TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Využití strojového učení pro odhad křivek přežití

Diplomová práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 3902T005 – Automatické řízení a inženýrská informatika
Autor práce: Bc. Tomáš Trdla
Vedoucí práce: Ing. Karel Paleček, Ph.D.

Liberec 2020

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání diplomové práce

Využití strojového učení pro odhad křivek přežití

Jméno a příjmení: **Bc. Tomáš Trdla**
Osobní číslo: M17000144
Studijní program: N2612 Elektrotechnika a informatika
Studijní obor: Automatické řízení a inženýrská informatika
Zadávající katedra: Ústav informačních technologií a elektroniky
Akademický rok: **2019/2020**

Zásady pro vypracování:

1. Seznamte se s postupy používanými ve zdravotnictví pro výpočty křivek přežití.
2. Vhodně zpracujte data a navrhněte algoritmy umělé inteligence pro výpočet křivek přežití.
3. Implementujte vybrané algoritmy a vyhodnoťte je na reálných datech pacientů.
4. Porovnejte dosažené výsledky a zhodnoťte výhody a nevýhody navrženého postupu oproti nejčastěji používaným metodám.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	40-50 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] SAINANI, Kristin, Joshua WALLACH a Michael MCAULIFFE. Statistics in Medicine. Stanford Online [online]. [cit. 2019-10-09]. Dostupné z: https://lagunita.stanford.edu/courses/Medicine/MedStats./Summer2015/about

[2] Bishop, C. Pattern Recognition and Machine Learning. 2006. ISBN 13: 978-038731073

[3] Goodfellow, Ian, Yoshua Bengio a Aaron Courville. Deep learning. The MIT Press 2016. ISBN 0262035618

Vedoucí práce:	Ing. Karel Paleček, Ph.D. Ústav informačních technologií a elektroniky
Datum zadání práce:	9. října 2019
Předpokládaný termín odevzdání:	18. května 2020

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

L.S.

Technická univerzita v Liberci
-18-

prof. Ing. Ondřej Novák, CSc.
vedoucí ústavu

V Liberci dne 18. října 2019

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Poděkování

Děkuji mé rodině a nejbližším za poskytnutí podmínek a pochopení při práci na tomto díle. Dále MUDr. Janu Vydrovi z ÚHKT za poskytnutí reálných dat a zájmu o toto téma. A v neposlední řadě vedoucímu práce Ing. Karlu Palečkovi, Ph.D. za nasměrování a vhodné připomínky.

Využití strojového učení pro odhad křivek přežití

Abstrakt

Tato práce se zabývá implementací a porovnáním vybraných algoritmů umělé inteligence na odhad křivek přežití. Výchozím modelem je nejpoužívanější Coxův model proporcionálních rizik, jehož nedostatek v podobě lineárních vztahů mezi kovariáty je základem úvah pro průzkum schopností a kvality nelineárních modelů z AI oblasti. Jako hodnotící kritéria jsou použity různé metody a přístupy, které dohromady dávají ucelenou představu o kvalitě modelu. Pro každý model je implementována metoda na popis vlivu vstupních proměnných na výsledné riziko selhání, aby byla zachována možnost zjištění tohoto vlivu, jako tomu je u Coxova modelu proporcionálních rizik. Veškeré testy jsou prováděny na reálných anonymizovaných datech z oddělení transplantací na Ústavu krevní hematologie a transfuze v Praze, kromě porovnání vlivu velikosti učícího datasetu, pro který bylo využito většího objemu dat. Kvůli povaze dat je k práci přistupováno i ze zdravotnického hlediska. Vzhledem k realistickému přístupu byl vytvořen vlastní preprocessor, který zohledňuje a řeší, že v reálných datech téměř vždy chybí některé údaje. Modely jsou otestovány na několika specifických cílech, které mohou být podstatné pro vývoj průběhu léčby. Výsledky prokazují rozdílnou kvalitu modelů na zkoumaných časech přežití a ovlivnění dané kvality velikostí učícího datasetu. Zároveň dokazují, že modely AI jsou schopny dosáhnout přesnějších výsledků než CoxPH model, avšak liší se při rozdílných cílech a datasetech, proto položily základ myšlence ensemble modelu, která je v této práci teoreticky popsána jako další možné řešení a výzkum.

Klíčová slova:

Křivky přežití, umělá inteligence, statistické testy, coxův model proporcionálních rizik, neuronová síť, rozhodovací stromové struktury, strojové učení

Using machine learning for survival analysis

Abstract

This work deals with the implementation and comparison of selected artificial intelligence algorithms for an estimation of survival curves. The default model is the most widely used Cox propotional hazard model, whose drawback of linear relationships between its covariates is the reasoning basis for exploring the capabilities and quality of nonlinear models from artificial intelligence domain. Various methods and approaches are used as evaluation criteria, which combination gives a comprehensive idea of quality of the model. For each model the method to describe the influence of input variables on resulting risk of failure is implemented in order to preserve the possibility of detecting this effect, as is the case with the Cox propotional hazard model. All tests are performed on real anonymized data from transplant department at the Institute of Blood Hematology and Transfusion in Prague, except the case of comparison of the effect of the size of the training dataset, for which a larger volume of data was used. Due to the nature of the data, the work is also approached from a medical point of view. In respect to the realistic approach a custom preprocessor has been created, which takes into account and solves, that some records are almost always missing in real data. Models are tested on several specific targets that may be essential for the development of the treatment. The results demonstrate different quality of models at the investigated survival times and detects influence on quality by given dataset size. At the same time the results prove that AI models are able to achieve more accurate results than CoxPH model, but differ in different goals and datasets, which laid the foundation for the ensemle model, which is theoretically described in this work as another possible solution and research.

Key words:

Survival curves, artificial intelligence, statictical testing, cox proportional hazard model, neural network, decision trees, machine learning

Obsah

1 Úvod 12

2 Teoretický rozbor 14

2.1 Křivky přežití 14

2.1.1 Kaplan-Meierova Estimace 15

2.1.2 Nelson Aalenova Estimace 16

2.1.3 Coxův model proporcionálních rizik 17

2.1.4 Využití křivek přežití ve zdravotnictví 19

2.2 Umělá inteligence a vybrané algoritmy 22

2.2.1 Neuronové sítě 23

2.2.2 Rozhodovací stromové struktury 26

2.3 Interpretovatelnost modelů a vliv proměnných 27

3 Praktická část 30

3.1 Analýza dat 31

3.1.1 Statistické vlastnosti 31

3.1.2 Preprocessing 32

3.2 Návrh algoritmů 34

3.2.1 Neuronové sítě 36

3.2.2 Decision tree modely 40

3.3 Interpretovatelnost modelů 43

4 Hodnotící metody 46

4.1 Způsoby hodnocení modelů 47

4.1.1 Log-rank test 47

4.1.2 Time-dependent Concordance index 48

4.1.3 Brier Score 49

4.1.4 Negative binomial log-likelihood 50

4.1.5 Váhová integrace 50

5 Výsledky 52

5.1 Porovnání všech modelů 52

5.1.1 Grafické výstupy 53

5.1.2 Rozbor výsledných grafů 56

5.2 Vliv proměnných na predikci a porovnání s CoxPH 58

5.3 Vliv velikosti datasetu na kvalitu modelů 59

5.4 Diskutování ensemble modelu 59

6 Závěr 61

Použitá literatura 64

Přílohy 67

A Výsledky ostatních cílů 67

B Grafy vlivu velikosti učícího datasetu 69

C Popis sloupců 71

D Části zdrojového kódu 74

Seznam obrázků

Obrázek 1: Kaplan-Meierova estimace 16
Obrázek 2: Nelson Aalenova estimace 17
Obrázek 3: Příklad použití CoxPH v praxi 21
Obrázek 4: Závislost kvality modelů na množství učících dat 23
Obrázek 5: Matematický model umělého neuronu 24
Obrázek 6: Příklad jednoduché stromové struktury 26
Obrázek 7: Interpretovatelnost modelů 28
Obrázek 8: Porovnání velikosti dat různých cílů po preprocessingu 32
Obrázek 9: Ukázka výstupních křivek přežití v porovnání s CoxPH 36
Obrázek 10: Porovnání SHAP vs regresní koeficienty u standardního CoxPH. 45
Obrázek 11: Histogram a Kernel Density Estimation 51
Obrázek 12: Porovnání Brier Score všech modelů 53
Obrázek 13: Porovnání NBLL všech modelů 54
Obrázek 14: Porovnání integrací křivek hodnotících metod 55
Obrázek 15: Porovnání Log-Rank testu všech modelů 55
Obrázek 16: Porovnání TDCI všech modelů 56
Obrázek 17: Porovnání délky učení modelů 56
Obrázek 18: Vliv vstupních proměnných GBSA modelu na riziko 58
Obrázek 19: Vliv vstupních proměnných CoxPH na riziko 59
Obrázek 20: Histogram časů událostí a cenzorování 60
Obrázek 21: Výsledky EOI „relaps“ s TTE „doba_do_relapsu“ 68
Obrázek 22: Vliv velikosti učícího datasetu – reálný dataset 69
Obrázek 23: Vliv velikosti učícího datasetu – umělý dataset 70

Seznam zdrojových kódů

Zdrojový kód 1: Konstruktor preprocessing třídy 74
Zdrojový kód 2: Vnitřní parametrizovatelná architektura neuronové sítě 75
Zdrojový kód 3: Average Negative Log Likelihood loss funkce (PyCox) 76
Zdrojový kód 4: Propojení základního kumulativního rizika s NN 76

Seznam zkratek

aGVHD	Akutní reakce štěpu proti hostiteli (z anglického Acute Graft-versus-Host Disease)
AI	Umělá inteligence (z anglického Artificial Intelligence)
AutoML	Automatic Machine Learning
BE	Breslow Estimator
BMI	Body mass index
BS	Brier Score
CoxPH	Coxův model proporcionálních rizik (z anglického Cox Proportional Hazards)
DRF	Distributed Random Forest
DT	Rozhodovací stromová struktura (z anglického Decision Trees)
EOI	Událost zájmu (a.j. Event of Interest)
GBM	Gradient Boosting Machine
GDPR	Obecné nařízení o ochraně osobních údajů (z anglického General Data Protection Regulation)
GLM	Generalized Linear Model
IPCW	Z anglického Inverse probability of censoring
KDE	Kernel Density Estimation
KME	Kaplan-Meierova Estimace
NBLL	Negative Binomial Log-Likelihood
NN	Umělé neuronové sítě (z anglického (Artificial) Neural Networks)
SHAP	SHapley Additive exPlanations
TDCI	Time-Dependent Concordance Index
TTE	Čas od vstupu subjektu do studie do jeho události/cenzorování (z anglického Time-To-Event). Stejný význam je „doba přežití“.
ÚHKT	Ústav hematologie a krevní transfuze v Praze

1 Úvod

V průběhu posledních několika let celým světem a velkým množstvím vědních, průmyslových i všedních oblastí našich životů hýbe rozmach umělé inteligence, ve které se díky pokroku v počítačové technice podařil průlom v konkrétní oblasti neuronových sítí [23].

Cílem této práce je prozkoumání schopností některých vybraných algoritmů umělé inteligence v oblasti odhadu křivek přežití. Pro experimentální účely byly poskytnuty reálné vzorky dat z oblasti zdravotnictví, konkrétně z databáze transplantací na Ústavu hematologie a krevní transfuze v Praze.

V praktickém světě, nejen zdravotnických výzkumů, jsou semi-parametrické modely jako Coxův model proporcionálních rizik využívány i ke zkoumání vlivu jednotlivých kovariátů vstupního vektoru na riziko výskytu cílové události. Avšak v mnoha aplikacích je tento přístup příliš zjednodušený, protože předpokládá, že riziková funkce je lineární. Tyto modely mají jistou podobnost ve vícenásobné lineární regresi, díky čemuž nejsou z matematické podstaty schopny zmapovat komplexní nelineární vazby v učících datech. Jedním z účelů této práce je věnovat se právě tomuto nedostatku pomocí modelů jako jsou např. neuronové sítě nebo stromové struktury, které jsou schopny tyto nelinearity pojmout.

Premisou této práce je, že pro vybrané modely není nutné připravovat a vybírat data podle různých parametrů a rozdělení pravděpodobnosti, jako je tomu v případě parametrických modelů. To znamená naučit model na velkém (dostupném) množství dat, resp. zkoumané populaci, a následně získat odhad křivky přežití nebo pravděpodobnosti výskytu události zájmu jednoho vybraného subjektu, která co nejpřesněji odpovídá realitě.

Pro účely hodnocení navržených algoritmů je využito několik různých algoritmů, jejichž kombinace zajišťuje ucelený přehled o kvalitě daného modelu. S využitím poskytnutých reálných dat jsou empiricky testovány varianty jednotlivých modelů ve snaze zhodnotit různé přístupy a dosáhnout lepšího výsledku, než jaký poskytuje současně nejpoužívanější metoda Coxova modelu proporcionálních rizik.

S narůstající popularitou umělé inteligence rostou i otázky „proč se model rozhodnul takto?“. Vzhledem k extrémně komplexní interní stavbě daných modelů je na ně nahlíženo jako na černou skříňku („*blackbox*“), což je v mnoha případech nedostačující. Praktická část této práce se proto věnuje i zahrnutí metod a algoritmů, které dokáží popsat vliv jednotlivých vstupních proměnných na výsledek modelu a docílit tak jisté podobnosti s využitím regresních koeficientů Coxova modelu proporcionálních rizik.

Z důvodu využití reálných dat, pocházejících z oblasti zdravotnictví, je na tuto práci nahlíženo částečně i z daného zdravotnického pohledu. Příslušné řešení a kódová implementace se tedy opírá o umožnění praktického a snadného využití.

2 Teoretický rozbor

Abychom se v úvahách a postupech dostali k jádru věci, je nejprve nutné se obeznámit s teoretickým pozadím problematiky a současně využívanou metodikou v obecné oblasti křivek přežití, jejich využití v oblasti zdravotnictví a následně pak s algoritmy umělé inteligence s teoretickým rozborem jejich praktického využití pro výpočet křivek přežití v parametrizovaném prostoru.

2.1 Křivky přežití

Křivky přežití jsou oblast statistiky a matematiky, která se věnuje analýze času (dále ve zkratce TTE z anglického Time to Event), kdy (resp. za který) se objeví událost, na kterou se daná analýza soustředí, pod anglickým názvem tzv. "Event of interest" (dále EOI) [25]. První analýzy křivek přežití našly využití ve zdravotnictví pro účely měření délky života určité populace lidí podstupujících léčbu. Postupem času se však začaly uplatňovat i obecněji a to například pro predikci změn počtu zaměstnanců a zákazníků firem, estimace životnosti strojů nejen v průmyslu, selhání senzorů a další.

Analogicky se využívá terminologie přejatá z původního účelu zdravotnictví, tedy událost nebo bod zrození (v anglickém jazyce *starting point* nebo *birth event*) a úmrtí (*ending point* nebo *death event*) [25], kdy například jako bod zrození může být považován čas, kdy zákazník začne svoji spolupráci s firmou a bod úmrtí chvíle, kdy danou spolupráci ukončí.

Křivky přežití jsou reprezentovány funkcí (1), nazývající se funkce přežití (*Survival Function*) $S(t)$, která definuje pravděpodobnost P s jakou se EOI **ne**objevil za čas t. Lze ji také interpretovat, jako pravděpodobnost přežití po čas t. T zde představuje náhodně vybranou proměnnou, vybranou ze zkoumané populace a její význam odpovídá času, za který se objeví EOI (tedy TTE). $S(t)$ se pohybuje mezi $S(0) = 1$ a $S(\infty) = 0$, přičemž 1 odpovídá 100% a jedná se o klesající funkci resp. ne-stoupající.

$$S(t) = P(T \geq t) \tag{1}$$

Další funkcí, kterou je nezbytné definovat, je tzv. riziková funkce (*Hazard Function*) nebo se jí také říká *Intensity Function*. Její význam je definován jako pravděpodobnost, že se u zkoumaného subjektu objeví EOI v nějakém krátkém

intervalu a to za předpokladu, že od počátku sledování se dosud neobjevil, avšak je nutné upozornit, že jejím výsledkem není samotná pravděpodobnost [25]. Rovnice (2) představuje velikost přírůstku, který odpovídá změně velikosti pravděpodobnosti v časovém rozmezí T a $T + dt$. Důvod limity je zde objasněn tím, že chceme zjistit risk, že se EOI objeví v nekonečně krátkém časovém okamžiku nebo-li přejatým slovem instantně.

$$\lambda(t) = \lim_{dt \to 0} \frac{P(t \leq T < t + dt | T \geq t)}{dt} \tag{2}$$

Dále je třeba definovat význam podstatné terminologie používané v kontextu analýzy přežití:

parametrické modely - Nebo také parametrizace znamená, že na rozdíl od neparametrického přístupu doba přežití sleduje konkrétní rozdělení pravděpodobnosti. Cílem je odhad jeho parametrů. Tyto modely jsou nezbytné pro extrapolaci výsledků za hranice dostupných dat.

cenzorování - Případ, kdy se u některých subjektů (prozatím) neobjevila EIO v průběhu zkoumané periody. Dále nastává, pokud subjekt nelze pozorovat v průběhu této periody, nebo u subjektu nastala jiná událost, která vylučuje výskyt prvotní EOI. V grafech bývá označeno vertikální čárkou v daném čase, zatímco výskyt události změnou hodnoty na ose y.

kovariáty – Vysvětlující (nezávislé) proměnné nebo-li hodnoty vstupující do modelu, které přímo náleží k danému subjektu.

regresní modely - Zde se k využívaným cenzorovaným proměnným délky přežití přidávají navíc i další data (kovariáty) jako například věk, pohlaví, výplata atd. daných subjektů, které mohou mít nějakým způsobem vliv na výskyt události či přežití.

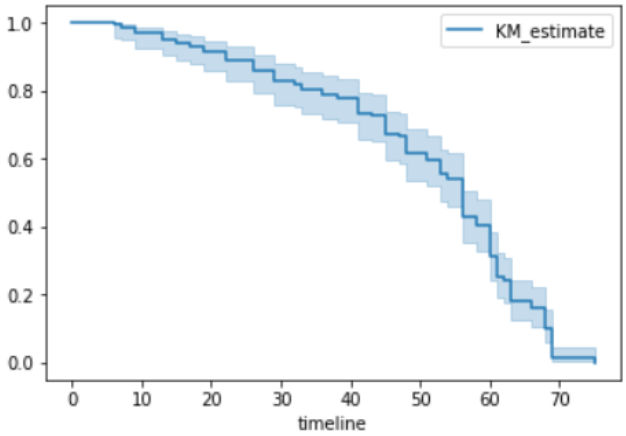
2.1.1 Kaplan-Meierova Estimace

Kaplan-Meierova estimace (dále KME) je pravděpodobně nejznámější a nejpoužívanější metodou pro neparametrický odhad funkce přežití. Křivka zde nabývá hodnot $< 0; 1 >$ na ose y a je klesající s počátkem v 1, což odpovídá 100% přežití v čase $t = 0$ (viz Obrázek 1).

$$S(t) = \prod_{t_i t} \frac{n_i - d_i}{n_i} \tag{3}$$

kde n_ipředstavuje počet ohrožených subjektů před časem t a d_ipředstavuje počet EOI v čase t.

Doba přežití (TTE) je zde definována jako doba/čas mezi bodem počátku a bodem, kdy se objeví EOI. Kaplan-Meierova křivka přežití je pak křivka, která znázorňuje pravděpodobnost přežití (osa y) pro danou délku intervalu (osa x). Pro každý časový okamžik t je pravděpodobnost počítána ze subjektů, které jsou v daném t "naživu" podělena množstvím riskujících subjektů (3).



Obrázek 1: Kaplan-Meierova estimace [1]

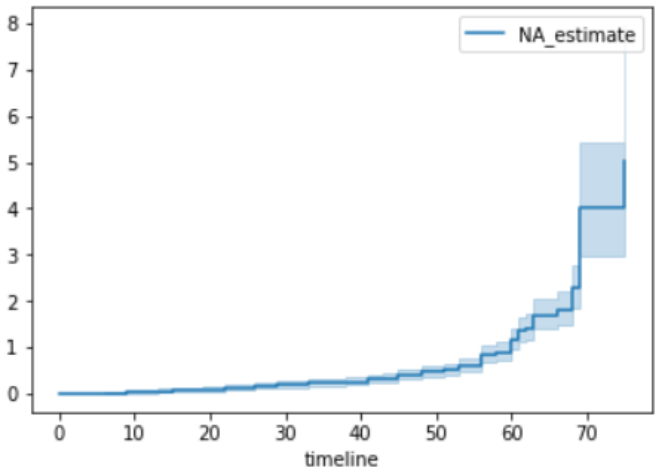
2.1.2 Nelson Aalenova Estimace

Podobně jako u KME se jedná o neparametrickou estimaci, ale zásadním rozdílem je zde výpočet pravděpodobnosti a následně i tvar výsledné křivky. Pravděpodobnost v čase t je totiž dána počtem "úmrtí" subjektů od počátku zkoumaného intervalu až po t poděleno množstvím riskujících subjektů (4). Křivka je tedy stoupající viz obrázek 2.

$$H(t) = \sum_{t_i \leq t} \frac{d_i}{n_i} \tag{4}$$

[1] Převzato z: ZAHID, Taimur. Survival Analysis — Part A. *Towards Data Science* [online]. 18.3.2019 [cit. 2020-05-17]. Dostupné z: https://towardsdatascience.com/survival-analysis-part-a-70213df21c2e

kde n_ipředstavuje počet ohrožených subjektů před časem t a d_ipředstavuje počet EOI v čase t.



Obrázek 2: Nelson Aalenova estimace [2]

2.1.3 Coxův model proporcionálních rizik

Výraz "proporcionální riziko" odkazuje na předpoklad konstantního vztahu mezi vysvětlujícími závislými proměnnými a regresními koeficienty. To v podstatě znamená, že riziková funkce dvou subjektů v náhodném bodě času t je proporcionální. Model proporcionálního rizika pak vyjadřuje, že jednotlivé kovariáty působí na rizikovou funkci multiplikativně. [19]

Tento model spadá do kategorie regresních modelů, bere v potaz vliv několika proměnných v čase a zkoumá jejich vztah k rozdělení přežití. Jistou podobnost lze nalézt ve vícenásobné regresní analýze, avšak rozdílem je, že roli závislé proměnné v čase t zde hraje riziková funkce.

$$\lambda(t|x) = \lambda_0(t) \cdot \exp\left(\sum_{i=1}^{n} b_i x_i\right) \tag{5}$$

[2] Převzato z: ZAHID, Taimur. Survival Analysis — Part A. *Towards Data Science* [online]. 18.3.2019 [cit. 2020-05-17]. Dostupné z: https://towardsdatascience.com/survival-analysis-part-a-70213df21c2e

Konkrétní riziková funkce je zde definována rovnicí (5) [19], kde b_i jsou regresní koeficienty a λ_0 základní riziková funkce, která je společná pro všechny subjekty. Hodnota základní rizikové funkce odpovídá pravděpodobnosti zaznamenání EOI v čase t v případě, kdy se všechny kovariáty rovnají 0. Je to také jediná proměnná závislá na čase, ale její přesná podoba není v modelu specifikována[29]

Proměnná x_i odpovídá členu vektoru, představující tzv. vysvětlující nezávislé proměnné, což mohou být například věk, plat atd. subjektu. Hodnoty v regresních koeficientech b_i ovlivňují základní rizikovou funkci λ_0 a každý člen i ji buď zvyšuje ($b_i > 1$) nebo snižuje ($b_i < 1$) její hodnotu. Koeficienty, které jsou rovny 1, nemají na riziko žádný vliv. Přesněji řečeno koeficienty b_i udávají změnu v riziku, že nastane EOI, která je spojená se změnou daného kovariátu x_i [9].

