicky novým situacím na základě již zjištěných dat. Tímto způsobem se stroje snaží napodobit lidskou schopnost učení se z předchozích chyb tak, že při neúspěchu hledají nové přístupy pro řešení problému. [1, 33]

### Hluboké učení (Deep Learning)

Specifickým rozšířením strojového učení je hluboké učení (Deep Learning - DL), které dokáže řešit složitější a komplexnější úlohy s dosahováním lepších výsledků. Pro svou

činnost využívá vícevrstvé neuronové sítě, které se více přibližují chování lidského mozku. Je využívané především v oblastech rozpoznávání lidské řeči, obrázků, virtuálních asistentů (např. Google Translate, Siri, Alexa, Face ID). Vyžaduje vyšší výpočetní výkon. [5]

Na následujícím obrázku je znázorněn vztah mezi umělou inteligencí, strojovým učením a hlubokým učením. [10]




*Obrázek 10 - Vztah mezi umělou inteligencí, strojovým učením a hlubokým učením*

### 3.2.1 Algoritmy strojového učení

Algoritmus strojového učení je programový kód, pomocí kterého můžeme analyzovat datový soubor. Na základě prováděných instrukcí algoritmus vytvoří model, který lze použít pro tvorbu budoucí předpovědi nebo rozhodování. [7]

**Rozdělení technik strojového učení** [1, 8]

- Učení s učitelem (Supervised learning)
- Učení bez učitele (Unsupervised learning)
- Kombinace učení s učitelem a bez učitele (Semi-supervised learning)
- Zpětnovazební učení (Reinforcement learning)

**Učení s učitelem**

Je metodou, kdy máme k dispozici data, u kterých známe pro každou množinu vstupních údajů správný výstup. Na základě těchto známých trénovacích dat je pak algoritmem řešen daný typ úlohy. [1, 8]

Máme 3 základní množiny dat. Jsou jimi trénovací množina, validační množina a testovací množina. Na trénovací množině dat provádíme samotné učení modelu. Výsledky průběhu

učení kontrolujeme na validační množině. Na testovací množinu dat, která by měla být odlišná od předchozích, poté aplikujeme naučený model a zjistíme úspěšnost jeho predikce.

**Učení bez učitele**

Je metodou, kdy máme k dispozici vstupní data, ale neznáme výstup. Algoritmy na základě vstupních dat sami dospějí k výsledkům. K typickým příkladům těchto algoritmů patří neuronové sítě. [1, 8]

**Kombinace učení s učitelem a bez učitele**

U kombinace učení s učitelem a bez učitele máme k dispozici část dat se známým výstupem, ale u části vstupních dat výstup neznáme. [1, 8]

**Zpětnovazební učení**

Je založeno na maximalizaci odměny. Jsou známa pouze vstupní data, u kterých algoritmus hledá nejlepší způsob, jakým lze dospět požadovanému výsledku. Je podobné algoritmům učení s učitelem, s tím rozdílem, že u zpětnovazebního učení výsledek není znám a algoritmus se na něj snaží sám přijít. [1, 8]

**Rozdělení algoritmů podle způsobu zpracování** [8]

- Dávkové
- Inkrementální

Dávkové algoritmy požadují mít všechna potřebná data před začátkem výpočtu.

Inkrementální algoritmy dokážou po dodání nových dat upravit model výpočtu, aniž by ho museli od začátku přepočítávat. Jinak řečeno se dokážou přiučit.

### 3.2.2 Úlohy strojového učení:

V predikci dat řešíme vytvoření modelu, který nám na základě historických dat předpoví data nová. Základní typy úloh, které řešíme, jsou klasifikace a regrese. [1]

- Klasifikace - rozděluje vstupní data do dvou i více tříd
- Regrese - na základě vstupních dat odhadujeme číselnou hodnotu výstupu

### 3.2.3 Přeučení a nedoučení

Přeučení a nedoučení jsou negativní jevy, které jsou z velké části příčinou špatné výkonosti algoritmů strojového učení. Mezi nimi se nachází ideální stav, kterého se snažíme dosáhnout.

**Přeučení**

Přeučení (overfitting) znamená, že náš model příliš dobře modeluje tréninková data. Je to typické úskalí vznikající v algoritmech strojového učení, když se model snaží přizpůsobit trénovacím datům do takové míry, že se naučí rozpoznávat datové vzory včetně šumu a náhodných výkyvů. [10, 32]

Indikátorem přeučení je vysoký rozptyl výkonu modelu. Vzniká při dlouhé době trénování modelu, nebo při jeho velké složitosti, kdy se naučí rozeznávat šum a nepodstatné informace v trénovacích datech.

Výsledkem přeučení je model, který velmi dobře funguje na trénovací množině dat, ale není schopen správně fungovat na množině nových dat.

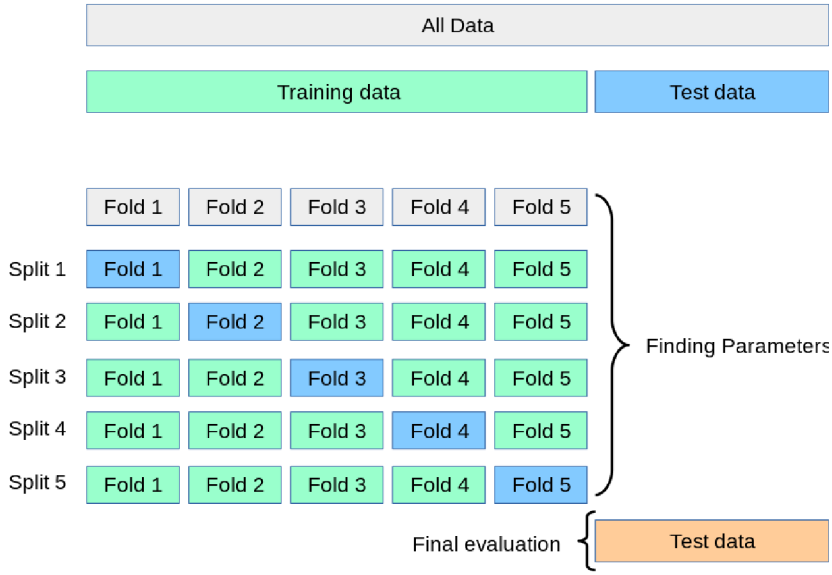
**Nedoučení**

Nedoučení (underfitting) znamená, že se náš model nedokázal správně naučit z trénovacích dat, a proto se ani nedokáže správně zobecnit na data nová. [10, 32]

K nedoučení může docházet, když máme nečistá zdrojová data, která obsahují šum nebo odlehlé hodnoty. Model pak není schopen odvodit vzory z datové sady. K nedoučení dochází také, pokud má model vysoké zkreslení, protože není schopen zachytit vztah mezi vstupy a cílovými hodnotami. Též může být příčinou nedoučení přílišná jednoduchost modelu.

### 3.2.4 Křížová validace

Křížová validace je metoda, která testuje schopnost předpovědi modelu. Vstupní data jsou rozdělena na několik podmnožin. Jedna z podmnožin slouží jako testovací a zbylé jako trénovací. Klasifikátor se natrénuje na trénovačích datech, přičemž jeho úspěšnost je ověřena na testovacích datech. Tento proces se opakuje k-krát, ale vždy s různým rozdělením vstupních dat. [1]




*Obrázek 11 - Křížová validace [35]*

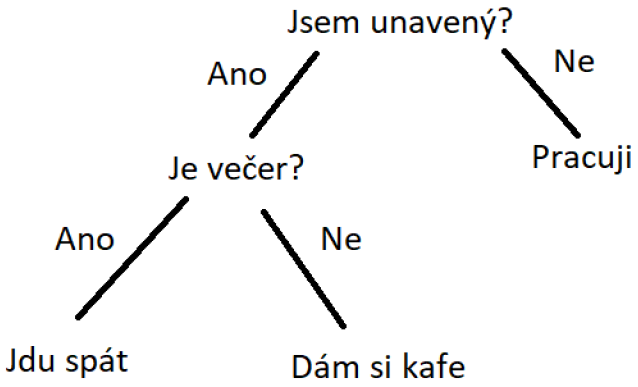
## 3.3 Modely strojového učení

Řeší úlohy prostřednictvím algoritmů strojového učení. Modelem strojového učení se stane algoritmus, který byl natrénován nad sadou trénovacích dat k rozpoznávání určitých typů vzorů. Natrénovaný model poté můžeme použít k provádění předpovědí nad novými daty, která doposud neviděl. [9]

### 3.3.1 Rozhodovací strom

Rozhodovací stromy (decision trees) jsou technikou vytěžování dat (data mining). Její hlavní výhodou je rychlé nalezení výsledku, který lze snadno interpretovat. Lze je využít v klasifikačních i regresních úlohách. [1]

Rozhodovací strom je tvořen stromovým grafem, což je souvislý graf neobsahující smyčku. Obsahuje kořenový vrchol, který je spojen pomocí hran s dalšími vrcholy, které mohou být dále větveny. Vrcholy stromu představují jednotlivá rozhodnutí, vystupující hrany jsou možnostmi daného rozhodnutí. Listy neboli vrcholy stromu, které nemají další potomky, jsou konečnými výsledky rozhodování. [10]




*Obrázek 12 - Rrozhodovací strom [11]*

Sestavování rozhodovacího stromu provádíme od kořenového vrcholu (uzlu) shora dolů. Postupně rozdělujeme trénovací data na čím dál menší podmnožiny (uzly) tak aby v nich převládaly prvky pouze jedné třídy. Uzly jsou vybírány na základě optimálního rozmístění prvků. Existují dvě základní kritéria výběru.

**Gini**

Gini nečistota je založena na měření frekvence, při které dojde k nesprávnému označení prvku datové sady, když je náhodně označen. Vypočítá se podle následujícího vzorce.

$$Gini = 1 - \sum_{i}^{n} p_i^2$$

$$kde\ p_i\ je\ pravděpodobnost\ třídy\ i$$

Pokud je hodnota Gini indexu 0, tak to znamená, že je uzel čistý, neboli všechny prvky v uzlu jsou jedné jedinečné třídy. Takovýto uzel nebude tedy znovu rozdělen. Podle vlastností majících nižší Gini index bude zvoleno optimální rozdělení. [12]

**Entropie**

Entropie je metrika pro výpočet nejistoty. Vypočítá se podle následujícího vzorce.

