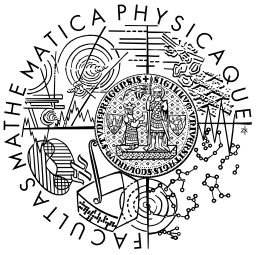


MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

DIPLOMOVÁ PRÁCE

Vojtěch Herrmann

Moderní predikční metody pro finanční časové řady

Katedra pravděpodobnosti a matematické statistiky

Vedoucí diplomové práce:	RNDr. Radek Hendrych, Ph.D.
Studijní program:	Matematika
Studijní obor:	Pravděpodobnost, matematická statistika a ekonometrie

Praha 2021

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V Praze dne 7. 1. 2021 .....

Vojtěch Herrmann

Rád bych na tomto místě vyjádřil poděkování vedoucímu této diplomové práce, RNDr. Radku Hendrychovi, Ph.D., jednak za vypsání velmi zajímavého tématu, jednak za velmi odborné a cenné rady a v neposlední řadě pak za příkladný smysl pro preciznost při vedení a kontrole celé práce.

Název práce: Moderní predikční metody pro finanční časové řady

Autor: Vojtěch Herrmann

Katedra: Katedra pravděpodobnosti a matematické statistiky

Vedoucí diplomové práce: RNDr. Radek Hendrych, Ph.D., Katedra pravděpodobnosti a matematické statistiky

Abstrakt: Tato práce se zabývá porovnáním vybraného tradičního přístupu k modelování a predikci časových řad (ARIMAX model) s vybraným moderním přístupem – pomocí gradientně boostovaných rozhodovacích stromů implementovaných v rámci knihovny XGBoost. V první části práce je představen teoretický rámec supervizovaného učení, modelu ARIMAX a gradientního boostingu v kontextu rozhodovacích stromů. V druhé části jsou identifikovány modely ARIMAX a XGBoost, které oba predikují konkrétní časovou řadu – denní zobchodovaný objem indexu S&P 500, což je pro řadu odvětví velmi důležitá úloha. Dále jsou porovnány výsledky jednotlivých přístupů, jsou popsány výhody XGBoost, které pravděpodobně vedly k jeho lepším výsledkům v této konkrétní simulační studii a je ukázána důležitost optimalizace hyperparametrů. Na závěr jsou metody porovnány i po praktické stránce, speciálně co do výpočetní náročnosti. V poslední části práce je odvozena teorie hybridního modelu a navrženy algoritmy pro nalezení optimálního hybridního modelu. Ty jsou následně aplikovány na problém predikce objemu. Optimální hybridní model kombinuje modely ARIMAX a XGBoost a dosahuje lepších výsledků než jednotlivé modely samostatně.

Klíčová slova: Akciový trh, ARIMAX, Hybridní model, Predikce, XGBoost

Title: Modern predictive methods for financial time series

Author: Vojtěch Herrmann

Department: Department of Probability and Mathematical Statistics

Supervisor: RNDr. Radek Hendrych, Ph.D., Department of Probability and Mathematical Statistics

Abstract: This thesis deals with comparing two approaches to modelling and predicting time series: a traditional one (the ARIMAX model) and a modern one (gradiently boosted decision trees within the framework of the XGBoost library). In the first part of the thesis we introduce the theoretical framework of supervised learning, the ARIMAX model and gradient boosting in the context of decision trees. In the second part we fit the ARIMAX and XGBoost models which both predict a specific time series, the daily volume of the S&P 500 index, which is a crucial task in many branches. After that we compare the results of the two approaches, we describe the advantages of the XGBoost model, which presumably lead to its better results in this specific simulation study and we show the importance of hyperparameter optimization. Afterwards, we compare the practicality of the methods, especially in regards to their computational demands. In the last part of the thesis, a hybrid model theory is derived and algorithms to get the optimal hybrid model are proposed. These algorithms are then used for the mentioned prediction problem. The optimal hybrid model combines ARIMAX and XGBoost models and performs better than each of the individual models on its own.

Keywords: ARIMAX, Hybrid model, Prediction, Stock Market, XGBoost

Obsah

Úvod **3**
Struktura práce . . . 4
Konvence a terminologie . . . 5
Prohlášení autora o out-of-sample množině . . . 8

1 Teoretický rámec **9**
1.1 Supervizované učení . . . 9
1.2 ARIMAX . . . 10
1.3 Teorie vedoucí k XGBoostu . . . 11
1.3.1 Rozhodovací strom typu CART . . . 11
1.3.2 Rozšíření stromu na náhodný les . . . 11
1.3.3 Rozšíření stromu na boostovaný strom . . . 12
1.3.4 Gradientně boostované stromy . . . 13
1.3.5 Algoritmy pro hledání optimálního stromu . . . 15
1.3.6 XGBoost . . . 17
1.3.7 Hyperparametry . . . 18
1.3.8 Důležitost prediktorů . . . 22
1.3.9 Predikční intervaly . . . 22

2 Data **23**
2.1 Zvolený index . . . 23
2.2 Zvolená časová řada . . . 23
2.3 Vstupní data . . . 23
2.3.1 Rozdělení na trénovací, testovací a out-of-sample množinu 26
2.3.2 Explorace dat . . . 26

3 Predikce pomocí modelu ARIMAX **28**
3.1 Určení řádů p, d, q . . . 28
3.1.1 Formální testy stacionarity . . . 28
3.1.2 Volba stavového prostoru pro p, d, q . . . 28
3.1.3 Účast exogenních proměnných . . . 30
3.1.4 Grid search . . . 32
3.2 Výběr exogenních proměnných . . . 35
3.2.1 Základní modely . . . 35
3.2.2 Skupiny exogenních proměnných . . . 36
3.2.3 Finální skupina proměnných . . . 38
3.2.4 Finální model . . . 38
3.3 Verifikace předpokladů modelu . . . 39
3.4 Zhodnocení zvoleného přístupu . . . 40

4 Predikce pomocí modelu XGBoost **41**
4.1 Trénování pomocí cross-validace . . . 41
4.2 Výchozí hyperparametry . . . 42
4.3 Skupiny proměnných . . . 42
4.3.1 Výsledky . . . 43

4.4 Neoptimalizovaný model . . . 43
4.5 Optimalizace hyperparametrů . . . 44
4.5.1 Finální model s optimalizovanými hyperparametry . . . 48
4.6 Výkonnost modelu po letech . . . 48
4.7 Důležitost prediktorů . . . 50
4.8 Zhodnocení zvoleného přístupu . . . 51

5 Porovnání přístupů **53**
5.1 Predikce . . . 53
5.2 Porovnání přístupů . . . 53
5.3 Celkové zhodnocení . . . 56

6 Hybridní model **57**
6.1 Teorie optimálního hybridního modelu . . . 57
6.2 Optimální hybridní model dle MSE . . . 58
6.2.1 Přesný algoritmus . . . 65
6.2.2 Aproximativní algoritmus . . . 68
6.2.3 Teoretické porovnání algoritmů . . . 72
6.2.4 Simulace . . . 73
6.2.5 Příklad použití: Aplikace na predikovanou časovou řadu . . . 75
6.3 Zhodnocení zvoleného přístupu . . . 77
6.3.1 Alternativní přístup . . . 78

Závěr **80**

Literatura **83**

Seznam obrázků **86**

Seznam tabulek **89**

A Přílohy **90**
A.1 Seznam chyb při trénování ARIMAX modelu . . . 90
A.2 Analýza skupin exogenních proměnných . . . 91
A.3 Průběh trénování XGBoost modelu . . . 97

Úvod

Tato diplomová práce se bude věnovat vybraným možnostem predikce finančních časových řad.

Pro oblast aplikace jsme zvolili akciový trh, a to z několika důvodů, které zde nyní popíšeme.

V posledních letech se výrazně zjednodušil přístup široké veřejnosti k možnostem obchodování s finančními instrumenty, kromě tradičních a zavedených brokerů byla založena řada firem umožňující investovat lidem velmi jednoduše online. Platformou pro investování se časem stala řada původně jinak zaměřených tzv. „fintechů“, firem startupové povahy přinášejících moderní technologie a zjednodušení do světa financí.

V době koronavirové pandemie v roce 2020 došlo nejprve k prudkému propadu ceny většiny akciových trhů a následně k ještě prudšímu růstu, který trval do doby uzavření obsahu této práce v září 2020.

Vidina možnosti rychlého zbohatnutí spolu s dostupností prostředků k jeho realizaci tak způsobila extrémní nárůst popularity obchodování na burze.

Typickou a přirozenou úlohou jsou pokusy o predikci cen jednotlivých titulů. Tato nesmírně náročná úloha často končí zjištěním, že nejlepší predikcí přírůstku ceny je 0, případně dlouhodobý průměr tohoto přírůstku (tj. vývoj ceny se modeluje pomocí dlouhodobého lineárního trendu). Závěrem takových studií většinou bývá, že investorská zkušenost doplněná o znalost prostředí a schopnosti predikovat politický a ekonomický vývoj vede často k přesnějším predikcím ceny než matematické modely. Část matematické finanční teorie je na předpokladu nulové střední hodnoty denního přírůstku na akciových trzích založena – například různé odhady volatilit se často počítají pouze jako kvadráty přírůstku, jak je uvedeno v [10]. Celá teorie ARCH/GARCH modelů, pomocí níž se často – např. v [2] – modeluje volatilita akcií, tuto vlastnost také předpokládá.

Další typickou úlohou jsou predikce volatility, a to nejen jednotlivých titulů, ale hlavně multidimenzionální predikce – nalezení negativní korelace mezi dvěma a více tituly může totiž vést k optimalizaci rizika v rámci investorova portfolia, což je velmi důležité primárně pro ty subjekty, které jsou na vývoji cen na trhu finančně závislé. Sem patří například profesionální obchodníci, ať už fyzické osoby, nebo specializované firmy a další.

My se zaměříme na jinou, nicméně pro řadu subjektů důležitou úlohu – predikci denního zobchodovaného objemu. Přesná predikce této řady je důležitá hlavně pro ty subjekty, které se chystají prodávat velké množství jedné akcie nebo podobných akcií. Existují expertní pravidla, která takovým subjektům doporučují, jak toto prodávání akcie provádět postupně, nikoliv najednou, aby velkou nabídkou na trhu nesnížily cenu, za kterou se obchod zrealizuje. Drtivá většina těchto pravidel jsou omezení ve smyslu stanovení maximálního denního podílu na trhu. Tedy množství jednotlivých titulů nemá překročit určitou hranici nastavenou relativně vůči celkovému zobchodovanému objemu dané akcie nebo na daném trhu.

Na hodnoty této časové řady může mít vliv řada faktorů, které v naší analýze zahrneme – chování investorů ovlivňuje sezónnost (např. v průběhu roku nebo týdne), dále různé mimořádné situace na trhu. S řadou z nich se nedá dopředu počítat, nicméně jsou takové, se kterými ano – tím myslíme zejména významné pravidelně oficiálně vydávané ekonomické zprávy.

V našem praktickém případě budeme modelovat denní hodnoty této časové řady. Důvody jsou ryze praktické – denní data ze všech hlavních světových burz jsou volně k dispozici, data s nižší granularitou se shánějí obtížně, a pokud má model fungovat v reálném čase, není tak jednoduché získat volně dostupný a spolehlivý zdroj dat pro model.

Omezíme se na jednokrokovou predikci, která ve většině případů postačuje. Pro praktické vyhodnocení budeme používat posuvné okno, vždy ale budeme predikovat pouze jeden den dopředu.

Cílem práce je tuto úlohu modelovat pomocí různých přístupů, z nichž některé bychom označili jako „tradiční“, některé jako „moderní“. Budou nás zajímat všechny aspekty daných přístupů – od výkonnosti nejlepších modelů až po nutné úsilí a časovou a technickou náročnost nalezení optimálního modelu.

Jako zástupce „tradičního“ přístupu volíme model ARIMAX, vzhledem k tomu, že je velmi běžně používaný a v porovnání s lineární regresí nebo modely AR, ARMA a ARIMA komplexní.

Moderních metod strojového učení je celá řada, např. neuronové sítě, náhodné lesy, metoda podpůrných vektorů (*Support vector machine*), gradientně boostované stromy a další. Vybrali jsme posledně jmenovaný přístup – gradientně boostované stromy – konkrétně jednu z nejmodernějších softwareových implementací, knihovnu XGBoost. Jedná se totiž o poměrně populární a v současnosti hojně používanou metodu, která je velmi úspěšná v řadě soutěží ([30]).

XGBoost se úspěšně využívá pro predikci nejen finančních časových řad, často jako hybridní model v kombinaci s tradičními přístupy (např. ARIMAX v [28]) nebo jinými moderními (např. s rekurentními neuronovými sítěmi v [26]).

V obou případech se pokusíme využít značného výpočetního výkonu, který nám umožní vyzkoušet natrénovat desítky tisíc modelů a tím postupně kalibrovat hyperparametry a optimálně pro ně vybírat proměnné. Protože prostor všech možných modelů je obrovský, nemůžeme natrénovat všechny modely, které bychom chtěli, a využíváme tedy postupného ořezávání tohoto prostoru na základě nejprve hrubých, později jemnějších kritérií. Tento postup nemusí vést vždy ke globálnímu optimu, což je riziko, jehož jsme si vědomi.

Struktura práce

V první části práce (kapitola 1) je představen teoretický rámec supervizovaného učení a modelu ARIMAX. Je uvedena potřebná teorie gradientního boostingu v kontextu rozhodovacích stromů a konkrétní aspekty implementace XGBoost. Velký důraz je kladen na popsání funkce hlavních hyperparametrů v XGBoostu.

V druhé části práce (kapitoly 2 až 4) jsou identifikovány modely ARIMAX a XGBoost, které oba predikují konkrétní časovou řadu – denní zobchodovaný objem indexu S&P 500, což je pro řadu odvětví velmi důležitá úloha. U obou modelů se pokoušíme pomocí grid-searchových metod optimalizovat většinu hyperparametrů. Výsledky obou přístupů porovnáme na out-of-sample množině. Na závěr jsou metody porovnány i po praktické stránce, speciálně co do výpočetní náročnosti.

Ve třetí části práce (kapitola 6) modely ARIMAX a XGBoost kombinujeme pomocí tzv. hybridního modelu. Ten kombinuje predikce vstupních modelů – kombinace predikcí může být obecně lepší než nejlepší z jednotlivých predikcí. K této problematice odvodíme teoretický rámec a navrhneme efektivní způsob řešení. Nakonec aplikujeme na naší řadu a ověříme výkonnost na out-of-sample množině.

Konvence a terminologie

Anglické termíny

Vzhledem k tomu, že statistické metody, jimiž se tato diplomová práce zabývá, jsou vesměs moderní, je přirozené, že řada termínů, které s nimi souvisí, ještě nemá běžně používaný český ekvivalent. V takovém případě definujeme termín anglicky (značíme *kurzívou*) a dává-li to smysl, skloňujeme ho jako anglicismus.

Rozdělování datového vzorku

Vysoce nekonzistentní používání pojmů trénovací, testovací, validační a cross-validační množina nás nutí k explicitnímu popisu námi používané terminologické konvence.

Data dělíme na tzv. *in-sample* a *out-of-sample*. In-sample množinu používáme pro trénování optimálního modelu a dělíme ji na trénovací a testovací podmnožinu. Trénovací množinu využíváme k odhadu optimálních parametrů modelu, testovací množinu využíváme pro kalibraci hyperparametrů modelu. Obě tyto množiny můžeme obecně následně použít pro trénování finálního modelu.

Speciální přístup k trénování je využívání cross-validační množiny. V takovém případě již in-sample data nedělíme na fixní trénovací a testovací množinu, ale rolí trénovacích a testovacích množin postupně nabývají disjunktní podmnožiny, na něž dopředu rozdělíme celá in-sample data (tzv. *foldy*). Podrobný popis této metodologie se nachází v sekci 4.1.

Z hlediska terminologie bývá běžnější hovořit o trénovací, testovací a validační množině – nicméně vzhledem k nejednoznačnosti dvojice pojmů testovací a validační množina (v části odborné i neodborné literatury se jejich významy obrací) využíváme výše zmíněnou terminologii.

Parametry a hyperparametry

V předchozím odstavci zmiňujeme parametry a hyperparametry. Hyperparametry modelu jsou takové parametry, které se typicky volí před trénováním modelu a jejich optimalizace není na rozdíl od klasických parametrů součást trénovacího procesu. Příkladem hyperparametru je řád p autoregresního modelu AR(p), příkladem parametrů v AR(1) jsou c, φ a σ^2 v zápise $X_t = c+\varphi X_{t-1}+\varepsilon_t$, $t = 1, 2, \ldots$, kde ε_t je bílý šum s rozptylem σ^2.

Používané metriky

Pro měření míry přesnosti modelů budeme používat různé metriky s následujícími definicemi.

Začneme nejprve metrikami, které jsou obecné a nezávislé na použitém modelu, sem patří:

- MSE (***M**ean **S**quare **E**rror* – střední čtvercová chyba),
- RMSE (***R**oot **M**ean **S**quare **E**rror*),
- MAE (***M**ean **A**bsolute **E**rror* – střední absolutní chyba),
- MAPE (***M**ean **A**bsolute **P**ercentage **E**rror* – střední absolutní procentuální chyba) a
- R^2.

Pro vektor skutečných hodnot $\mathbf{x}$ a vektor predikovaných hodnot $\widehat{\mathbf{x}}$ (oba s délkou n) definujeme:

$$\mathrm{MSE}(\widehat{\mathbf{x}}) := \frac{1}{n}\sum_{i=1}^{n}(x_i - \widehat{x}_i)^2,$$

$$\mathrm{RMSE}(\widehat{\mathbf{x}}) := \sqrt{\mathrm{MSE}(\widehat{\mathbf{x}})},$$

$$\mathrm{MAE}(\widehat{\mathbf{x}}) := \frac{1}{n}\sum_{i=1}^{n}\left|x_i - \widehat{x}_i\right|,$$

$$\mathrm{MAPE}(\widehat{\mathbf{x}}) := \frac{1}{n}\sum_{i=1}^{n}\left|\frac{x_i - \widehat{x}_i}{x_i}\right|,$$

$$R^2(\widehat{\mathbf{x}}) := 1 - \frac{\mathrm{RSS}(\widehat{\mathbf{x}})}{\mathrm{TSS}} := 1 - \frac{\sum_{i=1}^{n}(x_i - \widehat{x}_i)^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2},$$

kde $\overline{x}$ je aritmetický průměr hodnot $\mathbf{x}$, tj. $\frac{1}{n}\sum_{i=1}^{n} x_i$.

Tyto metriky se běžně používají a jejich interpretace je přímočará, popsána je např. v [10].

Dále budeme používat U, tzv. Theilův koeficient (index) nesouladu, v literatuře označovaný jako *Theil's Inequality Coefficient*. Tento pojem je však v literatuře

nekonzistentně používaný, mimo jiné z důvodu různých interpretací, z nichž některé jsou vhodnější pro časové řady, jiné pro neuspořadatelná pozorování. Vyjdeme z definice používané v [19], protože porovnává predikci modelu s naivní predikcí definovanou jako zpožděná odezva, což je v kontextu časových řad obzvlášť přímočarý přístup.

$$U(\widehat{\mathbf{x}}) := \frac{\sqrt{\sum_{i=1}^{n-1}\left(\frac{\widehat{x}_{i+1}-x_{i+1}}{x_i}\right)^2}}{\sqrt{\sum_{i=1}^{n-1}\left(\frac{x_{i+1}-x_i}{x_i}\right)^2}}.$$

Interpretace Theilova U je následující:

- Nabývá hodnot $[0, \infty)$.
- Perfektní předpověď ($\forall i : \widehat{x}_i = x_i$) znamená $U = 0$.
- Čím větší U, tím vzdálenější je predikce od skutečných hodnot.
- Hodnota $U = 1$ odpovídá kvalitě naivní predikce pomocí zpožděné odezvy, jak je zjevné z definice.
- Hodnoty $U < 1$ odpovídají modelům, které mají lepší predikční schopnost než zmíněná naivní predikce, hodnoty $U > 1$ naopak těm, které mají horší predikční schopnost než naivní predikce.

Jiná definice používaná v teoretické ekonomii je popsána např. v [24]. Nicméně tato alternativní definice je obecně kritizovanější a specifické články věnující se problematice Theilova koeficientu dávají v závěrech přednost námi používanému, např. [1], [4] a [15].

Pro modely založené na maximalizaci věrohodnosti také definujeme AIC (Akaikeho informační kritérium).

$$\mathrm{AIC}(\widehat{\mathbf{x}}) := -2l(\widehat{\mathbf{x}}) + 2K, \tag{1}$$

kde l je logaritmická věrohodnostní funkce a K je počet parametrů modelu. Za předpokladu normálně rozdělených chybových členů můžeme vzorec upravit na tvar

$$\mathrm{AIC}(\widehat{\mathbf{x}}) := n\log(\widehat{\sigma}^2) + 2K, \tag{2}$$

kde $\widehat{\sigma}^2 = \mathrm{RSS}(\widehat{\mathbf{x}})/n$ a n je počet pozorování. V literatuře se občas tento výraz ještě dělí počtem pozorování (např. [10]), nicméně my používáme výše uvedenou definici.

Prohlášení autora o out-of-sample množině

Out-of-sample množina byla určena pouze pro porovnání jednotlivých přístupů (ARIMAX a XGBoost). Autor práce prohlašuje, že tato množina nebyla žádným způsobem použita ani vzata v úvahu při trénování jednotlivých modelů a výsledek práce – porovnání jednotlivých přístupů – je v tomto ohledu zcela nezkreslený.

V závěru práce je zmíněno, zda by výsledky práce byly při jiném zvoleném postupu lepší.

V separátní kapitole 6 pak používáme out-of-sample množinu ještě pro jeden účel – určení optimálního hybridního modelu. Pro tuto analýzu out-of-sample množinu rozdělujeme na trénovací a validační část. I pro validační část out-of-sample množiny platí stejné prohlášení.

1. Teoretický rámec

Abychom ARIMAX a XGBoost zasadili do kontextu obecnější teorie, začneme definicí supervizovaného učení (též učení s učitelem).

1.1 Supervizované učení

Supervizované učení je problém nalezení vztahu mezi výstupní veličinou nebo vektorem, též odezvou, a vstupní veličinou nebo typičtěji vektorem. Formálně zapsáno:

Definice 1.1. ***Supervizovaným učením*** *rozumíme problém nalezení vztahu mezi vektorovými veličinami* $\mathbf{y}$ *a* $\mathbf{x}$ *na základě dat* $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i), \mathbf{x}_i \in \mathbb{R}^m, \mathbf{y}_i \in \mathbb{R}^k, i \in \{1, 2, \ldots, n\}\}$. $n \in \mathbb{N}$ *je* ***počet pozorování*** *v datech,* $\mathbf{y}$ *se nazývá* ***vysvětlovaná proměnná*** *nebo* ***odezva****, jednotlivé složky vektoru* $\mathbf{x}$ *se nazývají* ***regresory****,* ***vysvětlující proměnné*** *nebo* ***prediktory****. V případě jednorozměrného* $\mathbf{y}$ *mluvíme o* ***jednodimenzionálním problému****, v opačném případě o* ***multidimenzionálním*** *nebo též* ***mnohorozměrném****.*

Úkolem supervizovaného učení je nalézt model, který tento vztah popíše:

Definice 1.2. ***Modelem*** $\phi_{\xi,\theta}$ *rozumíme funkci, která přiřazuje každému vstupu* $\mathbf{x} \in \mathbb{R}^m$ *hodnotu* $\widehat{\mathbf{y}} \in \mathbb{R}^k$*, této hodnotě budeme říkat* ***vyrovnaná hodnota*** *(fitted value), pokud se jedná o* $\mathbf{x}$ *z* $\mathcal{D}$*, pokud ne, pak se používá termín* ***predikovaná hodnota****. Model závisí na* ***hyperparametrech*** $\xi \in \mathbb{R}^p$ *a* ***parametrech*** $\theta \in \mathbb{R}^q$*.*

Pro trénování modelu se typicky používá optimalizace vhodně zvolené funkce, tzv. *objective function*, která se skládá ze dvou částí:

- ztrátové funkce (*loss function*), která měří kvalitu modelu ve smyslu vzdálenosti $\widehat{\mathbf{y}}$ od skutečné hodnoty $\mathbf{y}$ a
- regularizačního členu (*regularization term*), který penalizuje příliš komplexní model.

Formálně:

Definice 1.3. ***Objective funkcí*** $\mathcal{L}(\phi_{\xi,\theta}, \mathcal{D})$ *rozumíme (typicky) reálnou funkci tvaru*

$$loss(\widehat{\mathbf{y}}, \mathbf{y}) + \Omega(\xi, \theta),$$

kde první člen je ***ztrátová funkce*** *a druhý* ***regularizační člen****.*

Typicky má ztrátová funkce vlastnost „čím přesnější fit, tím menší hodnota“ a regularizační člen vlastnost „čím komplexnější model, tím větší hodnota“. V takovém případě definujeme proces trénování modelu:

Definice 1.4. ***Trénováním modelu*** *rozumíme libovolný způsob hledání optimální hodnoty* θ *při zafixovaném* ϕ *a* ξ*, při které se na datech* $\mathcal{D}$ *minimalizuje hodnota* $\mathcal{L}(\phi_{\xi,\theta}, \mathcal{D})$*.*

V případě opačné interpretace hodnot ztrátové funkce, resp. regularizačního členu, se v Definici 1.4 místo minimalizace objeví maximalizace.

V této definici vidíme praktický rozdíl mezi parametry a hyperparametry. Zatímco hyperparametry ξ je třeba zafixovat před trénováním modelu, parametry θ jsou odhadovány, resp. kalibrovány, při trénování modelu – lépe řečeno odhad, resp. kalibrace, parametrů je synonymem trénování modelu.

Jako příklad lze uvést běžné regresní modely: lineární regrese (OLS) využívá jako ztrátovou funkci sumu kvadrátů rozdílů vyrovnaných a skutečných hodnot a nulový regularizační člen, objective je tedy $\sum_{i=1}^{n}(\widehat{x}_i - x_i)^2$, Lasso regrese využívá regularizační člen ve tvaru sumy absolutních hodnot všech parametrů v modelu, objective je tedy $\sum_{i=1}^{n}(\widehat{x}_i - x_i)^2 + \alpha \sum_{j=1}^{m} |\beta_j|$, kde α je hyperparametr a jednotlivé β_j jsou regresní koeficienty. V obou případech je $\widehat{x}_i$ i-tá vyrovnaná hodnota a x_i i-tá skutečná hodnota.

1.2 ARIMAX

Jedním z modelů spadajících do supervizovaného učení je model ARIMAX. Jedná se o rozšíření standardního modelu ARIMA (viz například [10]) o regresní (exogenní) proměnné. Tyto proměnné mohou modelovat například sezónnost i nad rámec možností modelu SARIMA (např. pomocí indikátorů specifických období).

Pro jednorozměrnou časovou řadu $\{y_t,\, t \in \{1, 2, \ldots, n\}\}$ a odpovídající data $\mathcal{D} = \{(\mathbf{x}_t, y_t),\, \mathbf{x}_t \in \mathbb{R}^m,\, t \in \{1, 2, \ldots, n\}\}$ definujeme model ARIMAX následujícím způsobem:

Definice 1.5. ARIMAX(p, d, q), *$p, d, q \in \mathbb{N}_0$, je model tvaru*

$$\phi(B)\Delta^d y_t = \alpha + \mathbf{x}_t^T \beta + \theta(B)\varepsilon_t, \tag{1.1}$$

kde B je operátor časového posunutí, tj. $B(y_t) = y_{t-1}$;

$$\phi(B) = 1 + \sum_{i=1}^{p} \phi_i B^i,$$

kde $\phi_i \in \mathbb{R}$ jsou parametry modelu, speciálně $\phi_p \neq 0$; dále Δ^d je d-tá mocnina diferenčního operátoru Δ, který je definovaný předpisem $\Delta y_t = y_t - y_{t-1}$; $\alpha \in \mathbb{R}$; $\beta \in \mathbb{R}^m$ je vektor regresních parametrů;

$$\theta(B) = 1 + \sum_{i=1}^{q} \theta_i B^i,$$

kde $\theta_i \in \mathbb{R}$ jsou parametry modelu, speciálně $\theta_q \neq 0$ a $\{\varepsilon_t,\, t \in \{1, 2, \ldots, n\}\}$ je bílý šum.

Parametry $\alpha, \phi, \theta, \beta$ a σ^2 (rozptyl bílého šumu) se odhadují sdruženě pomocí metody maximální věrohodnosti, parametry p, d, q a výběr exogenních proměnných promítající se do jejich počtu m jsou hyperparametry a při výpočtech souvisejících s hledáních prvně zmíněných parametrů figurují jako konstanty.

Podrobněji se modely typu ARIMA (bez regresní složky) zabývá článek [8] a jejich rozšířením na modely typu ARIMAX poté [3].

V praktické části budeme model ARIMAX reprezentovat výčtem parametrů p, d a q (v tomto pořadí) a dále výčtem proměnných použitých v regresním vektoru $\mathbf{x}_t$. Intercept α v modelu uvažujeme vždy, i když ho explicitně nezmiňujeme.

1.3 Teorie vedoucí k XGBoostu

1.3.1 Rozhodovací strom typu CART

Následující sekce vychází zejména z knihy [5].

Klasifikační a rozhodovací strom CART (***C**lassification **A**nd **R**egression **T**ree*) je speciální rekurzivní struktura reprezentující určitou funkci s konečným oborem hodnot. Tyto hodnoty můžou být numerické (regresní strom) nebo kategoriální (klasifikační strom). Protože náš problém je regresní, budeme se dále zabývat pouze regresním stromem:

Definice 1.6. ***Regresním stromem*** *rozumíme binární strom s T listy, každý s reprezentující hodnotou w_t pro $t = 1, 2, \ldots, T$, který každému vstupnímu vektoru $\mathbf{x} \in \mathbb{R}^m$ přiřadí právě jeden list t, resp. jeho reprezentující hodnotu w_t. Formálně jde tedy o funkci $f(\mathbf{x}) = w_{q(\mathbf{x})}$, kde $q : \mathbb{R}^m \to \{1, 2, \ldots, T\}$.*

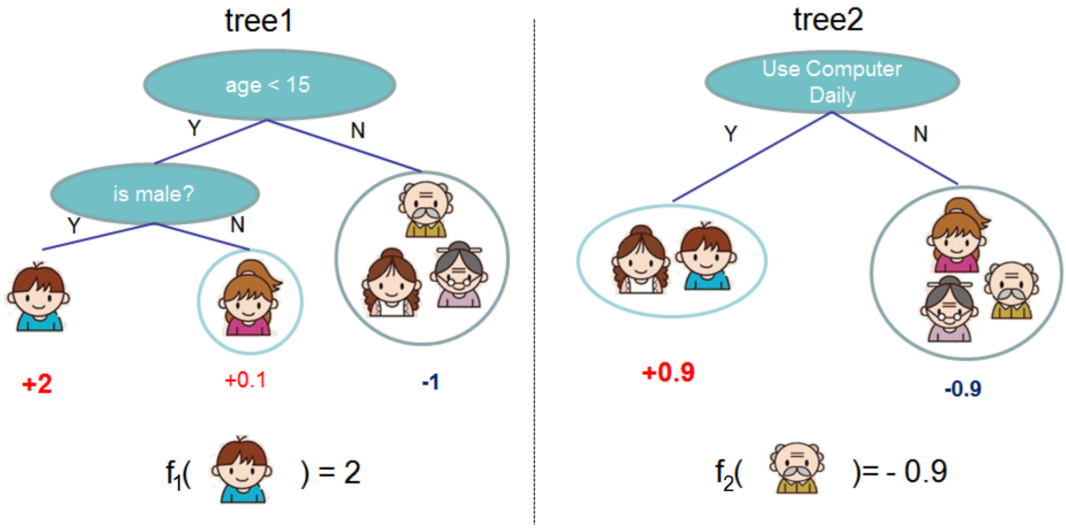
Tento definiční pohled na rozhodovací strom postrádá rekurzivní perspektivu, kterou lze lépe nahlédnout na Obrázku 1.1. Strom 2 (*tree2*) reprezentuje nejjednodušší stromovou strukturu – jeden bod dělení, tzv. *split*. Strom 1 (*tree1*) reprezentuje rekurzivní stromovou strukturu. U obou stromů vidíme důležitou vlastnost, že každé pozorování (v tomto konkrétním případě člen rodiny) náleží v každém stromu právě jednomu listu. Tento list může být reprezentovaný sdruženě sadou maximálně d současně nastávajících podmínek, kde d je maximální hloubka stromu, v našem případě 1, resp. 2.

Jeden samotný rozhodovací strom je v praxi často nedostatečně kvalitní model, obzvlášť s množstvím prediktorů větším než malé jednotky. To proto, že na dostatečnou vysvětlovací schopnost stromu je třeba použít větší množství prediktorů a tím vzniká nutnost dostatečně hlubokého stromu. Nicméně hluboké stromy bývají často přeučené na trénovací množinu – vyberou si menší počet prediktorů a v nich se přeučí. Proto se typicky přistupuje k trénování více stromů a jejich kombinování. Zde se rozdělují dvě významné třídy modelů – náhodný les (*Random Forrest*) a boostovaný strom (*Boosted Tree*). Základní rozdíl mezi náhodnými lesy a boostovanými stromy je v principu, jak se tato sada stromů vytváří.

1.3.2 Rozšíření stromu na náhodný les

Náhodný les je model vyvinutý v 90. letech 20. století v článcích [6] a [17].

Náhodný les bere stále dokola původní data, vybírá náhodnou (pokaždé novou) podmnožinu prediktorů, bootstrapuje pozorování (provádí náhodný výběr s opakováním) a na vzniklých datech trénuje regresní (nebo klasifikační) strom.



Obrázek 1.1: Příklad dvou regresních stromů a jejich reprezentace pomocí funkce f. *Zdroj:* [9], *Figure 1, upraveno.*

Protože všechny jednotlivé stromy predikují tu samou veličinu, výsledná predikce je určitá prostřední (nikoliv střední) nebo běžná hodnota predikovaná jednotlivými stromy. Pro klasifikační stromy se často bere modus (nejčastěji predikovaná hodnota), pro regresní pak medián nebo průměr.

Vzhledem k vzájemné nezávislosti procesů trénování jednotlivých stromů je proces trénování náhodného lesa triviálně paralelizovatelný – máme-li pro F požadovaných stromů k dispozici k vláken procesoru, na každém vlákně natrénujeme zhruba F/k stromů a vzhledem k tomu, že agregační logika je typicky velmi jednoduchá, lze výpočetní čas redukovat na libovolný násobek času potřebného k natrénování jednoho stromu.

1.3.3 Rozšíření stromu na boostovaný strom

Boostovaný strom nemá formální definici – jedná se spíše o koncept vytváření finální predikce – první strom vyprodukuje vyrovnané hodnoty a rezidua. Druhý strom se snaží „opravit“ chybu prvního stromu (tj. rezidua), třetí strom se snaží opravit chybu předchozích dvou atd. Vzhledem k této formulaci můžeme zapsat finální model jako

$$\widehat{y}_i = \sum_{i=1}^{K} f_k(\mathbf{x}_i),\ f_k \in \mathcal{F}, \tag{1.2}$$

kde $\mathcal{F}$ je prostor všech regresních stromů dle Definice 1.6 a K je počet stromů.

I fakt, že výraz (1.2) nelze v Eukleidovském prostoru optimalizovat s využitím tradičních optimalizačních metod (viz [9]), byl motivací k již popsanému itera-

tivnímu trénování modelu resultujícímu v aditivní predikci:

$$\begin{aligned} \widehat{y}_i^{(0)} &= 0, \\ \widehat{y}_i^{(1)} &= \widehat{y}_i^{(0)} + f_1(\mathbf{x}_i), \\ &\dots \\ \widehat{y}_i^{(t)} &= \widehat{y}_i^{(t-1)} + f_t(\mathbf{x}_i), \end{aligned} \tag{1.3}$$

pro iterativně rostoucí t. Hodnota t může být seshora omezená předem definovaným hyperparametrem K, ale vzhledem k tomu, že opravy přirozeně postupně zmenšují chybu a zvyšují komplexitu modelu (tj. nejprve zkvalitňují a poté přetrénovávají model), lze terminální t volit až dle průběhu trénování na základě různých kritérií – tento přístup bude později popsán detailněji.

Jako počáteční predikci $\widehat{y}_i^{(0)}$ lze kromě nuly brát například aritmetický průměr nebo medián všech hodnot v datasetu. Tato hodnota se pak přičítá k hodnotě ve vzorci (1.2). Současná implementace většiny softwarových knihoven nicméně začíná s nulou.

Jak je zjevné z předchozích rovnic, boostovaný strom dává smysl pouze pro regresní (tedy nikoliv klasifikační) stromy.

