ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Metody výběru příznaků pro klasifikační a regresní úlohy
Student:	Bc. Pavel Verner
Vedoucí:	doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Studijní program:	Informatika
Studijní obor:	Znalostní inženýrství
Katedra:	Katedra teoretické informatiky
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Cílem práce je komplexně zmapovat všechny přístupy a metody pro výběry vhodných příznaků (atributů), které slouží pro klasifikační a regresní úlohy.

1) Seznamte se s hlavními myšlenkami metod pro výběry vhodných příznaků a metod pro klasifikaci a regresi.
2) Proveďte důkladnou rešerši metod pro výběry vhodných příznaků.
3) Na základě této rešerše identifikujte kategorie metod z různých hledisek (např. filtrační, wrapper a embedded metody, univarietní a multivarientní výběry, úplné, heuristické a náhodné vyhledávání apod.) a proveďte začlenění dohledaných metod do daných kategorií.
4) Z každé kategorie vyberte aspoň jednu typickou metodu a implementujte ji.
5) Implementované metody aplikujte na reálné úlohy a proveďte srovnání dosažených výsledků pro jednotlivé metody a úlohy.
6) Dosažené výsledky vyhodnoťte a diskutujte možnosti a omezení jednotlivých metod.

Seznam odborné literatury

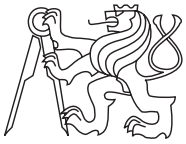
Dodá vedoucí práce.

doc. Ing. Jan Janoušek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 20. ledna 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra teoretické informatiky



Diplomová práce

Metody výběru příznaků pro klasifikační a regresní úlohy

Bc. Pavel Verner

Vedoucí práce: doc. RNDr. Ing. Marcel Jiřina, Ph.D

2. května 2017

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 2. května 2017 .....

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Verner, Pavel. *Metody výběru příznaků pro klasifikační a regresní úlohy.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Diplomová práce se zabývá analýzou metod selekce příznaků a následně porovnáním jejich výkonnosti. V první části jsou metody rozděleny do jednotlivých kategorií, obsahující společné znaky a vlastnosti. Dále rozebírá jejich matematický základ a princip fungování algoritmu k objasnění jejich použití. Realizační část se věnuje porovnání vybraných metod za použití vybraného vzorku datových souborů. Metody jsou porovnány z hlediska úspěšnosti na různých klasifikátorech, délky trvání selekce či ohodnocení a vztahu k datovému souboru. Na konci jsou uvedena doporučení, která reflektují vypozorované skutečnosti.

Klíčová slova Embedded metody, filtrační metody, klasifikační a regresní modely, ohodnocení atributu, selekce příznaků, výběr podmnožiny atributů, wrapperové metody

Abstract

Master's thesis deals with analysis of feature selection methods and its comparison. In the first part of the work we will divide method using their properties and common signs. Further we will analyse its mathematical basis and the principle of functioning. Practical part will focus on comparison selected methods using multiple data samples. Methods are compared based on their performance, time complexity and relationship to datasets. At the end of each section there is a recommendation, which reflect observed measurements.

Keywords Embedded methods, filter methods, classifiaction and regression models, attribute ranking, feature selection, subser search, wrapper methods

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Tématem diplomové práce jsou Metody výběru příznaků pro klasifikační a regresní úlohy. Příznaky či atributy dat si můžeme představit jako sloupce z relačních databází, které jsou nositelem nějaké více či méně relevantní informace. Klasifikační úloha představuje problém zařazení instance do nějaké třídy, kterou představuje diskrétní atribut s konečným počtem hodnot a seskupuje zbylé atributy podle podobných vlastností či vztahů a dochází k jeho logickému začlenění. Oproti tomu regresní úloha má tento atribut určující třídu spojitý, je tedy nekonečný a jedná se o problém predikce co nejpřesnější hodnoty.

Aby mohly tyto modely pracovat efektivně a rychle, je potřeba zredukovat dimenzi datového souboru, které můžeme dosáhnout dvěma způsoby. První možností je vytvoření nových atributů, které nahradí skupinu již existujících. To se nazývá *extrakce příznaků* a v této práci se metodami pracujícími s tímto principem zabývat nebudeme. Druhým způsobem je *selekce příznaků*, která z dostupných příznaků přidává ty nejvíce relevantní, případně odebírá naopak ty nejméně důležité. Právě tyto metody, které provádí selekci příznaků budou náplní této práce.

Selekce příznaků z datových souborů je rychle rozvíjející se obor dataminingu, kde se kladou nároky na rychlost trénování a dostatečnou přesnost modelu. Se stále narůstajícím objemem a důležitostí dat je potřeba držet krok s nároky na výkon a tím stále vznikají metody hodící se na odlišný druh problému. V praxi totiž nebývá časté, že jeden atribut má rozhodující vliv na třídu instance. Právě existence vzájemných vazeb atributů dělá z oboru selekce příznaků výzvu a existuje velké množství metod, schopných více či méně tyto vztahy odhalit a zpřesnit tím klasifikaci či regresi. Vazby mezi atributem a třídou mohou být různé, od lineárních přes monotónní až ke komplexním vztahům, k jejichž odhalení je potřeba specifických metod.

Hlavní náplní této práce bude metody selekce příznaků rozřadit do několika kategorií z hlediska společných znaků. Hlavní rozčlenění dělí metody podle typu učení, známé jako *filtrační*, *wrapperové* a *embedded* s různou rolí, kterou plní při trénování modelu. Jiným po-

hledem ke srovnání může být, zda jsou atributy vyhodnoceny dohromady - *multivarientní*, či zda vyhodnocující atributy jeden po druhém - *univarientní*. Další faktor, který může být důležitý při výběru vhodné metody je přítomnost nějakých nastavitelných parametrů, potřebná úroveň předzpracování dat či předpoklady pro datová rozdělení.

Další dělení je na úrovni podobnosti těchto metod v rámci dané kategorie. Metriky založené na ohodnocení atributů pracují s principy blízké statistice a matematice a atributy se snaží predikovat na základě jejich statistických vlastností, které lze dále seskupit. U ostatních se liší způsob optimalizace klasifikačních nebo predikčních modelů, případně směr hledání výsledné zredukované dimenze příznaků.

V druhé části této práce vybrané zástupce jednotlivých kategorií porovnáme na vzorových datových souborech s různým počtem atributů, tříd a velikosti. Tím se budeme snažit zobecnit jejich vlastnosti a odvodíme silné a slabé stránky jednotlivých metod na základě vypozorovaných chování. Porovnání metod bude probíhat z hlediska úspěšnosti na klasifikátorech, pro účel této práce budou použity *Support Vector Machines*, *k-Nejbližích sousedů* a *Naivní Bayesův klasifikátor*. Následně dojde k poměření časové náročnosti a vztahu k vlastnostem datového souboru, tedy zda například dochází s rostoucím počtem atributů k lepší úspěšnosti a naopak.

Kapitola 1

Analýza

V této kapitole budeme rozebírat funkčnost a teoretické vlastnosti jednotlivých metod. Metody zařadíme do jednotlivých kategorií podle podobných vlastností a zanalyzujeme jejich použití. Zmíníme také rozšíření jednotlivých metod a jaké představují vylepšení pro daný problém. Vybraní zástupci různých kategorií pak poslouží v implementační části k měření a porovnání.

1.1 Selekce příznaků

Tato kapitola se zabývá výběrem správných příznaků či atributů, které využijeme pro stavbu prediktivního modelu. Atributy můžeme chápat jako sloupce v relační databázi nebo charakteristiku jednotlivé populace. Čím užitečnější atributy jsou vybrány, tím lépe jsme schopni definovat daný datový soubor a jeho třídy. Cílem je vybrat takové atributy, které hodnotou co nejlépe rozdělují data do jednotlivých tříd v populaci a zároveň v této finální množině nejsou obsaženy redundantní či irelevantní atributy.

Omezení počtu takto vybraných příznaků slouží ke zvýšení efektivity klasifikátoru, neboť vyřazení nepotřebných (náhodných či nesouvisejících) příznaků urychlí celý proces učení a zvýší výslednou přesnost klasifikátoru. Atributy vhodné k vyloučení se nazývají *noise features*, které zvyšují klasifikační chybu pokud dojde k jejich zařazení do výsledné množiny atributů, tudíž se vyplatí tyto atributy vyřadit.

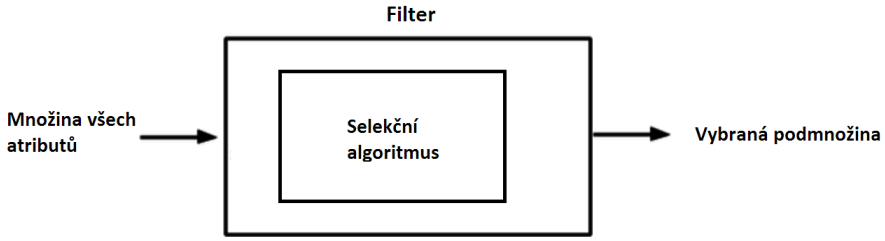
Existují různé metody této selekce, nejčastěji se uvádí dělení na metody **filtrační**, **wrapper** a **embedded** metody, které se tématicky dělí na další okruhy, avšak nelze všechny metody zaškatulkovat na konkrétní typ. Vlastnosti jednotlivých metod se občas prolínají.

1.2 Filtrační metody

Metody spadající do této kategorie jsou založené na ohodnocení jednotlivých atributů od nejlepšího po nejhorší (*ranking process*). Jedná se o nejjednodušší způsob selekce, ne-

boť využívá pouze statistické metody pro ohodnocení bez dalších vazeb na klasifikátor. Metody tedy nejsou závislé na typu učení, na rozdíl od embedded metod. Samotné vybrání konkrétní podmnožiny atributů probíhá až v následujícím kroku, což je rozdíl oproti wrapper metodám.

Filtrační metody bývají méně náročné z časového a implementačního hlediska než ostatní a hodí se pro získání prvotních informací o datovém souboru. Obecně můžeme filtrační metody definovat následovně: Mějme množinu atributů S na datech D a metriku $J(S|D)$, která určuje, jak relevantní je atribut S pro úlohu Y. Vzhledem tomu, že se data nemění a zůstávají fixní, můžeme si metriku zjednodušit na $J(S)$ značící relevanci atributu. Cílem je seřadit atributy na základě získané relevance na úloze Y a vybrat z nich k nejlepších, podle nějakého způsobu tvorby podmnožiny atributů.



Obrázek 1.1: Vizualizace filtračních metod

Standardní začlenění filtrovacích metod je rozděleno na metriky založené na **korelaci**, **statistické** metriky, **informačně-teoretické** metriky a **ostatní**. Rozřazení jednotlivých metod je znázorněno níže.

Zařazení metod

- Založené na korelaci
  - Pearsonův koeficient (*Univarietní*)
  - Spearmanův koeficient (*Univarietní*)
  - Kendallův koeficient (*Univarietní*)
- Statistické nezávislé metriky
  - Chí-kvadrát test (*Univarietní*)
  - T-test (*Univarietní*)
  - F-test (*Univarietní*)
- Informačně-teoretické metriky
  - Entropie (*Univarietní*)
  - Informační zisk (*Univarietní*)

- Ostatní

  Gini-index (*Multivarietní*)

Možné je ale i metody rozdělit dle toho, zda jsou **parametrické** či **neparametrické** [36], jejichž rozdíly jsou znázorněny v tabulce 1.2.

Sledovaný parametr	**Parametrické testy**	**Neparametrické testy**
Předpokládaná distribuce	normální	jakákoliv
Předpokládaná odchylka	homogenita	jakákoliv
Typická data	intervalové, poměrové	ordinální, nominální
Vztahy mezi atributy	nezávislost	jakýkoliv
Hlavní míra	průměr	medián
Benefity	statistická síla	jednoduchost, robustnost

Hlavním rozdílem tedy zůstává, že parametrické testy předpokládají normální distribuci dat, zatímco u neparametrických nemusí být rozdělení známé.