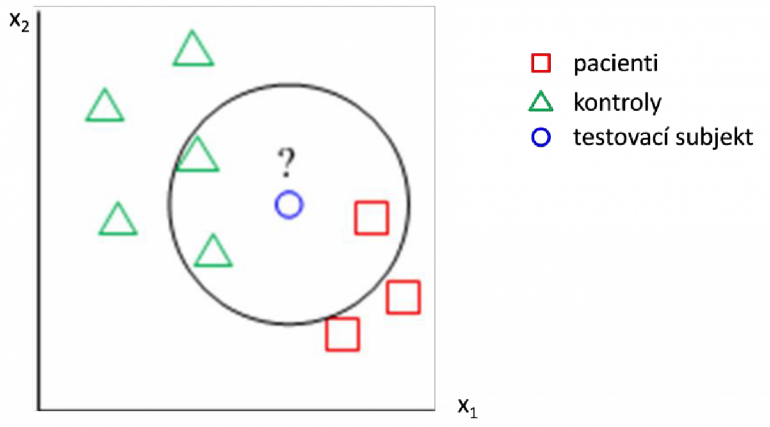
$$Entropie = -\sum_{i}^{n} p_i * log_2(p_i)$$

$$kde\ p_i\ je\ pravděpodobnost\ třídy\ i$$

Hodnota nula znamená, že všechny prvky v uzlu patří do stejné třídy. Optimální rozdělení je podobně jako u Gini indexu zvoleno podle prvků s nižší entropií. [12]

### 3.3.2 Metoda k-nejbližších sousedů

Metoda nejbližších sousedů patří mezi nejjednodušší klasifikátory. Je metodou strojového učení s učitelem, použitelná pro klasifikační úlohy i regresi. Cílem této metody je klasifikovat prvky reprezentované vícedimenzionálními vektory do několika tříd na základě jejich nejbližší vzdálenosti. Pro odstranění citlivosti na odlehlé hodnoty, používáme častěji k-nejbližších sousedů pro určení prvku do správné třídy (metoda k-nejbližších sousedů). [1, 13]



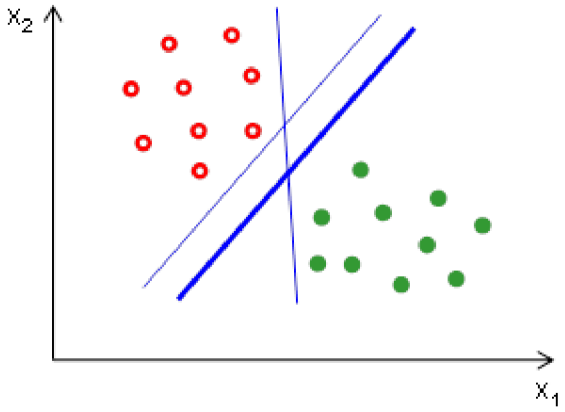

*Obrázek 13 - Metoda k-nejbližších sousedů [36]*

### 3.3.3 Metoda SVM

Metoda podpůrných vektorů (Support vector machines - SVM) je metodou strojového učení s učitelem, kterou lze využít pro regresní i klasifikační úlohy.

Princip této metody je založen na nalezení takové nadroviny v N-dimenzionálním prostoru, která bude rozdělovat datové body patřícím odlišným třídám z trénovací množiny dat. Takovýchto nadrovin lze však nalézt mnoho. Proto cílem metody podpůrných vektorů je

nalézt právě takovou nadrovinu, u které je hodnota minima vzdáleností bodů co největší. Maximalizací odstupu je zajištěna co největší jistota pro správnou klasifikaci nových bodů. Pruh bez bodů okolo nadroviny (též nazýván jako pásmo necitlivosti nebo hraniční pásmo) lze určit body, které leží na jeho okraji. Tyto hraniční body nazýváme podpůrné vektory (support vectors), podle nichž je nazvána i tato metoda. [1]




*Obrázek 14 - Metoda podpůrných vektorů [37]*

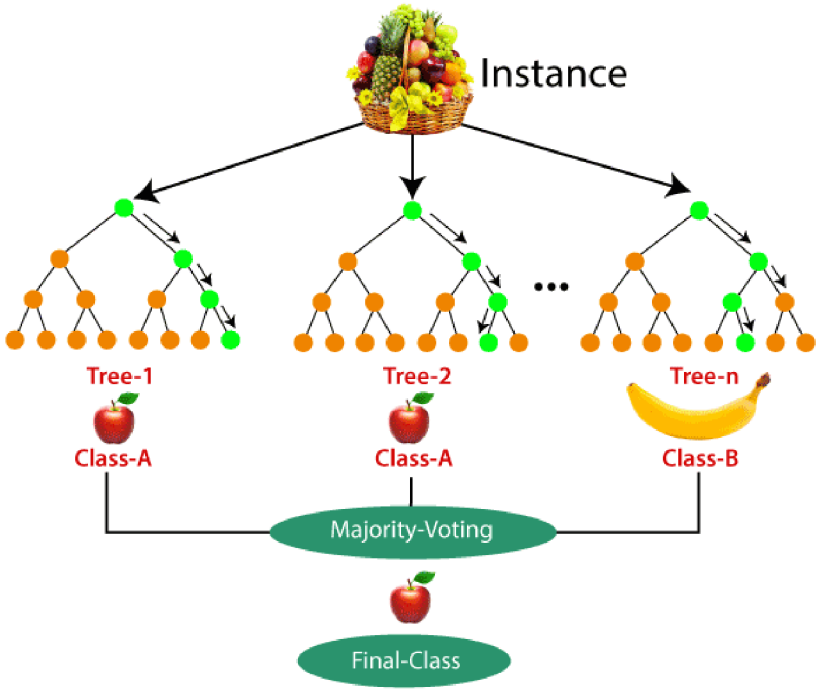
SVM algoritmy používají sadu matematických funkcí pro transformaci vstupních dat do požadované podoby. Těmto typům funkcí se říká jádra (Kernely) a mohou být různého typu. V současnosti se používají nejběžněji funkce RBF a lineární, které jsou vhodné pro většinu modelů. Dalšími častými alternativami mohou být například polynomiální nebo sigmoidální funkce.

### 3.3.4 Náhodný les

Náhodný les (Random forest) je flexibilní a též i snadno použitelný algoritmus strojového učení s učitelem, který i bez ladění jeho parametrů dokáže produkovat výborné výsledky. Pro svou jednoduchost a rozmanitost je jedním z nejpoužívanějších algoritmů. Je vhodný pro řešení klasifikačních i regresních úloh a používá se v různých odvětvích jako je bankovnictví, medicína, elektronické obchodování, akciový trh. [15]

Princip jeho činnosti spočívá v sestavení souboru mnoha rozhodovacích stromů, které spojuje dohromady pro získání přesnější předpovědi. Nejprve jsou vybrány náhodné datové body z trénovacího datového setu (technika pytlování - Bagging). Z nich jsou sestaveny rozhodovací stromy, které jsou nezávisle trénovány na přidělené podmnožině dat. Poté je na

základě výsledků předpovědí určeno klasifikátorem většinové rozhodnutí ze všech stromů jako správné.

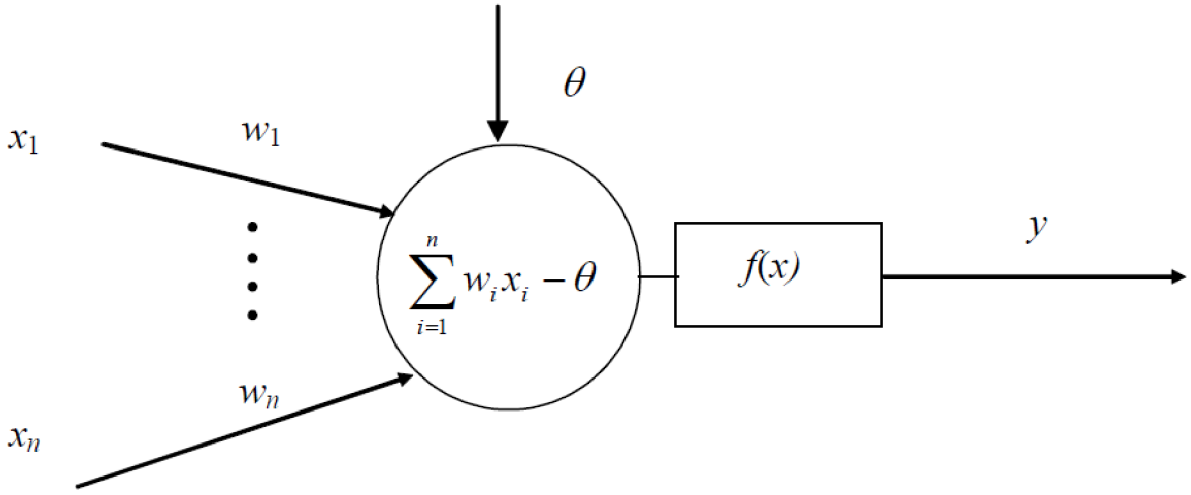



*Obrázek 15 - Náhodný les [16]*

### 3.3.5 Umělé neuronové sítě

Umělé neuronové sítě (artificial neural networks - zkratka ANN) jsou výpočetním modelem, který je používán v oblasti umělé inteligence (artificial inteligence - zkratka AI) a strojového učení. Můžeme pomocí nich řešit klasifikační i regresní úlohy.

Inspirací pro vznik umělých neuronových sítí je lidská nervová soustava, jejímž základním prvkem je neuronová buňka (neuron). Úkolem neuronů v organismu je přenášet, zpracovávat a uchovávat informace, které zajišťují jeho základní životní funkce. Stejně tak je neuron i základním prvkem umělých neuronových sítí. Neurony jsou spolu propojeny a předávají si navzájem signály. Platí přitom pravidlo, že každý neuron může mít několik vstupů, ale pouze jeden výstup, který může být rozvětven do více neuronů. Na vstupy neuronu mohou přicházet signály z jiných neuronů, nebo informace z vnějšku. [17, 18]



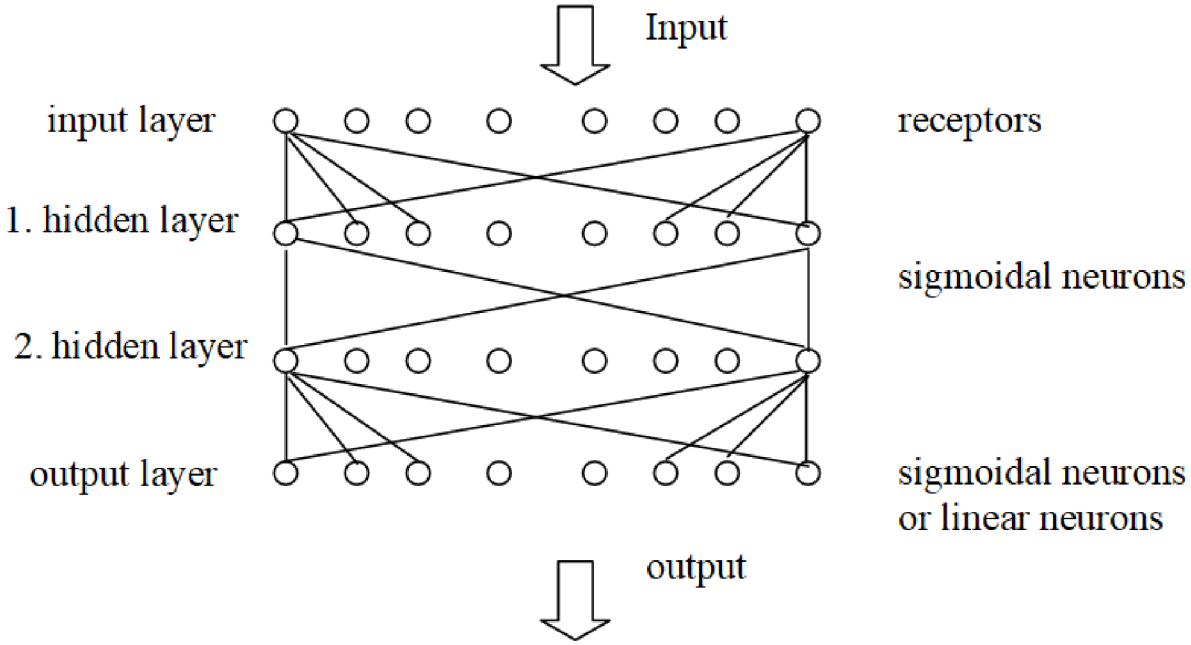

*Obrázek 16 - Neuron [20]*

Na vstupy neuronu přicházejí vstupní signály ($x_1$, $x_2$, …, $x_n$). Každý vstup má určenou váhu ($w_1$, $w_2$, …, $w_n$), která určuje jeho významnost. Po přijetí vstupních informací vynásobí neuron hodnoty vstupů vahami a sečte je. Pokud je vážený součin vstupů větší nebo roven prahové hodnotě (Θ), je neuron aktivován.

