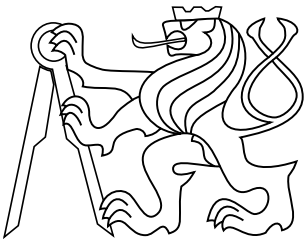
Diplomová práce

Aplikace strojového učení v integračním testování metodou Model-Based Testing

Bc. Jakub Klíma



Květen 2020

Vedoucí práce: Ing. Jan Sobotka, Ph.D.

České vysoké učení technické v Praze
Fakulta elektrotechnická, Katedra měření

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Klíma** Jméno: **Jakub** Osobní číslo: **420170**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra řídicí techniky**

Studijní program: **Otevřená informatika**

Studijní obor: **Počítačové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Aplikace strojového učení v integračním testování metodou Model-Based Testing

Název diplomové práce anglicky:

Application of Machine Learning in Integration Testing by Model-Based Approach

Pokyny pro vypracování:

1. Seznamte se s problematikou HiL integračního testování.
2. Analyzujte zdrojový kód a modelovací jazyk MBT nástroje Taster. Navrhněte rozšíření modelovacího jazyka o data použitelná pro predikci závad ve funkcionalitách testovaných systémů.
3. Analyzujte možnosti frameworku ML.NET pro strojové učení a jeho případnou integraci v nástroji Taster. V případě, že se framework ukáže jako nevhodný, najděte vhodnou alternativu.
4. Navrhněte a v nástroji Taster implementujte vhodný algoritmus strojového rozpoznávání a učení pro predikci závad v testovaných systémech. Navrhovaný algoritmus konzultuje s vedoucím práce.
5. Pomocí experimentů vyhodnoťte kvality navrženého algoritmu na případové studii.
6. Výsledky vyhodnoťte a navrhněte případné úpravy.

Seznam doporučené literatury:

[1] Bishop, Christopher M.Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, 2006.
[2] J. Zander, I. Schieferdecker, and P.J. Mosterman. Model-Based Testing for Embedded. Systems. Taylor & Francis, 2011.
[3] Johan Bengtsson, and Wang Yi. Timed Automata: Semantics, Algorithms and Tools. Lecture Notes in Computer Science. 2004.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Jan Sobotka, Ph.D., katedra měření FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **24.01.2019** Termín odevzdání diplomové práce: **24.05.2019**

Platnost zadání diplomové práce:
do konce letního semestru 2019/2020

Ing. Jan Sobotka, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Ing. Pavel Ripka, CSc.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

Poděkování

Rád bych poděkoval Ing. Janu Sobotkovi, Ph.D. a Ing. Lukáši Krejčímu za odborné vedení a konzultace k mé diplomové práci. Dále bych rád poděkoval rodině a přátelům, kteří mě psychicky podporovali po celou dobu studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne.....

Podpis autora práce.....

Abstrakt

Tato diplomová práce se zabývá metodami procházení sítě časovaných automatů. Úkolem je obohatit stávající metody o prvky strojového učení. Cílem této diplomové práce je implementovat takovou metodu, která bude procházet stavovým prostorem se snahou pokrýt co nejdříve hrany s vysokou váhou. V této diplomové práci jsou rozebrány obecné vlastnosti strojového učení a způsob konkrétní implementace pro výše zmíněné metody. V další části jsou znázorněny výsledky úspěšnosti jednotlivých metod. V neposlední řadě jsou součástí této diplomové práce i zdrojové kódy implementovaných metod. V závěru jsou zhodnoceny jednotlivé metody a jejich výkon při procházení stavovým prostorem. Na testovaném modelu vyšla nejlépe metoda AI Systematic s využitím Poissonovi regrese a statistiky hodnocení založeném na maximu a AI Random s využitím víceúrovňové klasifikace s hodnocením založeném na mediánu a maximu.

Klíčová slova

Strojové učení, Modelem řízené testování, Hardware ve smyčce, Taster, UPPAAL

Abstrakt

This Master thesis is about methods for searching in network of timed automata using machine learning. The task is to extend current methods with machine learning functionality. The goal of this thesis is to implement a method that search trough the state space in order to cover edges with high priority as soon as possible. The thesis also describes general properties of machine learning and the specific implementation above mentioned methods. The next section shows the results of the performance of individual methods. Last but not least, the source codes of the implemented methods are part of this thesis. In the end, the individual methods and their performance in traversing the state-space are evaluated. On the tested model methods AI Systematic with Poisson regression and maximum-based label and AI Random with multiclass classification with median and maximum based labels were performing best.

Keywords

Machine learning, Model Based Testing, Hardware–in–the–Loop, Taster, UPPAAL

Obsah

1 Úvod

Cílem této diplomové práce je rozšířit softwarový nástroj Taster, který je určený k testování automobilových elektronických systémů během vývoje nového vozu. Testování elektroniky (jedna nebo více řídicích jednotek) v tomto případě probíhá na integrační úrovni metodou Hardware-in-the-Loop (HiL).

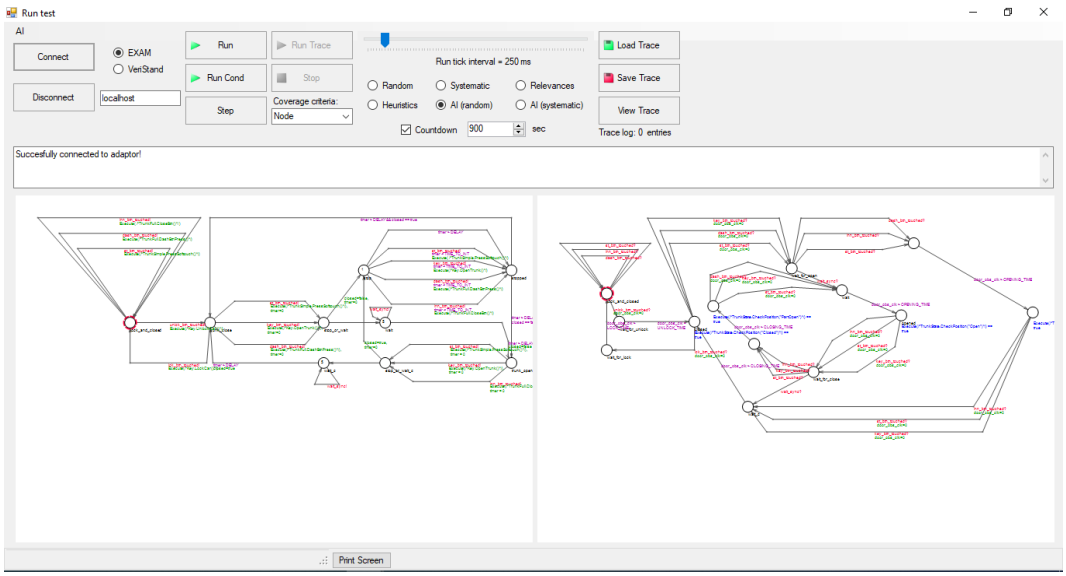
Integrační testování je fáze testování, kde se testují různé již otestované komponenty, zda fungují správně při interakci mezi sebou. Především se jedná o ověřování vzájemné komunikace a funkcí distribuovaných do více řídicích jednotek. Testovací metoda HiL spočívá v testování reálných hardwarových komponent za pomoci simulace vstupů a monitorování výstupů těchto komponent[1].

Jednotlivé testy jsou generovány na základě modelu, který reprezentuje chování testovaného systému. Tomuto typu testování se říká modelem řízené, jinak také anglicky Model-Based Testing (MBT)[2]. Model pro nástroj Taster je reprezentován jedním nebo více časovanými automaty. V takovém případě hovoříme o síti časovaných automatů. Model popisuje také chování uživatele (řidiče), které je využito pro generování testovacích vstupů. Důvodem je upřednostnění pravděpodobnějšího uživatelského scénáře při generování testů. Na základě modelu systému je potom ověřována správnost výstupů.

Taster generuje testovací sekvence tak, že prochází a vyhodnocuje model a na jeho základě ovládá vstupy systému a ověřuje výstupy. Taster obsahuje tři metody na procházení modelem nazvané Random, Systematic a Relevance, které jsou více popsány v kapitole 2.1. Všechny tyto metody prochází stavovým prostorem bez predikce následujících stavů. Cílem této diplomové práce je implementovat do nástroje Taster nové metody na procházení stavovým prostorem modelu založené na strojovém učení. Cílem je zefektivnit generované testy, za účelem co nejrychlejšího nalezení chyby nebo zvětšit pokrytí stavového prostoru v přiděleném čase.

2 Taster

Nástroj Taster je softwarový produkt vytvořený pod vedením pana Ing. Jana Sobotky Ph.D., který slouží k testování jednotlivých komponent automobilu metodou MBT. Taster využívá jako vstupní data soubor ve formátu eXtensible Markup Language, známém spíše jako XML, který reprezentuje model testovaného systému ve formě časovaného automatu, což je speciální typ stavového prostoru rozšířený o časová omezení[3]. Po načtení modelu stavového prostoru a převedení do vlastní datové reprezentace zobrazí Taster model graficky. Během testování prochází Taster stavový prostor a vizuálně zobrazuje aktuální stav a je tedy možné, aby uživatel sledoval průběh testování v reálném čase. Ukázka grafického rozhraní je na obrázku 1 a na obrázku 2 je pak znázorněn jeden graf během testování.



Obrázek 1 Ukázka uživatelského rozhraní Tasteru.

2.1 Metody procházení stavového prostoru

Taster původně obsahoval tři základní metody možného procházení modelem. První metoda je takzvaně náhodná, kde se vybírá další hrana náhodně. Druhá metoda je systematická, která se snaží dosáhnout co nejrychleji pokrytí všech hran, což znamená, že vybere vždy hranu s nejmenším počtem průchodů. Poslední třetí metoda má název Relevances a funguje na principu vah neboli relevancí. Některé hrany mají přiřazenou relevanci v rozsahu $\langle 0 - 9 \rangle$ a metoda Relevances se snaží upřednostnit ty hrany, které mají relevanci vyšší. Čím vyšší je relevance, tím vyšší je také šance, že daná hrana bude vybrána.

2.2 Ukončovací podmínky

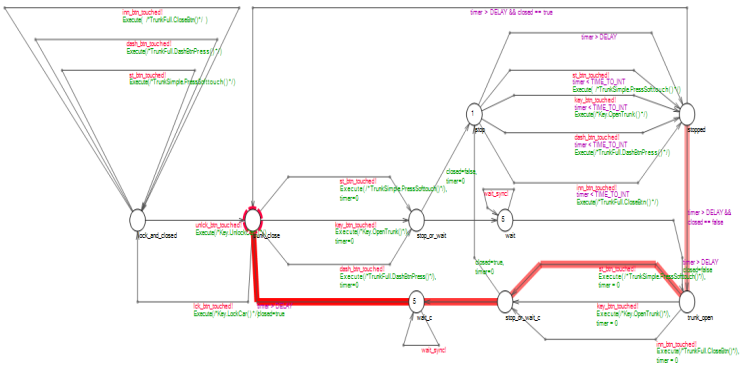
Testování je možné provádět s různými ukončovacími podmínkami. První možnost, jak omezit běh testu, je pomocí času, kde je možné nastavit čas v sekundách, po kterém se běh testu ukončí. Další z možností ukončení testu je při pokrytí všech hran nebo vrcholů. Po každém kroku se kontroluje pokrytí a jakmile budou pokryté všechny hrany nebo vrcholy, tak se nastaví příznak a je-li nastavena konečná podmínka, tak se test ukončí. V případě nalezení chyby u testované jednotky se průchod ukončí, zaznamená se chyba a test se spustí znovu od začátku. Pro provádění experimentů jsem využíval ukončovací podmínku pouze na čas. Při provádění reálného testování se využívá i ukončovací podmínka pro celkové pokrytí.

2.3 Záznam běhu

Program Taster podporuje záznam jednotlivých běhů a ukládá je do souboru ve formátu XML. Záznam je poté možné prohlížet po jednotlivých krocích a nebo spustit znovu. Pro lepší možnost prezentace jednotlivých běhů jsem implementoval možnost ukládat běh formou pohyblivého obrázku ve formátu GIF. Dále jsem implementoval možnost kdykoliv za běhu zaznamenat aktuální stav a uložit jako snímek aktuálního stavu. Obrázky i GIF se generují pro pevně zvolené rozlišení 1920x1080, tedy Full HD. Ukázka obrázku pořízeného za běhu testu je zobrazena na obrázku 2.

2.4 Další úpravy pro podporu experimentů

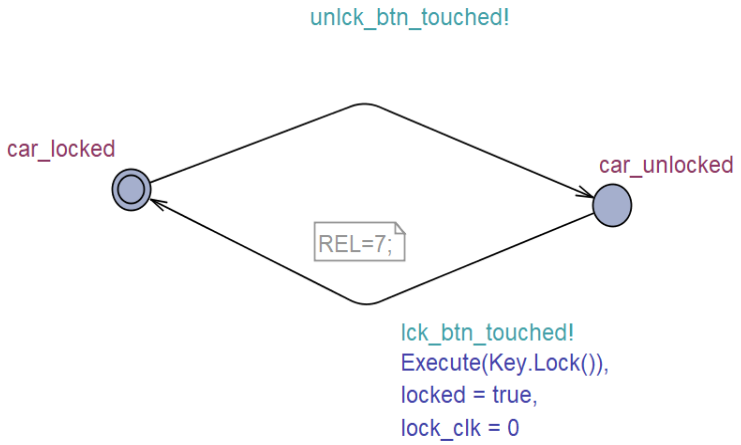
V programu Taster jsem vytvořil pár podpůrných funkcionalit, které pomůžou v ovládání, záznamu a kontrolování běhu testu. Mimo výše zmíněné zaznamenávání pohyblivého nebo statického obrázku je možné kliknout na jednotlivé panely, kde jsou zobrazeny jednotlivé časované automaty a zvětšit jeden požadovaný panel přes celé okno programu. Dále jsou jednotlivé hrany zvýrazněny červenou barvou, pokud přes ně v posledních čtyřech krocích prošel algoritmus, s tím že každá hrana je zvýrazněna méně v závislosti na pořadí průchodu. Čím je starší průchod přes hranu, tím je slabší zvýraznění, toto zvýraznění je vidět na obrázku 2.