Nicméně neparametrické testy mívají jiná kritéria, které může být náročně splnit - například data ze všech tříd musí mít stejný rozsah, což může ovlivnit volbu testu. Výhodou neparametrických testů je fakt, že vyžadují většinou menší vzorek dat na provedení testu a celkově jsou univerzálnější (robustní na abnormality, nemusí být známé rozdělení dat). Za zmínku také stojí, že v nějakých případech oblast zkoumání daleko lépe popíše medián než průměr, což souvisí s odolností neparametrický testů vůči vysokým hodnotám.

Rozdělení popsaných filtračních metod dle tohoto kritéria by vypadalo následovně:

- Parametrické

  Pearsonův korelační koeficient

  T-test

  F-test

- Neparametrické

  Spearmenův korelační koeficient

  Kendallův korelační koeficient

  Chí-kvadrát test

  Informačně-teoretické metriky

  Gini-index

1.2.1 Metody založené na korelaci

Do této kategorie patří **Pearsonův**, **Spearmanův** a **Kendallův korelační koeficient**, kde každý z nich nabízí odlišný přístup k určení míry vztahu mezi atributem a třídou, avšak všechny pracují s vlastnostmi korelace.

Korelace
Ještě před popisem jednotlivých metod je potřeba přiblížit, co se pojmem korelace myslí. Korelace znázorňuje statistickou závislost dvou proměnných X a Y. Tyto proměnné jsou korelované, jestliže určité hodnoty jedné proměnné mají tendenci vyskytovat se společně s určitými hodnotami druhé proměnné. Dělíme tuto korelaci r na několik kategorií podle toho, co se stane s X, pokud Y poroste.

- $r = 1$. Atribut X je *lineárně závislý* na třídě Y.
- $r > 0$ a $r < 1$. S rostoucí hodnotou atributu X roste počet zařazení do třídy Y - *jsou kladně korelované.*
- $r = 0$. Atribut X s třídou Y nemají žádný vztah - jsou kompletně nekorelované.
- $r > 0$ a $r < 1$. S rostoucí hodnotou atributu X klesá počet zařazení do třídy Y - *jsou záporně korelované.*
- $r = -1$. Atribut X je *nepřímo lineárně závislý* na třídě Y.

Naším cílem je zjistit, zda nalezneme mezi vztahy proměnných nějakou závislost, či zda se nějak ovlivňují, tedy zda jsou korelované. Tuto skutečnost lze vypozorovat na obrázku 1.2, kde je graficky znázorněn vztah mezi různými proměnnými. Jednoduše odhadneme, zda jsou proměnné korelované, či nikoliv.

1.2.1.1 Pearsonův korelační koeficient

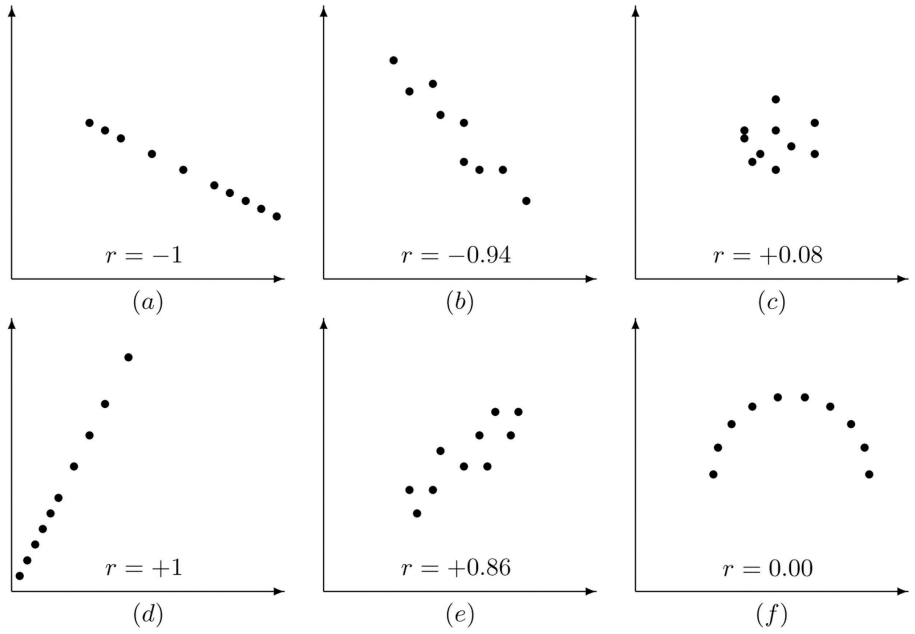
Pearsonův korelační koeficient je jednoduchá metoda, populární ve statistice, která slouží ke zjištění závislosti mezi proměnnými. V kontextu filtračních metod se myslí závislost mezi nějakým atributem X s hodnotami $x_1, \ldots, x_n$ a třídou Y s hodnotami $y_1, \ldots, y_m$.

Samotný výpočet se odvodí pomocí směrodatných odchylek proměnných a jejich kovariance.

$$\rho_{Pearson}(X,Y) = \frac{E(XY) - E(X)E(Y)}{\sqrt{\sigma^2(X)\sigma^2(Y)}} \tag{1.1}$$

Vzorec můžeme rozepsat na:

$$\rho_{Pearson}(X,Y) = \frac{\sum_i (x_i - x_i)(y_i - y)}{\sqrt{\sum_i (x_i - x_i)^2 \sum_j (y_i - y)^2}} \tag{1.2}$$



Obrázek 1.2: Vizualizace vzájemné korelace [19]

Výsledek se pohybuje typicky ve standardním intervalu $< -1; 1 >$ s vlastnostmi typickými pro korelaci a lze dle něho charakterizovat vzájemný vztah mezi třídou a atributem.

Výhodou této metody je bez pochyby její jednoduchost použití. Vcelku rychle lze ověřit závislosti mezi atributem a třídou. Bohužel však tato metoda není odolná vůči šumu a extrémům v datech, který velmi ovlivní tento koeficient. Pearsonův koeficient měří pouze lineární vztah mezi proměnnými, avšak v reálu většinou závisí více atributů společně na konkrétní třídě, případně vztah není přímo lineární, což tato metoda neumí zahrnout do výpočtu.

1.2.1.2 Spearmanův korelační koeficient

Spearmanův korelační koeficient je obecnější verze Pearsonova koeficinetu, který měří statistickou sílu monotónních vztahů mezi atributem a třídou. Obdobně jako u Pearsona i tento přístup vrátí výsledek v intervalu $< -1; 1 >$. Čím blíže se dostáváme k hraničním hodnotám, tím více je vztah monotónní.

Před výpočtem korelačního koeficientu potřebujeme znát ohodnocení atributu X a jeho třídy Y. Ten získáme určením pořadí hodnoty x_i, tedy atributu X na řádce i. Samotné ohodnocení x_i reprezentujeme číslem j, neboť x_i je právě j-tá nejvyšší hodnota v rámci X. Obdobně také zjistíme ohodnocení k, což představuje k-tou nejvyšší hodnotu v rámci Y, která je zároveň třídou pro x_i. Se znalostí vzdáleností těchto n dvojic můžeme vypočítat

Spearmanův korelační koeficient

$$\rho_{Spearmen}(X,Y) = 1 - \frac{6\sum_i D_i^2}{n(n^2-1)} \tag{1.3}$$

kde D_i představuje rozdíl (vzdálenost) mezi hodnocením atributu a třídy.

V závislosti na výstupní hodnotě můžeme určit vzájemný vztah dle následující příručky [46].

- 0,00 - 0,19: Velmi slabý
- 0,20 - 0,39: Slabý
- 0,40 - 0,59: Střední
- 0,60 - 0,79: Silný
- 0,80 - 1,00: Velmi silný

Oproti Pearsonovými koeficientu lze Spearmanův použít i na nelineárně závislé atributy, neboť zachycuje i obecně monotónní vztahy. Také je více rezistentní vůči odlehlým hodnotám a pro malé rozsahy dat je výpočet dokonce i méně pracný. Práce s touto metodou je ovšem náročnější a ve většině případu i trvá déle. Doporučuje se provést obě statistiky a následně je porovnat.

1.2.1.3 Kendallův korelační koeficient

Kendallův korelační koeficient je statistická metoda obdobná Spearmenovu koeficientu, která reprezentuje opět stupeň souladu dvou atributů (X, Y). Výstupní skóre je v intervalu $< -1; 1 >$, stejně jako u ostatních korelačních metod. Existuje více verzí tohoto koeficientu, nazývají se **tau-a**, **tau-b** a **tau-c**. Pro užití spojené se selekcí příznaků se nejčastěji užívá tau-b, který je i použitý v implementační části této práce.

Tau-a
Pro výpočet tau-a koeficientu potřebujeme nejprve spočítat počet shodných párů (C) a počet neshodných párů (D). K určení těchto počtů je nutné mít seřazené pole hodnot atributu X a k němu odpovídající zařazení do třídy Y. Následně pro každé x_0, x_1, $\ldots x_n$, kde n odpovídá velikosti instance, ověříme vztah páru (x_i, y_i) se všemi ostatními páry $\sum_{j=1, j\neq i}^{n} (x_j, y_j)$. Shodnost respektive neshodnost páru určíme následovně.

1. Pokud $x_i > x_j$, $y_i > y_j$ nebo $x_i < x_j$, $y_i < y_j$ - páry jsou **shodné**
2. Pokud $x_i > x_j$, $y_i < y_j$ nebo $x_i < x_j$, $y_i > y_j$ - páry jsou **neshodné**

Za každý nalezený shodný pár přičteme 1 do sumy shodných párů (značme C) pro konkrétní záznam i, tedy $C_i = C_i + 1$. V opačném případě zvýšíme počet neshodných párů (značme D) o jedna: $D_i = D_i + 1$. Ve výsledku součet shodných a neshodných párů pro konkrétní řádek musí být rovný velikosti instance bez jedné (neměříme shodnost se s sebou): $C_i + D_i = n - 1$.

Výsledný počet párů C je součtem všech shodných párů C_i, kde $i = 0, \ldots n$. Obdobně také pro počet neshodných párů. Finální vzorec na výpočet Kendallova tau-a koeficientu je následující.

$$\tau_a(X, Y) \quad = \quad \frac{C - D}{\frac{n}{2}(n-1)} \tag{1.4}$$

Tau-b
Kendallův tau-b koeficient se používá, pokud dochází ke shodě hodnot, které jsou porovnávány.

1. Pokud $y_i = y_j$ - páry jsou **shodné podle Y**
2. Pokud $x_i = x_j$ - páry jsou **shodné podle X**

Páry se shodnou hodnotou v atributu X nebo třídě Y značíme X_t, respektive Y_t. Pokud nastane shoda ve stejném páru jak v X, tak Y, pak není započítán ani do X_t, ani do Y_t. V tomto spočívá změna výpočtu Kendallova tau-b koeficientu.

$$\tau_b(X, Y) \quad = \quad \frac{C - D}{\sqrt{(C + D + Y_t)(C + D + X_t))}} \tag{1.5}$$

Kendallovo tau-b nabízí oproti Spearmanovi větší robustnost (menší chybu) a menší asymptotickou odchylku, tedy lepší efektivitu. Nicméně časová složitost výpočtu je $\mathcal{O}(n^2)$ s počtem n vzorků, oproti Spearmenovu koeficientu se složitostí $\mathcal{O}(n \log n))$. Existuje možnost využití modifikovaného merge sortu na seřazení pole, pak je složitost také jen $\mathcal{O}(n \log n)$, avšak pomalejší pro menší n.

Tau-c
Kendallovo tau-c je varianta pro větší instance optimalizující výpočet také pro nečtvercové tabulky. Změnou oproti tau-a je přidáním parametru m, který mění velikost tabulky. Celý vzorec pak vypadá následovně:

$$\tau_c(X, Y) \quad = \quad \frac{2m(C - D)}{n^2(m-1)} \tag{1.6}$$

Vlastnosti jsou však obdobné jako u tau-a verze.