**Neurony jsou v neuronové síti uspořádány do tří vrstev:**

1. vstupní vrstva - receptory, vstupní proměnné
2. skryté vrstvy - vrstvy neuronů mezi vstupní a výstupní vrstvou, může jich být více
3. výstupní vrstva - neurony vytvářející výstupní proměnné

Přičemž platí, že pouze neurony sousedních vrstev jsou spolu propojeny. [19]




*Obrázek 17 - Neuronová síť [20]*

# 4 Praktická část

V praktické části navrhnu postup, který lze použít pro řešení úloh strojového učení na Notebooku Jupyter.

Nejprve uvedu možnosti, jak získat potřebná data pro sestavování modelů. Dále také uvedenu zdroje různých platforem, kde lze již připravená data získat.

Poté zvolím konkrétní datový soubor, který rozdělím na trénovací a testovací část. Na trénovací množině dat budu pomocí algoritmů strojového učení vytvářet modely. Následně budu testovat úspěšnost jejich předpovědi na testovací množině dat.

Na závěr praktické části zhodnotím úspěšnost předpovědí všech vytvořených modelů a provedu jejich porovnání.

## 4.1 Získání dat

Pro testování algoritmů strojového učení je nutno získat vhodná zdrojová data. Ta je nutno nejdříve nasbírat. Sběr dat pro řešení specifického úkolu strojového učení bývá často kritickým problémem. Obzvláště techniky hlubokého učení vyžadují velké množství správně označených údajů.

**Získat data lze třemi základními způsoby:**

- Objevování dat
- Rozšiřování dat
- Generování dat

### 4.1.1 Objevování dat

Týká se dat, která jsou indexována a publikována pro sdílení. Jedná se především o data generovaná z nějakého systému. Tyto systémy ovšem nemusí být vždy uzpůsobené pro generování datových sad vhodných ke sdílení, v takovém případě je nutno data dodatečně upravit. [21]

Do této skupiny patří systémy, které jsou navrženy pro sdílení datových sad. Tyto systémy často umožňují uživatelům spolupráci na analýze datových sad a publikování výsledků na webu. Typickým příkladem je například Kaggle. [21]

Dalším příkladem této skupiny jsou systémy určené převážně pro vyhledávání datových sad. Tyto systémy jsou typické pro velké společnosti, které generují mnoho datových sad z různých zdrojů. Jelikož bývá problém datové sady nalézt, umožňují systémy pro jejich vyhledávání nalézt jednotlivcům datové sady, které mohou dále analyzovat a zpracovávat. Systém pro zpracování datových sad od IBM umožňuje jejich vlastníkům sady zpracovávat i prohledávat poté co jsou vytvořeny. Systém Google Data Search (GOODS) dokáže kategorizovat metadata mnoha datových sad dostupných z různých uložišť v rámci Google. Systém WebTables automaticky extrahuje strukturovaná data z online zdrojů publikovaná ve formě z tabulek. [21]

### 4.1.2 Rozšiřování dat

Datové sady lze rovněž získat pomocí přístupu rozšíření existujících sad o externí data.

Jedním ze způsobů obohacení datové sady je odvozením latentní sémantiky z dat. Využíváme zde významové blízkosti slov, vyskytujících se v podobných částech textu. Slova jsou převedena na vektory a následně trénována. Předpovídáno je buď jedno slovo na základě okolních slov (model CBOW), nebo okolní slova na základě vybraného slova (model Skip-Gram). Též lze použít i další metody, například model LDA, který umí na základě napozorovaných skupin určit podobné části dat. [21]

Další možností rozšiřování dat je použít rozšiřování entit. Tento způsob se používá u neúplných datových sad, u kterých je třeba doplnit chybějící informace. Lze to provést pomocí seskupení tabulek, ve kterých jsou identifikovány podobné atributy.

Datovou sadu lze též rozšířit i integrací dat, kdy ji obohatíme o další získané sady. Například spojením několika tabulek v relační databázi. [21]

### 4.1.3 Generování dat

Pokud neexistují vhodné datové sady, které lze použít pro trénování, můžeme je vygenerovat buď ručně, nebo automaticky.

Při ručním generování dat jsou pracovníkům zadávány úkoly pro získávání a shromažďování potřebných údajů, ze kterých se vytvoří požadovaný datový soubor. Existuje řada

crowdsourcingových platforem, kde jsou pracovníci odměňováni za řešení zadaných úkolů, jakým může být například označování obrázků a různých dat. [21]

Pro automatické generování dat používáme automatizované techniky. Výhodou tohoto způsobu získávání dat je nízká cena. Jednou z nejjednodušších metod je vygenerovat vzorek na základě rozdělení pravděpodobnosti určitých dat. Pokročilejšími technikami jsou generativní adversariální sítě (GAN) a další aplikačně specifické techniky. [21]

## 4.2 Zdroje datasetů pro strojové učení

Existuje řada platforem, které umožňují nalezení vhodných datových sad pro strojové učení. Nejpoužívanějšími v současnosti jsou UCI Machine Learning Repository a Kaggle, na které se zaměřím podrobněji. Kromě nich existuje řada dalších užitečných služeb. [23]

Google Dataset Search https://datasetsearch.research.google.com/ je vyhledávací službou od Google zaměřenou na hledání volně dostupných datových sad.

OpenML (Open Machine Learning) https://www.openml.org/ je online otevřená vědecká platforma pro strojové učení a sdílení dat. Datové sady jsou pravidelně aktualizovány. Každá je poté automaticky verzována, analyzována a doplněna metadaty pro efektivnější analýzu.

DataHub https://datahub.io/ obsahuje tisíce datových sad strojového učení, zahrnující finanční trhy, makroekonomii, růst populace až po ceny kryptoměn.

Papers with Code https://paperswithcode.com/ je komunitní projekt s otevřenými zdroji dat. Lze vybírat v současnosti z 3937 datových sad, které lze filtrovat podle různých kritérií.

VisualData https://visualdata.io/ je vyhledávač datových sad počítačového vidění.

**Dále lze využít veřejné vládní datové sady:**

data.gov.cz - otevřená data ČR

data.europa.eu - zveřejněná data agentur, institucí a dalších subjektů EU

data.worldbank.org - otevřená data světové banky

### 4.2.1 UCI Machine Learning Repository

Je archiv datasetů, které využívá komunita strojového učení pro analýzu a testování algoritmů strojového učení. Archiv vznikl v roce 1987, když ho vytvořil David Aha společně s dalšími postgraduálními studenty na UC Irvine. Od počátku se stal široce využívaným pedagogy, studenty a výzkumníky po celém světě jako zdroj datových sad pro úlohy strojového učení. V roce 2022 obsahuje 622 datových sad. V repozitáři lze vyhledávat a filtrovat datové sady podle různých charakteristik. [24]

### 4.2.2 Kaggle

Je dceřiná společnost Google LLC. Jedná se o crowdsourcingovou platformu, která umožňuje lidem z celého světa zapojit se a řešit problémy týkající se strojového učení, datové vědy a prediktivní analýzy. Kaggle mohou uživatelé využít k vyhledávání a publikování datových sad, též i k jejich prozkoumávání a vytváření modelů v prostředí Kaggle notebooku, což je obdoba Jupyter Notebooku. Platformu lze také využít ke vzdělávání v oblasti umělé inteligence a strojového učení. [25]

V roce 2021 Kaggle oznámil, že jeho komunita má více než 8 mil. uživatelů ze 194 zemí světa. Skladba komunity je různorodá, od uživatelů, kteří právě začínají až po nejznámější světové výzkumníky.

**Kaggle umožňuje:**

- Publikování, sdílení a analyzování datasetů
- Prohlížení kódů vytvořených notebooků uživateli
- Zakládat soutěže a účastnit se jich
- Diskutovat s ostatními uživateli na fóru
- Absolvovat výukové kurzy

Datasety v Kaggle lze vyhledávat jak pomocí klíčových slov, tak i prostřednictvím mnoha kategorií. Vyhledávání též podporuje filtrování výsledků podle velikosti, typu souborů, nebo licence. Též lze přidávat i vlastní datasety, které lze sdílet veřejně, nebo ponechat soukromé.

Datasety uživatelé využívají k tvorbě modelů v integrované platformě Kaggle notebooku, což je typ Jupyter Notebooku. Využívat lze programovací jazyky Python, nebo R. Vytvořené notebooky lze sdílet s ostatními uživateli a hodnotit je.

Kaggle pořádá též různé soutěže, do kterých se zapojují jednotlivci i týmy. Organizátor soutěže připraví data a popis řešeného problému. Následně účastníci experimentují s vytvořením nejlepšího modelu. Výsledek práce účastníků je veřejně sdílen. Po vyhodnocení pořadatel vyplatí vítězi odměnu výměnou za neomezenou licenci k vyvinutému algoritmu.

Na fóru v Kaggle platformě lze pokládat dotazy sdělovat názory a též jednoduše vyhledávat příspěvky ostatních uživatelů.

Na Kaggle jsou také dostupné kurzy, které jsou zaměřeny na výuku dovedností využitelných při datové analýze na jeho platformě. Jsou dostupné zdarma a po jejich absolvování získá uživatel certifikát, který může dále sdílet.