Ve skutečné implementaci se nepřičítá $f_t(\mathbf{x}_i)$, ale $\eta f_t(\mathbf{x}_i)$ pro předem zvolený hyperparametr $0 < \eta \leq 1$. Tento parametr se v praxi často označuje jako tzv. *learning rate* a je součást téměř každého algoritmu, který je založen na gradientních metodách. Ty fungují tak, že na základě parciálních derivací v určitém bodě (kde se zrovna algoritmus nachází) určují, v kterém směru dochází k největšímu poklesu objective funkce (při minimalizačních úlohách). Tam se zkoumaný bod posouvá. Aby ale tento bod neosciloval kolem skutečného minima, velikost posunu se poměrně redukuje právě zmíněnou konstantou η. V praxi se téměř vždy volí výrazně menší než 1, což odpovídajícím poměrem zvyšuje počet kroků algoritmu, ale zabraňuje popsanému problému. Nicméně pro určení optimálního f_t se η nepoužívá – jde o aplikaci tohoto „bezpečnostního postupu“ na nejlepší odhad algoritmu.

Otázkou nicméně zůstává, jak přesně volit jednotlivé funkce (resp. nové stromy) f_t. Odpověď na ni představíme v následující sekci.

1.3.4 Gradientně boostované stromy

Originální myšlenku gradientně boostovaných stromů představil J. H. Friedman na konci 20. století v článcích [13] a [14].

Základní myšlenka vychází z výše zmíněně logiky zlepšování již spočítané predikce. Zasadíme-li tuto myšlenku do kontextu teorie prezentované v sekci 1.3, můžeme odvodit, jak nový strom natrénovat.

Předpokládejme, že jsme v iteraci t a chceme najít v jistém smyslu optimální strom f_t. Optimalitu vyjádříme následujícím způsobem:

$$\begin{aligned} f_t &= \arg\min_{f_t \in \mathcal{F}} \ \mathcal{L}_t = \arg\min_{f_t \in \mathcal{F}} \ loss(\widehat{\mathbf{y}}^{(t)}, \mathbf{y}) + \Omega(\xi, \theta) = \\ &= \arg\min_{f_t \in \mathcal{F}} \ loss(\widehat{\mathbf{y}}^{(t-1)} + f_t(\mathbf{x}), \mathbf{y}) + \Omega(\xi, \theta), \end{aligned} \tag{1.4}$$

kde *loss* je nějaká ztrátová funkce, Ω je regularizační člen spočítaný z hyperparametrů ξ a parametrů θ stromu f_t.

Za předpokladu, že se dá ztrátová funkce *loss* zapsat jako suma ztrátových funkcí $loss^*$ na jednotlivých pozorováních a vyrovnaných hodnotách, můžeme dále upravit ve vzorci (1.4) optimalizovanou objective funkci:

$$\mathcal{L}_t = \sum_{i=1}^{n} loss^*(\widehat{y}_i^{(t-1)} + f_t(\mathbf{x}_i), y_i) + \Omega(\xi, \theta). \tag{1.5}$$

Za předpokladu, že je ztrátová funkce $loss^*$ dvakrát diferencovatelná, můžeme ji aproximovat Taylorovým polynomem druhého řádu následujícím způsobem:

$$\mathcal{L}_t \simeq \sum_{i=1}^{n} \left[loss^*(\widehat{y}_i^{(t-1)}, y_i) + g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i) \right] + \Omega(\xi, \theta), \tag{1.6}$$

kde $g_i = \partial_{\widehat{y}_i^{(t-1)}} loss^*(\widehat{y}_i^{(t-1)}, y_i)$ a $h_i = \partial^2_{\widehat{y}_i^{(t-1)}} loss^*(\widehat{y}_i^{(t-1)}, y_i)$ jsou první a druhá parciální derivace ztrátové funkce $loss^*$.

Vzhledem k tomu, že se nám podařilo izolovat hodnotu ztrátové funkce $loss^*$ předchozí predikce, která je v kontextu predikce v iteraci t konstantní, můžeme tento člen vypustit a definovat

$$\widetilde{\mathcal{L}}_t := \sum_{i=1}^{n} \left[g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i) \right] + \Omega(\xi, \theta). \tag{1.7}$$

Pro další úpravy potřebujeme zafixovat regularizační člen $\Omega(\xi, \theta)$. Možností, jak zvolit funkci Ω, je celá řada. V současné implementaci XGBoostu ([29]) je následující:

$$\Omega(\xi, \theta) := \gamma T + \frac{1}{2} \lambda \sum_{i=1}^{T} w_i^2, \tag{1.8}$$

kde γ a λ jsou dva z hyperparametrů v XGBoostu, které budou popsány později.

Dosazením vzorce (1.8) do rovnice (1.7) dostaneme

$$\widetilde{\mathcal{L}}_t = \sum_{i=1}^{n} \left[g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i) \right] + \gamma T + \frac{1}{2} \lambda \sum_{i=1}^{T} w_i^2. \tag{1.9}$$

Nyní zafixujme stromovou strukturu q.

Použijeme Definici 1.6. Přerovnáme sumu přes pozorování na sumu přes hodnoty $q(x)$, využíváme faktu, že $q(x)$ (tj. list) už určuje hodnotu $f(x)$.

$$\widetilde{\mathcal{L}}_t = \sum_{j=1}^{T} \left[(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2 \right] + \gamma T, \tag{1.10}$$

kde I_j je indexová množina pozorování náležících listu j, formálně $\{i, q(x_i) = j\}$.

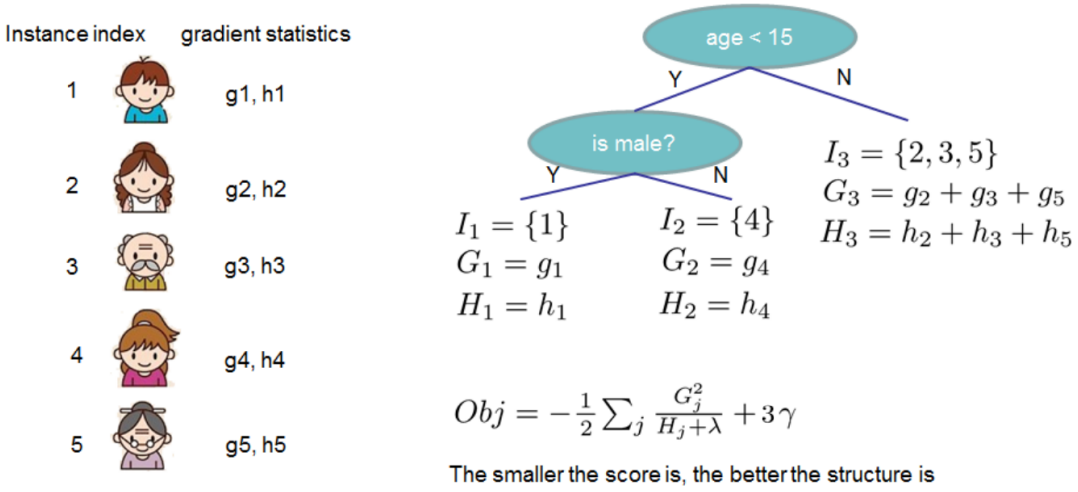
Vzhledem k tomu, že pro fixní stromovou strukturu q jsou jednotlivé optimální w_j^* na sobě navzájem nezávislé, můžeme na minimalizaci výrazu (1.10) nahlížet jako na minimalizaci T výrazů. Derivováním podle každého w_j a položením rovno nule zjistíme, že v každém prvku sumy je optimální hodnota w_j^* rovna

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}. \tag{1.11}$$

Optimální hodnota $\widetilde{\mathcal{L}}_t$ je pak rovna

$$\widetilde{\mathcal{L}}_t^* = -\frac{1}{2} \sum_{j=1}^{T} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T. \tag{1.12}$$

V kontextu Stromu 1 (*tree1*) na Obrázku 1.1 se optimální hodnota spočítá tak, jak je ukázáno na Obrázku 1.2.



Obrázek 1.2: Výpočetní optimální hodnoty objective funkce pro jeden konkrétní strom. *Zdroj:* [9], *Figure 2, upraveno.*

Výše popsaný postup (vzorce (1.11) a (1.12)) dává návod, jak při zafixované stromové struktuře q velmi efektivně nalézt optimální w_j a vyhodnotit objective funkci po přidání stromu v iteraci t, obzvlášť je-li ztrátová funkce jednoduchá a polynomická – například běžně používaná čtvercová odchylka má h_i konstantní a jednotlivé g_i jsou lineární.

Nicméně tento postup nedává žádný návod, jak zvolit q. V článku [18] je dokázáno, že zkonstruovat optimální binární rozhodovací strom je NP-úplný problém. Nyní tedy popíšeme algoritmus hledání optimálního stromu q.

1.3.5 Algoritmy pro hledání optimálního stromu

Všechny dnes používané algoritmy fungují iterativně. To v praxi znamená, že nejprve hledají optimální strom hloubky 1 (s jedním splitem) a za určitých podmínek se rekurzivně pokusí natrénovat na jednotlivých listech (na datech, která danému listu odpovídají) další strom hloubky 1. Hyperparametrů, které mohou

rekurzi zastavit, je celá řada – od přirozených (např. maximální hloubka stromu) až po méně přímočaré (např. parametr γ, jehož funkci záhy vysvětlíme).

Nejprve definujeme tzv. *gain*, hodnotu určující, o kolik lepší model dostaneme rozdělením listu na dva další podlisty. Vyjdeme z tvaru objective funkce ve vzorci (1.12).

Předpokládejme, že indexová množina I_j se splitem rozdělí na dvě disjunktní podmnožiny I_j^L a I_j^R, takové, že $I_j = I_j^L \cup I_j^R$.

$$gain := \frac{1}{2}\left[\frac{(\sum_{i\in I_j^L} g_i)^2}{\sum_{i\in I_j^L} h_i + \lambda} + \frac{(\sum_{i\in I_j^R} g_i)^2}{\sum_{i\in I_j^R} h_i + \lambda} - \frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda}\right] - \gamma \tag{1.13}$$

Gain má obrácené znaménko než objective funkce, takže minimalizace objective funkce se převádí na maximalizaci gainu. V tomto tvaru je dobře vidět interpretace nejen gainu, ale i parametru γ. Hlavní složka gainu je součet dvou nově (po potenciálním splitu) vzniklých hodnot objective funkce na jednotlivých potenciálních listech, od něhož je odečtena původní hodnota objective funkce (před potenciálním splitem). Tato složka tak reprezentuje přírůstek objective funkce při splitu. K němu dojde, pouze pokud zlepšení modelu splitem přesáhne hranici γ. Tento parametr tedy můžeme chápat jako penalizaci zesložitění modelu splitem.

Přesný hladový algoritmus

Základní algoritmus, který důkladněji popíšeme, je tzv. přesný hladový algoritmus (*Exact Greedy Algorithm*), který zkouší všechny možnosti splitu a vybere tu nejlepší.

Algoritmus 1.1: Přesný hladový algoritmus

Vstup: I_j, indexová množina pro aktuální list
Vstup: m, sloupcová dimenze dat $\mathcal{D}$
$gain := -\infty$
$split :=$ `false`
$G := \sum_{i\in I_j} g_i, H := \sum_{i\in I_j} h_i$
for $k \in \{1, 2, \ldots, m\}$ **do**
  $G_L := 0, H_L := 0$
  for $l \in$ *seřazené*$(I_j, \textit{podle}\, x_{lk})$ **do**
    $G_L \mathrel{+}= g_l, H_L \mathrel{+}= h_l$
    $G_R := G - G_L, H_R := H - H_L$
    $new\_gain := \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})\right] - \gamma$
    if $new\_gain > gain$ **then**
      $split :=$ `true`
      $gain := new\_gain$
    end
  end
end
Výstup: $split, k, l$

Podíváme se důkladněji na popsaný Algoritmus 1.1. Tento algoritmus postupně vyzkouší všechny prediktory (první for cyklus) a postupně zkusí zvolit jako split

všechny hodnoty tohoto prediktoru (druhý for cyklus). Nakonec spočítá nový gain, porovná ho s již nalezeným nejlepším modelem, a pokud je lepší, uloží se nejlepší nalezená dvojice k (prediktor) a l (index hodnoty, ve které dojde ke splitu).

Tento přístup se standardně používá, dokud to velikost dat dovolí – pro řádově tisíce pozorování a desítky spojitých prediktorů (tj. deseti- až statisícový prostor všech splitů) je běžně nalezení optimálního stromu otázkou maximálně stovek milisekund. Často se jím také trénuje finální model, i když optimalizace hyperparametrů a různé předběžné modely byly trénovány pomocí aproximativních algoritmů.

Aproximativní algoritmy

Základní aproximativní algoritmus funguje obdobně jako přesný hladový algoritmus, ale v procesu procházení seřazeného vektoru všech hodnot daného prediktoru bere pouze jeho kvantilové hodnoty. Jak jemný kvantil se použije, stejně jako místo, kde k hledání kvantilových hodnot dojde, se dá nastavit pomocí hyperparametrů.

Autoři článku [9] implementovali dva nové aproximativní algoritmy: *Sparsity-aware* a *Weighted Quantile Sketch*.

První zmíněný algoritmus rozšiřuje přesný hladový algoritmus o práci s chybějícími hodnotami. Algoritmus se nejprve pokusí přiřadit chybějící pozorování k levé indexové množině (menší hodnoty než bod splitu), poté k pravé indexové množině a hledá nejlepší variantu.

Druhý zmíněný algoritmus zkvalitňuje výběr kvantilových hodnot tím, že před určením kvantilů jednotlivá pozorování zváží vahami h_l, a tedy lépe reprezentuje důležitost pozorování při trénování modelu.

1.3.6 XGBoost

Samotná knihovna pro e**X**trémní **G**radientní **Boost**ing ([29]) již nepřináší nový teoretický koncept nad rámec gradientně boostovaných stromů, nicméně se jedná o revoluční nástroj z několika důvodů.

- Jedná se o první implementaci, která plně podporuje přístup k chybějícím pozorováním.
- Jedná se o implementaci ve velmi rychlém jazyce C++.
- Vzhledem k tomu, že časově nejnáročnější operace je řazení vektorů hodnot jednotlivých prediktorů, autoři optimalizovali ukládání dat do speciální datové struktury CSC (*Compressed Column Format*), která řazení zrychluje.
- Díky optimálnějšímu cacheování je algoritmus méně náročný na paměť, a tak je aplikovatelný i na problémy, u kterých ostatním implementacím dochází operační paměť.
- Autoři jako první implementovali „odkládání“ dat na hard disk, aby vyřešili problém s paměťovou náročností úlohy. Protože operace s hard diskem jsou

obecně velmi pomalé v porovnání s operacemi s operační pamětí, je součástí implementace proces řídící jednotlivá vlákna, aby paralelně k samotným výpočtům včas dopředu odkládala a komprimovala a zároveň načítala a dekomprimovala data z hard disku.

- Jedná se o implementaci, která plně podporuje paralelizaci.

Jak je zjevné z algoritmu, XGBoost nelze paralelizovat ve smyslu iterací – každá iterace musí mít k dispozici výsledky předchozí iterace. Nejde paralelizovat ani ve smyslu hledání celého optimálního stromu, každý split se musí hledat v kontextu znalosti zbytku stromu (např. protože existují parametry omezující počet listů apod.).

Paralelizace se ale dá použít na dva for cykly v Algoritmu 1.1, neboť všechny výpočty pro všechny dvojice hodnot jsou na sobě navzájem nezávislé. Vzhledem k tomu, že typický počet iterací bývají desítky až stovky, typický počet splitů v jednom stromu bývá $2^3 - 1$ až $2^7 - 1$ a počet variant jednoho splitu je v podstatě počet prediktorů krát průměrný počet unikátních hodnot prediktoru (což je počet, který i pro běžné úlohy může být řádově v milionech nebo miliardách), dá se paralelizovat počtem nejobsáhlejší část problému.

Ve skutečnosti je ale paralelizace výrazně složitější v případě, že se pracuje nejen nad operační pamětí, ale i nad hard diskem (při větším objemu dat). V takovém případě na vláknech probíhají nejen učící algoritmy, ale také práce s pamětí, komprese a dekomprese dat.

1.3.7 Hyperparametry

Následující část textu odpovídá verzi XGBoost 1.1.1 ze 7. 7. 2019 a jejímu ovladači pro Python stejného jména a verze vydanému 7. 6. 2020. Oficiální dokumentace obou knihoven lze nalézt v [29] a [31].

Obecné parametry

Tyto parametry určuje primárně povaha problému a nejsou součást optimalizačních procesů.

`booster` určuje, co je ve výpočtu jeden tzv. *base learner* neboli *weak learner*. Finální model je pak určitou kombinací těchto weak learnerů.

Možné hodnoty jsou:

- *gbtree* – regresní strom, kombinací je pak postupné sčítání predikcí těchto stromů,
- *gblinear* – lineární funkce, kombinace je pak vážená suma těchto funkcí,
- *dart* – speciální případ *gbtree*, který přidává možnost část stromů náhodně zahazovat – tento algoritmus v určitých případech řeší přetrénování, jak ukázali jeho autoři v [27].

Volíme výchozí hodnotu *gbtree*.

`objective` určuje kategorii problému (ve smyslu klasifikace, lineární/logistický regresní problém, ranking apod.) a tím i typ odezvy.

Možných hodnot je celá řada, běžně používanými jsou:

- *reg:squarederror* – odpovídá klasické lineární regresi se ztrátovou funkcí čtverců reziduí,
- *reg:logistic* – odpovídá logistické regresi se ztrátovou funkcí rovnou logaritmické věrohodnosti Bernoulliho rozdělení,
- *binary:logistic* – odpovídá binární klasifikaci (tj. v podstatě logistické regresi s predikovanou hodnotou 1 nebo 0 při pravděpodobnostním treshholdu 0.5) se ztrátovou funkcí rovnou logaritmické věrohodnosti Bernoulliho rozdělení.

Vzhledem k povaze naší odezvy volíme hodnotu *reg:squarederror*.

`eval_metric` určuje metodu, která bude použitá pro validaci modelu a tím pro zastavení trénování modelu. Trénování se dá monitorovat i více kritérii současně, nicméně pro zastavení trénování lze použít pouze jedno.

Možných hodnot je celá řada, výchozí hodnoty jsou závislé na zvolené `objective`, obecně pro regresní problémy je to RMSE, pro klasifikační problémy různé varianty chybového podílu (*error rate*).

Vzhledem k naší `objective` volíme hodnotu *rmse*.

Další obecné parametry jsou v zásadě už pouze technické, například počet využívaných jader procesoru, seed, míra vypisování mezivýsledků apod.

Parametry pro booster

Každý booster má svou sadu parametrů, vzhledem k našemu výběru popíšeme pouze parametry pro regresní strom. Většinu z těchto parametrů budeme optimalizovat, a na začátek proto expertně volíme jejich hodnoty.

`max_depth` je maximální hloubka stromu. Menší hodnota parametru zabraňuje přetrénování, neboť hloubka stromu je úměrná množství listů a splitů (tedy parametrů), tedy i dimenzi prostoru všech možných stromů. Výchozí hodnota 6 je typická pro průměrně velké datasety, obecně doporučované hodnoty jsou v různé literatuře 3-7 až 3-10. Pro počáteční hodnotu při optimalizaci se doporučuje 4-6. My vzhledem k rozsahu dat volíme 4.

`min_child_weight` je minimální hodnota výrazu $\sum_{i \in I_j} h_i$ pro každý list j. Pokud by byla menší, strom se nerozvětví. V lineární regresi (`objective` *reg:squarederror*), se kterou pracujeme, tato váha odpovídá počtu pozorování v každém listu. I tento parametr (resp. jeho vyšší hodnoty) zabraňuje přetrénování (tentokrát hlavně na jednotlivá odlehlá pozorování). Výchozí hodnota je 1, s ní také vzhledem k malé velikosti datasetu začínáme.

gamma je hodnota minimálního poklesu `objective`, která je vyžadována, aby se strom rozvětvil. Také tento parametr (resp. jeho vyšší hodnoty) zabraňuje přetrénování, neboť proti sobě staví minimální požadované zlepšení za cenu zvýšení parametrů přidáním dalšího splitu. I tentokrát vzhledem k malé velikosti datasetu začínáme s výchozí hodnotou 0.

eta, též `learning_rate`, je parametr robustifikace. Určuje, s jakou váhou se má přičíst predikce v každé iteraci k již spočítané predikci. Tedy v iteraci t se predikce spočítá jako $\hat{y}_i^{(t)} := \hat{y}_i^{(t-1)} + \eta f_t(\mathbf{x}_i)$, kde $f_t(\mathbf{x}_i)$ je predikce v iteraci t. Výchozí hodnota je 1, která ale příliš robustní není, a obecně se doporučuje začínat s hodnotami kolem 0.2. Také my začínáme s hodnotou 0.2.

subsample je parametr, který určuje, jak velká část dat (ve smyslu pozorování) se má brát v úvahu při trénování každého jednotlivého stromu. Pozorování jsou vybrána náhodně (parametr `sampling_method` je *uniform*, druhá možnost *gradient_based* je zatím k dispozici pouze při výpočtech na grafické kartě). Nejedná se o bootstraping (neboli *bagging*), kde se vybírají pozorování s opakováním. V případě tohoto procesu se vybírají pozorování bez opakování. Jako výchozí hodnotu vybereme doporučovanou 0.8 (tj. pro každý strom se použije náhodný sample 80 % dat).

colsample_by_* jsou parametry, které v určitý moment náhodně vybírají prediktory, které se použijí pro trénování stromu. Tyto parametry jsou tři:

- `tree` – analogicky jako `subsample` probíhá sampling pro každý jednotlivý strom (tj. iteraci);
- `level` – sampling probíhá iterativně na každé hladině stromu z již samplovaného datasetu pro předchozí hladinu; tento sampling probíhá až po samplování pomocí `colsample_bytree`;
- `node` – sampling probíhá navíc na každém splitu po samplování pomocí `colsample_bytree` a `colsample_bylevel`.

Druhý a třetí zmíněný parametr se doporučují používat na velkých datech, tedy uvažujeme pouze první zmíněný. S ním dle doporučení začínáme na hodnotě 0.8. Další dva necháváme na výchozí hodnotě 1 (žádný sampling) a neoptimalizujeme je.

lambda je koeficient L2-regularizačního členu, nastavili jsme ho na výchozí hodnotu 1.

alpha je koeficient L1-regularizačního členu, nastavili jsme ho na výchozí hodnotu 0.

max_delta_step a scale_pos_weight jsou dva parametry, které pomáhají zlepšovat predikční schopnost logistického modelu s výrazně nevyrovnaným počtem pozitivních a negativních pozorování. Protože ani jedno není náš případ, tyto parametry necháváme na výchozích hodnotách 0, resp. 1, a neoptimalizujeme je. Detaily k těmto parametrům popisuje oficiální dokumentace [29].

`monotone_constraints a interaction_constraints` umožňují vynutit monotonii predikce v určitých prediktorech a zakázat určitou kombinaci prediktorů v jednom stromu. Vzhledem k tomu, že na nastavení takovýchto omezení je většinou potřeba silná expertní znalost problému a navíc nejdou rozumně optimalizovat, necháváme je po celou dobu rovny *null*, čímž se neaplikují.

`num_parallel_tree` je parametr, který určuje, kolik stromů se má natrénovat v každé iteraci. Větší než výchozí hodnota 1 již v běžné terminologii definuje zcela novou rodinu modelů, tzv. boostované náhodné lesy (*boosted random forests*). Vzhledem k malému rozsahu dat necháváme tento parametr rovný 1 a neoptimalizujeme ho, protože boostované náhodné lesy přidávají novou úroveň komplexity nad rámec boostovaných stromů a vyžadují tím větší počet pozorování.

Parametry pro učící proces. Tyto parametry nebudeme optimalizovat.

- `num_boost_round` je počet iterací trénování (tj. finální predikce je suma predikcí tohoto počtu stromů). Nicméně v případě použití předčasného zastavení trénování (viz níže), funguje tento parametr pouze jako horní hranice, které by se správně nemělo dosahovat. Protože předčasné zastavení používáme, volíme ho 1000[1].
- `early_stopping_rounds` je parametr, který specifikuje, zda a jak má být použito předčasné zastavení trénování modelu. Je to číslo, které určuje, po dobu kolika iterací se nemá zlepšit tou dobou nejlepší hodnota `eval_metric` na testovací množině, aby se ukončilo trénování modelu. Za výsledný model je pak považován ten, který vznikne součtem predikcí po strom, s nímž bylo dosaženo nejlepšího výsledku. Praktický příklad lze nalézt v Příloze A.3. Volíme běžně doporučovanou hodnotu 10.
- `maximize` je parametr, který říká, v jakém vztahu je `eval_metric` ke kvalitě modelu (*True* znamená, že lepší model má vyšší hodnotu `eval_metric`). Vzhledem k námi zvolené metrice (RMSE) volíme *False*.

Technické parametry. Za zmínku stojí `tree_method`, který určuje, jakým algoritmem bude probíhat trénování jednoho regresního stromu. Výchozí parametr je *auto*, který heuristicky podle počtu dat vybírá buď přesný (který zkouší všechny kandidáty na split), nebo aproximativní algoritmus. Vzhledem k relativně malé komplexitě našeho problému předcházíme nejistotě v podobě výchozí hodnoty *auto* a rovnou volíme hodnotu *exact*, tedy přesný hladový algoritmus.

Ostatní parametry lze nalézt v dokumentaci. Jedná se už vesměs o parametry, které upravují chování aproximativních algoritmů pro trénování jednotlivých stromů, takže vzhledem k našemu nastavení na `tree_method` *exact* se nepoužijí.

[1] Žádný z modelů, které jsme v průběhu práce trénovali, nepřesáhl počtem iterací 200.

1.3.8 Důležitost prediktorů

Na rozdíl například od lineární regrese je efekt jednotlivých prediktorů na vysvětlovanou proměnnou skrytý hluboko ve struktuře modelu. Pro snadnější interpretaci efektů byla vytvořena řada nástrojů, jedním z nich je knihovna pro Python `shap` (***SH****apley* ***A****dditive ex****P****lanation*). Teorie k ní je popsaná v [22].

Tento postup využívá principu Shapleyho hodnot – ty zjednodušeně řečeno měří výkonnost modelu bez daného prediktoru a s ním. Rozdíl pak reprezentuje jeho důležitost. Konkrétně se Shapleyho hodnotou $\phi(p)$ prediktoru p nazve:

$$\phi(p) = \sum_{S \subseteq N \setminus \{p\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \left[y_x(S \cup \{p\}) - y_x(S) \right],$$

kde N je množina všech prediktorů, S je nějaká podmnožina všech prediktorů a $y_x(S)$ je podmíněná střední predikovaná hodnota stromu y při prediktorech x_S (mimo podmnožinu S jsou hodnoty nahrazeny chybějící hodnotou).

1.3.9 Predikční intervaly

Konstrukce predikčních intervalů v XGBoostu není příliš běžná úloha. V době uzavření obsahu této práce – v září 2020 – nebyla nalezena odborná studie, která by predikční intervaly spolu s předpovědí konstruovala.

Mimo odbornou literaturu lze nalézt metodu navrženou v [11], která konstruuje predikční intervaly pro kvantilovou regresi pomocí přidání randomizace do výpočtu gradientu. Navíc gradient, který je v případě kvantilové regrese skoková funkce, je v tomto řešení ještě zespojitěn lineární funkcí, jejíž směrnice je další hyperparametr.

V každém případě jde pro konstrukci intervalů použít neparametrický bootstrap.

2. Data

2.1 Zvolený index

Jako vstupní data použijeme denní data akciového indexu S&P 500 (Standard & Poor's 500). Tento index obsahuje akcie (přibližně) 500 největších (podle kapitalizace) na burze obchodovaných akcií v USA. Podle tržního podílu jsou také akcie v indexu váhově zastoupeny. Historie S&P 500 sahá do roku 1923, od roku 1957 pak udržuje velikost přibližně 500 akcií.

K 1. 6. 2020 jsou pěti akciemi s největším podílem v indexu:

- Microsoft Corporation (MSFT) se 4.39 %,
- Apple Inc. (AAPL) se 4.33 %,
- Amazon.com Inc. (AMZN) se 2.83 %,
- Facebook Inc. Class A (FB) s 1.80 % a
- Berkshire Hathaway Inc. Class B (BRK.B) s 1.67 %.

2.2 Zvolená časová řada

Jak už bylo popsáno v úvodu, modelovanou řadou bude denní zobchodovaný objem, tj. suma objemů všech jednotlivých obchodů s akciemi zastoupenými v tomto indexu (nákup i prodej).

2.3 Vstupní data

Zdrojem dat je finanční server `finance.yahoo.com`, kde jsou data v denní granularitě volně k dispozici.

K dispozici máme data v denní granularitě od 1. 1. 2000 do 31. 12. 2018 (pouze obchodní dny), celkem 4776 pozorování.

Vstupní data obsahují následující proměnné:

Proměnné společné pro všechny modely

Základními proměnnými jsou informace běžně dostupné na finančních portálech, sem patří:

- `date`: datum, primární klíč dat;
- `volume`: zobchodovaný objem tohoto dne (**predikovaná časová řada**) (v milionech USD);
- `previous_volume`: zobchodovaný objem předchozího obchodního dne (využíváme jako naivní predikci) (v milionech USD);

- `previous_intraday_range`: nejvyšší minus nejnižší cena předchozího obchodního dne;
- `previous_intraday_logreturn`: denní logaritmický výnos (počítaný z uzavírací[1] a otevírací ceny) předchozí obchodní den;
- `previous_intraday_logreturn_positive`: binární indikátor, zda je `previous_intraday_logreturn` > 0.

Dále definujeme vlastní prediktory:

- `day_name`: jméno dne v týdnu (1 = pondělí, ..., 5 = pátek);
- `month_name`: jméno měsíce (1 = leden, ..., 12 = prosinec);
- `days_since_last_trading_day`: počet dní od posledního obchodního dne (1 znamená, že se o den dříve obchodovalo);
- `days_until_next_trading_day`: počet dní do následujícího obchodního dne (1 znamená, že se o den později bude obchodovat);
- `year_month_rank`: inkrementální pořadí měsíce od ledna 2000 (1 = leden 2000, 2 = únor 2000, ..., 13 = leden 2001, ...);
- `important_event`: binární indikátor, zda se k danému dni vztahuje jedna z tzv. „důležitých událostí“, jejichž seznam lze nalézt níže;
- `important_event_previous_day`: binární indikátor, zda se k předchozímu obchodnímu dni vztahuje jedna z těchto událostí.

Proměnné specifické pro ARIMAX

Vzhledem k tomu, že ARIMAX je lineární model, přidáváme do dat následující proměnné, které umožní hledat i jiný než lineární trend:

- `year_month_rank_log`: log(`year_month_rank`),
- `year_month_rank_sqr`: `year_month_rank`^{2},
- `year_2009plus`: $\mathbb{1}$ (`date` $\geq$ 01. 01. 2009),
- `year_2011plus`: $\mathbb{1}$ (`date` $\geq$ 01. 01. 2011)

a interakce:

- `year_month_rank_2009plus_log`: `year_month_rank_log`·`year_2009plus`,
- `year_month_rank_2011plus_log`: `year_month_rank_log`·`year_2011plus`.

Data 1. 1. 2009 a 1. 1. 2011 jsou body zlomu zvoleny na základě explorativní analýzy v sekci 2.3.2.

Navíc definujeme logaritmickou transformaci predikované řady, kterou budeme testovat jako alternativu:

- `volume_log`: log(`volume`).

[1] uzavírací cena očištěná o dividendy a rozdělení akcií (*split*)

Proměnné specifické pro XGBoost

Vzhledem k tomu, že XGBoost není rekurzivní model, tj. v rámci pozorování nedostává informaci o zpožděných hodnotách, definujeme několik zpožděných hodnot `volume`. Volíme HA[2] (klouzavý průměr) a EWHA (exponenciálně vážený klouzavý průměr).

HA počítáme jako nevážený klouzavý průměr z posledních n dní (včetně aktuálního pozorování), kde n probíhá postupně hodnotami {2,3,4,5,10,20}. Maximální hodnota 20 odpovídá zhruba jednomu obchodnímu měsíci. Pro počátečních 19 pozorování je tato hodnota spočítaná pouze z dostupných pozorování. EWHA pro `volume` v čase t počítáme jako

$$\texttt{volume}_t = \frac{\sum_{i=1}^{t}(1-\alpha)^{t-i} \cdot \texttt{volume}_i}{\sum_{i=1}^{t}(1-\alpha)^{t-i}}, \tag{2.1}$$

kde α probíhá postupně hodnotami $\{0.1, 0.3, 0.5, 0.7, 0.9\}$.

Toto průměrování dává vzniknout proměnným `ha_volume_`α a `ewha_volume_`n`d`, kde α a n probíhají zmíněnými množinami.

„Důležitými událostmi“ se pro účely této analýzy rozumí následující důležité ekonomické zprávy v USA:

- *Nonfarm payroll* – součást zprávy o stavu pracovního trhu vydávané měsíčně United States Department of Labor (Ministerstvo práce USA). Informuje o nezaměstnanosti, vývoji počtu pracovních míst a podobně. Zpravidla vychází první pátek v měsíci, není-li kolize se svátkem.
- *FOMC Rate Decision* – rozhodnutí komise Federal Open Market Committee (FOMC) o úrokových sazbách na následující období. Zpravidla vychází ve středu jednou za 6 týdnů.

V tyto dny se obecně výrazně víc obchoduje, protože se investoři snaží co nejpružněji zareagovat na změny na trhu, které po zveřejnění takových zpráv zpravidla přichází.

Jak již bylo zmíněno výše, proměnná `important_event` je pouze binární indikátor. Trh ale obecně reaguje více na negativní zprávy než na ty pozitivní. Pokud by se tato proměnná ukázala být pro analýzu klíčová, bylo by dobré ji rozšířit například právě o informaci, jestli byla vydaná zpráva pro trh pozitivní nebo negativní.

[2] ***H**istorical **A**verage* používáme místo častějšího překladu ***M**oving **A**verage*, zkracovaného na MA, protože v sekci, kde se věnujeme ARIMAX modelu, má již MA zavedenou definici, která ovšem není konzistentní s tím, jak se termín *Moving Average* používá mimo kontext modelu ARMA. Zatímco v běžné terminologii znamená termín *Moving Average* různé autoregresní a někdy autoprogresivní nebo kombinované průměry, analogií v modelech ARMA není MA, nýbrž AR složka. MA v modelu ARMA označuje složku zpožděných nepozorovaných reziduí.

2.3.1 Rozdělení na trénovací, testovací a out-of-sample množinu

Data rozdělíme na tři části:

- in-sample trénovací množinu (od 1. 1. 2000 do 31. 12. 2015 včetně), 4024 pozorování,
- in-sample testovací množinu (od 1. 1. 2016 do 31. 12. 2017 včetně), 503 pozorování, a
- out-of-sample množinu (od 1. 1. 2018 do 31. 12. 2018 včetně), 249 pozorování.

2.3.2 Explorace dat

Od této sekce dál už neuvažujeme out-of-sample množinu až do chvíle, kdy ji využijeme pro porovnání modelů. Následující explorace se tedy týká trénovací a testovací množiny.

Nejprve vykreslíme řadu uzavíracích cen, dále celou zkoumanou řadu (`volume`) a její diference (Obrázek 2.1).

Z obrázku je patrné, že námi zkoumaná řada by mohla mít problémy se stacionaritou, nicméně diferencovaná řada už stacionární vypadá. Na první pohled si můžeme všimnout změny dynamiky řady v období hospodářské krize v letech 2008 až 2011. Na rozdíl od ceny, která v období krize – obzvlášť zpočátku – prudce klesala, objem naopak rostl. To je přirozené, neboť v období s více volatilní cenou a jejím poklesem je pravděpodobnější, že se investoři budou snažit obchodováním buď vyvarovat ztráty, nebo naopak využít nižší ceny k výhodnějšímu nákupu.