Některé kovariáty se nemohou řídit podle předpokladu proporcionálního rizika. Tyto kovariáty se sice mohou zahrnout do modelu, avšak nemají žádný vliv (regresní parametry) na jeho estimaci. Tento proces se nazývá stratifikace (počeštěný výraz z a.j. *stratification*, což lze přeložit jako rozvrstvení) [27].

Protože základní riziková funkce nemusí být specifikována, tak se tento model nazývá jako částečně parametrický (semi-parametric), ale zároveň pak předpokládá, že změna λ_0 je proporcionální k velikosti změny periody. Pokud rizikovou funkci vyjádříme jako:

$$\lambda_i(t) = \lambda(t|Z_i) = \lambda_0(t) \cdot \exp\left(\beta_1 Z_{1i} + \cdots + \beta_p Z_{pi}\right) \tag{6}$$

kde vektor kovariátů $Z_i = \left(Z_{1i},\ Z_{2i}, \ldots, Z_{pi}\right)$ a následně budeme uvažovat dvě skupiny pro $Z = 1$ a $Z = 0$, pak

$$\begin{aligned} \lambda_1(t) = \lambda(t|Z = 1) &= \lambda_0(t) \cdot \exp(\beta Z) \\ &= \lambda_0(t) \cdot \exp(\beta) \end{aligned} \tag{7}$$

což implikuje, že $exp(\beta)$ je konstantní a tudíž riziko obou skupin je proporcionální:

$$\frac{\lambda_1(t)}{\lambda_0(t)} = e^{\beta} \tag{8}$$

a tedy

$$\log\left\{\frac{\lambda_i(t)}{\lambda_0(t)}\right\} = \beta_1 Z_{1i} + \cdots + \beta_p Z_{pi} \tag{9}$$

dokazuje, že Coxův proporcionální model je lineární model logaritmu rizika.

Pro účely hledání/estimaci regresních koeficientů navrhl autor David Cox tzv. *metodu parciální věrohodnosti* (v a.j. partial likelihood method) (10) [19], která nahrazuje výchozí funkci věrohodnosti při výpočtu regresního modelu [9]. Tato metoda totiž závisí pouze na vektoru regresních koeficientů a není závislá na specifikování základní rizikové funkce b_0.

$$\prod_{i=1}^{n}\left[\frac{exp(\beta' Z_i)}{\sum_{j\epsilon R(X_i)}\exp(\beta' Z_j)}\right]^{\delta_i} \tag{10}$$

kde X_i je časově proměnná událost s možností cenzorace, δ_i je indikátor zaznamenání události nebo cenzorování (1=událost, 0=cenzorace), Z_i reprezentuje vektor kovariátů.

2.1.4 Využití křivek přežití ve zdravotnictví

Hlavními důvody pro použití křivek přežití ve zdravotnictví jsou zdravotně-ekonomické hodnocení spolu s hodnocením výsledků studií například nově vyvinuté léčby. Na stejnou úroveň důležitosti lze postavit i to, že modely jsou také využívány pro zkoumání kovariátů, který má nejsilnější dopad na dobu přežití pacienta, či výskyt životně podstatné události, jako je například relaps při léčbě rakoviny. Od těchto poznatků se následně mohou odvíjet úpravy léčby, návrh medikamentů a budoucí postup v průběhu léčby pacienta.[6]

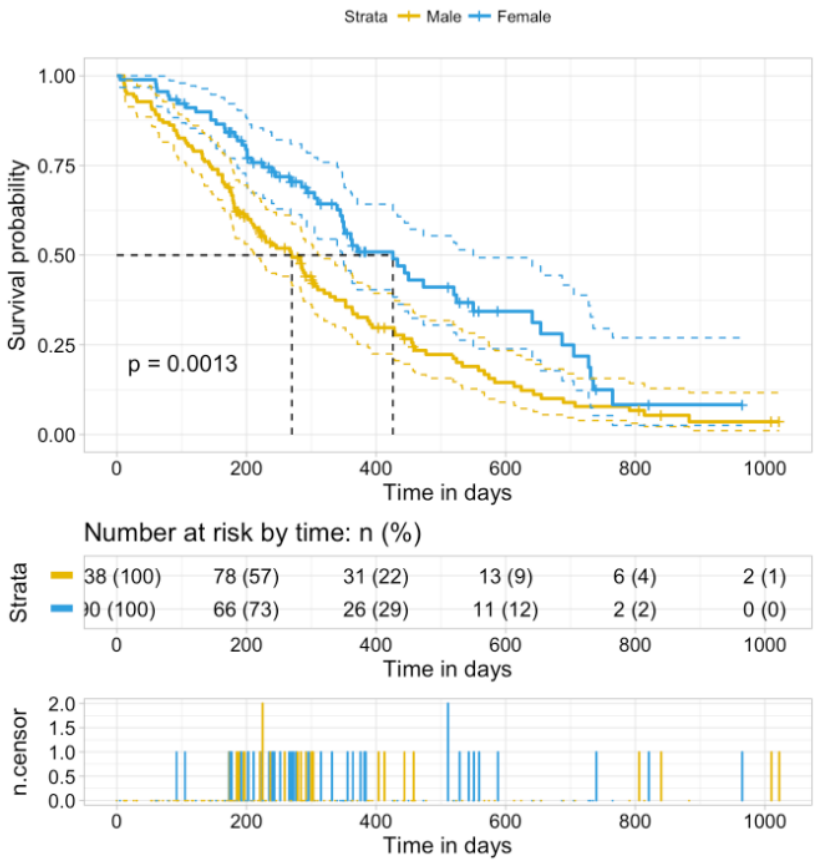
Pokud jde pouze o počet přeživších pacientů podstupující nějakou studii, tak lze využít neparametrické modely, kde nejčastější a nejsnadnější volbou je KME. V opačném případě je jednoznačnou volbou Coxův model proporcionálních rizik (dále jako zkratka CoxPH z a.j. Cox Proportional Hazards).

U Coxova modelu je významnou výhodou, že lze zjistit hodnoty regresních koeficientů každého kovariátu a tím získat užitečnou informaci o vlivu daného

kovariátu na změnu pravděpodobnosti rizika. Nicméně i přes jeho jednoduchost použití a snadnou interpretovatelnost jeho výsledků se v případě aplikování na data, která nesplňují jeho předpoklady, docílí zkreslených výsledků a tedy i jejich nevhodné interpretace.

Pro porovnání výsledků dvou a více studií je využívána statistická metoda nazývaná *Log-rank test* [5]. Nulovou hypotézou je zde předpoklad, že mezi zkoumanými populacemi pravděpodobnosti neexistuje žádný rozdíl ve všech TTE bodech. Analýza je založena porovnávání TTE, ve kterých se objevují EOI. Stupeň volnosti je odvozen od počtu studií respektive populací -1 i.e. $n = N - 1$, kde N je počet populací a n počet stupňů volnosti. Výsledkem testu je stejně jako u ostatních testů hypotéz p-hodnota (a.j. p-value), kdy $p \to 0$ značí významný rozdíl a $p \to 1$ naopak. Sice existují další testy, ale tento je preferován z důvodu snadného rozšíření na větší množství populací. Vzhledem k tomu, že tento test je čistě testem významnosti, tak nedokáže poskytnout odhad rozdílu nebo interval spolehlivosti mezi populacemi, pro takový případ se musejí zkoumat data samotná [5].

Pro práci s analýzou přežití je většinou využíván software R, který obsahuje balíčky přímo s příslušnými metodami. Výsledný graf pak může vypadat jako na ukázkovém obrázku 12, kde je součástí grafu i zmíněný *Log-rank test*, konfidenční intervaly, počty cenzorovaných subjektů v daném čase atd.



Obrázek 3: Příklad použití CoxPH v praxi[3]

Konkrétním praktickým příkladem aplikovatelnosti analýzy přežití a jejích statistických metod v oblasti zdravotnictví může být například studie [6] z roku 2018, která zkoumá riziko infarktu u náhodně vybraných 250 pacientů. Ve studii jsou využity proměnné jako věk, pohlaví, tepová frekvence, systolický krevní tlak a body-mass-index (dále BMI). Studie využívá Kaplan-Meierovu estimaci, která dokázala, že není žádný statisticky významný rozdíl v čase přežití mezi muži a ženami. Výsledky Coxovy regrese dále prokázaly, stejně jako KME, že pohlaví nemá na úmrtí pacienta významný vliv, avšak vliv ostatních kovariátů již statisticky významný je. Studie dále prokázala, že přírůstek v systolickém tlaku a BMI snižuje riziko úmrtí, zároveň pak, že toto riziko se zvyšuje s narůstajícím věkem, stejně tak jako s narůstající tepovou frekvencí.

[3] Převzato z: *Survival Analysis Basics* [online]. [cit. 2020-05-17]. Dostupné z: http://www.sthda.com/english/wiki/survival-analysis-basics

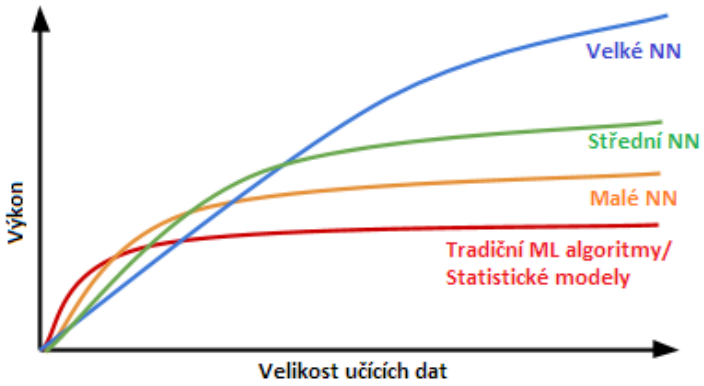
2.2 Umělá inteligence a vybrané algoritmy

První zmínky, představy a mýty o umělé inteligenci (dále AI z anglického Artificial Intelligence) sahají až do dob starověku, kdy si filozofové představovali autonomní a mechanické lidi, podobně jako my v dnešní době ve sci-fi kinematografii [24]. V roce 1921 představil český spisovatel Karel Čapek první světové označení autonomní entity slovem "robot" ve svém díle R.U.R.. V roce 1950 pak britský matematik, logik a kryptoanalytik Alan Mathison Turing, který proslul svými zásluhami o dešifrování nacistických tajných kódů enigmy, navrhl v práci *Computing machinery and intelligence* [28] myšlenku "Mohou stroje myslet?", což dalo základ tzv. *Turingově testu*. Tento test se poté stal nedílnou součástí filozofie AI, jež diskutuje inteligenci, vědomí a schopnosti strojů [24].

V dekádě od roku 2010 zaznamenalo odvětví AI obrovský vzestup popularity, neboť pokrok v tranzistorové technice umožnil práci s tzv. big data, což v minulém století nebylo možné prakticky realizovat. V roce 2012 se podařilo natrénovat rozsáhlou neuronovou síť na 10-ti milionech obrázcích z youtube videí, která rozpoznala kočku bez jakékoliv informace, že se jedná o kočku. 2014 pak první bezpilotní auto na světě prošlo v Americkém státě Nevada testem samořiditelnosti [24].

V roce 2017 označil pro veřejnost DataRobot CEO Jeremy Achin umělou inteligenci jako počítačový systém, který je schopen provádět běžné práce, které vyžadují lidskou inteligenci. Mnoho algoritmů AI jsou postaveny na základech strojového učení, deep learningu (viz Obrázek 4) nebo pouze sadou pravidel.

Ačkoliv se ve spojení s AI většinou mluví o jisté inteligenci a schopnosti rozhodovat se jako vědomá entita, tak se nesmí zapomínat, že samotné algoritmy mají základ ve statistice a pracují na prosté myšlence, kdy se danému algoritmu ukáže velké množství dat. Tento algoritmus v datech poté hledá spojitosti a analyzuje vazby jednotlivých proměnných. Proto se v této práci nahlíží na zvolené algoritmy jako na jiné, či kvalitnější statistické metody oproti běžně používaným. Především pak jako na regresní a klasifikační.



Obrázek 4: Závislost kvality modelů na množství učících dat [4]

2.2.1 Neuronové sítě

Právě neuronové sítě (dále jako NN z anglického Artificial Neural Networks) stojí za vzestupem popularity AI v poslední dekádě. Jedná se o modely spadající do kategorie strojového učení, které principielně napodobují základní biologické procesy ve struktuře mozku. NN se na první pohled mohou jevit jako black-box, nicméně jejich cíl je stejný jako u ostatních modelů - vytvářet co nejpřesnější odhad nebo-li predikci.[8]

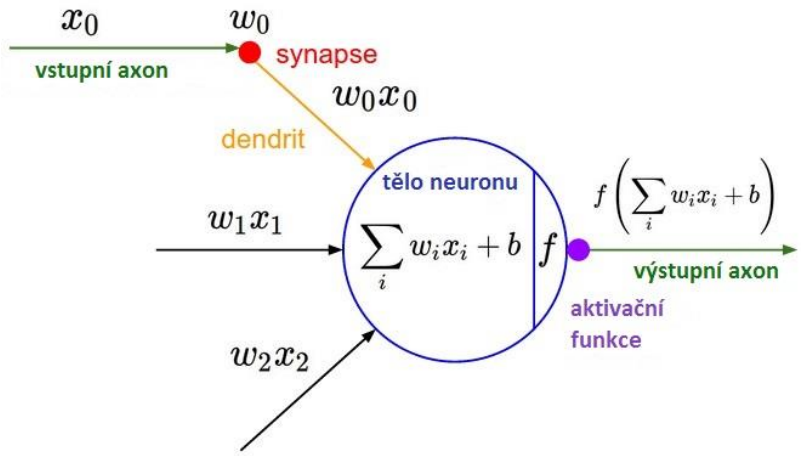
Lidský mozek obsahuje přibližně 50–100 miliard (tedy 10^{11}) neuronů, z nichž asi 10 % (10^{10}) jsou pyramidové buňky v mozkové kůře. Mezi nervovými buňkami existuje až 10^{15} synaptických spojení [12]. Neuron sám o sobě není moc silným nástrojem, avšak jeho síla spočívá v propojení s ostatními neurony do jednoho celku. V mozku se pak jedná o spojení s až tisíci jinými neurony. Kognitivní aktivity v mozku jsou vybuzovány signálem sestávajícím se z elektrických impulsů mezi jednotlivými neurony.

Pro adaptaci těchto poznatků neurovědy byl v roce 1943 McCullochem (neurovědec) a Pittsem (logik a matematik) vytvořen model umělého neuronu [22] (viz Obrázek 5), jehož chování lze interpretovat rovnicí (11), kde x_i reprezentuje jednotlivé vstupy (axony) neuronu, w_i jsou váhy jednotlivých synapsí, w_0 absolutní člen nebo-li

[4] Převzato z: WENG, Lilian. *An Overview of Deep Learning for Curious People* [online]. [cit. 2020-05-19]. Dostupné z: https://lilianweng.github.io/lil-log/2017/06/21/an-overview-of-deep-learning.html

bias, f aktivační funkce, která do modelu vnáší nelinearitu a y je výstup daného neuronu. Tento model je nazýván jako *Perceptron* a je to nejjednodušší forma umělé dopředné neuronové sítě. Pokud je jako aktivační funkce použit *sigmoid*, tak získáme prostou logistickou regresi.

$$y = f(z), \text{ kde } z = \sum_{i=1}^{n} w_i x_i + w_0 \tag{11}$$



Obrázek 5: Matematický model umělého neuronu[5]

Rozšířením tohoto modelu je paralelní pospojování N perceptronů, které vytvoří vrstvu (v a.j. layer). Tyto vrstvy se poté dále spojují jedna za druhou s libovolným počtem vnitřních neuronů a tím vzniká tzv. *Deep Neural Network* model (v č.j. občas používaný výraz hluboké neuronové sítě) [8] a lze ho reprezentovat pomocí (12). Kde nově doplněné proměnné mají následující význam: m představuje počet neuronů ve vrstvě l, váhy jsou reprezentovány jako $w_{i,k}^{l}$ k-tého neuronu v $(l-1)$ vrstvě pro i-tý neuron ve vrstvě l.

$$y_i^l = f(z_i^l), \text{ kde } z_i^l = \sum_{k=1}^{m^{(l-1)}} w_{i,k}^l y_k^{l-1} + w_{i,0}^l \tag{12}$$

Rovnici lze převést do kompaktnější více-dimenzionální formy, která mapuje D vstupů na C výstupů:

[5] Převzato z: *Artificial Neural Networks* [online]. [cit. 2020-05-17]. Dostupné z: https://wiki.tum.de/display/lfdv/Artificial+Neural+Networks

$$y(.,w): R^D \rightarrow R^C, x \rightarrow y(x,w) \quad (13)$$

Tento model se rozděluje na vstupní vrstvu (*input layer*), skryté vrstvy (*hidden layers*) a výstupní vrstvu (*output layer*). Emeprickými metodami napříč data-science komunitou, ale i logickým odvozením bylo zjištěno, že velký vliv na výsledky má počet vrstev této sítě. Díky jednotlivým vrstvám totiž model dokáže rozpoznat komplexnější a abstraktnější vztahy v datech a především jak jsou jednotlivé vstupy ovlivňovány ostatními vstupními proměnnými z pohledu výsledku. Teoreticky je sice možné navrhnout jednu vrstvu, která by tyto vztahy dokázala identifikovat také, ale správně navrhnout počet jejích neuronů je oproti přidání počtu vrstev příliš náročné. V tomto kontextu návrhu velikosti modelu totiž neexistuje žádné tzv. pravidlo pravé ruky, které by určilo, že při N vstupech bude třeba X neuronů a Y vrstev.

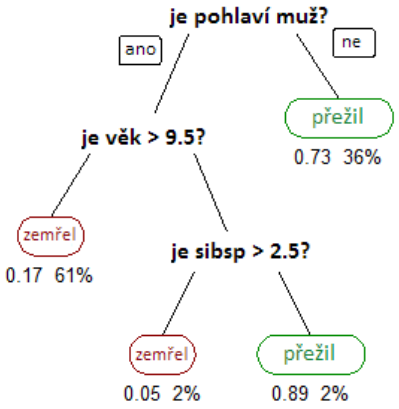
Další nedílnou součástí teoretického rozboru NN jsou aktivační funkce. Pomocí těchto funkcí je do modelu vnášena schopnost rozpoznávat nelineární vztahy a bez nich by i sebevětší NN byla stále lineární. Mezi nejpoužívanější patří následující: Treshhold, Sigmoid, Hyperbolic tangent, ReLU (Rectified Linear Unit). Každá z těchto funkcí umožňuje získat jiný typ výsledku - např. Sigmoid je vhodný pro odhad pravděpodobnosti a klasifikaci, zatímco ReLU je vhodná pro regresní úlohy. Tento fakt je daný jejich matematickou povahou. [8]

O způsobu učení NN lze prohlásit, že imituje učení biologického mozku, neboť potřebuje vstupní informaci a výsledek, ke kterému tato informace vedla. Takový model se poté snaží nalézt podobnost ve vstupní informaci, která by vedla na stejný výsledek. NN modelu je proto představen soubor vstupních a cílových (požadovaných) dat. Tento přístup se nazývá *Supervised Learning* a existují i další, které ale nejsou v rámci této práce používány. V průběhu mnoha iterací je prováděna tzv. *Back Propagation* optimalizace, která počítá odchylku (*loss function*), s jakou současný model odhadl požadovaný výsledek pro poskytnuté vstupy. Z této odchylky je pro každý parameter NN (váhy, biasy, parametry aktivační funkce, atd.) počítán gradient, kterým je třeba parameter upravit, aby se výstup při další iteraci přiblížil požadovanému výsledku [4]. Tento proces optimalizace se nazývá *Gradient descent* a právě pomocí loss funkce hledá lokální (v reálných případech nelze mluvit o globálních) minima vnitřních parametrů modelu. Existuje mnoho druhů *loss* funkcí (např. Mean-Square-Error, Cross-

Entropy, atd.), přičemž každá má svůj účel a využití pro jiný typ úloh. Optimalizačních metod také existuje mnoho, populární je např. *Adam* [14], ale jejich využití je obecné, tedy jednu metodu lze využít pro různé typy úloh, což je právě případ zmíněného algoritmu *Adam*.

2.2.2 Rozhodovací stromové struktury

Samotný název (v a.j. *Decision Trees* a odtud i nadále používaná zkratka DT) vypovídá o struktuře těchto modelů viz obrázek 6. Jedná se o soubor podmínek, jejichž výsledkem je pouze ano a ne, v případě některých zdrojů je používána notace test, který je buďto splněn nebo není a každá nová větev odpovídá výsledku. Každá úroveň těchto podmínek vede na 2 nové podmínky (větve v a.j. *Branch* nebo *Sub-Tree*), které takto vedou až ke konečné množině prvků (list nebo-li v a.j. *Leaf Node*), jež splňuje všechny předchozí. První z těchto podmínek je označována jako kořenový uzel (v a.j. *Root Node*). Každá podmínka resp. uzel stromu představuje rozhodování podle jedné proměnné ze vstupní sady [15].



Obrázek 6: Příklad jednoduché stromové struktury[6]

Tento přístup je velice příhodný pro programovou implementaci, neboť většinu logiky lze interpretovat právě sadou podmínek "*if - else*". Navzdory prvotnímu dojmu,

[6] Převzato z: GUPTA, Prashant. Decision Trees in Machine Learning. *Towards Data Science* [online]. 17.5.2017 [cit. 2020-05-17]. Dostupné z: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052

že jejich využití je pouze pro klasifikační účely, tak existují i varianty, které řeší regresi. Proto jsou tyto DT algoritmy v literaturách označovány zkratkou *CART* (Classification and Regression Trees) [15].