Obrázek 2 Ukázka snímku pořízeného za běhu testu.

3 Modelovací jazyk

Model stavového prostoru dokáže Taster načíst ze souboru ve formátu XML. Tento formát odpovídá formátu, který generuje nástroj UPPAAL. Na obrázku 3 je zobrazen jeden časovaný automat na odemykání a zamykání automobilu. Jeden model se skládá z několika takových automatů.



Obrázek 3 Ukázka časovaného automatu na odemykání a zamykání.

3.1 UPPAAL

UPPAAL je nástroj pro modelování, validaci a verifikaci systémů reálného času, reprezentovaných jako časované automaty. Je vytvořen za spolupráce univerzity Aalborg v Dánsku a univerzity Uppsala ve Švédsku [4]. UPPAAL se skládá ze tří základních částí, popisného jazyka, simulátoru a validátoru [5]. Popisný jazyk popisuje jednotlivé modely, simulátor poskytuje možnost kontroly v brzkých modelových fází a validátor provádí komplexní kontrolu zkoumáním stavového prostoru systému, ověření dostupnosti jednotlivých částí modelu a kontroly invariantu. K tvorbě modelů pro Taster se využívá právě nástroje UPPAAL.

3.2 Rozšíření modelovacího jazyka

Modelovací jazyk neobsahuje možnost přidání relevancí k hranám nebo vrcholům. Proto se využívalo řešení přidání relevance jako komentáře k jednotlivým vrcholům. Vrchol byl zvolen, protože dříve bylo možné přidat komentář jen k vrcholům a nikoli k hranám. Vrchol v modelu reprezentuje stav a hrana akci, která reprezentuje přechod mezi stavy. Vzhledem k tomu, že do jednoho stavu může vést více hran a každá reprezentuje jinou akci, je lepší mít ohodnocené hrany namísto vrcholů. Proto jsem změnil způsob zadávání relevancí. Modelovací jazyk je nyní využíván tak, že jednotlivé hrany nesou údaj o

své relevanci. Tento způsob nahradil přiřazování relevancí k vrcholům. Relevance se k hranám přidá zapsáním do komentáře, stejně jako se přidávala k vrcholu.

Při tvorbě modelu je to realizováno zapsáním komentáře k hraně v následujícím formátu

```
REL=6;
```

Následně byla upravena i metoda Relevances, která nyní používá již relevance z hran, stejně jako je používají metody pro strojové učení, namísto toho aby využívala relevance z vrcholů.

Na obrázku 3 je možné vidět relevanci přiřazenou hraně z vrcholu car_unlocked do vrcholu car_locked.

4 Analýza ML.NET

Program Taster je psán v jazyce C# a pro tento jazyk existuje framework pro strojové učení nazvaný ML.NET. Tento framework má dostupnou rozsáhlou dokumentaci na stránkách společnosti Microsoft. Pro implementaci algoritmu procházení stavovým prostorem, za využití strojového učení, je potřeba, aby daný framework splňoval určité požadavky. Tyto požadavky jsou popsány konkrétně na ML.NET v kapitolách níže.

4.1 Strojového učení

Strojové učení obecně funguje tak, že se vezme soubor vstupních dat a model strojového učení se podle nich naučí předvídat hodnocení jiných dat stejného typu. Soubor vstupních dat se skládá z takzvaných datových bodů, přičemž každý datový bod obsahuje vektor příznaků a hodnocení. Vektor příznaků může být vektor čísel nebo třeba textový řetězec.

Strojové učení se dělí na učení s učitelem a učení bez učitele. Skupina učení s učitelem obsahuje metody, jejichž data dokážeme jako uživatel ohodnotit, například lineární regrese nebo klasifikace. Na druhé straně je skupina metod učení bez učitele a v této skupině jsou metody, jako například shlukování, které se hodí pro data o nichž sami nevíme jaké hodnocení přiřadit a strojové učení mezi nimi hledá společné vlastnosti.

Naučený model strojového učení funguje tak, že se mu jako vstup předají data určená k hodnocení a strojové učení na výstupu vygeneruje hodnocení tak, aby pokud možno dodržovalo všechny vzorce chování definované datovými body, ze souboru dat pro učení.

4.2 Strojového učení v nástroji Taster

Při použití strojového učení v nástroji Taster reprezentuje vektor příznaků určitý stav a hodnocení zase jeho kvalitu. Toto hodnocení je typicky jedno číslo nebo textové označení. Finální soubor dat pro učení se skládá ze stovek nebo více vektorů příznaků, každý se svým hodnocením. V případě, že data mají velký rozptyl, je možné provádět normalizaci dat, za účelem zjednodušení rozpoznání jednotlivých stavů.
Jakmile jsou všechny úpravy dat hotové, tak se podle nich vytvoří model na základě metody, kterou si programátor vybere. ML.NET nabízí hned několik metod, které jsou vhodné pro použití v nástroji Taster. Mezi ně patří například lineární regrese, řazení nebo víceúrovňová klasifikace. Všechny tyto metody patří do skupiny metod "učení s učitelem".

Dále při tvorbě modelu je také možnost použít křížovou validaci. Křížová validace spočívá v tom, že se rozdělí data pro učení na n náhodných skupin. Pro každou kombinaci dat z $n - 1$ skupin vytvoří jeden model strojového učení a na poslední skupině dat z jedné zbývající skupiny otestuje, jak dobře model predikuje hodnocení na datech, které ještě neviděl. Následně si programátor vybere ten nejlepší model, který mu pro danou situaci vyhovuje.
Po vytvoření modelu je možné ho uložit na disk do souboru a při dalším spuštění programu načíst tento model a nevytvářet nový, tím dojde k úspoře času při spouštění.

Jakmile je model vytvořený případně načtený, tak může predikovat hodnocení.

4.3 Vektor příznaků

Vektor příznaků je popis nějakého stavu nebo objektu. V případě nástroje Taster to je aktuální stav automatu. Do vektoru příznaků se musí vložit informace nutné k rozpoznání typu stavu, ale nemusí obsahovat informace potřebné k rekonstrukci stavu. Například při vytváření datového bodu ze stavového prostoru pro nástroj Taster, není potřeba používat informaci o tom, v jakém je automat aktuálním stavu, ale stačí použít informace o relevancích hran, které z aktuálního stavu vedou.
ML.NET podporuje tvorbu vektoru příznaků, jak z datového typu integer, tak i z typu float, což je pro účely nástroje Taster vyhovující.

4.4 Data pro učení

ML.NET umožňuje načítat data pro učení z textového souboru s koncovkou .txt. Data pro učení uložená v souboru jsou ve formátu jeden datový bod na jeden řádek. Uložený datový bod může mít víc dat než bude nakonec použito pro učení modelu. Například se může uložit vektor příznaků s více různými hodnoceními a pro učení se jen vybere jedno, které se bude používat.

V implementaci pro nástroj Taster má jeden řádek dat pro učení následující formát:

p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	tc	h1	h2	h3	h4	h5

px – prvek histogramu priorit
tc – počet průchodů přes hranu
hx – hodnocení

Konkrétní ukázka ze souboru je v tabulce níže. V ukázce je pouze 10 datových bodů, celý soubor jich obsahuje v řádu stovek až tisíců.

10	0	0	2	1	3	3	1	1	1	8	10	8	10	7.78	9.1
12	0	0	2	1	3	3	2	0	2	3	9.64	8.53	9.64	8.39	9.64
17	0	0	2	0	3	3	2	0	2	2	9.01	6.64	9.64	6.31	6.92
14	0	0	2	1	3	4	1	1	2	6	9.58	6.58	9.58	6.25	7.68
13	1	1	1	3	2	4	4	1	1	2	9.88	5.88	9.88	5.44	6.55
6	1	1	0	3	2	4	4	1	1	2	8.11	5.94	9.94	5.5	6.06
7	1	1	2	2	2	3	3	0	2	6	6.54	5.71	9.71	5.26	5.57
9	1	1	2	2	2	3	3	0	2	4	6.33	5.94	9.94	5.5	5.71
11	1	0	2	1	1	2	1	1	2	4	8.51	9.94	9.94	9.94	9.49
13	1	0	2	1	1	2	1	1	1	2	9.29	9.71	9.71	9.71	9.59

Při spuštění Tasteru se data načtou do paměti, kde jsou uchována všechna načtená data. Dále je nutné z načtených dat vybrat ty, které chceme použít pro aktuální model a jakým způsobem se použijí. K tomu lze použít například metodu Concatenate, obsaženou v ML.NET, jenž vybere z načtených dat jen ta data, která chceme použít jako vektor příznaků a data která chceme použít jako hodnocení neboli label. Dále je možné v této fázi data upravit, například normalizovat nebo filtrovat a použít jen vhodné vektory.

4.5 Normalizace a filtrování

Po úspěšném vybrání dat je možné je normalizovat nebo filtrovat. Filtrováním dat se zahodí data, která nesplňují námi zvolená kritéria, například přesahují určitou hodnotu. Normalizace dat upraví jejich rozsah, aby nebylo obsazení dat v originálním rozsahu příliš řídké.
V každém případě je nutné následně stejným způsobem upravit i data u kterých budeme chtít predikovat hodnocení. Tedy pokud originální data měla rozsah 0-100 a po normalizaci mají rozsah 0-1, tak se data při predikci musí také upravit na rozsah 0-1, v tomto případě vydělit číslem 100.
Vzhledem k tomu, že je možné model strojového učení načíst ze souboru, je také nutné si ke každému modelu uložit koeficienty použité k normalizaci, aby se mohly stejným způsobem upravit i data při predikování.

4.6 Tvorba modelu strojového učení

Po zpracování vstupních dat je nutné určit, kterou metodu pro učení vybrat. ML.NET jich má několik. Pro cíle této diplomové práce je potřeba takového modelu, který dokáže hodnotit více hran a roztřídit je podle kvality, proto se vybízí použít metody jako je víceúrovňová klasifikace, řazení nebo lineární regrese.
Následně zbývá poslední krok a tím je vytvoření samotného modelu pomocí metody Fit. Poté je vše připraveno pro to, aby model mohl dělat predikce pomocí metody Predict. Této metodě se jako vstupní parametr předá popis aktuálního stavu ve stejném formátu, jako byl použit vektor příznaků pro učení, akorát nyní se data předají bez hodnocení, které se naopak chce zjistit.

4.7 Druhy strojového učení

Strojové učení využívá k učení a k predikci různé metody, které jsou vhodné na různé problémy. Základní dělení je na učení s učitelem a učení bez učitele. Učení s učitelem se snaží vytvořit funkci, která bude predikovat hodnocení co nejpřesněji podle vzorových dat. Učení bez učitele je vhodné naopak pro data u kterých si nejsme jisti podle čeho by se měla hodnotit[6]. Pro nástroj Taster se hodí metody, které dokáží ohodnotit vstupní data prioritami. Mezi takové metody patří například

- lineární regrese
- víceúrovňová klasifikace
- řazení

všechny tyto metody jsou ze skupiny učení s učitelem, do skupiny učení bez učitele patří například shlukování.

4.7.1 Lineární regrese

Lineární regrese predikuje hodnocení na základě následující rovnice[7]

$$Ax + b = y \tag{1}$$

kde:

A je matice váh vytvořená ze vstupních dat
x je feature vector
b je koeficient
y je hodnocení

Matice A a koeficient b jsou nejdříve náhodná čísla. Během fáze učení se ale mění tak, aby co nejméně minimalizovala chybu pro predikované $y(pred)$ a skutečné y podle vzorce níže[8]:

$$\arg\min(\frac{1}{n}\sum_{i=1}^{n}(pred_i - y_i)^2) \tag{2}$$

Jakmile se najdou hodnoty A a b které minimalizují chybu mezi všemi vektory x a číslem y, tak se z rovnice výše určí hodnocení. Lineární regrese podporuje několik metod pro optimalizování hodnot A a b. V ML.NET je možné využít optimalizátor online gradient descend, poissonovu metodu a metodu SDCA neboli stochaistic dual coordinate ascent.

Online snižování gradientu

Tato metoda je založena na iterativním algoritmu pro hledání minima funkce pomocí snižování gradientu [9]. V tomto případě hledání minimální odchylky predikovaného hodnocení a skutečného hodnocení. Kdy se k jednotlivým hodnotám z feature vektoru přidávají váhy, které se mění v průběhu iterací a to dokud není odlišnost od dat z datasetu minimální. Slovo Online v názvu metody znamená, že se s každou provedenou predikcí model "přiučí"a aktualizuje se.

Poissonova regrese

Poissonova regrese je vhodná pro data, která obsahují počet něčeho na nějaké závislosti, například na čase. V případě programu Taster lze jako počet použít počet vysokých relevancí za zvolenou hranou v závislosti na aktuálním stavu. Pro to, aby Poissonova regrese fungovala, nesmí data pro učení obsahovat záporná čísla.

Poissonova regrese predikuje hodnocení na základě této funkce:

$$E[Y|x] = \alpha + \beta' x \tag{3}$$

Y – hodnocení dat
x – proměnná reprezentující distribuci dat
α – offset
β – vektor vah

SDCA

Stochastické snižování duálních koordinátů je další optimalizátor který ML.NET podporuje pro lineární regresi. Tento optimalizátor je podobný snižování gradientu, akorát zde se zvyšují koordináty z duální úlohy[10]. Tím se dosáhne obecně lepších vlastností, než u metody snižování gradientu. SDCA například konverguje rychleji k řešení a je jasně daná podmínka ukončení.
Metoda SDCA kombinuje několik výhodných vlastností, které způsobují rychlejší tvorbu modelu strojového učení. První z nich je možnost streamovaného učení, to znamená učení se na datech bez toho, aby se celá načítala do paměti[11]. Další výhodná vlastnost je možnost ignorovat nulové hodnoty v řídkých datových sadách. Obě tyto vlastnosti vedou k úspoře výpočetního výkonu a tím pádem rychlejší tvorbě modelu.