Popisné statistiky

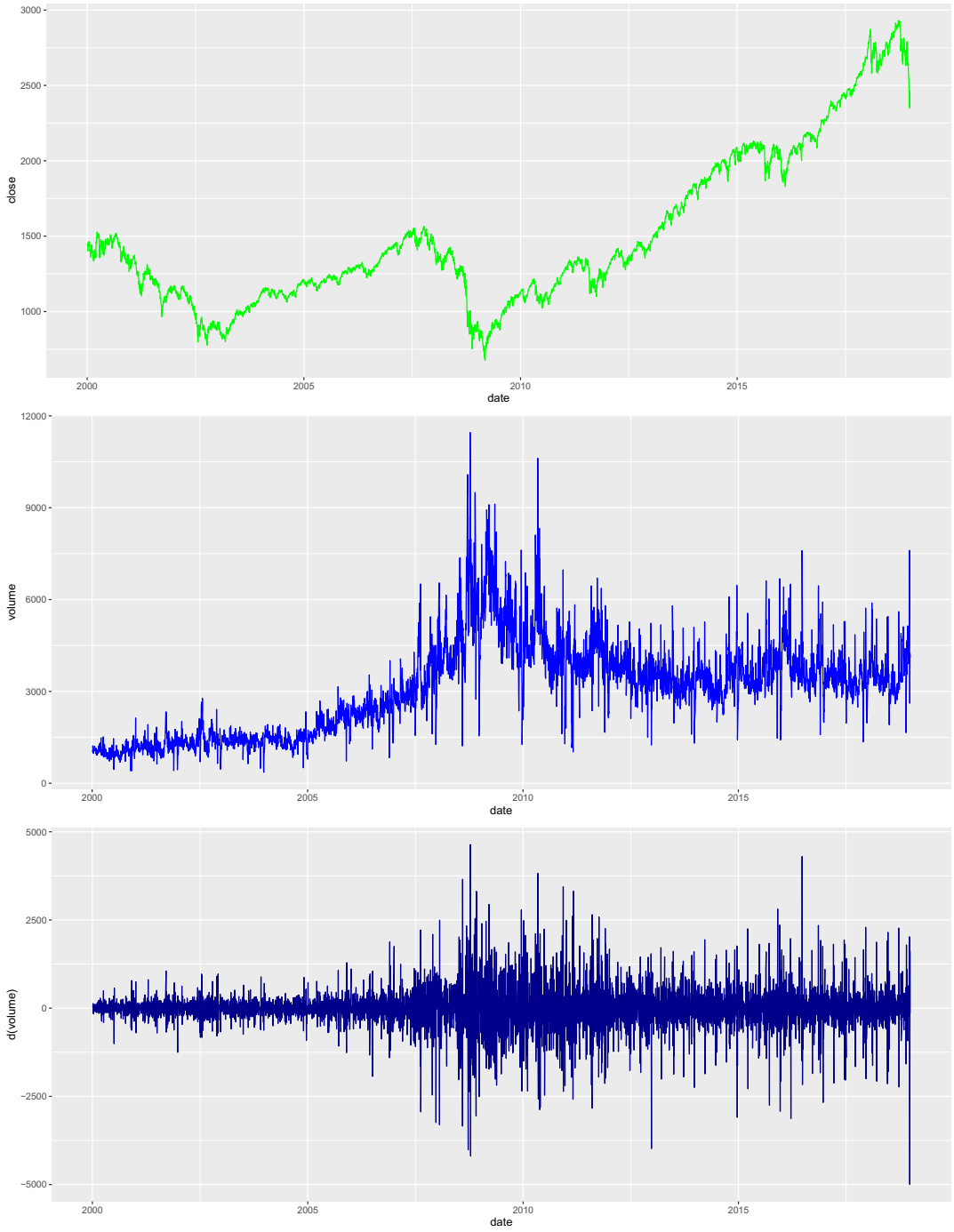
V datech máme tři spojité veličiny, jejich popisné statistiky nalezneme níže:

Proměnná	**Min**	Q_1	**Průměr**	**Medián**
`volume`	356.070	1613.530	3068.233	3173.295
`previous_intraday_range`	2.900	10.298	18.180	14.980
`previous_intraday_logreturn`	-0.091	-0.005	0.000	0.000

Proměnná	Q_3	**Max**	**SD**
`volume`	3930.168	11456.230	1503.062
`previous_intraday_range`	21.905	125.220	12.377
`previous_intraday_logreturn`	0.005	0.102	0.012

Dále máme v datech proměnné, které jsou buď faktorové, nebo nabývají pouze malého počtu hodnot. Jejich frekvenční tabulky nalezneme níže:

`day_name`	1	2	3	4	5
Počet	896	978	981	963	958



Obrázek 2.1: Grafy uzavírací ceny, `volume` a diferencované `volume`.

`month_name`	1	2	3	4	5	6	7	8	9	10	11	12
Počet	384	365	415	392	404	406	400	423	382	420	389	396

`days_since_last_trading_day` `days_until_next_trading_day`	1	2	3	4	5	7
Počet	3740	46	864	123	2	1

`important_event` `important_event_previous_day`	False	True
Počet	4249	527

3. Predikce pomocí modelu ARIMAX

V této kapitole máme k dispozici dvoje data: trénovací (4024 pozorování v období od 1. 1. 2000 do 31. 12. 2015 včetně) a dále testovací (503 pozorování v období od 1. 1. 2016 do 31. 12. 2017 včetně). Oba datové vzorky obsahují kromě predikované řady ještě 15 exogenních proměnných, které jsou popsány v sekci 2.3.

3.1 Určení řádů p, d, q

Prozatím analyzujeme kompletní časovou řadu (tedy trénovací i testovací data).

3.1.1 Formální testy stacionarity

Již při úvodní exploraci dat v sekci 2.3.2 bylo z grafu zřejmé, že řada by nemusela být stacionární. Provedeme tedy dva formální testy stacionarity:

KPSS test zamítá nulovou hypotézu, že řada je stacionární s $p < 0.01$ a nezamítá hypotézu, že diferencovaná řada je stacionární s $p > 0.1$. Výsledek tedy indikuje nutnost jedné diference.

ADF test zamítá nulovou hypotézu, že řada je nestacionární s $p < 0.01$. Výsledek tedy neindikuje nutnost diference.

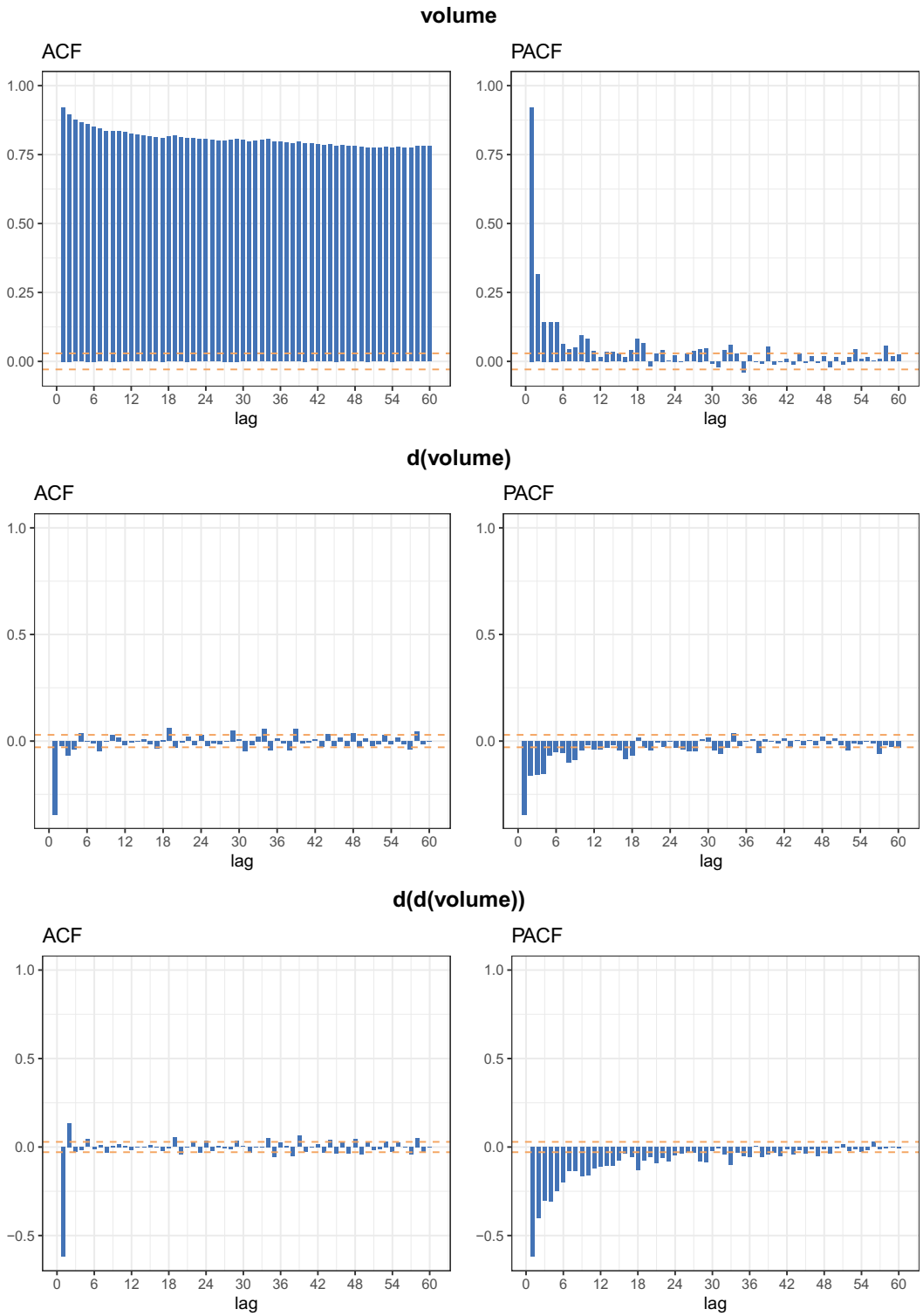
Celkový výsledek je tedy neprůkazný, a z toho důvodu nevyužijeme metodu formálních testů k volbě parametru d.

3.1.2 Volba stavového prostoru pro p, d, q

K určení řádů p, d, q využijeme metodu grid searche.

Na základě předchozí analýzy máme indicie, že optimální d bude $<= 1$. Abychom potvrdili, že tomu tak skutečně je, budeme chtít vidět systematické zhoršení při volbě $d = 2$. Navíc přidáme ještě $d = 3$ pro případ, že by formální testy byly příliš konzervativní. Grid search parametru d tedy proběhne přes hodnoty $\{0, 1, 2, 3\}$. Hodnoty parametru větší než 1 bývají velmi neobvyklé a uvažujeme je tedy hlavně kvůli zmíněnému potvrzení zhoršení vůči menším hodnotám.

Pro určení maximálního řádu p, resp. q se pokusíme použít korelogramy na Obrázku 3.1. Vzhledem k vysoké korelovanosti vysvětlované proměnné do vysokých řádů se kloníme k diferencování, správnost naší představy si nicméně potvrdíme při grid searchi. Nicméně na žádném z grafů nevidíme žádný bod useknutí, za kterým už by byla autokorelace nesignifikantní. Z toho důvodu pro grid search vezmeme do úvahy dostatečně velký řád p i q. Typicky se maximální řád volí v intervalu 10-12, nicméně vzhledem k dostupnému výpočetnímu výkonu volíme maximální hodnotu obou parametrů 20. To odpovídá 4 obchodním týdnům, a navíc dává grid searchi stejně jako v kontextu parametru d prostor k viditelnému zhoršení proti menším hodnotám parametrů.



Obrázek 3.1: Korelogramy a parciální korelogramy pro časovou řadu objemu, její první a druhé diference.

Pro oba parametry p a q tedy proběhne grid search přes hodnoty $\{0, 1, \ldots, 20\}$.

3.1.3 Účast exogenních proměnných

Vzhledem k faktu, že exogenní proměnné mohou nést řadu informací o předchozích pozorováních i o vlastních zpožděných hodnotách řady, nechceme určovat optimální řády p, d, q bez jejich účasti. Z důvodu výpočetní náročnosti však není možné do grid searche přidat i všechny kombinace exogenních proměnných.

Zvolíme tedy následující přístup. Maximálně očistíme řadu `volume` o vliv všech exogenních proměnných a na očištěné řadě spustíme grid search.

Očištěnou řadou pro nás budou rezidua lineárního a loglineárního modelu s interceptem formálně zapsanými jako:

$$\begin{aligned}\texttt{volume}_t = \alpha &+ \beta_1\texttt{day\_name}_t + \beta_2\texttt{month\_name}_t \\ &+ \beta_3\texttt{days\_since\_last\_trading\_day}_t \\ &+ \beta_4\texttt{days\_until\_next\_trading\_day}_t + \beta_5\texttt{year\_month\_rank}_t \\ &+ \beta_6\texttt{important\_event}_t + \beta_7\texttt{important\_event\_previous\_day}_t \\ &+ \beta_8\texttt{previous\_intraday\_range}_t + \beta_9\texttt{previous\_intraday\_logreturn}_t \\ &+ \beta_{10}\texttt{previous\_intraday\_logreturn\_positive}_t \\ &+ \beta_{11}\texttt{year\_month\_rank\_log}_t + \beta_{12}\texttt{year\_month\_rank\_sqr}_t \\ &+ \beta_{13}\texttt{year\_2009plus}_t + \beta_{14}\texttt{year\_2011plus}_t \\ &+ \beta_{15}\texttt{year\_month\_rank\_2009plus\_log}_t \\ &+ \beta_{16}\texttt{year\_month\_rank\_2011plus\_log}_t + \varepsilon_t,\ \varepsilon_t \overset{iid}{\sim} \mathcal{N}(0,\sigma^2) \qquad (3.1)\end{aligned}$$

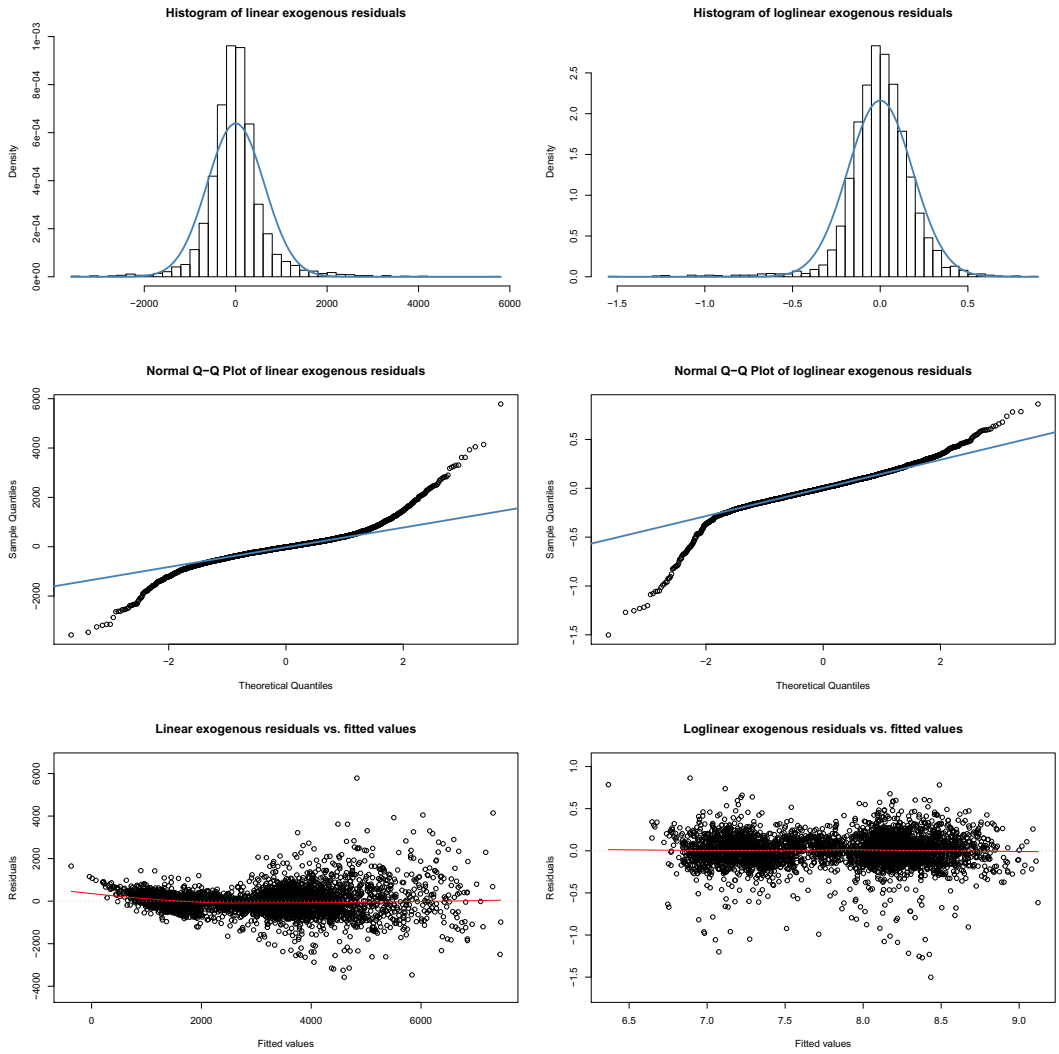
a

$$\begin{aligned}\log(\texttt{volume}_t) = \alpha &+ \beta_1\texttt{day\_name}_t + \beta_2\texttt{month\_name}_t \\ &+ \beta_3\texttt{days\_since\_last\_trading\_day}_t \\ &+ \beta_4\texttt{days\_until\_next\_trading\_day}_t + \beta_5\texttt{year\_month\_rank}_t \\ &+ \beta_6\texttt{important\_event}_t + \beta_7\texttt{important\_event\_previous\_day}_t \\ &+ \beta_8\texttt{previous\_intraday\_range}_t \\ &+ \beta_9\texttt{previous\_intraday\_logreturn}_t \\ &+ \beta_{10}\texttt{previous\_intraday\_logreturn\_positive}_t \\ &+ \beta_{11}\texttt{year\_month\_rank\_log}_t + \beta_{12}\texttt{year\_month\_rank\_sqr}_t \\ &+ \beta_{13}\texttt{year\_2009plus}_t + \beta_{14}\texttt{year\_2011plus}_t \\ &+ \beta_{15}\texttt{year\_month\_rank\_2009plus\_log}_t \\ &+ \beta_{16}\texttt{year\_month\_rank\_2011plus\_log}_t + \varepsilon_t,\ \varepsilon_t \overset{iid}{\sim} \mathcal{N}(0,\sigma^2). \qquad (3.2)\end{aligned}$$

Stavový prostor pro grid search tedy rozšiřujeme o binární parametr `use_log`, podle kterého se pokoušíme modelovat řadu reziduí lineárního, resp. loglineárního modelu.

Trénování modelu exogenních reziduí

Pojmem lineární, resp. loglineární (exogenní) reziduum budeme rozumět rezidua modelu (3.1), resp. jeho loglineární analogie (3.2).



Obrázek 3.2: Analýza reziduí při hledání exogenních lineárních a loglineárních reziduí pomocí histogramu, Q-Q plotu a scatter plotu reziduí proti vyrovnaným hodnotám.

Předpoklady lineárního modelu nejsou zcela splněny. To demonstruje například Obrázek 3.2, kde můžeme vidět, že rezidua těchto modelů nejsou zcela normálně rozdělena a model bez logaritmické transformace odezvy vykazuje mírnou heteroskedasticitu. Nicméně vzhledem k tomu, že smyslem tohoto modelu není predikce objemu, jedná se pouze o očištění časové řady o exogenní vlivy pro účely vybrání optimálního řádu (ne pro trénování finálního modelu), spokojíme se s modely, jak jsou.

Na Obrázku 3.3 vidíme, jak by vypadala jednoduchá predikce vysvětlované proměnné bez využití modelů časových řad – pouze na základě exogenních proměnných. Z grafu je vidět, že i takový model dokáže modelovat základní trend a sezónnost.

Obrázek 3.3: Predikce objemu pomocí exogenních proměnných, bez ARIMA struktury.

3.1.4 Grid search

Od této chvíle trénujeme modely pouze na trénovacích datech a testovací data používáme pro porovnání predikční síly jednotlivých modelů.

Grid search probíhá přes 4 parametry – binární `use_log`, diskrétní $d \in \{0, 1, 2, 3\}$ a diskrétní $(p, q)^T \in \{0, 1, \ldots, 20\}^2$. Celkově jsme tedy natrénovali 3528 modelů.

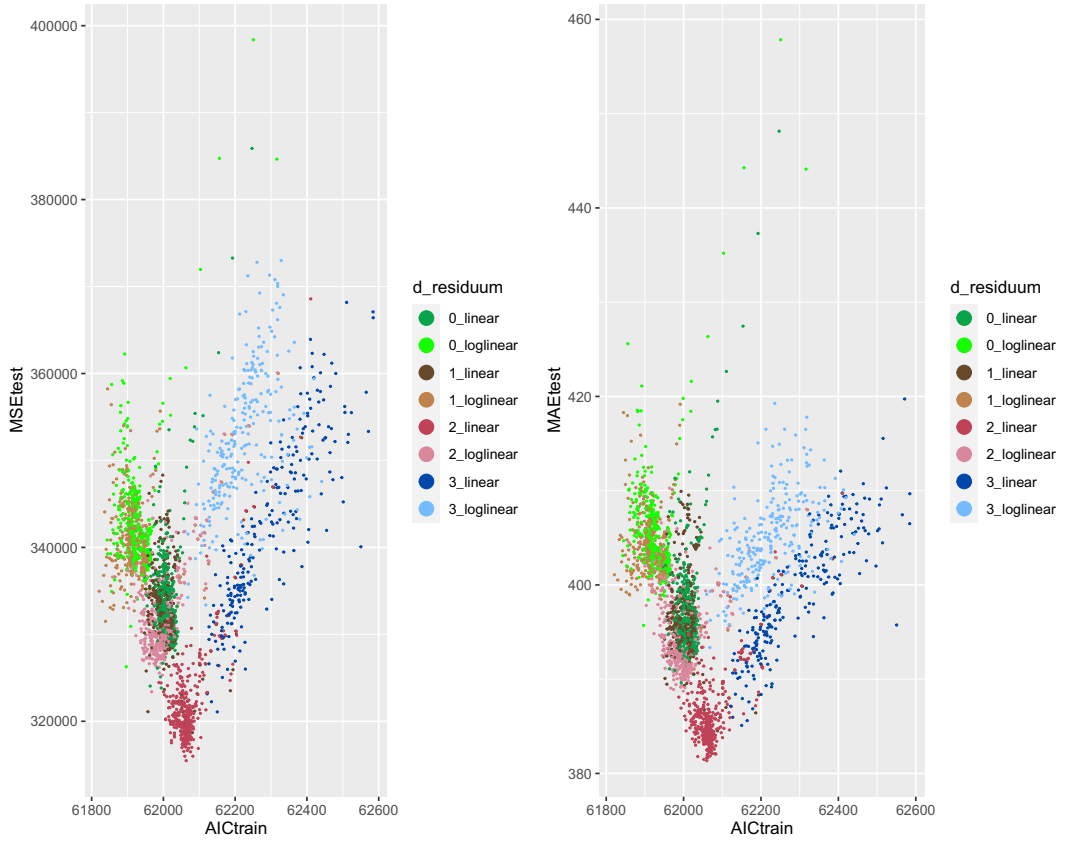
Jako kritéria kvality modelu volíme MSE (Mean Square Error) na testovací množině, MAE (Mean Absolute Error) na testovací množině a AIC (Akaikeho informační kritérium). Vzhledem k tomu, že hlavním úkolem našeho modelu je predikce, budeme kritériím na testovací množině přikládat větší váhu než AIC. Pokud by docházelo ke sporu mezi MSE a MAE, budeme mírně preferovat MSE, které je konzervativnější ve smyslu menší robustnosti vůči extrémně špatným predikcím.

3.1.4.1 Výběr d a `use_log`

Vykreslíme výsledky 3190 modelů – ty, které se v alespoň jednom kritériu dostaly mezi 80 % (tj. 2822) nejlepších. V grafu zanedbáme p a q a budeme se soustředit na závislost zmíněných kritérií na `use_log` a d.

Na obrázku 3.4 je patrné několik věcí:

- Pro všechny hodnoty d jsou modely lineárních reziduí téměř bez výjimky lepší než modely loglineárních reziduí, pokud jejich kvalitu měříme pomocí



Obrázek 3.4: Graf popisující závislost AIC, MSE a MAE na d a `use_log` napříč všemi řády p a q.

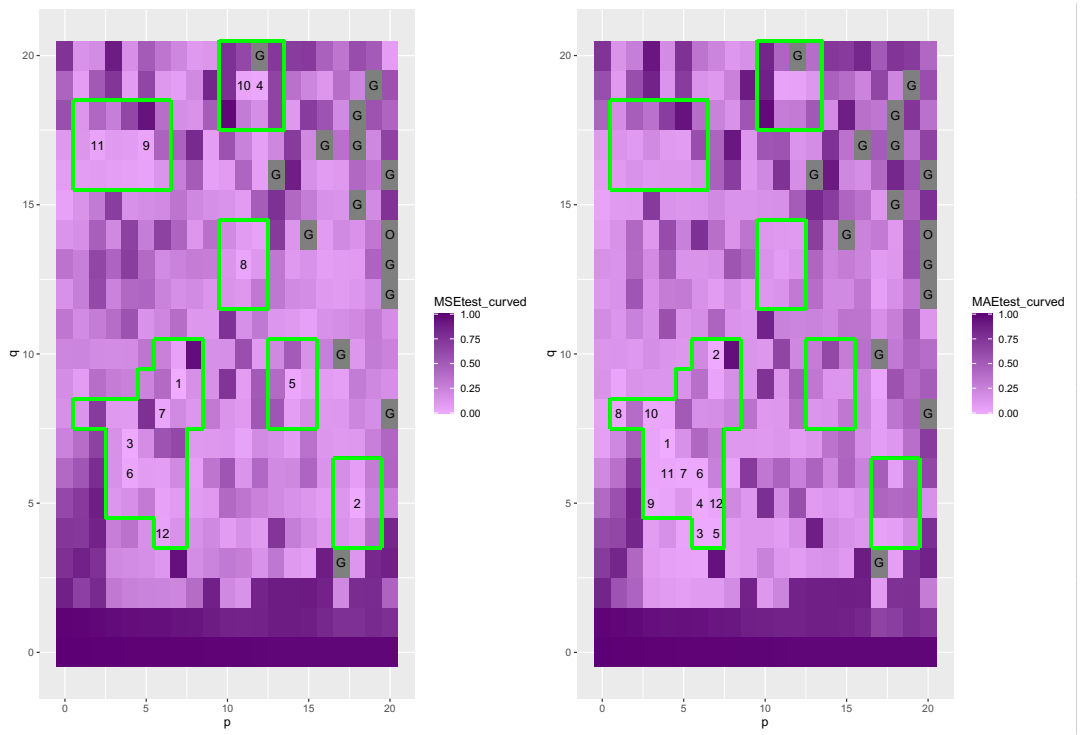
AIC.

- Pro hodnoty $d = 0$ a $d = 1$ jsou modely lineárních reziduí srovnatelně dobré až mírně horší než modely loglineárních reziduí, pokud jejich kvalitu měříme pomocí MSE nebo MAE.
- Pro hodnoty $d = 2$ a $d = 3$ jsou modely lineárních reziduí výrazně lepší než modely loglineárních reziduí, pokud jejich kvalitu měříme pomocí MSE nebo MAE.
- Měřeno pomocí AIC, nejlepších výsledků dosahují modely lineárních reziduí s $d = 0$ a $d = 1$.
- Měřeno pomocí MSE nebo MAE, **nejlepších výsledků dosahují modely lineárních reziduí s $d = 2$.**

Kritéria AIC a MSE dávají shodný závěr, co se týče výběru loglineárních nebo lineárních reziduí – lepších výsledků dosahují modely lineárních reziduí. Liší se ve výběru optimálního řádu diference. Na základě MSE ale víme, že model s $d = 2$ dokáže minimálně po dobu dvou let tuto řadu predikovat lépe. Protože naším hlavním úkolem je vytvoření predikčního a nikoliv popisného modelu, dáme verzi s $d = 2$ přednost. **Dále pokračujeme ve výběru p a q pouze mezi modely s $d = 2$ a lineárními rezidui.**

3.1.4.2 Výběr p a q

Vzhledem k závěrům předchozí sekce máme nyní stavový prostor pouze dvoudimenzionální, proto vykreslíme heatmapu všech natrénovaných modelů a podíváme se na vizualizaci závislosti kritérií kvality modelu. Již budeme zkoumat výsledky pouze pro MSE a MAE, neboť při výběru parametrů d a `use_log` se závěry plynoucí z AIC lišily od závěrů plynoucích z MSE a MAE a my jsme upřednostnili závěry z MSE a MAE.



Obrázek 3.5: Graf popisující závislost MSE a MAE na p, q při `use_log = False` a $d = 2$. Seznam chybových kódu je v Příloze A.1. Barevná škála reprezentuje monotónní, ale nelineární transformaci MSE, resp. MAE, tak, aby byla barevná škála reprezentativní.

Z Obrázku 3.5 je vidět, že existují oblasti (navzájem blízkých dvojic p, q ve smyslu metriky $p + q$), kde se vyskytují nejlepší modely buď podle MSE, nebo

podle MAE. Vybereme konkrétně oblasti:

$$\begin{aligned}
PQ := &\left\{(p,q)^T,\, p \in \{1,2,\ldots,6\},\, q \in \{16,17,18\}\right\} \\
&\cup \left\{(p,q)^T,\, p \in \{10,11,12,13\},\, q \in \{18,19,20\}\right\} \\
&\cup \left\{(p,q)^T,\, p \in \{10,11,12\},\, q \in \{12,13,14\}\right\} \\
&\cup \left\{(p,q)^T,\, p \in \{13,14,15\},\, q \in \{8,9,10\}\right\} \\
&\cup \left\{(p,q)^T,\, p \in \{17,18,19\},\, q \in \{4,5,6\}\right\} \\
&\cup \left\{(p,10)^T,\, p \in \{6,7,8\}\right\} \\
&\cup \left\{(p,9)^T,\, p \in \{5,6,7,8\}\right\} \\
&\cup \left\{(p,8)^T,\, p \in \{1,2,\ldots,8\}\right\} \\
&\cup \left\{(p,q)^T,\, p \in \{3,4,\ldots,7\},\, q \in \{5,6,7\}\right\} \\
&\cup \left\{(p,4)^T,\, p \in \{6,7\}\right\}.
\end{aligned}$$

S těmito oblastmi budeme pokračovat v analýze.

3.2 Výběr exogenních proměnných

Momentálně máme vybráno 89 trojic $(p,d,q)^T$, konkrétně

$$PDQ := \left\{(p,2,q)^T,\, (p,q)^T \in PQ\ \right\},$$

které uvažujeme pro další analýzu. Exogenní proměnné rozdělíme do 4 téměř disjunktních skupin proměnných:

- `days_months`: `day_name`, `month_name`, `days_since_last_trading_day`, `days_until_next_trading_day`
- `trend`: `year_month_rank`, `year_month_rank_log`, `year_month_rank_sqr`, `year_2009plus`, `year_2011plus`, `year_month_rank_2009plus_log`, `year_month_rank_2011plus_log`
- `price`: `previous_intraday_range`, `previous_intraday_logreturn`, `previous_intraday_logreturn_positive`
- `important_event`: `day_name`, `important_event`, `important_event_previous_day`

Proměnná `day_name` je přítomna nejen v množině `days_months`, ale i `important_event`, protože tyto důležité události nastávají vždy ve stejný den v týdnu.

3.2.1 Základní modely

Základními modely (`base_models`) rozumíme modely ARIMAX(p,d,q), kde $(p,d,q)^T \in PDQ$.

3.2.2 Skupiny exogenních proměnných

Pro každou skupinu proměnných a pro každý základní model zkusíme přidat všechny kombinace exogenních proměnných v dané skupině. Poté vybereme jednu nebo víc nejslibnějších kombinací, které budeme uvažovat při výběru finálního modelu.

Postup pro výběr těchto kombinací bude následující:

1. Pro každou kombinaci proměnných (a pro základní modely) vybereme 10 nejlepších modelů přes p, d, q a zjistíme, jestli daná kombinace proměnných zlepšila průměrný výkon modelu (na základě MSE a MAE). Pokud ne, danou kombinaci nebudeme vůbec uvažovat.
2. Z kombinací, které splnily podmínku 1, vybereme ty, ve kterých se vyskytují nejlepší modely napříč všemi kombinacemi. Tuto vlastnost budeme zkoumat primárně pohledem do grafu.

`days_month`

Pro zjednodušení přiřadíme jednotlivým proměnným kódy:

Proměnná	Kód
`day_name`	A
`month_name`	B
`days_since_last_trading_day`	C
`days_until_next_trading_day`	D

Podmínku 1 podle MSE na testovací množině splnily následující kombinace (v pořadí od nejlepší průměrné výkonnosti na nejlepších 10 modelech): CD, BCD, BD, B, D, ABCD. Podle MAE navíc AB, AD, AC a ABD.

Na základě grafické analýzy (viz Obrázek A.1) vybíráme jedinou kombinaci: CD.

Do modelů tedy uvažujeme proměnné `days_since_last_trading_day` a `days_until_next_trading_day`.

`trend`

Jednotlivým proměnným opět přiřadíme kódy:

Proměnná	Kód
`year_month_rank`	A
`year_month_rank_log`	B
`year_month_rank_sqr`	C
`year_2009plus`	D
`year_2011plus`	E
`year_month_rank_2009plus_log`	F
`year_month_rank_2011plus_log`	G

Podmínku 1 podle MSE na testovací množině splnily následující kombinace (v pořadí od nejlepší průměrné výkonnosti na nejlepších 10 modelech): BD a D. Podle MAE to byly BFG, BDE a G.

Dáme přednost MSE a vybíráme obě kombinace: BD a D. Grafické znázornění výkonnosti modelů je na Obrázcích A.2 a A.3.

Do modelů tedy uvažujeme proměnnou `year_2009plus` a budeme testovat přítomnost proměnné `year_month_rank_log`.

`price`

Jednotlivým proměnným opět přiřadíme kódy:

Proměnná	**Kód**
`previous_intraday_range`	A
`previous_intraday_logreturn`	B
`previous_intraday_logreturn_positive`	C

Podmínku 1 podle MSE na testovací množině splnily všechny kombinace s výjimkou BC. Podle MAE všechny kromě BC a A.

Na základě grafické analýzy (viz Obrázek A.4) vybíráme kombinace A, AB, AC a ABC.

Do modelů tedy uvažujeme proměnnou `previous_intraday_range` a budeme testovat přítomnost proměnných `previous_intraday_logreturn` a `previous_intraday_logreturn_positive` (samostatně i společně).

`important_event`

Jednotlivým proměnným opět přiřadíme kódy:

Proměnná	**Kód**
`day_name`	A
`important_event`	B
`important_event_previous_day`	C

Podmínku 1 podle MSE na testovací množině splnily AC a ABC. Podle MAE navíc BC.

Na základě grafické analýzy (viz Obrázek A.5) vybíráme jedinou kombinaci ABC.

Nicméně proměnnou `day_name` jsme již testovali, když jsme analyzovali proměnné ve skupině `days_month` a rozhodli jsme se ji v modelu vůbec netestovat. Vzhledem k opačným závěrům v obou skupinách proměnných volíme bezpečný způsob – přítomnost `day_name` budeme ještě testovat.

Do modelů tedy uvažujeme proměnné `important_event` a `important_event_previous_day` a budeme testovat přítomnost proměnné `day_name`.

3.2.3 Finální skupina proměnných

V předchozí čtyřech sekcích jsme tedy rozhodli, že do modelu zahrneme proměnné `important_event`, `important_event_previous_day`, `days_since_last_trading_day`, `days_until_next_trading_day`, `previous_intraday_range` a `year_2009plus` (základní exogenní proměnné).

Proměnné, které do modelů se základnímu exogenními proměnnými budeme navíc zkoušet přidat, jsou `day_name`, `year_month_rank_log`, `previous_intraday_logreturn` a `previous_intraday_logreturn_positive` (skupina `final`). Opět označíme kódy:

Proměnná	**Kód**
`day_name`	A
`year_month_rank_log`	B
`previous_intraday_logreturn`	C
`previous_intraday_logreturn_positive`	D

Zvolíme analogický způsob jako v jednotlivých sekcích, nicméně v podmínce 1 budeme požadovat kromě zlepšení základních modelů ještě zlepšení modelu se základními exogenními proměnnými.

Tohoto zlepšení na MSE dosáhly modely s proměnnými AC, ABC, ABD, ACD, ABCD, B, C, BD a BCD. Na MAE to byly všechny kombinace modelů neobsahující A. Grafická reprezentace je na Obrázku A.6.

3.2.4 Finální model

Nyní již vybíráme nejlepší model. Vzhledem k blízkým hodnotám MSE u nejlepších modelů volíme nejlepších modelů více (konkrétně 10), predikcí našeho postupu pak myslíme průměrnou predikci těchto 10 modelů.

Těchto 10 modelů popíšeme následující tabulkou:

#	p	d	q	**Sada proměnných**
8	2	2	8	$X \cup \{\texttt{PIL}, \texttt{PILP}\}$
4	2	2	16	$X \cup \{\texttt{YMRL}, \texttt{PILP}\}$
10	1	2	16	$X \cup \{\texttt{YMRL}, \texttt{PILP}\}$
2	3	2	18	$X \cup \{\texttt{YMRL}, \texttt{PIL}\}$
3	3	2	8	$X \cup \{\texttt{YMRL}, \texttt{PILP}\}$
5	5	2	7	$X \cup \{\texttt{YMRL}, \texttt{PIL}\}$
7	5	2	5	$X \cup \{\texttt{YMRL}, \texttt{PILP}\}$
9	6	2	5	$X \cup \{\texttt{YMRL}, \texttt{PILP}\}$
6	7	2	9	$X \cup \{\texttt{YMRL}, \texttt{PILP}\}$
1	13	2	9	$X \cup \{\texttt{YMRL}, \texttt{PILP}\}$

Tabulka 3.1: Deset nejlepších modelů s původním pořadím v prvním sloupci.

$X = \{$`important_event`, `important_event_previous_day`, `days_since_last_trading_day`, `days_until_next_trading_day`, `previous_intraday_range`,

`year_2009plus`, `day_name`}.
`YMRL` je zkratka `year_month_rank_log`.
`PIL` je zkratka `previous_intraday_logreturn`.
`PILP` je zkratka `previous_intraday_logreturn_positive`.

Všechny tyto modely natrénujeme znovu tak, aby brali trénovací i testovací množinu, tj. všechna dostupná data kromě out-of-sample množiny, kterou použijeme na porovnání s jinými třídami modelů.

3.3 Verifikace předpokladů modelu

Vlastnosti reziduí všech 10 modelů shrnuje Tabulka 3.2.