1.2.2 Statistické metriky

Statistické metriky jsou sady testů, které využívají pravděpodobnost k ohodnocení jednotlivých atributů. Metriky mohou porovnávat průměrné hodnoty, odchylky či měří závislost jednotlivých atributů na základě vypozorovaných frekvencí. V této sekci se budeme zabývat **Chi-kvadrát testem**, **F-testem** a **T-testem**.

1.2.2.1 Chi-kvadrát test

Chi-kvadrát test nebo také test dobré shody je test na zjištění nezávislosti atributu a třídy.

Pokud je nějaký atribut X nezávislý na třídě Y, pak

$$P(XY) = P(X)P(Y) \tag{1.7}$$
$$P(Y|X) = P(Y) \text{ (X nedodá žádnou informaci k poznání Y)} \tag{1.8}$$

Počet párů hodnot, které obsahují hodnoty (x_i, y_j) z atributu $x_i \in X$ a třídy $y_j \in Y$ značíme N_{ij}. Tomuto počtu se také říká „vypozorovaná četnost“ (*observed frequency*).

Spočítáme také „očekávanou frekvenci“ (*expected frequency*) E_{ij}, definovanou jako $E_{ij} = \frac{N_i N_j}{n}$, kde N_i je součet hodnot přes řádky v matici N_{ij} a N_j je součet hodnot přes sloupce v matici N_{ij}. Hodnota n představuje počet vzorků.

Výsledný vzorec na výpočet Chi-kvadrátu má tuto podobu.

$$\chi^2 = \sum_i \sum_j \frac{(N_{ij} - E_{ij})^2}{E_{ij}} \tag{1.9}$$

Je potřeba vyvarovat se rovnosti $E_{ij} = 0$ použitím takových dat, která mají nenulový počet vazeb atributů v třídách. Případně se dají takovéto hodnoty nahradit libovolnými malými hodnotami.

Výsledek testu značí, jak moc se odchyluje očekávaná frekvence vůči naměřené frekvenci. Nízké hodnoty χ^2 indikují, že je jednotlivý atribut a třída nezávislá (očekávaná hodnota se od naměřené příliš neodchyluje), naproti tomu vysoké hodnoty indikují závislost.

Nevýhodou této metriky je, že měřené atributy musí být nezávislé - jedinec nezapadá do více než jedné kategorie. To může být občas problematické a často tuto informaci ani nemáme k dispozici. Dále se předpokládá také náhodný výběr hodnot z celé množiny dat, jinak může být statistika nevypovídající. Nespornou výhodou je však její jednoduchost, neklade žádné předpoklady o rozdělení dat a lze pracovat i s nominálními daty. Nicméně není možné testovat atributy se zápornými hodnotami.

1.2.2.2 T-test

T-test je často používaným parametrickým testem a existuje ve **jednovýběrové** a **dvouvýběrové** variantě.

Jednovýběrový T-test
Jednovýběrový test se používá v situacích, kdy známe střední hodnotu souboru μ, kterou následně považujeme za konstantu pro srovnání. Následně pak vybereme atribut X a ověřujeme, zda střední hodnota tohoto výběru je stejná, jako tato konstanta.

Pro výpočet potřebujeme u vybraného atributu X o velikosti n_X vypočítat průměr $\overline{X}$ a jeho rozptyl s_x. Hodnota jednovýběrového T-testu je pak rovna:

$$T_{test} = \frac{|\overline{X} - \mu|}{\sqrt{\frac{s_x^2}{n_x}}} \tag{1.10}$$

V praxi se jednobodový T-test příliš nepoužívá, jelikož většinou neznáme střední hodnotu μ. Rozšířením je dvouvýběrový test, který porovnává dvě proměnné, atribut vůči třídě.

Dvouvýběrový T-test
V případě dvouvýběrového T-testu (také studentova T-testu) se používá jeho nezávislá a nepárová verze, kde neznáme střední hodnotu souboru a pracujeme s předpokladem, že proměnné X a Y mají shodné rozptyly. Pak obdobně jako u jednobodového testu vypočítáme skóre pouhým nahrazením střední hodnoty průměrnou hodnotou třídy Y, značme $\overline{Y}$.

$$T_{student} = \frac{\overline{X} - \overline{Y}}{s_p^2\sqrt{\frac{1}{n_X} + \frac{1}{n_Y}}} \tag{1.11}$$

Nezávislý dvouvýběrový T-test předpokládá normální rozdělení dat a shodné rozptyly jednotlivých proměnných. Avšak ve své podstatě je celkem robustní a odolný a představuje jakousi první volbu pro měření statistických vlastností atributů.

Welchův T-test
Welchův T-test představuje vylepšení studentova T-testu, který se používá ke zjištění, zda se průměry atributu a třídy od sebe statisticky liší v případě rozdílných rozptylů.

U T-testu s nerovností rozptylů není cílem zjistit, zda jsou průměry populací shodné, ale jak daleko od sebe jednotlivé průměry jsou. Welchův T-test oproti studentova T-testu tedy dělí odchylku v každé skupině hodnotou představující velikost dané skupiny n.

$$T_{welch} = \frac{\overline{X} - \overline{Y}}{s_p^2\sqrt{\frac{s_X^2}{n_X} + \frac{s_Y^2}{n_Y}}} \tag{1.12}$$

Zatímco studentův T-test předpokládá, že třídy populace mají normální rozdělení a shodnou rozptylů (standardní nepárový T-test), Welchův T-test předpokládá sice také normální rozdělení, avšak nepředpokládá shodnost rozptylů. Nicméně pokud se rozptyly rovnají, oba testy vrátí stejnou p-hodnotu. Informaci, zda se rozptyly shodují můžeme zjistit například na základě F-testu.

1.2.2.3 F-test

F-test představuje test významnosti dvou rozptylů, který ověřuje jejich shodu na základě vstupních hypotéz a povolené chyby. Prvním krokem je spočítání rozptylu obou atributů

pomocí následujícího vzorce:

$$\sigma_X^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x_i})^2 \tag{1.13}$$

Počet hodnot v atributu X značí proměnná n s průměrnou hodnotou $\overline{x_i}$. F-test je jednoduché porovnání takto vypočítaných rozptylů.

$$F_{test} = \frac{\sigma_X^2}{\sigma_Y^2} \text{ za předpokladu, že } \sigma_X^2 > \sigma_Y^2 \tag{1.14}$$

Výsledkem je poměrová hodnota. Tedy pokud je výsledek rovný jedné, rozptyly populací se rovnají, případně můžeme rozhodnout o zamítnutí či nezamítnutí hypotézy o rovnosti rozptylů s definovanou pravděpodobností.

ANOVA F-test
ANOVA používá F-distribuci k analýze rozptylů a porovnává vztah tří a více proměnných o proti normálnímu F-testu.

Mějme atribut X, který můžeme rozdělit do M tříd $y_0, y_1, \ldots, y_m \in Y$. Pak můžeme vypočítat celkový průměr $\overline{X}$ a celkový rozptyl σ_X^2 přes všechny třídy. Potřebujeme ale také nalézt rozptyl a průměr v rámci jednotlivých tříd y_i. ANOVA je založena právě na analyzování původu celkového rozptylu na základě znalosti jednotlivých podtříd. Pro každý vzorek x_{ij} třídy y_i totiž můžeme určit:

1. Jaká je vzdálenost vzorku x_{ij} od průměru i-té třídy $\overline{x_i}$
2. Jaká je vzdálenost vzorku x_{ij} od celkového průměru $\overline{X}$

Tento vztah vyjádříme poměrem rozptylu „mezi třídami“ a rozptylu „uvnitř tříd“, což představuje celkový výsledek ANOVA F-testu.

Výpočet začíná vypočítáním rozptylu *mezi třídami*, což je dost obdobná formule jako u obyčejného F-testu.

$$\sigma_{mezi}^2 = \frac{\sum_{i=1}^{M} n_i(\overline{x_i} - \overline{X})^2}{M-1} \tag{1.15}$$

Proměnná n_i značí počet vzorků v třídě y_i. Následně zbývá již jen spočítat rozptyl *uvitř třídy*

$$\sigma_{uvnitř}^2 = \frac{\sum_{i=1}^{M}\sum_{j=1}^{n_i}(x_{ij} - \overline{x_i})^2}{n-M} \tag{1.16}$$

s proměnou n reprezentující celkový počet řádek (velikost instance). Výsledné skóre získáme podílem těchto rozptylů.

$$F_{test_{ANOVA}} = \frac{\sigma^2_{mezi}}{\sigma^2_{uvnitř}} \tag{1.17}$$

Výhodou tohoto testu je jeho použitelnost na více tříd a poskytuje analýzu, zda pocházejí z populací o stejném průměru, či nikoliv. F-test také nevyžaduje, aby měly jednotlivé třídy stejný rozptyl, jako je tomu u T-testu.

1.2.3 Informačně-teoretické metriky

Informačně-teoretické metriky jsou metriky (míry) založené na **entropii**, která ohodnocuje atributy dle jejich důležitosti. Základní princip metody je odvozen z entropie, která nám udává míru neurčitosti obsaženou v daných atributech.

Entropie pro atribut X se značí $H(X)$ a počítá se

$$H(X) = -\sum_i P(x_i) \log_2(P(x_i)) \tag{1.18}$$

kde $P(x_i)$ značí pravděpodobnost výskytu hodnoty x_i. Nejvyšší entropie tedy dosáhneme při rovnoměrném rozložení hodnot. Pokud se hodnoty v atributu X nacházejí pouze v jedné třídě, entropie je nulová.

Rozšířením definice pro dvě neznáme získáme *sdruženou entropii*, pro kterou platí $H(X,Y) \leq H(X) + H(Y)$ s rovností v případě nezávislosti veličin. Hodnotu sdružené entropie (*joint entropy*) získáme:

$$H(X,Y) = -\sum_i \sum_j P(x_i, y_j) \log_2 P(x_i, y_j) \tag{1.19}$$

Tím máme již vše připravené pro výpočet **Informačního zisku** (IG) nebo občas uváděné **Vzájemné informace** (MI), čímž jsme schopni dobře posoudit důležitost atributu.

$$IG(X,Y) = H(X) + H(Y) - H(X,Y) \tag{1.20}$$

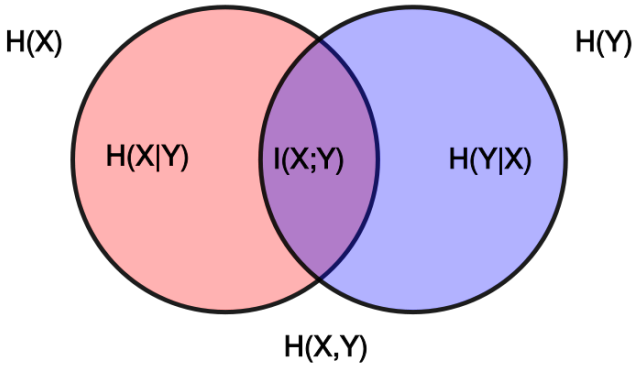
$$IG(X;Y) = H(Y) - H(Y|X) \tag{1.21}$$

$$IG(X,Y) = -\sum_{i,j} P(x_i, y_j) \log_2 \frac{P(x_i, y_j)}{P(x_i)P(y_j)} \tag{1.22}$$

Ve výpočtu se objevuje ještě *podmíněná entropie*, tedy míra neurčitosti proměnné Y za předpokladu pozorování atributu X, čili zbývající neurčitost Y se známostí atributu X.

$$H(Y|X) = -\sum_j P(x_j) \sum_i P(y_i|x_j) \log_2(P(y_i|x_j)) \tag{1.23}$$

Informační zisk je skóre, které je citlivé na hledání vztahu mezi atributem a třídou a hodí se i pro nelineární závislosti. Nicméně výpočet trvá déle v porovnání s ostatními filtračními metodami.



Obrázek 1.3: Vztah informačního zisku a entropie [23]

Pokud jsou atributy X a Y nezávislé, pak platí $H(Y) = H(Y|X)$ a informační zisk je tedy nulový. Vztahy mezi entropiemi a informačním ziskem je znázorněn na obrázku 1.3, z kterého je patrné, že způsobů výpočtu může být více.