4.7.2 Víceúrovňová klasifikace

Víceúrovňová klasifikace uspořádá data do jednotlivých úrovní. V nástroji Taster bude vhodné využít víceúrovňovou klasifikaci pro rozdělení hran do úrovní 0-9 podle jejich kvality. Víceúrovňová klasifikace se dělí na dvě skupiny, první využívající binární klasifikaci, tam patří například metody jeden-versus-ostatní a jeden-versus-jeden a druhá skupina využívá metody, jako například regrese a podobné [12]. V nástroji Taster jsem implementoval metodu maximální entropie, což je metoda podobná logistické regresi s rozdílem, že logistická regrese rozděluje problémy jen binárně a maximální entropie je zvládne rozdělit do více tříd [13].

Jeden–versus–ostatní – pro každou úroveň je jeden binární klasifikátor, který prohlásí o zkoumaném objektu jestli patří do této úrovně nebo nepatří. Zde může dojít k tomu, že jeden objekt bude patřit do více úrovní.

Jeden–versus–jeden – pro každou dvojici je jeden binární klasifikátor, který určí jestli objekt patří do jedné nebo druhé úrovně. Poté co je objekt ohodnocen všemi klasifikátory, tak se přiřadí do úrovně, pro kterou hlasovalo nejvíce klasifikátorů.

Maximální entropie – metoda maximální entropie je založená na stejném principu jako logistická regrese, jen dokáže třídit data do více než dvou skupin[14]. Skóre pro úroveň c se vypočte dle následujícího vzorce.

$$y^c = \mathbf{w}_c\mathbf{x} + b_c \tag{4}$$

A pravděpodobnost, že x patří do úrovně c se určí podle vzorce 5.

$$P(c|\mathbf{x}) = \frac{e^{y^c}}{\sum_c^m = 1^{e^{y^c}}} \tag{5}$$

w – vektor vah pro úroveň c
b – bias pro úroveň c
c – reprezentuje úroveň 1 až m
x – zkoumaný vektor příznaků

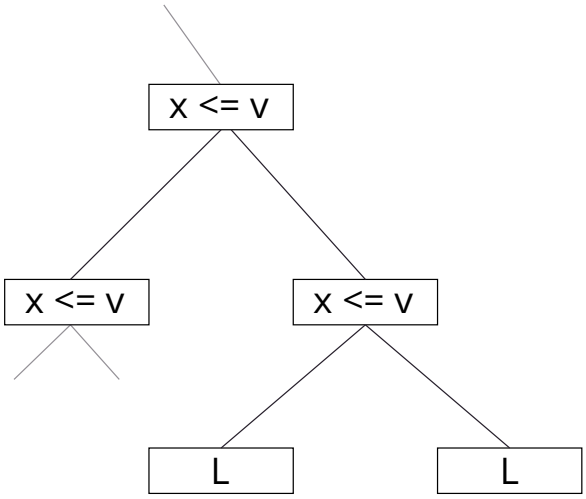
Pro metodu víceúrovňové klasifikace v nástroji Taster je použit optimalizátor založený na maximální entropii.

4.7.3 Seřazení

Cíl této metody je seřadit všechna data podle jejich kvality, přičemž ta s nejlepším hodnocením mají lepší kvalitu a jsou na prvních místech a poté následují data s horším hodnocením.
Pro metodu Řazení jsem použil optimalizátor Fast Tree[15], který využívá implementace MART gradient boosting algoritmu [16]. Ten spočívá na iterativním vytváření stromu. Nejprve se vytvoří jeden strom, zkontroluje chybovost a podle toho se vytvoří další a následně se vybere ten nejlepší strom.
Výsledný strom má podobu binárního rozhodovacího stromu. V každém vrcholu se rozhoduje kterým ze dvou potomků bude pokračovat, podle dat z vektoru příznaků. Jednotlivé rozhodnutí je založeno na porovnání:

$$x <= v \tag{6}$$

Kde x je prvek z vektoru příznaků a v je jedna z možných hodnot tohoto příznaku. Následně v listech je hodnota predikce(L). Princip tohoto stromu je znázorněn na obrázku 4.



Obrázek 4 Znázornění binárního rozhodovacího stromu.

5 Implementace

Strojové učení v nástroji Taster slouží k tomu, aby pomohlo určit jakou hranou se z aktuálního vrcholu vydat dál při procházení stavovým prostorem s cílem projít co nejrychleji vysoké priority a zároveň dosáhnout co největšího pokrytí hran. V následujících kapitolách je popsáno, jak jsem strojové učení implementoval pomocí frameworku ML.NET.

5.1 Histogram

Při vybírání hrany, kterou se půjde dál, během procházení stavovým prostorem je důležité, jaké hrany se zpřístupní při dalších krocích. Z toho důvodu jsem se rozhodl pro každou hranu vytvořit histogram všech dalších relevancí do určité hloubky. Model testovaného systému většinou obsahuje více časovaných automatů, které se prochází současně a je tedy více aktuálních stavů, ze kterých je možné se vydat dále. Histogram se počítá pro všechny možnosti volby hran, přes všechny grafy. Vzhledem k tomu, že takových grafů je více a z jednoho stavu může být více hran, bylo by výpočetně náročné proházet stavový prostor celý. Pokud by bylo x grafů každý s N stavy a přechody by byly mezi každým stavem v daném grafu, tak bude celkem E hran.

$$E = x * N^2 \tag{7}$$

Což je pro větší modely příliš velké číslo na to, aby se takový výpočet prováděl při každém rozhodování u výběru hrany. Z důvodu úspory výpočetního výkonu jsem se tedy rozhodl omezit hloubku zkoumání dalších hran pro tvorbu histogramu na úroveň čtyři, nicméně toto číslo je v kódu použito jako konstanta a může se tedy v budoucnu snadno změnit. Po omezení hloubky průchodu bude vypadat maximální počet hran následovně.

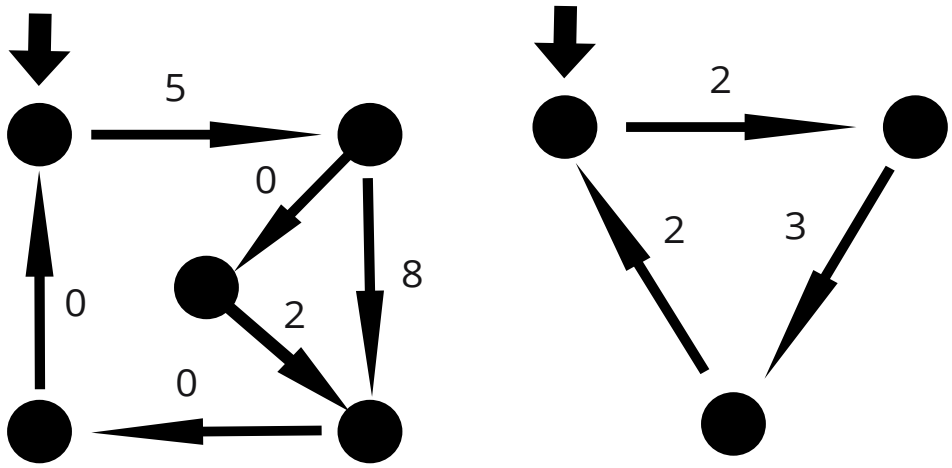
$$E = x * 4 * N \tag{8}$$

Nyní jde již o lineární složitost namísto kvadratické.
Nicméně i provádění takového výpočtu při každém kroku je na výsledné době běhu programu znát. Proto jsem se rozhodl vytvořit pomocnou paměť s před-počítanými údaji pro data z histogramu. Tato pomocná paměť je realizovaná jako soubor obsahující uložené histogramy. Tento soubor lze při příštím průchodu použít a zmizí tak nutnost výpočtu některých dat. Vzhledem k tomu, že se data ukládají do souboru, je možné tuto pomocnou paměť využít i po restartování aplikace.

5.1.1 Tvorba histogramu

Tvorba histogramu je založená na prohledávání do hloubky(DFS). Projdou se všechny možnosti do hloubky čtyři a originální relevance se zaznamenají do histogramu. Kdybychom měli model jako je na obrázku 5, tak by algoritmus tvorby histogramu našel následující cesty do hloubky čtyři.

```
5 -> 8 -> 0 -> 0
5 -> 8 -> 0 -> 2
5 -> 8 -> 2 -> 3
5 -> 0 -> 2 -> 0
5 -> 0 -> 2 -> 2
5 -> 0 -> 2 -> 3
5 -> 2 -> 3 -> 2
...
```



Obrázek 5 Ukázkový model.

Pro vytvoření finálního histogramu je potřeba zbavit se duplicitních relevancí. Výsledný histogram by vypadal následovně.

3	0	3	1	0	1	0	0	1	0

Histogram pro jednotlivou hranu obsahuje i danou hranu. Řekneme-li, že je histogram do hloubky čtyři, tak první úroveň obsahuje hranu která se bude v aktuálním kroku vybírat.

5.1.2 Ukládání histogramu

Aby se histogram nemusel počítat neustále znovu pro stejné vstupní hodnoty, ukládá se každý histogram do souboru a slouží jako pomocná paměť s před-počítanými daty. Histogram se ukládá do souboru pojmenovaném stejně, jako vstupní soubor reprezentující model pro testování a má koncovku .histogramData.
V souboru se používá formát jeden histogram na jeden řádek, přičemž řádek začíná s identifikací aktuálního globálního stavu. Aktuální globální stav je výpis aktuálních stavů v každém grafu, oddělen mezerou. Následuje popis hrany, která je reprezentovaná formátem $PrvníVrchol- > DruhýVrchol(Relevance)$. Následuje už jen výpis jednotlivých hodnot z histogramu, začínající na indexu 0. Pro lepší představu je formát ukládání zobrazen na obrázku 6.

```
id2 id18 :id2->id2(2) 8 1 1 0 2 0 3 3 0 1
id2 id18 :id2->id2(4) 8 1 1 0 2 0 3 3 0 1
id2 id18 :id2->id9(7) 8 1 1 0 2 0 3 3 0 1
id2 id18 :id18->id18(0) 11 1 1 2 2 1 3 3 0 2
id2 id18 :id18->id19(0) 8 1 1 0 2 0 3 3 0 1
id9 id18 :id9->id2(6) 8 1 1 0 2 0 3 3 0 1
```

Obrázek 6 Ukázka ukládání histogramu.

5.2 Vektor příznaků

Vektor příznaků je soubor dat, který reprezentuje jednu možnou volbu, při rozhodování se mezi různými cestami. Každý vektor příznaků je ohodnocen nějakým hodnocením. Jako vektor příznaků je nutné zvolit správná data, která budou reprezentovat konkrétní situaci a je možné se podle nich rozhodnout, jak dobré ohodnocení lze použít v porovnání s ostatními možnostmi.
V nástroji Taster je zapotřebí zvolit data do vektoru příznaků taková, která popíšou možnosti dalších cest z daného stavu. Nabízí se zvolit například průměrnou prioritu za danou hranou nebo podobnou statistiku. Aby se dosáhlo co nejméně zkresleného výsledku, jako například u průměrování, rozhodl jsem se využívat histogram všech priorit při cestě přes danou hranu do určité hloubky, konkrétně do hloubky čtyři. Histogram se získá z průchodu stavovým prostorem metodou prohledávání do hloubky, při kterém se sbírají priority na hranách a následně se odstraní duplikáty a vytvoří se z nich histogram.
Aby se dosáhlo co největšího pokrytí a algoritmus neprocházel jen nejvyšší priority stále dokola, bylo nutné vložit ještě jeden údaj, který zohlední počet průchodů přes danou hranu. Hotový vektor příznaků se tedy skládá z histogramu priorit, celkem deset čísel, dále pak z počtu průchodů přes danou hranu a poslední údaj je hodnocení. Dohromady obsahuje jeden datový bod 12 čísel.

p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	tc	h

p0 – p9 – prvek histogramu priorit
tc – počet průchodů přes hranu
h – hodnocení

5.3 Ohodnocení

Pokud je popsán aktuální stav, je ještě potřeba ho nějak ohodnotit, aby strojové učení mělo nějaký návod, jak se rozhodovat při finálním běhu. Jako ohodnocení jsem zvolil více metod, všechny ale vychází ze stejné myšlenky a to je porovnání relevancí nasbíraných v histogramu se skutečným průchodem. Uživatel si při používání aplikace Taster musí vybrat, jakou metodu hodnocení chce použít, vždy lze použít jen jedna. Výpočet hodnocení v praxi funguje tak, že se uloží poslední čtyři kroky a ty se porovnávají s histogramem priorit. Porovnání probíhá dle vzorců 12, 13, 14, 15, 16.
V tomto hodnocení je nicméně započítán pouze histogram z vektoru příznaků a není započítáno kolikrát se přes danou hranu přešlo. Tato metrika, nazvaná *takeCountRatio*, se započítá v dalším kroku tím, že se vezme poměr průchodů přes danou hranu ve vzorci pojmenovaném jako *takeCount* vůči počtu průchodů přes všechny hrany ve vzorci *edgeCount*.

$$takeCountRatio = \frac{takeCount}{edgeCount} \tag{9}$$

Z tohoto poměru se spočítá kolik desítek procent zabírá daná hrana na celkovém pokrytí ($takeCountTens$).

$$takeCountTens = \lfloor takeCountRatio * 10 \rfloor \tag{10}$$

Nakonec se vypočte finální korekce hodnocení ($takeCountDeduct$) podle následujícího vzorce:

$$takeCountDeduct = takeCountRatio * (1 + takeCountTens) \tag{11}$$

Tato hodnota se již jen odečte od finálního hodnocení v jednotlivých statistikách a zaručí se tím tak, že čím vícekrát se přes danou hranu projde, tím více se zmenší její hodnocení a bude tím pádem horší.