#	Průměr	σ	p L-B[1]	p B-P[2]	p S-W[3]
1	-3.19	512.24	>0.999	0.296	<0.001
2	-4.05	512.95	>0.999	0.045	<0.001
3	-2.82	515.74	0.635	0.200	<0.001
4	-6.36	514.39	>0.999	0.368	<0.001
5	-2.15	512.56	0.420	0.081	<0.001
6	-3.31	513.51	0.512	0.175	<0.001
7	-2.76	515.76	0.353	0.368	<0.001
8	-2.78	514.69	0.032	0.249	<0.001
9	-4.22	515.43	0.561	0.287	<0.001
10	-3.12	514.43	>0.999	0.156	<0.001

[1] p-hodnota Ljung-Boxova testu
[2] p-hodnota Breusch-Paganova testu
[3] p-hodnota Shapiro-Wilkova testu

Tabulka 3.2: Popisné statistiky reziduí a p-hodnoty testů na jejich nekorelovanost, homoskedasticitu a normalitu. Za pořadím dle MSE následuje průměrná hodnota a směrodatná odchylka vektoru reziduí. V posledních třech sloupcích jsou po řadě hodnoty Ljung-Boxova, Breusch-Paganova a Shapiro-Wilkova testu, resp. jejich p-hodnoty.

Pro test zbylé autokorelace mezi rezidui volíme Ljung-Boxův test popsaný v [21], hledáme autokorelaci až do řádu 9. Hodnotu 9 jsme určili na základě běžně používaného pravidla $\lceil \log(n) \rceil$. Pro testování homoskedasticity volíme Breusch-Paganův test popsaný v [7] a pro normalitu Shapiro-Wilkův test popsaný v [25].

Vidíme, že pro všechny modely je střední hodnota reziduí lehce vychýlená směrem k záporným hodnotám, přestože v modelu máme intercept. To může být způsobeno ne zcela symetrickým rozdělením reziduí. Nicméně hodnota tohoto vychýlení je o dva řády nižší než směrodatná odchylka, což je v podstatě zanedbatelné.

V tabulce je dále vidět, že s výjimkou modelu 8 nezamítáme na hladině 5 % nulovou hypotézu, že v reziduích do řádu 9 není autokorelace. Na stejné hladině nezamítáme s výjimkou modelu 2 ani homoskedasticitu.

Rezidua příliš neodpovídají normálnímu rozdělení. To nemá vliv na nestrannost nebo konzistenci odhadů. Nicméně pokud bychom pomocí těchto modelů predikovali intervaly nebo pásy spolehlivosti, jejich délka by nebyla ani asymptoticky

správná – to však v našem postupu neděláme. Další riziko je špatně spočítané AIC, nicméně modely byly určeny na základě MSE na testovací množině, takže pro jejich výběr nehrálo AIC roli.

Co se týče zamítnutí nulové hypotézy nekorelovanosti reziduí, resp. homoskedasticity u dvou modelů, rozhodli jsme se je v celkovém modelu (který vzniká jako průměr predikcí jednotlivých částečných) ponechat. Důvodem je to, že p-hodnoty jsou velmi blízko kritické hodnotě 0.05 a my provádíme mnohonásobné testování. Z definice p-hodnoty za předpokladu splnění obou hypotéz má každý model pravděpodobnost 0.05 zamítnutí této platné nulové hypotézy. V našem případě je výskyt 1 z 10 na obou kritériích, což je jen dvakrát větší četnost než střední a může být celkem snadno pouze dílem náhody. Při aplikaci Bonferoniho metody bychom nezamítli žádnou hypotézu (s výjimkou normality), nicméně hladina takového postupu by už mohla být nižší než 0.05.

3.4 Zhodnocení zvoleného přístupu

Začali jsme s velmi širokou množinou potenciálních modelů a postupným ořezáváním prostoru možných hodnot hyperparametrů a výběrem proměnných jsme se dostávali k finálnímu modelu. Tento přístup je pochopitelně suboptimální a není jisté, že námi nalezený model je nejlepší ve smyslu námi používaných kritérií. Je možné, že jsme vybrali v jistém smyslu lokálně, nikoliv globálně nejlepší model.

Nicméně testování všech modelů je časově neúnosná úloha. O výpočetní náročnosti námi zvoleného postupu hovoří podrobněji sekce 5.2.

V každém jednotlivém kroku tohoto postupu (výběr p a q, výběr exogenních proměnných v jednotlivých skupinách, výběr exogenních proměnných z finální skupiny) se signifikantně snížilo sledované kritérium MSE proti předchozímu kroku. I to nás vede k domněnce, že postup je v rámci technických možností přijatelným kompromisem.

U komponent výsledného modelu bylo s celkovým úspěchem ověřeno, že klíčové předpoklady modelu ARIMAX nejsou nesplněny. I z toho důvodu věříme, že výsledný model je dobrým kandidátem třídy ARIMAX na úspěšnou predikci na out-of-sample množině.

4. Predikce pomocí modelu XGBoost

V této kapitole máme k dispozici stejně jako při modelování pomocí ARIMAX dvoje data: trénovací (4024 pozorování v období od 1. 1. 2000 do 31. 12. 2015 včetně) a testovací (503 pozorování v období od 1. 1. 2016 do 31. 12. 2017 včetně). Oba datové vzorky obsahují kromě predikované ještě 20 exogenních proměnných, které jsou popsány v sekci 2.3.

XGBoost model neumí využít uspořádanosti pozorování, neumí využít zpožděné pozorování a zpožděná rezidua. Z toho důvodu je polovina exogenních proměnných reprezentací zpožděných hodnot. Nicméně tento fakt znamená, že již nemusíme na pozorování nahlížet jako na uspořádanou časovou řadu a můžeme použít běžnější přístup k trénování a validaci modelu pomocí cross-validace.

Obě množiny (tedy trénovací a testovací) tedy spojíme dohromady na nová trénovací data, máme tedy 4527 pozorování.

Budeme postupovat následujícím způsobem:

- Nejprve expertně určíme sadu výchozích hyperparametrů.
- Pomocí běžných metrik popisujících míru shody predikovaných a skutečných hodnot určíme optimální skupinu proměnných pro trénování modelu.
- Na všech dostupných datech (s výjimkou out-of-sample dat), která rozdělíme náhodně na trénovací a testovací, natrénujeme „benchmarkový“ model s výchozími hyperparametry.
- Optimalizujeme postupně jednotlivé hyperparametry a průběžně porovnáváme výsledky se zmíněným „benchmarkovým“ modelem.

4.1 Trénování pomocí cross-validace

Pro cross-validaci použijeme počet foldů 5. To znamená, že celý proces trénování modelu zopakujeme pětkrát. V každé iteraci budou 4/5 dat trénovací a 1/5 tzv. cross-validační množina. Zmíněná pětina jsou pokaždé jiná pozorování, takže v rámci celého procesu vystupuje každé pozorování čtyřikrát jako součást trénovací a jednou jako součást cross-validační množiny.

Cross-validační množina má dvě funkce:

- jednak vystupuje pro každou iteraci jako množina, na které se měří výkonnost postupně vznikajícího modelu, a určuje, kdy se trénování zastaví,
- jednak je predikce pro cross-validační množinu považována za predikovanou in-sample hodnotu modelu a měří se pomocí ní výkonnost modelu na trénovací množině.

Tedy zmíněná predikce nemá vlastnosti ani tradiční in-sample predikce (vyrovnané hodnoty), ani tradiční out-of-sample predikce. I proto je třeba mít na pa-

měti, že porovnání míry shody těchto hodnot s realitou není možné dělat napříč rodinami modelů.

4.2 Výchozí hyperparametry

Již v sekci 1.3.7 jsme představili hyperparametry modelu XGBoost a u většiny popsali, jaké výchozí hodnoty budeme používat a proč. Tyto hodnoty ještě jednou sepíšeme.

Parametry pro výběr boosteru

Pro výběr boosteru volíme následující parametry:

- `booster` *gbtree*,
- `objective` *reg:linear*,
- `eval_metric` *rmse*,

což dohromady znamená, že budeme trénovat rozhodovací stromy (`booster`) pro predikci přímo hodnoty `volume` (`objective`) a kvalitu modelu budeme posuzovat pomocí RMSE (Root Mean Square Error)

Parametry pro booster

Pro booster začneme s následujícími výchozími parametry, později je budeme optimalizovat:

- `eta` 0.2,
- `gamma` 0,
- `max_depth` 4,
- `subsample` 0.8,
- `colsample_bytree` 0.8,
- `min_child_weight` 1,
- `lambda` 1,
- `alpha` 0.

4.3 Skupiny proměnných

XGBoost obecně nemá problém s přetrénováním z důvodu velkého počtu prediktorů, protože implicitně obsahuje algoritmus pro výběr prediktorů (tzv. *feature selection*) a zastavovací kritérium proti přetrénování. Nicméně konzistentně s přístupem, který jsme použili při modelování pomocí ARIMAX, ověříme postupným přidáváním skupin proměnných, zda má nejlepší predikční sílu model se všemi proměnnými. Definujeme proto skupiny proměnných následujícím způsobem:

- `important_event`: `important_event`, `important_event_previous_day`;

- `trend`: `year_month_rank`;
- `days_month`: `day_name`, `month_name`, `days_since_last_trading_day`, `days_until_next_trading_day`;
- `volume`: `ha_volume_`α, `ewha_volume_`n`d`, kde $\alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ a $n \in \{2,3,4,5,10,20\}$;
- `price`: `previous_intraday_range`, `previous_intraday_logreturn`.

Dále ještě spojíme skupiny dle businessově-matematické logiky:

- `volume_price`: `volume` $\cup$ `price`,
- `calendar_trend_events`: `days_month` $\cup$ `trend` $\cup$ `important_event`,
- `all`: `volume_price` $\cup$ `calendar_trend_events`, tedy všech 20 proměnných.

4.3.1 Výsledky

Jako „benchmarkový“ model použijeme naivní model (`naive`), což je o jeden krok zpožděná odezva.

Model	**MSE**	**MAE**	R^2
`naive`	368 303.6	379.5	0.843
`volume`	313 199.4	356.4	0.866
`price`	2 209 448.0	1165.2	0.046
`trend`	353 685.1	373.2	0.849
`days_months`	2 364 453.1	1236.1	-0.011
`important_event`	2 335 984.9	1231.8	0.002
`volume_price`	314 765.6	355.1	0.865
`calendar_trend_events`	328 811.3	358.5	0.859
`all`	268 866.0	323.2	0.885

Tabulka 4.1: Tabulka výsledků modelů XGBoost s výchozími hyperparametry a s různými skupinami proměnných (sloupec **Model**). Metriky popisující míru shody jsou počítány na cross-validační množině.

Z výsledků je zjevné, že přidávání proměnných do modelu obecně zlepšuje fit v rámci všech měřených metrik. Nadále budeme tedy pokračovat s modelem `all`, který využívá všechny proměnné.

4.4 Neoptimalizovaný model

Nyní bývá běžné vzít všechna data pro trénování finálního modelu. Když se berou všechna data, není možnost využít zastavovací kritérium, neboť nemáme nezávislou testovací množinu. V takovém případě se obvykle bere fixně průměrný počet iterací z trénování na cross-validační množině, případně se tento počet expertně upravuje, aby reflektoval změnu rozsahu trénovacích dat.

My jsme se přesto rozhodli využít rozdělení in-sample dat na trénovací a testovací množinu v poměru 80:20 (tedy stejně jako při cross-validaci), abychom nemuseli

počet iterací odhadovat. Na testovací množině se počítá zastavovací kritérium a také metriky popisující míru shody se skutečnými hodnotami.

Reálně bychom nemuseli trénovat nový model, ekvivalentní by bylo vzít libovolný z 5 modelů z cross-validačního přístupu, nicméně model natrénujeme z terminologických důvodů znovu na nových náhodně rozdělených datech a výsledek označíme jako `neoptimalizovaný model` vzhledem k faktu, že využívá neoptimalizované výchozí hyperparametry. Tento model pro nás bude benchmark pro následnou optimalizaci hyperparametrů.

Pro jistotu ověříme v Tabulce 4.2, že nedošlo k výrazné změně metrik při opětovném natrénování modelu.

Model	**MSE**	**MAE**	R^2
`all`	268 866.0	323.2	0.885
`neoptimalizovaný model`	265 581.4	328.4	0.875

Tabulka 4.2: Tabulka výsledků modelů XGBoost s výchozími hyperparametry. Metriky popisující míru shody jsou počítány pro `all` na cross-validační množině, pro `neoptimalizovaný model` na testovací množině.

4.5 Optimalizace hyperparametrů

Optimalizovat budeme 8 hyperparametrů popsaných v sekcích 1.3.7 a 4.2. Tyto parametry pro přehlednost vypíšeme spolu s jejich výchozí hodnotou (v našem trénování), doménou a typickými hodnotami.

Parametr	**Kategorie**	**Vých. h.**	**Doména a typické hodnoty**
`max_depth`	Komplexita	4	$\mathbb{N}$, (typicky $<= 10$)
`min_child_weight`	Komplexita	1	$\mathbb{N}_0$, (typicky $<= 10$)
`gamma`	Komplexita	0	$\mathbb{R}_0^+$, (typicky 10^{-4} až 10^4)
`subsample`	Robustnost	0.8	$(0, 1]$, (typicky $>= 0.5$)
`colsample_bytree`	Robustnost	0.8	$(0, 1]$, (typicky $>= 0.5$)
`eta`	Robustnost	0.2	$(0, 1]$, (typicky $<= 0.3$)
`alpha`	Regularizace	0	$\mathbb{R}_0^+$, (typicky 10^{-4} až 10^4)
`lambda`	Regularizace	1	$\mathbb{R}_0^+$, (typicky 10^{-4} až 10^4)

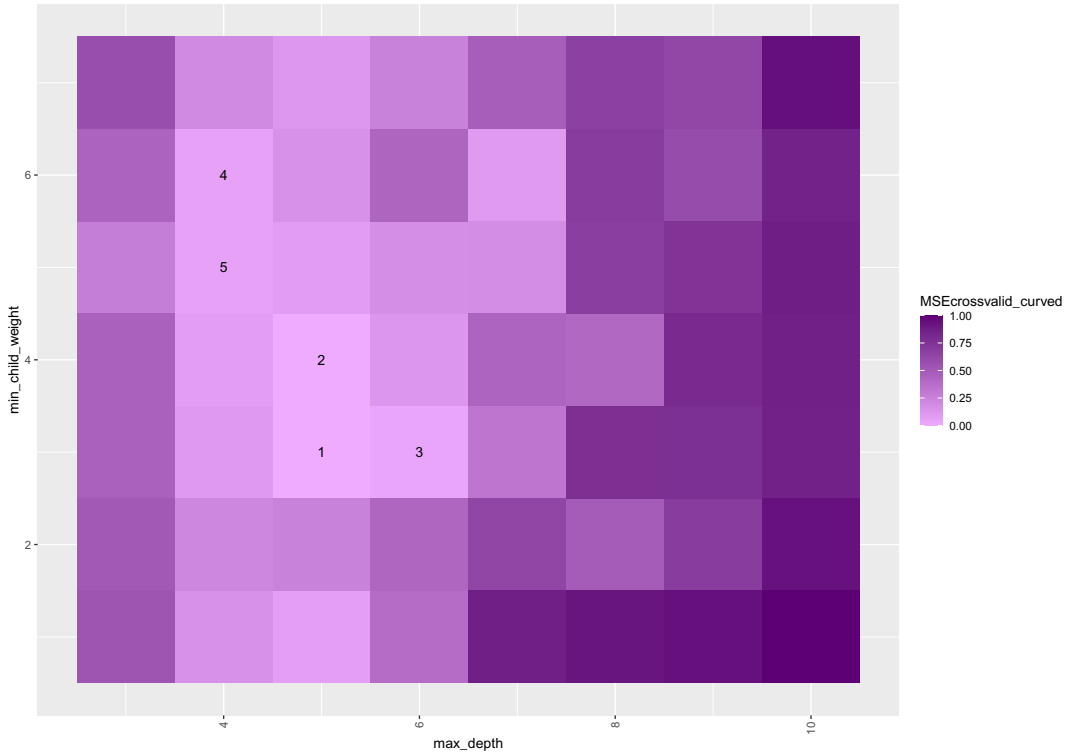
Tabulka 4.3: Tabulka hyperparametrů před optimalizací.

Z důvodu časově výpočetní náročnosti není možné optimalizovat všechny parametry najednou. Situaci navíc komplikuje i to, že u některých parametrů (`gamma`, `alpha`, `lambda`) neznáme ani řád, kde se optimální hodnota nachází – tedy musíme optimalizovat vícekrokově. Parametry na sobě ovšem mohou záviset a řada studií postupně prokázala, že ne všechna pořadí optimalizace parametrů jsou stejně vhodná.

V našem postupu vycházíme převážně z [23]. Optimalizace bude probíhat grid searchem po skupinách proměnných. Modely porovnáme pomocí MSE na cross-validační množině. U proměnných, kde není znám řád, využijeme dvoukrokovou optimalizaci (nejprve řád, potom hodnotu). Každý optimalizovaný parametr zafixujeme a další parametry už trénujeme s jeho optimální hodnotou.

Komplexita modelu

Jako první optimalizujeme dvojici parametrů `max_depth` a `min_child_weight` přes hodnoty $\{3, 4, \ldots, 10\} \times \{1, 2, \ldots, 7\}$. Na Obrázku 4.1 vidíme, že nejlepších výsledků dosahuje model s `max_depth` $= 5$ a `min_child_weight` $= 3$. Tyto parametry tedy fixujeme a pokračujeme v optimalizaci.



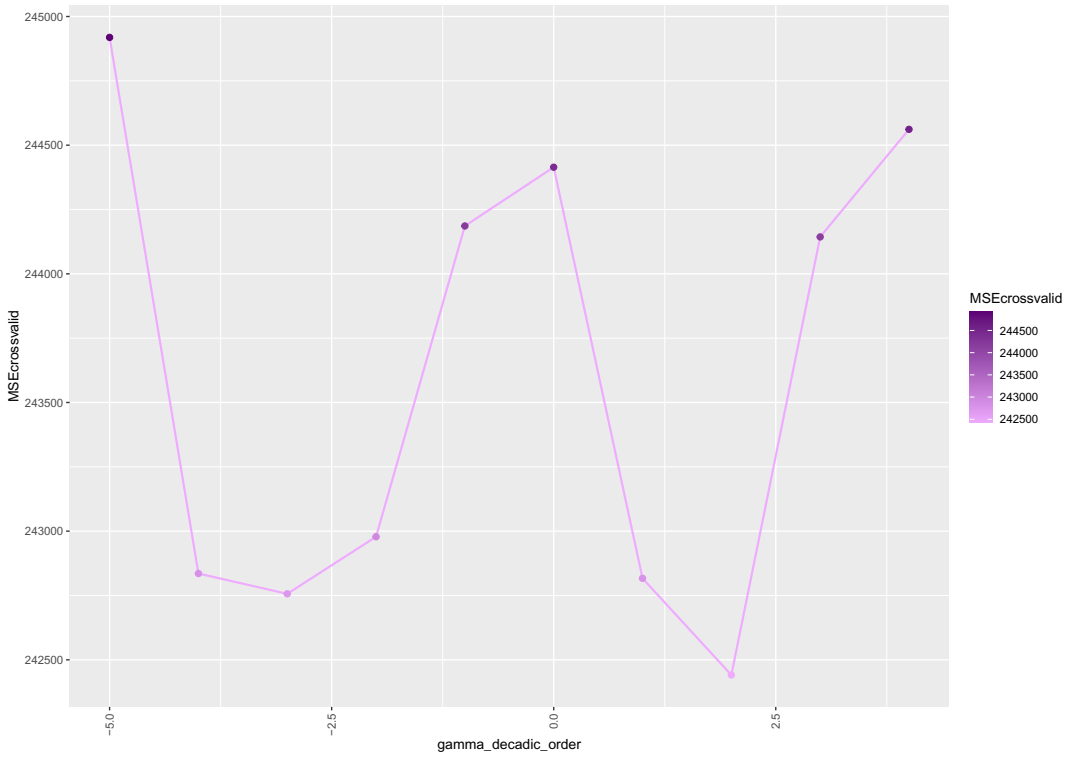
Obrázek 4.1: Výkonnost modelů při optimalizaci hyperparametrů `max_depth` a `min_child_weight`. Barevná škála reprezentuje monotónní, ale nelineární transformaci MSE, tak, aby byla barevná škála reprezentativní.

`gamma`

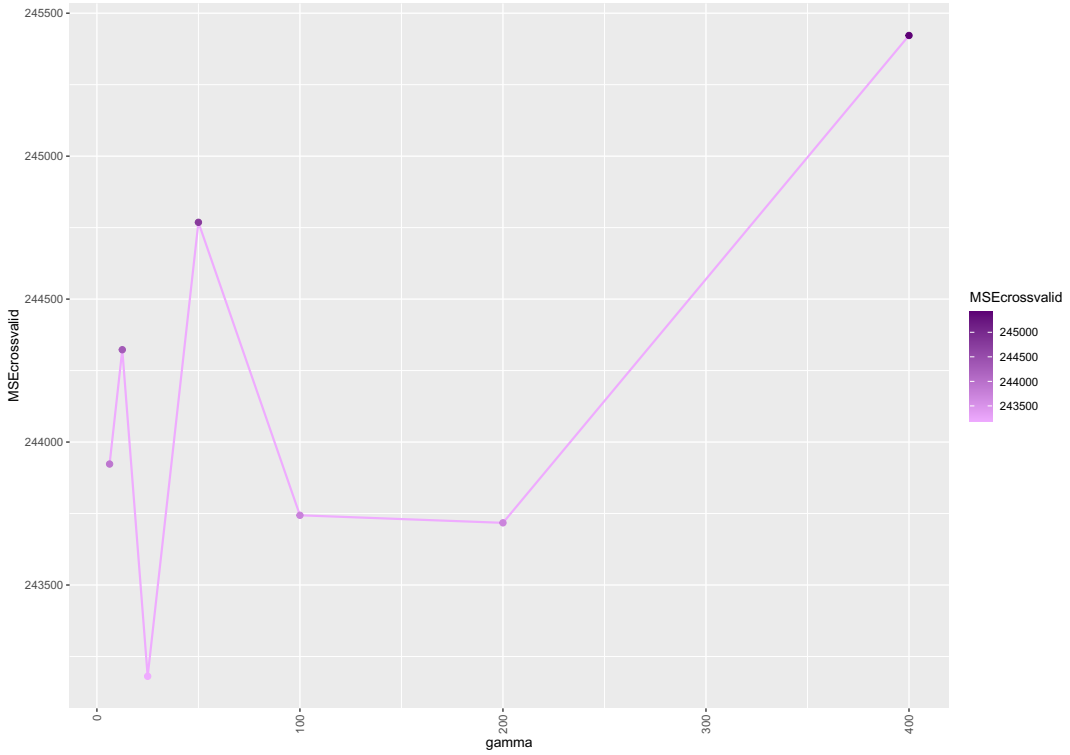
Parametr `gamma` optimalizujeme dvoukrokově. Na Obrázku 4.2 vidíme výkonnost při grid searchi přes hodnoty $\{0\} \cup \{10^i, i \in \{-5, -4, \ldots 4\}\}$. Vidíme dvě lokální minima v kladných a záporných hodnotách logaritmu, mezi kterými je lokální maximum. Vzhledem k tomu, že hodnoty $\gamma > 1$ bývají běžnější a navíc zde dochází k o něco lepším výsledkům v kontextu MSE, volíme pro další postup dekadický řád cca. 1-2, konkrétně budeme optimalizovat přímo γ přes hodnoty $\{6.25, 12.5, 25, 100, 200, 400\}$. Výsledky vykresluje Obrázek 4.3. Kromě hodnoty pro $\gamma = 25$, kterou považujeme za odlehlé pozorování, vychází minimum mezi 100 a 200, jako optimální γ volíme 150.

Robustnost modelu

Robustnost řešíme samplingem na prediktorech i pozorováních, konkrétně tedy parametry `subsample` a `colsample_by_tree`. Pro tyto parametry volíme grid search na množině $\{0.5, 0.55, 0.6, \ldots, 1\}^2$. Na Obrázku 4.4 vidíme preferenci větších hodnot. Nicméně funkce má evidentně mnoho lokálních minim. Vizuálně se

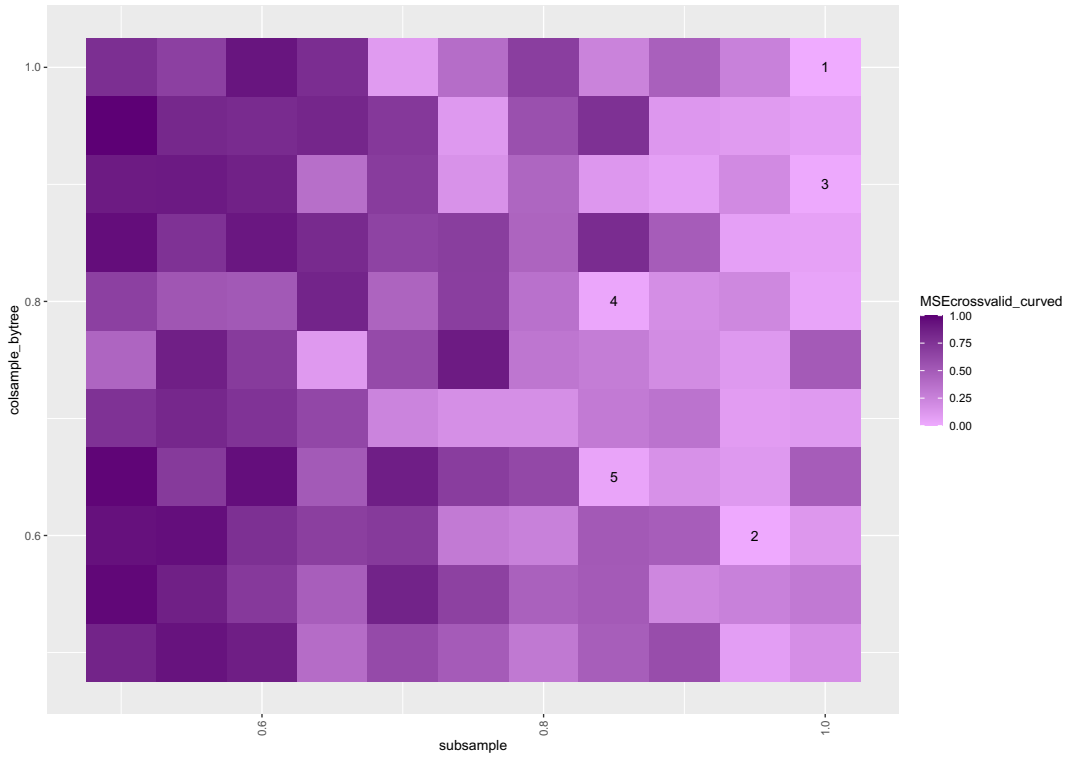


Obrázek 4.2: Výkonnost modelů při optimalizaci řádu hyperparametru `gamma`. Na ose x je $\log_{10}\gamma$. Graf neobsahuje hodnotu $\gamma = 0$, která ale nereprezentuje žádný z nejlepších 5 modelů. Barevná škála reprezentuje MSE.



Obrázek 4.3: Výkonnost modelů při optimalizaci hyperparametru `gamma`. Barevná škála reprezentuje MSE.

zdá, že v obou parametrech platí, že čím větší hodnota, tím lepší výsledek. Navíc nejlepší model vychází skutečně při hodnotách obou parametrů 1. Volíme tedy tyto, tj. nepoužíváme žádný sampling.



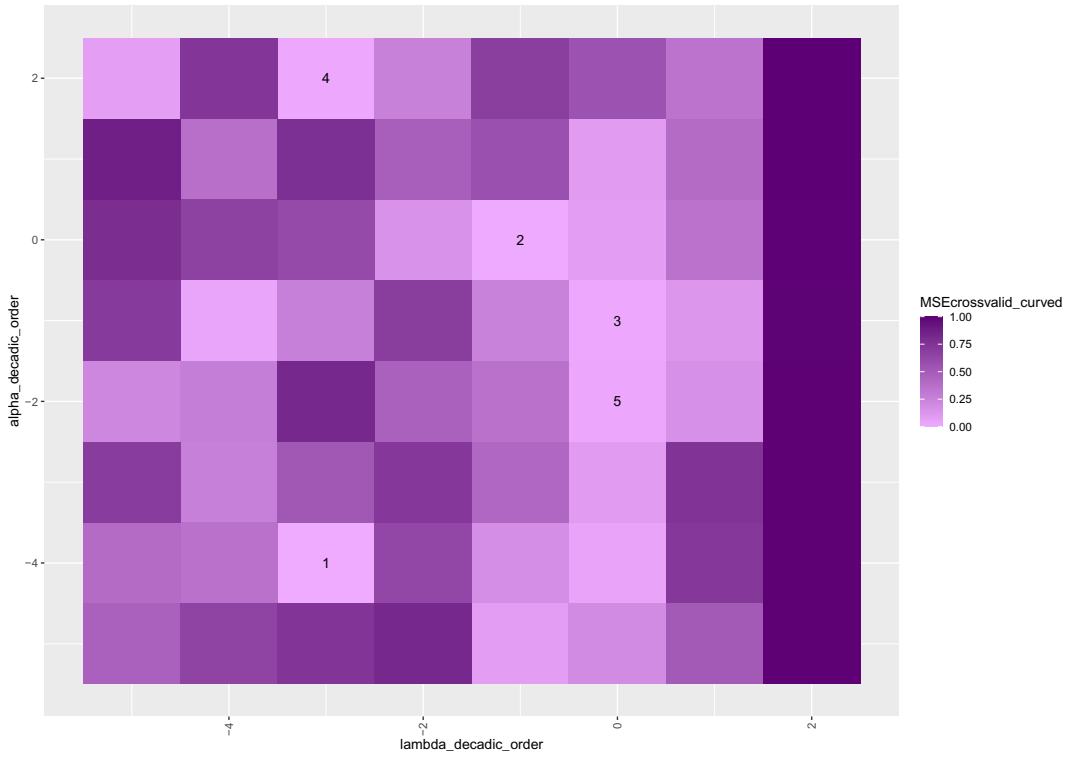
Obrázek 4.4: Výkonnost modelů při optimalizaci hyperparametrů `subsample` a `colsample_bytree`. Barevná škála reprezentuje monotónní, ale nelineární transformaci MSE, tak, aby byla barevná škála reprezentativní.

Regularizace

XGBoost podporuje dvojí regularizaci, L1 (člen $\alpha\|w\|_1$) a L2 (člen $\lambda\|w\|_2^2$). Jim odpovídají po řadě parametry `alpha` a `lambda`. Oba optimalizujeme dvoukrokově. Na Obrázku 4.5 vidíme výkonnost při grid searchi přes hodnoty $(\{0\} \cup \{10^i, i \in \{-5, -4, \dots 2\}\})^2$. Nejlepší model dle heatmapy považujeme za outlier a zaměříme se na oblast, kde jsou druhý a třetí nejlepší model. Pokračujeme podrobnějším zkoumáním této oblasti grid searchem na množině $\{0.1, 0.2, \dots, 1\}^2$. Pro finální model pak vybíráme na základě Obrázku 4.6 hodnoty $\alpha = 0.3$ a $\lambda = 1$.

`eta`

Jako poslední zjemníme (pravděpodobně) *learning rate*, optimalizujeme přes hodnoty $\{0.02, 0.04, 0.06, \dots, 0.3\}$. Výsledky vidíme na Obrázku 4.7. Hodnotu v $\eta = 0.04$ považujeme za outlier, po vyhlazení náhlých extrémů v $\eta = 0.16$ a $\eta = 0.22$, vybíráme optimální hodnotu 0.14.



Obrázek 4.5: Výkonnost modelů při optimalizaci řádů hyperparametrů `alpha` a `lambda`. Na osách je dekadický logaritmus těchto parametrů. Graf neobsahuje hodnoty $\lambda = 0$ a $\alpha = 0$, které ale nereprezentují žádný z nejlepších 5 modelů. Barevná škála reprezentuje monotónní, ale nelineární transformaci MSE, tak, aby byla barevná škála reprezentativní.

4.5.1 Finální model s optimalizovanými hyperparametry

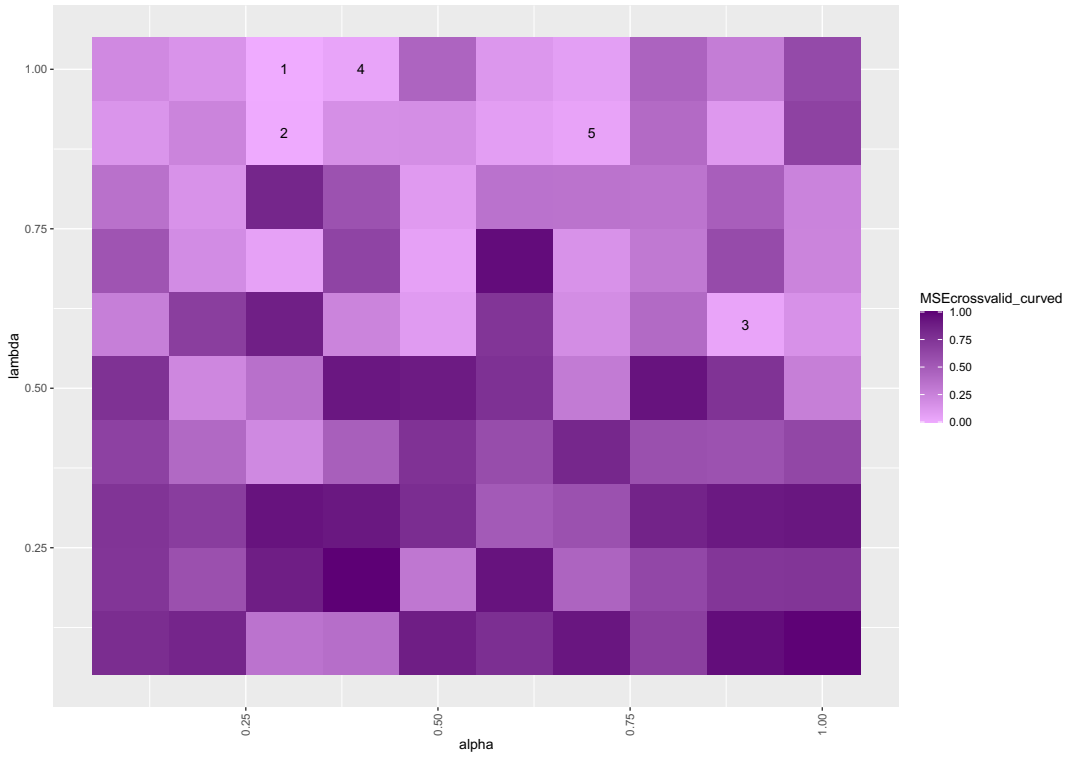
Finální model tedy trénujeme s parametry v Tabulce 4.4. I v tomto případě ponecháváme 20 % dat jako testovací množinu, na které počítáme zastavovací kritérium.

Parametr	**Kategorie**	**Optimalizovaná hodnota**
`max_depth`	Komplexita	5
`min_child_weight`	Komplexita	3
`gamma`	Komplexita	150
`subsample`	Robustnost	1
`colsample_bytree`	Robustnost	1
`eta`	Robustnost	0.14
`alpha`	Regularizace	0.3
`lambda`	Regularizace	1

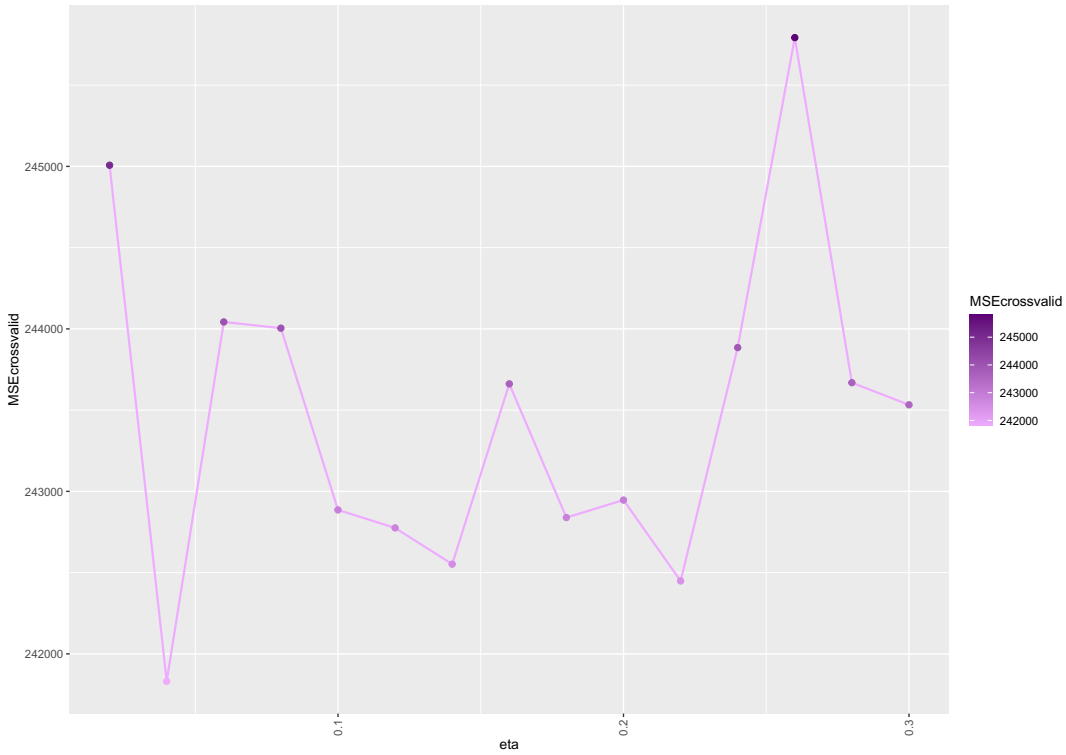
Tabulka 4.4: Tabulka optimalizovaných hyperparametrů.