Čím větší hodnota informačního zisku, tím je atribut užitečnější, neboť $H(Y|X)$ bude blízké nule, což značí dobré zařazení do tříd. Pro každou hodnotu takových atributů bude při nejlepším existovat pouze jedna třída.

Pokud však atribut obsahoval jedinečné hodnoty (například umělý klíč), informační zisk by byl maximální (jedné hodnotě odpovídá jeden záznam), avšak přesto je tento atribut nevhodný pro klasifikaci. Na trénovací množině by na základě těchto hodnot bezchybně určil třídu, nicméně na nových datech by byl zcela nepoužitelný.

Výše uvedený vzorec pro Informační zisk můžeme tedy upravit, aby uvažoval i počet hodnot atributu. Tato úprava se nazývá **Poměrný informační zisk** (IR)

$$IR(X;Y) = \frac{I(X;Y)}{H(X)} \tag{1.24}$$

který nabývá hodnot od nuly do jedné a je více robustní co se týče ohodnocení atributů.

1.2.4 Ostatní

Existuje spousta dalších metod, které nespadají do žádné kategorie a jsou více či méně použitelné pro problém selekce příznaků. Pro srovnání je v této sekci popsáno ohodnocení podle **Gini-index**, který je hojně využíván hlavně v ekonomii.

1.2.4.1 Gini-index

Gini-index je metoda sloužící pro oddělování „nečistých“ atributů z dat. Patří mezi multivarietní algoritmy s dohledem.

Mějme datový set S, jehož atributy patří do m různých tříd označené C_i, $(i = 1, \ldots, m)$. V závislosti na třídě můžeme tento datový set S rozdělit na m podmnožin S_i $(i = 1, \ldots, m)$, přičemž každá tato podmnožina patří do právě jedné třídy C_i a obsahuje s_i vzorků. Gini-index atributu X je pak rovný

$$G_{gini}(S) \quad = \quad 1 - \sum_{i=1}^{m} P(C_i|X)^2 \tag{1.25}$$

přičemž $P(C_i|X)$ značí pravděpodobnost, že atribut X patří zrovna do třídy C_i v případě výskytu atributu X.

Čím menší výsledek $G_{gini}(S)$ je, tím menší výsledná nečistota atributu, tedy lepší atribut. Minimální výsledek $G_{gini}(S) = 0$ značí, že všechny záznamy v kolekci patří do té samé třídy a tím pádem je atribut spojován právě s touto třídou (je „čistý“). Naopak pokud jsou vzorky uniformě rozděleny, výsledek Gini-indexu je maximální. Tím jsme o atributu X získali nejméně informací a považujeme ho za „nečistý“.

Selekce příznaků probíhá tak, že se každý atribut ohodnotí nezávisle na ostatních a seřadí se dle výsledného Gini-indexu. Vrchních k atributů s nejmenším indexem jsou následně vybrány ke klasifikaci.

Existuje i vylepšení této metriky *Improved Gini Index Shang et al.* pro textovou klasifikaci. Metoda měří kvalitu atributu X vzhledem k třídě C_i rozšířením i o opačnou podmíněnou pravděpodobnost.

$$G_{vylepšený\_gini}(S) \quad = \quad 1 - \sum_{i=1}^{m} P(X|C_i)^2 P(C_i|X)^2 \tag{1.26}$$

Přidaná hodnota $P(X|C_i)$ je pravděpodobnost, že atribut X je obsažen v třídě C_i a vynásobí se původní pravděpodobností, že atribut X patří zrovna do třídy C_i, v případě výskytu atributu X.

Výhodou této metody je, že zahrnuje veškerá data, zbavuje se redundantních dat a funguje dobře pro běžná data. Nevýhodou je, že tato metrika umožňuje modelovat pouze lineární závislosti.

1.3 Wrapper metody

Wrapperové metody přistupují k výběru příznaku jako kombinační problém. Různé příznaky jsou vybrány, vyzkoušeny na nějakém klasifikátoru a následně ohodnoceny a porovnány s ostatními kombinacemi.

Procesů hledání správný podmnožin může být více, může se jednat o metody **hrubé síly**, **dopředných** či **zpětných prohledávání** nebo za použití pokročilých **stochastických algoritmů**. Klasifikační model algoritmu je zde brán jako blackbox aparát, který má na vstupu různé kombinace atributů a provádí klasifikaci, která vrací ohodnocený výstup. Toto ohodnocení může být implementováno jako procentuální úspěšnost správnosti zařazení do tříd. Následně podmnožina atributů s největším získaným ohodnocením je vybrána jako finální podmnožina atributů. Na závěr je klasifikátor otestován testovacími daty, které nebyly použity k trénování výběru správných atributů a určí se výsledná přesnost.

Obrázek 1.4: Vizualizace wrapperových metod

Cílem klasifikátorů je, aby měly co možná největší přesnost (tedy nejmenší chybovost). Při trénování wrapper metodou je lepší nemaximalizovat přesnost klasifikátoru, ale odhalit takové atributy, které jsou zbytečné, či generují šum od těch, které jsou relevantní. Tyto dva přístupy se totiž mohou lišit.

Wrapperové metody můžeme rozdělit do kategorií v závislosti na způsobu výběru podmnožiny atributů, přičemž pracují s celou množinou atributů, jedná se tedy o multivarietní metody.

- Metody založené na hrubé síle
  - Hrubá síla
  - Hrubá síla s omezeným počtem atributů
  - Metoda větví a hranic
- Sekvenční selekce
  - Sekvenční dopředná selekce
  - Sekvenční zpětná eliminace
  - Zpětný chod při hledání

Paprskové prohledávání

- Stochastické prohledávání

  Genetický algoritmus

  Simulované ochlazování

  Metoda zakázaného prohledávání

  Optimalizace hejnem částic

Přičemž můžeme zde zařadit i veškeré filtrační metody, kde tvorba výsledné podmnožiny atributů se nevytváří konstantními funkcemi (počet, poměr). Samotné ohodnocení atributů má zde význam jakési heuristiky pro výběr správných atributů wrapperovým způsobem (hladové přidávání nejvýznamnějších atributů), které je však jednodušší a rychlejší než většina čistě wrapperových metod.

- Hladový dopředný výběr
- Hladový zpětný výběr
- Inflexní výběr

1.3.1 Metody založené na hrubé síle

Metody založené na hrubé síly vybírají ty nejlepší atributy s množiny a nikdy neuváznou v lokálním optimu. Vzhledem ale k existenci množství různých metod, řešící selekci alternativně můžeme usoudit, že hrubou sílu lze aplikovat pouze na menší datové sety a musíme počítat s větší časovou dotací, než je kolikrát nutné.

1.3.1.1 Hrubá síla

Hrubá síla je výpočetně náročná a zkouší postupně všechny možné kombinace atributů, což například u klasického datového souboru Kosatců (*Iris*) není problém, ale s přibývajícím počtem atributů roste časová složitost exponenciálně. Pokud máme n atributů, musíme projít $2^n - 1$ řešení, než můžeme jedno z nich prohlásit za optimální.

Tímto způsobem je i pro střední počet atributů téměř nemožné, nebo přinejmenším velmi zdlouhavé najít řešení v reálném čase. V této práci se touto metodou proto příliš zabývat nebudeme, použijeme ji pouze k srovnání u menších datových souborů.

1.3.1.2 Hrubá síla s omezeným počtem atributů

Pokud víme, či se pouze spokojíme s myšlenkou, že hledaných atributů je d, není nutné procházet celý stavový prostor, ale pouze těmi stavy, které mají d atributů. To je jenom $\binom{n}{d}$ podmnožin, což je ale stejně příliš mnoho.

1.3.1.3 Metoda větví a hranic

Ačkoli prohledávat všechny možné kombinace atributů často není možné, metoda větví a hranic (*Branch and Bound*) umožní prohledávat pouze část stavového prostoru a přesto poskytne exaktní řešení. Podmínkou je, aby vyhodnocovací funkce (ohodnocení klasifikátoru) byla monotonní, tedy aby přidání atributu nikdy nezpůsobilo zhoršení úspěšnosti klasifikace a dále také přesný počet atributů d, což je požadovaná velikost výsledného datového setu.

Strategie je založená na faktu, že pokud je vybráno více jak d atributů z dat, tak víme díky monotónnosti vyhodnocení, že žádná jeho podmnožina není lepší, než tato současná. Tedy není nutné jej vůbec procházet, pokud ovšem nalezená podmnožina není zrovna nejlepší doposud nalezená. V tomto případě algoritmus pokračuje ve vyhodnocování dále.

Časová složitost algoritmu je v nejhorším případě stále exponenciální, což je příliš mnoho pro větší počet atributů. Použití algoritmu je také závislé na předem známému počtu atributů d, kterou má mít výsledná množina příznaků a také na monotónnosti vyhodnocovací funkce, což nemusí být často splněno. Proto není použití této metody pro selekci příznaků příliš vhodné.

1.3.2 Sekvenční selekce

Kvůli problémům s hledáním optimálních strategií selekce příznaků se vymýšlely další možnosti, jak vybrat přijatelně dobré atributy, ale zároveň je neprocházet všechny, jako je tomu u hrubé síly. Mezi základní a snadno implementovatelné patří **sekvenční selekce**, které nabízejí i nějaká rozšíření a alternativy.

1.3.2.1 Sekvenční dopředná selekce

Výběr příznaků pomocí Sekvenční dopředné selekce (*SDS*) je jedna ze základních wrapperových metod, která funguje na principu nabalování atributů, bez možnosti zpětného kroku.

Při inicializaci je množina výstupních atributů prázdná (rozdíl oproti zpětné eliminační selekci) a cílem je přidávat do této množiny postupně atributy, dokud přesnost klasifikátoru roste. Ukončovací kritérium nastane, pokud všechny možné množiny atributů F' o velikosti m nezlepšují výkonnost klasifikátoru nad množinou atributů F o velikosti $m-1$, přičemž $F' \in F$.

Výstupem je tedy množina F vybraných suboptimálních atributů o velikosti m, kde $m \leq n$ a n značí celkový počet všech atributů.

Nabalováním atributů je myšleno spojení současného nejlepšího řešení se všemi zbývajícími atributy v kombinaci bez opakování, tedy těch, které ještě nebyly vybrány z nějakých

předchozích kroků. Následně je vybrán a přidán do řešení atribut, který nejlépe vylepšil přesnost klasifikátoru. Pokud žádný takový atribut není, algoritmus končí. Celý algoritmus je naznačen v pseudokódu 1.5, kde S je množina všech atributů daného datové setu, F je výstupní podmnožina atributů s nejlepšími výsledky.

Algorithm 1: Dopředná selekce

```
1:  F = ∅
2:  for feature index i in S do
3:    if size(F) ≠ i then
4:      return F
5:    end if
6:    F'_optimal = ∅
7:    R_max = 0
8:    for S_i in S and not in F do
9:      F' = F + S_i
10:     R_i = Eval(F', target)
11:     if R_i > R_max then
12:       R_max = R_i
13:       F'_optimal = F'
14:     end if
15:   end for
16:   F = F'_optimal
17: end for
18: return F
```

Algorithm 2: Zpětná eliminace

```
1:  F = S
2:  for feature index i in S do
3:    if size(F)+i ≠ size(S) then
4:      return F
5:    end if
6:    F'_optimal = ∅
7:    R_max = 0
8:    for S_i in S and in F do
9:      F' = F − S_i
10:     R_i = Eval(F', target)
11:     if R_i > R_max then
12:       R_max = R_i
13:       F'_optimal = F'
14:     end if
15:   end for
16:   F = F'_optimal
17: end for
18: return F
```

Obrázek 1.5: Pseudokód sekvenčních algoritmů

1.3.2.2 Sekvenční zpětná eliminace

Sekvenční zpětná eliminace, jak název napovídá, je metoda postupující z opačného konce jak SDS. Metoda začíná s kompletní množinou atributů a každou iterací se z ní odebere ten nejméně vhodný. Tímto způsobem se pokračuje do té doby, dokud množina obsahuje alespoň jeden atribut, či pokud odebráním atributu již nedojde k zlepšení modelu.