5.3.1 Průměr

Tento způsob hodnocení se zakládá na porovnání aritmetických průměrů skutečného běhu a nalezeného histogramu. Vypočítá se z aritmetického průměru vah v histogramu $avgHist$ a aritmetického průměru ve skutečném průchodu avgRel.

$$Hodnocení = \frac{avgReal}{avgHist} - takeCountDeduct \tag{12}$$

5.3.2 Maximum

Hodnocení metodou maximum probíhá tak, že se porovná maximální priorita z histogramu $maxHist$ a maximální priorita z reálného průchodu $maxReal$

$$Hodnocení = \frac{maxReal}{maxHist} - takeCountDeduct \tag{13}$$

5.3.3 Minimum

Tato metoda je podobná jako metoda maximum, jen se počítá s minimem místo maxima. Minimum ze skutečného průchodu $minReal$ se vydělí minimem z histogramu $minHist$.

$$Hodnocení = \frac{minHist}{minReal} - takeCountDeduct \tag{14}$$

5.3.4 Medián

Metoda hodnocení na základě mediánu je založena na poměru mediánu ze skutečného průběhu $medReal$ a mediánu z histogramu $medHist$.

$$Hodnocení = \frac{medReal}{medHist} - takeCountDeduct \tag{15}$$

5.3.5 Průměr a maximum

Poslední metoda spočívá v sečtení průměru a maxima a následně porovnání jako u všech ostatních metod. Sečte se tedy průměr a maximum z reálného průchodu $avgMaxReal$ a stejně tak i z histogramu $avgMaxHist$.

$$Hodnocení = \frac{avgMaxReal}{avgMaxHist} - takeCountDeduct \tag{16}$$

5.4 Křížová validace

Křížová validace je použita při tvorbě modelu kvůli zpřesnění výsledků. Obecně se používá pokud není jisté, jak velký rozsah dat použít a aby nedošlo k přeučení nebo nedoučení v případě moc velkého respektive malého rozsahu dat. Konkrétně se zde křížová validace využije tak, že se rozdělí vstupní data do pěti skupin a poté se vytvoří pět modelů každý za pomoci čtyř skupin dat. Následně se vyhodnotí jejich přesnost předvídání na poslední zbývající skupině dat. Toto je znázorněno na obrázeku 7.

Model	Data				
Model 1	Test	Tvorba	Tvorba	Tvorba	Tvorba
Model 2	Tvorba	Test	Tvorba	Tvorba	Tvorba
Model 3	Tvorba	Tvorba	Test	Tvorba	Tvorba
Model 4	Tvorba	Tvorba	Tvorba	Test	Tvorba
Model 5	Tvorba	Tvorba	Tvorba	Tvorba	Test

Obrázek 7 Znázornění principu křížové validace.

Model strojového učení, který se ukáže být nejpřesnější se použije jako definitivní, uloží se na disk a bude se používat k predikcím při testování.

5.5 Vzorový příklad

V aktuálním stavu se spočítá histogram možných priorit při všech přípustných krocích. Dejme tomu že histogram bude mít následující podobu:

11.00	1.00	1.00	2.00	2.00	1.00	3.00	3.00	0.00	2.00

Tento histogram se uloží k hraně a bude se, společně s počtem průchodů přes danou hranu, používat k predikci a počítání hodnocení.

Po uběhnutí standardně čtyř kroků se spustí výpočet hodnocení. Pro hodnocení statistikou průměru proběhne výpočet dle těchto kroků:

1. Průměr z histogramu

$$avgHist = \frac{\sum_{i=0}^{9} Hist[i]}{10} \tag{17}$$

2. Průměr z předchozích hran

Průměr z předchozích hran se vypočítá z hran které model prošel od doby, co prvně prošel hranou s histogramem $Hist$ a aktuálním stavem (tj. čtyři kroky).

$$avgRel = \frac{\sum_{i=1}^{4} PrioritaHrany_i}{4} \tag{18}$$

3. Korekce počtem průchodů

Aktuálně se prošlo přes danou hranu třikrát a celkem je v aktuálním stavovém prostoru 57 hran. Výpočet korekce pro počet průchodů probíhá dle vzorce 11

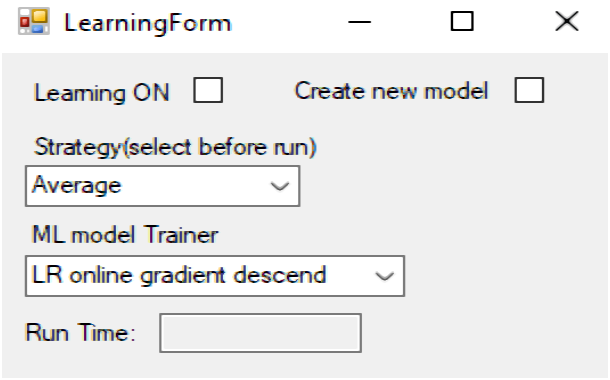
4. Získání konečného hodnocení

Celkové hodnocení je výsledek vzorce 12. V tomto případě je to $Hodnocení = 8{,}68$
Výsledný datový bod skládající se z vektoru příznaků a hodnocení bude vypadat takto.

11.0	1.0	1.0	2.0	2.0	1.0	3.0	3.0	0.0	2.0	3.0	8.68	0	0	7.60	7.88

5.6 Model strojového učení

Náplní této diplomové práce je mimo jiné vyzkoušet více modelů a metod strojového učení a také vyzkoušet, která kombinace bude produkovat nejlepší výsledky. Proto je implementováno více variant strojového učení a uživatel si může vybrat jakou chce v nastavení programu Taster, ukázka je zobrazena na obrázku 8. Uživatel může také využít možnosti vybrat si jakou statistiku použít pro hodnocení hran. V neposlední řadě je možnost povolit sběr dat pro učení a nebo upřednostnit tvorbu nového modelu před načítáním již vytvořeného modelu z disku.



Obrázek 8 Ukázka nastavení strojového učení.

5.7 Metody využívající strojové učení

Strojové učení bylo do nástroje Taster implementováno ve formě dvou metod. Jedná se o metody založené na principu metod dříve již implementovaných, a to Random a Systematic. Odvozené metody se strojovým učením jsou pojmenované AI Random a AI Systematic.

5.7.1 AI Random

Metoda AI random funguje na principu výběru náhodné hrany ze seznamu, ve kterém jsou hrany s větší prioritou zastoupeny víckrát, než hrany s nízkou prioritou. A to konkrétně tak, že pokud má hrana prioritu, určenou strojovým učením, osm bude v seznamu osmkrát, zatímco hrana které model strojového učení predikoval prioritu tři bude v seznamu jen třikrát.

5.7.2 AI Systematic

AI Systematic vybírá vždy nejlepší hranu podobně jako metoda Systematic, ale zde se nevybírá podle nejmenšího počtu průchodů, ale podle nejvyšší priority predikované strojovým učením.

U této metody je tedy důležité, aby se vhodně započítával počet průchodů přes danou hranu, jinak by docházelo k situaci, ve které se test zacyklí a bude procházet stavovým prostorem stále tou stejnou cestou dokola.

Z tohoto důvodu byla upravena korekce hodnocení založená na počtu průchodu, *takeCountDeduct* viz 11. Pro metodu AI Systematic byla zvýšena její váha dvacetkrát. Výsledný vzorec výpočtu *takeCountDeduct* pro metodu AI Systematic je následující:

$$takeCountDeduct = takeCountRatio * (1 + takeCountTens) * 20 \tag{19}$$

6 Experimenty

Hlavní náplní této diplomové práce je implementace a provedení experimentů s jednotlivými algoritmy strojového učení a následné vyhodnocení jejich výsledků. Cílem je najít algoritmus takový, který dokáže projít přes všechny hrany s vysokou relevancí co nejrychleji a zároveň pokrýt co nejvíce hran. V následujících kapitolách jsou popsány jednotlivé algoritmy a jejich výsledky porovnané s ostatními.

6.1 Definice experimentů

Jednotlivé experimenty se řídily následujícími pravidly.

- Doba běhu: 15min
- Počet běhů: 10

Použitý HW při testování není rozhodující, protože se metody porovnávají v závislosti na počtu kroků. Testy níže popsané se prováděly na ne příliš výkonném osobním notebooku s následujícími HW specifikacemi.

- AMD A4 quad-core
- 4GB RAM

Při běhu se po každém kroku vypisují navštívené relevance a tento seznam se následně využije k vyhodnocení. Po skončení všech deseti běhů se všechny experimenty oříznou, podle toho který obsahuje nejmenší počet kroků. A následně se z nich vytváří průměr a už jen stačí porovnat s ostatními metodami. Rychlost pokrytí hran modelu se počítá v závislosti na počtu kroků a ne na reálném čase z důvodu snahy o eliminaci vnějších vlivů.

6.2 Výsledky experimentů

V experimentech se snažím zjistit, které metody mají jaké vlastnosti při skutečném běhu. Mezi hlavní kritéria, které zde jsou probrána patří rychlost pokrytí specifických priorit, ať už jednotlivých nebo kumulačně sečtených, tím se myslí například pokrytí pro priority 7-9 atd. Dále pak celkové pokrytí všech hran.

6.2.1 Random metoda

První testovanou metodou je základní metoda Random, která vybírá další hranu náhodně. Tato metoda je součástí nástroje Taster od začátku a je tedy brána jako referenční metoda pro porovnání s ostatními nově implementovanými metodami ze skupiny AI Random. U této metody je velký rozdíl mezi jednotlivými běhy v pokrývání konkrétních hran, ale v celkovém pokrytí je výsledek srovnatelný. To jen potvrzuje, že výběr hran při procházení je skutečně náhodný.

6.2.2 Systematic metoda

Tato metoda patří také mezi základní metody, které jsou v nástroji Taster od začátku. Tato metoda slouží jako referenční při porovnání s metodami ze skupiny AI Systematic.

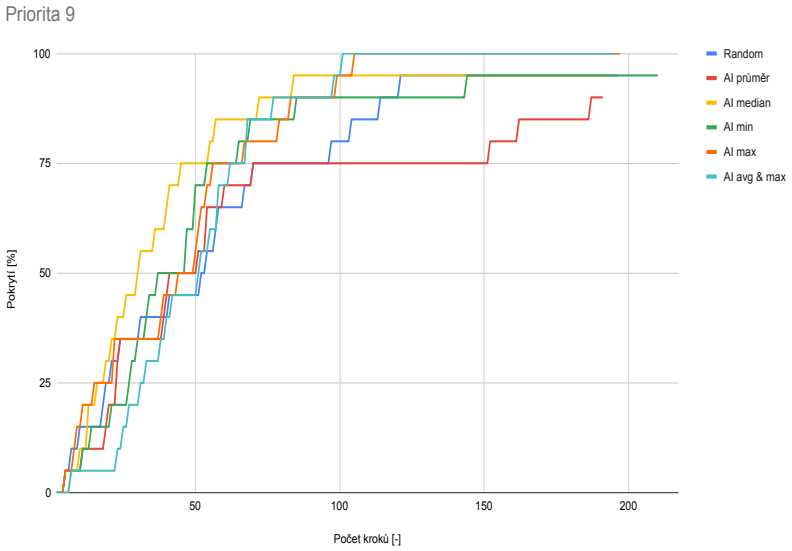
U této metody lze pozorovat větší rychlost v pokrytí grafu do určité meze. Metoda Random pokryje graf v nekonečném čase celý, ale metoda Systematic se v jisté chvíli zacyklí a nějaké hrany nikdy neprojde.

6.2.3 AI Random metoda

Metoda AI Random je založena na principu náhodného výběru ze seznamu hran, ve kterém jsou hrany s vysokou prioritou zastoupeny víckrát než hrany s nižší prioritou. U této metody jsou výsledky většinou lepší nebo stejné jako u metody Random, v závislosti na použité statistice k určení kvality hrany a v závislosti na zvolené metodě strojového učení.

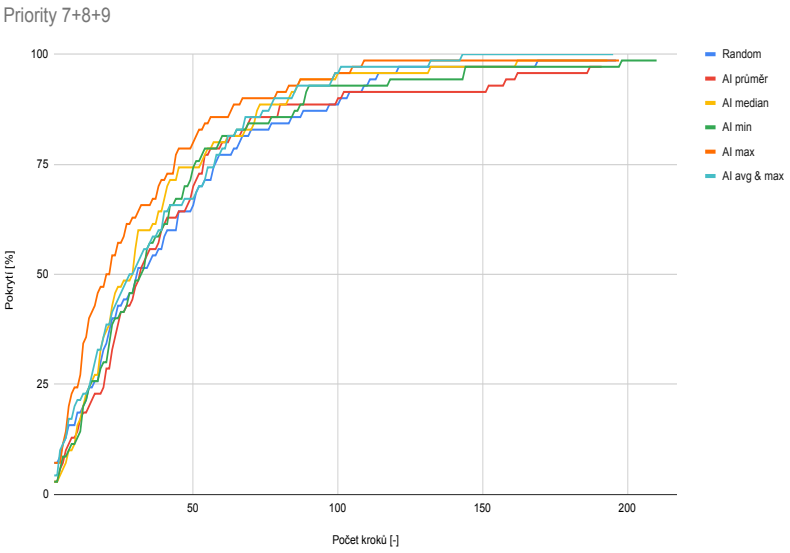
Lineární regrese (Online snižování gradientu)

Pokrytí nejvyšší relevance – Na obrázku 9 jsou znázorněny všechny druhy statistik AI Random metody pro lineární regresi s použitím optimalizátoru online snižování gradientu a Random metoda. Tento graf zobrazuje jejich výkon v pokrývání nejvyšší relevance, tedy relevance 9. Z grafu lze vyčíst, že statistika pro hodnocení hran založená na mediánu měla nejlepší rychlost pokrytí v ranných fázích průchodu a následně při větším počtu kroků se srovnala s ostatními metodami založenými na strojovém učení, kromě statistiky průměr, která vykazovala zvláštní chování způsobené pravděpodobně chvilkovým zacyklením v určité části stavového prostoru. Na druhé straně statistika založená na průměru a maximu měla ze začátku výrazně horší výsledky než ostatní metody, ale ke konci svůj výkon srovnala a celkové pokrytí nejvyšší relevance má srovnatelné s ostatními statistikami.