4.6 Výkonnost modelu po letech

Zkontrolujeme, jak se vyvíjí výkonnost modelu v letech. Pro tyto účely použijeme dvě z relativních kritérií – R^2 a U. Vývoj vidíme na Obrázku 4.8 – je patrné, že

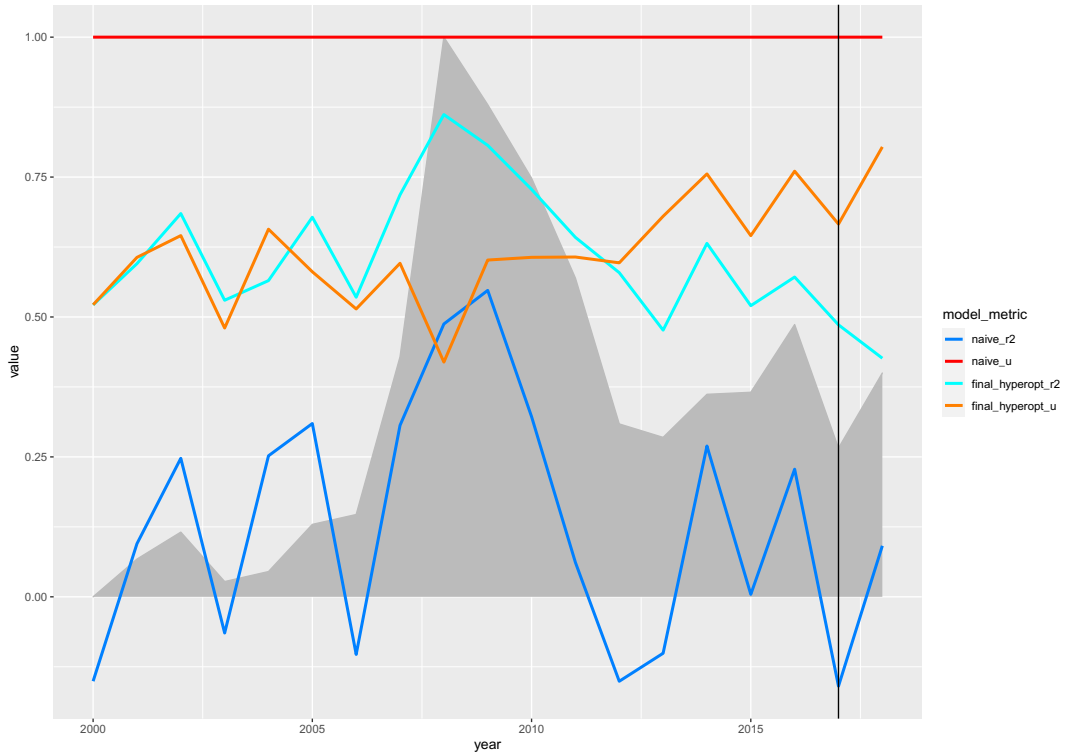


Obrázek 4.6: Výkonnost modelů při optimalizaci hyperparametrů `alpha` a `lambda`. Barevná škála reprezentuje monotónní, ale nelineární transformaci MSE, tak, aby byla barevná škála reprezentativní.



Obrázek 4.7: Výkonnost modelů při optimalizaci hyperparametru `eta`. Barevná škála reprezentuje MSE.

model je poměrně stabilní, obzvlášť měříme-li jeho výkonnost pomocí U. Dobře si poradil i s obdobím hospodářské krize, kdy došlo k výraznému nárůstu volatility. Na out-of-sample množině model performuje lehce hůř než na trénovacích datech, což je ale očekávané a bylo již diskutováno.



Obrázek 4.8: Výkonnost modelů po letech měřena pomocí R^2 a U. Šedým podbarvením reprezentujeme znormovanou směrodatnou odchylku objemu v daném roce – tj. míru volatility. Černá vertikála odděluje in-sample a out-of-sample množinu.

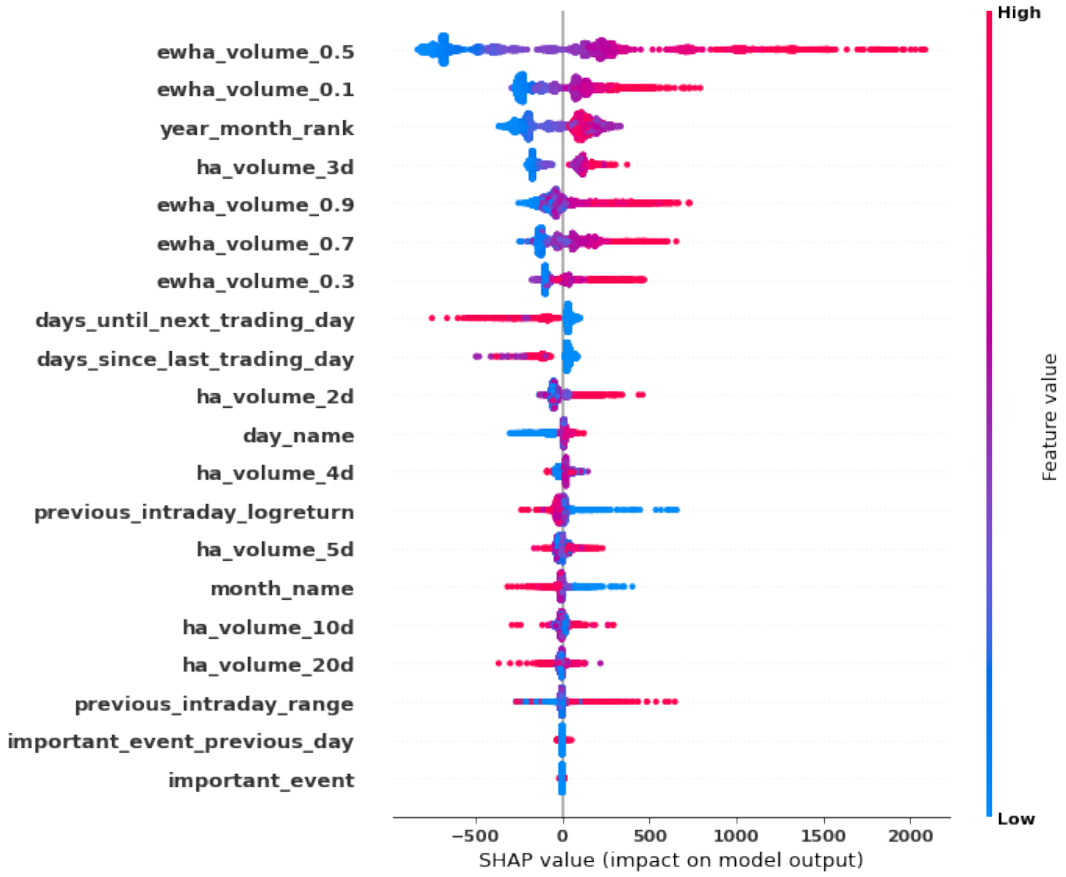
4.7 Důležitost prediktorů

Jak bylo popsáno teoreticky v sekci 1.3.8, využijeme knihovnu `shap` (***SH****apley* ***A****dditive ex****P****lanation*), díky níž uvidíme nejen relativní důležitost prediktorů, ale také vztah mezi hodnotami jednotlivých prediktorů a predikovanou hodnotou modelu.

Na Obrázku 4.9 vidíme, že nejdůležitějšími prediktory jsou zpožděná pozorování ve formě EWHA, dále ve formě HA, poté trendová složka (`year_month_rank`) a počet dní od posledního a do následujícího obchodního dne. Menší vliv pak mají prediktory předchozí denní logreturn, den v týdnu, měsíc a předchozí denní rozpětí cen. Nejméně významné prediktory jsou pak indikátory mimořádných událostí v daný nebo předchozí den.

Pro prediktory modelující zpožděnou odezvu (obzvláště ty, jejichž důležitost se ukazuje jako vysoká) můžeme graf interpretovat následujícím způsobem (vybrané prediktory):

- Vysoké hodnoty prediktorů `ewha_volume_*` a `ha_volume_*`, tedy zvýšený zobchodovaný objem v předchozích dnech (červené odstíny), obecně způ-



Obrázek 4.9: Relativní důležitost prediktorů v optimalizovaném modelu.

sobují zvýšenou hodnotu zobchodovaného objemu v předpovídaném dni (SHAP > 0).

- Zvýšený denní return způsobí pokles zobchodovaného objemu následující den.

4.8 Zhodnocení zvoleného přístupu

Použili jsme standardní způsob trénování modelu:

1. výběr proměnných na výchozích hyperparametrech, který typicky, stejně jako v našem případě, končí výběrem všech proměnných, neboť XGBoost obsahuje implicitní výběr proměnných;
2. optimalizace hyperparametrů.
3. Trochu atypicky jsme nakonec nevyužili všechna data pro trénování finálního modelu s odhadnutým fixním počtem iterací, ale zachovali jsme poměr 80:20 a využili zastavovací kritérium na testovací množině.

Z důvodu výpočetní náročnosti nebylo možné optimalizovat všechny hyperparametry najednou, zvolili jsme proto způsob postupné optimalizace na základě literatury. Nicméně tento suboptimální způsob mohl vést (obzvlášť na malých datech) k neoptimálním výsledkům. Nicméně, jak uvidíme v následujících sek-

cích, samotná optimalizace hyperaparametrů byl nutný a v zásadě úspěšný krok na cestě k lepšímu modelu.

5. Porovnání přístupů

5.1 Predikce

Budeme porovnávat 4 modely:

- naivní model – zpožděnou odezvu,
- finální ARIMAX model ze sekce 3.2.4,
- XGBoost bez optimalizovaných hyperparametrů ze sekce 4.4 a
- XGBoost s optimalizovanými hyperparametry ze sekce 4.5.1.

Výkonnosti modelů porovnáme na out-of-sample množině na kritériích MSE, MAE, R^2 a U. Out-of-sample množina má 249 pozorování od 1. 1. 2018 do 31. 12. 2018 včetně. Výsledky jsou v Tabulce 5.1. Vidíme, že kritéria nejsou zcela konzistentní v uspořádání modelů dle kvality. S výjimkou kritéria MAE vidíme signifikantní zlepšení zejména při přechodu od naivního modelu k modelu ARIMAX nebo neoptimalizovanému modelu XGBooost a při přechodu od neoptimalizovaného k optimalizovanému modelu XGBooost. Zajímavé je také, že modely ARIMAX a neoptimalizovaný XGBoost dávají podobně kvalitní predikce – na základě U jsou stejně kvalitní, na základě R^2 a MSE je mírně kvalitnější model ARIMAX, na základě MAE zase XGBoost.

Model	**MSE**	**MAE**	R^2	U
`naivní`	432 990.4	393.4	0.091	1.000
`ARIMAX`	370 771.5	425.3	0.222	0.951
`XGBoost neoptimalizovaný`	384 728.0	398.6	0.192	0.951
`XGBoost optimalizovaný`	260 985.8	336.6	0.452	0.790

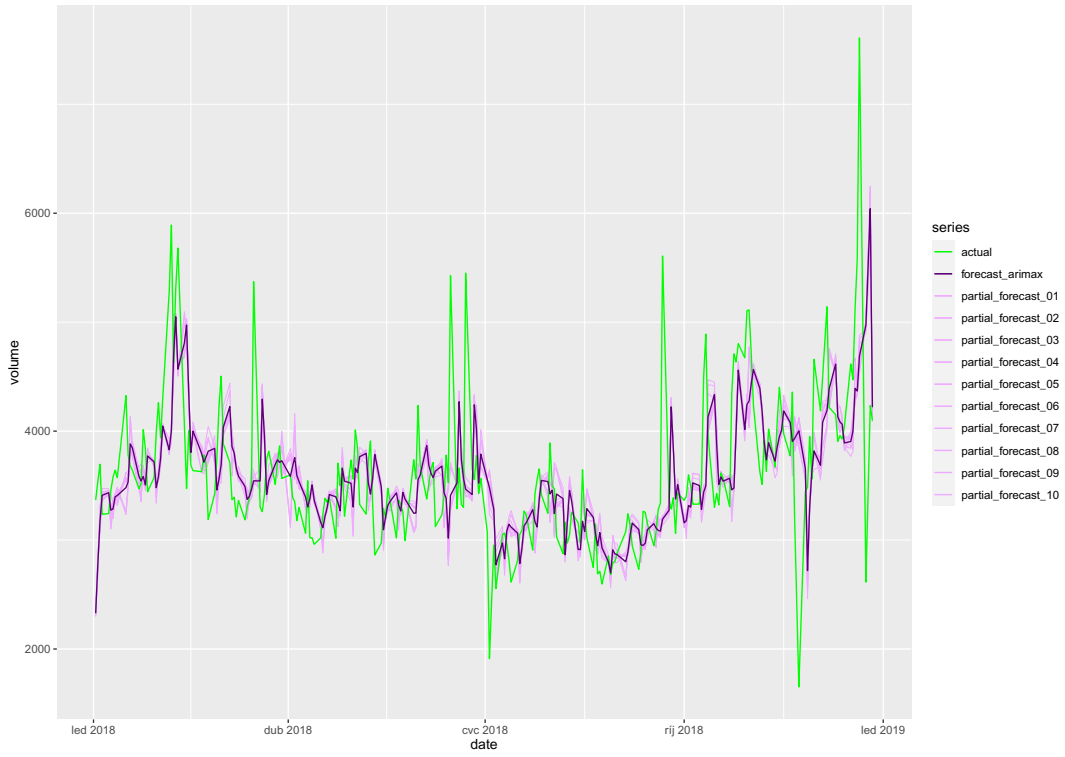
Tabulka 5.1: Tabulka výkonnosti modelů hyperparametrů.

Na Obrázcích 5.1 a 5.2 vidíme graf predikcí a skutečných hodnot na out-of-sample množině. Od pohledu nevidíme mezi grafy žádný tendenční rozdíl, například lepší schopnost jednoho modelu predikovat odlehlá pozorování, modelovat trend nebo sezónnost. Rozdílné hodnoty ve sledovaných kritériích tak pravděpodobně budou způsobeny kombinací mírně rozdílných výsledků ve více těchto zmíněných oblastech.

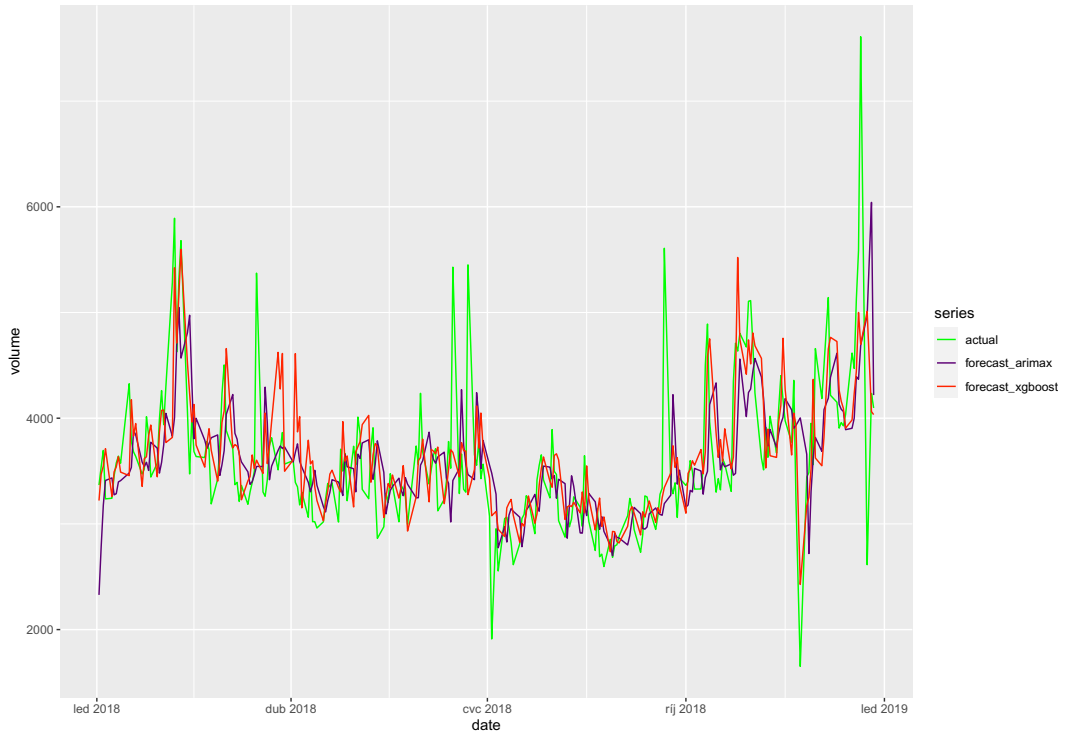
5.2 Porovnání přístupů

Modely ARIMAX a XGBoost se v řadě významných faktorů liší:

- ARIMAX je lineární model. Jakákoliv nelinearita musí být modelována pomocí dalších transformovaných prediktorů, které se ale musí apriori zvolit, a nemusí být vždy jednoduché nebo možné optimální transformace najít. Naproti tomu XGBoost lineární model není.



Obrázek 5.1: Predikce pomocí modelu ARIMAX na out-of-sample množině včetně všech 10 submodelů, které dohromady tvoří finální model.



Obrázek 5.2: Predikce pomocí modelu XGBoost s optimalizovanými parametry na out-of-sample množině. Pro srovnání vykreslen i model ARIMAX.

- Předchozí bod umožňuje velmi přímočarou interpretaci výsledků modelů

ARIMAX, a to jak regresní, tak autoregresní složky. K interpretaci výsledků XGBoostu jsou třeba speciální nepřímé metody, například Shapleyho hodnoty.

- XGBoost při trénování provádí implicitní výběr proměnných – vybírá ke splitu nejvhodnější proměnnou a bod splitu a v rámci nastavených parametrů „dá příležitost“ dalším prediktorům, jen pokud jejich zahrnutí zlepšuje model. ARIMAX vyžaduje apriorní výběr proměnných a následnou retrospektivní analýzu jejich přínosnosti. Při silně korelovaných prediktorech nebo při jejich vysokém počtu může být tato analýza problematická a je třeba přistupovat ke složitějším přístupům redukce dimenze prostoru prediktorů, např. analýzou hlavních komponent.
- XGBoost jako knihovna si poradí s vysokým počtem pozorování, může běžet paralelně, na GPU a s daty i mimo RAM. Žádná knihovna, která by něco podobného umožňovala s modelem ARIMAX, v současnosti neexistuje.
- XGBoost byl vydán jako stabilní verze teprve v roce 2019. I nadále jsou hlášeny chyby, ovšem již jen poměrně zřídka.
- Na druhou stranu je XGBoost tzv. *end-to-end system*. To znamená, že celý výpočet je součást knihovny a není závislý na knihovnách třetích stran. Naproti tomu například použitá knihovna `forecast` v jazyce R, konkrétně metoda `ARIMAX` využívá řadu dalších knihoven, využívá vnitřní R solver pro optimalizaci věrohodnostní funkce, což zvyšuje riziko technické nebo programátorské chyby, a navíc výrazně ztěžuje proces oprav.

Specifická věc k porovnání je časová výpočetní náročnost obou přístupů.

Výpočetní náročnost

Na trénování všech modelů byl využit virtuální počítač v Microsoft Azure, konkrétně D2s v4 s 8 GB paměti RAM, 2 vCPU[1] procesoru Intel® Xeon® Platinum 8272CL s frekvencí 2.5 GHz, při hyperthreadingu 3.4 GHz.

- Při trénování ARIMAX modelu bylo postupně natrénováno 18 836 modelů, celkový výpočetní čas byl cca 16 dní a 4 hodiny čistého času.
- Při trénování XGBoost modelu bylo postupně natrénováno 34 574 modelů, celkový výpočetní čas byl cca 6 hodin, tedy v přepočtu na jeden model řádově stokrát méně.

Zde je na praktickém příkladu dobře vidět jedna z výhod implementace XGBoostu – knihovna je psaná kompletně v C++, v němž jsou jsou výpočty násobně rychlejší v porovnání s vysokoúrovňovým R, i když řada knihoven pro R je psaná také v C++.

Nicméně ještě větší výhoda XGBoostu (jeho škálovatelnost, distribuovatelné trénování) se na našem vzorku nemohla projevit, neboť jsme ho pouštěli na jednom počítači a nikoliv na clusteru. Další výhody jako např. optimalizace práce s cache, operační pamětí a hard diskem jsme také nevyužili, neboť náš datový vzorek nebyl

[1] virtuální jádra procesoru

dost velký na to, aby přístup k datům a jejich řazení signifikantně zpomalovaly výpočet.

5.3 Celkové zhodnocení

Závěrem lze tedy říct následující:

- Úlohu má jistě cenu modelovat, neboť naivní predikce vychází ze všech zkoumaných přístupů jako výrazně nejhorší. Tím se úloha liší od většiny pokusů o predikci ceny akcií.
- Model ARIMAX je vhodným instrumentem pro predikci objemu. Umožňuje na rozdíl od jednoduššího modelu ARIMA zahrnout do časové řady i exogenní proměnné, z nichž část se ukázala být signifikantními prediktory.
- XGBoost model přes své nesporné výhody (zejména implicitní výběr relevantních proměnných a schopnost postihnout nelineární vliv prediktorů) by nebyl lepším modelem než ARIMAX, kdybychom neudělali následnou optimalizaci hyperparametrů.
- Optimalizovaný model XGBoost je pro tuto úlohu výrazně lepším přístupem, na základě všech uvažovaných kritérií dosahuje nejlepších výsledků.
- Trénování modelu XGBoost je řádově rychlejší.

6. Hybridní model

V současnosti jsou velmi populární tzv. hybridní modely, pro predikci časových řad se používají např. v [26] a [28]. Hybridní model funguje tak, že se natrénuje m modelů typicky různých rodin, v každé rodině se vyvine optimální model a predikce teoretického finálního modelu (někdy též metamodelu) je pak nějakou konvexní kombinací jednotlivých modelů.

Pro určení vah w_j jednotlivých modelů j je několik přístupů:

- každý model vstupuje do metamodelu se stejnou váhou $w_j = 1/m$, $\forall j$;
- každý model vstupuje do metamodelu s váhou odpovídající jeho pořadí mezi modely dle vybraného kritéria, $w_j = 2(m + 1 - \mathrm{ord}(j))/m(m + 1)$, $\forall j$, kde $\mathrm{ord}(j)$ je funkce, která vrací pořadí modelu j mezi m modely, tedy 1 pro nejlepší model, m pro nejhorší model;
- každý model vstupuje do metamodelu s váhou odpovídající jeho kvalitě měřené dle vybraného kritéria, např. pro R^2 je $w_j = R^2(j)/\sum_{i=1}^{m} R^2(m)$, $\forall j$;
- váhy se odhadnou expertně;
- využití optimalizačních metod.

První zmíněný způsob (rovnoměrné váhy) jsme již využili v sekci 3.2.4, kde jsme vybrali 10 nejslibnějších ARIMAX modelů a za finální model označili průměr jejich predikcí.

Naším cílem pro tuto práci bude najít optimální model dle MSE. K tomu odvodíme v následující sekci příslušnou teorii.

6.1 Teorie optimálního hybridního modelu

V celé sekci označujeme vektor skutečných hodnot jako $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T$, vektory predikovaných hodnot pak jako $\hat{\mathbf{y}}^{(j)}$, $j = 1, 2, \ldots, m$ pro každý model j.

Začneme definicí optimálního hybridního modelu:

Definice 6.1. ***Optimálním hybridním modelem*** $HM_{\mathbf{w}^*}$ *nazveme (libovolný) model, který predikuje* $\hat{\mathbf{y}}^{(HM_{\mathbf{w}^*})} := \sum_{j=1}^{m} w_j \hat{\mathbf{y}}^{(j)}$ *(vektory se sčítají po prvcích), tak, že*

$$\mathbf{w}^* = (w_1, w_2, \ldots, w_m)^T := \underset{\substack{w_i \geq 0,\ \forall i \in \{1,2,\ldots,m\}, \\ \sum w_i = 1}}{\arg\min} loss(\hat{\mathbf{y}}^{(HM_{\mathbf{w}^*})}, \mathbf{y}), \tag{6.1}$$

kde loss je nějaká ztrátová funkce.

Tato optimalizační úloha může být pro některé ztrátové funkce výpočetně velmi náročná. V takovém případě se pro malý počet modelů dá využít například grid search, pro větší počet potom některé z moderních optimalizačních metod, například genetické algoritmy.

Nicméně pro některé vhodné ztrátové funkce jde tato teorie výrazně více rozvinout. My v této práci odvodíme teorii pro MSE, které v průběhu celé práce používáme jako jedno z hlavních kritérií.

6.2 Optimální hybridní model dle MSE

Nejprve připomeneme definici MSE:

$$\mathrm{MSE}(\widehat{\mathbf{y}}) := \frac{1}{n}\sum_{i=1}^{n}(\widehat{y}_i - y_i)^2,$$

V následujícím postupu budeme využívat zobecnění MSE pro dvojici vektorů, které nazveme MCME[1]:

Definice 6.2. *Pro vektory* $\widehat{\mathbf{y}}^{(1)}, \widehat{\mathbf{y}}^{(2)} \in \mathbb{R}^n$ *definujeme*

$$\mathrm{MCME}(\widehat{\mathbf{y}}^{(1)}, \widehat{\mathbf{y}}^{(2)}) := \frac{1}{n}\sum_{i=1}^{n}(\widehat{y}_i^{(1)} - y_i)(\widehat{y}_i^{(2)} - y_i).$$

Základní vlastnosti shrnuje následující triviální pozorování:

Pozorování 6.1. *Pro libovolné vektory* $\widehat{\mathbf{y}}^{(1)}, \widehat{\mathbf{y}}^{(2)} \in \mathbb{R}^n$ *platí*

1. $\mathrm{MCME}(\widehat{\mathbf{y}}^{(1)}, \widehat{\mathbf{y}}^{(1)}) = \mathrm{MSE}(\widehat{\mathbf{y}}^{(1)})$,
2. $\mathrm{MCME}(\widehat{\mathbf{y}}^{(1)}, \widehat{\mathbf{y}}^{(2)}) = \mathrm{MCME}(\widehat{\mathbf{y}}^{(2)}, \widehat{\mathbf{y}}^{(1)})$,
3. $\mathrm{MCME}(\widehat{\mathbf{y}}^{(1)}, \mathbf{y}) = 0$.

Dále dokážeme jedno technické tvrzení.

Tvrzení 6.2. *Pro libovolné vektory* $\widehat{\mathbf{y}}^{(1)}, \widehat{\mathbf{y}}^{(2)}, \widehat{\mathbf{y}}^{(3)} \in \mathbb{R}^n$ *platí*

$$\begin{aligned}&\mathrm{MSE}(\widehat{\mathbf{y}}^{(1)}) + \mathrm{MCME}(\widehat{\mathbf{y}}^{(2)}, \widehat{\mathbf{y}}^{(3)}) - \mathrm{MCME}(\widehat{\mathbf{y}}^{(1)}, \widehat{\mathbf{y}}^{(2)}) - \mathrm{MCME}(\widehat{\mathbf{y}}^{(1)}, \widehat{\mathbf{y}}^{(3)}) = \\ &\quad = \frac{1}{n}\sum_{i=1}^{n}(\widehat{y}_i^{(2)} - \widehat{y}_i^{(1)})(\widehat{y}_i^{(3)} - \widehat{y}_i^{(1)}).\end{aligned}$$

Důkaz.

$$\begin{aligned}&\mathrm{MSE}(\widehat{\mathbf{y}}^{(1)}) + \mathrm{MCME}(\widehat{\mathbf{y}}^{(2)}, \widehat{\mathbf{y}}^{(3)}) - \mathrm{MCME}(\widehat{\mathbf{y}}^{(1)}, \widehat{\mathbf{y}}^{(2)}) - \mathrm{MCME}(\widehat{\mathbf{y}}^{(1)}, \widehat{\mathbf{y}}^{(3)}) = \\ &= \frac{1}{n}\sum_{i=1}^{n}\Big((\widehat{y}_i^{(1)} - y_i)(\widehat{y}_i^{(1)} - y_i) + (\widehat{y}_i^{(2)} - y_i)(\widehat{y}_i^{(3)} - y_i) \\ &\qquad - (\widehat{y}_i^{(1)} - y_i)(\widehat{y}_i^{(2)} - y_i) - (\widehat{y}_i^{(1)} - y_i)(\widehat{y}_i^{(3)} - y_i)\Big) = \\ &= \frac{1}{n}\sum_{i=1}^{n}\Big((\widehat{y}_i^{(1)} - y_i)(\widehat{y}_i^{(1)} - y_i - \widehat{y}_i^{(2)} + y_i) \\ &\qquad + (\widehat{y}_i^{(3)} - y_i)(\widehat{y}_i^{(2)} - y_i - \widehat{y}_i^{(1)} + y_i)\Big) = \\ &= \frac{1}{n}\sum_{i=1}^{n}\Big((\widehat{y}_i^{(1)} - y_i)(\widehat{y}_i^{(1)} - \widehat{y}_i^{(2)}) + (\widehat{y}_i^{(3)} - y_i)(\widehat{y}_i^{(1)} - \widehat{y}_i^{(2)})\Big) = \\ &= \frac{1}{n}\sum_{i=1}^{n}(\widehat{y}_i^{(1)} - \widehat{y}_i^{(2)})(\widehat{y}_i^{(1)} - y_i - \widehat{y}_i^{(3)} + y_i) = \frac{1}{n}\sum_{i=1}^{n}(\widehat{y}_i^{(2)} - \widehat{y}_i^{(1)})(\widehat{y}_i^{(3)} - \widehat{y}_i^{(1)}). \square\end{aligned}$$

[1] z anglického ***M**ean **C**ross-**M**ultiplicative **E**rror*

Před samotným odvozením metody dokážeme následující jednoduché aritmetické lemma:

Lemma 6.3. *Pro libovolná $x_i \in \mathbb{R}, i \in \{1,2,\ldots,n\}$ platí*

$$\left(\sum_{i=1}^{n} x_i\right)^2 = \sum_{i=1}^{n} x_i^2 + \sum_{\substack{j,k \in \{1,2,\ldots,n\},\\ j<k}} 2 x_j x_k.$$

Důkaz. Lemma se snadno dokáže indukcí podle n. Pro $n = 1$ zjevně platí. Předpokládejme platnost pro $n-1$ a dokážeme platnost pro n:

$$\begin{aligned}
\left(\sum_{i=1}^{n} x_i\right)^2 &= \left(\sum_{i=1}^{n-1} x_i + x_n\right)^2 = \\
&\overset{\text{roznásobení}}{=} \left(\sum_{i=1}^{n-1} x_i\right)^2 + x_n^2 + 2x_n \sum_{i=1}^{n-1} x_i = \\
&\overset{\text{indukce}}{=} \sum_{i=1}^{n-1} x_i^2 + \sum_{\substack{j,k \in \{1,2,\ldots,n-1\},\\ j<k}} 2 x_j x_k + x_n^2 + 2x_n \sum_{i=1}^{n-1} x_i = \\
&= \sum_{i=1}^{n} x_i^2 + \sum_{\substack{j,k \in \{1,2,\ldots,n\},\\ j<k}} 2 x_j x_k.
\end{aligned}$$

□

Protože jedna ze dvou podmínek na vektor vah $(w_1, w_2, \ldots, w_m)^T$ je $\sum_{i=1}^{m} w_i = 1$, definujeme $w_m := 1 - \sum_{i=1}^{m-1} w_i$ a neznámých vah tak uvažujeme $m-1$. Označíme $\mathbf{w} := (w_1, w_2, \ldots, w_{m-1})^T$.