Předpokládáme také, že čím méně atributů, tím lépe. Pokud tedy model stagnuje, přikloníme se k řešení s méně atributy. Fungování algoritmu je znázorněno také na pseudokódu 1.5. Doba výpočtu je rozdílná pro obě metody a SDS je obvykle vykonána rychleji než SZE, neboť na začátku, kdy je možných spousta kombinací bez opakování, se větvě stavového prostoru rozpínají až do šířky n. A zatímco u SDS metoda pracuje rychle, neboť na začátku vyhodnocuje malý počet atributů, SZE má v této části výpočtu datový set nejrozsáhlejší. Ke konci se situace však obrací, ale to již většinou nezbývá příliš možností k vyhodnocení a větve stavového prostoru jsou ořezány.

Naproti tomu SDS nedokáže podchytit atributy, které jsou užitečné pouze v kombinaci s jinými. SDS vyhodnocuje pouze atributy zařazené do množiny a pokud atribut přináší benefity modelu pouze se zapojením jiného, dalšího atributu, je velmi obtížné pro SDS tuto vazbu podchytit. SZE tento úkol zvládá, neboť postupnou eliminací se daří tyto vazby zachovat.

1.3.2.3 Zpětný chod při hledání

Jedná se o rozšíření SDS a SZE s možností vrácení se zpět ve stavovém prostoru řešení. Velkou nevýhodou SDS a SZE je, že pokud přidáme (odebereme) atribut již na začátku hledání, zůstane (nezahrne se) ve výsledném setu až do konce a nemůže být v průběhu odstraněn (přidán). Tomuto problému se říká *nesting effect*. Špatné rozhodnutí na počátku výběru nemůže být později opraveno, neboť vybraná podmnožina atributů je vždy součástí podmnožiny následující. Algoritmus **Plus ℓ-take away r** ($PTA(\ell, r)$) řeší tento problém kombinací těchto přístupů.

Každá iterace algoritmu je rozdělena na dva kroky, přičemž první krok odpovídá běhu SDS. Dochází k přidání ℓ atributů dle jejich výkonnosti. Následuje druhý krok, kdy se po vzoru SZE odebere r atributů z množiny, která obsahuje již vybrané atributy. Pokud je počet atributů na přidání větší než na odebrání ($\ell > r$), počáteční množina atributů je prázdná. V opačném případě je množina atributů v prvním kroku plná (obsahující celá data) a $PTA(\ell, r)$ začíná až v druhém kroku - odebíráním atributů. $PTA(\ell, r)$ končí, pokud přidáním ani odebráním se výkonost modelu nezlepší.

1.3.2.4 Paprskové prohledávání

Paprskové prohledávání (*Beam search*) je algoritmus, co si uchovává v paměti seznam nejvíce zajímavých větví, které ještě nebyly prozkoumané, ale bylo by škoda je zahodit. V závislosti na velikosti proměnné, která udržuje velikost listu, můžeme s algoritmem manipulovat směrem od hrubé síly až k SDS v případě velikosti rovné jedné.

Uchovávání seznamu umožní algoritmu zpracovat stavy v podprostoru, které by v čistě sekvenčních verzích nebyly analyzovány, jelikož v počátku nebyl odhalen celý jejich potenciál.

1.3.3 Stochastické prohledávání

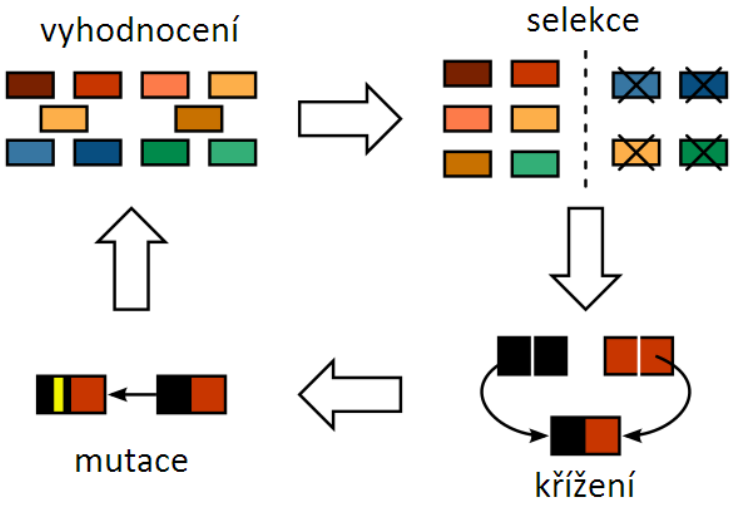
Stochastické prohledávání nabízí oproti ostatní algoritmům prvek náhody, tedy že za daných vstupů může algoritmus vrátit odlišné výstupy, atributy. Tento výstup však může být velmi citlivý na vstupní datový soubor a malá změna v něm může způsobit velmi odlišný výstup. Právě náhodnost u těchto algoritmů může vést v průměru k lepším výsledkům ve výběru atributů za cenu náročnější konfigurace a delšího výpočetního času.

1.3.3.1 Genetický algoritmus

Výběr vhodné podmnožiny atributů může být realizován i Genetickým algoritmem. Genetický algoritmus je algoritmus inspirovaný přírodou a směřuje do globálního či lokálního optima operacemi *selekce, křížení, mutace a vyhodnocení*. Koloběh střídající tyto operace v každé generaci je naznačený na obrázku 1.6.

Doba výpočtu souvisí s počtem generací, po které je řešení hledáno, což je jeden z parametrů, které algoritmu dodáváme. Při malém počtu generací není schopen algoritmus konvergovat k lepšímu řešení, naproti tomu nastavení příliš velké hodnoty způsobí nárůst výpočetního času bez nárůstu výkonu.

Ukončením poslední generace algoritmus končí a výstupem je nejlepší jedinec, který je průběžně ukládán na základě toho, jak dobré řešení reprezentuje. Ukončovací podmínkou může však být i jiné kritérium (požadovaná výkonnost) a výpočet může skončit dříve.



Obrázek 1.6: Diagram genetického algoritmu [22]

Inicializace
Inicializace spočívá ve vytvoření většinou náhodných jedinců, jejichž počet je jeden z nastavitelných parametrů. Každý jedinec představuje řešení daného problému, zakódovaný takovým způsobem, aby bylo možné jednoduše vypočítat jeho zdatnost. V kontextu vyhledávání vhodných atributů každý jedinec představuje vektor binárních hodnot (0, 1) o stejné velikosti jako je počet příznaků v datovém setu. Pokud vektor na pozici i obsahuje 1, to odpovídá vybranému příznaku, naopak 0 značí absenci příznaku.

Fitness
Po inicializaci je nutné dané jedince ohodnotit dle toho, jak kvalitní řešení nesou. Hodno-

tící (fitness) funkce ohodnotí dané řešení na základě výkonnosti modelu, který je trénovaný na podmnožině atributů doporučených od daného jedince. Výkonost modelu můžeme změřit pomocí střední kvadratické chyby (RMSE) nebo jeho procentuální úspěšností. Jedinec s vyšší hodnotou fitness má větší šanci setrvat v populaci i v následující generaci ve stádiu selekce. Fitness funkce se počítá opakovaně pro každého jedince, proto její výpočet musí být dostatečně rychlý. Je tedy důležité vybrat klasifikátor, který pracuje co nejrychleji.

Selekce
Selekce může být implementována několik způsoby, nejčastější je *Roulette Wheel Selection (RWS)*, která vybírá jedince z populace na základě jejich relativní fitness. Pravděpodobnost $P(i)$ vybrání jedince i se vypočítá

$$P(i) = \frac{f_i}{\sum_{j=1}^{n} f_j} \tag{1.27}$$

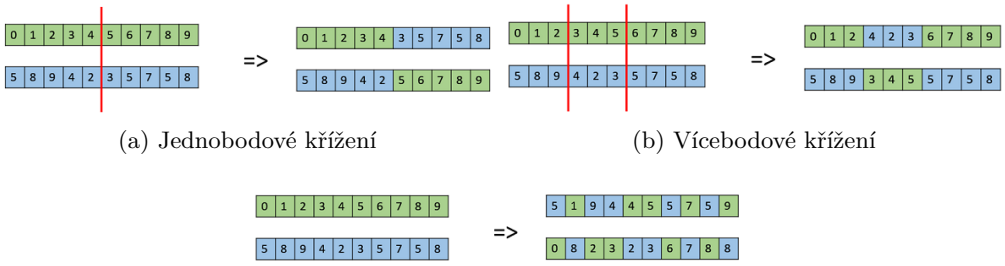
kde f_i je fitness jedince i a n je velikost populace. Problém tohoto přístupu může nastat v situaci, kdy se jednotlivá fitness jedinců hodně liší, šance ostatních (méně zdatných) jedinců na vybrání je pak velmi malá.

Zde se hodí použít *Rank Selection (RS)*. RS jedince nejprve ohodnotí dle jejich pořadí, nejhorší jedinec bude mít hodnocení 1, druhý nejhorší 2, ..., a nejlepší n. Jednotlivá ohodnocení představují nové fitness hodnoty, nad kterými se provede selekce ruletou. Zde se ovšem jednotlivé fitness liší příliš málo a konvergence populace je malá.

Další možností je *Stochastin Universal Sampling (SUS)*, což je obdoba RWS, která vybírá jedince jediným otočením rulety. Od prvního jedince vybraného ruletou vybere dalších n jedinců, vzdálených o stejnou délku. Tím se zaručí, že jedinec, co zabírá 4,5% ruletového prostoru, bude vybrán přesně 3-4 krát.

Mezi hojně používané selekční metody patří i *Tournament selection (TS)*. Postup výběru začíná vybíráním náhodných k jedinců a nejlepší z nich se vybere jako rodič. Tento proces se opakuje n krát. Parametrem zde je hodnota k (velikost turnaje). Pokud je rovna jedné, jedná se o náhodný výběr.

Křížení
Ve fázi křížení probíhá vytváření nových potomků z rodičovských chromozomu (vybraných selekcí) se snahou o vylepšení stávajících řešení a ideálně konvergovat ke globálnímu optimu. Pravděpodobnost, zda rodič vstoupí do této fáze je dáno nastavením parametru pravděpodobnosti křížení. Samotné křížení může být *jednobodové* - vygeneruje se náhodný bod rozdělení chromozomu rodičů a potomci jsou vytvoření kombinací těchto částí. Pokud je těchto vygenerovaných bodů rozdělení více, jedná se o *vícebodové* křížení. Potomky můžeme také vytvářet *uniformním* křížením, kdy potomek má přibližně 50% genu od prvního rodiče a 50% od druhého, nicméně tyto geny jsou vybírány náhodně.



(a) Jednobodové křížení

(b) Vícebodové křížení

(c) Uniformní křížení

Obrázek 1.7: Křížení chromozomů [42]

Mutace
Mutace je genetický operátor pro zachování diverzity populace. Mutaci si můžeme představit buď jako inverzi náhodného bitu, výměnu náhodného bitu v chromozomu, zašumění bitů, náhodné zresetování bitu a další.

Pravděpodobnost mutace je opět nastavitelný parametr, je nutné ale brát na ohled, že s příliš velkou pravděpodobností nebude populace schopná konvergovat, naopak s malým výskytem mutace nebude populace různorodá.