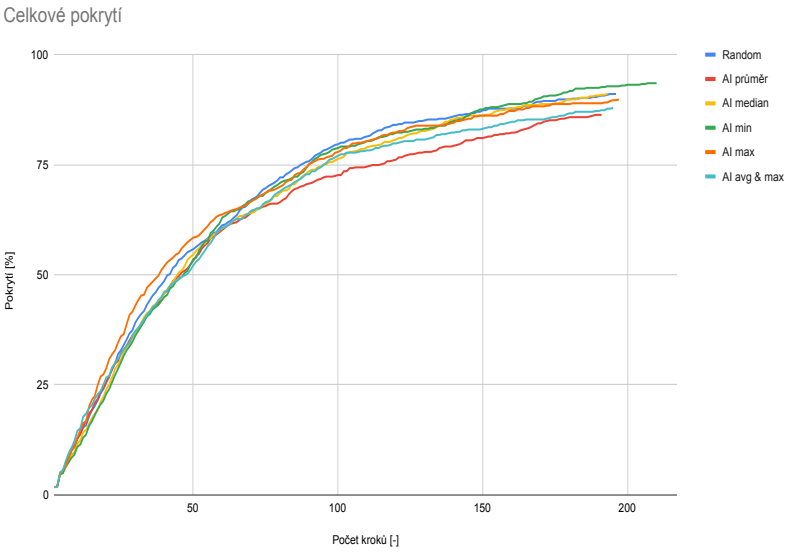
Obrázek 9 Rychlost pokrytí relevance 9.

Kumulativní pokrytí vysokých relevancí – Pokrytí relevancí 7 a výš je znázorněno na obrázku 10. Zde se ukázala, jako nejlepší statistika maxima. Metoda založená na statistice založené na mediánu, která vyšla nejlépe při pokrývání pouze relevance 9 byla horší, nicméně měla nepatrně lepší výsledky než ostatní metody.



Obrázek 10 Rychlost pokrytí relevancí 7,8 a 9.

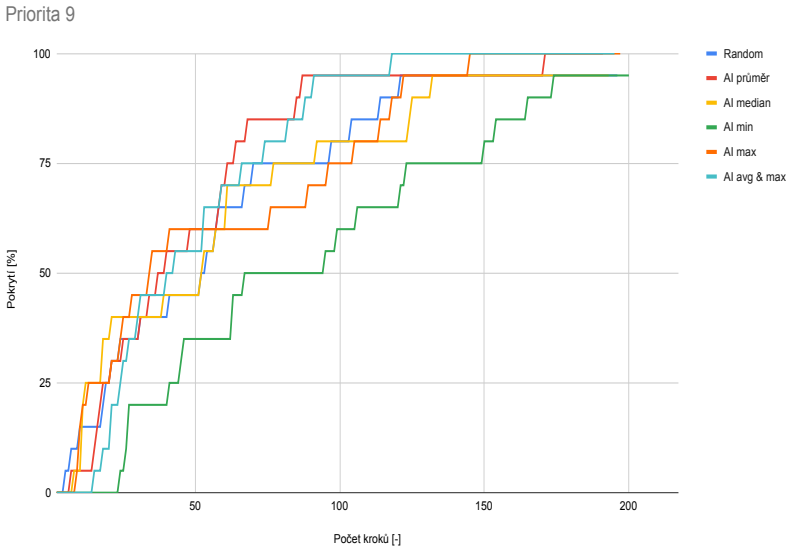
Celkové pokrytí hran – V celkovém pokrytí hran nebyly výsledky tak jednoznačné viz obrázek 11. Do zhruba poloviny běhu měla mírně nejlepší výsledky opět statistika maxima, nicméně při větším počtu kroků se začala propadat a nakonec byla se všemi metodami téměř identická. Jediná metoda průměru měla lehce horší výsledky než ostatní metody a to ve všech výše zmíněných grafech. Rychlost pokrývání všech relevancí by se dala ovlivnit změnou započítávání počtu průchodů přes hranu při výpočtu hodnocení.



Obrázek 11 Celkové pokrytí všech hran.

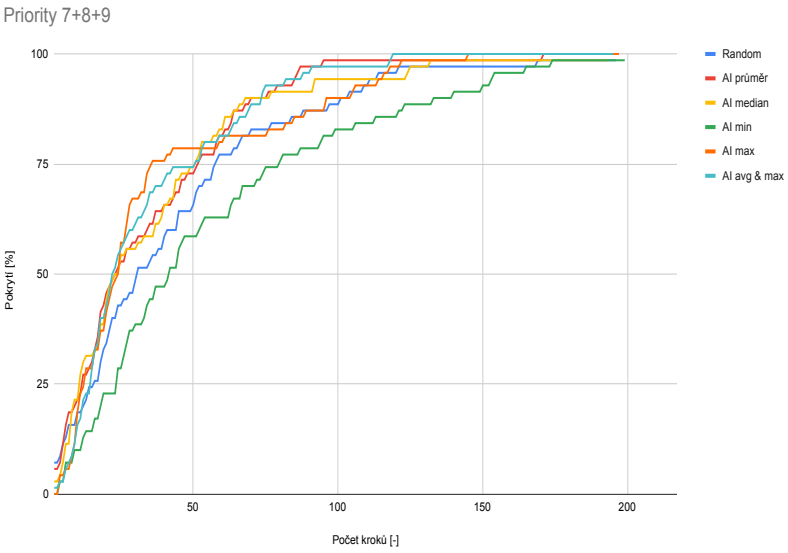
Lineární regrese (Poissonova regrese)

Pokrytí nejvyšší relevance – Na obrázku 12 jsou znázorněny výsledky metod založených na Poissonově regresi. Tyto metody pokrývaly nejvyšší relevanci přibližně stejně efektivně jako metoda Random, s jediným rozdílem u metody založené na statistice minima, která měla znatelně horší výsledky než všechny ostatní metody.



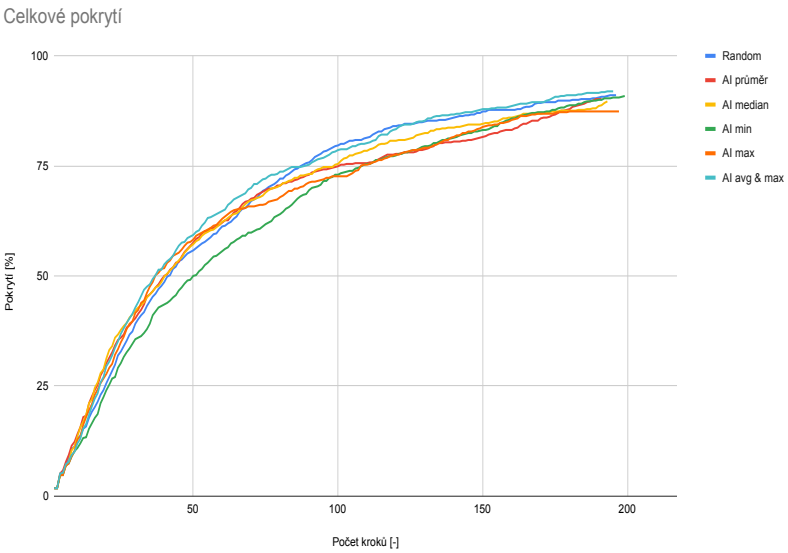
Obrázek 12 Rychlost pokrytí relevance 9.

Kumulativní pokrytí vysokých relevancí – Při započítání více relevancí, konkrétně relevancí větších než šest, se již všechny metody lišily od metody Random. Metoda založená na statistice minima byla stále znatelně horší, zatímco na druhé straně všechny ostatní metody využívající strojové učení měly lepší výsledky než metoda Random. Znázorněno je to na obrázku 13



Obrázek 13 Rychlost pokrytí relevancí 7,8 a 9.

Celkové pokrytí hran – Celkové pokrytí znázorněné na obrázku 14 ukazuje pro všechny metody téměř stejné výsledky jako u metod s optimalizátorem online snižování gradientu. U optimalizátoru Poissonova regrese je ale opět výrazně horší metoda založená na statistice minima.

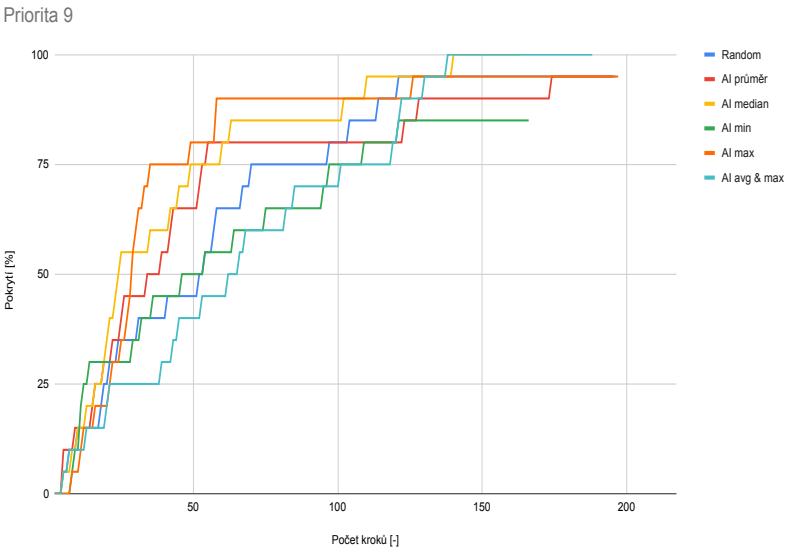


Obrázek 14 Celkové pokrytí všech hran.

Lineární regrese (SDCA)

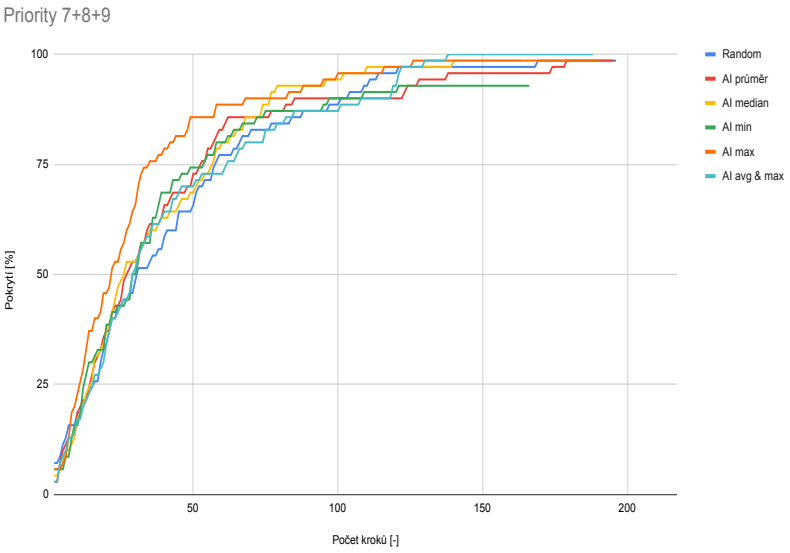
Při použití optimalizátoru metodou SDCA se model pro větší soubor dat nedokázal naučit. Proto jsem použil data jen ze 4 běhů, na kterých se model naučil a dával predikce.

Pokrytí nejvyšší relevance – Při prvním pohledu je u této metody vidět znatelný rozdíl oproti předchozím metodám. Statistiky maxima, mediánu a průměru jsou znatelně lepší při pokrývání nejvyšší relevance než metoda Random a zbylé dvě statistiky strojového učení, přičemž statistika maxima má nejlepší výsledek.



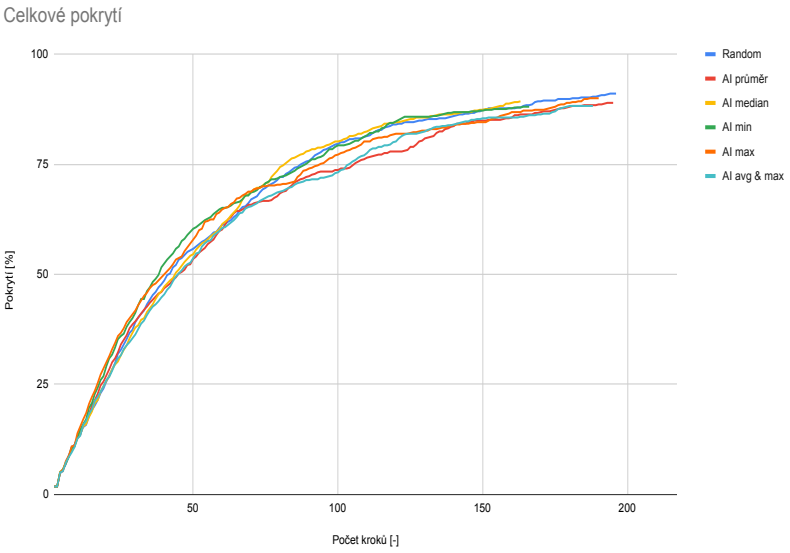
Obrázek 15 Rychlost pokrytí relevance 9.

Kumulativní pokrytí vysokých relevancí – Při započítání dalších dvou relevancí k nejvyšší relevanci se stále ukazuje výrazný přínos v rychlosti pokrývání u statistiky maxima. Ostatní metody se ovšem zhoršily v porovnání jen s nejvyšší relevancí a nyní jsou na tom všechny statistiky, kromě maxima, stejně jako metoda Random.



Obrázek 16 Rychlost pokrytí relevancí 7,8 a 9.

Celkové pokrytí hran – Při celkovém pokrytí hran je vidět jeden velký shluk a znamená to, že všechny metody pokrývají celý graf, včetně těch nejmenších relevancí, přibližně stejně rychle. Vzhledem k tomu, že k pokrytí vysokých relevancí dochází rychleji se strojovým učením než bez něj, je tento výsledek považován za úspěch.