## 4.3 Bankink Dataset - Marketing targets

Pro praktickou část práce, ve které provedu testování rozhodovacích algoritmů, jsem zvolil Bankink Dataset - Marketing targets dostupný zde: https://www.kaggle.com/prakharrathi25/banking-dataset-marketing-targets. [25]

Budu hledat vhodný model, na jehož základě bychom dokázali predikovat, zda klient uzavře termínovaný klad v bance. K predikci využiji údaje obsažené ve zvoleném datové souboru. Modely budu vytvářet pomocí algoritmů strojového učení, které jsem uvedl v teoretické části práce.

Termínovaný vklad je druh spořícího účtu, na kterém má klient garantovanou úrokovou sazbu vložených finančních prostředků. Jsou sjednávány s klientem na určitou dobu, po kterou není možno peníze vybrat ani vkládat. Termínované vklady jsou jedním z významných zdrojů příjmů pro banku, která má vložené finance klienta po celou dobu vkladu k dispozici a může s nimi libovolně nakládat a investovat je. [1, 25]

Jedním ze způsobů, jak oslovit nové lidi pro založení termínovaného vkladu jsou telefonické marketingové kampaně. Ve zvoleném datové souboru jsou obsaženy údaje získané

v marketingové telefonické kampani portugalské bankovní instituce. Budeme predikovat, zda klient uzavře termínovaný vklad.

Popis všech sloupců obsažených ve zvoleném datovém souboru je uveden v následující tabulce.

Tabulka 1 - Popis sloupců datového souboru

| číslo sloupce | název | popis |
|---|---|---|
| 1 | age | věk klienta |
| 2 | job | typ práce |
| 3 | marital | rodinný stav |
| 4 | education | vzdělání |
| 5 | default | má úvěr v prodlení |
| 6 | balance | zůstatek |
| 7 | housing | úvěr na bydlení |
| 8 | loan | půjčka |
| 9 | contact | typ komunikace |
| 10 | day | poslední den kontaktu v měsíci |
| 11 | month | poslední měsíc kontaktu v roce |
| 12 | duration | doba poslední kontaktu |
| 13 | compaign | počet uskutečněných kontaktů během kampaně |
| 14 | pdays | počet uplynulých dní, kdy byl klient kontaktován |
| 15 | previous | počet kontaktů uskutečněných před touto kampaní |
| 16 | poutcome | výsledek předchozí marketingové kampaně |
| 17 | y | Založil klient termínovaný vklad? |

**Dataset obsahuje 2 soubory:**

- **train.csv** - obsahuje 45211 řádků, 17 sloupců
- **test.csv** - obsahuje 4521 řádků, 17 sloupců

V souboru test.csv je ovšem obsažena podmnožina dat ze souboru train.csv. My ovšem potřebujeme unikátní data, proto budeme pracovat pouze se souborem train.csv, který obsahuje unikátní záznamy. Soubor rozdělíme na testovací a trénovací data.

## 4.4 Příprava a analýza datasetu

Pro testování rozhodovacích algoritmů na Notebooku Jupyter je nejprve potřeba načíst data. K tomu využijeme datový rámec (dataframe) obsažený v knihovně pandas, což je vhodný datový typ pro práci s datovými soubory. Po načtení je potřeba data zkontrolovat a případně upravit do vhodné podoby pro následné testování.

### 4.4.1 Analýza dat

Než začneme s testováním rozhodovacích algoritmů, bude třeba data nejprve analyzovat a případně upravit do vhodné podoby pro následné testování. Podrobný postup analýzy včetně všech výstupů je součástí souboru Jupyter Notebooku přiloženého k této práci.

**Je třeba zjistit:**

- Chybějící hodnoty v datech - všechny záznamy musí obsahovat kompletní data.
- Duplicitní záznamy - všechny záznamy musí být unikátní
- Zda není přítomna multikolinearita
- Konzistenci hodnot ve všech sloupcích

| | age | job | marital | education | default | balance | housing | loan | contact | day | month | duration | campaign | pdays | previous | poutcome | y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 58 | management | married | tertiary | no | 2143 | yes | no | unknown | 5 | may | 261 | 1 | -1 | 0 | unknown | no |
| 1 | 44 | technician | single | secondary | no | 29 | yes | no | unknown | 5 | may | 151 | 1 | -1 | 0 | unknown | no |
| 2 | 33 | entrepreneur | married | secondary | no | 2 | yes | yes | unknown | 5 | may | 76 | 1 | -1 | 0 | unknown | no |
| 3 | 47 | blue-collar | married | unknown | no | 1506 | yes | no | unknown | 5 | may | 92 | 1 | -1 | 0 | unknown | no |
| 4 | 33 | unknown | single | unknown | no | 1 | no | no | unknown | 5 | may | 198 | 1 | -1 | 0 | unknown | no |

*Obrázek 18 - Datový soubor - hlavička*

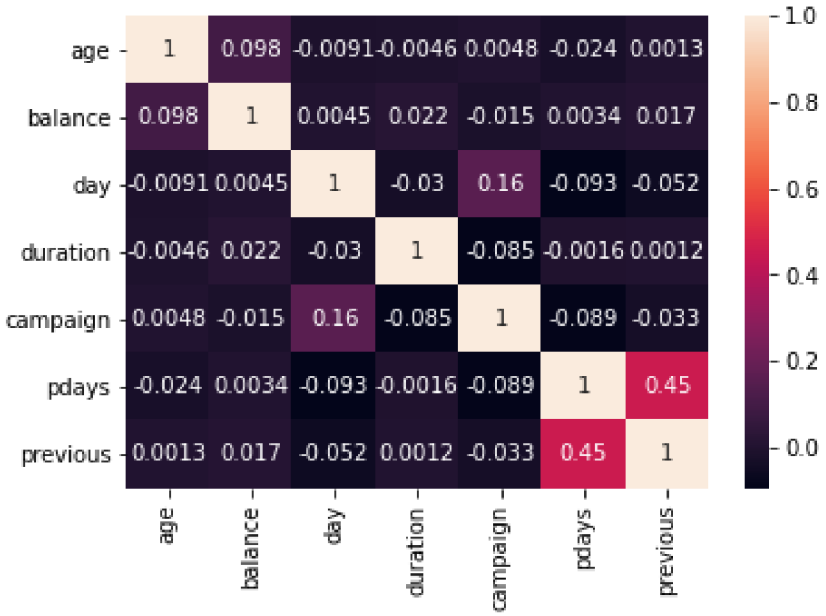
### Chybějící hodnoty v datech a duplicitní záznamy

Analýzou bylo zjištěno, že zdrojová data neobsahují žádné chybějící hodnoty. Duplicitní záznamy též nebyly nalezeny.

### Korelační matice

Korelační matice je jednoduchá tabulka, která nám zobrazuje korelační koeficienty v intervalu <-1, 1> pro různé proměnné. Můžeme z ní zjistit korelaci mezi všemi sloupci v našem datovém souboru.

Pokud je korelační koeficient v absolutní hodnotě větší než 0,8, dochází k multikolinearitě, neboli závislosti hodnot v daných sloupcích. Je to nežádoucí jev, který může negativně ovlivnit kvalitu vytvářeného modelu.




*Obrázek 19 - Korelační matice*

Z vytvořené korelační matice je zřejmé, že v našem datovém souboru se multikolinearita nevyskytuje.

### Konzistence hodnot a validita dat

Kontrolou odlehlých hodnot a validity dat bylo zjištěno, že všechny číselné hodnoty jsou v reálných mezích.

Analýzou bylo zjištěno, že sloupce job, education, contact a poutcome obsahují skupinu hodnot označenou jako unknown (neznámé). U sloupců poutcome a contact se jedná o významnou část obsažených hodnot, proto tyto sloupce budeme ignorovat. Sloupce job a education obsahují neznámé hodnoty pouze v malém množství případů, proto u nich vyřešíme tento problém odstraněním záznamů obsahujících neznámý údaj. Tím dojde u datového souboru ke snížení jeho záznamů z 45211 na 43193.

### 4.4.2 Rozdělení datasetu

Dataset se kterým pracujeme potřebujeme dále rozdělit na 2 skupiny:

- **Trénovací data** - budeme je požívat k naučení zvoleného klasifikačního algoritmu
- **Testovací data** - budou použita pro ověření úspěšnosti naučeného algoritmu na trénovacích datech