1.3.3.2 Simulované ochlazování

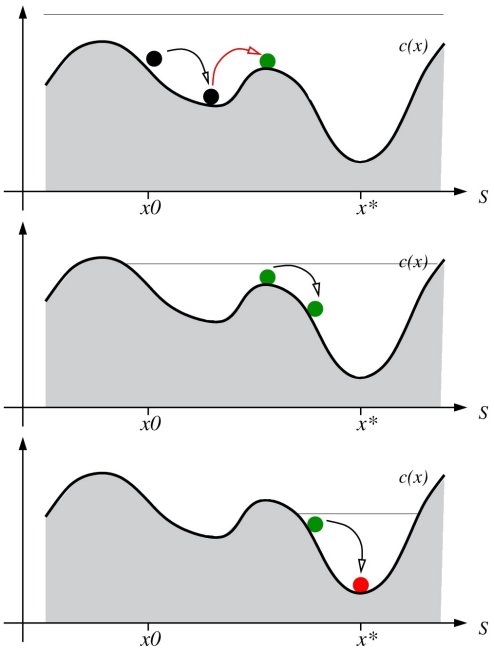
Simulované ochlazování (*Simulated annealing*) je lokální metoda prohledávání stavového prostoru, inspirovaná technikou z metalurgie, kdy dochází k pozvolnému ochlazování kovu, čímž se zlepšují fyzikální parametry materiálu.

Inicializace
Hledání atributů začíná s inicializační (náhodnou, plnou) podmnožinou atributů s vysokou počáteční teplotou. Z počáteční množiny se v dalších krocích provedou menší změny ve vybraných atributech a od klasifikátoru dostane zpětnou vazbu v podobě skóre nebo kvadratické odchylky. Tento proces reprezentuje cenovou funkci, kdy ohodnocení stavu $E(s)$ nám dá výslednou cenu řešení c. Na základě toho se rozhodneme, zda je tento výběr lepší, čí horší než předchozí a rozhodneme, zda nový stav přijmeme či nikoliv.

Přijmutí dalšího stavu
Pokud je nalezený stav výhodnější než předchozí, řešení je přijato (stavy s lepším řešením jsou vždy přijaty). Ovšem pokud je nalezený stav horší, řešení může být také přijato, a to s pravděpodobností, která se nazývá „acceptace probabilty“.

Tato pravděpodobnost se značí $P(c, c', T)$ a je zásadní změnou oproti ostatním algoritmům, neboť povolením zhoršení výsledku se zmenší šance na uváznutí v lokálním maximu nebo minimu a zvýší se jeho exaktnost. Naproti tomu s příliš velkou ochotou ke zhoršení by al-

goritmus nedokázal konvergovat. Je to tedy parametr závislý nejen na kvalitě současného řešení (c), kvalitě nalezeného řešení (c'), ale i teplotě (T), což je znázorněno na 1.8. Čím větší je teplota, tím spíše je algoritmus ochotný spokojit se s horším řešením. S postupem času dochází k ochlazování a řešení by v ideálním případě mělo konvergovat ke globálnímu optimu.



Obrázek 1.8: Konvergence simulovaného ochlazování[44]

Nastavitelné parametry

Tato metoda disponuje větším množstvím volitelných parametrů, jejichž správné nastavení má velký vliv na výsledek. Kromě počáteční teploty a konečné teploty je to i koeficient ochlazování, který určuje rychlost konvergence. Další neznámou je počet iterací, což představuje opakování tohoto procesu. Pseudokód této lokální metody je znázorněn na 3, kde metoda $NewS(s, T)$ vyhodnocuje kvalitu řešení a také určuje, zda bude přijato či nikoliv.

1: $T = InitT()$
2: $s = InitS()$
3: **while** $T > T_{min}$ **do**
4: $\quad InitStep()$
5: $\quad$ **while** $iter < iter_{max}$ **do**
6: $\quad\quad s = NewS(s, T)$
7: $\quad\quad s_{best} = SetBest(s_{best}, s)$
8: $\quad$ **end while**

9: $CoolDown(T)$
10: **end while**
11: **return** s_{best}

Algorithm 3: Pseudokód simulovaného ochlazování

1.3.3.3 Metoda zakázaného prohledávání

Dalším zástupcem stochastických algoritmů je metoda zakázaného prohledávání (*Tabu search*), kterou lze použít i pro hledání vhodných atributů.

Metoda je založená na *horolezeckém algoritmu (hill climbing)*, jedná se tedy o variantu gradientní metody, protože směr nejprudšího spádu se určí prohledáváním okolí. Obdobně jako u jiných gradientních algoritmů, i zakázané prohledávání často končí v lokálním minimu a nedosáhne minima globálního.

Okolí řešení
Získání nového řešení začíná vytvořením okolí stávajícího řešení pomocí určitých transformací. Počáteční řešení může být náhodné a symbolizuje atributy, která mají být použity pro klasifikaci. Pomocí transformací vznikne okolí, ze kterého se vybere nejlepší řešení - lokální minimum. Toto řešení je využito v následující iteraci opět jako střed při vytvoření nového okolí. Počet těchto iterací je nastavitelný parametr a při skončení se vybere řešení, které bylo zaznamenáno jako nejlepší v průběhu běhu algoritmu. Kvalitu řešení určuje klasifikátor pomocí metody nejmenších čtverců či procentuální úspěšnosti. Aby nedošlo k zacyklení, pustí se algoritmus vícekrát pokaždé s jinými počátečními stavy a výsledkem je právě řešení dosahující nejlepších výsledků.

Krátkodobá paměť
Zavedením krátkodobé paměti do algoritmu umožníme algoritmu pamatovat si inverzní transformace k lokálně optimálním transformacím řešením, které vygenerovaly příslušná okolí a byly použity k získání nových středů. Tyto inverzní transformace jsou pak zakázány (tabu) při hledání nového okolí bodu a tím se omezí prostor řešení, které je potřeba vyhodnocovat. Také se tímto způsobem daří omezit zacyklení při pádu do lokálního minima.

Krátkodobá paměť je realizována zásobníkovým *zakázaným seznamem TL (tabu list)*, který má velikost k_{max} a při inicializaci algoritmu je prázdný. Při každé další iteraci se do něho vloží inverzní transformace k transformaci, která poskytla nejlepší řešení a zamezí se její použití pro lokální minimalizaci v rámci okolí aktuálního řešení. Logicky, zakázaný seznam musí být kratší, než je počet možných transformací generující okolí, aby bylo možné řešení směrovat nějakým směrem. Po k_{max} iteracích je seznam plný a každé další přidání musí být doprovázeno vyhozením té nejstarší transformace ze seznamu. Tomu se říká *cyklické obnovování seznamu.*

Velikost zakázaného seznamu
Velikost parametru k_{max} je důležitým parametrem ovlivňující možnost vymanit se z lokálních minim. Pokud je hodnota příliš malá, může dojít k zacyklení algoritmu, obdobně jako u horolezeckého algoritmu. Naopak při příliš vysoké hodnotě k_{max} dojde s velkou pravděpodobností k přeskočení nadějných podmnožin atributů.

1: $s = RandomFeatures()$
2: $TL = \emptyset$
3: $f_{min} = \infty$
4: $iter = 0$
5: **while** $iter < iter_{max}$ **do**
6: $iter = iter + 1$
7: $S = Neighborhood(s)$
8: $f_{loc-min} = \infty$
9: **for** $solution \in S$ **do**
10: **if** $f(solution) < f_{loc-min}$ and ($f(solution) < f_{min}$ or $solution \notin TL$) **then**
11: $f_{loc-min} = f(solution)$
12: $solution_{best} = solution$
13: **end if**
14: **end for**
15: $s = solution_{best}$
16: **if** $f_{loc-min} < f_{min}$ **then**
17: $f_{min} = f_{loc-min}$
18: **end if**
19: **if** $|TL| < k_{max}$ **then**
20: $TL = TL + solution_{best}$
21: **else**
22: $TL = (TL - solution_{oldest}) + solution_{best}$
23: **end if**
24: **end while**
25: **return** f_{min}

Algorithm 4: Pseudokód zakázaného prohledávání

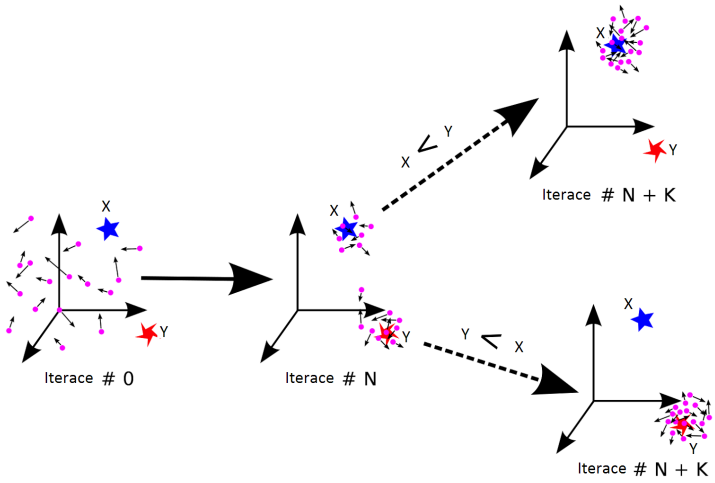
Aspirační kriterium
Zakázaný seznam se používá k vytvoření okolí aktuálního bodu. Součástí okolí již nejsou řešení, která vznikla transformacemi z TL. Lokální minimalizace se vykonává tedy pouze z tohoto modifikovaného okolí, s výjimkou aspiračního kriteria. Aspirační kriterium dovoluje použití zakázané transformace v případě, že řešení bylo lepší, než je dočasné nejlepší řešení. Fungování celého algoritmu je znázorněno na pseudokódu 4.

1.3.3.4 Optimalizace hejnem částic

Optimalizace hejnem částic (*Particle swarm optimalization*) je metoda stochastické optimalizace simulující sociální chování organismů (stěhování ptactva či migrací ryb) pro automatický evoluční systém.

Každý částice (organismus) reprezentuje jedno řešení (kombinaci atributů) v hledaném prostoru. Přitom každá z nich využívá svojí paměť a také vědomosti získané hejnem jako celkem pro získání nejlepšího řešení. Všechny částice mají fitness hodnotu, která udává zdatnost jejich řešení. Tato hodnota se získá z fitness funkce a cílem je nalézt řešení, které je co nejlépe ohodnocené touto funkcí.

Další jejich vlastností je rychlost, která určuje pohyb v prostoru řešení. Tento pohyb je ovlivněn jak vlastní zkušeností částice, tak zkušeností sousedících částic. Následně částice upravují svojí pozici na základě těchto informací. Průběh hledání optima je znázorněno na obrázku 1.9, kde každou iteraci se částice posouvají podle jejich vypočítaného pohybu, až konvergují k nějakým řešením. Proces hledání řešení začíná inicializací, pokračuje aktualizací jednotlivých částic a nakonec dojde k ukončení.



Obrázek 1.9: Migrace částic do dvou lokálních minim [33]

Inicializace
Počáteční hejno je obvykle rozdistribuované náhodně po celém prohledávaném prostoru. Velikost hejna je nastavitelný parametr a algoritmus pokračuje dále jejich aktualizací.

Aktualizace
Při každé iteraci se každá částice aktualizuje podle dvou „nejlepších“ hodnot *pbest* a *gbest*. Hodnota *pbest* představuje nejlepší koordinátory v prostoru, které si uchovává do vlastní paměti, a dosahovaly v nich nejlepších řešení (na základě fitness). Oproti tomu *gbest* před-

stavuje nejlepší hodnotu v celém okolí.

Rychlost částice se upraví za použití právě těchto maxim, ale také nastavitelných akceleračních faktorů c_1 a c_2 (obvykle rovné dvou) a váhy w. Omezení rychlosti částic pro každou dimenzi se upravuje použitím parametrů V_{min} a V_{max}. Nová pozice se následně vypočítá součtem současné pozice a rychlosti.

Ukončení
Algoritmus končí po proběhnutí určitého počtu iterací, či pokud je splněné ukončovací kritérium [7], což explicitně definujeme hranicí požadované vlastnosti.

- Dosažená fitness je dostačující
- Výpočetní čas je nad limitem
- Míra zlepšení hejna je nízká
- Pohyb hejna je malý
- Vzdálenost (průměrná, procentuální, odchylka) všech částic je pod požadovanou hranicí

Celý pseudokód je pak naznačen v algoritmu 5, obsahující kromě proměnných popsaných výše i definovanou fitness funkci $f()$, inicializační funkci $InitSwarm()$, která rozmístí částice po prostoru a generátorem náhodných čísel $Rand()$.