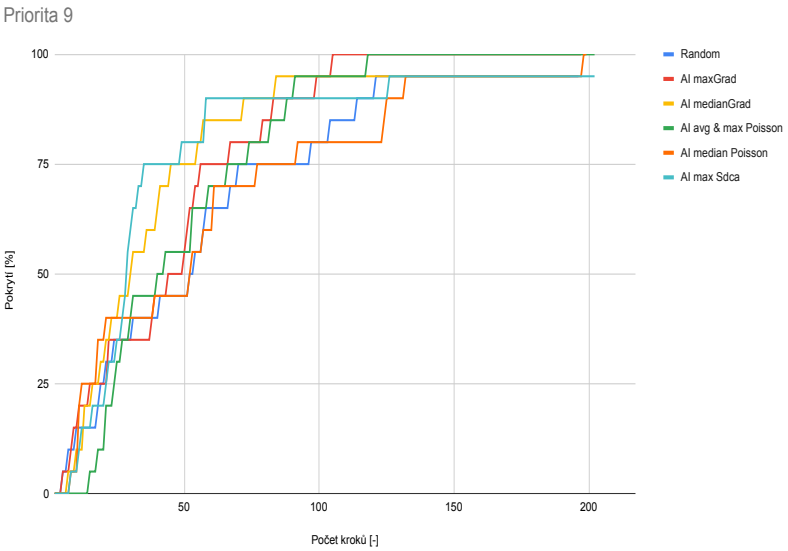


Obrázek 17 Celkové pokrytí všech hran.

Lineární regrese (Porovnání optimalizátorů)

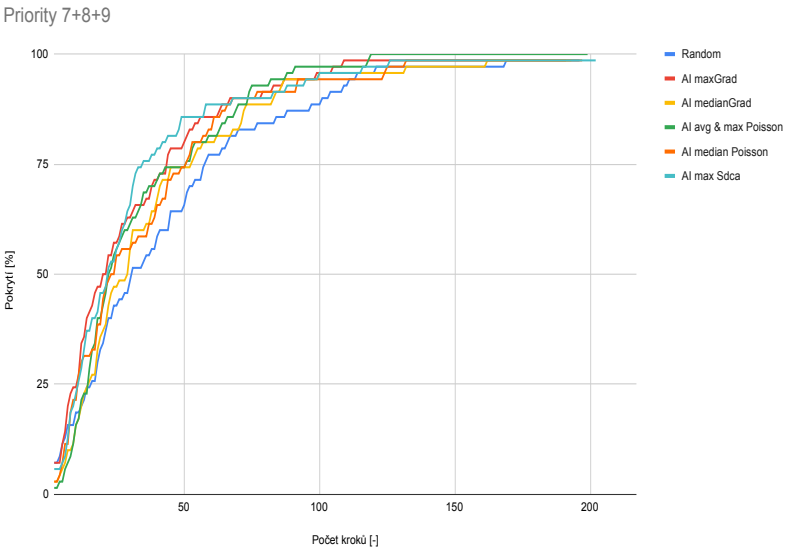
V této části jsou porovnané nejlepší metody z jednotlivých skupin reprezentovaných výše. Nejlepší metody z Online snižování gradientu (Medián a Maximum), Poissonovy regrese (Medián a průměr & maximum) a SDCA (maximum).

Pokrytí nejvyšší relevance – V pokrývání pouze nejvyšší relevance si Poissonova regrese nevedla vůbec dobře a nejlépe dopadla metoda založená na statistice mediánu s optimalizátorem online snižování gradientu společně s metodou založenou na statistice maxima a optimalizátorem SDCA. Znázorněno na obrázku 18



Obrázek 18 Rychlost pokrytí relevance 9.

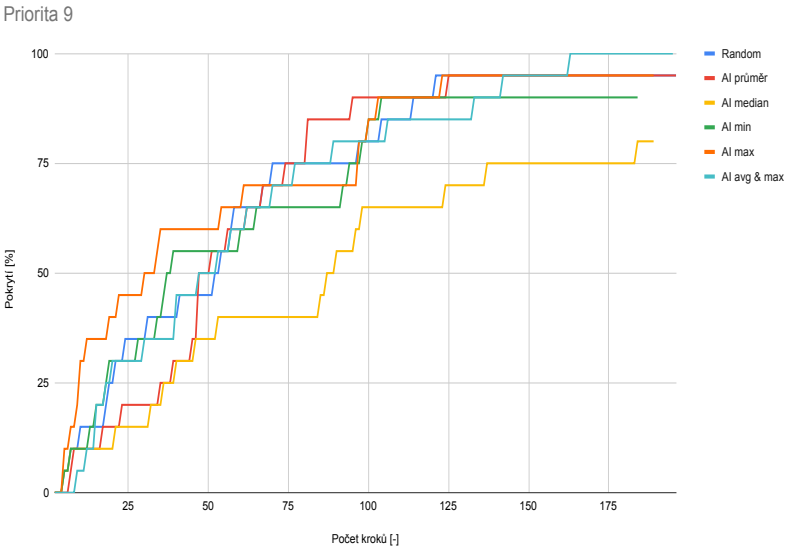
Kumulativní pokrytí vysokých relevancí – Při porovnávání nad třemi nejvyššími relevancemi byly již výsledky jasnější. V porovnání s metodou Random bez strojového učení byly všechny metody lepší. Nicméně při porovnání nejlepších metod z různých optimalizátorů vyšly všechny výsledky přibližně stejně. Pokrytí 75% dosáhla nejdříve statistika maxima s optimalizátorem SDCA. Konkrétně je to vidět na obrázku 19



Obrázek 19 Rychlost pokrytí relevancí 7,8 a 9.

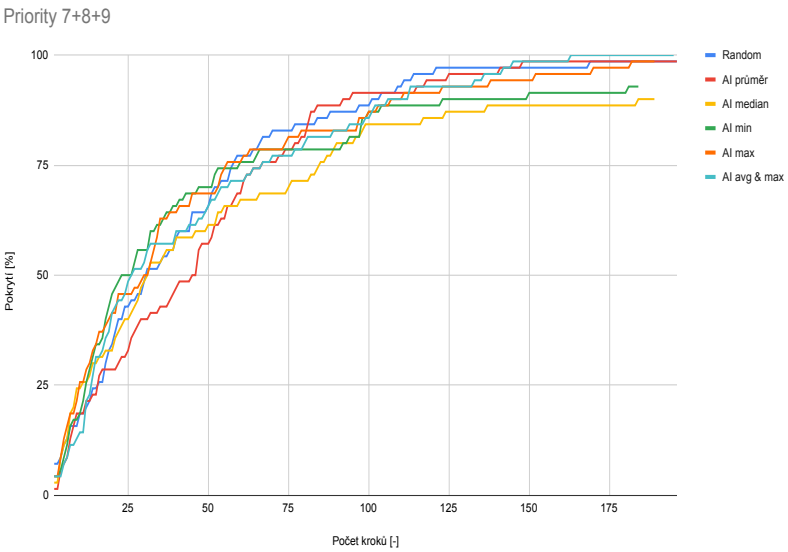
Řazení

Pokrytí nejvyšší relevance – Při využití metody řazení nebyly pozorovány moc úspěšné výsledky. Všechny statistiky měly rychlost pokrývání největší relevance zhruba stejnou jako metoda Random, jen statistika mediánu, která v jiných metodách produkovala poměrně dobré výsledky, byla znatelně horší.



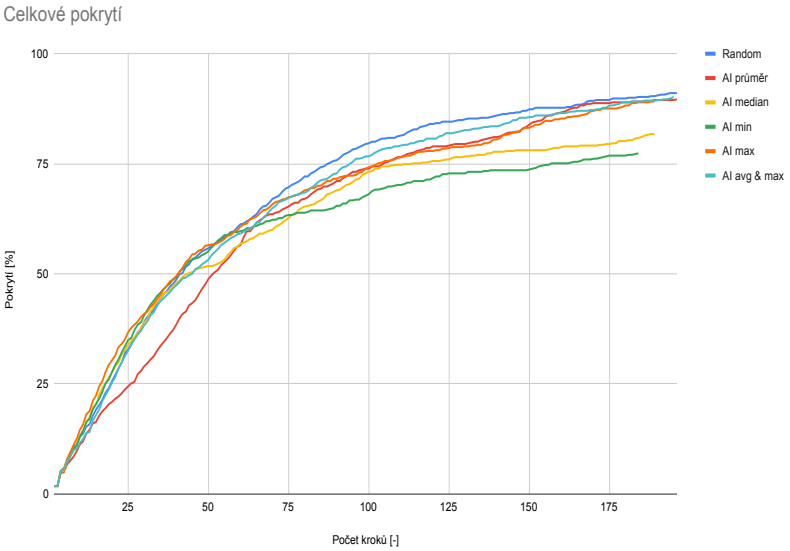
Obrázek 20 Rychlost pokrytí relevance 9.

Kumulativní pokrytí vysokých relevancí – Při započítání tří největších relevancí je stále výsledek statistiky mediánu nejhorší, nicméně již ne tak znatelně. Zbylé statistiky jsou pořád na stejné úrovni jako metoda Random.



Obrázek 21 Rychlost pokrytí relevancí 7,8 a 9.

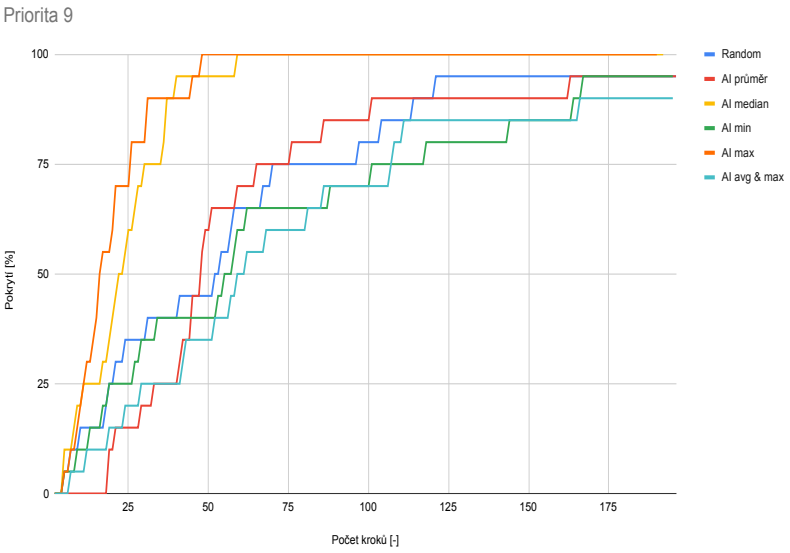
Celkové pokrytí hran – Při porovnání celkového pokrytí všech hran je vidět horší výsledek opět u statistiky mediánu a ještě horší u statistiky minima. Dále má v prvotních fázích procházení grafem (prvních 50 kroků) horší výsledek metoda průměru, zatímco všechny ostatní jsou naprosto totožné a až poté se zhorší statistika minima a mediánu.



Obrázek 22 Celkové pokrytí všech hran.

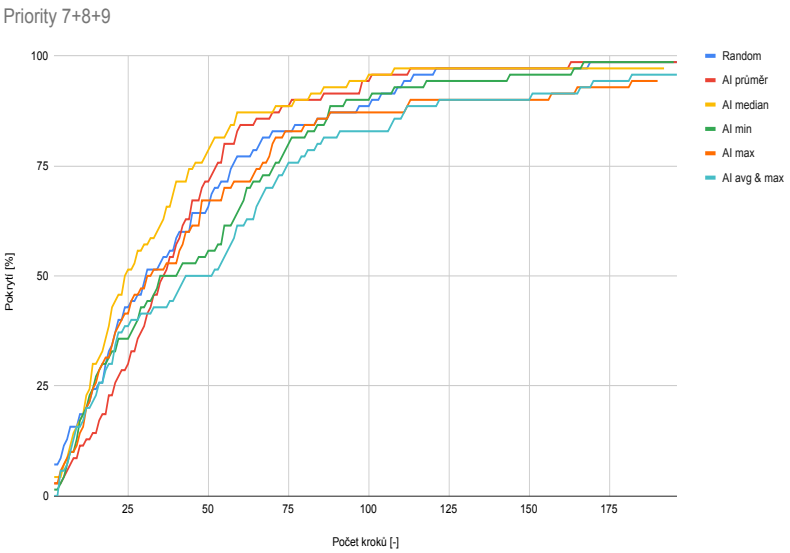
Víceúrovňová klasifikace

Pokrytí nejvyšší relevance – Při použití víceúrovňové klasifikace je vidět asi nejvýraznější rozdíl ve výkonu. Při pokrývání nejvyšší priority bylo u statistik maxima a mediánu zaznamenáno rekordně rychlé pokrytí. Ostatní statistiky měly zhruba stejné výsledky jako metoda Random.



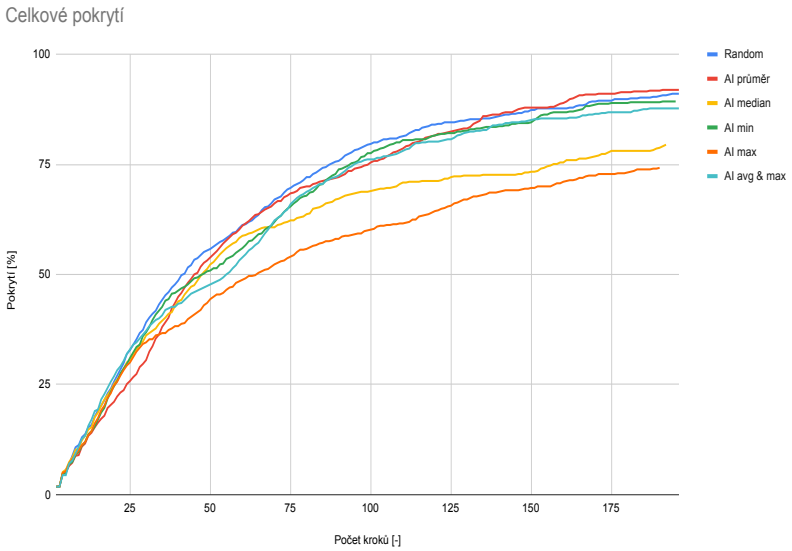
Obrázek 23 Rychlost pokrytí relevance 9.

Kumulativní pokrytí vysokých relevancí – Při měření pokrytí třech nejvyšších relevancí se výše zmíněný rozdíl smazal a zůstal jen mírně lepší výkon statistice mediánu, zatímco statistika maxima měla stejný výkon jako metoda Random a ostatní statisiky.



Obrázek 24 Rychlost pokrytí relevancí 7,8 a 9.