Nyní již vyjádříme MSE hybridního modelu:

$$
\begin{aligned}
&\mathrm{MSE}(\widehat{\mathbf{y}}^{(HM_{\mathbf{w}})}) = \\
&= \mathrm{MSE}\left(\sum_{j=1}^{m} w_j \widehat{\mathbf{y}}^{(j)}\right) = \\
&= \mathrm{MSE}\left(\sum_{j=1}^{m-1} w_j \widehat{\mathbf{y}}^{(j)} + \left(1 - \sum_{z=1}^{m-1} w_z\right) \widehat{\mathbf{y}}^{(m)}\right) = \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{m-1} w_j \widehat{y}_i^{(j)} + \left(1 - \sum_{z=1}^{m-1} w_z\right) \widehat{y}_i^{(m)} - y_i\right)^2 = \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{m-1} w_j \widehat{y}_i^{(j)} + \left(1 - \sum_{z=1}^{m-1} w_z\right) \widehat{y}_i^{(m)} - y_i + \sum_{j=1}^{m-1} w_j y_i - \sum_{j=1}^{m-1} w_j y_i\right)^2 = \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{m-1} w_j \widehat{y}_i^{(j)} + \left(1 - \sum_{z=1}^{m-1} w_z\right) \widehat{y}_i^{(m)} - y_i + \sum_{j=1}^{m-1} w_j y_i - \sum_{j=1}^{m-1} w_j y_i\right)^2 = \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{m-1} w_j (\widehat{y}_i^{(j)} - y_i) + \left(1 - \sum_{z=1}^{m-1} w_z\right) (\widehat{y}_i^{(m)} - y_i)\right)^2 = \\
&\overset{\text{Lemma 6.3}}{=} \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{m-1} w_j^2 (\widehat{y}_i^{(j)} - y_i)^2 + \left(1 - \sum_{z=1}^{m-1} w_z\right)^2 (\widehat{y}_i^{(m)} - y_i)^2\right. \\
&\qquad\qquad + 2\sum_{\substack{k,l\in\{1,2,\ldots,m-1\}\\ k<l}} w_k w_l (\widehat{y}_i^{(k)} - y_i)(\widehat{y}_i^{(l)} - y_i) \\
&\qquad\qquad \left. + 2\sum_{j=1}^{m-1} w_j \left(1 - \sum_{z=1}^{m-1} w_z\right) (\widehat{y}_i^{(j)} - y_i)(\widehat{y}_i^{(m)} - y_i)\right).
\end{aligned}
\tag{6.2}
$$

Sumy přes i, j a k a l jsou na sobě nezávislé, takže na jejich pořadí nezáleží. Tedy

$$
\begin{aligned}
&\mathrm{MSE}(\widehat{\mathbf{y}}^{(HM_{\mathbf{w}})}) = \\
&= \sum_{j=1}^{m-1} w_j^2 \frac{1}{n}\sum_{i=1}^{n}(\widehat{y}_i^{(j)} - y_i)^2 + \left(1 - \sum_{z=1}^{m-1} w_z\right)^2 \frac{1}{n}\sum_{i=1}^{n}(\widehat{y}_i^{(m)} - y_i)^2 \\
&\quad + 2\sum_{\substack{k,l\in\{1,2,\ldots,m-1\}\\ k<l}} w_k w_l \frac{1}{n}\sum_{i=1}^{n}(\widehat{y}_i^{(k)} - y_i)(\widehat{y}_i^{(l)} - y_i) \\
&\quad + 2\sum_{j=1}^{m-1} w_j \left(1 - \sum_{z=1}^{m-1} w_z\right) \frac{1}{n}\sum_{i=1}^{n}(\widehat{y}_i^{(j)} - y_i)(\widehat{y}_i^{(m)} - y_i) \\
&= \sum_{j=1}^{m-1} w_j^2 \,\mathrm{MSE}(\widehat{\mathbf{y}}^{(j)}) + \left(1 - \sum_{z=1}^{m-1} w_z\right)^2 \mathrm{MSE}(\widehat{\mathbf{y}}^{(m)}) \\
&\quad + 2\sum_{\substack{k,l\in\{1,2,\ldots,m-1\}\\ k<l}} w_k w_l \,\mathrm{MCME}(\widehat{\mathbf{y}}^{(k)}, \widehat{\mathbf{y}}^{(l)}) \\
&\quad + 2\sum_{j=1}^{m-1} w_j \left(1 - \sum_{z=1}^{m-1} w_z\right) \mathrm{MCME}(\widehat{\mathbf{y}}^{(j)}, \widehat{\mathbf{y}}^{(m)}).
\end{aligned}
\tag{6.3}
$$

Nyní postupně proderivujeme výraz (6.3) podle všech proměnných, tedy podle $w_p = 1, 2, \ldots, m-1$ a položíme rovno 0:

$$\begin{aligned}\frac{\partial \operatorname{MSE}(\widehat{\mathbf{y}}^{(HM_{\mathbf{w}})})}{\partial w_p}(\mathbf{w}) = {} & 2w_p \operatorname{MSE}(\widehat{\mathbf{y}}^{(p)}) - 2\left(1 - \sum_{z=1}^{m-1} w_z\right) \operatorname{MSE}(\widehat{\mathbf{y}}^{(m)}) \\ & - 2 \sum_{\substack{j \in \{1,2,\ldots,m-1\} \\ j \neq p}} w_j \operatorname{MCME}(\widehat{\mathbf{y}}^{(j)}, \widehat{\mathbf{y}}^{(m)}) \\ & + 2 \operatorname{MCME}(\widehat{\mathbf{y}}^{(p)}, \widehat{\mathbf{y}}^{(m)}) \\ & - 2 \sum_{\substack{j \in \{1,2,\ldots,m-1\} \\ j \neq p}} w_j \operatorname{MCME}(\widehat{\mathbf{y}}^{(p)}, \widehat{\mathbf{y}}^{(m)}) \\ & - 4w_p \operatorname{MCME}(\widehat{\mathbf{y}}^{(p)}, \widehat{\mathbf{y}}^{(m)}) \\ & + 2 \sum_{\substack{j \in \{1,2,\ldots,m-1\} \\ j \neq p}} w_j \operatorname{MCME}(\widehat{\mathbf{y}}^{(j)}, \widehat{\mathbf{y}}^{(p)}) \\ \stackrel{!}{=} {} & 0 \end{aligned} \tag{6.4}$$

Těchto $m-1$ rovností z výrazu (6.4) zapíšeme maticově:

$$(\mathbb{C} + \mathbb{A})\mathbf{w} = \mathbf{c} - \mathbf{b}, \tag{6.5}$$

kde $\mathbb{C}$ a $\mathbf{c}$ jsou matice a vektor tvořené pouze konstantou $\operatorname{MSE}(\widehat{\mathbf{y}}^{(m)})$, tedy

$$\mathbb{C} = \begin{pmatrix} \operatorname{MSE}(\widehat{\mathbf{y}}^{(m)}) & \cdots & \operatorname{MSE}(\widehat{\mathbf{y}}^{(m)}) & \cdots & \operatorname{MSE}(\widehat{\mathbf{y}}^{(m)}) \\ \vdots & \ddots & \vdots & & \vdots \\ \operatorname{MSE}(\widehat{\mathbf{y}}^{(m)}) & \cdots & \operatorname{MSE}(\widehat{\mathbf{y}}^{(m)}) & \cdots & \operatorname{MSE}(\widehat{\mathbf{y}}^{(m)}) \\ \vdots & & \vdots & \ddots & \vdots \\ \operatorname{MSE}(\widehat{\mathbf{y}}^{(m)}) & \cdots & \operatorname{MSE}(\widehat{\mathbf{y}}^{(m)}) & \cdots & \operatorname{MSE}(\widehat{\mathbf{y}}^{(m)}) \end{pmatrix},$$

$$\mathbf{c} = \begin{pmatrix} \operatorname{MSE}(\widehat{\mathbf{y}}^{(m)}) \\ \vdots \\ \operatorname{MSE}(\widehat{\mathbf{y}}^{(m)}) \\ \vdots \\ \operatorname{MSE}(\widehat{\mathbf{y}}^{(m)}) \end{pmatrix}, \text{dále pak}$$

$$\mathbf{b} = \begin{pmatrix} \operatorname{MSE}(\widehat{\mathbf{y}}^{(1)}) \\ \vdots \\ \operatorname{MSE}(\widehat{\mathbf{y}}^{(i)}) \\ \vdots \\ \operatorname{MSE}(\widehat{\mathbf{y}}^{(m-1)}) \end{pmatrix} \cdots i \quad \text{a}$$

$$
\mathbb{A} = \begin{pmatrix}
\overset{\cdots 1}{\vdots} & & \overset{\cdots i}{\vdots} & & \overset{\cdots j}{\vdots} & & \overset{\cdots m-1}{\vdots} & \\
\mathrm{MSE}(\widehat{\mathbf{y}}^{(1)}) & & \mathrm{MCME}(\widehat{\mathbf{y}}^{(1)},\widehat{\mathbf{y}}^{(i)}) & & \mathrm{MCME}(\widehat{\mathbf{y}}^{(1)},\widehat{\mathbf{y}}^{(j)}) & & \mathrm{MCME}(\widehat{\mathbf{y}}^{(1)},\widehat{\mathbf{y}}^{(m-1)}) & \\
-2\,\mathrm{MCME}(\widehat{\mathbf{y}}^{(1)},\widehat{\mathbf{y}}^{(m)}) & \cdots & -\mathrm{MCME}(\widehat{\mathbf{y}}^{(1)},\widehat{\mathbf{y}}^{(m)}) & \cdots & -\mathrm{MCME}(\widehat{\mathbf{y}}^{(1)},\widehat{\mathbf{y}}^{(m)}) & \cdots & -\mathrm{MCME}(\widehat{\mathbf{y}}^{(1)},\widehat{\mathbf{y}}^{(m)}) & \cdots 1 \\
 & & -\mathrm{MCME}(\widehat{\mathbf{y}}^{(i)},\widehat{\mathbf{y}}^{(m)}) & & -\mathrm{MCME}(\widehat{\mathbf{y}}^{(j)},\widehat{\mathbf{y}}^{(m)}) & & -\mathrm{MCME}(\widehat{\mathbf{y}}^{(m-1)},\widehat{\mathbf{y}}^{(m)}) & \\
\vdots & \ddots & \vdots & & \vdots & & \vdots & \\
\mathrm{MCME}(\widehat{\mathbf{y}}^{(i)},\widehat{\mathbf{y}}^{(1)}) & & \mathrm{MSE}(\widehat{\mathbf{y}}^{(i)}) & & \mathrm{MCME}(\widehat{\mathbf{y}}^{(i)},\widehat{\mathbf{y}}^{(j)}) & & \mathrm{MCME}(\widehat{\mathbf{y}}^{(i)},\widehat{\mathbf{y}}^{(m-1)}) & \\
-\mathrm{MCME}(\widehat{\mathbf{y}}^{(i)},\widehat{\mathbf{y}}^{(m)}) & \cdots & -2\,\mathrm{MCME}(\widehat{\mathbf{y}}^{(i)},\widehat{\mathbf{y}}^{(m)}) & \cdots & -\mathrm{MCME}(\widehat{\mathbf{y}}^{(i)},\widehat{\mathbf{y}}^{(m)}) & \cdots & -\mathrm{MCME}(\widehat{\mathbf{y}}^{(i)},\widehat{\mathbf{y}}^{(m)}) & \cdots i \\
-\mathrm{MCME}(\widehat{\mathbf{y}}^{(1)},\widehat{\mathbf{y}}^{(m)}) & & & & -\mathrm{MCME}(\widehat{\mathbf{y}}^{(j)},\widehat{\mathbf{y}}^{(m)}) & & -\mathrm{MCME}(\widehat{\mathbf{y}}^{(m-1)},\widehat{\mathbf{y}}^{(m)}) & \\
\vdots & & \vdots & \ddots & \vdots & & \vdots & \\
\mathrm{MCME}(\widehat{\mathbf{y}}^{(j)},\widehat{\mathbf{y}}^{(1)}) & & \mathrm{MCME}(\widehat{\mathbf{y}}^{(j)},\widehat{\mathbf{y}}^{(i)}) & & \mathrm{MSE}(\widehat{\mathbf{y}}^{(j)}) & & \mathrm{MCME}(\widehat{\mathbf{y}}^{(j)},\widehat{\mathbf{y}}^{(m-1)}) & \\
-\mathrm{MCME}(\widehat{\mathbf{y}}^{(j)},\widehat{\mathbf{y}}^{(m)}) & \cdots & -\mathrm{MCME}(\widehat{\mathbf{y}}^{(j)},\widehat{\mathbf{y}}^{(m)}) & \cdots & -2\,\mathrm{MCME}(\widehat{\mathbf{y}}^{(j)},\widehat{\mathbf{y}}^{(m)}) & \cdots & -\mathrm{MCME}(\widehat{\mathbf{y}}^{(j)},\widehat{\mathbf{y}}^{(m)}) & \cdots j \\
-\mathrm{MCME}(\widehat{\mathbf{y}}^{(1)},\widehat{\mathbf{y}}^{(m)}) & & -\mathrm{MCME}(\widehat{\mathbf{y}}^{(i)},\widehat{\mathbf{y}}^{(m)}) & & & & -\mathrm{MCME}(\widehat{\mathbf{y}}^{(m-1)},\widehat{\mathbf{y}}^{(m)}) & \\
\vdots & & \vdots & & \vdots & \ddots & \vdots & \\
\mathrm{MCME}(\widehat{\mathbf{y}}^{(m-1)},\widehat{\mathbf{y}}^{(1)}) & & \mathrm{MCME}(\widehat{\mathbf{y}}^{(m-1)},\widehat{\mathbf{y}}^{(i)}) & & \mathrm{MCME}(\widehat{\mathbf{y}}^{(m-1)},\widehat{\mathbf{y}}^{(j)}) & & \mathrm{MSE}(\widehat{\mathbf{y}}^{(m-1)}) & \\
-\mathrm{MCME}(\widehat{\mathbf{y}}^{(m-1)},\widehat{\mathbf{y}}^{(m)}) & \cdots & -\mathrm{MCME}(\widehat{\mathbf{y}}^{(m-1)},\widehat{\mathbf{y}}^{(m)}) & \cdots & -\mathrm{MCME}(\widehat{\mathbf{y}}^{(m-1)},\widehat{\mathbf{y}}^{(m)}) & \cdots & -2\,\mathrm{MCME}(\widehat{\mathbf{y}}^{(m-1)},\widehat{\mathbf{y}}^{(m)}) & \cdots m-1 \\
-\mathrm{MCME}(\widehat{\mathbf{y}}^{(1)},\widehat{\mathbf{y}}^{(m)}) & & -\mathrm{MCME}(\widehat{\mathbf{y}}^{(i)},\widehat{\mathbf{y}}^{(m)}) & & -\mathrm{MCME}(\widehat{\mathbf{y}}^{(j)},\widehat{\mathbf{y}}^{(m)}) & & & \\
\end{pmatrix}. \tag{6.6}
$$

Matice $\mathbb{A}$ se dá také zapsat kompaktně jako

$$\begin{aligned}\mathbb{A} &= \left(a_{k,l}\right)_{k,l\in\{1,2,\ldots,m-1\}} = \\ &= \Big(\mathrm{MCME}(\widehat{\mathbf{y}}^{(k)}, \widehat{\mathbf{y}}^{(l)}) - \mathrm{MCME}(\widehat{\mathbf{y}}^{(k)}, \widehat{\mathbf{y}}^{(m)}) \\ &\qquad - \mathrm{MCME}(\widehat{\mathbf{y}}^{(l)}, \widehat{\mathbf{y}}^{(m)})\Big)_{k,l\in\{1,2,\ldots,m-1\}}\end{aligned} \tag{6.7}$$

a matice $(\mathbb{C}+\mathbb{A})$ jako

$$\begin{aligned}\mathbb{C}+\mathbb{A} &= \left(ca_{k,l}\right)_{k,l\in\{1,2,\ldots,m-1\}} = \\ &= \Big(\mathrm{MSE}(\widehat{\mathbf{y}}^{(m)}) + \mathrm{MCME}(\widehat{\mathbf{y}}^{(k)}, \widehat{\mathbf{y}}^{(l)}) - \mathrm{MCME}(\widehat{\mathbf{y}}^{(k)}, \widehat{\mathbf{y}}^{(m)}) \\ &\qquad - \mathrm{MCME}(\widehat{\mathbf{y}}^{(l)}, \widehat{\mathbf{y}}^{(m)})\Big)_{k,l\in\{1,2,\ldots,m-1\}}.\end{aligned} \tag{6.8}$$

Výraz (6.8) jde dále zjednodušit dle Tvrzení 6.2:

$$\mathbb{C}+\mathbb{A} = \left(ca_{k,l}\right)_{k,l\in\{1,2,\ldots,m-1\}} = \left(\frac{1}{n}\sum_{i=1}^{n}(\widehat{y}_i^{(k)} - \widehat{y}_i^{(m)})(\widehat{y}_i^{(l)} - \widehat{y}_i^{(m)})\right)_{k,l\in\{1,2,\ldots,m-1\}}. \tag{6.9}$$

Nyní formulujeme a dokážeme hlavní větu této kapitoly.

Věta 6.4 (O optimálním hybridním modelu)**.** *Pokud je matice* $(\mathbb{C}+\mathbb{A})$ *invertovatelná, definujme* $\widehat{\mathbf{w}} := (\mathbb{C}+\mathbb{A})^{-1}(\mathbf{c}-\mathbf{b})$*, a* $\widehat{\mathbf{w}}^* := (\widehat{w}_1, \widehat{w}_2, \ldots, \widehat{w}_{m-1}, 1-\sum_{i=1}^{m-1}\widehat{w}_i)^T$*. Pokud* $\widehat{\mathbf{w}}^* \geq \mathbf{0}$*, pak* $\widehat{\mathbf{w}}^*$ *je optimálním řešením úlohy (6.1) pro ztrátovou funkci* MSE*. Toto optimum je navíc jednoznačné.*

Důkaz.

Část 1: Jednoznačnost.

Vektor $\widehat{\mathbf{w}}$ je vyjádřený pomocí jednoznačně určené inverzní matice. Z jednoznačnosti $\widehat{\mathbf{w}}$ pak plyne jednoznačnost $\widehat{\mathbf{w}}^*$.

Část 2: Všechny predikce jsou navzájem různé.

Nejprve si uvědomíme, že z podmínky invertovatelnosti plyne, že neexistují dva modely, jejichž predikce vstupující do hybridního modelu by byly stejné. Pokud by totiž platilo, že $\widehat{\mathbf{y}}^{(i)} = \widehat{\mathbf{y}}^{(j)}$ pro $i \neq j$, pak z výrazů (6.5) a (6.7) plyne, že i-tý a j-tý sloupec matice $(\mathbb{C}+\mathbb{A})$ jsou stejné. Matice pak nemá plnou hodnost, a není tedy invertovatelná, což je spor s předpokladem.

Část 3: Výpočet pro perfektní predikci.

Pokud existuje index s takový, že $\widehat{\mathbf{y}}^{(s)} = \mathbf{y}$, pak z *Části 2* víme, že je takový pouze jeden. Jsou dvě možnosti, $s \in \{1,2,\ldots,m-1\}$ nebo $s = m$.

Část 3a: Výpočet pro perfektní predikci, $s \in \{1,2,\ldots,m-1\}$.

S využitím Pozorování 6.1, bodu 3, a faktu, že $\mathrm{MSE}(\widehat{\mathbf{y}}^{(s)}) = \mathrm{MSE}(\mathbf{y}) = 0$, se zjednoduší matice $\mathbb{A}$ ve výrazu (6.6) na tvar

$$\Big(\overset{\vdots\,1}{\mathbf{v}_1} \,\Big|\, \cdots \,\Big|\, \overset{\vdots\,i}{\mathbf{v}_i} \,\Big|\, \cdots \,\Big|\, \overset{\vdots\,s}{-\mathbf{b}} \,\Big|\, \cdots \,\Big|\, \overset{\vdots\,m-1}{\mathbf{v}_{m-1}}\Big) \tag{6.10}$$

pro nějaké vektory $\mathbf{v_i} \in \mathbb{R}^{m-1}, i = 1, 2, \ldots, s-1, s+1, \ldots, m-1$. Rovnice (6.5) pak vypadají takto:

$$\Big(\overset{\vdots\,1}{\mathbf{c}+\mathbf{v}_1} \,\Big|\, \cdots \,\Big|\, \overset{\vdots\,i}{\mathbf{c}+\mathbf{v}_i} \,\Big|\, \cdots \,\Big|\, \overset{\vdots\,s}{\mathbf{c}-\mathbf{b}} \,\Big|\, \cdots \,\Big|\, \overset{\vdots\,m-1}{\mathbf{c}+\mathbf{v}_{m-1}}\Big)\mathbf{w} = \mathbf{c}-\mathbf{b}. \tag{6.11}$$

Rovnice (6.11) splňuje řešení $\widehat{\mathbf{w}} = \mathbf{e}_s$, tedy $\widehat{\mathbf{w}}^* = \mathbf{e}_s$, to je navíc dle *Části 1* jednoznačné. Toto řešení je jistě optimální, neboť $\mathrm{MSE}(\widehat{\mathbf{y}}^{(HM_{\widehat{\mathbf{w}}})}) = \mathrm{MSE}(\widehat{\mathbf{y}}^{(s)}) = \mathrm{MSE}(\mathbf{y}) = 0$ a MSE je vždy nezáporné.

Část 3b: Výpočet pro perfektní predikci, $s = m$.

Opět s využitím Pozorování 6.1, bodu 3, a faktu, že $\mathrm{MSE}(\widehat{\mathbf{y}}^{(s)}) = \mathrm{MSE}(\mathbf{y}) = 0$, se výraz (6.5) zjednoduší na $\mathbb{A}\mathbf{w} = \mathbf{0}$. Protože matice $\mathbb{A}$ je dle předpokladu věty invertovatelná, je jediné řešení homogenní soustavy rovnic vektor $\widehat{\mathbf{w}} = \mathbf{0}$, z toho plyne $\widehat{\mathbf{w}}^* = \mathbf{e}_m$. Ze stejného důvodu jako v *Části 3a* je nalezené řešení optimální.

Část 4: $(\mathbb{C} + \mathbb{A})$ *je pozitivně semidefinitní.*

Využijeme kompaktního zápisu (6.9) matice $(\mathbb{C} + \mathbb{A})$. Odtud plyne, že se matice dá zapsat také jako součet

$$\begin{aligned}&\frac{1}{n}\sum_{i=1}^{n}\Big((\widehat{y}_i^{(k)} - \widehat{y}_i^{(m)})(\widehat{y}_i^{(l)} - \widehat{y}_i^{(m)})\Big)_{k,l\in\{1,2,\ldots,m-1\}} = \\ &\quad\frac{1}{n}\sum_{i=1}^{n}\Big(\widehat{y}_i^{(k)} - \widehat{y}_i^{(m)}\Big)_{k\in\{1,2,\ldots,m-1\}}\Big(\widehat{y}_i^{(l)} - \widehat{y}_i^{(m)}\Big)^T_{l\in\{1,2,\ldots,m-1\}}.\end{aligned}$$

Pro symetrické matice, což $(\mathbb{C} + \mathbb{A})$ z Pozorování 6.1, části 2, je, platí, že jsou pozitivně semidefinitní právě tehdy, když se dají zapsat jako $\mathbb{X}\mathbb{X}^T$ pro nějakou matici $\mathbb{X}$, což jsme právě dokázali.

Část 5: Výpočet pro neperfektní predikci.

Funkce $\mathrm{MSE}(\widehat{\mathbf{y}}^{(HM_{\widehat{\mathbf{w}}})})(\mathbf{w})$ je zjevně nekonečně hladká na $\mathbb{R}^{m-1}$, neboť jde o součet polynomů. Z výrazů (6.4) a (6.5) plyne, že

$$\frac{\partial^2 \mathrm{MSE}(\widehat{\mathbf{y}}^{(HM_{\mathbf{w}})})}{\partial\mathbf{w}\partial\mathbf{w}^T}(\mathbf{w}) = \mathbb{C} + \mathbb{A}.$$

Tedy Hessova matice je v každém bodě $\mathbb{R}^{m-1}$ pozitivně semidefinitní. Z toho plyne, že pokud existuje řešení výrazu (6.5) (což předpokládáme) a je jediné (což jsme dokázali), pak se jedná o globální minimum, a tedy optimální řešení. □

6.2.1 Přesný algoritmus

Věta 6.4 nám říká, že pokud se nám podaří vyjádřit vektor $\widehat{\mathbf{w}}^*$ a všechny jeho složky jsou nezáporné, nalezené řešení je optimální. Navrhneme tedy přesný algoritmus pro nalezení řešení úlohy 6.1:

Algoritmus 6.1: Přesný algoritmus pro nalezení hybridního modelu.

Vstup: $\mathbf{y} \in \mathbb{R}^n$, vektor skutečných hodnot
Vstup: $M \in \mathbb{N}$, počet modelů
Vstup: $\widehat{\mathbf{y}}^{(1)}, \widehat{\mathbf{y}}^{(2)}, \ldots, \widehat{\mathbf{y}}^{(M)} \in \mathbb{R}^n$, predikce modelů $1, 2, \ldots, M$
optimal_mse $:= \infty$
optimal_vector := `null`
for i $\in \{1, 2, \ldots, M\}$ **do**
  mse_new $:=$ MSE$(\widehat{\mathbf{y}}^{(i)})$
  if *mse_new* $<$ *mse_optimal* **then**
    mse_optimal $:=$ *mse_new*
    optimal_vector $:= \mathbf{e}_i$ // délka vektoru je M
  end
end
all_invertible := `true`
max_models_count_found := `false`
$m := M$
while (**not** *max_models_count_found*) **and** $m \geq 2$ **do**
  for $\mathbf{v} \in \{0,1\}^M$, $\sum \mathbf{v} = m$ **do**
    Vyber modely s indexy i, kde $\mathbf{v}_i = 1$ a přečísluj je na $1, 2, \ldots, m$.
    Spočítej $\mathbb{C}, \mathbb{A}, \mathbf{c}, \mathbf{b}$ dle (6.6).
    invertible $:= (\mathbb{C} + \mathbb{A})$ je invertovatelná
    if *invertible* **then**
      Spočítej $\widehat{\mathbf{w}}$ a $\widehat{\mathbf{w}}^*$ dle Věty 6.4.
      if $\widehat{\mathbf{w}}^* \geq \mathbf{0}$ **then**
        max_models_count_found := `true`
        mse_new $:=$ MSE$(\widehat{\mathbf{y}}^{(HM_{\widehat{\mathbf{w}}})})$
        if *mse_new* $<$ *mse_optimal* **then**
          mse_optimal $:=$ *mse_new*
          Přečísluj vektor $\widehat{\mathbf{w}}^*$ zpět na původní indexy $1, 2, \ldots, M$
            a ostatní prvky nahraď 0.
          optimal_vector $:= \widehat{\mathbf{w}}^*$
        end
      end
    else
      all_invertible := `false`
    end
  end
  m −= 1
end
if *all_invertible* **then**
  Výstup: *optimal_vector*, řešení je optimální
else
  Výstup: *optimal_vector*, řešení nemusí být optimální
end

Složitost

Určíme složitost tohoto algoritmu v nejhorším případě. Postup vyčíslení ukazuje Tabulka 6.1.

Složitost algoritmu je $\mathcal{O}\left(2^m(m^3 + nm)\right)$, kde n je počet pozorování a m je počet modelů.

Teoreticky je možné snížit složitost na zhruba $\mathcal{O}\left(2^m(m^{2.373} + nm)\right)$, protože prvek m^3 vychází ze složitosti invertování matice o velikosti m krát m. Standardní inverze pomocí Gaussovy-Jordanovy eliminace má složitost právě $\mathcal{O}\left(m^3\right)$. Existují efektivnější algoritmy, momentálně nejrychlejší algoritmus z roku 2014 (viz [20]) má složitost $\mathcal{O}\left(m^{2.3728639}\right)$, nicméně většina algoritmů s nižší než kubickou složitostí není rychlejší na malých maticích. Vzhledem k tomu, že reálná m se pohybují v jednotkách, nejvýše malých desítkách, nebylo by efektivní používat pro invertování matice jiný algoritmus než Gaussovu-Jordanovu eliminaci, proto celkovou složitost uvádíme s touto implementací.

Ovšem místo, kde algoritmus reálně ztrácí nejvíce času, je testování všech podmnožin indexové množiny $\{1,2,\ldots,m\}$, tedy testování všech kombinací všech modelů. Tento cyklus dělá z polynomiálně složitého algoritmu algoritmus s exponenciální složitostí.

Proto dává dobrý smysl řešit problém hledání optimální kombinace modelů v nižší než exponenciální složitosti.

Část	**Kolikrát**	**Podčást**	**Algoritmus**	**Složitost**	**Suma**	**Produkt**	**Finální suma**
Přípravné výpočty	1-krát					$\mathcal{O}(1)$	
		Vlastní algoritmus	m-krát MSE	$\mathcal{O}(nm)$			
			$(m^2-m)/2$-krát MCME	$\mathcal{O}(nm^2)$			
				$\mathcal{O}(nm^2)$	$\mathcal{O}(nm^2)$	$\mathcal{O}(nm^2)$	
						$\mathcal{O}(nm^2)$	$\mathcal{O}(nm^2)$
For cyklus	m-krát					$\mathcal{O}(m)$	
		Vlastní algoritmus	Určení MSE	$\mathcal{O}(1)$			
			Vyhodnocení iterace	$\mathcal{O}(1)$			
				$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	
						$\mathcal{O}(m)$	$\mathcal{O}(m)$
While cyklus	2^{m-1}-krát					$\mathcal{O}(2^m)$	
		Výpočet matic	$\mathbb{C}$	$\mathcal{O}(1)$			
			$\mathbb{A}$	$\mathcal{O}(m^2)$			
			$\mathbf{c}$	$\mathcal{O}(1)$			
			$\mathbf{b}$	$\mathcal{O}(m)$			
				$\mathcal{O}(m^2)$	$\mathcal{O}(m^2)$		
		Vlastní algoritmus	výpočet $(\mathbb{C}+\mathbb{A})$	$\mathcal{O}(m^2)$			
			invertování $(\mathbb{C}+\mathbb{A})$	$\mathcal{O}(m^3)$ [2]			
			výpočet $\widehat{\mathbf{w}}$ a $\widehat{\mathbf{w}}^*$	$\mathcal{O}(m^2)$			
			ověření $\widehat{\mathbf{w}}^* \geq \mathbf{0}$	$\mathcal{O}(m)$			
			predikce HM[3]	$\mathcal{O}(nm)$			
			výpočet MSE HM[3]	$\mathcal{O}(n)$			
			vyhodnocení iterace	$\mathcal{O}(1)$			
				$\mathcal{O}(nm+m^3)$	$\mathcal{O}(nm+m^3)$		
					$\mathcal{O}(nm+m^3)$	$\mathcal{O}(nm+m^3)$	
						$\mathcal{O}\big((2^m)(nm+m^3)\big)$	$\mathcal{O}\big((2^m)(nm+m^3)\big)$
Celý algoritmus							$\mathcal{O}\big((2^m)(nm+m^3)\big)$

[2] lze snížit na $\mathcal{O}\big(m^{2.3728639}\big)$

[3] **H**ybridní **M**odel

Tabulka 6.1: Postupná kalkulace výpočetní složitosti přesného algoritmu.

6.2.2 Aproximativní algoritmus

V této části navrhneme modifikaci přesného Algoritmu 6.1. Základem je jednoduchá heuristika:

- Začneme se všemi m modely a vypočítáme $\widehat{\mathbf{w}}^*$.
- Dokud nedostaneme nezáporný vektor $\widehat{\mathbf{w}}^*$, odebíráme z hybridního modelu postupně vždy nejhorší model z předchozí iterace. Nejhorší model definujeme jako model s nejnižší váhou.
- Pokud se hybridní model redukuje na 2 modely a přesto není vektor $\widehat{\mathbf{w}}^*$ nezáporný, pak je optimální hybridní model tvořen pouze jedním modelem, tím, který měl ve $\widehat{\mathbf{w}}^*$ váhu větší než 1.

Navrhneme tedy Algoritmus 6.2, aproximativní verzi přesného algoritmu s aplikací výše zmíněné heuristiky.

Algoritmus 6.2: Aproximativní algoritmus pro nalezení hybridního modelu.

Vstup: $\mathbf{y} \in \mathbb{R}^n$, vektor skutečných hodnot
Vstup: $M \in \mathbb{N}$, počet modelů
Vstup: $\widehat{\mathbf{y}}^{(1)}, \widehat{\mathbf{y}}^{(2)}, \ldots, \widehat{\mathbf{y}}^{(M)} \in \mathbb{R}^n$, predikce modelů $1, 2, \ldots, M$
pseudooptimal_vector := `null`
invertible := `true`
if $M = 1$ **then**
  pseudooptimal_vector := 1
end
indices := $\{1, 2, \ldots, M\}$
while *invertible* **and** *pseudooptimal_vector* **is null do**
  Vyber modely s indexy *indices* a přečísluj je na $1, 2, \ldots, m$.
  Spočítej $\mathbb{C}, \mathbb{A}, \mathbf{c}, \mathbf{b}$ dle (6.6).
  invertible := $(\mathbb{C} + \mathbb{A})$ je invertovatelná
  if *invertible* **then**
    Spočítej $\widehat{\mathbf{w}}$ a $\widehat{\mathbf{w}}^*$ dle Věty 6.4.
    Přečísluj vektor $\widehat{\mathbf{w}}^*$ zpět na původní indexy $1, 2, \ldots, M$
      a ostatní prvky nahraď 0.
    if $\widehat{\mathbf{w}}^* \geq \mathbf{0}$ **then**
      pseudooptimal_vector := $\widehat{\mathbf{w}}^*$
    else
      if *délka*(*indices*) > 2 **then**
        Vyber nejmenší index i takový, že $\widehat{w}_i^* = \min(\widehat{\mathbf{w}}^*)$
        indices := *indices* $\setminus \{i\}$
      else
        Vyber index j takový, že $\widehat{w}_j^* = \max(\widehat{\mathbf{w}}^*)$
        pseudooptimal_vector := $\mathbf{e}_j$ // délka vektoru je M
      end
    end
  end
end
if *invertible* **then**
  Výstup: *pseudooptimal_vector*
else
  Výstup: *Error*
end

Složitost

Opět určíme složitost toho algoritmu v nejhorším případě. Postup vyčíslení ukazuje Tabulka 6.2.

Složitost algoritmu je $\mathcal{O}\left(m^4 + nm^2\right)$, kde n je počet pozorování a m je počet modelů.

Redukcí počtu kroků v hlavním *while* cyklu algoritmu z 2^{m-1} na $m - 1$ se nám podařilo z algoritmu řešitelného v exponenciálním čase udělat algoritmus řešitelný v polynomiálním čase.

V praxi je počet modelů m, které budou do algoritmu vstupovat, velmi malý

– jednotky, nejvýše malé desítky. Polynomiální složitost čtvrtého řádu v m tak bude hrát ve skutečném výpočetním čase jen velmi malou roli, navíc v n zůstává složitost stále lineární.

Část	**Kolikrát**	**Podčást**	**Algoritmus**	**Složitost**	**Suma**	**Produkt**	**Finální suma**
Přípravné výpočty	1-krát					$\mathcal{O}(1)$	
		Vlastní algoritmus	m-krát MSE	$\mathcal{O}(nm)$			
			$(m^2 - m)/2$-krát MCME	$\mathcal{O}(nm^2)$			
				$\mathcal{O}(nm^2)$	$\mathcal{O}(nm^2)$	$\mathcal{O}(nm^2)$	
						$\mathcal{O}(nm^2)$	$\mathcal{O}(nm^2)$
While cyklus	m-krát					$\mathcal{O}(m)$	
		Výpočet matic	$\mathbb{C}$	$\mathcal{O}(1)$			
			$\mathbb{A}$	$\mathcal{O}(m^2)$			
			$\mathbf{c}$	$\mathcal{O}(1)$			
			$\mathbf{b}$	$\mathcal{O}(m)$			
				$\mathcal{O}(m^2)$	$\mathcal{O}(m^2)$		
		Vlastní algoritmus	výpočet $(\mathbb{C} + \mathbb{A})$	$\mathcal{O}(m^2)$			
			invertování $(\mathbb{C} + \mathbb{A})$	$\mathcal{O}(m^3)$ [4]			
			výpočet $\widehat{\mathbf{w}}$ a $\widehat{\mathbf{w}}^*$	$\mathcal{O}(m^2)$			
			ověření $\widehat{\mathbf{w}}^* \geq \mathbf{0}$	$\mathcal{O}(m)$			
			vybrání nejhoršího modelu	$\mathcal{O}(m)$			
			vyhodnocení iterace	$\mathcal{O}(1)$			
				$\mathcal{O}(nm + m^3)$	$\mathcal{O}(nm + m^3)$		
					$\mathcal{O}(nm + m^3)$	$\mathcal{O}(nm + m^3)$	
						$\mathcal{O}(nm^2 + m^4)$	$\mathcal{O}(nm^2 + m^4)$
Celý algoritmus							$\mathcal{O}(nm^2 + m^4)$

[4] lze snížit na $\mathcal{O}(m^{2.3728639})$

Tabulka 6.2: Postupná kalkulace výpočetní složitosti aproximativního algoritmu.

6.2.3 Teoretické porovnání algoritmů

6.2.3.1 Složitost a možnost distribuovaného počítání

Již jsme uvedli, že aproximativní algoritmus řeší úlohu v polynomiálním čase na rozdíl od přesného algoritmu, který úlohu řeší v exponenciálním čase.

Technickým rozdílem mezi oběma variantami algoritmu je možnost distribuovaného výpočtu. Hlavní výpočetní náročnost jedné iterace algoritmu spočívá v invertování matice. Inverze matice je snadno distribuovatelná úloha a algoritmus je již naimplementovaný například pro framework Apache Spark. Přesný algoritmus pak provádí $m + 2^{m-1}$ na sobě nezávislých iterací, ty lze tedy triviálně paralelizovat. Aproximativní algoritmus provádí iterací pouze m, ale ty paralelizovat nejdou, neboť každá iterace potřebuje na vstupu znát výsledek iterace předchozí.

Je ovšem třeba si uvědomit, že celková výpočetní náročnost je pro m řádově v jednotkách nebo menších desítkách v podstatě zanedbatelná. Jedná se maximálně o desetitisíce iterací algoritmu, který se dá pro fixní m počítat v $\mathcal{O}(n)$. Algoritmus i pro miliony pozorování doběhne řádově ve stovkách milisekund, nejhůře v desítkách sekund, bavíme-li se o vysokoúrovňových jazycích typu Python nebo R. V takovém případě nemá smysl algoritmus distribuovat, protože samotná distribuce zabere více času než nedistribuovaný výpočet.

My ovšem aproximativní algoritmus navrhujeme, aby byl k dispozici, pokud bychom chtěli hybridní model nalézt mnohokrát, například při grid-searchové optimalizaci hyperparametrů modelů. V takovém případě chceme co nejrychlejší exekuci algoritmu.

6.2.3.2 Rozdíly ve výsledcích

Cenou za časové zefektivnění algoritmu jsou dva rozdíly proti původní přesné verzi:

1. Je-li matice $(\mathbb{C}+\mathbb{A})$ singulární, pak přesná verze pokračuje dál a vrátí řešení, o němž není možné rozhodnout, zda je optimální, nicméně může být pořád lepší než žádné nebo expertní a jeho kvalita se dá ověřit. Aproximativní algoritmus při singulární matici $(\mathbb{C} + \mathbb{A})$ skončí.
2. Nalezené řešení nemusí být optimální.

Rozebereme oba body.

Singularita matice První bod nastává ve velmi vzácných situacích, kdy je matice $(\mathbb{C} + \mathbb{A})$ singulární.

Takové případy nastávají (nikoliv výlučně), když (minimálně) dva modely dávají totožné predikce – v takové situaci skutečně nemůže existovat jednoznačný optimální model a do hybridního modelu je třeba zahrnout ze stejných pouze jeden model.

Nicméně prostor všech možných navzájem různých odezev m modelů délky n je $\mathbb{R}^{nm-m(m+1)/2}$ a prostor všech možných matic $(\mathbb{C} + \mathbb{A})$ je podprostorem $\mathbb{R}^{(m-1)^2}$.

Pak v případě, že $nm - m(m+1)/2 > (m-1)^2$ – což je splněno (nikoliv výlučně), když $n > 3m/2$ – je prostor všech navzájem různých odezev větší než prostor všech jim odpovídajících matic ($\mathbb{C} + \mathbb{A}$). Tedy musí existovat i různé modely, pro něž jsou predikce různé, ale vzhledem k vnitřní struktuře vektorů $\widehat{\mathbf{y}}^{(i)}$ a $\widehat{\mathbf{y}}$ nelze optimální hybridní model jednoznačně určit.

Nerovnost $n > 3m/2$ je v praxi téměř jistě splněna, tedy tyto případy mohou s nenulovou pravděpodobností nastat. Kromě skutečné singularity může také dojít k „výpočetní“ singularitě, kdy se z původně regulární matice stane vlivem zaokrouhlování matice skoro singulární (ve výpočetních softwarech se často definuje singulární matice jako matice s číslem podmíněnosti $< 10^{-16}$).

Na to, jak často je matice singulární (ať už skutečně, nebo výpočetně) se podíváme v simulaci. Vzhledem k tomu, že modely jsou různé (dávají různé predikce), mělo by v praxi stačit zvolit jinou množinu trénovacích dat (například odebrat z úlohy jedno pozorování).

Neoptimalita řešení Ztráta garance optimality je běžný problém heuristik tohoto typu. Se stejným problémem se potýká řada algoritmů, například kroková regrese (*stepwise regression*). I náš postup při trénování optimálního ARIMAX modelu využíval částečně tuto heuristiku (i když alespoň kombinovanou s blokovým grid searchem).

Pomocí simulace ověříme, jak vážný problém je ztráta této garance.

6.2.4 Simulace

Uvažujeme 240 variant simulace. Každá varianta je popsána čtveřicí parametrů m, n, `offset` a `deviance_order`:

- m nabývá hodnot 3, 5, 7 a 10;
- n nabývá hodnot 10, 100 a 1000;
- `offset` nabývá hodnot 0.25, 0.5, 0.75 a 1 a
- `deviance_order` nabývá hodnot $-4, -3.5, -3, -2.5$ a -2.