Rozdělení dat provedeme v poměru 80% (34554 záznamů) trénovací data, 20% (8639 záznamů) testovací data. Využijeme funkci train_test_split z knihovny scikit-learn.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 34554 entries, 33601 to 2908
Data columns (total 14 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        34554 non-null  int64
 1   job        34554 non-null  object
 2   marital    34554 non-null  object
 3   education  34554 non-null  object
 4   default    34554 non-null  object
 5   balance    34554 non-null  int64
 6   housing    34554 non-null  object
 7   loan       34554 non-null  object
 8   day        34554 non-null  int64
 9   month      34554 non-null  object
 10  duration   34554 non-null  int64
 11  campaign   34554 non-null  int64
 12  pdays      34554 non-null  int64
 13  previous   34554 non-null  int64
dtypes: int64(7), object(7)
memory usage: 4.0+ MB
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8639 entries, 13082 to 12171
Data columns (total 14 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        8639 non-null   int64
 1   job        8639 non-null   object
 2   marital    8639 non-null   object
 3   education  8639 non-null   object
 4   default    8639 non-null   object
 5   balance    8639 non-null   int64
 6   housing    8639 non-null   object
 7   loan       8639 non-null   object
 8   day        8639 non-null   int64
 9   month      8639 non-null   object
 10  duration   8639 non-null   int64
 11  campaign   8639 non-null   int64
 12  pdays      8639 non-null   int64
 13  previous   8639 non-null   int64
dtypes: int64(7), object(7)
memory usage: 1012.4+ KB
```

*Obrázek 20 - Trénovací a testovací data (info)*

### 4.4.3 Konverze kategorických proměnných

Většina algoritmů strojového učení pracuje s čísly. Proto je nutné, aby všechny sloupce obsahovali pouze číselné hodnoty, což je problém u kategorických proměnných. V našem

řešeném příkladu se jedná o sloupce job, marital, education, default, housing, loan a month. Tento problém lze vyřešit několika způsoby. Buď můžeme přiřadit konkrétním slovním názvům unikátní číselnou hodnotu. Nebo můžeme rozdělit kategorické hodnoty do sloupců a přiřadit jim hodnotu 0 (není obsažena v záznamu), nebo 1 (je obsažena v záznamu). V našem případě využijeme funkci get_dummies z knihovny pandas, která nám kategorické hodnoty rozdělí.

U sloupce y, který obsahuje hledané odpovědi (yes / no) provedeme též substituci na číselnou hodnotu:

no: 0 - negativní odpověď

yes: 1 - pozitivní odpověď

## 4.5 Měření výkonu klasifikačního modelu

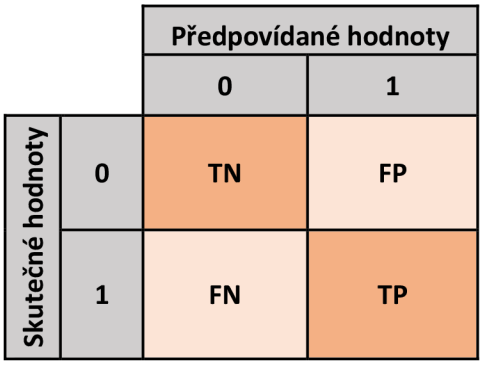
Pro posouzení úspěšnosti predikce vytvořených modelů prostřednictvím algoritmů strojového učení je důležité vyhodnotit jejich přesnost, abychom zjistili, jak jsou úspěšné při vytváření předpovědí v nových situacích.

### 4.5.1 Knihovna scikit-learn

Scikit-learn, též známá jako sklearn je open source knihovna obsahující sadu jednoduchých a efektivních nástrojů pro datovou analýzu a řešení úloh strojového učení v Pythonu. V knihovně jsou obsaženy všechny algoritmy strojového učení, které budu v praktické části využívat. Též jsou v ní zahrnuty užitečné funkce, které nám umožňují zhodnotit úspěšnost aplikace vybraného algoritmu strojového učení. [26]

### 4.5.2 Chybová matice (confusion matrix)

Je to čtvercová matice obsahující skutečnou hodnotu předpovědi ve sloupcích a předpovídanou hodnotu v řádcích. Na základě této matice jsou vypočítány uvedené metriky. [27]




*Obrázek 21 - Chybová matice*

- **TP - True Positive**
  Počet záznamů, které byli předpovězeny jako pozitivní a byli ve skutečnosti označeny jako pozitivní
- **FP - False positive**
  Počet záznamů, které byli předpovězeny jako pozitivní, ale byli ve skutečnosti označeny jako negativní
- **TN - True Negative**
  Počet záznamů, které byli předpovězeny jako negativní a byli ve skutečnosti označeny jako negativní
- **FN - False Negative**
  Počet záznamů, které byli předpovězeny jako negativní, ale byli ve skutečnosti označeny jako pozitivní

### 4.5.3 Klasifikační metriky

Klasifikační metriky používané pro posouzení výkonosti nám pomáhají lépe pochopit silné a slabé stránky zvoleného modelu při vytváření předpovědí. Zahrnují přesnost (accuracy), preciznost (precision), recall a F1-score. V Pythonu budu využívat knihovnu Scikit-learn, která uvedené metriky obsahuje. Pro zobrazení výstupu všech metrik využiji metodu Classification report, která nám zobrazí výsledky všech metrik měření výkonu klasifikačního modelu v přehledné tabulce. [28]

**Přesnost (acuracy)**

Tato metrika nám dává informaci o tom, jak často můžeme očekávat správně předpovězený výsledek z celkového počtu odpovědí v našem modelu.

Je definována jako poměr TP a TN ke všem pozitivním i negativním pozorováním.

Acuracy = (TP + TN) / (TP + FN +TN + FP)

**Preciznost (precision)**

U pozitivních předpovědí určuje, jak je model schopen správně předpovídat pozitiva ze všech pozitivních předpovědí které učinil. Je definován jako poměr TP ke všem pozitivním předpovědím.

Precision (positive) = TP / (FP + TP)

U negativních předpovědí určuje, jak je model schopen správně předpovídat negativa ze všech negativních předpovědí které učinil. Je definován jako poměr TN ke všem negativním předpovědím.

Precision (negative) = TN / (FN + TN)

**Recall**

U pozitivních předpovědí nám říká, jaká je schopnost modelu správně předpovídat pozitiva ze skutečných pozitiv. Je definován jako poměr TP k součtu FN + TP

Recall (positive) = TP / (FN + TP)

U negativních předpovědí nám říká, jaká je schopnost modelu správně předpovídat negativa ze skutečných negativ. Je definován jako poměr TN k součtu FP + TN

Recall (negative) = TN / (FP + TN)

Čím vyšší toto skóre je, tím lépe dokáže náš model správně identifikovat pozitivní i negativní případy. U nevyvážených tříd je užitečným měřítkem úspěšnosti predikce.

**F1-score**

Metrika výkonu reprezentující skóre modelu jako harmonický průměr přesnosti (precision) a recall. Lze spočítat pro pozitivní i negativní předpovědi. Používá se často jako hodnota, která nám sděluje celkové informace o kvalitě výstupu modelu.

F1-score = 2 * Precision Score * Recall Score / (Precision Score + Recall Score)

**Macro average**

Je makroprůměrné skóre, které se vypočítává tak, že se vezme aritmetický (nevážený) průměr ze všech skóre dané metriky.

**Weighted Average**

Je vážené průměrné skóre, které se vypočítá tak, že vynásobíme poměr výskytu u každé třídy s vypočítanou metrikou. Všechny hodnoty poté sečteme.

**Support**

V klasifikačním reportu udává počet záznamů patřících k dané třídě.

## 4.6 Sestavování modelů

V této části se budu zabývat sestavováním modelů pomocí různých algoritmů popsaných v teoretické části na zvoleném datovém souboru. Pro zhodnocení úspěšnosti a přesnosti predikce modelů budu měřit jejich výkon pomocí dříve uvedených metrik.

Při trénování moedlů budeme využívat křížovou validaci, kterou funkce z knihovny scikit-learn standardně používají.

Na zvoleném datovém souboru budu zkoumat za použití vytvořených modelů pomocí zvolených algoritmů, zda klient banky na základě zjištěných údajů uzavře termínovaný vklad. Jelikož se vklad rozhodne uzavřít pouze menšina s oslovených klientů, bude výsledná přesnost modelu ovlivněna z velké části správnou předpovědí negativních odpovědí a z malé části správnou předpovědí pozitivních odpovědí. U každého modelu uvedu vygenerovanou výslednou chybovou matici a report s výsledky predikce naučeného modelu na testovací množině dat. Ve vytvořených modelech budu hodnotit hlavně schopnost předpovídat pozitivní předpovědi (yes - 1), neboli zda klient uzavře termínovaný vklad. Pro hodnocení budu využívat klasifikační metriky. Též budu měřit čas potřebný k vytvoření modelu. Na závěr u každé metody provedu její celkové zhodnocení.

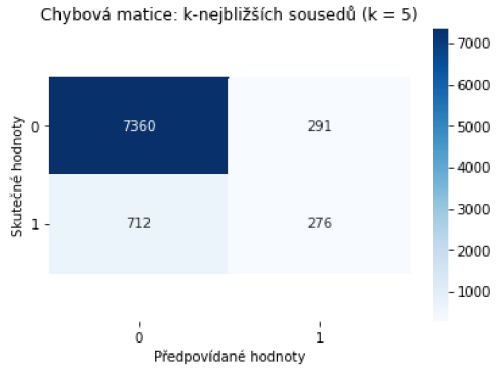