Využití nacházejí DT v mnoha odvětvích např. manufaktura - hodnocení chemických materiálů pro výrobu, biomedicína - identifikace funkcí, které mají být použity v implantovatelných zařízeních, farmaceutický průmysl - analýza efektivity léčiv, a další. Jednou z nejpopulárnějších oblastí je data-mining, kde jsou využívány k výběru strategie pro dosažení určitého (požadovaného) cíle.

Rozdíl regresních oproti rozhodovacím stromovým strukturám spočívá v tom, že u regresních je jedním z kritérií počet prvků v konečném listu. Jinak se učící proces shoduje - oba přístupy rozdělují stavový prostor nezávislých proměnných do ne-překrývajících se regionů. DT spadají do *Supervised Learning* kategorie, neboť je pro vytvoření modelu potřeba dataset s popsanými požadovanými výstupy.

Hledání DT modelu skrývá problém v podobě *overfittingu* (v č.j. lze přeložit jako přetrénování), který se objevuje, když se model snaží vytvářet další a další úrovně podmínek/testů ve snaze co nejvíce zredukovat chybu vzniklou při učení (v a.j. *training error*). Toto se většinou stává, když je vytvořeno mnoho větví v důsledku krajních hodnot a nesrovnalostí. Řešením tohoto problému bývá limitování velikosti stromové struktury, nebo tzv. pruning, což je metoda na odstranění irelevantních větví.

Vylepšením DT metod je tzv. *boosting*. Toto označení referuje na sestavení několika DT modelů do sekvence, přičemž cílem každého z nich je napravit nedostatky předchozího modelu.

2.3 Interpretovatelnost modelů a vliv proměnných

Nedílnou součástí tématiky umělé inteligence a strojového učení je otázka, jak se vlastně daný model rozhoduje a proč došel k takovému výsledku? Tyto otázky jsou předmětem jedné z největších debat probíhajících v této oblasti v roce 2019, protože uživatelé, vědci a především lidé, kteří jsou těmito modely ovlivněny (nebo mohou být v nadcházejících letech - viz autonomní vozidla), na ně chtějí znát odpovědi. V mnoha případech nestačí brát model jako “black box” (viz Obrázek 7).

Obrázek 7: Interpretovatelnost modelů[7]

Např. v USA soudní systém začíná používat algoritmy pro hodnocení rizik v případu obžalovaného, přičemž takový přístup přináší nejen problém chybějící průhlednosti modelu, ale i požadavek na právní obhajitelnost takového výsledku[1]. Dalším příkladem je diagnóza onemocnění u pacienta - v roce 2015 skupina vědců vytvořila *Deep Patient* software [18], pro jehož natrénování bylo použito přibližně $700 \cdot 10^3$ dat pacientů se stovkami proměnných z různých testů, lékařských vizit atd. Jeho schopnost predikce na nových datech se ukázala jako mimořádná, avšak zároveň že je to poněkud *black box*. Model totiž dokázal překvapivě dobře vyhodnotit schizofrenní poruchu, která je mezi lékaři notoricky známá pro její obtížnou identifikovatelnost a daný software nedokázal poskytnout jasnou odpověď, jak k tomu došel.

V případě statistických modelů jako CoxPH nebo KM je otázka interpretovatelnosti poměrně snadno vysvětlitelná - KM je modelem, který je podložen statistickým výpočtem křivky. CoxPH jakožto model vycházející z lineární regrese obsahuje regresní koeficienty b_i, které přímo souvisí jak s učícími daty, tak s ovlivněním výsledku vycházejícím z (5) (násobení vstupního vektoru). Tyto koeficienty tedy přesně říkají, jaký vliv má daná proměnná na výsledek funkce.

Jak již bylo nastíněno pomocí uvedených příkladů, tak AI modely, které se dokáží s vysokou přesností naučit vazby v moderních obrovských datových sadách, bývají většinou natolik komplexní, že i přední odborníci v dané oblasti nedokáží vysvětlit, jak

[7] Převzato z: *Shap* [knihovna]. [cit. 2020-05-18]. Dostupné z: https://github.com/slundberg/shap

a proč to takový model dokáže. V posledních letech proto bylo navrženo několik přístupů, které tuto problematiku řeší. Některými vybranými jsou:

- LIME [20] - metoda interpretuje jednotlivé predikce modelu na základě lokální aproximace modelu kolem dané predikce.
- DeepLIFT [26] - metoda pro vysvětlení deep learning modelů pomocí rekurzivní predikce. Využívá výpočet sklonu namísto gradientu, který popisuje, jak se y mění, když se x liší od základní linie aktivační funkce neuronu.

Studie [17] sjednocuje několik takových metod a jako vysvětlující hodnotu představuje tzv. "*SHAP values*" (SHapley Additive exPlanations), která udává význam proměnných (kovariátů) vstupujících do AI modelu. Konkrétně lze význam těchto hodnot vyložit jako průměr mezních příspěvků napříč všemi permutacemi.

Využívá přístupu herní teorie k vysvětlení jakéhokoliv modelu strojového učení. Propojuje přidělení optimálního příspěvku s lokálním významem *SHAP* hodnotami pocházejícími z herní teorie. Studie zároveň dokazuje její využitelnost a přínos do vědní oblasti AI na provedených experimentech neboť přináší hned 3 výhody:

1. Obecná interpretovatelnost – Umožňuje zobrazit velikost příspěvku každé vstupní proměnné do výsledku modelu.
2. Lokální interpretovatelnost – Každé pozorování je ohodnoceno zvlášť, takže lze zobrazit význam jednotlivých proměnných i pro 1 vybrané pozorování.
3. SHAP hodnoty lze spočítat pro jakýkoliv model (DT, NN, lineární atd.)

3 Praktická část

Pro potřeby praktické části práce byla poskytnuta reálná data z pražského Ústavu hematologie a krevní transfuze (dále ÚHKT). Konkrétně se jedná o 1385 záznamů z databáze transplantací souvisejících s léčbou leukémie. Na data se vztahuje zákon o Obecném nařízení o ochraně osobních údajů známý pod anglickou zkratkou GDPR (zkratka pro General Data Protection Regulation), proto jsou veškerá data zcela anonymní.

Důležitými informacemi v datech jsou ty, které jsou nějakým způsobem zajímavé, nebo podstatné pro pokračování léčby pacienta. Z poskytnutých dat jsou to následující:

- Doba přežití → počet dní, po jejichž dobu je pacient sledován, nebo uplynulá doba před jeho úmrtím od počátku sledování
- Doba do relapsu → počet dní, před tím, než se u pacienta nemoc znovu projeví. Tato událost nemusí nastat
- Doba do aGVHD → zkratka aGVHD vychází z anglického Acute Graft-versus-Host Disease a v českém jazyce je označována jako reakce štěpu proti hostiteli. Jedná se o hlavní zdravotní komplikaci po přijetí transplantace. Tento výstup by tedy měl predikovat, po jaké době u pacienta aGVHD nastane
- Doba do cGVHD → počet dní, po kterých by mohlo dojít k rozvoji chronické GVHD

Ostatní pole (viz příloha C Popis sloupců) lze pojmout jako prediktory, kovariáty a vysvětlující nezávislé proměnné. Samotné otázky pro zodpovězení modelem mohou být tedy následující:

- Jaká je pravděpodobnost, na základě prediktorů, že pacient zemře?
- Jaká je pravděpodobnost, na základě prediktorů, že pacient rozvine akutní GvHD?
- Jaká je pravděpodobnost, na základě prediktorů, že pacient rozvine chronickou GvHD?

- Jaká je pravděpodobnost, na základě prediktorů, že dojde k rozvoji relapsu?

Eventuelně i:

- Pokud se u pacienta rozvine akutní GvHD, jaká je pravděpodobnost, že zemře?

3.1 Analýza dat

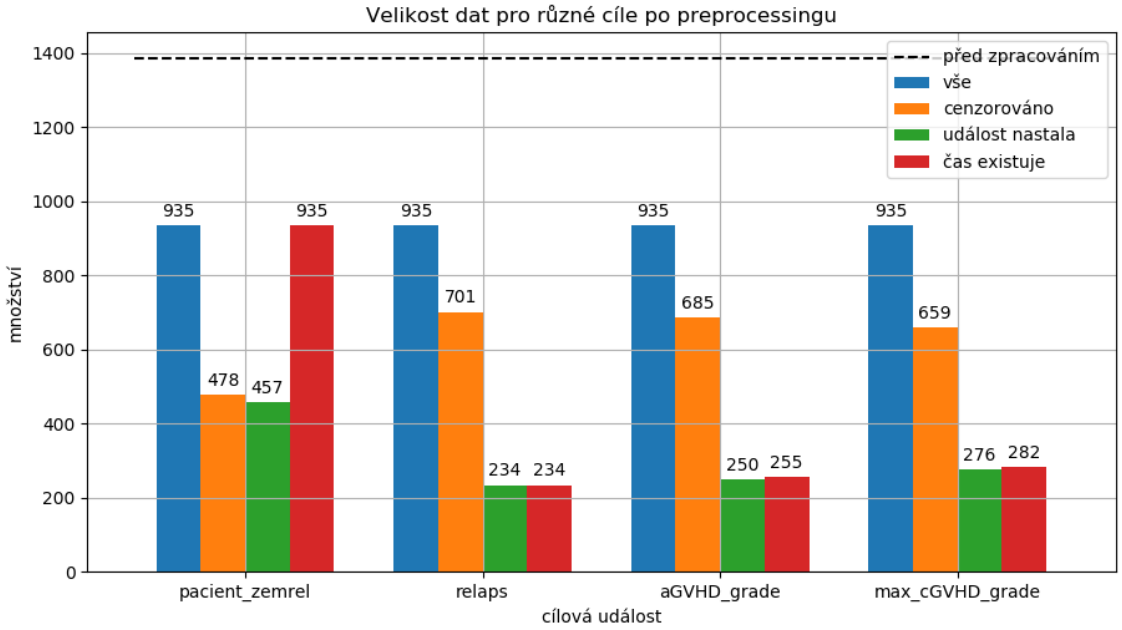
Tato kapitola se věnuje základnímu rozboru poskytnutých dat. Jejich interpretování tak, aby byla srozumitelná i pro člověka, je popsáno v příloze C Popis sloupců. Dále je zde zobrazeno několik základních statistických vlastností těchto dat. A poté způsob jejich před-zpracování pro strojové učení.

Sloupce NRM, DFS, relaps, pacient_zemrel jsou indikátory výskytu události, což znamená, že danou událost je třeba ji vyloučit ze vstupních kovariátů spolu s jejím TTE (pokud existuje).

3.1.1 Statistické vlastnosti

V případě práce s reálnými daty je zcela iracionální předpokládat, že budeme mít k dispozici dataset, ve kterém jsou všechny údaje vyplněné. Důvodem pro chybějící údaj může být, že ho zadávající pracovník nevyplnil, nebo subjekt nebyl na danou veličinu testován atd. Důvod je nicméně irrelevantní a bylo třeba zajistit, aby tato data byla vyloučena z učícího datasetu. Pro tento účel byl vytvořen algoritmus, který je součásti preprocessingu (aneb česky před-zpracování) a je blíže popsán v kapitole 3.1.2. Původní velikost dat je na obrázku 8 znázorněna čárkovanou čarou a nová plnou.

Protože cenzorované údaje mají jiný vliv na učení modelu (alespoň u Coxova modelu) než údaje, u kterých se událost vyskytla, tak je vhodné si zobrazit rozdíl v jejich četnosti. Na obrázku 8 je tento rozdíl graficky zobrazen pro různé cíle (uvedeny v kapitole 3) po jejich před-zpracování a vyloučení neúplných subjektů. Lze si všimnout, že u všech, krom cíle “pacient_zemrel”, je počet subjektů s vyplněným TTE podstatně menší, než dostupná velikost. Dalším grafickým znázorněním je obrázek 20, který je z důvodů přehlednosti umístěn na relevantnějším místě této práce a popisuje histogram TTE pacientů napříč časovým intervalem (cíl je “pacient_zemrel”).



Obrázek 8: Porovnání velikosti dat různých cílů po preprocessingu

3.1.2 Preprocessing

Vzhledem k realistickému přístupu k datům, tedy předpoklad, že uživatel využívá databázi, nebo mu data byla poskytnuta v excelové tabulce, nebyly nalezeny žádné již hotové algoritmy pro preprocessing dat pro křivky přežití, které by zároveň řešily některé nedostatky. Např. že nedokázaly rozlišit kategorické veličiny od ostatních (a poté váhu pacienta rozdělily podle One-Hot-Encoding pravidel), velkou část přípravy bylo nutno vyřešit manuálně atd.

Především pak specifická vlastnost datasetu poskytnutého pro tuto práci - některé veličiny (sloupce) jako např. ty, co v názvu obsahují *aGVHD*, mohou být vyplněny pouze v případě, že je vyplněn i *aGVHD_grade*. Jedná se tedy o tzv. podmíněně vyplněné údaje, jejichž vliv na přežití pacienta může hrát nezanedbatelnou roli. Zcela jistě lze předpokládat, že tuto vlastnost mohou mít i jiné soubory dat.

Pravděpodobně hlavním nedostatkem nalezených preprocessing algoritmů bylo odstranění neúplných záznamů, tedy řádků, kde alespoň jedna hodnota je `null/None` nebo `NaN`. Po využití běžně používaných přístupů se velikost datasetu zmenšila o téměř 97,5% (z původních 1385 na 35), což je nedostatečná velikost dat pro strojové učení.

Pro řešení této práce byl tedy vytvořen vlastní `Preprocessor` jehož konstruktor třídy s docstring (v a.j.) vysvětlivkou je k nalezení v příloze Zdrojový kód 1. Mimo vlastní přístup pak třída využívá preprocessing metody dostupné z knihovny Sklearn[8].

První výhodou navrženého `Preprocessor`-u oproti ostatním je, že uživateli je umožněno specifikovat sloupce, se kterými je nutno provést některé operace, aniž by data dopředu upravoval a připravoval manuálně. Takto lze specifikovat sloupce, které obsahují kategorické veličiny, ty jsou automaticky zakódovány z textových hodnot (např. pohlaví muž/žena) na číselnou reprezentaci (tzv. *Label-Encoded*) a následně rozděleny do nových binárních veličin obsahující pouze 0 a 1 pomocí *One-Hot-Encoding* algoritmu. Dále lze specifikovat sloupce, pro které se využije algoritmus standardizace (`StandardScaler`) namísto výchozí normalizace pro kterou jsou sloupce detekovány automaticky porovnáním hodnot, zda obsahují i něco jiného než pouze 0 a 1 (takové sloupce jsou ponechány beze změny). Je možno také specifikovat sloupce, které budou při procesu odstraněny (využití např. pro databázové ID).

Další specifikace řeší již zmíněné podmíněně vyplněné hodnoty, neboť v mnoha případech v databázi je brána prázdná hodnota, jako že u pacienta se daný údaj nevyskytl (viz *aGVHD*), nebyl testován anebo byl výsledek negativní. V tomto případě jsou zadány jako list pythnovských `tuple`, kde prvním elementem `tuple` je název hlavního sloupce (dále v této kapitole pro přehlednost jako A), který definuje přítomnost testu/události a ve druhém elementu je seznam sloupců, které jsou na první element vázány (dále v této kapitole pro přehlednost jako B). Tyto vázané sloupce B mohou a nemusí být vyplněny, proto se nevyplněné nahradí 0. Pokud A chybí, tak se zakóduje jako nová kategorie reprezentující chybějící údaj a ta je po *One-Hot-Encoding* procesu odstraněna, což zanechá 0 v ostatních kategoriích dané veličiny (stejného přístupu je využito i ve zpracování kategorických veličin). Z toho zároveň vyplývá, že jako A mohou být specifikovány pouze údaje, které jsou v datech reprezentovány kategoricky nebo binárně.

Druhou výhodou je automatické čištění kategorických dat pomocí nahrazení chybějícího údaje vlastní kategorií, která je nakonec odstraněna z výsledného datasetu

[8] *Scikit-learn* [knihovna]. [cit. 2020-05-18]. Dostupné z: https://scikit-learn.org/stable/index.html

(podobnost s nahrazením podmíněně vyplněnými hodnotami, avšak jiný případ). Důvodem pro tento komplexní postup je, aby se zachovalo co nejvíce možných zdrojových dat, aniž by byla výrazně změněna strukturální integrita dat - algoritmy pro např. učení neuronových sítí vypočítávají změnu jednotlivých vah na synapsích porovnáváním a minimalizací výsledku *loss* funkce se vstupy skrze *backpropagation*. Ze základní matematické podstaty NN (11) lze odvodit, že váha w_i na vstupu, kde $x_i = 0$ se nezmění neboť jejich násobek je také 0, ale může mít dodatečný malý vliv na bias w_0 neuronu a ostatní váhy.

Třetí užitečná vlastnost je zpětné pojmenování sloupců. Základní metody totiž sice zpracují hodnoty, ale již jim zpětně nepřiřadí názvy sloupců/proměnných. Tento problém je obzvlášť znatelný u kategorických proměnných a *One-Hot-Encoded*, protože po zpracování není možnost dohledat, který sloupec odpovídá jaké původní hodnotě. V této třídě je tento problém vyřešen a po každém preprocessingu jsou názvy sloupců zpětně přiřazeny i s jejich původními kategorickými hodnotami. Využitelnost je především v hodnocení vlivu proměnných (kovariátů) na výsledek, neboť bez této implementace by byly jednotlivé proměnné uváděny pouze jejich indexem v datasetu.

Pro další potřeby této práce je do `Preprocesor`-u zahrnuta možnost vynechat *One-Hot-Encoding*, neboť použitá knihovna Lifelines[9] pro práci s CoxPH modelem toto řeší sama a u většiny binárně kódovaných veličin vypisuje *ConvergenceWarning*, což je varování značící nízkou varianci v datech. Vzhledem k velkému počtu dat, která nemají vyplněný TTE údaj události (viz Obrázek 8) byla zahrnuta možnost vyloučit takové subjekty z výsledného datasetu a ponechat tedy pouze ty, kde TTE události nebo cenzorování > 0.

3.2 Návrh algoritmů

Z důvodu povahy této práce bylo ke všem modelům přistupováno tak, aby bylo umožněno hodnocení jejich vzájemné podobnosti resp. kvality spolu s možností fungování jako samostatný model. Proto byla vytvořena výchozí rodičovská třída aneb

[9] *Lifelines* [knihovna]. [cit. 2020-05-18]. Dostupné z: https://lifelines.readthedocs.io/en/latest/index.html

tzv. `base class` jejíž metody jsou děděny každým modelem, eventuelně přepisovány, pokud je tomu třeba. Třída zároveň zajišťuje stejný přístup ke vstupním datům.

Obvyklým způsobem, při trénování modelů a obecně strojovém učení, je rozdělit dostupná data na učící a validační (tzv. *test set* a *train set*). Standardně se volí poměr 80% do train setu a 20% do test setu. Některé přístupy dokonce využívají rozdělení na 3 části (např. 70-20-10%), kdy train a test set jsou využity v průběhu učení a 3. až jako validační po dokončení učení, protože se model časem dokáže "naučit" i vazby v daném testovacím setu[10]. V této práci je použita druhá možnost a to z několika důvodů, které ani nesouvisí s učícími schopnostmi modelů.

Pokud chceme zachovat logiku hodnocení modelu na datech, které mu nebyly představeny při trénování, tak je třeba rozdělit data na 2 části - učící a validační (nazvěme je pro tuto kapitolu T a V), nicméně v průběhu učení je třeba daný proces validovat také, proto je učící set T znovu rozdělen na 2 části v poměru 80 a 20%. Avšak ve validačním setu V se mohou nacházet kategorické hodnoty, které nebyly přítomny v setu T (`Preprocessor` po jeho *nafitování* má striktně daný počet výsledných veličin ve vektoru a od toho se odvíjí velikost vstupních vrstev modelů, které jsou dány při jejich inicializaci), tudíž by `Preprocessor` nedokázal tuto kategorii interpretovat a metoda by selhala. Tento přístup je dále podpořen tím, že uživatel může chtít získat křivku přežití pro určité subjekty, ale nechce, aby byl model ovlivněn jejich současným stavem.

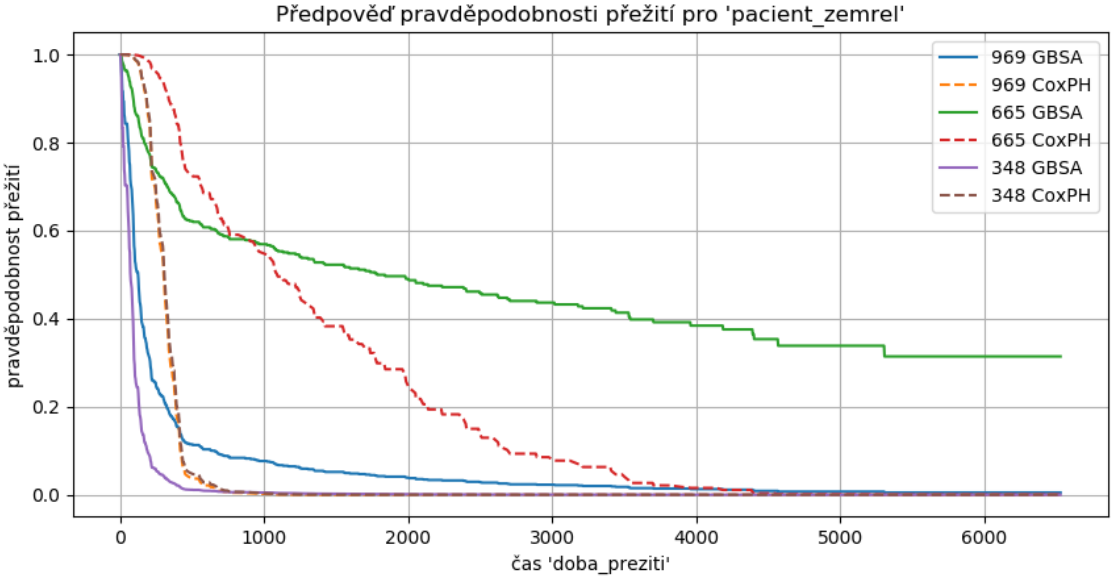
Z tohoto důvodu jsou součástí inicializačních parametrů `base class` 3 sety a to učící (T), validační (V), který ale není povinný, a všechna data. `Preprocessor` je tedy vytvořen na základě všech dostupných dat, aby zmapoval všechny vyskytující se kategorie. Následné učení je prováděno pouze na training setu T.

Pro porovnání výsledné křivky přežití samostatného modelu vůči CoxPH je součástí této `base class` i možnost (parametr `include_cox`) vytvořit standardní CoxPH model z knihovny Lifelines[11], jehož výsledek je poté zahrnut do grafů, kde jsou

[10] *Machine Learning Crash Course* [online]. [cit. 2020-05-18]. Dostupné z: https://developers.google.com/machine-learning/crash-course
[11] *Lifelines* [knihovna]. [cit. 2020-05-18]. Dostupné z: https://lifelines.readthedocs.io/en/latest/index.html

pod názvem “CoxPH”. Do `base class` je mimo jiné zahrnuta i možnost vykreslení Kaplan Meierovy křivky (také knihovna [11]). Další součástí je hodnocení modelu, které je detailně popsáno v kapitole 4.