Nejprve se vygeneruje náhodná odezva $\mathbf{r}$ délky n:

$$\mathbf{r} = (r_1, r_2, \ldots, r_n)^T,\ r_i \sim \mathrm{U}(-1,\ 1), \tag{6.12}$$

kde U je rovnoměrné rozdělení.

Dále se pro každý model s indexem m^* z $\{1, 2, \ldots, m\}$ vygeneruje posunutí:

$$o^{(m^*)} \sim (m^* + 5 \cdot \mathbb{1}(m^* \geq 2))(-1)^{\mathrm{Alt}(1/2)\texttt{offset}}, \tag{6.13}$$

kde Alt je Bernoulliho rozdělení.

Finální predikce $\mathbf{r}^{(m^*)}$ délky n se vygeneruje dle vzorce:

$$\begin{aligned}
&\mathbf{r}^{(m^*)} = (r_1^{(m^*)}, r_2^{(m^*)}, \ldots, r_n^{(m^*)})^T, \\
&r_i^{(m^*)} \sim r_i + o^{(m^*)} + \mathrm{U}(-10^{\texttt{deviance_order}} \cdot m^*,\ 10^{\texttt{deviance_order}} \cdot m^*),
\end{aligned} \tag{6.14}$$

kde U je opět rovnoměrné rozdělení.

Tyto předpisy byly stejně jako testované hodnoty parametrů nalezeny tak, aby v rámci simulací byla dostatečná variabilita – konkrétně aby existovaly optimální hybridní modely s malým i velkým počtem modelů.

Pro každou z 240 variant simulace simulujeme 1000 iterací následujícího postupu:

- Vygenerujeme data dle (6.12), (6.13) a (6.14).
- Na datech spustíme přesný algoritmus 6.1. Zaznamenáme, jestli je matice $(\mathbb{C} + \mathbb{A})$ invertovatelná a pokud ano, uložíme řešení $\widehat{\mathbf{w}}^*_{\text{přesný}}$.
- Pokud předchozí algoritmus nevrátil neinvertovatelnost, spustíme na datech aproximativní algoritmus 6.2. Žádná matice v průběhu nebude neinvertovatelná, takže algoritmus doběhne a vrátí řešení $\widehat{\mathbf{w}}^*_{\text{aproximativní}}$.
- Máme-li řešení, porovnáme je a zaznamenáme, zda jsou shodná.

Pro každou variantu nás zajímá

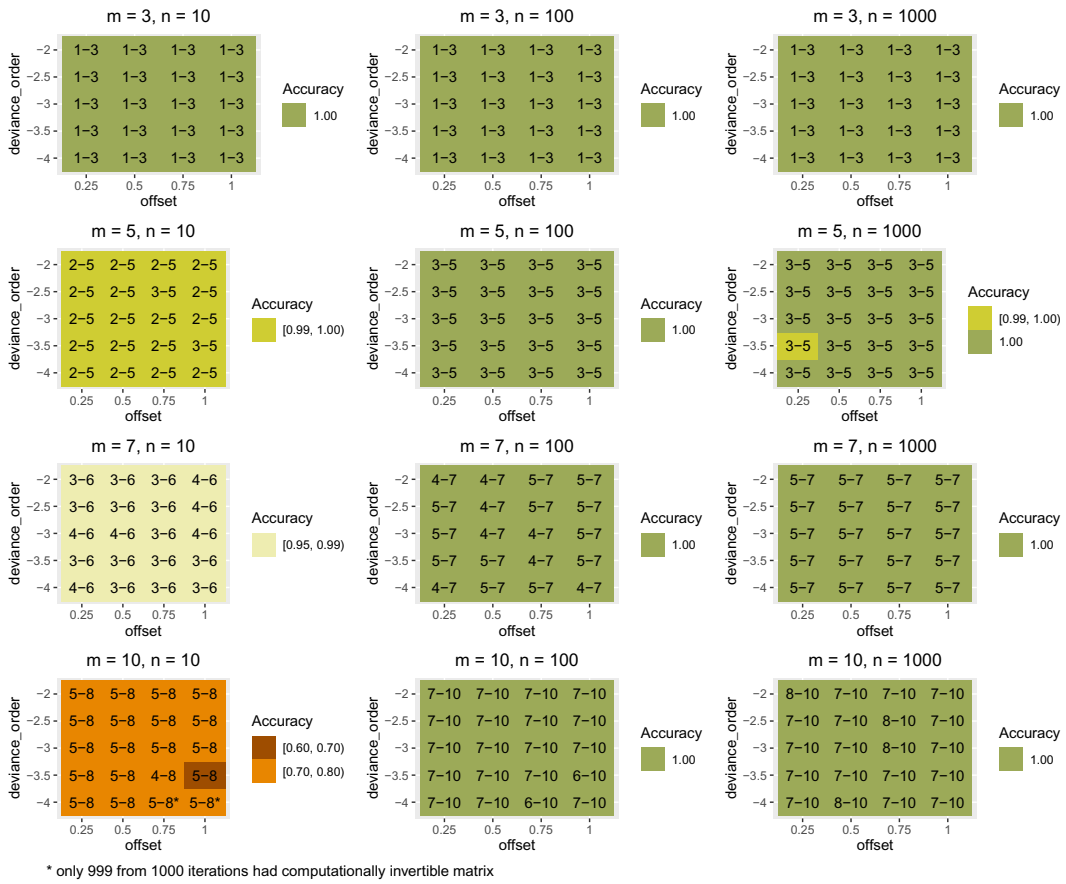
- kolik z 1000 iterací doběhlo, tj. jak často máme garantované optimální řešení, a
- pokud iterace doběhla, jak často je ve shodě garantované optimální řešení z přesného algoritmu s řešením z aproximativního algoritmu.

Výsledky Z 240 000 simulací nedoběhly pouze 2, obě při variantě $n = m = 10$. Ze 180 000 simulací, kdy bylo $n > 3m$, se výsledky shodovaly ve 179 999 případech, tj. jednou se lišily. V případech, kdy se n začalo blížit hodnotám m, přesnost klesala. Pro $m = 5, n = 10$ byla zhruba 99.7 %; pro $m = 7, n = 10$ zhruba 97 % a pro $n = m = 10$ zhruba 74 %.

Kompletní výsledky včetně rozpadu dle parametrů `offset` a `deviance_order` lze nahlédnout na Obrázku 6.1. Barevná škála zde označuje úspěšnost aproximativního algoritmu. Interval vizualizovaný v každém jednotlivém bloku označuje 10% a 90% výběrový kvantil počtu nenulových vah v optimálním hybridním modelu (tomuto číslu budeme říkat optimální počet modelů v optimálním hybridním modelu).

Například v bloku odpovídajícímu $m = 5$ a $n = 10$, na místě odpovídajícímu `offset` $= 0.25$ a `deviance_order` $= -2$ vidíme hodnotu 2-5. To znamená, že alespoň 10 % iterací (tj. 100) mělo optimální počet modelů v optimálním hybridním modelu maximálně 2 a alespoň 10 % minimálně 5 (protože $m = 5$, tak přesně).

Pokud je optimální počet modelů rovný m, pak je ekvivalence algoritmů zaručena (lze snadno nahlédnout z implementace a důkazu Věty 6.4). Pro optimální počet modelů 1 to sice nelze garantovat, ale intuitivně je to pravděpodobnější než pro počet modelů ostře mezi 1 a m. Proto jsme výrazy (6.13), (6.14) a hodnoty parametrů volili tak, abychom rozdělení tohoto počtu udělali co nejširší. Výsledky prezentované na Obrázku 6.1 jsou v tomto ohledu nejlepší dosažené.



Obrázek 6.1: Výsledky aproximativního algoritmu validovaného pomocí přesného algoritmu.

Ze simulace tedy plyne, že aproximativní algoritmus téměř jistě dojde ke správnému řešení, je-li počet pozorování řádově větší než počet modelů. Pro 3 modely se ukázalo být dostačující dokonce 10 pozorování.

6.2.5 Příklad použití: Aplikace na predikovanou časovou řadu

Ukážeme si příklad použití aproximativního algoritmu 6.2 na námi nafitovaných modelech. Abychom neukazovali pouze triviální případ dvou algoritmů, přidáme k XGBoost modelu s optimalizovanými hyperparametry ze sekce 4.5.1 (XGB-O) a finálnímu ARIMAX modelu ze sekce 3.2.4 ještě další dva: naivní model (NAIVE) – zpožděnou odezvu a XGBoost model bez optimalizovaných hyperparametrů (XGB-NO) ze sekce 4.4.

Out-of-sample množinu rozdělíme náhodně na (téměř) poloviny pro určení (125 pozorování) a validaci (124 pozorování) hybridního modelu.

V Tabulce 6.3 vidíme postup aproximativního algoritmu. V první iteraci algoritmus nezastaví, protože vektor vah má dva záporné prvky – vyřazujeme naivní model, protože jeho váha je nejmenší. V druhé iteraci vyřazujeme na základě stejné logiky XGBoost model bez optimalizovaných hyperparametrů. Ve třetí iteraci již vektor vah splňuje podmínku nezápornosti a algoritmus končí. Celkové MSE se

	XGB-O	**XGB-NO**	**ARIMAX**	**NAIVE**	
MSE	249 038.0	373 020.8	357 751.2	466 279.8	**MSE HM**[5]
1. iterace	1.011	-0.028	0.147	-0.130	–
2. iterace	0.973	-0.079	0.107	NA	–
3. iterace	0.914	NA	0.086	NA	–
Výstup	0.914	0	0.086	0	248 069.7

[5] **H**ybridní **M**odel

Tabulka 6.3: Tabulka demonstrující postup aproximativního algoritmu na trénovací části out-of-sample množiny.

proti nejlepšímu samostatnému modelu zlepšilo z 249 038 na 248 070.

Abychom pro hybridní model vytvořili i jiný benchmark než jednotlivé nekombinované modely, využijeme jednoduché alternativní přístupy uvedené v úvodu této kapitoly, konkrétně všechny kromě expertního odhadu.

Pro benchmarky budeme využívat pouze ARIMAX model a oba XGBoost modely, tedy vynecháváme naivní predikci. Tu nepoužíváme, protože z průběhu trénování je jasné, že úloha je rozumně modelovatelná – to znamená, že naivní predikce není rozumnou alternativou. Náš algoritmus toto odhalí a naivní predikci dá váhu 0, nicméně pokud bychom jí přidali fixní váhu, neférově bychom ubírali sílu alternativních přístupů, neboť v praxi bychom do hybridního modelu v tomto případě naivní predikci jistě nedávali.

Použijeme tedy tři přístupy:

- **rovnoměrné váhy**, tj. každý model vstupuje do metamodelu se stejnou váhou 1/3;
- **váhy dle pořadí**, tj. nejlepší model (optimalizovaný XGBoost) vstupuje do metamodelu s váhou 1/2, druhý nejlepší (ARIMAX) s váhou 1/3 a poslední (neoptimalizovaný XGBoost) s váhou 1/6.
- **váhy dle MSE**, tj. váhy budou ve stejném poměru, v jakém jsou hodnoty 1/MSE.

Celkem tedy porovnáváme čtyři přístupy, jak shrnuje Tabulka 6.4.

	XGB-O	**XGB-NO**	**ARIMAX**	**NAIVE**	
MSE	249 038.0	373 020.8	357 751.2	466 279.8	**MSE HM**[6]
Rovnoměrné váhy	1/3	1/3	1/3	0	279 841.5
Váhy dle pořadí	1/2	1/6	1/3	0	264 825.3
Váhy dle MSE	0.423	0.282	0.294	0	271 514.0
Optimální HM[6]	0.914	0	0.086	0	248 069.7

[6] **H**ybridní **M**odel

Tabulka 6.4: Tabulka porovnávající váhy optimálního hybridního modelu a váhy tří alternativních přístupů spolu s výsledným MSE na trénovací části out-of-sample množiny.

Důležitá ale bude výkonnost na validační části out-of-sample množiny.

Hybridní model	**MSE train-OOS**[7]	**MSE valid-OOS**[8]
XGB-O	249 038.0	273 029.8
XGB-NO	373 020.8	396 529.6
ARIMAX	357 751.2	383 896.7
NAIVE	466 279.8	399 432.6
Rovnoměrné váhy	279 841.5	303 927.1
Váhy dle pořadí	264 825.3	290 270.9
Váhy dle MSE	271 514.0	296 112.4
Optimální HM[9]	248 069.7	272 937.9

[7] MSE na trénovací části out-of-sample množiny

[8] MSE na validační části out-of-sample množiny

[9] **H**ybridní **M**odel

Tabulka 6.5: Tabulka MSE jednotlivých modelů a hybridních modelů na trénovací a validační části out-of-sample dat.

Když se podíváme na výsledky na validační části out-of-sample množiny do Tabulky 6.5, vidíme, že hybridní model zůstal i na validační množině lepší než nejlepší ze všech jednotlivých modelů, i když už jen o trochu.

Je to jediný hybridní model, který je jak na trénovací, tak na validační části dat lepší než samotný optimalizovaný XGBoost. Tento konkrétní případ tedy ukazuje, že využít optimální hybridní model je lepší, než se spokojit pouze s jednoduchými alternativními přístupy.

Nicméně je třeba podotknout, že v tomto konkrétním případě je MSE optimalizovaného XGBoost modelu výrazně nižší než MSE všech ostatních vstupních modelů, což může výsledky zkreslovat.

K zjištění, za jakých podmínek je obecně signifikantně lepší kombinovat modely optimálně, by byla potřeba nezávislá simulační studie na různých datech a s různými modely, což je téma nad rámec této práce.

6.3 Zhodnocení zvoleného přístupu

V této sekci jsme teoreticky vyřešili otázku, jak nalézt optimální hybridní model dle Definice 6.1 pro kritérium MSE, navrhli jsme přesný algoritmus 6.1 k jeho nalezení a aproximativní verzi 6.2. U obou verzí jsme určili výpočetní složitost. Pomocí simulace jsme ukázali, že pro $n > 3m$ má aproximativní verze téměř stoprocentní úspěšnost. Použití algoritmu jsme pak demonstrovali na řešeném problému a ukázali výhodu použití optimálního hybridního modelu jednak proti jednotlivým modelům, jednak proti jednoduchým alternativám, jak hybridní model konstruovat.

Aproximativní algoritmus je vhodné využít, když nepotřebujeme 100% jistotu správného výsledku – například při optimalizaci hyperparametrů vstupních modelů. Po zoptimalizování je pak na finální model vhodné použít přesnou verzi. Stejný postup se často využívá i v XGBoostu, kde existují přesná i aproximativní verze algoritmu na hledání optimálního stromu, jak bylo popsáno v sekci 1.3.5.

Obecně jsme v této kapitole pracovali s předpokladem, že nejprve v rámci testovaných rodin a tříd modelů nalezneme nejlepší modely (nebo kandidáty na nejlepší modely) a ty pak zkombinujeme do hybridního modelu, vstupních modelů pro hybrid by tak byly jednotky, nejvýše malé desítky. Zcela alternativní přístup by mohl být využití optimálního hybridního modelu jako selekčního kritéria. Takový přístup by znamenal, že se místo grid-searchové optimalizace hyperparametrů například XGBoost modelu natrénují všechny varianty a na tyto modely se pustí algoritmus pro nalezení optimálního hybridního modelu. Ten by měl dát největší váhu těm kombinacím hyperparametrů modelu, které nejlépe predikují odezvu na validační množině, což je stejné kritérium, jaké používáme při „tradiční“ optimalizaci hyperparametrů. Tato metodologie může být předmětem dalšího zkoumání, které je ale nad rámec této diplomové práce.

Hledání optimálního metamodelu je možné dále rozšířit uvolněním dosud implicitního předpokladu, že vektor vah je pro každé pozorování stejný. Tento přístup však vnáší do problému výrazně větší míru komplexity, navíc bez omezení vztahu jednotlivých váhových vektorů se už jedná dokonce o neparametrickou úlohu.

6.3.1 Alternativní přístup

Úloha by se dala řešit také pomocí kvadratického programování.

Velmi podobná úloha je například Markowitzův model portfolia s minimální volatilitou a bez krátkých prodejů (*short sales*), popsaná například v [32], s jedním rozdílem – v naší úloze není analogie podmínky na váženou střední hodnotu (která v Markowitzově modelu reprezentuje požadovaný odhadnutý výnos portfolia). Jinak matice tvořená MCME odpovídá matici kovariancí mezi jednotlivými tituly a jednotlivé váhy modelů odpovídají poměru investic do jednotlivých titulů. Markowitzův model s krátkými prodeji pak uvolňuje omezení na nezápornost jednotlivých vah.

Existují dvě možné formulace úlohy nalezení hybridního modelu jako úlohy kvadratického programování:

- Úlohu formulujeme s $m-1$ vahami a poslední váhu dopočítáme, v takovém případě je matice kvadratické formy stejná jako ve výrazu (6.9). Omezení jsou pak dvě – nezápornost jednotlivých vah a jejich součet menší nebo roven 1.
- Úlohu formulujeme s m vahami. Pak se druhé omezení na součet vah změní (součet musí být přesně 1).

Náš přístup se však na tuto konkrétní problematiku dívá jinou optikou (než kvadratické programování) specifickou pro problém hledání hybridního modelu dle MSE. Místo řešení pomocí numerické optimalizace řeší úlohu pomocí přesného řešení rovnic (6.5). Ve Větě 6.4 jsme ukázali, za jakých podmínek dostáváme optimum, a velmi přesně jsme vyčíslili složitost takto formulovaného problému.

Bylo by nad rámec této diplomové práce určit, zda má tato specifická metoda v praxi výhody v porovnání s již používanými obecnými solvery pro kvadratické programování. Důvodů je několik:

- Definovat „běžná“ data, na kterých by se porovnání provádělo, je obtížné, navíc oblasti i rozsahy praktických problémů se velmi rychle mění.
- Solverů existuje několik desítek a každý z nich má několik variant. Například populární Frankův-Wolfeho algoritmus ([12]) je pouze algoritmem pro převedení konvexního programování na programování lineární. Lineární problém se typicky dá řešit velmi efektivně pomocí simplexové metody, nicméně je stále otevřený problém, zda má simplexová metoda obecně polynomiální složitost (viz např. [16]).
- Naše verze algoritmu by bylo třeba naimplementovat v nízkoúrovňových jazycích, ve kterých jsou napsány stávající solvery, aby bylo porovnání průkazné. I v takových případech je ale konkrétní výpočetní čas závislý na konkrétní implementaci.

Z výše zmíněných důvodů neuzavíráme tuto kapitolu s jednoznačným stanoviskem, zda jsme navrhli v nějakém smyslu lepší nebo horší metodu pro řešení úlohy nalezení hybridního modelu dle MSE, pouze poskytujeme alternativu k již existujícím řešením.

Závěr

V této diplomové práci jsme po uvedení teoretického rámce natrénovali postupně z našeho pohledu optimální modely ARIMAX a XGBoost pro predikci denního zobchodovaného objemu indexu S&P 500 a jejich výsledky porovnali na out-of-sample množině, která nebyla po celou dobu trénování vzata žádným způsobem v úvahu.

Ukázalo se, že pro tento typ úlohy je model XGBoost vhodnější, neboť dokáže výrazně přesněji predikovat vysvětlovanou proměnnou – je nicméně nutné optimalizovat hyperparametry. Neoptimalizovaný model byl mírně horší než model ARIMAX, dle některých kritérií mezi modely nebyl rozdíl.

Natrénovat model XGBoost trvalo řádově kratší dobu. Z důvodu rychlejšího natrénování jednoho konkrétního modelu může být navíc celý postup hledání finálního modelu přímočařejší, neboť při trénování modelu ARIMAX jsme museli přistupovat k postupnému ořezávání prostoru hyperparametrů, abychom byli úlohu schopni v rozumném čase dovést k finálnímu modelu.

Jako nejvýznamnější prediktory se ukázaly zpožděné hodnoty a trend, následované počtem dní od posledního a do následujícího obchodního dne.

V poslední části práce jsme odvodili teorii tzv. hybridního modelu. Ten kombinuje predikce vstupních modelů – kombinace predikcí může být obecně lepší než nejlepší z jednotlivých predikcí. Pro vybrané kritérium (MSE) jsme odvodili, jak vypadá hybridní model na základě optimalizační úlohy minimalizace MSE na prostoru všech konvexních kombinací vstupních modelů. Navrhli jsme přesný algoritmus, jak takový hybridní model najít a jeho aproximativní verzi, která je méně výpočetně náročná. Pomocí simulace jsme ukázali, že pokud je počet pozorování řádově větší než počet modelů, dává aproximativní algoritmus téměř vždy stejný výsledek.

Hybridní model jsme na out-of-sample množině použili i na námi identifikované modely.

Indikátory důležitých ekonomických zpráv

Prediktory, o kterých jsme předpokládali, že by mohly mít na vysvětlovanou proměnnou silný vliv – indikátory důležitých ekonomických zpráv v daný a předchozí obchodní den – se ukázaly v kontextu dalších uvažovaných prediktorů jako nejméně významné. Zajímavé by mohlo být například zkoumat vlastnosti těchto proměnných, pokud by byly definovány detailněji, například indikátory různých typů zpráv nebo indikátory pozitivních a negativních zpráv, překvapivých nebo naopak očekávaných závěrů těchto zpráv. Nicméně by bylo poměrně problematické takto označené události získat zpětně dostatečně hluboko do historie.

Alternativní postupy optimalizace hyperparametrů modelu XGBoost

V průběhu optimalizace hyperparametrů modelu XGBoostu byly voleny různé alternativní způsoby, mimo jiné expertní odhady některých parametrů, optimalizace hyperparametrů po jednom, a ne sdruženě. Konzistentně s prohlášením v úvodu práce se do závěrečné verze práce dostal pouze způsob, který jsme před jeho validací na out-of-sample považovali za nejlepší.

Ve skutečnosti se ukázalo, že nejlepší nebyl – mimo hranice výzkumné části práce lze přiznat, že expertní odhady dávaly obecně lepší výsledky – konkrétně zafixovaná hloubka stromu 3 (proti zoptimalizované hodnotě 5), neoptimalizování parametru `gamma`, resp. jeho ponechání na 0, a subsampling kolem hodnoty 0.8.

Lepších výsledků by model paradoxně dosáhl, pokud by se žádná skupina proměnných neoptimalizovala sdruženě, nýbrž docházelo-li by pouze k postupnému optimalizování jednotlivých proměnných. Jedním z důvodů by bylo například, že při expertně zvolených defaultních hodnotách parametrů (hlavně `gamma` na 0) a včasné optimalizaci maximální hloubky stromu, by algoritmus zvolil jako optimální již zmíněnou hodnotu 3.

V řeči čísel by se jednalo o následující zlepšení: Neoptimalizovaný XGBoost model má MSE v tisících cca 365, námi optimalizovaný cca 266, nejlepší nalezený ovšem dokonce 235.

Důvodem dosažení tohoto neoptimálního výsledku apriori optimálním přístupem je pravděpodobně malý počet pozorování, který způsobil, že při sdružené optimalizaci docházelo k nacházení lokálních namísto globálních optim, přestože to nebylo na první pohled v hrubém grid searchi vidět.

Při trénování jsme optimalizovali 8 hyperparametrů, což může být pro rozsahem takto malou úlohu příliš mnoho. I z toho důvodu byl pravděpodobně expertní názor na správné hodnoty klíčových parametrů lepším přístupem než obecně korektnější grid search. Věříme, že při větším počtu pozorování by se námi zvolený způsob ukázal jako optimálnější.

Jak se ukázalo, volba hyperparametrů je v rámci „moderních“ metod strojového učení klíčová. V současnosti neexistuje jednotný přístup k jejich optimalizaci, a to i z toho důvodu, že pro různé typy a velikosti úlohy může být optimální přístup jiný.

Možnost dalšího výzkumu v oblasti hybridních modelů

V nejrozsáhlejší části práce, kapitole 6, jsme odvodili, jak určit optimální hybridní model dle MSE, a k této problematice navrhli a otestovali dva algoritmy. Jsme si vědomi toho, že k výsledku se kromě námi navrženými algoritmy dá dostat i pokud problém formulujeme jako problém kvadratického programování a řešíme ho pomocí numerických solverů.

Výstupem této práce není jednoznačné zjištění, zda nebo případně za jakých podmínek může být námi navržený algoritmus vhodné využít na úkor tradičních přístupů. Je sice určena jeho teoretická složitost, dokázána jeho korektnost (v přesné verzi) a pomocí simulace je určena přesnost aproximativní verze, nicméně zhodnocení reálného přínosu bylo nad rámec této práce. V závěru kapitoly, konkrétně sekci 6.3.1, jsou uvedeny důvody, proč je takové objektivní porovnání velmi náročné. Nicméně by mohlo být předmětem separátního výzkumu.

Navíc jsme v závěru této kapitoly naznačili myšlenku, jak hybridní model využít pro optimalizaci hyperparametrů různých modelů, a to nejen různých modelů zvlášť, ale i různých modelů (z různých rodin) současně. Tudy by se mohl ubírat další separátní výzkum.

Literatura

[1] AHLBURG, Dennis. Forecast evaluation and improvement using theil's decomposition. *Journal of Forecasting*. 1984, **3**(3), 345-351. ISSN 02776693. Dostupné z: http://doi.wiley.com/10.1002/for.3980030313

[2] AL-NAJJAR, Dana. Modelling and Estimation of Volatility Using ARCH/-GARCH Models in Jordan's Stock Market. *Asian Journal of Finance & Accounting*. 2016, **8**(1), 152-167. ISSN 1946-052X. Dostupné z: http://www.macrothink.org/journal/index.php/ajfa/article/view/9129

[3] ANGGRAENI, Wiwik, Retno Aulia VINARTI a Yuni Dwi KURNIAWATI. Performance Comparisons between Arima and Arimax Method in Moslem Kids Clothes Demand Forecasting: Case Study. *Procedia Computer Science*. 2015, **72**(1), 630-637. ISSN 18770509. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S1877050915036339

[4] BLIEMEL, Friedhelm. Theil's Forecast Accuracy Coefficient: A Clarification. *Journal of Marketing Research*. 1973, **10**(4). ISSN 00222437. Dostupné z: https://www.jstor.org/stable/3149394?origin=crossref

[5] BREIMAN, Leo. *Classification and Regression Trees*. 1. Monterey: Wadsworth & Brooks/Cole Advanced Books & Software, 1984. ISBN 978-0534980535.

[6] BREIMAN, Leo. Random Forests. *Machine Learning*. **45**(1), 5-32. ISSN 08856125. Dostupné z: http://link.springer.com/10.1023/A:1010933404324

[7] BREUSCH, Trevor a Adrian PAGAN. A Simple Test for Heteroscedasticity and Random Coefficient Variation. *Econometrica*. 1979, **47**(5). ISSN 00129682. Dostupné z: https://www.jstor.org/stable/1911963?origin=crossref

[8] BOX, George a Gwilym JENKINS. *Time series analysis: forecasting and control*. 2. San Francisco: Holden-Day, 1976. ISBN 978-0816211043.

[9] CHEN, Tianqi a Carlos GUESTRIN. XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2016, 785-794. ISBN 9781450342322. Dostupné z: https://dl.acm.org/doi/10.1145/2939672.2939785

[10] CIPRA, Tomáš. *Finanční ekonometrie*. 2., upr. vyd. Praha: Ekopress, 2013. ISBN 978-80-86929-93-4.

[11] DESCAMPS, Benoit. *Regression prediction intervals with XGBOOST* [online]. 25. 4. 2017 [cit. 2020-09-23]. Dostupné z: https://towardsdatascience.com/regression-prediction-intervals-with-xgboost-428e0a018b

[12] FRANK, Marguerite a Philip WOLFE. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*. 1956, **3**(1-2), 95-110. ISSN 00281441. Dostupné z: http://doi.wiley.com/10.1002/nav.3800030109

[13] FRIEDMAN, Jerome. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*. 2001, **29**(5), 1189-1232. ISSN 0090-5364. Dostupné z: http://projecteuclid.org/euclid.aos/1013203451

[14] FRIEDMAN, Jerome. Stochastic gradient boosting. *Computational Statistics & Data Analysis*. 2002, **38**(4), 367-378. ISSN 01679473. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S0167947301000652

[15] GRANGER, Clive a Paul NEWBOLD. Some comments on the evaluation of economic forecasts. *Applied Economics*. 2006, **5**(1), 35-47. ISSN 0003-6846. Dostupné z: https://www.tandfonline.com/doi/full/10.1080/00036847300000003

[16] HANSEN, Thomas a Uri ZWICK. An Improved Version of the Random-Facet Pivoting Rule for the Simplex Algorithm. *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing - STOC '15*. New York, New York, USA: ACM Press, 2015, 209-218. ISBN 9781450335362. Dostupné z: http://dl.acm.org/citation.cfm?doid=2746539.2746557

[17] HO, Tim Kam. Random decision forests. *Proceedings of 3rd International Conference on Document Analysis and Recognition*. IEEE Comput. Soc. Press, 1995, 278-282. ISBN 0-8186-7128-9. Dostupné z: http://ieeexplore.ieee.org/document/598994/

[18] HYAFIL, Laurent a Ronald RIVEST. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*. 1976, **5**(1), 15-17. ISSN 00200190. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/0020019076900958

[19] MAKRIDAKIS, Spyros, Steven WHEELWRIGHT a Rob HYNDMAN. *Forecasting: Methods and Applications*. 3. New York: Wiley, 1997. ISBN 978-0-471-53233-0.

[20] LE GALL, François. Powers of tensors and fast matrix multiplication. *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation - ISSAC '14*. New York, NY, USA: ACM, 2014, 296-303. ISBN 9781450325011. Dostupné z: http://dl.acm.org/citation.cfm?doid=2608628.2608664

[21] LJUNG, Greta a George BOX. On a measure of lack of fit in time series models. *Biometrika*. 1978, **65**(2), 297-303. ISSN 0006-3444. Dostupné z: https://academic.oup.com/biomet/article-lookup/doi/10.1093/biomet/65.2.297

[22] LUNDBERG, Scott a Su-In LEE. Consistent feature attribution for tree ensembles. *Computing Research Repository*. 2017. Dostupné z: https://arxiv.org/abs/1706.06060v6

[23] NIELSEN, Didrik. *Tree Boosting With XGBoost - Why Does XGBoost Win „Every" Machine Learning Competition?*. Trondheim, 2016. Diplomová práce. Norwegian University of Science and Technology. Vedoucí práce Håvard Rue.

[24] PINDYCK, Robert a Daniel RUBINFELD. *Econometric Models and Economic Forecasts*. 4. Londýn: McGraw-Hill, 1997. ISBN 978-0079132925.

[25] SHAPIRO, Samuel a Martin WILK. An analysis of variance test for normality (complete samples). *Biometrika*. 1965, **52**(3-4), 591-611. ISSN 0006-3444. Dostupné z: https://academic.oup.com/biomet/article-lookup/doi/10.1093/biomet/52.3-4.591

[26] VANICHRUJEE, Ukrish, Teerayut HORANONT, Wasan PATTARA-ATIKOM, Thanaruk THEERAMUNKONG a Takahiro SHINOZAKI. Taxi Demand Prediction using Ensemble Model Based on RNNs and XGBOOST. *2018 International Conference on Embedded Systems and Intelligent Technology & International Conference on Information and Communication Technology for Embedded Systems (ICESIT-ICICTES)*, Khon Kaen, 2018, 1-6, ISBN 978-1-5386-7063-7. Dostupné z: https://ieeexplore.ieee.org/document/8442063/

[27] VINAYAK, Rashmi Korlakai a Ran GILAD-BACHRACH. DART: Dropouts meet Multiple Additive Regression Trees. *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*. 2015, PMLR 38, 489-497. Dostupné z: http://proceedings.mlr.press/v38/korlakaivinayak15.html

[28] WANG, Yan a Yuankai GUO. Forecasting method of stock market volatility in time series data based on mixed model of ARIMA and XGBoost. *China Communications*. 2020, **17**(3), 205-221. ISSN 1673-5447. Dostupné z: https://ieeexplore.ieee.org/document/9058617/

[29] XGBoost Documentation [online]. [cit. 2020-09-06]. Dostupné z: https://xgboost.readthedocs.io/en/latest/

[30] XGBoost Package Readme: Machine Learning Challenge Winning Solutions [online]. [cit. 2020-09-06]. Dostupné z: https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions

[31] XGBoost Python Documentation [online]. [cit. 2020-09-06]. Dostupné z: https://xgboost.readthedocs.io/en/latest/python

[32] ZIVOT, Eric. Portfolio Theory with No Short Sales: Econ 424 Notes. University of Washington, 2014. Dostupné z: https://faculty.washington.edu/ezivot/econ424/portfoliotheorynoshortsalesslides.pdf

Seznam obrázků

1.1 Příklad dvou regresních stromů a jejich reprezentace pomocí funkce f. *Zdroj:* [9]*, Figure 1, upraveno.* 12
1.2 Výpočetní optimální hodnoty objective funkce pro jeden konkrétní strom. *Zdroj:* [9]*, Figure 2, upraveno.* 15

2.1 Grafy uzavírací ceny, `volume` a diferencované `volume`. 27

3.1 Korelogramy a parciální korelogramy pro časovou řadu objemu, její první a druhé diference. 29
3.2 Analýza reziduí při hledání exogenních lineárních a loglineárních reziduí pomocí histogramu, Q-Q plotu a scatter plotu reziduí proti vyrovnaným hodnotám. 31
3.3 Predikce objemu pomocí exogenních proměnných, bez ARIMA struktury. 32
3.4 Graf popisující závislost AIC, MSE a MAE na d a `use_log` napříč všemi řády p a q. 33
3.5 Graf popisující závislost MSE a MAE na p, q při `use_log = False` a $d = 2$. Seznam chybových kódu je v Příloze A.1. Barevná škála reprezentuje monotónní, ale nelineární transformaci MSE, resp. MAE, tak, aby byla barevná škála reprezentativní. 34

4.1 Výkonnost modelů při optimalizaci hyperparametrů `max_depth` a `min_child_weight`. Barevná škála reprezentuje monotónní, ale nelineární transformaci MSE, tak, aby byla barevná škála reprezentativní. 45
4.2 Výkonnost modelů při optimalizaci řádu hyperparametru `gamma`. Na ose x je $\log_{10}\gamma$. Graf neobsahuje hodnotu $\gamma = 0$, která ale nereprezentuje žádný z nejlepších 5 modelů. Barevná škála reprezentuje MSE. 46
4.3 Výkonnost modelů při optimalizaci hyperparametru `gamma`. Barevná škála reprezentuje MSE. 46
4.4 Výkonnost modelů při optimalizaci hyperparametrů `subsample` a `colsample_bytree`. Barevná škála reprezentuje monotónní, ale nelineární transformaci MSE, tak, aby byla barevná škála reprezentativní. 47
4.5 Výkonnost modelů při optimalizaci řádů hyperparametrů `alpha` a `lambda`. Na osách je dekadický logaritmus těchto parametrů. Graf neobsahuje hodnoty $\lambda = 0$ a $\alpha = 0$, které ale nereprezentují žádný z nejlepších 5 modelů. Barevná škála reprezentuje monotónní, ale nelineární transformaci MSE, tak, aby byla barevná škála reprezentativní. 48
4.6 Výkonnost modelů při optimalizaci hyperparametrů `alpha` a `lambda`. Barevná škála reprezentuje monotónní, ale nelineární transformaci MSE, tak, aby byla barevná škála reprezentativní. 49
4.7 Výkonnost modelů při optimalizaci hyperparametru `eta`. Barevná škála reprezentuje MSE. 49

4.8 Výkonnost modelů po letech měřena pomocí R^2 a U. Šedým podbarvením reprezentujeme znormovanou směrodatnou odchylku objemu v daném roce – tj. míru volatility. Černá vertikála odděluje in-sample a out-of-sample množinu. 50
4.9 Relativní důležitost prediktorů v optimalizovaném modelu. 51

5.1 Predikce pomocí modelu ARIMAX na out-of-sample množině včetně všech 10 submodelů, které dohromady tvoří finální model. 54
5.2 Predikce pomocí modelu XGBoost s optimalizovanými parametry na out-of-sample množině. Pro srovnání vykreslen i model ARIMAX. 54