### 4.6.1 Metoda k-nejbližších sousedů (K-Nearest-Neighbor)

V této metodě je třeba zvolit vhodný počet nejbližších sousedů (k), na kterém závisí správná přesnost klasifikace. Při nízké hodnotě budou mít odlehlé hodnoty velký vliv na výsledek.

Při vysoké hodnotě bude sestavení modelu výpočetně náročnější, hranice rozhodování hladší, ale vychýlení vyšší. Zvolil jsem několik různých hodnot k-nejbližších sousedů.

**Metoda nejbližších sousedů - 5 sousedů**

U modelu metody nejbližších sousedů se zvoleným počtem 5 byl čas jeho vytvoření 10,33 sekund. Výsledná přesnost modelu dosahuje 88%. Pozitivní předpovědi činí 49% úspěšnosti podle metriky precision a 28/% úspěšnosti podle metriky recall.

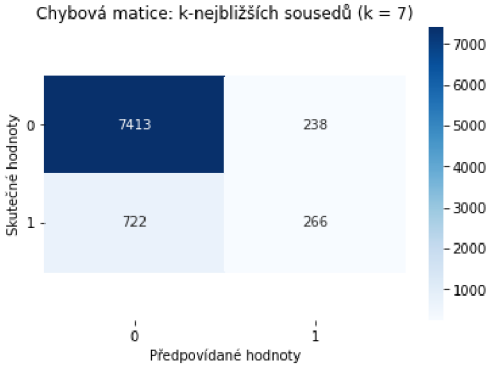



*Obrázek 22 - Metoda k-nejbližších sousedů (k=5)*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.91 | 0.96 | 0.94 | 7651 |
| 1 | 0.49 | 0.28 | 0.35 | 988 |
| accuracy | | | 0.88 | 8639 |
| macro avg | 0.70 | 0.62 | 0.65 | 8639 |
| weighted avg | 0.86 | 0.88 | 0.87 | 8639 |

**Metoda nejbližších sousedů - 7 sousedů**

Model metody nejbližších sousedů se zvoleným počtem 7 byl vytvořen za 6,17 sekundy. Přesnost předpovědi činí 89%, přičemž podle metriky precision je kvalita pozitivních předpovědí 53% a podle metriky recall 27%.

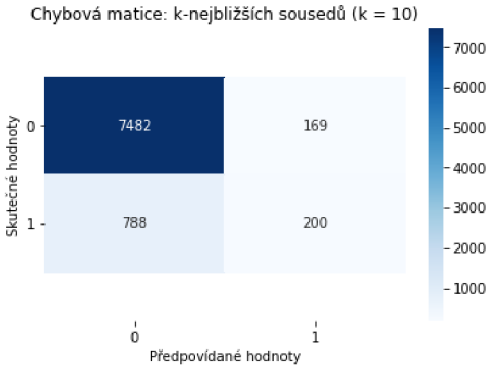



*Obrázek 23 - Metoda k-nejbližších sousedů (k=7)*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.91 | 0.97 | 0.94 | 7651 |
| 1 | 0.53 | 0.27 | 0.36 | 988 |
| accuracy | | | 0.89 | 8639 |
| macro avg | 0.72 | 0.62 | 0.65 | 8639 |
| weighted avg | 0.87 | 0.89 | 0.87 | 8639 |

### Metoda nejbližších sousedů - 10 sousedů

Model metody nejbližších sousedů se zvoleným počtem 10 byl vytvořen za 6,05 sekundy. Výsledná přesnost modelu dosahuje 89%, přičemž kvalita výstupu pozitivních předpovědí modelu podle metriky precision činí 54% a podle recall 20%.




*Obrázek 24 - Metoda k-nejbližších sousedů (k=10)*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.90 | 0.98 | 0.94 | 7651 |
| 1 | 0.54 | 0.20 | 0.29 | 988 |
| accuracy | | | 0.89 | 8639 |
| macro avg | 0.72 | 0.59 | 0.62 | 8639 |
| weighted avg | 0.86 | 0.89 | 0.87 | 8639 |

**Zhodnocení metody k-nejbližších sousedů**

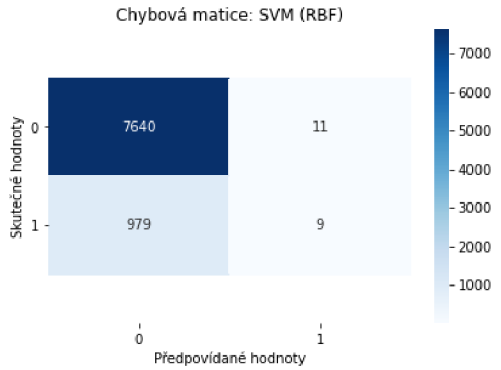
U všech vytvořených modelů metodou nejbližších sousedů bylo dosaženo 88-89% přesnosti. U modelu s 10 sousedy však již dochází ke snižování kvality pozitivních předpovědí podle metriky recall. Proto bych za vhodný model pro predikci podle této metody určil model se 7 sousedy.

### 4.6.2 Metoda SVM

V tomto algoritmu je důležitým parametrem zvolení vhodného typu kernelu. Zvolil jsem nejprve RBF kernel, který je doporučován jako vhodný pro většinu řešených problémů. Poté jsem zvolil lineární kernel.

**Metoda SVM - kernel RBF**

Model SVM se zvoleným RBF kernelem byl hotov za 41,8 sekundy. Výsledná přesnost modelu dosahuje 89%. Kvalita výstupu pozitivních předpovědí modelu je 45% podle metriky precision a 1% podle metriky recall.

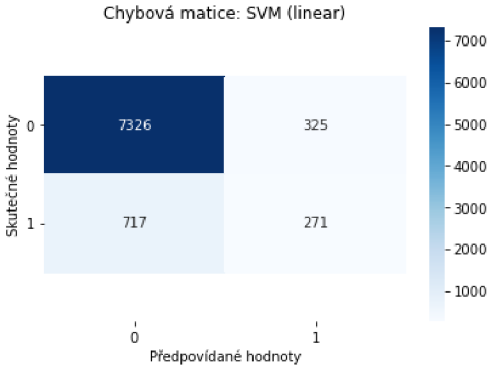



*Obrázek 25 - Metoda SVM - kernel RBF*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.89 | 1.00 | 0.94 | 7651 |
| 1 | 0.45 | 0.01 | 0.02 | 988 |
| accuracy | | | 0.89 | 8639 |
| macro avg | 0.67 | 0.50 | 0.48 | 8639 |
| weighted avg | 0.84 | 0.89 | 0.83 | 8639 |

### Metoda SVM - kernel lineární

Sestavování modelu SVM se zvoleným lineárním kernelem trvalo 3 hodiny a 32 minut. Výsledná přesnost modelu dosahuje 88%. Pozitivní předpovědi činí 45% úspěšnosti podle metriky precision a 27/% úspěšnosti podle metriky recall.




*Obrázek 26 - Metoda SVM - kernel lineární*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.91 | 0.96 | 0.93 | 7651 |
| 1 | 0.45 | 0.27 | 0.34 | 988 |
| accuracy | | | 0.88 | 8639 |
| macro avg | 0.68 | 0.62 | 0.64 | 8639 |
| weighted avg | 0.86 | 0.88 | 0.87 | 8639 |

### Zhodnocení metody SVM

Model SVM vytvořený pomocí RBF kernelu byl sestaven za 41,7 sekund, kde přesnost modelu dosahovala sice 89%, ale kvalita pozitivních předpovědí podle recall činila pouze 1%. U modelu SVM sestaveným pomocí lineárního kernelu činila doba vytvoření modelu 3

hodiny a 32 minut, přičemž za tuto dlouho dobu jsme dostali model s neuspokojivými výsledky.

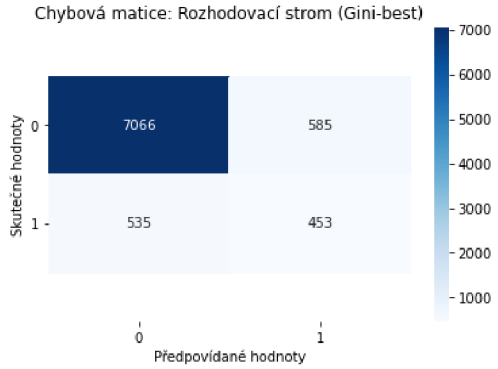
Vzhledem k uvedeným nedobrým výsledkům RBF kernelu a vysoké časové náročnosti lineárního kernelu bych tuto metodu nedoporučil pro predikci zvoleného datasetu.

### 4.6.3 Rozhodovací stromy (Decision Trees)

U Algoritmu rozhodovacího stromu je možno zvolit ze dvou funkčních kritérií výběru optimálního rozdělení prvků pro výběr uzlu. Jsou jimi „Gini“ nebo „Entropy“. Strategii rozdělení každého uzlu lze ovlivnit zvolením „best“ pro výběr nejlepšího rozdělení, nebo random pro výběr nejlepšího náhodného rozdělení.

**Rozhodovací strom - Gini-best**

Model rozhodovacího stromu s nastaveným kritériem rozdělení Gini a strategií best byl vytvořen během zlomku vteřiny. Výsledná přesnost modelu dosahuje 87%, přičemž podle metriky precision je kvalita pozitivních předpovědí 44% a podle metriky recall 46%.

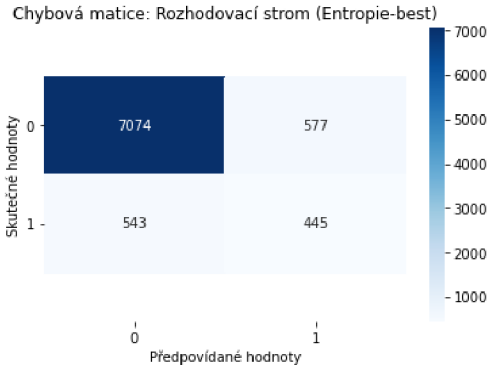



*Obrázek 27 - Rozhodovací strom (Gini-best)*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.93 | 0.92 | 0.93 | 7651 |
| 1 | 0.44 | 0.46 | 0.45 | 988 |
| accuracy | | | 0.87 | 8639 |
| macro avg | 0.68 | 0.69 | 0.69 | 8639 |
| weighted avg | 0.87 | 0.87 | 0.87 | 8639 |

**Rozhodovací strom - Entropie-best**

U modelu rozhodovacího stromu s nastaveným kritériem rozdělení Entropie a best strategií byl čas vytvoření okamžitý. Výsledná přesnost modelu dosahuje 87%. Pozitivní předpovědi činí 44% úspěšnosti podle metriky precision a 45/% úspěšnosti podle metriky recall.




*Obrázek 28 - Rozhodovací strom (Entropie-best)*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.93 | 0.92 | 0.93 | 7651 |
| 1 | 0.44 | 0.45 | 0.44 | 988 |
| accuracy | | | 0.87 | 8639 |
| macro avg | 0.68 | 0.69 | 0.68 | 8639 |
| weighted avg | 0.87 | 0.87 | 0.87 | 8639 |

**Zhodnocení metody rozhodovacího stromu**

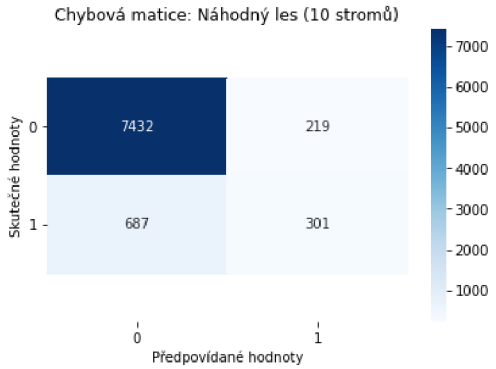
U všech modelů rozhodovacího stromu byla dosažena přesnost 87%. Kritérium rozdělení prvků nemělo zásadní vliv na úspěšnost předpovědi.

### 4.6.4 Náhodný les (Random Forest)

V tomto algoritmu je třeba zvolit vhodný počet stromů v lese. Algoritmus jsem spustil s několika počty stromů, přičemž platí, že čím vyšší hodnota, tím náročnější byl výpočet a delší doba trvání sestavení modelu.

### Náhodný les - 10 stromů

Vytvoření modelu algoritmu náhodného lesa v počtu 10 stromů proběhlo prakticky ihned během zlomku vteřiny. Výsledná přesnost modelu dosahuje 90%, přičemž podle metriky precision je kvalita pozitivních předpovědí 58% a podle metriky recall 30%.

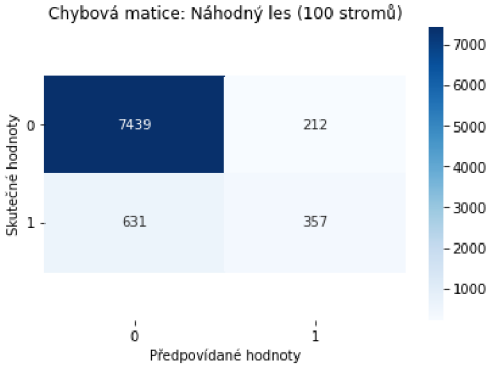



*Obrázek 29 - Náhodný les (10 stromů)*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.92 | 0.97 | 0.94 | 7651 |
| 1 | 0.58 | 0.30 | 0.40 | 988 |
| accuracy | | | 0.90 | 8639 |
| macro avg | 0.75 | 0.64 | 0.67 | 8639 |
| weighted avg | 0.88 | 0.90 | 0.88 | 8639 |

### Náhodný les - 100 stromů

U modelu algoritmu náhodného lesa v počtu 100 stromů byl výpočet modelu hotov za 3,52 sekundy. Výsledná přesnost modelu dosahuje 90%, přičemž podle metriky precision je kvalita pozitivních předpovědí 63% a podle metriky recall 36%.

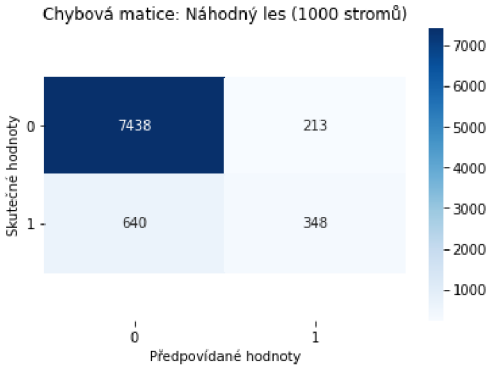



*Obrázek 30 - Náhodný les (100 stromů)*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.92 | 0.97 | 0.95 | 7651 |
| 1 | 0.63 | 0.36 | 0.46 | 988 |
| accuracy | | | 0.90 | 8639 |
| macro avg | 0.77 | 0.67 | 0.70 | 8639 |
| weighted avg | 0.89 | 0.90 | 0.89 | 8639 |

### Náhodný les - 1000 stromů

Pro model algoritmu náhodného lesa v počtu 1000 stromů trval výpočet modelu 36,2 sekund. Výsledná přesnost modelu dosahuje 90%, přičemž podle metriky precision je kvalita pozitivních předpovědí 62% a podle metriky recall 35%.