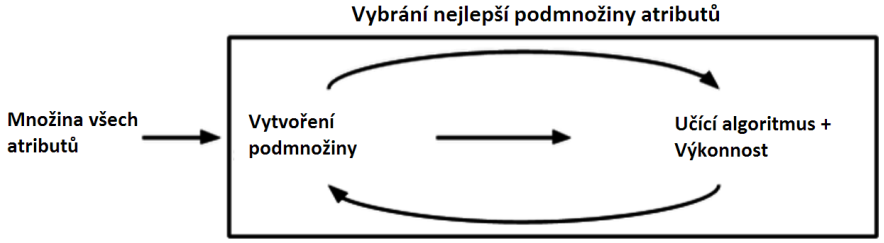
1: $S = InitSwarm()$
2: **while** $iter < iter_{max}$ or stopping criterion **do**
3: $iter = iter + 1$
4: **for** $p \in S$ **do**
5: **if** $f(p) > f(pbest_p)$ **then**
6: $pbest_p = p$
7: **end if**
8: **for** $n \in Neighbors(p)$ **do**
9: **if** $f(n) > f(gbest)$ **then**
10: $gbest = n$
11: **end if**
12: **end for**
13: **for** $d \in Dimensions$ **do**
14: $v_{p_d} = w \cdot v_{p_d} + c_1 \cdot Rand(0,1) \cdot (pbest_{p_d} - p_d) + c_2 \cdot Rand(0,1) \cdot (gbest_d - p_d)$
15: **if** $v_{p_d} \notin (V_{min}, V_{max})$ **then**
16: $v_{p_d} = max(min(V_{max}, v_{p_d}), V_{min})$
17: **end if**
18: $p_d = p_d + v_{p_d}$
19: **end for**

20: **end for**
21: **end while**
22: **return** *gbest*

Algorithm 5: Pseudokód optimalizace hejnem částic

1.4 Embedded metody

Embedded metody jsou takové způsoby selekce atributů, kde se provádí samotná selekce jako součástí učícího procesu. Filtrační metody na rozdíl od embedded nezahrnují do selekce proces učení, neboť vyhodnocují kvalitu atributu ještě před samotným trénováním, nezávisle na trénovacím algoritmu. Wrapper metody používají prediktor na zjištění kvality vybrané podmnožiny atributů, ale nemají přímou vazbu na konkrétní klasifikátor - nezahrnou do výsledku znalost konkrétní struktury a klasifikační funkce.



Obrázek 1.10: Vizualizace embedded metod

Embedded metody přistupují k problematice odlišně. Neodděluje se selekce se samotným způsobem učení, nýbrž se vybírají příznaky, které se váží ke konkrétnímu klasifikačnímu či regresnímu algoritmu. Tedy příznaky, které byly vyhodnoceny jako důležité pro SVM a klasifikace na základě těchto atributů dosahuje dobrých výsledků, nemusí fungovat dostatečně například pro nejbližší sousedy, či jiné klasifikační algoritmy, které se navzájem více odlišují.

Mějme množinu všech atributů označenou jako vektor S, nabývající hodnot 1 či 0 v závislosti na tom, zda je vybrán do klasifikačního setu či nikoli. Tedy $S = \{0, 1\}^n$, kde n je počet všech atributů. Klasifikační a regresní funkce f je definována jako $R(\vec{\alpha}, x) \mapsto f(\vec{\alpha}, x)$ s $\vec{\alpha}$ definovanou jako konkrétní nastavení jednotlivých parametrů klasifikátoru za účelem minimalizace chybné klasifikace v případě klasifikátorů a chybné predikce v případě prediktorů. U SVM tímto vektorem $\vec{\alpha}$ myslíme zakódované váhové vektory $\vec{w}$ spolu s proměnnou b udávající přesnou polohu hyperplane.

Cílem je tedy nalézt takovou podmnožinu S, kde bude minimální chyba při $\vec{\alpha}$. Jedná se tedy o minimalizační problém. Pro trénovací data (X, Y), ztrátovou funkci L a funkci měření vlivu na trénovacích datech P mějme:

$$R(\vec{\alpha}, x) \quad = \quad \int L[f(\vec{\alpha}, Sx), y] dP(x, y) \tag{1.28}$$

Přičemž součin Sx není definován standardním maticovým násobením, ale komutativním Hadamardovým násobením.

Postup k aproximativní minimalizaci vzorce může být formulován následovně

$$\vec{\alpha} = \widetilde{T}(S, X, Y) \tag{1.29}$$

$$S = \min_{S \in \{0,1\}^n} \tag{1.30}$$

$$R(\vec{\alpha}, x) = G(\vec{\alpha}, \widetilde{T}, S, X, Y) \tag{1.31}$$

Pokud si tento vzorec rozebereme, tak funkce G představuje prvek, jenž měří výkonnost a přesnost klasifikátoru. Výsledkem samozřejmě chceme co nejpřesnější klasifikátor. Na rozdíl od wrapperových metod, funkce G je závislá na konkrétním klasifikátoru označeným jako $\widetilde{T}$, který naopak dává funkci G zpětnou vazbu a tím řídí její výpočet. Díky této vlastnosti je umožněno brát si a zpracovávat informace o konkrétním klasifikátoru $\widetilde{T}$ (jeden z parametrů G) a jeho struktuře.

Klasifikátor (prediktor) $\widetilde{T}$ může být jakýkoliv klasifikační respektive regresní algoritmus, který zpracovává trénovací data X, Y za použití atributů S. Výstupem je vektor $\vec{\alpha}$, což představuje nějaké konkrétní nastavení daného algoritmu.

Samozřejmě chceme využít co nejméně atributů, což povede k rychlejšímu, přehlednějšímu a často i efektivnějšímu výpočtu, hledáme tedy S minimální za cenu stejné výkonnosti klasifikátoru. Metody tohoto typu (vyjádřitelné tímto vzorcem) nazýváme embedded.

Existuje mnoho variant embedded metod a můžeme je zařadit do dvou hlavních skupin.

1. Metody přidávající či ubírající atributy iterativně k hladové aproximaci minimalizačního problému
2. Regularizační metody řešící selekci příznaků jako optimalizační problém

Z hlediska rozděleni na multivarietní a univarietní vypadá rozdělení následovně.

- Univarietní

  Rozhodovací strom (ID3)

- Mutlivarietní

  Rozhodovací strom (CART)

  Grafting

  Gramova-Schmidtova ortogonalizace

  Rekurzivní eliminace atributů

  Konkávní minimalizace

  Multiplicative Update

  ℓ_1 Support Vector Machine

  LASSO

1.4.1 Forward-Backward Metody

Metody patřící do této kategorie iterativně přidávají nebo ubírají atributy z datového souboru. Jednou z možností jsou takzvané **forward selection** metody, které začínají s jedním či malým počtem atributů a postupně se přidávají další dokud není výsledek uspokojivý - ukončovací kritérium. Tyto metody jsou svojí formou odlišné od wrapperových přímým ovlivňováním klasifikátoru.

Dalším způsobem je **zpětná eliminace**, kde je v inicializaci přítomný plný počet atributů. Jak název vypovídá, postupně se z dat ubírají jednotlivé atributy dle optimalizačních kriterií.

Poslední možností jsou **zahnízděné metody**, což je kombinace předešlých dvou. Atributy smí být v průběhu iteraci přidávány i odebírány.

1.4.1.1 Dopředná selekce

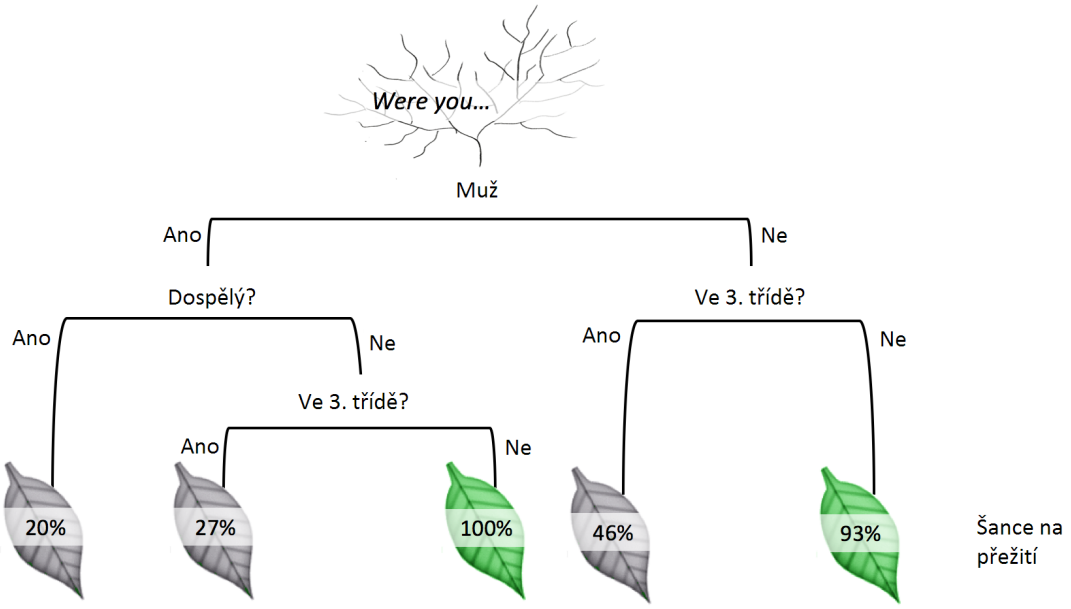
V této sekci se budeme zabývat metodami dopředných selekcí, jako jsou **Rozhodovací stromy**, **Gram-Schmidt ortogonalizace** a **Grafting metoda**.

Rozhodovací stromy
Rozhodovací stromy se konstruují iterativně a větví se na základě různých atributů dat. V případě přežití na Titanicu by rozhodovací strom mohl vypadat takto 1.11. Tyto atributy, které rozdělují data do jednotlivých tříd, jsou vybrány na základě důležitosti. Jednou možností, jakým způsobem přiřadit atributům míru důležitosti, je informační zisk mezi atributem a výstupem, který je popsán v kategorii filtračních metod. Také je možné data rozdělit na základě minimalizace empirické chyby.

Ve většině případu je zapotřebí pouze část atributů na popsání zdrojových dat, naopak zachování většího množství může zvýšit procento neúspěšné klasifikace. Proto algoritmy konstruující rozhodovací stromy mají selekci příznaků zabudovanou v sobě a není potřeba data předzpracovávat.

Riziko rozhodovacích stromů tkví v jejich přeučení, neboť větší počet trénovacích dat upraví strukturu stromu tím, že zvýší jeho komplexnost a dochází k zvýšení počtu rozhodovací podmínek. Strom pak narůstá do velkých rozměrů a jeho klasifikační schopnost mimo trénovací data není velká. Tento vliv se snaží z větší části redukovat algoritmus **Náhodný les**, což je pouze množina rozhodovacích stromů, avšak finální zařazení do třídy se určuje na základě majority výsledků v jednotlivých dílčích stromech.

Na sestavení stromu existuje více přístupů. Jendou možností je *ID3 (Iterative Dichotomiser 3)*, což je algoritmus vynalezený Rossem Quinlanem zveřejněný v knize Machine learning v roce 1975. Algoritmus ohodnocuje atributy metodou informačního zisku. Atribut s největší hodnotou informačního zisku X_i je použit jako kořenový uzel a rozdělí se na k_{X_i} větví, což



Obrázek 1.11: Rozhodovací strom znázorňující šanci na přežití na Titanicu [21]

je množství hodnot, kterých atribut X_i nabývá.

Následně pokračujeme zvolením atributu X_j s druhým největším informačním ziskem a data opět rozdělíme do k_{X_j} větví. Iterativně pokračujeme tak dlouho, dokud data neobsahují pouze jedinou třídu nebo nezbyl žádný atribut. Samotná klasifikační třída je implementovaná jako list stromu.