6.1 Výsledky aproximativního algoritmu validovaného pomocí přesného algoritmu. 75

A.1 Kombinace proměnných ve skupině `days_month` a výkonnost základních modelů po jejich přidání. Písmeny v grafu jsou označeny potenciální chybové hlášky (jejich vysvětlení je v Příloze A.1), čísly je označeno 12 nejlepších modelů. Barevná škála reprezentuje monotónní, ale nelineární transformaci MSE, resp. MAE, tak, aby byla barevná škála reprezentativní. 91
A.2 Kombinace proměnných ve skupině `trend` a výkonnost základních modelů po jejich přidání. Písmeny v grafu jsou označeny potenciální chybové hlášky (jejich vysvětlení je v Příloze A.1), čísly je označeno 12 nejlepších modelů. Barevná škála reprezentuje monotónní, ale nelineární transformaci MSE, tak, aby byla barevná škála reprezentativní. Z grafu byly vypuštěny ty kombinace, kde podíl nenatrénovaných modelů kvůli singularitě převýšil 90 %. . . 92
A.3 Kombinace proměnných ve skupině `trend` a výkonnost základních modelů po jejich přidání. Písmeny v grafu jsou označeny potenciální chybové hlášky (jejich vysvětlení je v Příloze A.1), čísly je označeno 12 nejlepších modelů. Barevná škála reprezentuje monotónní, ale nelineární transformaci MAE, tak, aby byla barevná škála reprezentativní. Z grafu byly vypuštěny ty kombinace, kde podíl nenatrénovaných modelů kvůli singularitě převýšil 90 %. . . 93
A.4 Kombinace proměnných ve skupině `price` a výkonnost základních modelů po jejich přidání. Písmeny v grafu jsou označeny potenciální chybové hlášky (jejich vysvětlení je v Příloze A.1), čísly je označeno 12 nejlepších modelů. Barevná škála reprezentuje monotónní, ale nelineární transformaci MSE, resp. MAE, tak, aby byla barevná škála reprezentativní. 94
A.5 Kombinace proměnných ve skupině `important_event` a výkonnost základních modelů po jejich přidání. Písmeny v grafu jsou označeny potenciální chybové hlášky (jejich vysvětlení je v Příloze A.1), čísly je označeno 12 nejlepších modelů. Barevná škála reprezentuje monotónní, ale nelineární transformaci MSE, resp. MAE, tak, aby byla barevná škála reprezentativní. 95

A.6 Kombinace proměnných ve skupině `final` a výkonnost modelů se základními exogenními proměnnými po jejich přidání. Písmeny v grafu jsou označeny potenciální chybové hlášky (jejich vysvětlení je v Příloze A.1), čísly je označeno 12 nejlepších modelů. Barevná škála reprezentuje monotónní, ale nelineární transformaci MSE, resp. MAE, tak, aby byla barevná škála reprezentativní. 96
A.7 Průběh trénování jednoho XGBoost modelu (celkový a od iterace 8). Modrá přerušovaná vertikála značí přeučení na trénovací množinu, modrá plná vertikála značí přeučení na testovací množinu, a tím konec trénování. Zelená přerušovaná horizontála ukazuje nejlepší hodnotu zvolené `eval_metric` (RMSE) na testovací množině. 98

Seznam tabulek

3.1 Deset nejlepších modelů s původním pořadím v prvním sloupci. . 38
3.2 Popisné statistiky reziduí a p-hodnoty testů na jejich nekorelovanost, homoskedasticitu a normalitu. Za pořadím dle MSE následuje průměrná hodnota a směrodatná odchylka vektoru reziduí. V posledních třech sloupcích jsou po řadě hodnoty Ljung-Boxova, Breusch-Paganova a Shapiro-Wilkova testu, resp. jejich p-hodnoty. 39

4.1 Tabulka výsledků modelů XGBoost s výchozími hyperparametry a s různými skupinami proměnných (sloupec **Model**). Metriky popisující míru shody jsou počítány na cross-validační množině. . . . 43
4.2 Tabulka výsledků modelů XGBoost s výchozími hyperparametry. Metriky popisující míru shody jsou počítány pro `all` na cross-validační množině, pro `neoptimalizovaný model` na testovací množině. 44
4.3 Tabulka hyperparametrů před optimalizací. 44
4.4 Tabulka optimalizovaných hyperparametrů. 48

5.1 Tabulka výkonnosti modelů hyperparametrů. 53

6.1 Postupná kalkulace výpočetní složitosti přesného algoritmu. . . . 67
6.2 Postupná kalkulace výpočetní složitosti aproximativního algoritmu. 71
6.3 Tabulka demonstrující postup aproximativního algoritmu na trénovací části out-of-sample množiny. 76
6.4 Tabulka porovnávající váhy optimálního hybridního modelu a váhy tří alternativních přístupů spolu s výsledným MSE na trénovací části out-of-sample množiny. 76
6.5 Tabulka MSE jednotlivých modelů a hybridních modelů na trénovací a validační části out-of-sample dat. 77

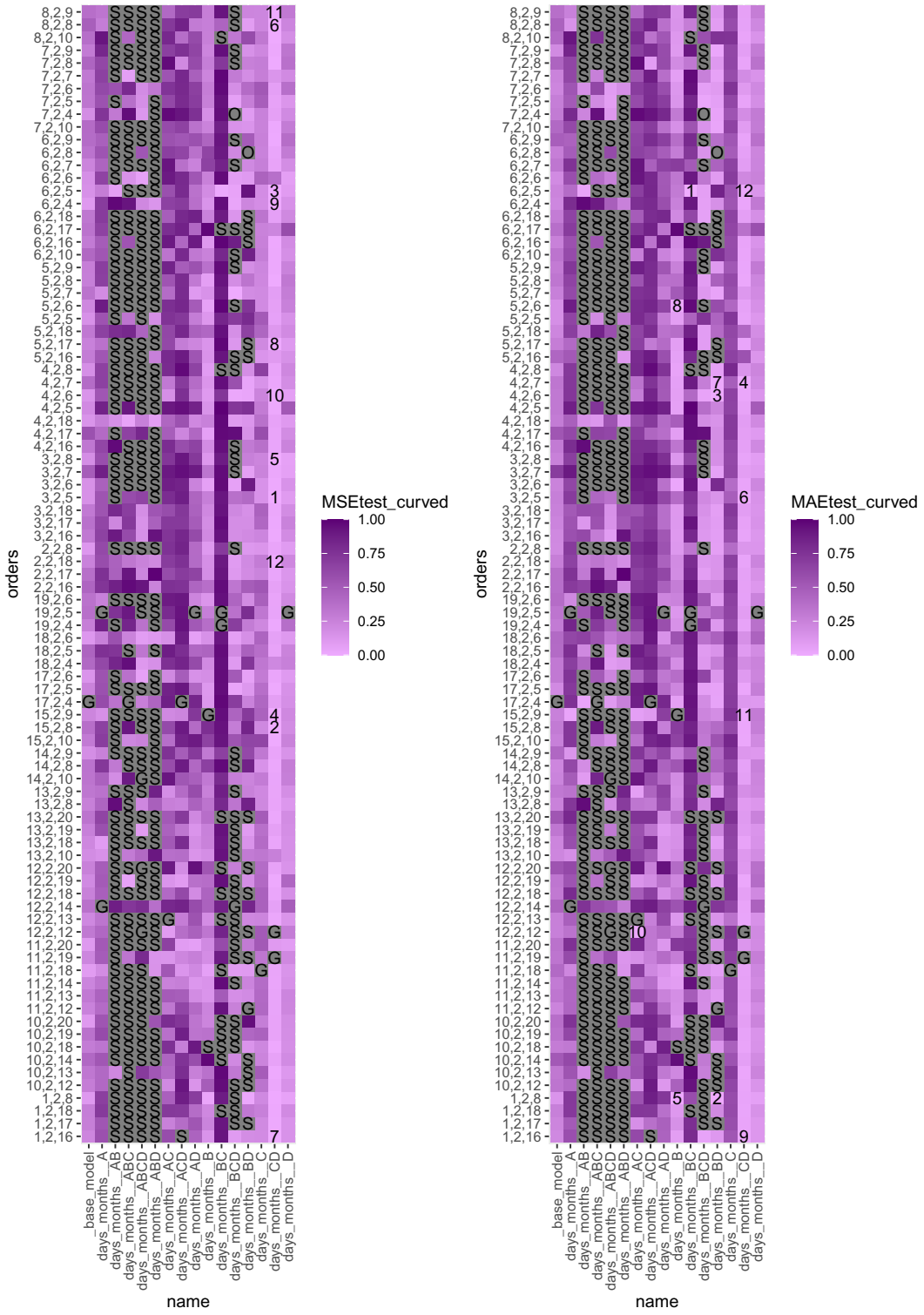
A. Přílohy

A.1 Seznam chyb při trénování ARIMAX modelu

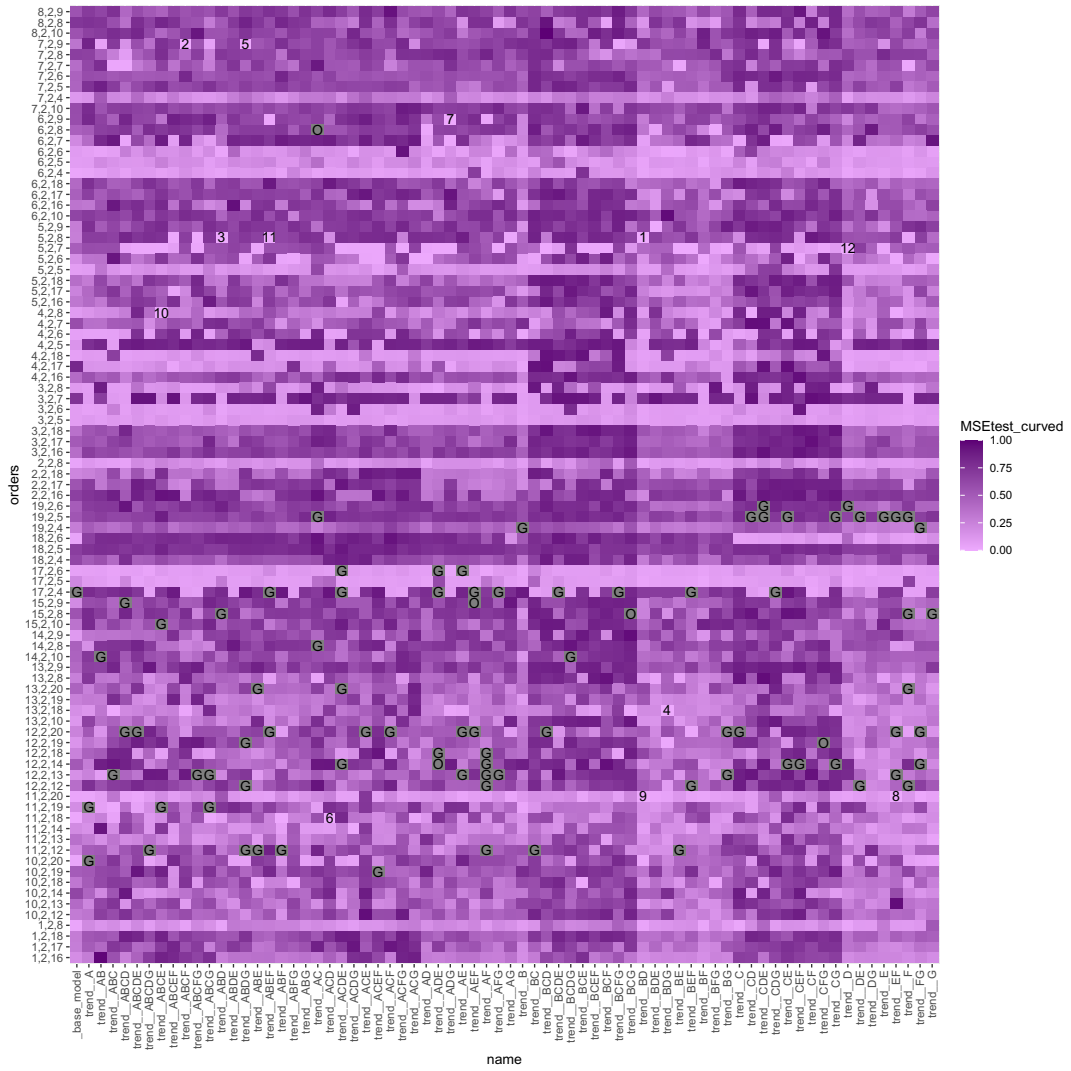
Toto je seznam chyb, které se vyskytly při trénování ARIMAX modelů. Použitý jazyk byl `R` ve verzi 3.5.3, knihovna `forecast` ve verzi 8.12 a metoda `ARIMAX`:

Kód	**Jméno**	**Popis**
S	Singularita	Zcela nebo téměř singulární regresní matice.
G	Gradient	MLE odhad blízko hranici přípustné množiny, aby byl model stacionární.
O	Nestacionarita	Charakteristický polynom modelu má zcela nebo téměř jednotkový kořen.

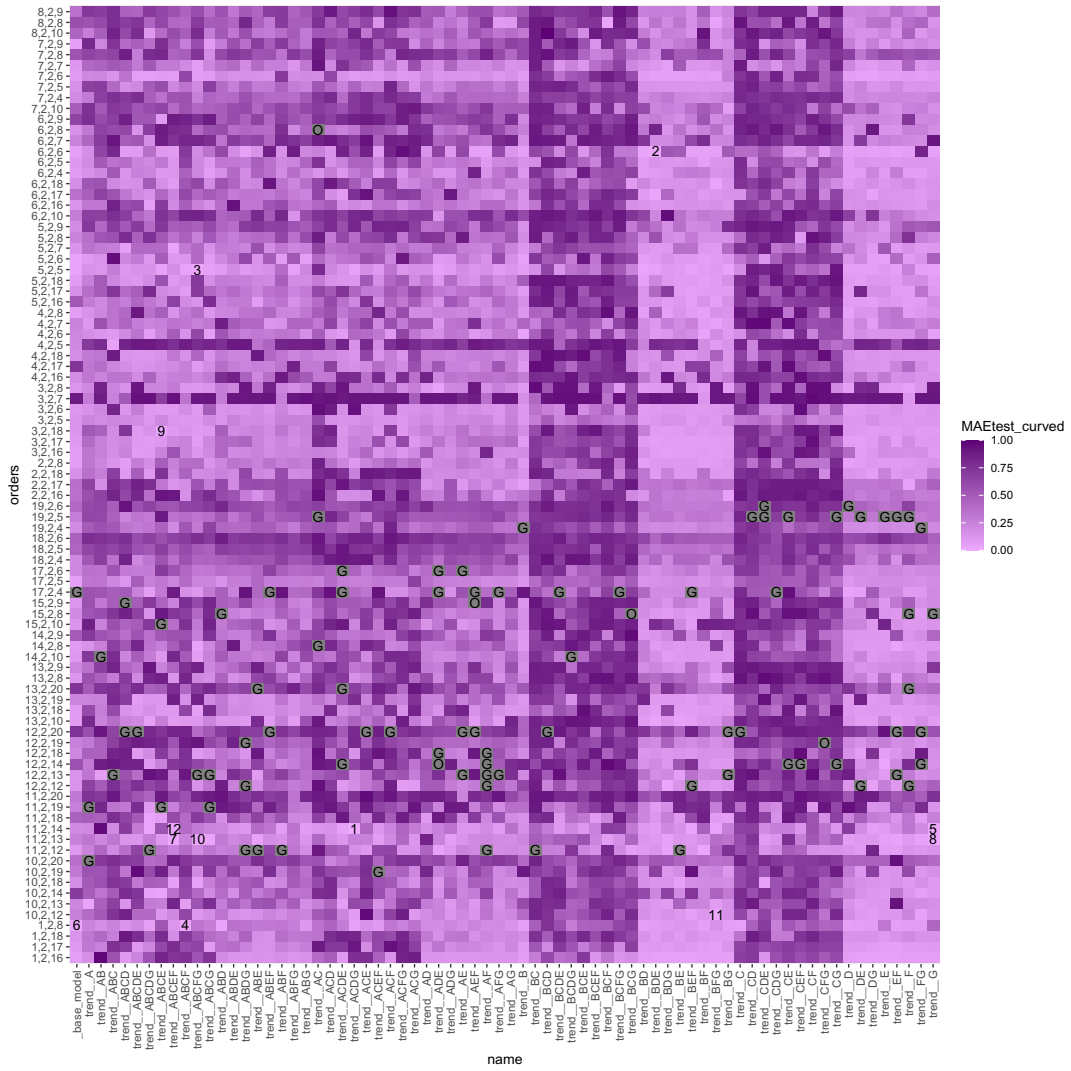
A.2 Analýza skupin exogenních proměnných



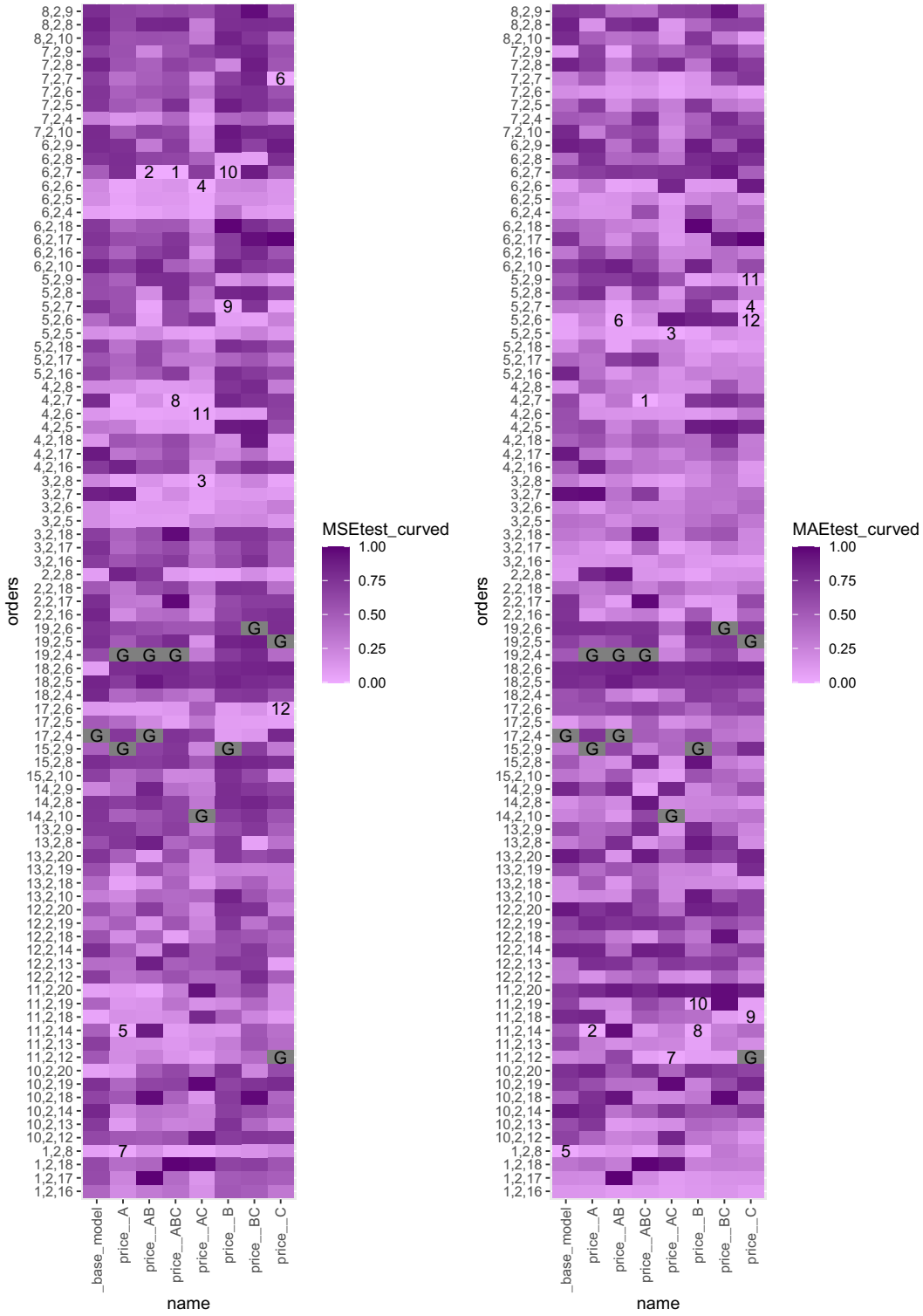
Obrázek A.1: Kombinace proměnných ve skupině `days_month` a výkonnost základních modelů po jejich přidání. Písmeny v grafu jsou označeny potenciální chybové hlášky (jejich vysvětlení je v Příloze A.1), čísly je označeno 12 nejlepších modelů. Barevná škála reprezentuje monotónní, ale nelineární transformaci MSE, resp. MAE, tak, aby byla barevná škála reprezentativní.



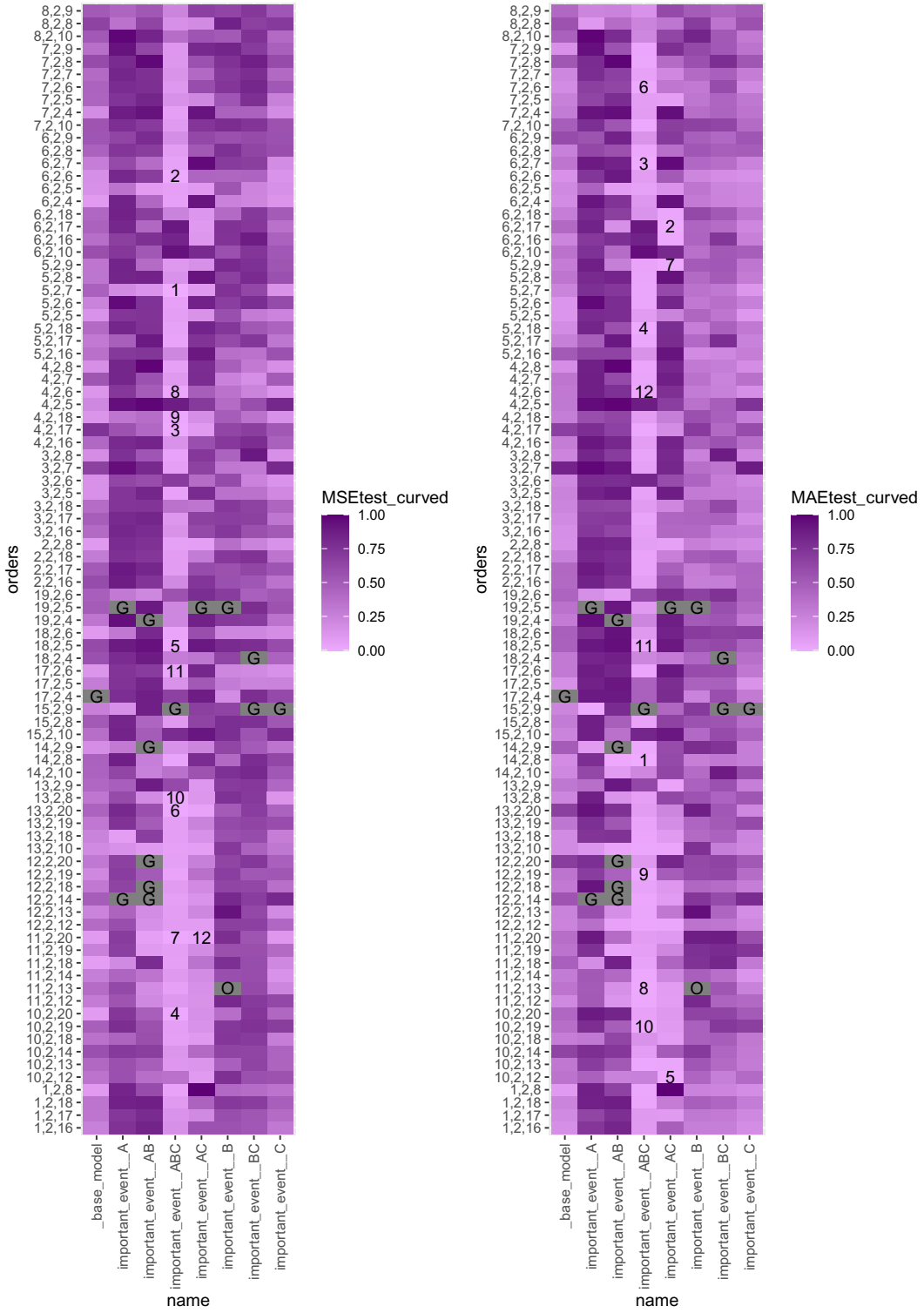
Obrázek A.2: Kombinace proměnných ve skupině `trend` a výkonnost základních modelů po jejich přidání. Písmeny v grafu jsou označeny potenciální chybové hlášky (jejich vysvětlení je v Příloze A.1), čísly je označeno 12 nejlepších modelů. Barevná škála reprezentuje monotónní, ale nelineární transformaci MSE, tak, aby byla barevná škála reprezentativní. Z grafu byly vypuštěny ty kombinace, kde podíl nenatrénovaných modelů kvůli singularitě převýšil 90 %.



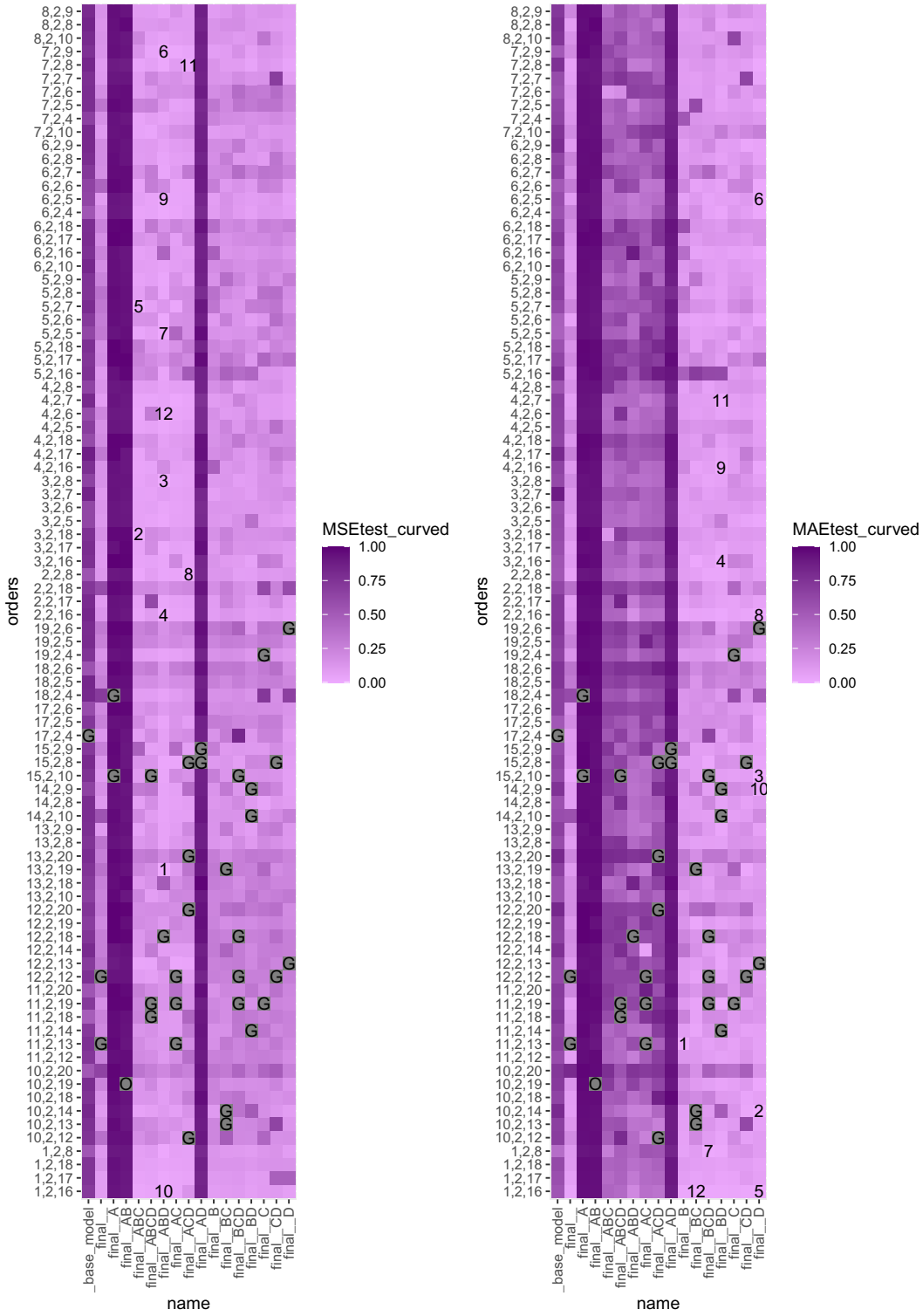
Obrázek A.3: Kombinace proměnných ve skupině `trend` a výkonnost základních modelů po jejich přidání. Písmeny v grafu jsou označeny potenciální chybové hlášky (jejich vysvětlení je v Příloze A.1), čísly je označeno 12 nejlepších modelů. Barevná škála reprezentuje monotónní, ale nelineární transformaci MAE, tak, aby byla barevná škála reprezentativní. Z grafu byly vypuštěny ty kombinace, kde podíl nenatrénovaných modelů kvůli singularitě převýšil 90 %.



Obrázek A.4: Kombinace proměnných ve skupině `price` a výkonnost základních modelů po jejich přidání. Písmeny v grafu jsou označeny potenciální chybové hlášky (jejich vysvětlení je v Příloze A.1), čísly je označeno 12 nejlepších modelů. Barevná škála reprezentuje monotónní, ale nelineární transformaci MSE, resp. MAE, tak, aby byla barevná škála reprezentativní.



Obrázek A.5: Kombinace proměnných ve skupině `important_event` a výkonnost základních modelů po jejich přidání. Písmeny v grafu jsou označeny potenciální chybové hlášky (jejich vysvětlení je v Příloze A.1), čísly je označeno 12 nejlepších modelů. Barevná škála reprezentuje monotónní, ale nelineární transformaci MSE, resp. MAE, tak, aby byla barevná škála reprezentativní.

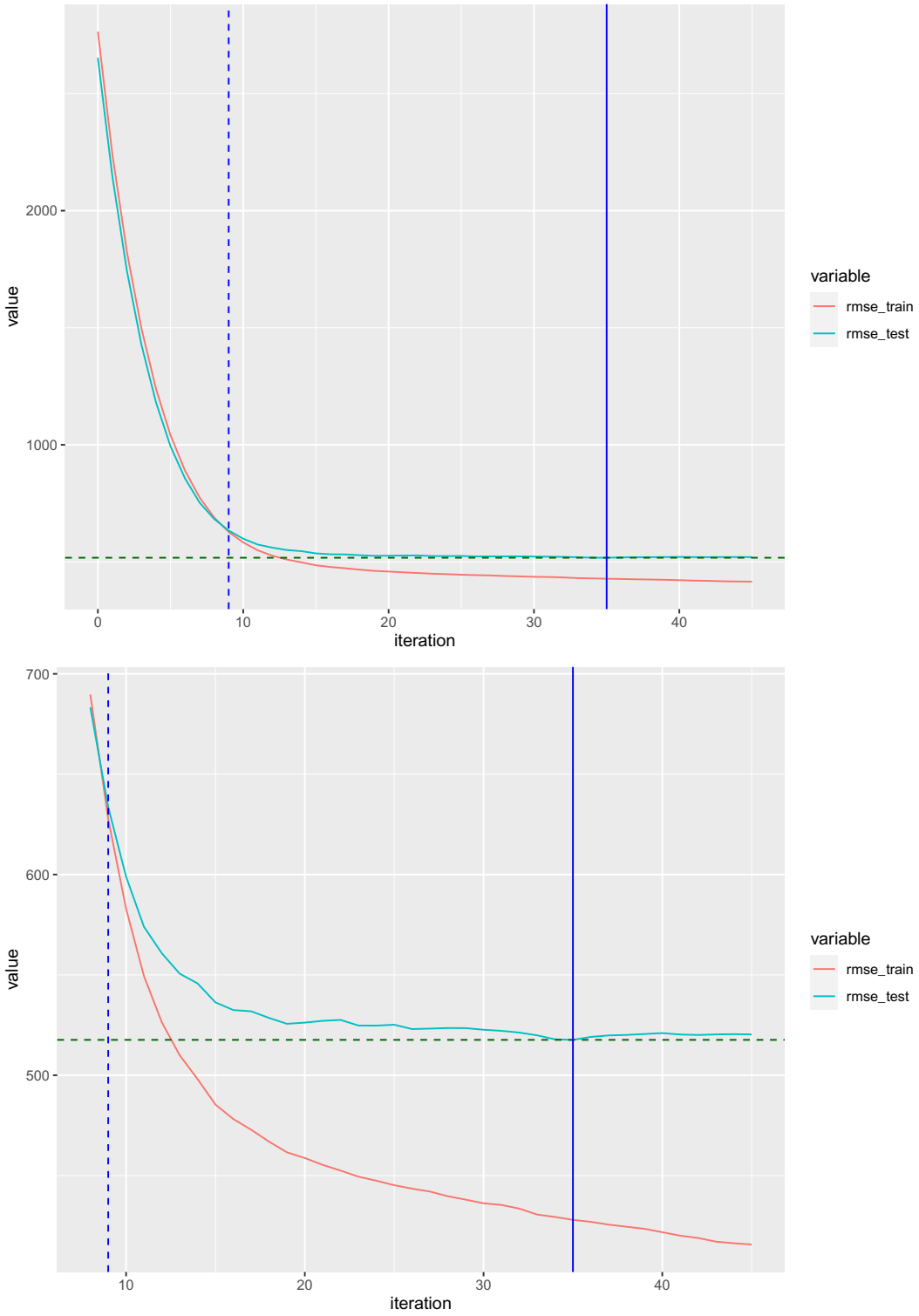


Obrázek A.6: Kombinace proměnných ve skupině `final` a výkonnost modelů se základními exogenními proměnnými po jejich přidání. Písmeny v grafu jsou označeny potenciální chybové hlášky (jejich vysvětlení je v Příloze A.1), čísly je označeno 12 nejlepších modelů. Barevná škála reprezentuje monotónní, ale nelineární transformaci MSE, resp. MAE, tak, aby byla barevná škála reprezentativní.

A.3 Průběh trénování XGBoost modelu

Zde je příklad průběhu trénování XGBoost modelu s předčasným zastavením:

```
[0]     train-rmse:2764.60303   test-rmse:2652.97437
[1]     train-rmse:2239.83838   test-rmse:2145.28955
[2]     train-rmse:1823.43994   test-rmse:1742.46362
[3]     train-rmse:1496.09180   test-rmse:1426.66382
[4]     train-rmse:1239.08972   test-rmse:1181.14465
[5]     train-rmse:1040.34314   test-rmse:993.31946
[6]     train-rmse:889.15430    test-rmse:855.84576
[7]     train-rmse:774.98993    test-rmse:753.04871
[8]     train-rmse:689.71582    test-rmse:683.30115
[9]     train-rmse:627.66467    test-rmse:633.62659
[10]    train-rmse:582.97168    test-rmse:598.95935
[11]    train-rmse:549.39465    test-rmse:573.98132
[12]    train-rmse:526.38440    test-rmse:560.77344
[13]    train-rmse:509.94806    test-rmse:550.60968
[14]    train-rmse:498.09708    test-rmse:545.73950
[15]    train-rmse:485.37857    test-rmse:536.30066
[16]    train-rmse:478.17456    test-rmse:532.47681
[17]    train-rmse:472.82944    test-rmse:531.85474
[18]    train-rmse:466.90378    test-rmse:528.54218
[19]    train-rmse:461.58118    test-rmse:525.62921
[20]    train-rmse:458.73117    test-rmse:526.21472
[21]    train-rmse:455.34549    test-rmse:527.11719
[22]    train-rmse:452.49576    test-rmse:527.56061
[23]    train-rmse:449.45065    test-rmse:524.77081
[24]    train-rmse:447.45715    test-rmse:524.73877
[25]    train-rmse:445.22238    test-rmse:525.17194
[26]    train-rmse:443.48480    test-rmse:523.00848
[27]    train-rmse:442.04654    test-rmse:523.23730
[28]    train-rmse:439.70566    test-rmse:523.51782
[29]    train-rmse:438.00516    test-rmse:523.47363
[30]    train-rmse:436.19205    test-rmse:522.67609
[31]    train-rmse:435.37048    test-rmse:522.12866
[32]    train-rmse:433.53049    test-rmse:521.26465
[33]    train-rmse:430.58667    test-rmse:519.88013
[34]    train-rmse:429.37106    test-rmse:517.88794
[35]    train-rmse:427.94272    test-rmse:517.62256
[36]    train-rmse:426.95743    test-rmse:519.09131
[37]    train-rmse:425.56024    test-rmse:519.83124
[38]    train-rmse:424.51977    test-rmse:520.08893
[39]    train-rmse:423.47894    test-rmse:520.53815
[40]    train-rmse:421.79507    test-rmse:521.00531
[41]    train-rmse:420.05505    test-rmse:520.28796
[42]    train-rmse:418.91779    test-rmse:520.06311
[43]    train-rmse:417.07538    test-rmse:520.34601
[44]    train-rmse:416.28433    test-rmse:520.47742
[45]    train-rmse:415.67151    test-rmse:520.31610
Stopping. Best iteration:
[35]    train-rmse:427.94272    test-rmse:517.62256
```



Obrázek A.7: Průběh trénování jednoho XGBoost modelu (celkový a od iterace 8). Modrá přerušovaná vertikála značí přeučení na trénovací množinu, modrá plná vertikála značí přeučení na testovací množinu, a tím konec trénování. Zelená přerušovaná horizontála ukazuje nejlepší hodnotu zvolené `eval_metric` (RMSE) na testovací množině.

V hodnotách a na Obrázku A.7, který je vizualizuje, je dobře vidět, že model je přeučený na trénovací množinu už od iterace 9 (menší chyba na trénovací než na

testovací množině), nicméně i takový dává čím dal tím lepší predikci až do iterace 35, po které již dochází k přeučení na testovací množinu. Neboť se výsledky po dobu 10 iterací (zvolený parametr `early_stopping_rounds`) nezlepšily proti iteraci 35, trénování končí a stromy z iterací 36-45 jsou z predikce vyřazeny.