Celkové pokrytí hran – Při znázornění výkonu u pokrývání všech hran se všechny metody držely u sebe, až na metodu mediánu, která byla lehce horší a metody maxima, která byla nejhorší. Tento výsledek na celkové hodnocení statistiky mediánu není tak kritický, jako rychlost pokrytí vysokých priorit, a proto je tento výsledek považován za úspěšný.



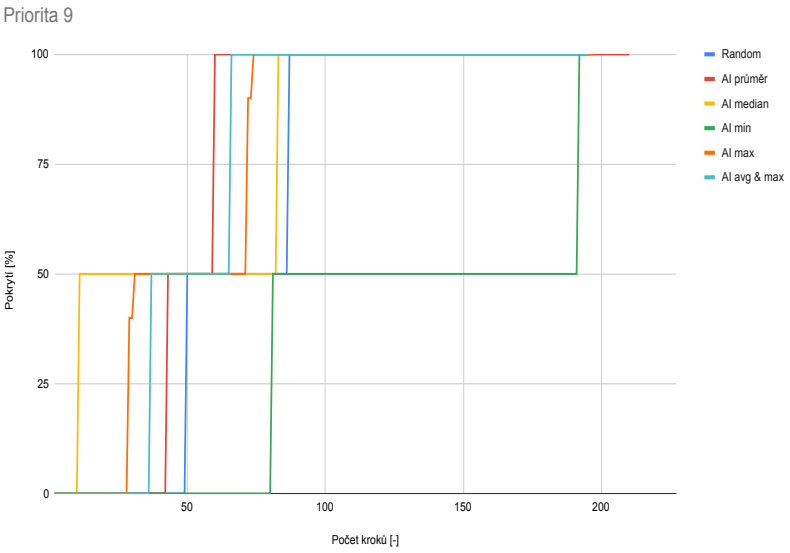
Obrázek 25 Celkové pokrytí všech hran.

6.2.4 AISystematic metoda

AI Systematic vychází z metody Systematic, která vybírá hranu s nejmenším počtem průchodů přes danou hranu. AI Systematic vybírá hranu, které přiřadí strojové učení nejvyšší prioritu. U této metody hraje velkou roli započítávání průchodu přes danou hranu. Pokud se tato hodnota nebude započítávat vůbec, tak se testování zacyklí a bude procházet neustále stejnou cestu. Z toho důvodu je výpočet hodnocení upraven a korekce zohledňující počet průchodů přes hranu je zvýšena dvacetkrát.

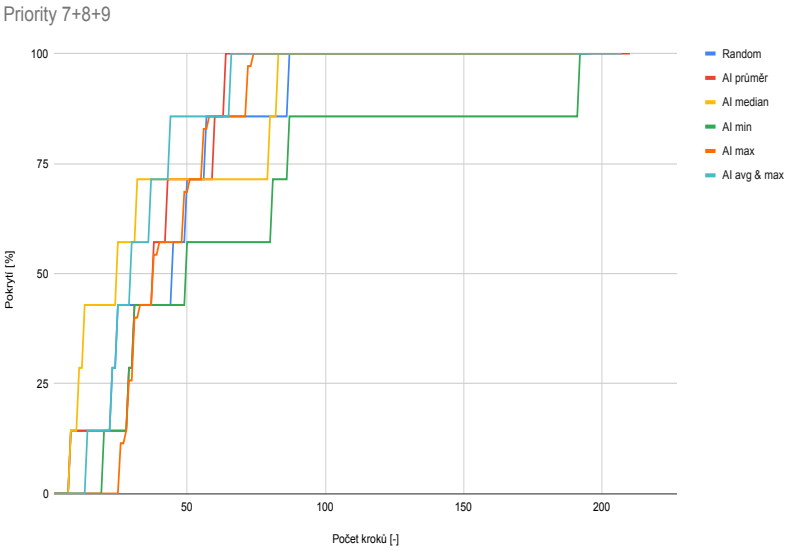
Lineární regrese (Online snižování gradientu)

Pokrytí nejvyšší relevance – V lineární regresi s optimalizátorem online snižování gradientu byly výsledky pro pokrývání nejvyšší relevance u většiny metod založených na strojovém učení lepší, než výsledky u metody bez strojového učení. Pouze metoda využívající statistiku minima měla horší výsledek. Porovnání je znázorněno na obrázku 26.



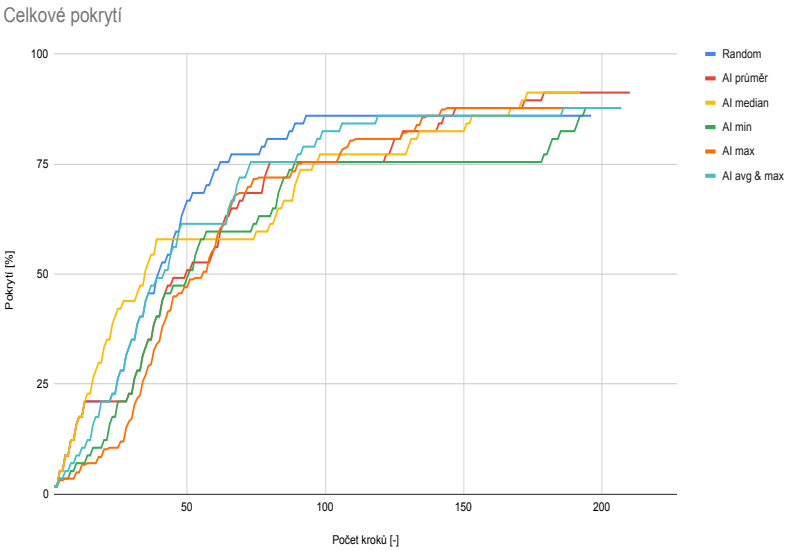
Obrázek 26 Rychlost pokrytí relevance 9.

Kumulativní pokrytí vysokých relevancí – Při zkoumání rychlosti pokrytí tři největších relevancí byly výsledky všech metod téměř totožné. Při dosažení pokrytí okolo 50% se začala metoda založená na statistice minima mírně propadat a celkového pokrytí nejvyšších relevancí dosáhla výrazně později než ostatní metody.



Obrázek 27 Rychlost pokrytí relevancí 7,8 a 9.

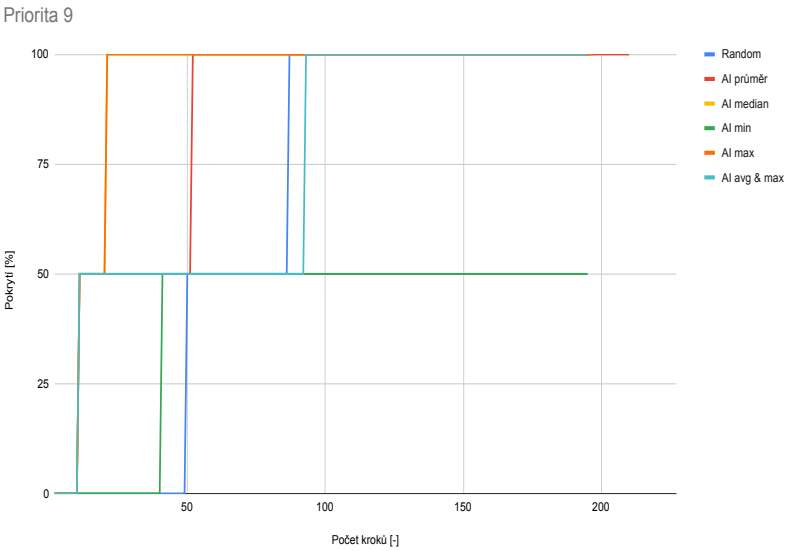
Celkové pokrytí hran – V celkovém pokrytí veškerých hran byly výsledky všech metod opět velice podobné. V prvních krocích dosáhla pokrytí 50% nejrychleji metoda se statistikou mediánu. Během celého běhu měly metody se strojovým učením lehce horší výsledky, než metoda bez strojového učení, nicméně metoda bez strojového učení dosáhla menšího maximálního pokrytí, než metody založené na strojovém učení. Vizualizováno je to na obrázku 28.



Obrázek 28 Celkové pokrytí všech hran.

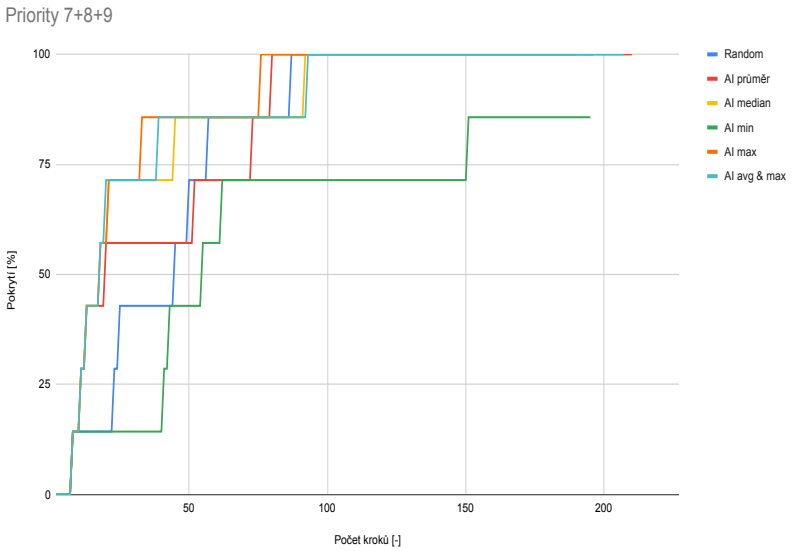
Lineární regrese (Poissonova regrese)

Pokrytí nejvyšší relevance – Porovnání rychlosti pokrytí nejvyšší relevance mezi metodami z třídy Poissonovy regrese ukázalo překvapivé výsledky. Metody založené na mediánu a maximu měly identický výkon a dosáhly 100% pokrytí již ve 21. kroku. Statistika průměru byla výkonnostně mezi výše zmíněnými dvěma a metodou bez strojového učení. Statistika kombinovaná z průměru a maxima byla do 50% stejně výkonná jako dvě nejlepší, ale vyššího pokrytí než 50% se dostala až po statistice bez strojového učení. Statistika minima měla výsledky horší než metoda bez strojového učení a dosáhla maximálně 50% pokrytí. Viz obrázek 29.



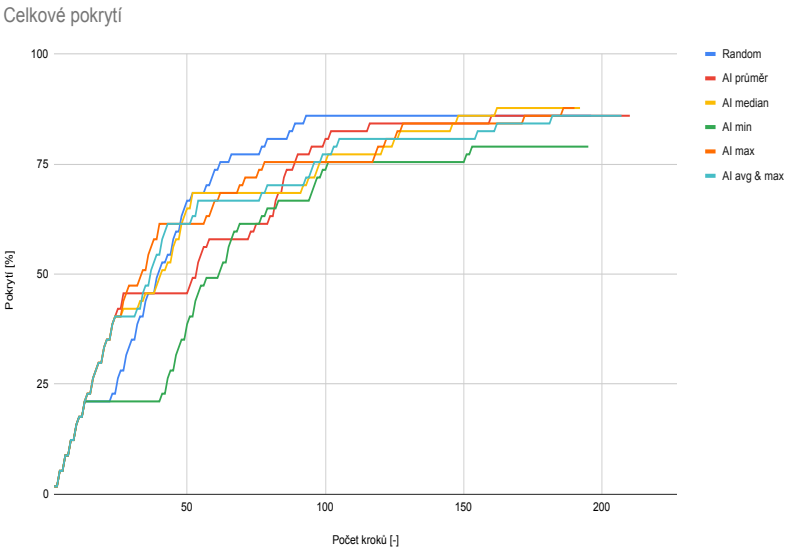
Obrázek 29 Rychlost pokrytí relevance 9.

Kumulativní pokrytí vysokých relevancí – Při započítání tří největších relevancí se rozdíl mezi jednotlivými metodami zmenšil. Nicméně metody vycházející ze statistiky maxima a mediánu měly stále nejlepší výkon a se stejným výsledkem skončila i statistika zkombinovaná z průměru a maxima. Statistika průměru byla zhruba stejná jako metoda bez strojového učení a statistika minima byla nejhorší, jak je vidět na obrázku 30.



Obrázek 30 Rychlost pokrytí relevancí 7,8 a 9.

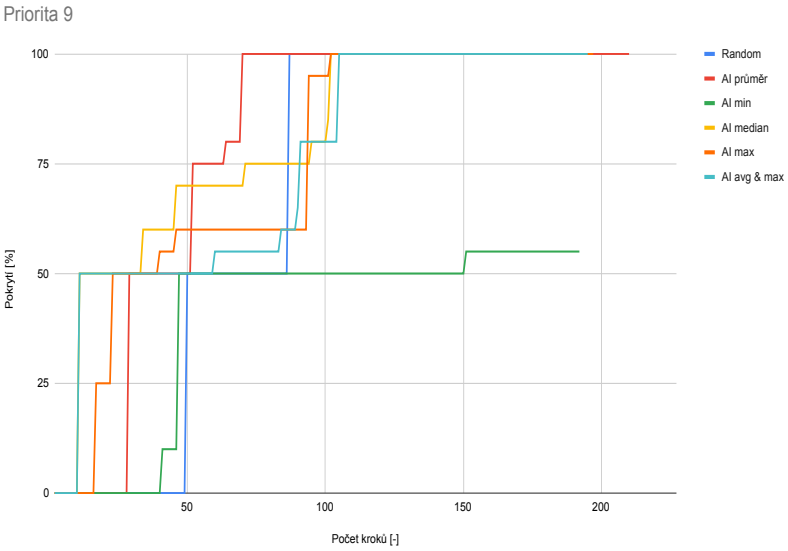
Celkové pokrytí hran – V celkovém pokrytí byly všechny statistiky, kromě statistiky minima, lepší než metoda bez strojového učení, v rychlosti pokrytí do 60%. Okolo 60% se metody se strojovým učením na pár kroků zacyklily mezi hrany, které již prošly a metoda bez strojového učení dosáhla 85% pokrytí nejdříve, ovšem ostatní metody postupně dosahovaly vyšších hodnot pokrytí. Podrobněji je toto chování znázorněno na obrázku 31.



Obrázek 31 Celkové pokrytí všech hran.