*Obrázek 31 - Náhodný les (1000 stromů)*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.92 | 0.97 | 0.95 | 7651 |
| 1 | 0.62 | 0.35 | 0.45 | 988 |
| accuracy | | | 0.90 | 8639 |
| macro avg | 0.77 | 0.66 | 0.70 | 8639 |
| weighted avg | 0.89 | 0.90 | 0.89 | 8639 |

**Zhodnocení metody náhodného lesa**

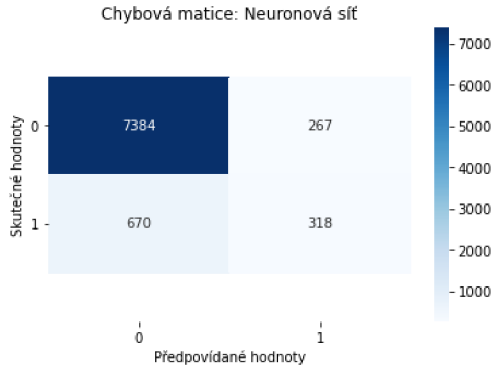
U všech vytvořených modelů pomocí metody náhodného lesa bylo dosaženo 90% přesnosti. Jako nejlepší z těchto modelů pro praktické využití bych zvolil model s nastaveným počtem 100 stromů, u kterého je podle precision i recall lepší predikce pozitivních předpovědí, než v případě modelu s 10 stromy v rozumném čase.

### 4.6.5 Neuronové sítě (Neural networks)

U algoritmu neuronové sítě je důležitým parametrem určení počtu skrytých vrstev, pro které lze volit z několika aktivačních funkcí.

**Neuronová síť - skryté vrstvy (100)**

Vytvoření modelu neuronové sítě s jednou skrytou vrstvou se 100 neurony trvalo 3,45 sekundy. Přesnost předpovědi činila 89%. Pozitivní předpovědi činí 54% úspěšnosti podle metriky precision a 32/% úspěšnosti podle metriky recall.

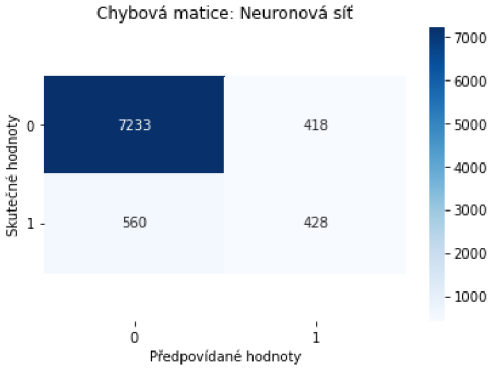



*Obrázek 32 - Neuronová síť (100)*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.92 | 0.97 | 0.94 | 7651 |
| 1 | 0.54 | 0.32 | 0.40 | 988 |
| accuracy | | | 0.89 | 8639 |
| macro avg | 0.73 | 0.64 | 0.67 | 8639 |
| weighted avg | 0.87 | 0.89 | 0.88 | 8639 |

**Neuronová síť - skryté vrstvy (100, 100, 100)**

Model neuronové sítě se třemi skrytými vrstvami v počtu sta neuronů byl vytvořen za 2 minuty a 34 vteřin. Přesnost modelu dosahovala 89%, přičemž podle metriky precision je kvalita pozitivních předpovědí 51% a podle metriky recall 43%.




*Obrázek 33 - Neuronová síť (100, 100, 100)*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.93 | 0.95 | 0.94 | 7651 |
| 1 | 0.51 | 0.43 | 0.47 | 988 |
| accuracy | | | 0.89 | 8639 |
| macro avg | 0.72 | 0.69 | 0.70 | 8639 |
| weighted avg | 0.88 | 0.89 | 0.88 | 8639 |

**Zhodnocení neuronové sítě**

Model neuronové sítě se třemi vrstvami po sto neuronech dosáhl podle recall větší přesnosti pozitivních výsledků než model s jednou vrstvou. I přes časově náročnější výpočet bych tento typ modelu neuronové sítě pro daný dataset zvolil za lepší z těchto dvou testovaných modelů.

## 4.7 Hodnocení vytvořených modelů

Výsledky důležitých ukazatelů z reportů všech vytvořených modelů jsou uvedeny v následující tabulce.

Tabulka 2 - Výsledky předpovědí modelů

| Metoda | Parametry | čas | Předpověď | Precision | Recall | F1-score | Přesnost |
|---|---|---|---|---|---|---|---|
| KNN | k=5 | 6,33s | 0 | 0,91 | 0,96 | 0,94 | 0,88 |
| | | | 1 | 0,49 | 0,28 | 0,35 | |
| | k=7 | 6,17s | 0 | 0,91 | 0,97 | 0,94 | 0,89 |
| | | | 1 | 0,53 | 0,27 | 0,36 | |
| | k=10 | 6,05s | 0 | 0,9 | 0,98 | 0,94 | 0,89 |
| | | | 1 | 0,54 | 0,2 | 0,29 | |
| SVM | kernel: RBF | 41,8s | 0 | 0,89 | 1 | 0,94 | 0,89 |
| | | | 1 | 0,45 | 0,01 | 0,02 | |
| | kernel: Lineární | 3h 32m | 0 | 0,91 | 0,96 | 0,93 | 0,88 |
| | | | 1 | 0,45 | 0,27 | 0,34 | |
| DT | krit. - Gini | 260ms | 0 | 0,93 | 0,92 | 0,93 | 0,87 |
| | | | 1 | 0,44 | 0,46 | 0,45 | |
| | krit. - Entropie | 248ms | 0 | 0,93 | 0,92 | 0,93 | 0,87 |
| | | | 1 | 0,44 | 0,45 | 0,44 | |
| RF | stromů: 10 | 358ms | 0 | 0,92 | 0,97 | 0,94 | 0,9 |
| | | | 1 | 0,58 | 0,3 | 0,4 | |
| | stromů: 100 | 3,52s | 0 | 0,92 | 0,97 | 0,95 | 0,9 |
| | | | 1 | 0,63 | 0,36 | 0,46 | |
| | stromů: 1000 | 36,2s | 0 | 0,92 | 0,97 | 0,95 | 0,9 |
| | | | 1 | 0,62 | 0,35 | 0,45 | |
| NS | neuronů: 100 | 3,45s | 0 | 0,92 | 0,97 | 0,94 | 0,89 |
| | | | 1 | 0,54 | 0,32 | 0,4 | |
| | neuronů: (100, 100, 100) | 2min. 34s. | 0 | 0,93 | 0,95 | 0,94 | 0,89 |
| | | | 1 | 0,51 | 0,43 | 0,47 | |

Z výsledků úspěšnosti modelů předpovídat rozhodování na testovací množině dat lze usoudit, vhodnost jednotlivých algoritmů pro tvorbu modelů předpovědí na zkoumaném souboru dat.

Přesnost předpovědí všech vytvořených modelů se pohybovala mezi 87% - 90% a byla podle předpokladů silně ovlivněna negativními odpověďmi, kterých bylo nejvíce z celkového počtu.

Nejhůře si vedli modely vtvořené pomocí algoritmů SVM a metody k-nejbližších sousedů (KNN), čemuž napovídá i jejich nízké F1-score. U SVM modelu se zvoleným RBF kernelem bylo dosaženo pouhé 1% pozitivní předpovědi podle metriky recall a při zvolení lineárního kernelu trval čas jeho vytvoření přes 3 hodiny s neuspokojivými výsledky. U metody k-nejbližších sousedů byla nízká úspěšnost pozitivní předpovědi především podle metriky recall.

Dobře si vedly modely vytvořené pomocí algoritmů rozhodovacího stromu (DT) a náhodného lesa (RF), jelikož dokáží v rychlém čase vytvořit vhodný model podávající dobré výsledky. Modely rozhodovacího stromu dosahovaly vysokých hodnot metriky recall u pozitivních předpovědí, ale nižší hodnotou precision. Modely náhodného lesa dosahovaly nejvyšších hodnot precision u pozitivních předpovědí, přičemž hodnota ukazatele recall byla též dobrá. U Modelu neuronových sítí se třemi skrytými vrstvami v počtu 100 neuronů bylo dosaženo dobrých výsledků podle precision i recall metriky, nevýhodou byla delší doba potřebná k vytvoření modelu.

# 5 Závěr

Aplikace notebooku Jupyter je velice užitečný nástroj použitelný pro datovou vědu. Vděčí za to přehlednému rozhraní, podpoře mnoha jazyků a spoustě knihoven, které umožňují nejen datovou analýzu, ale i vizualizaci zjištěných výsledků. Proto je vhodný nejen pro datovou analýzu ale i jako výukový nástroj.

Hlavním účelem této práce bylo demonstrovat, jak lze notebook Jupyter využít pro datovou analýzu a řešení rozhodovacích problémů pomocí algoritmů strojového učení. K tomu je zapotřebí znát nejen práci v jeho prostředí, ale mít i teoretické znalosti z oblasti řešené problematiky. Všechny tyto aspekty jsou v práci zahrnuty.

V teoretické části práce je nejprve představen Jupyter Notebook, vysvětlena jeho architektura i práce s ním. Další kapitola objasňuje problematiku týkající se umělé inteligence a strojového učení. V navazující kapitole jsou vysvětleny algoritmy, které jsou použity v praktické části pro sestavování modelů.

V praktické části diplomové práce jsem se věnoval vypracování metodiky řešení úloh strojového učení na Jupyter Notebooku. Nejprve jsem řešil možnosti získání potřebných dat pro sestavování modelů. Následně jsem vybral datový soubor obsahující zjištěné údaje klientů banky a provedl jeho analýzu. Poté jsem za pomoci vybraných algoritmů strojového učení sestavil několik modelů, na základě kterých by se dalo předpovědět, zda klient uzavře termínovaný vklad. U každého modelu jsem zhodnotil úspěšnost jeho předpovědi a následně provedl jejich souhrnné hodnocení.

Nejlépe si vedly modely vytvořené pomocí algoritmů rozhodovacích stromů, náhodného lesa a neuronových sítí.

U modelů vytvořených pomocí algoritmu rozhodovacího stromu bylo dosaženo vysoké hodnoty recall. Proto je použití těchto modelů vhodné, pokud požadujeme mít co nejméně lidí, kteří byli falešně označeni, že nemají o termínovaný vklad zájem.

U modelů vytvořených pomocí algoritmu náhodného lesa bylo dosaženo vysoké hodnoty precision. Tyto modely jsou vhodné pro použití, jestliže chceme oslovit co nejméně lidí, kteří ve skutečnosti nemají o termínovaný vklad zájem.

Model vytvořený pomocí algoritmu neuronové sítě se třemi skrytými vrstvami po sto neuronech podává dobré výsledky podle precision i recall skóre. Jeho nevýhodou je delší doba potřebná k vytvoření tohoto modelu.

Algoritmy z knihovny Scikit-learn, které jsem využíval, mají mnoho různých parametrů, kterými lze ovlivnit sestavování modelů. U modelů, jež byli vytvořeny, bylo experimentováno se základními parametry jednotlivých algoritmů, které by měli nejvýrazněji ovlivňovat sestavování výsledných modelů a dávat hrubou představu o jejich celkové úspěšnosti na obdobné typy příkladů.