Hlavní funkcionalita se odehrává v metodě `predict_survival_line(…)`, která jako povinný vstup bere soubor subjektů, pro které je vypočítána křivka přežití. Výpočet probíhá tak, že každý subjekt zvlášť je duplikován N-krát a každému duplikátu je přiřazen jiný TTE údaj z intervalu, který je ve výchozím nastavení od 0 do 2000 (při evaluaci je použit celý interval TTE událostí v testovacích datech). Tento soubor duplikátů je následně před-zpracován preprocessorem a poté vložen do modelu, který vrátí list predikcí přežití pro jednotlivé zahrnuté TTE údaje. Pokud bylo zvoleno zahrnutí CoxPH modelu, tak je stejný postup aplikován i na něj. Po zpracování celého souboru subjektů je možnost vykreslit graf křivek přežití pro jednotlivé subjekty, takový graf je znázorněn na obrázku 9, kde v legendě křivek je uveden model a id pacienta a znázorňuje průběh pravděpodobnosti 3 konkrétních pacientů.



Obrázek 9: Ukázka výstupních křivek přežití v porovnání s CoxPH

3.2.1 Neuronové sítě

Kromě Coxova modelu proporcionálních rizik (součást `base class`) jsou neuronové sítě prvním typem implementovaných modelů. Pro samotné modely byla využita knihovna PyTorch[12] kvůli její přívětivosti vzhledem k low-level experimentování, které se uplatňuje především v akademické sféře a výzkumech, ale zároveň i z důvodu již dříve získaných zkušeností.

Pro usnadnění implementace učícího procesu byla využita knihovna Skorch[13], což je wrapper okolo PyTorch-e. Tento wrapper obsahuje kompletní algoritmus pro učení modelů, ve kterém je zahrnuto mnoho postupů, které byly v posledních letech využívány při učení neuronových sítí, jako jsou např. *Checkpoint*, *EarlyStop* nebo sledování vývoje modelu atd. Lze navolit, že po ukončení trénování modelu, je historie učícího procesu vykreslena do grafu. Jako optimalizační *gradient descent* algoritmus byl zvolen běžně používaný Adam [14]. Jednotlivý *criterion* aneb *loss* funkce se liší v závislosti na modelu.

Toto učení je stejné pro všechny typy implementovaných NN modelů a nachází se ve společné třídě (jenž dědí z `base class`), ze které NN modely dědí. Modely lze parametrizovat pomocí tzv. `kwargs` nebo-li *key-word-argumentů*, které mohou být vloženy do konstruktoru modelu. Tímto způsobem lze nastavit velikost sítě (počet vrstev a neuronů v každé z nich), dropout a zda se má využít technika zvaná *BatchNormalization*. Zdrojový kód podoby neuronové sítě je k nahlédnutí v příloze viz Zdrojový kód 2.

3.2.1.1 Standardní model

Za standardní přístup je v tomto případě považován takový model, který je použit jako klasifikátor mezi cílovými hodnotami 0 a 1, resp. tedy v případě této úlohy předpoklad, že se událost zájmu objeví pro daný vstupní vektor. Vzhledem k povaze úlohy je v tomto případě možné mluvit i o pravděpodobnosti, že se vyskytne událost.

Protože nás zajímají 2 cílové hodnoty (cenzorováno/událost), tak lze k návrhu sítě přistupovat dvěma způsoby:

[12] *PyTorch* [knihovna]. [cit. 2020-05-18]. Dostupné z: https://pytorch.org/
[13] *Skorch* [knihovna]. [cit. 2020-05-18]. Dostupné z: https://skorch.readthedocs.io/en/stable/index.html

1. Výstupní vrstva sítě by obsahovala aktivační funkci *sigmoid*, která zajišťuje limitování výstupu do intervalu $< 0; 1 >$. Data by se ponechala "tak jak jsou", čímž by se zachovala logika, kdy 1 na výstupu značí událost a naopak. Jako *loss* funkce se v tomto případě používá *BCEloss* (zkratka pro *Binary Crossentropy*). Avšak navzdory první myšlence, co se nabízí a často plete, výstupem tohoto přístupu není pravděpodobnost, neboť se jedná spíše o logistickou regresi, než o klasifikaci příslušnosti k jedné ze dvou skupin. Proto je vhodnější na tento výsledek nahlížet, jako na jistotu modelu.

2. Výstupní vrstva by obsahovala 2 výstupy - cenzorováno a událost. Tento přístup již spadá do skupiny *multi-class classification*, ve které se místo aktivační funkce *sigmoid* na výstupu sítě používá funkce *softmax*, která přiřazuje pravděpodobnosti každému z výstupů. Jako *loss* funkce se používá *CrossEntropy*, jejíž implementace v PyTorch-y kombinuje *LogSoftmax* a *Negative-Log-Likelihood* a protože chceme na výstupu *softmax* zachovat tak je výstup v průběhu učení jiný než po jeho dokončení (aplikovaný logaritmus na vektor výstupu z funkce *softmax* činí učení rychlejší a kvalitnější, neboť se více projevují rozdíly vůči porovnávanému cíli). Díky implementaci *CrossEntropy* v PyTorch-y není třeba cílový vektor událostí nijak měnit, protože při učení je jako cílová hodnota požadován index kategorie namísto *One-Hot-Encoded* kategorií. Výstupem z tohoto modelu jsou tedy 2 hodnoty pro každou ze tříd cenzorováno/událost. Při běžné klasifikaci nás zajímá pouze třída s nejvyšší hodnotou, avšak v tomto případě lze právě danou hodnotu považovat za pravděpodobnost, se kterou model přiřadí vstupní vektor jedné z distribucí zmapovaných funkcí *softmax*).

Článek [7] využívá podobný přístup, který vzhledem k tehdejším technologickým možnostem obsahuje v NN pouze 1 skrytou vrstvu o 2 nebo 3 neuronech.

3.2.1.2 Adaptace Coxova modelu proporcionálních rizik

Jedním z prvních výzkumů, které se věnovaly aplikaci neuronových sítí na estimaci křivek přežití je [7], avšak v roce 1995, kdy tento článek vyšel, nebyly NN ani zdaleka na úrovni, na které jsou dnes. Podobné problematice se věnuje i novější (2018) studie [13], která představila, že navzdory dřívějším testům, které nedokázaly překonat

kvalitu semi-parametrických modelů, toho dnešní algoritmy již schopné jsou. Tento výzkum dokonce označuje jejich "risk NNs" jako state-of-art přístup, který může být použit v různých zdravotnických aplikacích.

Adaptace CoxPH spočívá v nahrazení částečně rizikové funkce ($\sum_{i=1}^{n} b_i(x_i)$ z (5)) $h_i(x)$ v (14) (jinak zapsaná rovnice CoxPH (5)), kde i resp. τ představuje typ léčby a v dané studii [13] slouží k návrhu nejvhodnější léčby danému pacientovi na základě jeho příznaků) neuronovou sítí, kde x je vstupní vektor kovariátů. Architektura neuronové sítě v článku [13] se skládá z *fully-connected* vrstev neuronů následované dropout metodou s jedním neuronem a lineární aktivační funkcí na výstupu. V implementaci této práce je využito inicializačního algoritmu NN z Zdrojový kód 2.

$$\lambda(t; x|\tau = i) = \lambda_0(t) \cdot e^{h_i(x)} \tag{14}$$

Jednou z nejpodstatnějších částí této implementace je jiná *loss* funkce (15), než které se běžně pro učení NN používají. Konkrétně tedy average-negative-log-likelihood z (10) doplněné o regularizaci, tedy:

$$l(\theta) := -\frac{1}{N_{E=1}} \sum_{i:E_i=1} \left(\widehat{h_\theta}(x_i) - log \sum_{j \in R(T_i)} e^{\widehat{h_\theta}(x_j)} \right) + \gamma \cdot \|\theta\|_2^2 \tag{15}$$

Kde $\hat{h}_\theta(x)$ je výstup z neuronové sítě, $N_{E=1}$ počet pacientů s pozorovatelnou událostí a γ je L2 regularizační parametr. Pro nalezení vah v NN je použita *gradient descent* optimalizace, která tuto funkci minimalizuje.

Implementace (15) se nachází v příloze (viz Zdrojový kód 3) a byla přejata a upravena pro potřeby algoritmů této práce ze zdrojového kódu knihovny PyCox[14]. Vzhledem k závislosti (15) na 3 údajích - výstup NN, událost (0/1) a velikosti TTE údaje bylo třeba upravit učící dataset, aby s touto změnou korespondoval. Daný dataset je objekt vrácený metodou `prepare_data()`, kterou je nutno implementovat pro každý NN model.

[14] *Pycox* [knihovna]. [cit. 2020-05-18]. Dostupné z: https://github.com/havakv/pycox

Základní riziková funkce $\lambda_0(t)$ pak zůstává stejná jako v klasickém CoxPH modelu. Za její estimátor se nejčastěji používá Breslow Estimator(dále jako zkratka BE), který odhaduje hodnotu základního rizika jako exponencielu kumulativní základní rizikové funkce[30]. Kvůli několika číselným problémům v implementaci bylo v této práci nakonec využito jiného přístupu - BE jakožto neparametrický estimátor $\hat{S}(t)$ základní rizikové funkce podle TTE distribuce je definován jako exponenciela negativního kumulativního rizika $\hat{A}(t)$ Nelson-Aanlen (viz kapitola 2.1.2) funkce, tedy (16)[15]. Proto bylo přistoupeno k řešení pomocí právě Nelson-Aanlen estimátoru. Učícími daty pro tento estimátor jsou podobně jako např. pro KME pouze TTE hodnoty a události.

$$\hat{S}(t) = \exp(-\hat{A}(t)) \tag{16}$$

V této práci jsou učící data před-zpracovávána ve výchozím nastavení pomocí normalizace, na rozdíl od [13], kde je využita standarizace. Dalším rozdílem je, že zde je v NN využita *BatchNormalization* technika, ale opět nastavitelná. Dané rozdíly, spolu s nastavením architektury sítě, byly porovnány empirickým způsobem. Implementovaný zdrojový kód přístupu a získání výsledků z tohoto modelu je k nahlédnutí v příloze (viz Zdrojový kód 4).

3.2.2 Decision tree modely

3.2.2.1 AutoML

Pro implementaci těchto modelů byl využit nástroj AutoML[16] (Automatic Machine Learning) vyvíjený společností h2o.ai. Důvodem pro tento výběr bylo, že daný nástroj obsahuje širokou škálu typů modelů a především se jedná o sjednocený soubor state-of-art přístupů k umělé inteligenci, která využívá dokonce i automatizovaný výběr a parametrizaci modelu. Dalším pozitivem je, že AutoML již obsahuje podrobné informace o průběhu učení vybraného modelu včetně vlivu a důležitosti jednotlivých vstupních proměnných. Velká výhoda spočívá právě v automatizovaném učení modelů,

[15] *Survive* [knihovna]. [cit. 2020-05-18]. Dostupné z: https://survive-python.readthedocs.io/index.html
[16] *H2O.ai* [knihovna]. [cit. 2020-05-18]. Dostupné z: http://docs.h2o.ai/h2o/latest-stable/h2o-docs/index.html

díky čemuž není třeba nastavovat žádné parametry jako např. *learning rate* nebo architekturu jako tomu je u neuronových sítí.

V průběhu učení algoritmus postupně zkouší různé typy modelů, pro které při další iteraci změní vnitřní hyperparametry s ohledem na předchozí výsledek. Konkrétní dostupné typy jsou:

- DRF - Distributed Random Forest. Místo jednoho rozhodovacího nebo regresního stromu je vytvořen soubor těchto struktur a jejich výstupem je průměrná predikce z celé populace.
- GLM - Generalized Linear Model. Mimo Gaussovo rozdělení obsahuje i Poissonovo, binomialní atd.. Modely jsou provázány tzv. *link* funkcí, která jednotlivé kovariáty propojuje s vybraným rozdělením.
- XGBoost - Paralelní gradient boosted decision trees. Viz kapitola 2.2.2.
- GBM - Gradient Boosting Machine. Paralelně skládané regresní stromové struktury, jejichž výsledky jsou aproximovány.
- DeepLearning - viz kapitola 2.2.1.
- StackedEnsemle - Metoda používající několik algoritmů pro zajištění lepší prediktivní schopnosti, než jaká by byla možná dosáhnout každým z těchto algoritmů samostatně.

S ohledem na povahu této práce a požadavek lepšího zpracování nelineárních vazeb v datech byly z učících algoritmů vyloučeny GLM a DeepLearning, který je řešen v kapitole 3.2.1 pomocí jiného přístupu. Výsledný typ modelu tedy není předem definován a je ponechán internímu výběru a parametrizaci. Vstupní data do tohoto algoritmu a učícího procesu jsou stejně jako v předchozích případech nejprve předzpracována vlastní preprocessing třídou `Preprocessor`. Cílová data pro učení jsou také stejná a to pouze vektor nebo sloupec obsahující EOI. TTE je zde součástí vstupního vektoru.

3.2.2.2 XGBoost

Pro další pokusy z oblasti DT byla využita knihovna XGBoost[17], která je stejně jako PyTorch[18] a na rozdíl od H2O.ai kompatibilní s algoritmy použitými pro interpretaci modelů a hodnocení vlivu vstupních kovariátů na výstup modelu (viz kapitola 2.3). Ačkoliv byly implementovány základní modely podobně jako Standardní u NN (viz kapitola 3.2.1.1) hlavním cílem byla snaha o stejnou adaptaci CoxPH modelu jako u NN (viz kapitola 3.2.1.2) Tento pokus však selhal při snaze implementovat funkci pro výpočet gradientu při učení modelu. Sice je umožněno definovat vlastní funkci, avšak nelze ovlivnit tvar cílových dat - knihovna podporuje pouze [n,1] podobu cílového vektoru (parametr label v `DMatrix`), zatímco (10) vyžaduje [n,2] (TTE a výskyt EOI). Alternativa pro vícenásobný výstup není poskytnuta. Ani obejití (sloučit data do `tuple` a poté uložit do jediného sloupce) nelze provést, neboť knihovna kontroluje datový typ a akceptuje pouze numerické hodnoty.

Nicméně metoda byla v práci ponechána i s nevhodnou učící funkcí (konkrétně `binary:logistic`), takže výstupní hodnoty jsou v rozmezí $< 0; 1 >$ namísto zamýšlené logaritmické škály. Jako základní riziková funkce je zde využit stejný přístup jako u NN CoxPH a to Nelson-Aalen estimátor.

3.2.2.3 Random Survival Forest

Tato metoda byla představena v roce 2008 publikací [11]. Využívá random forest přístup, což je ensemble DT metoda, která využívá randomizaci během učícího procesu. Ta je prováděna ve dvou formách - nejprve se z učících dat vezme náhodný vzorek, ze kterého se vytvoří DT, poté je v každém uzlu vybrána podmnožina proměnných (kovariátů) jako kandidát pro rozdělení. Výsledky všech stromů jsou průměrovány, což spolu s randomizací při učení poskytuje nízkou zobecňující chybovost.

Pro rozdělení každého uzlu ve stromě je použito kritérium zahrnující (stejně jako u NN CoxPH) jak výskyty EOI tak i TTE. Jako kumulativní hazard funkce je využit Nelson-Aalen estimátor (viz kapitola 2.1.2) v koncovém uzlu. Ensemble je tedy průměr těchto koncových rizikových funkcí.

---

[17] *XGBoost* [knihovna]. [cit. 2020-05-18]. Dostupné z: https://xgboost.readthedocs.io/en/latest/index.html
[18] *PyTorch* [knihovna]. [cit. 2020-05-18]. Dostupné z: https://pytorch.org/

Pro implementaci této metody byla využita knihovna *scikit-survival*[19] nebo-li sksurv, přičemž hyperparametry pro učení byly nalezeny empiricky.

3.2.2.4 Gradient Boosting Survival Analysis

Knihovna *scikit-survival*[19] obsahuje i další ensemble metodu. Implementuje přístup Gradient-boosted Cox proportional hazard loss s využitím regresních stromových struktur jako základ učení, což znamená, že v každé fázi učení je regresní strom učen na základě negativního gradientu *loss* funkce. Právě *loss* funkce je zde log-partial likelihood (17), avšak bez regularizace jako v (15). V algoritmu je požadavkem maximalizovat logPL, proto se pouze vynásobí -1 a zbytek algoritmu již pokračuje beze změn[21].

$$\psi(y, F) = -\sum_{i=1}^{N} \delta_i \left[F(x_i) - log\left(\sum_{j=1}^{N} I(t_j \geq t_i) e^{F(x_j)} \right) \right] \tag{17}$$

Výsledný model představuje částečně rizikovou funkci, kterou lze stejným způsobem napojit na základní kumulativní rizikovou funkci, která již byla v této práci popsána v kapitole 3.2.1.2 a využívá Nelson-Aalenův estimátor. Rozdílem oproti NN CoxPH je, že zde je pro preprocessing využit jako výchozí preprocessor numerických sloupců `StandardScaler`, namísto `Normalizer`-u, a to v důsledku testů a porovnání, kdy daná změna měla pozitivní dopad na hodnotící kritéria pouze u tohoto modelu.

3.3 Interpretovatelnost modelů

Základní CoxPH model (součást `base class`) z knihovny Lifelines[20] obsahuje funkci na vykreslení grafu, který zobrazuje hodnoty koeficientů a jejich potenciální chybovost či rozptyl hodnot pomocí boxplot-u. Z grafu lze snadno rozlišit, které vstupní proměnné mají na riziko jak největší tak i kladný a záporný vliv.

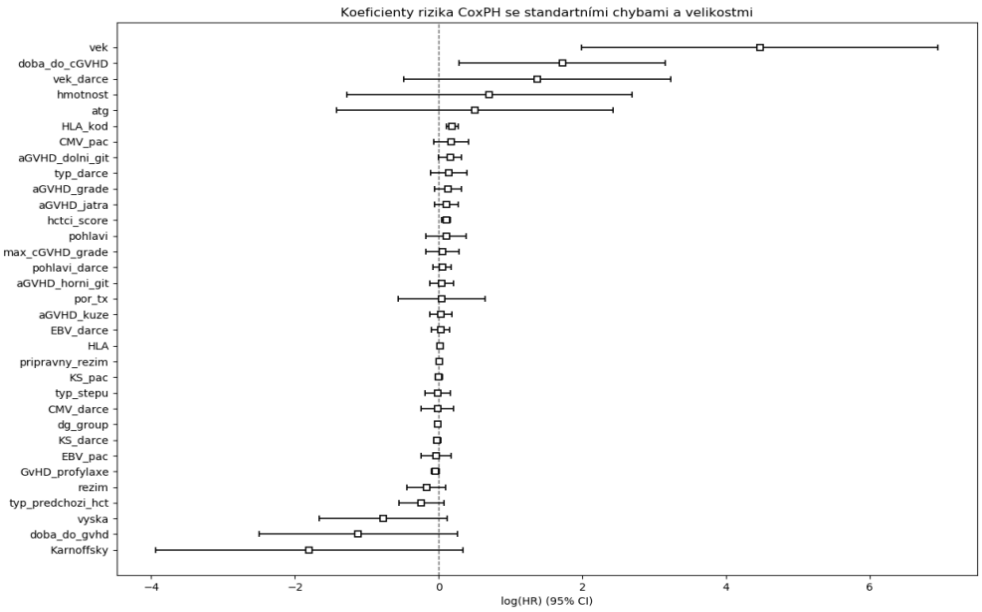
Metoda "*SHAP values*", která byla zmíněna v kapitole 2.3, je implementována knihovnou Shap [21]. Tato knihovna obsahuje rozdílné metody pro neuronové sítě (`DeepExplainer`) a rozhodovací stromové struktury (`TreeExplainer`), zároveň

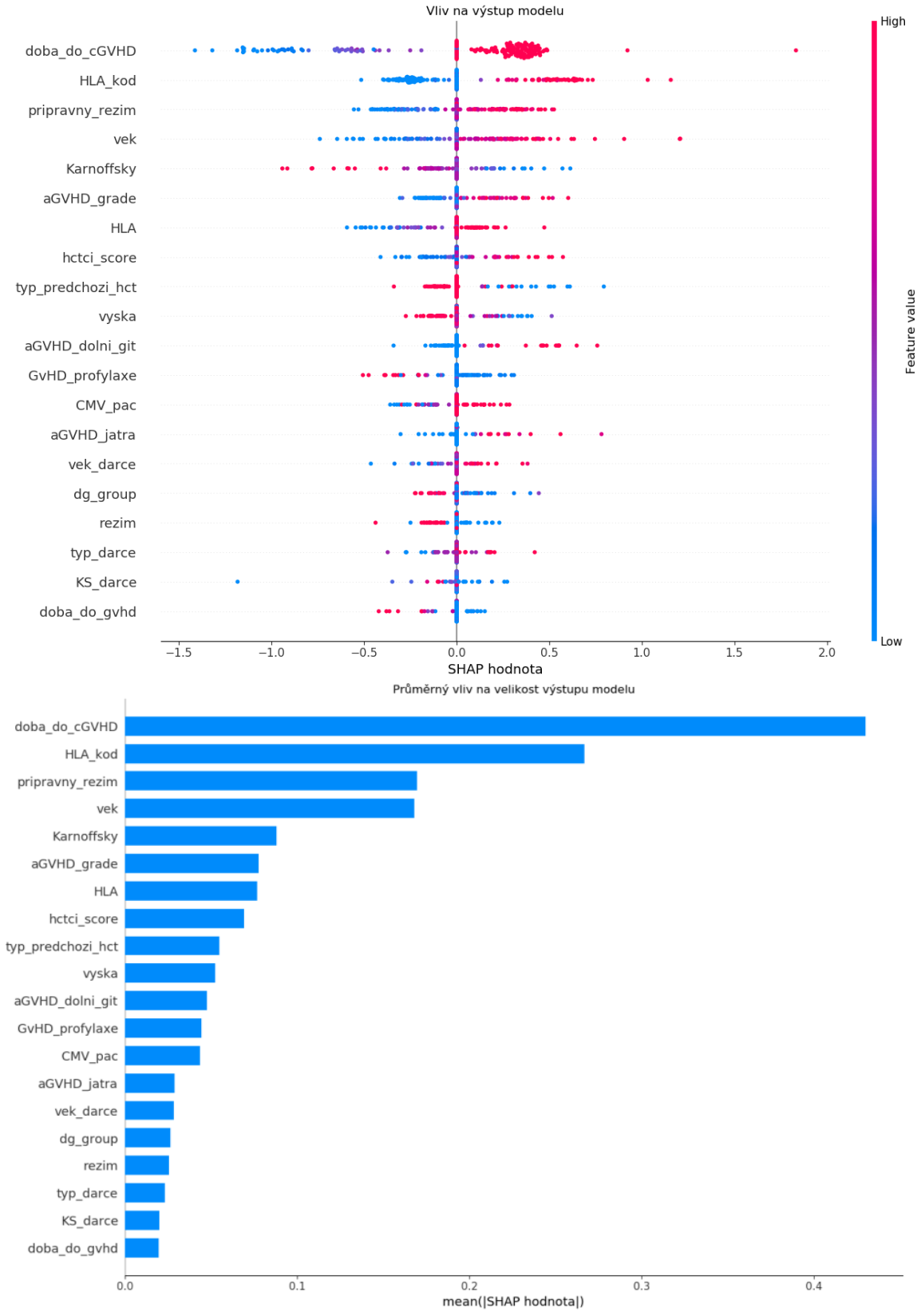
[19] *Scikit-survival* [knihovna]. [cit. 2020-05-18]. Dostupné z: https://github.com/sebp/scikit-survival
[20] *Lifelines* [knihovna]. [cit. 2020-05-18]. Dostupné z: https://lifelines.readthedocs.io/en/latest/index.html
[21] *Shap* [knihovna]. [cit. 2020-05-18]. Dostupné z: https://github.com/slundberg/shap

obsahuje i metodu, která je univerzální pro libovolný model (`KernelExplainer`). Výstupem jsou grafy, kde je znázorněno n nejvýznamnějších vstupních kovariátů. Jeden graf ukazuje přímo *SHAP* hodnoty s heatmapou znázorňující velikost původní hodnoty kovariátů - z toho lze vyčíst, zda ovlivňuje výsledek pozitivně nebo negativně v závislosti na její hodnotě. Druhý graf ukazuje jejich průměrnou absolutní hodnotu - tedy celkovou důležitost proměnné pro model.