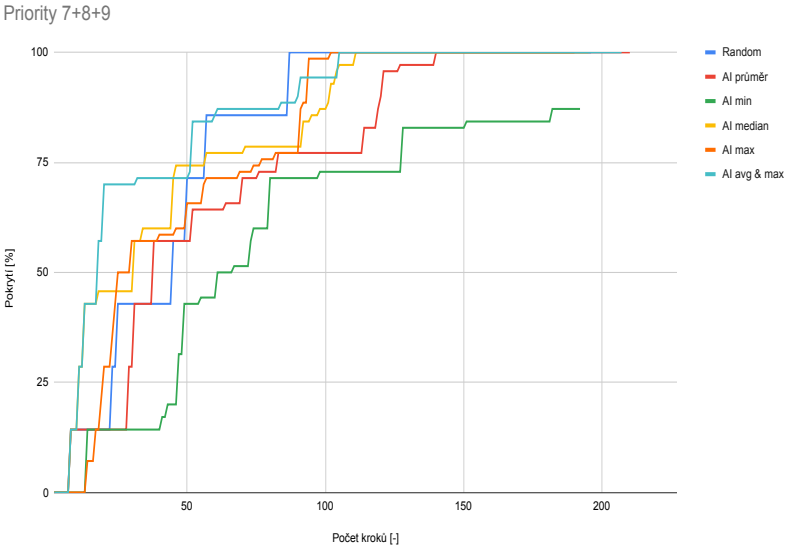
Lineární regrese (SDCA)

Pokrytí nejvyšší relevance – V metodě lineární regrese s optimalizátorem SDCA měly statistiky lepší výsledky než metoda random převážně v prvních fázích běhu. Pokrytí 50% dosáhly všechny statistiky se strojovým učením dříve než metoda bez strojového učení, ale pokrytí 100% dosáhla dříve jen statistika průměru. Ostatní statistiky dosáhly plného pokrytí o pár kroků později než metoda bez strojového učení.



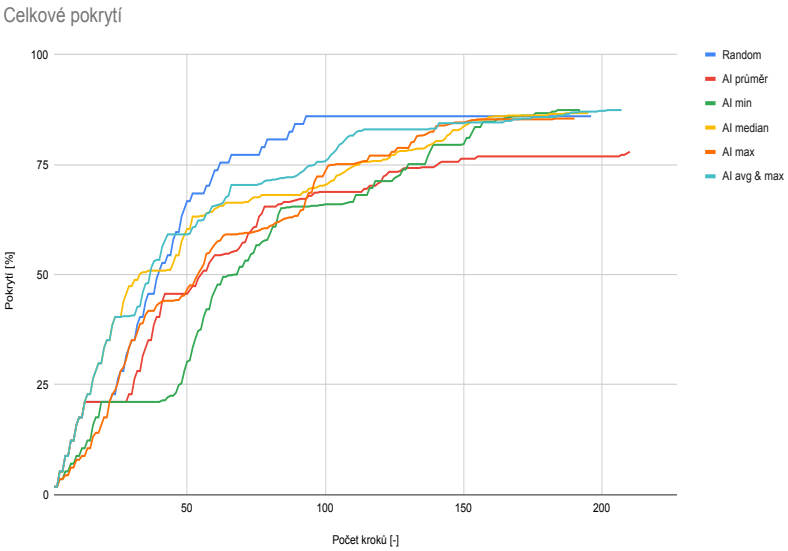
Obrázek 32 Rychlost pokrytí relevance 9.

Kumulativní pokrytí vysokých relevancí – Výkon metod při započítání třech nejvyšších relevancí byl horší, než při započítání pouze nejvyšší relevance. Pouze statistika založená na kombinaci průměru a maxima měla výsledek lepší, než metoda bez strojového učení. Naopak statistika minima měla výsledek znatelně horší než ostatní metody.



Obrázek 33 Rychlost pokrytí relevancí 7,8 a 9.

Celkové pokrytí hran – V celkovém pokrytí všech hran byly metody se strojovým učením pomalejší, ale metoda bez strojového učení nedosáhla maximálního pokrytí a po určitém počtu kroků se již nezlepšovala. Na druhou stranu metody se strojovým učením se zlepšovaly a po určitém počtu kroků by pravděpodobně prošly všechny hrany.



Obrázek 34 Celkové pokrytí všech hran.

7 Zhodnocení experimentů

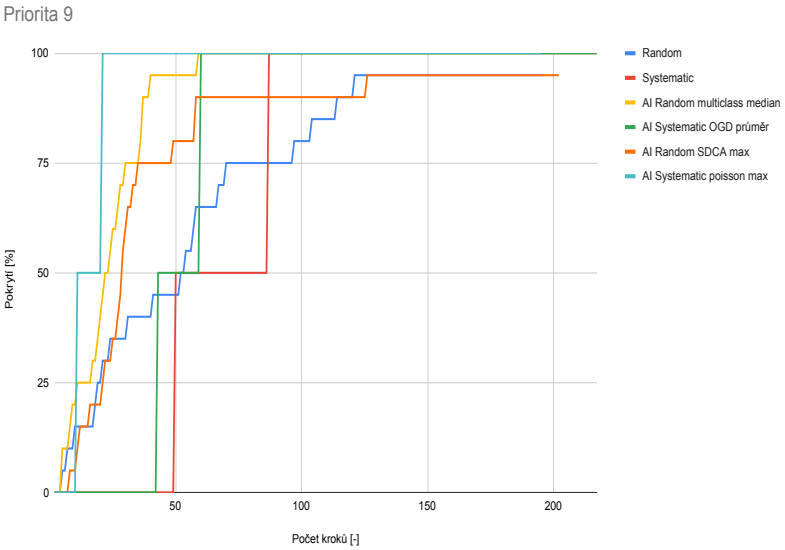
Provedené experimenty přinesly zajímavé výsledky. Téměř všechny kombinace strojového učení produkovaly lepší výsledky než metody bez strojového učení. Nejvýraznější zefektivnění průchodu se dá pozorovat u metod:

- AI Random, víceúrovňová klasifikace, statistika mediánu
- AI Random, Lineární regrese SDCA, statistika maxima
- AI Systematic, Linerární regrese online zvyšování gradientu, statistika průměru
- AI Systematic, Lineární regrese poissonova, statistika maxima

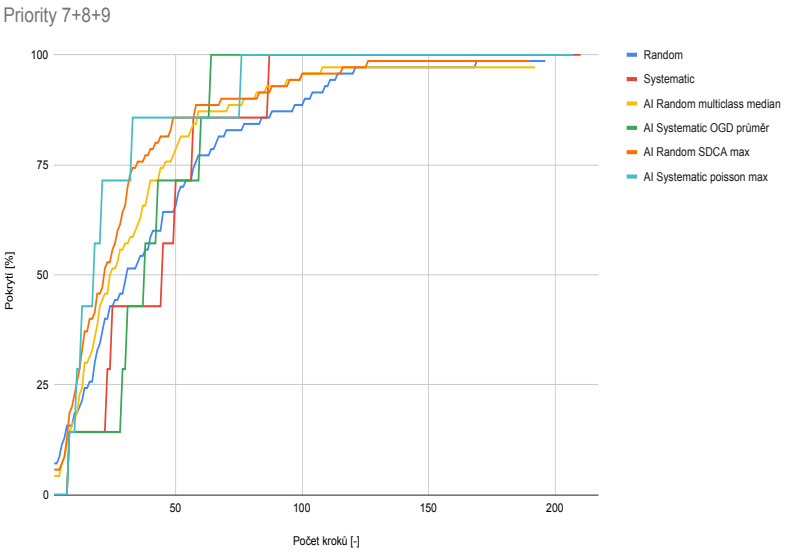
Konkrétní porovnání těchto metod a jejich protějšků bez strojového učení je znázorněno na obrázku 35 a na obrázku 36.

Některé metody se ovšem nedokázaly za stejných podmínek naučit a vytvořit model strojového učení, případně jejich výsledky byly příliš špatné. Například při trénování modelu pro lineární regresi s optimalizátorem SDCA jsem použil data na učení sesbíraná ze čtyř běhů namísto z deseti, jako pro ostatní metody. Dále pro všechny metody AI Systematic jsem používal upravenou statistiku hodnocení, u které jsem zvětšil váhu započítávání průchodu přes hranu dvacetkrát.

Všechny experimenty byly provedeny na modelu řídící jednotky ovládání kufru automobilu a prezentované výsledky nelze tedy zobecnit, ale dá se očekávat velmi podobné chování. Při zachování stejného principu hodnocení hran, je možné naučit strojové učení na jednom modelu a následně spouštět na jiném. V tomto případě by výsledky mohly být odlišné, ale při dostatečně rozsáhlém souboru dat pro učení se bude strojové učení chovat na všech modelech podobně.



Obrázek 35 Rychlost pokrytí relevance 9.



Obrázek 36 Rychlost pokrytí relevancí 7,8 a 9.

8 Závěr

Cílem této diplomové práce bylo rozšířit nástroj Taster o algoritmy využívající strojové učení. Nové metody byly implementovány pomocí frameworku ML.NET v jazyce C# a byly pojmenovány AI Random a AI Systemaic. Metody AI Random a AI Systematic vychází ze stávajících metod, konkrétně metody Random a Systematic. Strojové učení bylo implementováno s více druhy učitelů a více způsoby hodnocení. Konkrétně se jedná o učitele ze skupiny lineární regrese, víceúrovňové klasifikace a řazení. Způsoby hodnocení jsou založeny na principu průměru, mediánu, minima a maxima očekávaných relevancí. Pro tyto nové metody byla vytvořena nová třída TAMachineL a bylo přepracováno grafické uživatelské rozhraní. Dále byl také upraven formát modelů pro nástroj Taster, tak aby podporovaly přiřazování relevancí na hrany.

Další částí této diplomové práce bylo pomocí experimentů vyhodnotit kvality nových metod a případně navrhnout další úpravy. Vyhodnocení kvality probíhalo na modelu řídící jednotky ovládání kufru automobilu neboli modelu, který reprezentuje chování uživatele a automobilu při otevírání a zavírání kufru. Provedl jsem testy na všech variantách nových metod a porovnal jejich výsledky.

Zlepšení výkonu při testování by se dalo dosáhnout automatickým přiřazováním relevancí k hranám. Současně se relevance na hrany, kde se očekává porucha, přidávají ručně. Kdyby se tento způsob nahradil například algoritmem hodnocení využívající strojové učení, přineslo by to výrazné zrychlení celého procesu testování. Například pokud nový automobil využívá celý systém z jiného automobilu, je velice pravděpodobné, že tento systém bude fungovat bez problémů a tudíž by jeho testování mohlo mít nízké relevance.

Literatura

[1] Deepa Ramaswamy et al. "A Case Study in Hardware-In-the-Loop Testing: Development of an ECU for a Hybrid Electric Vehicle". In: (dub. 2). DOI: 10.4271/2004-01-0303.

[2] Eckard Bringmann a A. Kramer. "Model-Based Testing of Automotive Systems". In: květ. 2008, s. 485–493. ISBN: 978-0-7695-3127-4. DOI: 10.1109/ICST.2008.45.

[3] Rajeev Alur. "Timed Automata". In: *Computer Aided Verification*. Ed. Nicolas Halbwachs a Doron Peled. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, s. 8–22. ISBN: 978-3-540-48683-1.

[4] Aalborg university Uppsala universitet. *Uppaal home*. 2020. URL: https://www.it.uu.se/research/group/darts/uppaal/index.shtml (cit. 28. 01. 2020).

[5] Aalborg university Uppsala universitet. *Uppaal about*. 2020. URL: https://www.it.uu.se/research/group/darts/uppaal/about.shtml (cit. 28. 01. 2020).

[6] Marta Vomlelová. *Strojové učení*. URL: http://kti.mff.cuni.cz/~marta/uvod.pdf (cit. 14. 01. 2020).

[7] Google Developers. *Linear regression*. 2020. URL: https://developers.google.com/machine-learning/crash-course/descending-into-ml/linear-regression (cit. 28. 01. 2020).

[8] Adarsh Menon. *Linear Regression using Gradient Descent*. 2020. URL: https://towardsdatascience.com/linear-regression-using-gradient-descent-97a6c8700931 (cit. 28. 01. 2020).

[9] ML.NET community. *Linear regression online gradient descent documentation*. 2020. URL: https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.trainers.onlinegradientdescenttrainer?view=ml-dotnet (cit. 14. 01. 2020).

[10] ML.NET community. *Linear regression SDCA trainer*. 2020. URL: https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.trainers.sdcaregressiontrainer?view=ml-dotnet (cit. 14. 01. 2020).

[11] Cho-Jui Hsieh et al. "A Dual Coordinate Descent Method for Large-Scale Linear SVM". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. Helsinki, Finland: Association for Computing Machinery, 2008, s. 408–415. ISBN: 9781605582054. DOI: 10.1145/1390156.1390208. URL: https://doi.org/10.1145/1390156.1390208.

[12] Ryan Rifkin. *Multiclass Classification*. 2020. URL: https://www.mit.edu/~9.520/spring09/Classes/multiclass.pdf (cit. 28. 01. 2020).

[13] ML.NET community. *Multi class trainer - Maximum Entropy*. 2020. URL: https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.trainers.lbfgsmaximumentropymulticlasstrainer?view=ml-dotnet (cit. 28. 01. 2020).

[14] Hsiang-Fu Yu, Fang-Lan Huang a Chih-Jen Lin. "Dual coordinate descent methods for logistic regression and maximum entropy models". In: *Machine Learning* 85 (říj. 2011), s. 41–75. DOI: 10.1007/s10994-010-5221-8.

[15] ML.NET community. *Ranking trainer - FastTree*. 2020. URL: https://docs.microsoft.com/en-us/machine-learning-server/r-reference/microsoftml/rxfasttrees (cit. 28. 01. 2020).

[16] K. V. Rashmi a Ran Gilad-Bachrach. "DART: Dropouts meet Multiple Additive Regression Trees". In: *CoRR* abs/1505.01866 (2015). arXiv: 1505.01866. URL: http://arxiv.org/abs/1505.01866.