Metodologie řešení popsaná v této diplomové práci může nadále sloužit jako návod pro řešení obdobných typů úloh. Využitelná může být i pro jiné bankovní instituce, kde je ale třeba zajistit, aby modely byli sestavovány na aktuálních reálných datech dané instituce. Pomocí volitelných parametrů, které jsou součástí použitých metod, lze vyladit výsledné modely a ještě o něco zpřesnit jejich předpovědi.

# 6 Seznam použitých zdrojů


[1] Wikipedia [online]. San Francisco [cit. 2022-01-10]. Dostupné z: https://www.wikipedia.org/

[2] TIŠNOVSKÝ, Pavel. Jupyter Notebook – nástroj pro programátory, výzkumníky i lektory. ROOT.CZ [online]. 2020 [cit. 2022-01-10]. Dostupné z: https://www.root.cz/clanky/jupyter-notebook-nastroj-pro-programatory-vyzkumniky-i-lektory/

[3] Jupyter Project Documentation: Architecture. *Jupyter* [online]. [cit. 2022-01-10]. Dostupné z: https://docs.jupyter.org/en/latest/projects/architecture/content-architecture.html

[4] SLOUKA, David. AI vs ML: co je umělá inteligence a co strojové učení? [online]. [cit.2022-01-10] Dostupné z: https://www.computerworld.cz/clanky/ai-vs-ml-co-je-umela-inteligence-a-co-strojove-uceni/

[5] KAĎOUSKOVÁ, Barbora. UMĚLÁ INTELIGENCE, HLUBOKÉ A STROJOVÉ UČENÍ, V ČEM JE ROZDÍL. *Rascasone* [online]. 2021 [cit. 2022-01-10]. Dostupné z: https://www.rascasone.com/cs/blog/umela-inteligence-strojove-hluboke-uceni-rozdil

[6] KAĎOUSKOVÁ, Barbora. CO JE STROJOVÉ UČENÍ A JAK SOUVISÍ S UMĚLOU INTELIGENCÍ?. *Rascasone* [online]. 2021 [cit. 2022-03-11]. Dostupné z: https://www.rascasone.com/cs/blog/strojove-uceni-ml-metody-klasifikace

[7] Algoritmy strojového učení. *Microsoft Azure* [online]. [cit. 2022-03-11]. Dostupné z: https://azure.microsoft.com/cs-cz/overview/machine-learning-algorithms/

[8] MATOUŠEK, Václav. Úvod do znalostního inženýrství: Strojové učení. *Katedra informatiky a výpočetní techniky ZČU* [online]. Plzeň [cit. 2022-03-11]. Dostupné z: https://www.kiv.zcu.cz/studies/predmety/uzi/Folie_ZS/Stroj_uceni.pdf

[9] Windows Machine Learning [online]. 2021 [cit.2022-03-11]. Dostupné z: https://docs.microsoft.com/en-us/windows/ai/windows-ml/

[10] ŠTOL, Jan. *Strojové učení s využitím metody transfer learning*. Hradec Králové, 2019, 72 s. Dostupné také z: https://theses.cz/id/idr9ob/STAG93694.pdf. Diplomová práce. Univerzita Hradec Králové. Vedoucí práce Karel Mls.

[11] Neuronové sítě a deep learning v Pythonu: Lekce 8 - Rozhodovací stromy v Pythonu. *Itnetwork.cz* [online]. [cit. 2022-03-11]. Dostupné z: https://www.itnetwork.cz/python/neuronove-site/rozhodovaci-stromy-v-pythonu

[12] AZNAR, Pablo. Decision Trees: Gini vs Entropy [online]. 2020 [cit.2022-03-11]. Dostupné z: https://quantdare.com/decision-trees-gini-vs-entropy/

[13] Strojové učení: Klasifikace metodou nejbližšího souseda. *Elements of AI* [online]. [cit. 2022-03-11]. Dostupné z: https://course.elementsofai.com/cs/4/2

[14] JANOUŠOVÁ, Eva, Jiří HOLČÍK, Danka HARUŠTIAKOVÁ, Simona LITTNEROVÁ a Jiří JARKOVSKÝ. Analýza a hodnocení biologických dat: Vícerozměrné metody pro analýzu a klasifikaci dat. *Matematická biologie a biomedicína* [online]. [cit. 2022-03-11]. Dostupné z: https://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--vicerozmerne-metody-pro-analyzu-dat

[15] DONGES, Niklas. A Complete Guide to the Random Forest Algorithm. *Built In* [online]. 2021 [cit. 2022-03-11]. Dostupné z: https://builtin.com/data-science/random-forest-algorithm

[16] E R, Sruthi. Understanding Random Forest. *Analytics Vidhya* [online]. 2021 [cit. 2022-03-11]. Dostupné z: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

[17] VOLNÁ, Eva. Neuronové sítě 1: *Studijní materiály pro distanční kurz: Neuronové sítě 1* [online]. Druhé. Ostrava: Ostravská univerzita v Ostravě, 2008 [cit. 2022-03-11]. Dostupné z: https://web.osu.cz/~Volna/Neuronove_site_skripta.pdf

[18] DURČÁK, Pavel. Neuronové sítě a princip jejich fungování. *NaPočítači.cz* [online]. 2017 [cit. 2022-03-11]. Dostupné z: https://www.napocitaci.cz/33/neuronove-site-a-princip-jejich-fungovani-uniqueidgOkE4NvrWuNY54vrLeM670eFNQh552VdDDulZX7UDBY/

[19] Deep Learning 101: Beginners Guide to Neural Network. *Analytics Vidhya* [online]. 2021 [cit. 2022-03-11]. Dostupné z: https://www.analyticsvidhya.com/blog/2021/03/basics-of-neural-network/

[20] VESELÝ, Arnošt. *ARTIFICIAL INTELLIGENCE*. Praha, 2016, 94 s.

[21] ROH, Yuji, Geon HEO a Steven EUIJONG WHANG. *A Survey on Data Collection for Machine Learning* [online]. 2019 [cit. 2022-03-11]. Dostupné z: https://arxiv.org/pdf/1811.03402.pdf

[22] Anaconda [online]. [cit.2022-03-11]. Dostupné z: https://www.anaconda.com/

[23] RIZZOLI, Alberto. 65+ Best Free Datasets for Machine Learning. *V7* [online]. 2022 [cit. 2022-03-28]. Dostupné z: https://www.v7labs.com/blog/best-free-datasets-for-machine-learning

[24] UCI Machine Learning Repository [online]. [cit.2022-03-11]. Dostupné z: https://archive.ics.uci.edu/ml/index.php

[25] Kaggle [online]. [cit.2022-03-11]. Dostupné z: https://www.kaggle.com/

[26] SCIKIT-LEARN developers. *Scikit-learn* [online]. [cit.2022-03-11]. Dostupné z: https://scikit-learn.org/

[27] KUMAR, Ajitesh. Confusion Matrix Explained with Python Code Examples. *Vitalflux.com* [online]. 2019 [cit. 2022-03-11]. Dostupné z: https://vitalflux.com/models-confusion-matrix-examples/

[28] KUMAR, Ajitesh. Accuracy, Precision, Recall & F1-Score – Python Examples. *Vitalflux.com* [online]. 2022 [cit. 2022-03-11]. Dostupné z: https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/

[29] Co je umělá inteligence a jak ji využíváme?. *Evropský parlament* [online]. Praha, 2020 [cit. 2022-03-11]. Dostupné z: https://www.europarl.europa.eu/news/cs/headlines/society/20200827STO85804/umela-inteligence-definice-a-vyuziti

[30] BERKA, Petr. Rozhodovací stromy. *Sorry.vse.cz* [online]. [cit. 2022-03-11]. Dostupné z: https://sorry.vse.cz/~berka/docs/izi456/kap_5.1.pdf

[31] ROSSANT, Cyrille. IPython Cookbook, Second Edition: Introducing JupyterLab. *IPython Cookbook* [online]. 2018 [cit. 2022-03-11]. Dostupné z: https://ipython-books.github.io/36-introducing-jupyterlab/

[32] BAHETI, Pragati. What is Overfitting in Deep Learning and How to Avoid It. *V7* [online]. 2022 [cit. 2022-03-11]. Dostupné z: https://www.v7labs.com/blog/overfitting

[33] WLODARCZAK, Peter. *Machine Learning and its Applications*. Boca Raton: CRC Press, 2019 [cit. 2022-03-11]. 1st Edition, 1. edice. ISBN 9780429448782. Dostupné z: https://doi-org.infozdroje.czu.cz/10.1201/9780429448782

[34] GALEA, Alex. *Beginning Data Science with Python and Jupyter*. Birmingham: Pact Publishing, 2018 [cit. 2022-03-11]. ISBN 9781789532029.

[35] Cross-validation: evaluating estimator performance. *Scikit-learn* [online]. [cit. 2022-03-11]. Dostupné z: https://scikit-learn.org/stable/modules/cross_validation.html

[36] JANOUŠOVÁ, Eva, Jiří HOLČÍK, Danka HARUŠTIAKOVÁ, Simona LITTNEROVÁ a Jiří JARKOVSKÝ. Vícerozměrné metody pro analýzu a klasifikaci dat: Metoda nejbližšího souseda. *Matematická biologie a biomedicína* [online]. [cit. 2022-03-26]. Dostupné z: https://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--vicerozmerne-metody-pro-analyzu-dat--klasifikace--klasifikace-podle-minimalni-vzdalenosti--princip-klasifikace-podle-minimalni-vzdalenosti--metoda-nejblizsiho-souseda

[37] JANOUŠOVÁ, Eva, Jiří HOLČÍK, Danka HARUŠTIAKOVÁ, Simona LITTNEROVÁ a Jiří JARKOVSKÝ. Vícerozměrné metody pro analýzu a klasifikaci dat: Lineární verze metody podpůrných vektorů – lineárně separabilní třídy. *Matematická biologie a biomedicína* [online]. [cit. 2022-03-26]. Dostupné z: https://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--vicerozmerne-metody-pro-analyzu-dat--klasifikace--klasifikace-pomoci-hranic-v-obrazovem-prostoru-flda-svm-linearni-a-nelinearni--metoda-podpurnych-vektoru--linearni-verze-metody-podpurnych-vektoru-linearne-separabilni-tridy