`DeepExplainer` byl využit pro NN modely, `TreeExplainer` pro XGBoost, `KernelExplainer` pro standardní CoxPH model (pro porovnání s nativními hodnotami koeficientů rizikové funkce) a RSF s GBSA. H2O.ai obsahuje vlastní výpočet vlivu kovariátů a nelze ho napojit na *SHAP* metody. Tato hodnotící metoda byla zařazena do základní třídy estimátoru `base class` s voláním příslušných funkcí každé dědící třídy. Její prvotní inicializace je prováděna po naučení modelu pomocí učících dat, pozdější výpočet *SHAP* hodnot je prováděn na testovacích datech stejně jako zbytek evaluačních testů.

Na obrázku 10 je znázorněno porovnání *SHAP* hodnot oproti regresním koeficientům standardního CoxPH modelu z knihovny Lifelines[20]. Dokazuje, že ty nejvýznamnější kovariáty jsou detekovány oběma metodami s podobným významem. Velkou výhodou je, že pomocí navrženého preprocessingu si datasety zachovaly popis sloupců včetně jejich přiřazených *One-Hot-Encoded* kategorií se skutečným původním názvem namísto čísla kategorie či indexu proměnné v datasetu.





Obrázek 10: Porovnání SHAP vs regresní koeficienty u standardního CoxPH.

4 Hodnotící metody

Hlavním cílem modelů je co nejpřesněji zodpovědět dotaz, zda se u subjektu vyskytne cílová událost (EOI) nebo ne, případně s jakou pravděpodobností se dá očekávat, že subjekt tuto událost "přežije" po určitý počet dní nebo jiných jednotek času (TTE). V průběhu testování se z této skutečnosti vychází následujícím způsobem:

1. Modely jsou natrénovány na jiných datech, než na kterých jsou testovány (viz rozdělení train a test setu v kapitole 3.2, aby nebyly žádným způsobem ovlivněny k dosažení lepších výsledků.

2. Testovací data obsahují skutečnou požadovanou resp. očekávanou hodnotu výsledku z modelu a to jak cenzorované události, tak jejich výskyt spolu s TTE údajem.

3. Modely jsou uzpůsobeny tak, aby vrátily pravděpodobnosti přežití/selhání každého subjektu napříč TTE intervalem od 0 do nejvyšší hodnoty, která se v testovacích datech vyskytuje (manuálně lze nastavit i na jinou maximální hodnotu). To znamená např. pro vektor TTE [0,5,10,15,...] získají vektor pravděpodobností přežití [1,0.98,0.90,0.75,...] pro jeden ze subjektů, zatímco skutečný údaj tohoto subjektu je, že na čase 30 je jeho událost 1 (tedy událost se objevila v čase 30 a dále již subjekt nebyl zkoumán).

4. V použitých testovacích metodách se vždy jako cílová data používají skutečná a vůči nim se porovnávají modely predikované výstupy.

Vzhledem k vhodně implementované základní třídě modelů `base class` je způsob volání testování identický pro všechny modely. Proto byla vytvořena třída `Model_Tester` specielně pro testování většího počtu modelů, kde stačí zadat jeho třídu a nastavující *key-word-argumenty*, které nejsou povinné, ale pokud se zadají, tak by měly obsahovat i název modelu pro odlišení od ostatních v budoucích grafech. Výsledky použitých metod z kapitoy 4.1 jsou po skončení hodnocení zobrazeny graficky tak, aby byl zřejmý výsledek každého modelu v porovnání s ostatními.

4.1 Způsoby hodnocení modelů

4.1.1 Log-rank test

Tento test [5] zkoumá rozdíl mezi rizikovou funkcí 2 sérií událostí. Funguje pouze za předpokladu vzájemně proporcionální rizikové funkce (t.j. s narůstajícím časem narůstá i rozdíl v odhadech funkcí přežití) a v opačném případě jako test selhává. Nulovou hypotézu zde představuje tvrzení, že obě série mají identickou rizikovou funkci a jeho rozdělení pravděpodobnosti přibližně odpovídá chí-kvadrátu s jedním stupněm volnosti [9].

Pro tuto metodu byla využita knihovna Lifelines[22], která vypočítává jak hodnotu chí-kvadrátu, tak i p-hodnotu. Jejími vstupními daty jsou pro obě série TTE a pozorované události (EOI) $E_{0-n}(t)$, tedy podle dokumentace: durations_A, durations_B, event_observed_A, event_observed_B. Roli série A hrají skutečné údaje z testovacího setu, takže není třeba žádného post-processingu.

Sérii B jakožto predikce z modelu ve stávajícím formátu popsaném v seznamu č. 3 v kapitole 4. je třeba zpracovat tak, aby odpovídala stejnému formátu jako série A. Pro tento postup byl navržen systém meze (a.j. *threshold*) $T(t)$ a mezní hodnotou je úroveň pravděpodobnosti $P(t)$. Hledá se první t, kdy $E_i(t) = T_i(t) > P_i(t)$, tedy $E_i(t) = 1$ nebo-li True (událost nastala), poté je uloženo příslušné t na jejíž hodnotě se výsledek porovnání změnil, protože $P(0) = 1$, pokud ke změně nedošlo, tak je uloženo $E_i(t) = 0$ na maximální hodnotě t. Tento systém je aplikován pro každý subjekt i, čímž je docíleno, že velikost série B je identická s A a je tedy umožněno test provést.

Provedením Log-rank testu na intervalu meze $T < 0;\ 0.9 >$ lze znázornit, na jaké úrovni pravděpodobnosti jsou výsledky modelu nejblíže skutečnosti (testovacímu setu a sérii A). Oba výsledky z testu (p-hodnota i testová statistika) jsou ukládány a poté vykresleny do grafu. Standardně platí, že čím nižší p-hodnota, tím menší pravděpodobnost, že platí nulová hypotéza [9]. Do grafu jsou tedy zahrnuty i 5% a 1% hranice p-hodnoty, pod kterými se nulová hypotéza zamítá.

[22] *Lifelines* [knihovna]. [cit. 2020-05-18]. Dostupné z: https://lifelines.readthedocs.io/en/latest/index.html

4.1.2 Time-dependent Concordance index

Originálně se tento test nazývá jako Time-dependent discrimination index a jeho detaily jsou popsány v práci [2]. Jeho cílem je zajištění vhodných měření přesnosti, která musí být použitelná pro obecnou třídu modelů. Diskriminační analýza jako taková testuje podobnost mezi sériemi objektů, které jsou charakterizovány pozorovatelnými znaky.

Většina klinických aplikací se obecně nezabývá predikcí jednotlivých časů selhání (přežití), což původní Harrellův C diskriminační index (rozšíření oblasti pod křivkou ROC na případ cenzorovaných údajů o přežití) vyžaduje. Tato metoda je založena na nové definici shody: subjekt, který EOI vyvinul, by měl mít menší předpovězenou pravděpodobnost přežití po dobu přežití (TTE) než jakýkoliv subjekt, který přežil déle [2]. Předpovídaná funkce přežití u subjektu, který vyvinul událost, se porovná s:

1. funkcí subjektů, u kterých se událost vyvinula před časem jejího přežití
2. funkcí subjektů, u kterých se událost vyvinula nebo byla cenzorována po jeho / její době přežití. Subjekty, které byly cenzorovány, jsou zapojeny do srovnání se subjekty, u kterých se událost vyvinula před jejich pozorovanými časy. Navrhovaný index se používá k vyhodnocení diskriminační schopnosti modelu, včetně kovariátů s časově závislými účinky [2].

Testovací metoda byla implementována knihovnou PyCox[23], ze které byla zároveň použita pro tuto práci. Na rozdíl od předchozího Log-rank testu není třeba dalšího post-processingu krom transpozice (podle dokumentace mají být v řádcích predikce a ve sloupcích jednotlivé subjekty). Vstupem tedy jsou TTE skutečného datasetu a predikce přežití resp. selhání na zkoumaném časovém intervalu pro každý subjekt.

Výstupem metody je Time Dependent Concordance Index (dále zkratka TDCI), který stejně jako diskriminační index vyjadřuje vztah mezi bodovým hodnocením úlohy a hodnocením za celý test. Diskriminační analýza říká, že čím vyšší pozitivní hodnota (blíže k 1.0) indexu, tím silnější je vztah jednoho testu k celkovému hodnocení a tím

[23] *Pycox* [knihovna]. [cit. 2020-05-18]. Dostupné z: https://github.com/havakv/pycox

pádem i vyšší podobnost s trénovací množinou, kterou jsou v tomto případě EOI a TTE reálných dat. V případě negativního indexu se na dané úloze předpokládalo vyšší hodnocení.

4.1.3 Brier Score

Tato metoda je způsob, jak verifikovat přesnost předpovědi pravděpodobnosti. Brier Score (dále ve zkratce BS) test lze použít pouze v případech, kdy jsou výsledky binární tedy pravda nebo nepravda (True/False). Výsledek testu se pohybuje v intervalu $< 0;\ 1 >$ s tím, že čím nižší, tím lépe neboť hodnota 0 odpovídá naprosté přesnosti [3]. Výsledné skóre sice zodpovídá, jak přesná byla předpověď, avšak neříká, jak přesná byla v porovnání s něčím dalším.

V případě aplikování na predikci křivek přežití a pravě-cenzorovanými pozorováními je toto skóre váženo pomocí tzv. Inverzně pravděpodobnostního cenzorování (a.j. Inverse probability of censoring ve zkratce pak IPCW) [10] pro zachování originální interpretace. Studie [16] diskutuje celkový dopad cenzorování na BS a dokazuje, že může být problematický. Konkrétně v případech, kdy cenzorování může být identifikováno přímo z kovariátů - tehdy už IPCW není platné. Daná studie předkládá alternativní přístup tzv. Administrative Brier Score, které již nevyžaduje odhad cenzorované distribuce a je platné i v případě, kdy cenzorování může být identifikováno z kovariátů.

Pro implementování této metody byla opět využita knihovna PyCox[24], kde je BS jedním z výstupů testovací metody `EvalSurv`. V knihovně je implementováno jak původní BS, tak i jeho rozšíření Administrative BS. Dále zahrnuje numerickou integraci výsledků BS napříč testovanými časovými hodnotami - vzhledem k tomu, že výsledek je jedno číslo, tak ho lze považovat za užitečnou obecnou informaci pro porovnání s ostatními modely, neboť na všech časech je zachována výchozí věta "čím menší, tím lepší" a numerická integrace pouze sjednotí celý časový interval do jednoho reprezentujícího čísla.

[24] *Pycox* [knihovna]. [cit. 2020-05-18]. Dostupné z: https://github.com/havakv/pycox

Vstupní data pro tento test jsou stejná jako v případě předchozího Concordance testu. Pro vypočítání Administrative BS však je potřeba další vektor s hodnotami časů cenzorování, který lze z dostupných dat vytvořit pouze jako duplikát vektoru TTE a v tom případě pak knihovna vypisuje varování, že tyto časy cenzorování jsou pravděpodobně chybné.

4.1.4 Negative binomial log-likelihood

Další metoda, která je často využívána ve spojení s predikcí či odhadem křivek přežití. Likelihood (v č.j. věrohodnost) je statistickým aparátem, který testuje, s jakou pravděpodobností pozorované hodnoty odpovídají zvolenému rozdělení pravděpodobnosti. V případě biologické povahy dat se využívá binomického rozdělení, protože podle dlouhodobých pozorování časy přežití či selhání nejblíže odpovídají právě křivce binomického rozdělení. Výsledkem likelihood rovnice bývá pravděpodobnost, tedy $< 0; 1 >$, která po aplikaci přirozeného logaritmu umocňuje hodnoty blížící se nulové pravděpodobnosti a zároveň výsledek obrací do záporných hodnot - odtud plyne termín negativní, proto se výsledek pro lepší čitelnost a sjednocení s ostatními testy pouze násobí hodnotou -1 Rovnice (18) je speciálním tvarem log-likelihood právě pro binomický model. Tudíž znovu platí, že čím menší hodnota, tím lepší výsledek.

$$\begin{aligned} \log_e(L) &= \log_e\big(L(p|n,y)\big) \\ &= \log_e\binom{n}{y} + y \cdot \log_e(p) + (n-y) \cdot \log_e(1-p) \end{aligned} \tag{18}$$

Knihovna, implementující tuto metodu, je opět PyCox[25] a stejně jako v případě BS je zde implementována IPCW [10] verze spolu s administrativním přístupem [16] a numerickou integrací. Vstupní data jsou také identická.

4.1.5 Váhová integrace

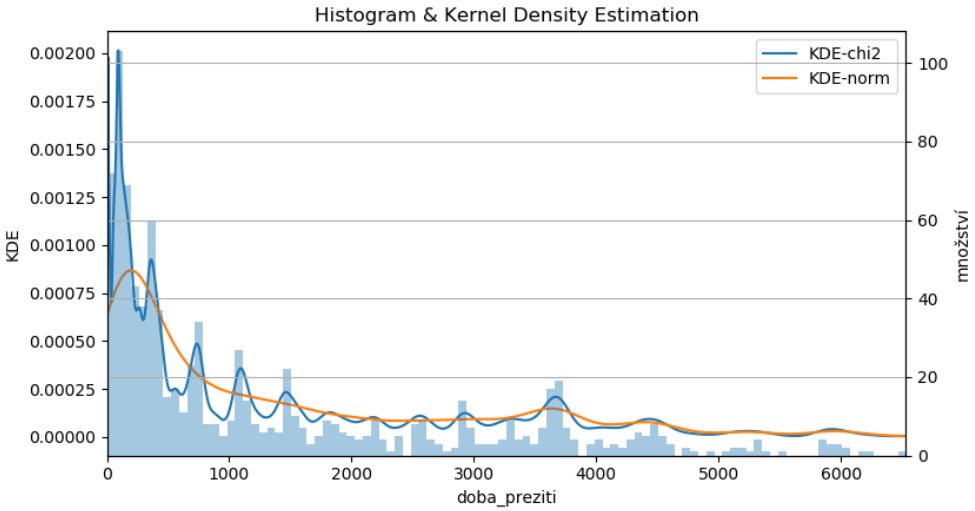
Vzhledem k nevyváženým časovým hodnotám v datech byla implementována další integrační metoda s cílem zohlednit tuto nevyváženost v datech spolu s větším zaměřením tohoto kritéria na nízké TTE, které jsou obzvlášť ve zdravotnictví důležitější

[25] *Pycox* [knihovna]. [cit. 2020-05-18]. Dostupné z: https://github.com/havakv/pycox

než vysoké. Navržený přístup využívá výpočtu *Kernel Density Estimation* (dále ve zkratce KDE) křivky, která se používá pro odhad tvaru resp. hustoty rozdělení pravděpodobnosti z histogramu dat. Touto KDE funkcí jsou poté přenásobeny hodnoty grafů BS a NBLL a integrovány pomocí lichoběžníkové integrace z knihovny SciPy[26].

Standardně se pro výpočet využívá normálního rozdělení, avšak bylo otestováno i chí-kvadrát rozdělení (porovnání viz Obrázek 11), které však data proloží příliš ostře a parametr *bandwidth* pouze posouval křivku doprava na ose x. Implementovaný přístup vychází z návodu pro knihovnu [27] a při výpočtu je ve vyhlazovacím faktoru (19) místo 1.06 použita hodnota 0.9, kvůli přesnějšímu proložení dat. Jako integrační funkce je využita lichoběžníková integrace z knihovny SciPy[26]. Následně je na výslednou funkci použit přepočet, který maximální hodnotu převede na 1 a minimální na 0 - `MinMaxScaler` z knihovny Sklearn[28] a to z důvodu, že výsledné hodnoty se pohybovaly v řádech 10^{-4}, což po vynásobení výsledků BS a NBLL a následné integraci zapříčinilo neznatelné rozdíly z důvodu příliš nízkých hodnot.

$$h = \left(\frac{4\hat{\sigma}^5}{3n}\right)^{\frac{1}{5}} \approx 1.06\hat{\sigma}n^{-1/5} \tag{19}$$



Obrázek 11: Histogram a Kernel Density Estimation

[26] *SciPy* [knihovna]. [cit. 2020-05-18]. Dostupné z: https://docs.scipy.org/doc/scipy/reference/index.html
[27] *Seaborn: statistical data visualization* [knihovna]. [cit. 2020-05-18]. Dostupné z: https://seaborn.pydata.org/index.html
[28] *Scikit-learn* [knihovna]. [cit. 2020-05-18]. Dostupné z: https://scikit-learn.org/stable/index.html

5 Výsledky

Všechny testy byly prováděny na poskytnutých reálných datech pomocí třídy `Model_tester`. Tato třída implementuje *cross-validation* (v č.j. křížovou validaci) pomocí metody `K-Fold` z knihovny Sklearn[29] s rozdělením na 5 částí, což znamená, že každý model je natrénován a hodnocen 5x, pokaždé na jiných datech, čímž vznikne 5 objektů, každý jiný, ale dohromady pojmou celý dostupný dataset. Poměr odpovídá 80% učících dat a 20% validačních.

Hodnocení modelu je stále prováděno na jiných než učících datech a proto lze výsledky předpovědí přežití sloučit, přiřadit k nim data, ze kterých vzešla, a až poté aplikovat hodnotící metody popsané níže. Tímto přístupem se získá znatelně větší statistická významnost výsledků, než kdyby bylo využito pouze 1 učení a testování na náhodně promíchaných a vybraných datech. AutoML bylo omezeno časově na maximálně 10 minut.

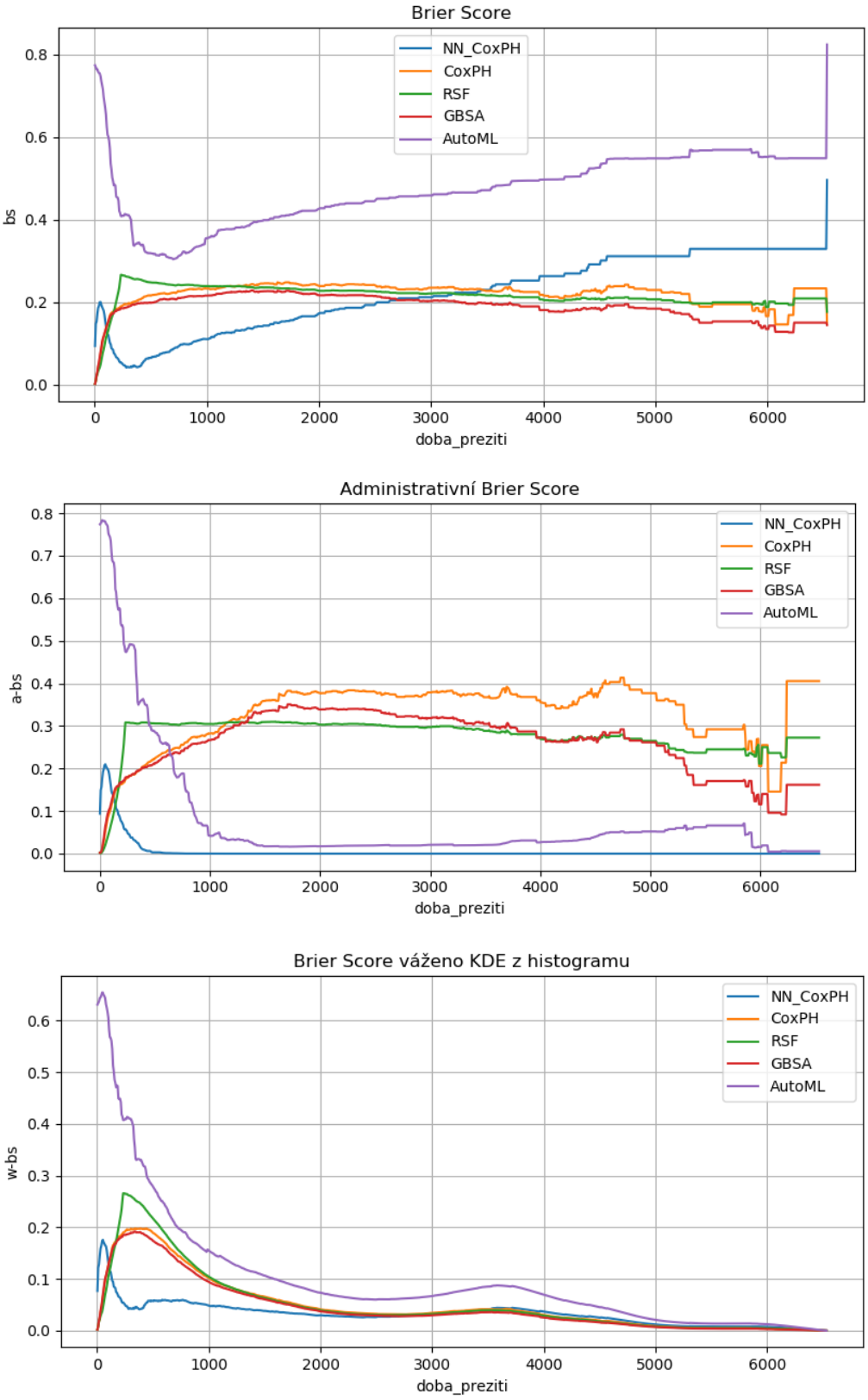
5.1 Porovnání všech modelů

V následující kapitole 5.1.1 je zobrazeno finální porovnání kvality a přesnosti jednotlivých modelů (legendy jsou umisťovány automaticky bez možnosti jejich pozdějšího posunu). EOI je “pacient_zemrel” a TTE “doba_preziti”.

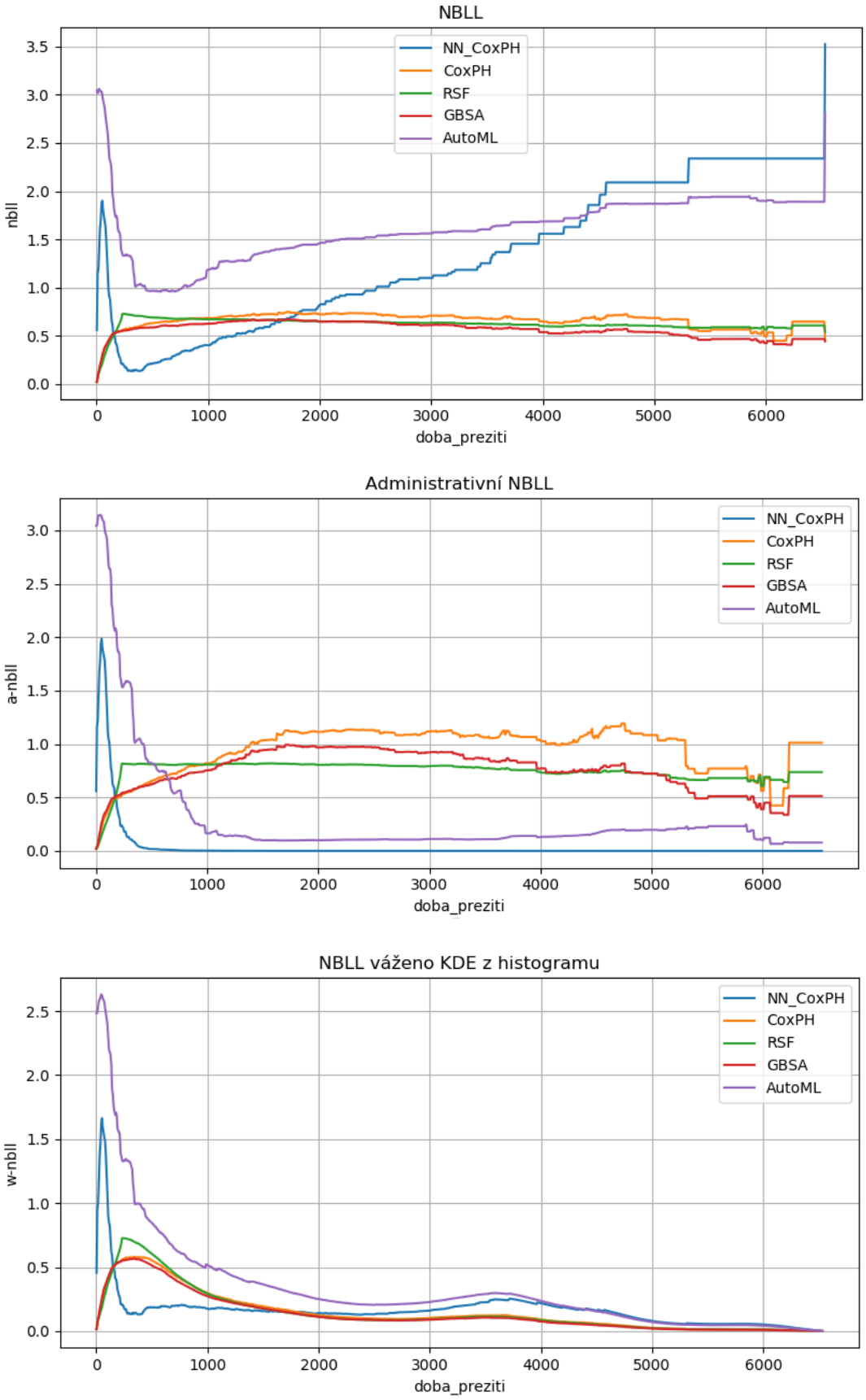
Standardní NN a XGB model byly z grafu vyřazeny, protože jejich výsledky vycházely příliš špatně ($TDCI < 0.5$ zatímco ostatní modely > 0.7, zamítnutá nulová hypotéza log rank testu pro všechny meze, značné rozdíly v integracích oproti ostatním modelům) a zároveň se grafy s tímto počtem modelů stávaly nepřehledné, nicméně p-hodnota jejich Log-Rank testu dosáhla i na 0.4. Rozbor porovnání je v kapitole 5.1.2.

[29] *Scikit-learn* [knihovna]. [cit. 2020-05-18]. Dostupné z: https://scikit-learn.org/stable/index.html

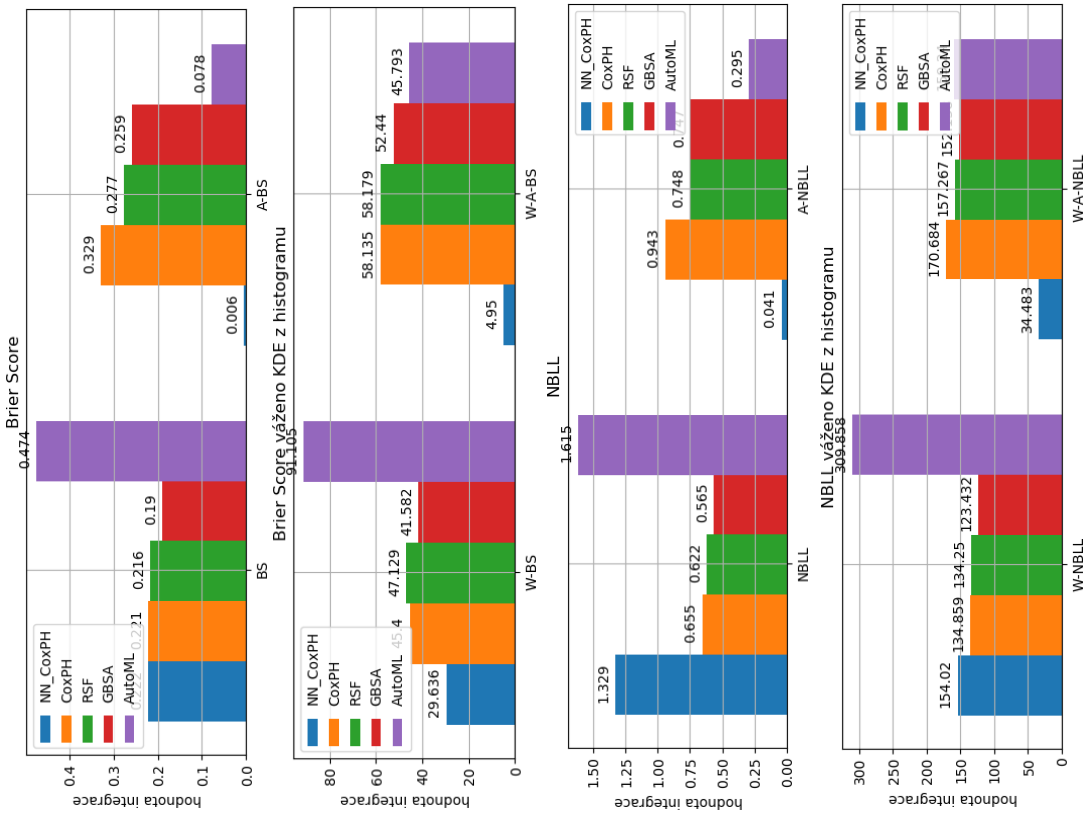
5.1.1 Grafické výstupy



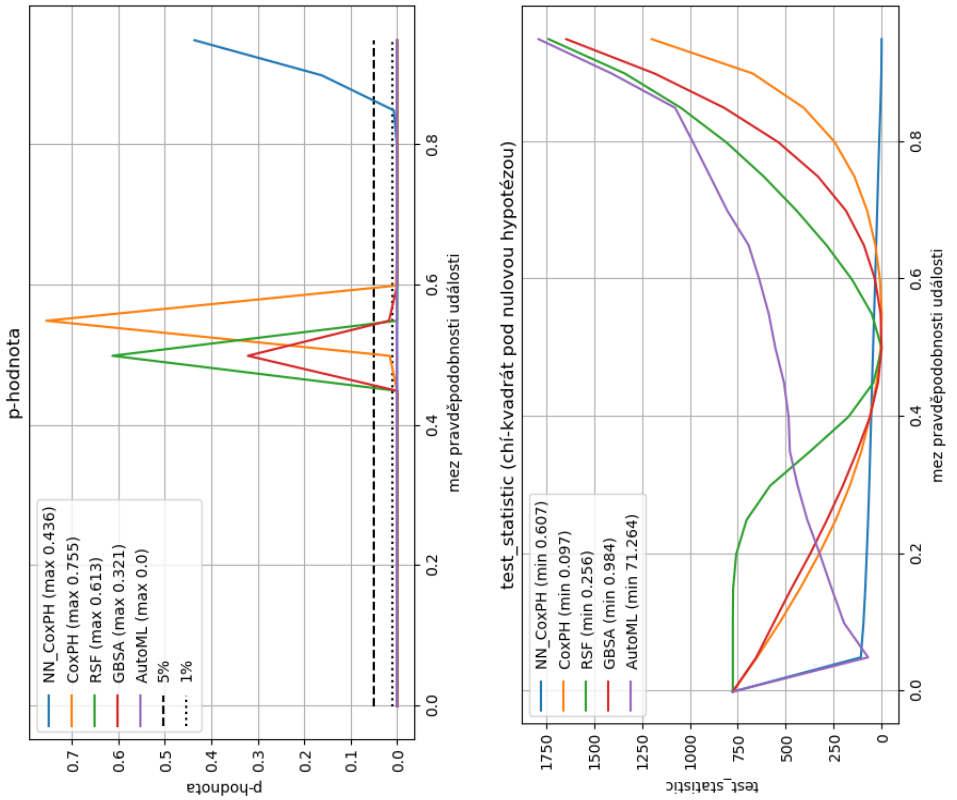
Obrázek 12: Porovnání Brier Score všech modelů



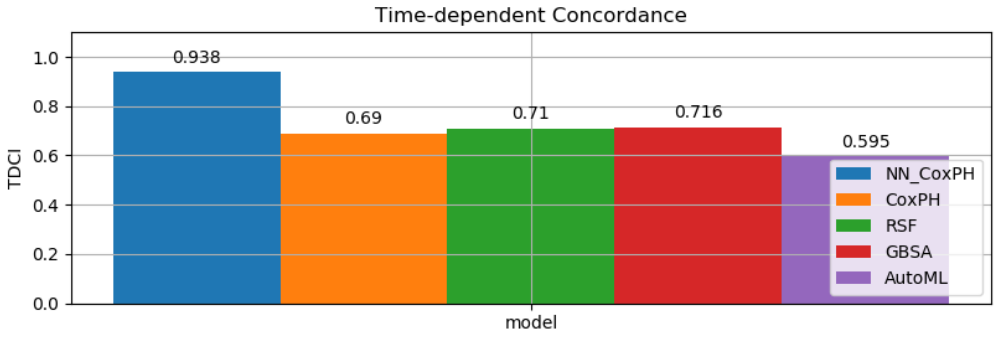
Obrázek 13: Porovnání NBLL všech modelů



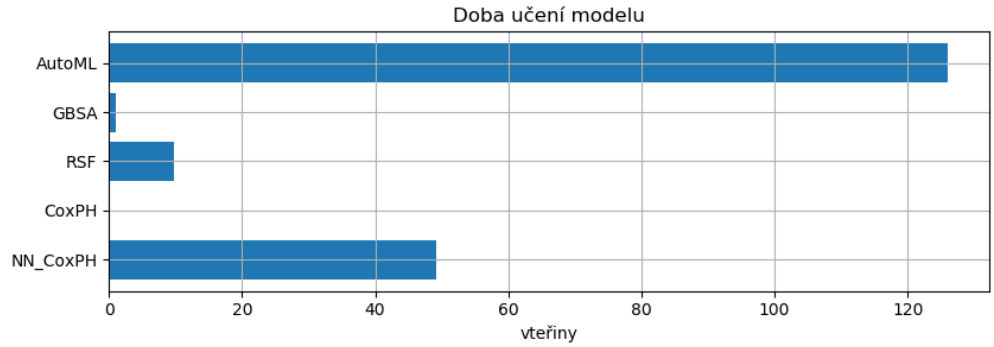
Obrázek 14: Porovnání integrací křivek hodnotících metod



Obrázek 15: Porovnání Log-Rank testu všech modelů



Obrázek 16: Porovnání TDCI všech modelů



Obrázek 17: Porovnání délky učení modelů

5.1.2 Rozbor výsledných grafů

1. NN_CoxPH má nejlepší TDCI, avšak administrativní hodnoty BS a NBLL jsou velice nízké, protože křivka se blíží nulovým hodnotám po většinu x-ové osy, zatímco hodnoty základního BS a NBLL jsou nejvyšší - to pravděpodobně indikuje, že se na tyto integrované hodnoty nedá spolehnout. Rozhodnutí poté poskytuje Log-Rank test, kde je až do meze ~0.85 nulová hypotéza identické rizikové funkce s reálnými daty zamítnuta na hladině významnosti 1%.

2. AutoML nedosahuje na kvalitu standardního CoxPH modelu v žádném z použitých testů. Nulová hypotéza Log-Rank testu je zamítnuta v celém rozsahu meze. Hodnota TDCI je výrazně menší, než ostatní metody. Zajímavostí však je, že před vyloučením NRM, DFS a relapsu ze vstupních proměnných byla i tato metoda validní a ačkoliv nedosáhla na kvalitu standardního CoxPH modelu, tak se mu přibližovala.

3. RSF model poskytuje nejlepší výsledky na nízkých hodnotách časů události/cenzorování, dokonce lepší než klasický CoxPH a to až do ~150 dní, což jsou pouze 2% ze škály testovaných (a dostupných) časů. Od času přibližně 1000 dní (20%) se jeho BS pohybuje na podobné a dokonce o trochu lepší úrovni jak CoxPH, pro NBLL je to podobné. V integrovaných hodnotách pak BS i NBLL vychází lépe s tím, že administrativní i vážené přístupy tento rozdíl pouze umocňují. Log-Rank test neprokázal, že se významně liší RSF základní riziková funkce od reálných dat na mezi od 0.45 do 0,55, kde jeho p-hodnota dokonce dosáhla 0.613 (CoxPH maximum je 0.755). TDCI vychází lépe než CoxPH o 0,02.

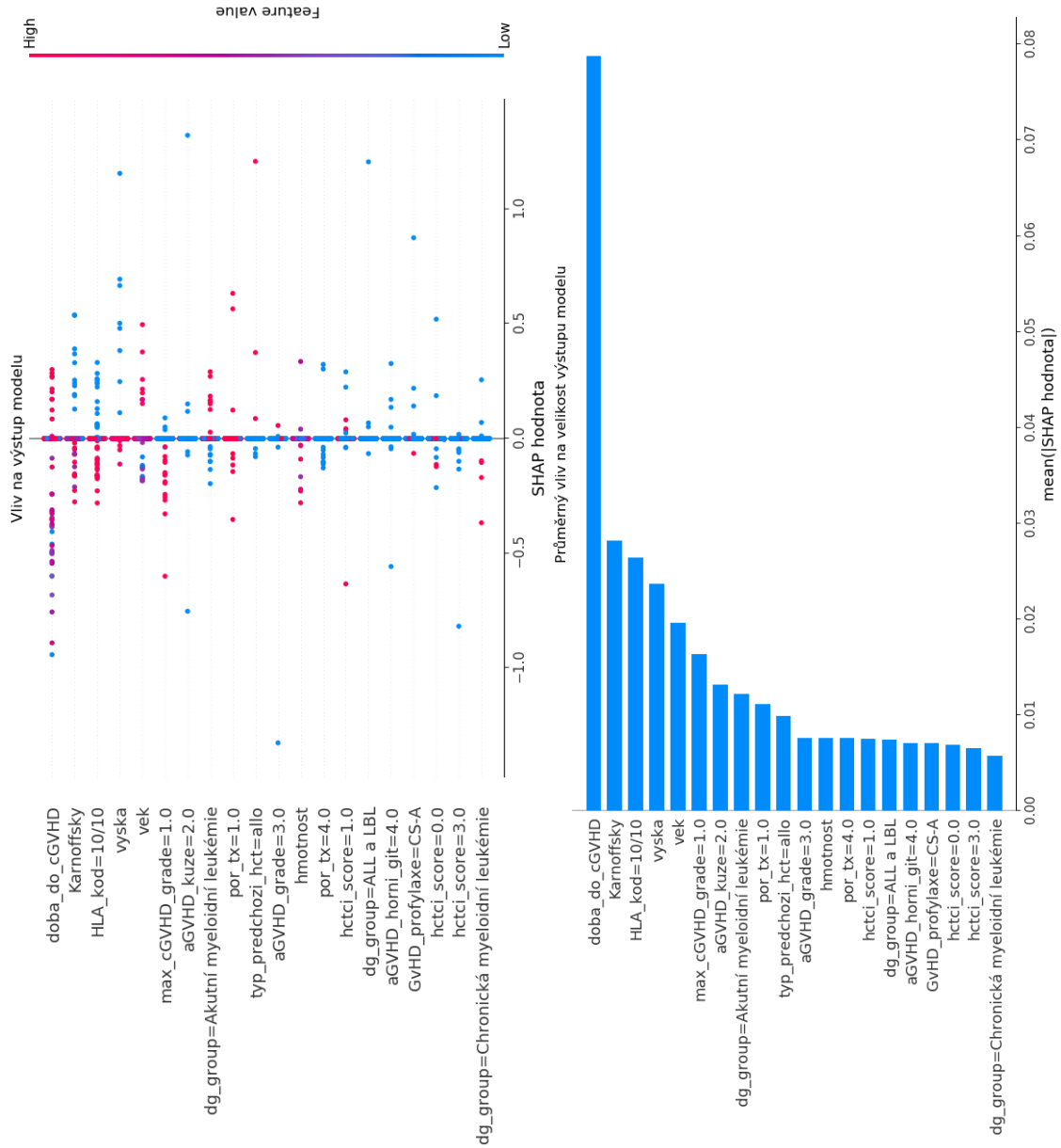
4. Nakonec GBSA. TDCI této metody sice nevyšlo nejlépe (2. místo), ale převyšuje standardní CoxPH o $2 \cdot 10^{-2}$. Ve všech integračních grafech vychází GBSA nejlépe, kromě administrativních verzí, kde je z důvodu 0, a pravděpodobně chybných hodnot NN_CoxPH a stejně tak neplatnému AUTOML až na 3. místě. Na BS křivce se GBSA nachází pod CoxPH od ~150 (pro NBLL je to od ~400 dní), kdy administrativní i vážené grafy tento rozdíl podporují. Výsledek Log-rank testu nevyvrací nulovou hypotézu podobně jako CoxPH, s rozdílem, že je tomu na mezi 0.5 až 0.55, zatímco CoxPH mezi 0.5 a 0.6 a jeho maximální hodnota odpovídá 0.321, což je téměř polovina maximální hodnty CoxPH.

Z porovnání tedy vyplývá, že CoxPH adaptace gradient boosted regresními stromovými strukturami podává lepší výsledky než standardní CoxPH model a to na převážné většině časového intervalu, kde je výjimkou pouze počátečních ~100 dní (1.5%) podle NBLL a ~150 dní (2.2%) podle BS. Na rozdíl má pravděpodobně vliv malý počet učících dat pro tyto vyšší TTE (viz Obrázek 20).

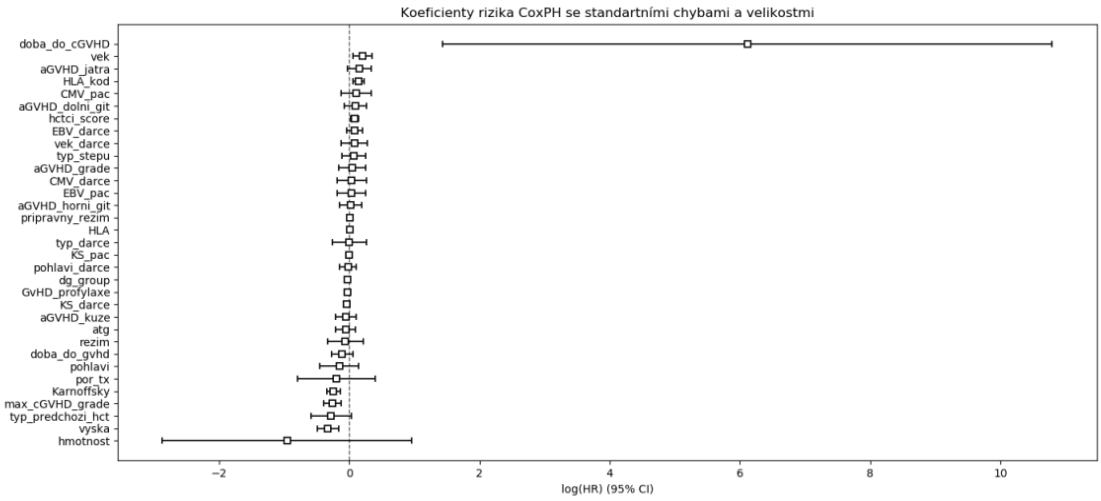
Test se změnou cílového EOI na relaps a TTE na "*doba_do_relapsu*" však prokázal, že i NN_CoxPH je použitelný (viz příloha B Grafy vlivu velikosti učícího datasetu). Příslušné výsledky dokazují podobné výsledky a modelů jako byly popsány v této kapitole a zároveň díky Log-rank testu ukazují, že pomyslné pořadí se může zcela otočit a zde nepoužitelný NN_CoxPH zdaleka předčí ostatní modely.

5.2 Vliv proměnných na predikci a porovnání s CoxPH

Obrázek 18 a obrázek 19 zobrazují vliv vstupních proměnných na výsledné riziko prodělání EOI “pacient_zemrel”. Lze si všimnout, že jak CoxPH regresní koeficienty, tak SHAP hodnoty GBSA modelu vyhodnotily vstupní parametr “doba_co_cGVHD” jako největší vliv na úmrtí pacienta. Z SHAP hodnot lze vyčíst, že nízké hodnoty daného parametru mají tendency riziko snižovat, zatímco vysoké naopak. U ostatních vstupních proměnných však moc podobností nalézt nelze.



Obrázek 18: Vliv vstupních proměnných GBSA modelu na riziko



Obrázek 19: Vliv vstupních proměnných CoxPH na riziko

5.3 Vliv velikosti datasetu na kvalitu modelů

V příloze na obrázku 22 a obrázku 23 je zobrazeno, jak se výsledné hodnotící metody (integrační verze, TDCI a p-hodnota Log-ranku) liší v závislosti na velikosti učícího datasetu. První z nich je proveden simulací na reálných datech jako v předchozích testech, zatímco druhý je výsledek simulací na `sac_admin5` datasetu z knihovny PyCox[30]. Dohromady tyto grafy dokazují, že nelze striktně prohlásit, že je jeden typ modelu lepší než druhý, neboť každý může vycházet lépe pro jiný dataset a obzvlášť při rozdílných učících velikostech.

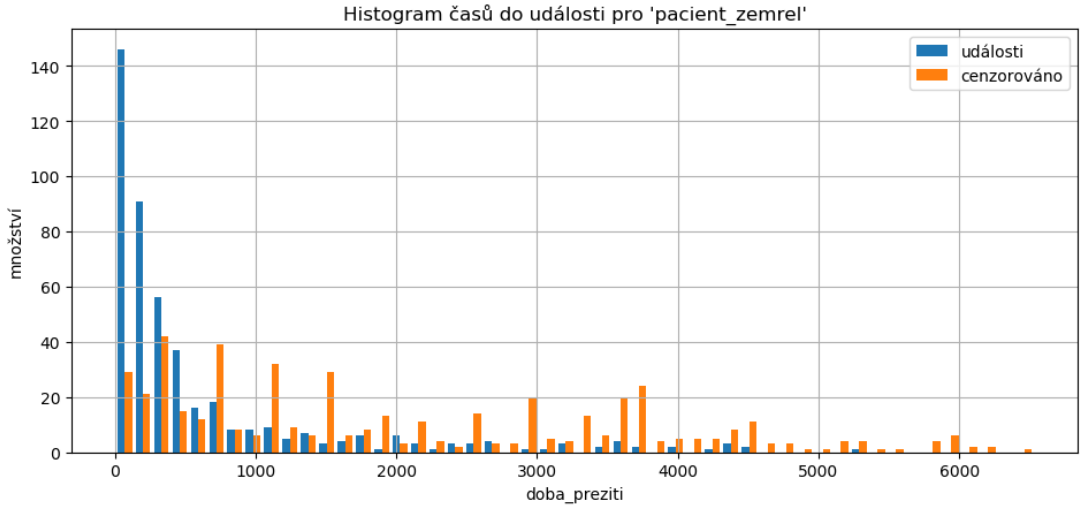
5.4 Diskutování ensemble modelu

Jak dokazuje obrázek 20, právě prvních cca 500 dní je v dané úloze nejpodstatnějších, protože se zde objevuje největší výskyt prodělaných EOI. Ze zjištěných výsledků lze odvodit teoretickou myšlenku, kdy se sestaví jeden model z několika implementovaných modelů. Každý z nich by pak byl využit na jiné části časového intervalu, podle automatického výběru na základě hodnotících kritérií. Takový přístup je dále podpořen přílohou B Grafy vlivu velikosti učícího datasetu, kde výsledky vychází výrazně jinak po změně cílového EOI.

[30] *Pycox* [knihovna]. [cit. 2020-05-18]. Dostupné z: https://github.com/havakv/pycox

Při testování vlivu velikosti učícího datasetu bylo zároveň zjištěno, že pořadí modelů v žebříčku kvality se s velikostí datasetu liší. Proto by takový automatizovaný výběr mohl poskytnout univerzální přístup, který by dokázal podávat stabilní a přesnější výsledky, než pouze jeden vybraný model, a to dokonce se zohledněním TTE histogramu.

Výběr by byl řešen podle toho, která hodnotící kritéria na daném čase t vycházejí nejlépe (ideálně se zprůměrují v místech křížení daných křivek). V této úloze by tak prvních ~100 dní bylo řešeno pomocí RSF a zbytek GBSA. Zahrnutí NN_CoxPH mezi 100 a 600 dny by byla otázka dalšího testování. Složená by bylo možné řešit automaticky, lze však předpokládat, že výsledné křivky přežití nebudou čistě klesající, proto by bylo třeba je proložit vyhlazující klesající funkcí. Tento návrh nebylo možné do této práce zahrnout z důvodu jejího limitovaného rozsahu.



Obrázek 20: Histogram časů událostí a cenzorování

6 Závěr

V teoretické části byly podrobně popsány jednotlivé metody, které se v současné době používají pro výpočet a analýzu křivek přežití nejen ve zdravotnictví. Byl kladen důraz na rozbor jejich uplatnění a způsoby využívání ve zdravotnictví, kde jsou nedílnou součástí léčby pacientů, analýzy a hodnocení nových i stávajících studií a metodik. Součástí dané kapitoly je i praktická ukázka použití.

V návaznosti na popis vlivu vstupních kovariátů na výsledek modelu, který Coxův model proporcionálních rizik obsahuje v jednotlivých regresních koeficientech, byla uvedena "*black-box*" problematika modelů umělé inteligence. Vybrané řešení představuje studie [17] sjednocující několik algoritmů, které samostatně dokáží interpretovat příslušné modely umělé inteligence, pomocí tzv. "*SHAP values*". Obrázek 10 porovnává tyto *SHAP* hodnoty s regresními koeficienty CoxPH modelu a dokazuje, že použití této metody je validní, protože většina nejvíce ovlivňujících proměnných byla detekována v regresních koeficientech i *SHAP* hodnotách podobně.

V rámci této práce byl vytvořen vlastní přístup k předzpracování dat (*preprocessing*) viz kapitola 3.1.2. Jedním z hlavních důvodů byl problém poskytnutých reálných dat, kdy se po zpracování klasickými postupy velikost (resp. délka) použitelných dat zmenšila o 97,5% (z původních 1385 na 35), zatímco tímto přístupem pouze o 32,5% (z 1385 na 935). Dalším důvodem byla ztráta názvu sloupců, která by učinila zjištění vlivu proměnných na výsledek zbytečným, neboť by byly zobrazeny pouze nic neříkající indexy.

V práci byly implementovány následující modely:

1. běžně používaný Coxův model proporcionálních rizik
2. klasická neuronová síť s aktivační funkcí sigmoid ve výstupní vrstvě (cíl je pravděpodobnost resp. jistota modelu, že se událost projeví)
3. klasifikační neuronová síť s aktivační funkcí softmax ve výstupní vrstvě (rozhodnutí, zda se událost projeví nebo ne)
4. adaptace CoxPH pomocí nahrazení částečně-rizikové funkce neuronovou sítí a místo metody parciální věrohodnosti jako *loss* funkce použita *average-*

negative-log-likelihood (10) doplněná o regularizaci L2, což pochází ze studie [13]

5. základní XGBoost model, kde snaha o podobné využití na adaptaci CoxPH jako NN selhala na implementování funkce pro výpočet gradientu kvůli příslušné knihovně, která to neumožňuje – vysvětleno v kapitole 3.2.2.2
6. Automatic Machine Learning společnosti h2o.ai (cílem bylo využít *state-of-art* algoritmy pro výběr a parametrizaci modelů)
7. Random Survival Forest [11]
8. Gradient Boosting Survival Analysis model, který představuje částečně-rizikovou funkci a je doplněn Nelson-Aalen estimátorem pro odhad základní kumulativní rizikové funkce (stejně jako NN adaptace) což dohromady dává další adaptaci CoxPH modelu

Pro hodnocení výsledků kvality a přesnosti jednotlivých modelů bylo využito následujících algoritmů: Log-rank [5], Time-dependent Concordance [2], Brier Score [16], Negative binomial log-likelihood [16], vážení těchto grafů pomocí Kernel Density Estimation z histogramu a nakonec integrace daných grafů pro ucelený přehled. Každý z těchto testů je prováděn na stejných datech a porovnává výsledek modelu vůči skutečnému stavu přežití pacienta. Pro vzájemné porovnávání modelů implementovanou třídou `Model_Tester` je při učení využito křížové validace (*cross validation*) metodou `KFold`, což zajišťuje učení i testování na všech dostupných datech a díky tomu vyšší statistickou významnost, než pouze jeden test na náhodně vybraných datech.

Samotné výsledky jsou detailněji popsány v kapitole 5. Na konkrétním příkladu predikce přežití pacienta bylo prokázáno, že adaptace CoxPH pomocí nahrazení částečně rizikové funkce Gradient Boosted Survival Analysis modelem (regresní rozhodovací stromová struktura učená upravenou *log-partial-likelihood* funkcí (17)) dosahuje lepších výsledků na většině intervalu TTE časů. Na nízkých časech (do ~150dní) poté nejlépe vychází Random Survival Forest, následován NN adaptací CoxPH, kde je součástí NN vstupu i simulovaný čas události. Tento NN model však podává nestabilní výsledky a jeho nulová hypotéza Log-rank testu (identická základní riziková funkce s reálnými vzorovými daty) byla zamítnuta na intervalu 1% i 5%, avšak

při změně cílové EOI na relaps již platný je (viz příloha Obrázek 21) a dokonce vychází podstatně lépe, než ostatní metody, napříč testovacími metodami.

Porovnáním několika cílů včetně vlivu velikosti učícího datasetu se došlo k závěru, že nelze striktně říci, že jeden typ modelu je lepší než druhý, neboť pořadí pomyslného žebříčku je proměnné s učícími daty, cílem, ale i jejich velikostí. Avšak ve většině případů docílí AI modely lepších výsledků než standardní CoxPH. Proto byl diskutován návrh ensemble modelu (viz kapitola 5.4). Ten však nebylo možné realizovat z důvodu omezeného rozsahu práce. Může tedy být předmětem další studie v oblasti umělé inteligence a křivek přežití.

Použitá literatura

[1] ANGWIN, Julia, Jeff LARSON, Surya MATTU a Lauren KIRCHNER. Machine Bias. *ProPublica* [online]. [cit. 2020-05-18]. Dostupné z: https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

[2] ANTOLINI, Laura, Patrizia BORACCHI a Elia BIGANZOLI. A time-dependent discrimination index for survival data. *Statistics in Medicine* [online]. 2005, **24**(24), 3927-3944 [cit. 2020-05-18]. DOI: 10.1002/sim.2427. ISSN 0277-6715. Dostupné z: http://doi.wiley.com/10.1002/sim.2427

[3] *BENEDETTI, Riccardo. Scoring Rules for Forecast Verification. In: Monthly Weather Review [online]. 2010, 2020, **138**(1), s. 203-211 [cit. 2020-05-21]. DOI: 10.1175/2009MWR2945.1. ISSN 0027-0644. Dostupné z: http://journals.ametsoc.org/doi/10.1175/2009MWR2945.1*

[4] Bishop, C. Pattern Recognition and Machine Learning. 2006. ISBN 13: 978-038731073

[5] BLAND, J Martin a Douglas G ALTMAN. The logrank test. *BMJ* [online]. 2004, **328**(7447) [cit. 2020-05-17]. DOI: 10.1136/bmj.328.7447.1073. ISSN 0959-8138. Dostupné z: http://www.bmj.com/lookup/doi/10.1136/bmj.328.7447.1073

[6] ETIKAN, İlker a Ogunjesa BABATOPE. *Survival Analysis: A Major Decision Technique in Healthcare Practices* [online]. [cit. 2020-05-17]. Dostupné z: http://ijsrm.humanjournals.com/wp-content/uploads/2018/03/12.%C4%B0lker-Etikan-Ogunjesa-Babatope.pdf

[7] FARAGGI, David a Richard SIMON. A neural network model for survival data. *Statistics in Medicine* [online]. 1995, **14**(1), 73-82 [cit. 2020-05-18]. DOI: 10.1002/sim.4780140108. ISSN 02776715. Dostupné z: http://doi.wiley.com/10.1002/sim.4780140108

[8] Goodfellow, Ian, Yoshua Bengio a Aaron Courville. Deep learning. The MIT Press 2016. ISBN 0262035618

[9] HOLČÍK, Jiří, KOMENDA, Martin (eds.) a kol. *Matematická biologie: e-learningová učebnice* [online], 1. vydání. Brno: Masarykova univerzita, 2015. ISBN 978-80-210-8095-9 Dostupné z: https://portal.matematickabiologie.cz/

[10] CHAKLADAR, Sujatro, Michael G. HUDGENS, M. Elizabeth HALLORAN, John D. CLEMENS, Mohammad ALI a Michael E. EMCH. Inverse Probability Weighted Estimators of Vaccine Effects Accommodating Partial Interference and Censoring. *Arxiv* [online]. [cit. 2020-05-21]. Dostupné z: https://arxiv.org/pdf/1910.03536.pdf

[11] ISHWARAN, Hemant, Udaya B. KOGALUR, Eugene H. BLACKSTONE a Michael S. LAUER. Random survival forests. *The Annals of Applied Statistics* [online]. 2008, **2**(3), 841-860 [cit. 2020-05-18]. DOI: 10.1214/08-AOAS169. ISSN 1932-6157. Dostupné z: http://projecteuclid.org/euclid.aoas/1223908043

[12] KANE, Frank. Machine Learning, Data Science and Deep Learning with Python. *Udemy* [online]. [cit. 2020-05-17]. Dostupné z:

https://www.udemy.com/course/data-science-and-machine-learning-with-python-hands-on/

[13] KATZMAN, Jared L., Uri SHAHAM, Alexander CLONINGER, Jonathan BATES, Tingting JIANG a Yuval KLUGER. DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network. *BMC Medical Research Methodology* [online]. 2018, **18**(1) [cit. 2020-05-18]. DOI: 10.1186/s12874-018-0482-1. ISSN 1471-2288. Dostupné z: https://bmcmedresmethodol.biomedcentral.com/articles/10.1186/s12874-018-0482-1

[14] KINGMA, Diederik P. a Jimmy Lei BA. Adam: A Method for Stochastic Optimization. *Arxiv* [online]. [cit. 2020-05-17]. Dostupné z: https://arxiv.org/pdf/1412.6980.pdf

[15] KLUSOWSKI, Jason M. Analyzing CART. In: *Arxiv* [online]. 2020 [cit. 2020-05-21]. Dostupné z: https://arxiv.org/pdf/1906.10086.pdf

[16] KVAMME, Havard a Ørnulf BORGAN. The Brier Score under Administrative Censoring: Problems and Solutions. *Arxiv* [online]. [cit. 2020-05-18]. Dostupné z: https://arxiv.org/pdf/1912.08581.pdf

[17] LUNDBERG, Scott M. a Su-In LEE. A Unified Approach to Interpreting Model Predictions. *Arxiv* [online]. [cit. 2020-05-18]. Dostupné z: https://arxiv.org/pdf/1705.07874.pdf

[18] MIOTTO, Riccardo, Li LI, Brian A. KIDD a Joel T. DUDLEY. Deep Patient: An Unsupervised Representation to Predict the Future of Patients from the Electronic Health Records. *Scientific Reports* [online]. 2016, **6**(1) [cit. 2020-05-18]. DOI: 10.1038/srep26094. ISSN 2045-2322. Dostupné z: http://www.nature.com/articles/srep26094

[19] PRENTICE, Ross L. Introduction to Cox (1972) Regression Models and Life-Tables. *Breakthroughs in Statistics* [online]. New York, NY: Springer New York, 1992, 1992, , 519-526 [cit. 2020-05-20]. Springer Series in Statistics. DOI: 10.1007/978-1-4612-4380-9_36. ISBN 978-0-387-94039-7. Dostupné z: http://link.springer.com/10.1007/978-1-4612-4380-9_36

[20] RIBEIRO, Marco Tulio, Sameer SINGH a Carlos GUESTRIN. "Why Should I Trust You?" Explaining the Predictions of Any Classifier. *Arxiv* [online]. [cit. 2020-05-18]. Dostupné z: https://arxiv.org/pdf/1602.04938.pdf

[21] RIDGEWAY, Greg. *The State of Boosting* [online]. [cit. 2020-05-18]. Dostupné z: http://www.ressources-actuarielles.net/EXT/ISFA/1226.nsf/0/a29acbd26d902d6fc125822a0031c09b/$FILE/boosting.pdf

[22] ROSENBLATT, F. The perceptron, a perceiving and recognizing automaton. Buffalo, NY: Cornell Aeronautical Laboratory, 1957.

[23] RUSSAKOVSKY, Olga, Jia DENG, Hao SU, et al. ImageNet Large Scale Visual Recognition Challenge. *Arxiv* [online]. 2015 [cit. 2020-05-18]. Dostupné z: https://arxiv.org/pdf/1409.0575.pdf

[24] REYNOSO, Rebecca. A Complete History of Artificial Intelligence. *Learning Hub* [online]. [cit. 2020-05-17]. Dostupné z: https://learn.g2.com/history-of-artificial-intelligence

[25] SAINANI, Kristin, Joshua WALLACH a Michael MCAULIFFE. Statistics in Medicine. Stanford Online [online]. [cit. 2019-10-09]. Dostupné z: https://lagunita.stanford.edu/courses/Medicine/MedStats./Summer2015/about

[26] SHRIKUMAR, Avanti, Peyton GREENSIDE a Anshul KUNDAJE. Learning Important Features Through Propagating Activation Differences. *Arxiv* [online]. [cit. 2020-05-18]. Dostupné z: https://arxiv.org/pdf/1704.02685.pdf

[27] TABLEMAN, Mara a Jong Sung KIM. *Survival Analysis Using S: Analysis of Time-to-Event Data: Chapman & Hall/CRC Texts in Statistical Science*. Ilustrované vydání. CRC Press, 2003. ISBN 0203501411.

[28] TURING, A. M. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind* [online]. 1950, **LIX**(236), 433-460 [cit. 2020-05-17]. DOI: 10.1093/mind/LIX.236.433. ISSN 1460-2113. Dostupné z: https://academic.oup.com/mind/article/LIX/236/433/986238

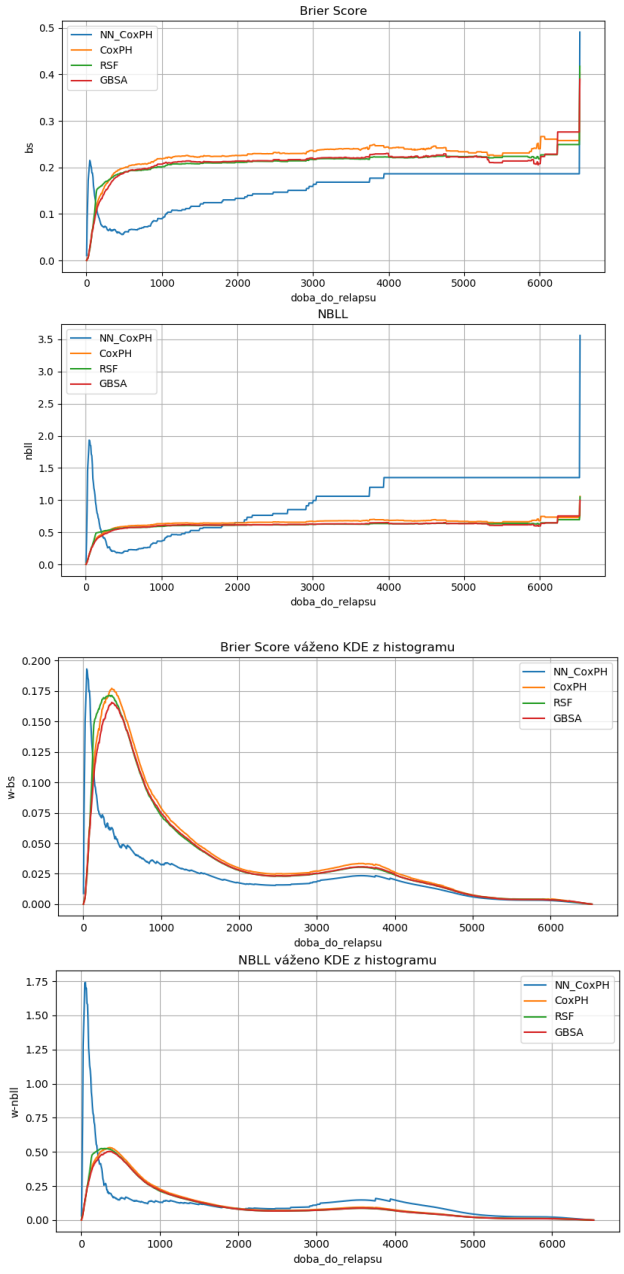
[29] WALTERS, Stephen. *What is a Cox model?* [online]. [cit. 2020-05-17]. Dostupné z: https://www.whatisseries.co.uk/what-is-a-cox-model/

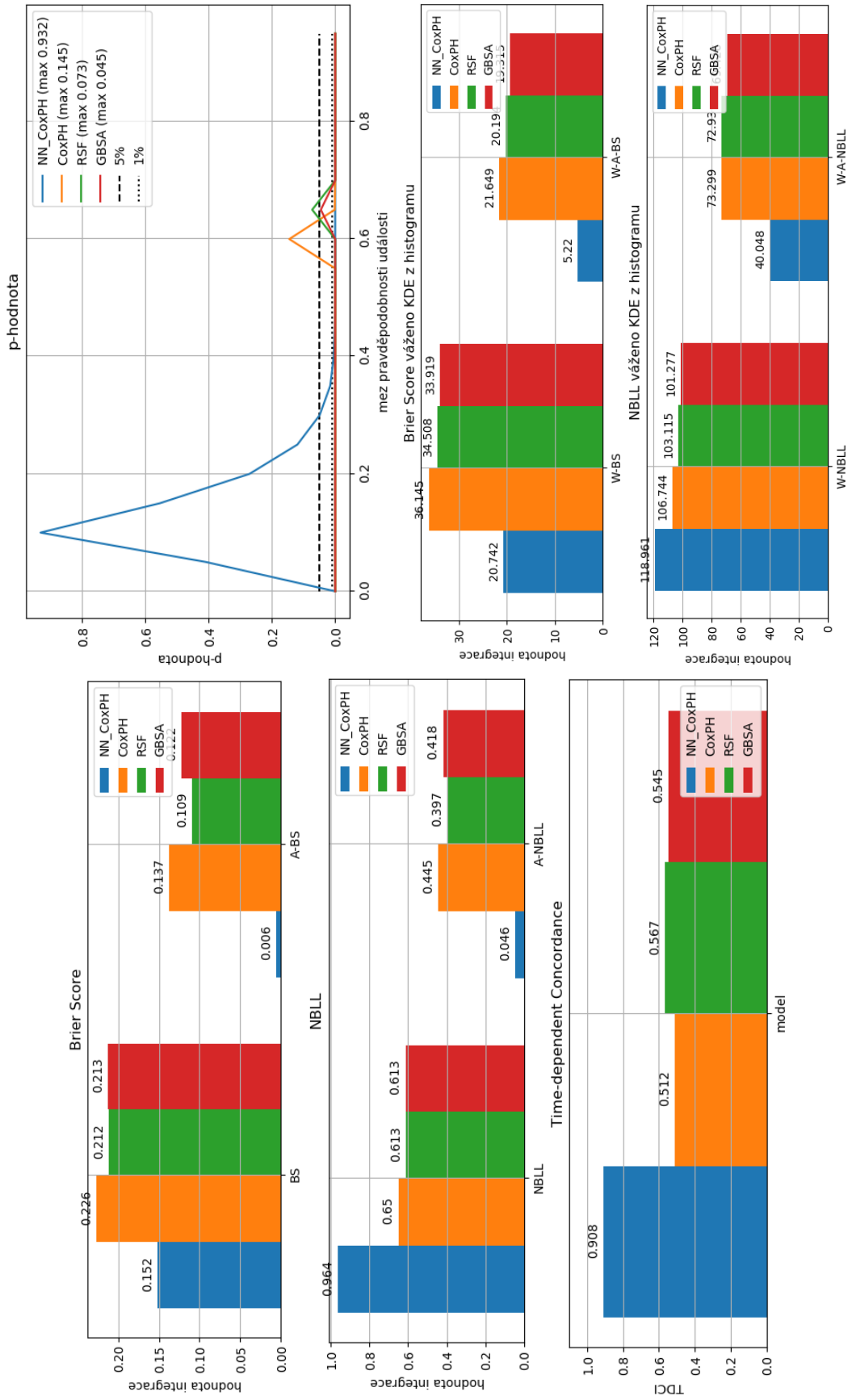
[30] XIA, Fang, Jing NING a Xuelin HUANG. *Empirical Comparison of the Breslow Estimator and the Kalbfleisch Prentice Estimator for Survival Functions* [online]. 2018, **09**(02) [cit. 2020-05-18]. DOI: 10.4172/2155-6180.1000392. ISSN 21556180. Dostupné z: https://www.omicsonline.org/open-access/empirical-comparison-of-the-breslow-estimator-and-the-kalbfleisch-prentice-estimator-for-survival-functions-2155-6180-1000392-99906.html

Přílohy

A Výsledky ostatních cílů

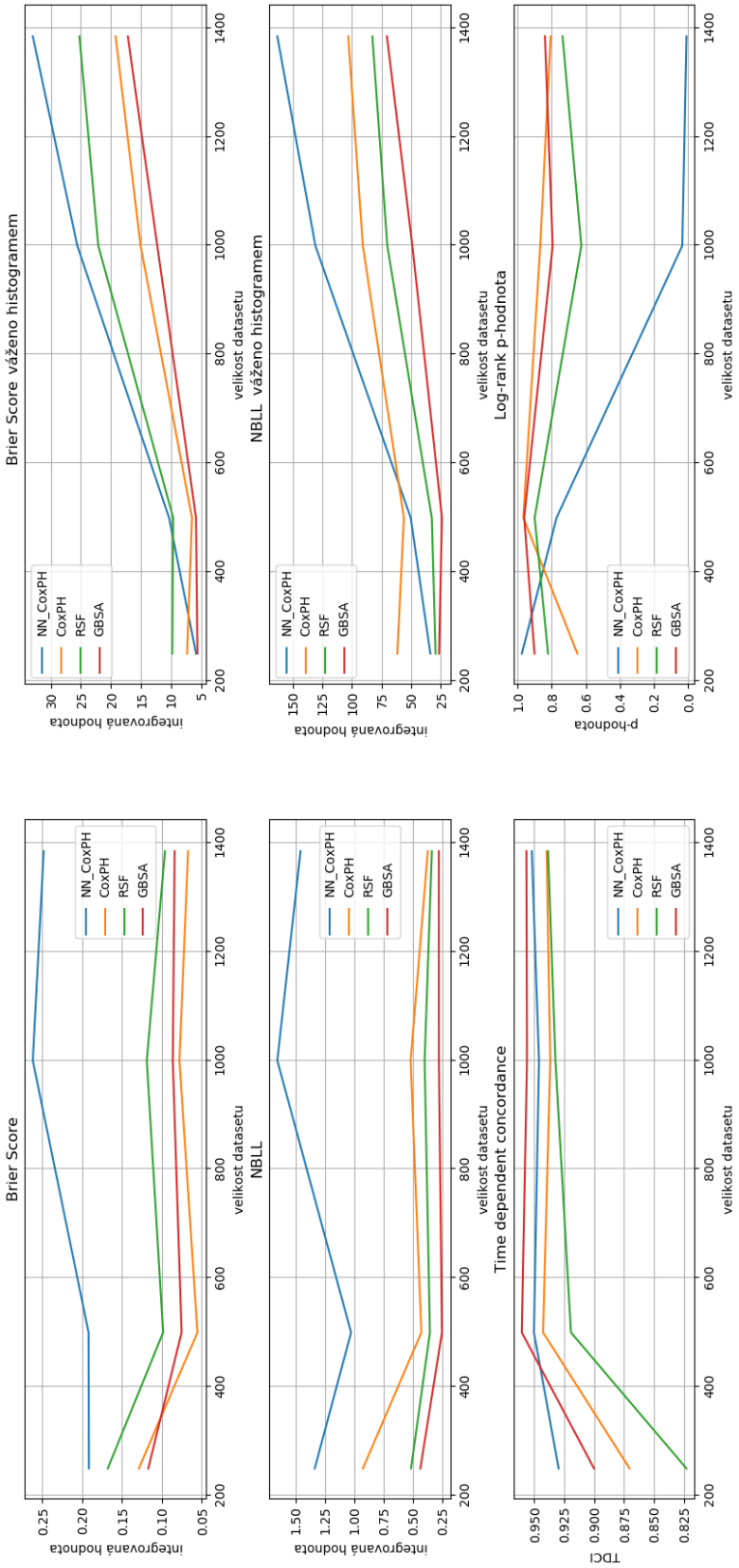
Z časových důvodů byl vyloučen AutoML, jehož *cross validace* trvala extrémně dlouho. Dále z důvodu rozsahu práce jsou zde zobrazeny pouze nejpodstatnější grafy – ostatní budou na přiloženém CD. Cíl aGVHD a max_cGVHD se nepodařilo zpracovat kvůli problému s výpočtem statistik, který skončil chybou, že TTE není monotónní a nezbyl čas na řešení tohoto problému.



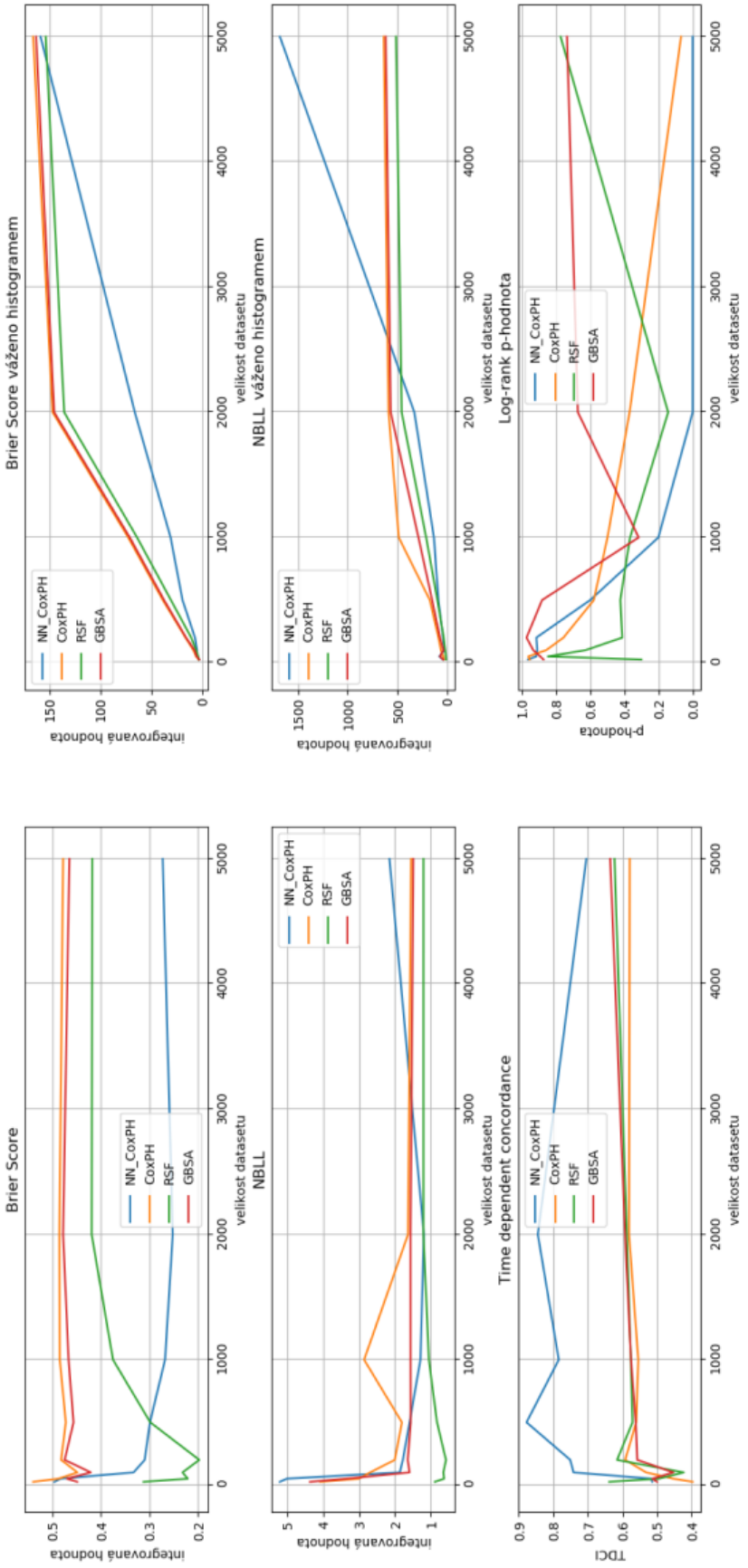


Obrázek 21: Výsledky EOI „relaps“ s TTE „doba_do_relapsu“

B Grafy vlivu velikosti učícího datasetu



Obrázek 22: Vliv velikosti učícího datasetu – reálný dataset



Obrázek 23: Vliv velikosti učícího datasetu – umělý dataset

C Popis sloupců

- id → identifikační číslo
- por_tx → pořadí transplantace, resp. o kolikátou transplantaci se u pacienta jedná
- vek → věk pacienta v době transplantace
- pohlavi → pohlaví pacienta
- dg_group → označení onemocnění, jeho název nebo druh, se kterým se daný pacient léčí
- hctci_score → HCT-CI zkratka vychází z anglického Hematopoietic Cell Transplantation-Comorbidity Index. Tento index poskytuje informaci o celkové i non-relaps hrozbě úmrtí, které je u pacienta pravděpodobná po transplantaci krvetvorných buněk
- doba_preziti → počet dnů od započetí léčby/transplantace, kdy pacient přežil. Údaj je dán poslední kontrolou nebo úmrtím
- pacient_zemrel → binárně označeno, zda pacient zemřel (1), nebo je stále naživu (0)
- doba_do_relapsu → doba od započetí léčby/transplantace, než byl pacient zasažen stavem onemocnění, který u něho již nastal v minulosti
- relaps → binární indikátor, že k relapsu došlo (označeno 1)
- NRM → nerelapsová mortalita. Binárně označeno, zda pacient zemřel z jiného důvodu, než na relaps původního onemocnění.
- DFS → zkratka vychází z anglického disease-free-survival nebo-li doba, za kterou se u pacienta po primární léčbě neobjevily žádné příznaky původního onemocnění
- aGVHD_grade → grade označuje celkové ohrožení života podle kombinace a stádia postižení jednotlivých orgánů .
- doba_do_gvhd → doba, než se u pacienta projevila reakce na provedenou transplantaci
- aGVHD_horni_git → hodnota postižení aGVHD pro horní část trávícího traktu (git je zkratka z anglického Gastro-intestinal tract)
- aGVHD_dolni_git → stejně jako předchozí je tento údaj hodnota postižení aGVHD trávícího traktu, ale tentokrát spodní části
- aGVHD_kuze → hodnota postižení aGVHD kůže pacienta
- aGVHD_jatra → hodnota postižení aGVHD jater

- den100_komplet → binární označení, zda od transplantace uběhlo 100 dnů (1), během nichž se projevilo aGVHD. Akutní reakce vzniká do 100 dní po transplantaci. Od 100 dní pak může vzniknout chronická[31].
- doba_do_cGVHD → doba, před výskytem Chronic Graft Versus Host Diseaese (odtud zkratka cGVHD). Tento druh reakce štěpu proti hostitely se může vyskytnout zpravidla později po transplantaci. Vzniká, když buňky dárce transplantátu považují tělo hostitele za hrozbu a k boji proti ní využijí hostitelův vlastní imunitní systém[32].
- max_cGVHD_grade → maximální označení vážnosti cGVHD
- pripravny_rezim → druh přípravného režimu pacienta, který má zajistit velmi intenzivní potlačení imunity, protože pouze tak organizmus přijme imunitu a krvetvorbu dárce[33].
- GvHD_profylaxe → viz [36] (další strana)
- rezim → vychází z rozdělení přípravného předtransplantačního režimu na myleoablativní (ireverzibilní nebo-li nezvratné potlačení příjemcovy krvetvorby) a nemyleoablativní (imunosuprese příjemce je dostatečná)[34].
- typ_stepu → typ štěpu buďto PBPC (z anglického peripheral blood progenitor cell, česky pak progenitorová buňka z periferní krve, což je krvetvorná kmenová buňka) nebo kostní dřeň
- typ_darce → příbuznost dárce k pacientovi
- HLA → zkratka z anglického Human Leukocyte Antigens. Kombinace, specifická pro každého jedince, antigenů hlavního histokompatibilního systému na buňkách lidského organismu s vyjímkou zralých červených krvinek.
- HLA_kod → viz [35]
- vek_darce → věk dárce

---

[31] ADAM, Zdeněk, Marta KREJČÍ a Jiří VORLÍČEK. *Hematologie: přehled maligních hematologických nemocí* [online]. 2., dopl. a zcela přeprac. vyd. Praha: Grada, 2008 [cit. 2020-05-17]. ISBN 978-80-247-2502-4

[32] *Understanding Chronic Graft Versus Host Disease* [online]. [cit. 2020-05-17]. Dostupné z: https://imbruvica.com/cgvhd/what-is-cgvhd

[33] PENKA, Miroslav a Eva SLAVÍČKOVÁ. *Hematologie a transfuzní lékařství* [online]. Praha: Grada, 2011 [cit. 2020-05-17]. ISBN 978-80-247-3459-0

[34] RAIDA, Luděk. *Nemyeloablativní alogenní transplantace krvetvorných kmenových buněk v léčbě hematologických malignit* [online]. [cit. 2020-05-17]. Dostupné z: https://www.internimedicina.cz/pdfs/int/2007/07/07.pdf

[35] Přesný význam nebyl zjištěn, neboť se v důsledku současné situace COVID 19 nepodařilo kontaktovat MUDr. Vydru

- pohlavi_darce → pohlaví dárce
- KS_pac → krevní skupina pacienta
- KS_darce → krevní skupina dárce
- CVM_pac → zkratka z názvu Cytomegalovirus. Jedná se o vir, kterým je postiženo 50-80% lidské populace. Nákaza většinou nepředstavuje problém, avšak u osob s nedostatkem imunity může způsobit selhání jater[36]. Tento údaj je pro pacienta.
- CVM_darce → pozitivní nebo negativní detekce CVM viru u dárce.
- EBV_pac → zkratka z anglického Virus Epsteina a Barrové. Jedná se o vir, který v organismu nejčastěji napadá buňky imunitního systému. Toto pole je pro pacienta a nabývá hodnot negativní nebo pozitivní
- EBV_darce → pozitivní nebo negativní detekce viru Epsteina a Barrové u dárce
- Karnoffsky → Karnofského skóre, které hodnotí celkový stav pacienta a udává jak moc je schopný pokračovat v normální aktivitě[37].
- hmotnost → hmotnost pacienta
- vyska → výška pacienta
- atg → viz [35]
- ebmt_score → viz [35]
- typ_predchozi_hct → typ předchozí transplantace (pokud existuje)
- dat_predchozi_hct → datum předchozí transplantace

[36] *Cytomegalovirová infekce* [online]. [cit. 2020-05-18]. Dostupné z: https://www.mojezdravi.cz/nemoci/cytomegalovirova-infekce-2139.html
[37] *Karnofsky Performance Status Scale* [online]. [cit. 2020-05-17]. Dostupné z: https://www.hospicepatients.org/karnofsky.html

D Části zdrojového kódu

```
class Preprocessor():
    def __init__(self,
                 target_event: str,
                 time_to_event: str,
                 other_events: list = [],
                 categorical_columns: list = [],
                 standardize_columns: list = [],
                 remove_columns: list = [],
                 remove_empty_time: bool = False,
                 preprocess_time: bool = False,
                 default_scaler = Normalizer,
                 **kwargs):
        """
        - Creates preprocessor which handles specific dataset by fitted rules.
        The default preprocessing method for non-categorical data is Normalization
        from sklearn (can be specified by 'standardize_columns' argument.
        For categorical the labeling and One-Hot-Encoding is done
        automatically.
        In categorical columns the algorithm checks for missing values and encodes
        them as new category which's new One-Hot-Encoded column is later removed
        from returned data
        During fitting this checks for unspecified columns for binary data [0;1]
        which are not preprocessed, the rest is Normalized.

        Parameters
        ----------
        target_event (str):     Required - Column name with event
        time_to_event (str):    Required - Column name with time from subject
                                entering study to event or cenzoring
        other_events (list):    List of tuples containing columns bound to first
                                element in tuple:
                                (main column name, [bound col1, bound col2, ...])
                                If the main col. is not NaN the bound cols are
                                filled with zeros. Main col. with NaN is
                                treated in same way as categorical values
        categorical_columns (list): List of column names which contain categorical
                                value
        standardize_columns (list): List of column names for which Standardization
                                will be used instead of default Normalization
        remove_columns (list):  List of column names which will be removed
        remove_empty_time (bool): If true, rows where time to event is 0 or NaN
                                are removed. Default False.
        preprocess_time (bool): Setting where time column will be preprocessed
                                aswell. If False the original times are kept.
                                Default False.
        default_scaler:         Class of sklearn preprocessor used for unspecified
                                processing numerical values. Default is Normalizer
        """
```

Zdrojový kód 1: Konstruktor preprocessing třídy

```
@abstractmethod
def _return_method(self,x):
    '''
    Method implementing output of network (activation function after otp layer)
    '''
    raise NotImplementedError('Implement me!')

def __init_layers(self):
    # input layer
    self.l_inpt = torch.nn.Sequential(
                    nn.Linear(self.__input_size,self.__nn_size),
                    nn.ReLU())
    # core layers
    core_layers = []
    for i in range(self.__n_layers):
        layer = []
        layer.append(nn.Linear(self.__nn_size,self.__nn_size))
        if self.__batchNormalize:
            layer.append(nn.BatchNorm1d(self.__nn_size))
        if self.__dropout>0:
            layer.append(nn.Dropout(p=self.__dropout))
        layer.append(nn.ReLU())
        # add to layer-list
        core_layers.extend(layer)
    self.l_core = torch.nn.Sequential(*core_layers)
    # output layer
    self.l_otp = torch.nn.Sequential(
                    nn.Linear(self.__nn_size,self.__output_size))

def forward(self, x, **kwargs):
    x = self.l_inpt(x)
    x = self.l_core(x)
    x = self.l_otp(x)
    return self._return_method(x)
```

Zdrojový kód 2: Vnitřní parametrizovatelná architektura neuronové sítě

```
class CoxPHLoss(torch.nn.Module):
    """Loss for CoxPH model. If data is sorted by descending duration, see `cox_ph_loss_sorted`.

    We calculate the negative log of $(\frac{h_i}{\sum_{j \in R_i} h_j})^d$,
    where h = exp(log_h) are the hazards and R is the risk set, and d is event.

    We just compute a cumulative sum, and not the true Risk sets. This is a
    limitation, but simple and fast.
    """
    def forward(self, *args, **kwargs) -> Tensor:
        return cox_ph_loss(log_h = args[0],
                           durations = args[1][0],
                           events = args[1][1])

def cox_ph_loss(log_h: Tensor, durations: Tensor, events: Tensor, eps: float = 1e-7) -> Tensor:
    """Loss for CoxPH model. If data is sorted by descending duration, see `cox_ph_loss_sorted`.

    We calculate the negative log of $(\frac{h_i}{\sum_{j \in R_i} h_j})^d$,
```

```
    where h = exp(log_h) are the hazards and R is the risk set, and d is event.

    We just compute a cumulative sum, and not the true Risk sets. This is a
    limitation, but simple and fast.
    """
    idx = durations.sort(descending=True)[1]
    events = events[idx]
    log_h = log_h[idx]
    return cox_ph_loss_sorted(log_h, events, eps)

def cox_ph_loss_sorted(log_h: Tensor, events: Tensor, eps: float = 1e-7) -> Tensor:
    """Requires the input to be sorted by descending duration time.
    See DatasetDurationSorted.

    We calculate the negative log of $(\frac{h_i}{\sum_{j \in R_i} h_j})^d$,
    where h = exp(log_h) are the hazards and R is the risk set, and d is event.

    We just compute a cumulative sum, and not the true Risk sets. This is a
    limitation, but simple and fast.
    """
    if events.dtype is torch.bool:
        events = events.float()
    events = events.view(-1)
    log_h = log_h.view(-1)
    gamma = log_h.max()
    log_cumsum_h = log_h.sub(gamma).exp().cumsum(0).add(eps).log().add(gamma)
    return - log_h.sub(log_cumsum_h).mul(events).sum().div(events.sum())
```

Zdrojový kód 3: Average Negative Log Likelihood loss funkce (PyCox [30])

```
def _predict_by_model(self,
                      inpt:pd.DataFrame,
                      event:pd.DataFrame = None,
                      times:pd.DataFrame=None,
                      times_orig:pd.DataFrame=None,
                      unprocessed:pd.DataFrame=None) -> list:
    risks = np.exp(self._predict_log_risk(inpt))
    if self._orig_time_for_baseline:
        hazards = self._predict_cumulative_hazard(times_orig)
    else:
        hazards = self._predict_cumulative_hazard(times)

    risk_test = pd.DataFrame(risks.astype('float32'))
    result=pd.DataFrame(hazards.values.reshape(-1, 1).dot(risk_test[0]))

    return list(np.exp(-result[0]).values)
```

Zdrojový kód 4: Propojení základního kumulativního rizika s NN