Model *CART (Classification and Regression Trees)* představený Leem Breimanem je binární strom, kde každý uzel reprezentuje jeden atribut a rozdělí data do dvou větví (nikoli do k_{A_i}). V praxi tedy uzel rozvětvíme například na základě formule „větší než“, namísto konkrétní hodnoty, jako je tomu u ID3. Listy stromu obsahují opět třídní zařazení.

Posledním modelem, který zmíníme je *C4.5*, což je softwarové rozšíření ID3 algoritmu navržené také Rossem Quinlanem, který řeší některé jeho nedostatky [40].

- Nastavení maximální hloubky k redukci přeučení
- Práce se spojitými atributy
- Ořezání stromu
- Výběr příhodného selekčního mechanismu
- Doplnění chybějících hodnot atributů

- Práce s atributy s rozdílnou váhou
- Zvýšení výpočetní efektivity

Grafting
Metoda, jejíž název je odvozen z Gradient Feature Tessting, navrhl v roce 2003 Perking et al. Předpokládejme, že chceme nalézt klasifikační funkci, která vstupní proměnou X převede na výstupní hodnotu reprezentující její třídu. Například pro binární klasifikaci, kde výstup nabývá hodnot $\{-1, 1\}$, je to $Y = sign(f(X))$. Funkce f představuje tedy určitý klasifikátor spadající do klasifikační či regresní rodiny, obsahující n (velikost testovacích dat) náhodných páru (X, Y) vybraných z dat.

Tento klasifikátor má parametry $\vec{\alpha}$, lze tedy konkrétně definovat klasifikátor z této rodiny jako $f_{\vec{\alpha}}$. Naším cílem je nalézt takové $\vec{\alpha}$, pro které bude riziko nesprávné klasifikace co nejmenší.

$$\begin{aligned} L(f_{\vec{\alpha}}(X), Y) &= \text{ztrátová funkce specifikující míru penalizace} \\ p(X, Y) &= \text{sdružená hustota pravděpodobnosti } X \text{ a } Y \\ R(\vec{\alpha}) &= \int L(f_{\vec{\alpha}}(X), Y) p(X, Y) dX dY \end{aligned} \tag{1.32}$$

Obecně hodnotu $p(X, Y)$ neznáme, není tedy možné přímo vypočítat 1.32.

Místo toho si však můžeme spočítat hodnotu C, což představuje empirické riziko vypočítané z množiny trénovacích dat. Jelikož trénovacích dat je spočetný počet (n), nahradíme v rovnici integrál za sumu přes všechny trénovací datové body. Vypočítání empirického rizika namísto celkového přináší problém s přeučením klasifikátoru na konkrétní množinu dat. Je zvykem v těchto případech minimalizovat jak na základě empirického rizika, tak přidat nějaký mechanismus, který penalizuje příliš složitá řešení signalizující přeučení modelu. Výsledný vzorec vypadá následovně:

$$\begin{aligned} \Omega(\vec{\alpha}) &= \text{regularizační funkce} \\ C(\vec{\alpha}) &= \frac{1}{n} \sum_{i=1}^{n} L(f_{\vec{\alpha}}(X_i), Y_i) + \Omega(\vec{\alpha}) \end{aligned} \tag{1.33}$$

Regularizační funkce nabývá vysokých hodnot pro komplexní klasifikátory $f_{\vec{\alpha}}$ a naopak. Pro lineárně separabilní třídy a velkou část vícevrstvých perceptronů má následující podobu.

$$\begin{aligned} \Omega_q(\vec{\alpha}) &= \lambda \sum_{i=1}^{p} \beta_i |\vec{\alpha}|^q \\ \vec{\alpha} &= \text{parametr klasifikátoru (například váhový vektor)} \\ p &= \text{velikost parametrů (váhy)} \\ \beta &= \text{vektor kladných čísel, obvykle } \beta \in \{0, 1\} \\ q &= \text{typ regulační funkce} \end{aligned} \tag{1.34}$$

Tuto funkci můžeme rozdělit do několika kategorii v závislosti na rozdílnosti typu Minkowského formy (parametr q), Ω_q se tedy často nazývá ℓ_g regularizátor, přičemž nejzajímavější regularizátory zahrnují $q \in \{0, 1, 2\}$.

Ω_2: Tento regularizátor se používá v ridge regresních modelech, SVM a regularizačních sítí. Na rozdíl od ostatních Ω_q generuje stejné řešení pod libovolnými rozpoložením os v prostoru atributů.

Ω_1: Nebo také ℓ_1 regularizátor, zvaný „lasso“. Jeho výhodou je, že často vede k řešení v případech, kde některý z elementů α je nulový.

Ω_0: Za předpokladu, že $0^0 = 0$, tak získáme fixní penalizaci β_i pro každý element $\alpha_i \neq 0$.

Za předpokladu, že k optimalizaci využijeme všechny tyto tři regulátory, vzorec se nám rozšíří

$$\begin{aligned} C(\alpha) &= \frac{1}{m}\sum_{k=1}^{m} L(f(x_i), y_i) + \\ &+ \lambda_2 \sum_{i=1}^{p} \beta_{2,i}|\alpha_i|^2 + \lambda_1 \sum_{i=1}^{p} \beta_{1,i}|\alpha_i|^1 + \lambda_0 \sum_{i=1}^{p} \beta_{0,i}|\alpha_i|^0 \end{aligned} \tag{1.35}$$

Minimalizaci funkce 1.35 je cílem algoritmu graftování. Předpokládejme, že:

- $\lambda_0 > 0$, $\lambda_1 > 0$ a $\lambda_2 > 0$.
- Klasifikační model je parametrizován vektorem $\vec{\alpha}$ a výstup má parciální derivaci (lze spočítat $\frac{\partial f(x_i)}{\partial \vec{\alpha_j}}$).

Pak se v každé iteraci rozšíří počet parametrů $\vec{\alpha_i}$ o jeden a provede se opět minimalizace nad tímto rozšířenou podmnožinou atributů. Selekční kriterium pro nové parametry odpovídá podílu parciálních derivací $\frac{\partial C}{\partial \vec{\alpha_i}}$. Vybere se tedy $\vec{\alpha_i}$ takové, aby byl sestup funkce C co největší v každém gradientním kroku. Jedná se o algoritmus hladového typu.

Gramova-Schmidtova Ortogonalizace
Gramova-Schmidtova Ortogonalizace využívá úhel, co svírá atribut i na trénovacích datech X se svým zařazením do třídy Y. Tento úhel se používá jako kritérium k vyhodnocení důležitosti atributu. Pro všechna i je úhel vypočítán následovně.

$$cos(X^i, Y) = \frac{\langle X^i, Y\rangle^2}{||X^i||^2||X^i||^2} \tag{1.36}$$

Tímto způsobem je úhel všech příznaků vzhledem ke třídě iterativně vypočítán a je vybrán takový příznak, pro kterou byl výsledek maximální. Zbylé příznaky, které nebyly vybrány se namapují do prostoru příznaků, které byly vybrány v předchozím kroku a běh pokračuje

další iterací.

Při každé iteraci jsou ohodnocené atributy využity k počítání řešení nejmenší čtverců lineárního modelu. Tato skutečnost metodu kategorizuje do embedded skupiny pro lineární prediktor nejmenších čtverců, podobné algoritmu *Částečných nejmenších čtverců (PLS).*

Ukončovací kritérium, představené Stoppiglia et al. (2003), přidává do množiny atributů náhodný atribut. Následně podle jejich ohodnocení se určí, zda je atribut irelevantní či nese nějakou informaci o třídě v závislosti právě na porovnání s tímto náhodným atributem.

Gramova-Schmidtova ortogonalizace se také využívá k aproximaci PCA (Principal Component Analysis) - APCA, které je jednodušeji interpretované než PCA vzhledem k tomu, že ortonormální báze jsou instance původního datového souboru.

1.4.1.2 Zpětná eliminace

Začínáme se všemi atributy, přičemž zpětná eliminace postupně jednotlivé atributy na základě selekčního kriteria odebírá, dokud není splněno ukončující kritérium.

Rekurzivní eliminace atributů
V originále *Recursive Feature Elimination (RFE)* algoritmus byl popsán v roce 2002 Guyon et al. Cílem této metody je najít co nejlepší podmnožinu atributů o velikosti $\sigma_0 < n$. Postup se tedy pokusí vybrat σ_0 atributů postupnou eliminací atributů na základě jeho váhy. Tento algoritmus je vytvořený pro SVM klasifikaci, výběr atributu je spjat s parametry tohoto klasifikátoru. Využívá se tedy v kombinaci pouze s tímto klasifikátorem. Postup výběru atributů je následující:

1. Najdi w a b trénováním lineárního SVM klasifikátoru
2. Odstraň atribut s nejmenší hodnotou $|w_i|$
3. Návrat k bodu 1, dokud počet atributů není σ_0

Při každé iteraci je odstraněn atribut, který způsobí nejmenší zmenšení margin (vzdálenost support vektorů od hyperplane) a tím minimalizuje riziko špatné klasifikace.

Další variantou RFE může být eliminace více atributů najednou, tedy bod 2 se změní na: *Odstraň k atributů s nejmenší hodnotou $|w_i|$ (často polovina atributů v případě tisíců atributů).*

Právě pro rozsáhlé datové soubory může mít tato úprava výkonnostní nárůst a doba selekce atributů se zkrátí. Je možné také upravit algoritmus pro nelineární SVM klasifikátor.

1.4.2 Regularizační metody

Regularizační či penalizační metody jsou hodně používané embedded metody, jejichž funkce tkví v přidávání dalších podmínek k optimalizaci prediktivních algoritmů pro snížení jejich komplexnosti.

Většina lineárních modelů může být bráno jako následující minimalizační problém:

$$min\frac{1}{n}\sum_{k=1}^{n} L(wx_k + b, y_k) + C\Omega(w) \tag{1.37}$$

kde $L(f(x_k), y_k)$ měří ztrátovou funkci $f(X)$ na trénovacích bodě (x_k, x_k). Příklady ztrátových funkcí jsou

- ℓ_1 ztráta

  Pro kladný výsledek: $\ell_1(wX + b, Y) = |1 - Y(wX + b)|$

  Jinak: $\ell_1(wX + b, Y) = 0$

- ℓ_2 ztráta

  $\ell_2(wX + b, Y) = (wX + b - Y^2)$

- Logistická ztráta

  $\ell_{Logistic}(wX + b, Y) = \log(1 + e^{-Y(wX+b)})$

Výraz $\Omega(w)$ představuje penalizační term, který může být dvou typů

- ℓ_0 penalizace

  $\Omega(w) = \ell_0(w)$, reprezentující nenulové souřadnice w.

- ℓ_1 penalizace

  $\Omega(w) = \sum_{i=1}^{n} |w_i|$

Tabulka 1.1: Embedded metody zajišťující selekci atributů během trénování lineárního modelu

Penalizace / Ztráta	ℓ_0	ℓ_1
ℓ_1	FSV	ℓ_1 SVM
ℓ_2	Multiplicative update	LASSO
$\ell_{Logistic}$	x	Generalized LASSO

1.4.2.1 ℓ_1 Support Vector Machine

ℓ_1 Support Vector Machine je jeden ze způsobů, jak získat řídká (sparse) řešení, tedy s malým počtem nenulových váhových koeficientů. Pro úlohu klasifikace je potřeba řešit následující optimalizační problém váhového vektoru $\vec{w}$, kolmého na hyperplane

$$min \sum_{i=1}^{n} |w_i| + C \sum_{k=1}^{m} \xi k \tag{1.38}$$

za podmínky $\xi k \leq 0$ a $y_k(wk_k + b) \leq 1 - \xi k$. Hlavním rozdílem oproti SVM je nahrazením regulační hodnoty definované jako vzdálenost od hyperplane $\frac{2}{\vec{w}}$, normou $\ell_1 = \sum_{i=1}^{n} |w_i|$. Tato změna má velký dopad na finální optimalizaci v důsledku striktně konvexního tvaru tohoto kvadratického výrazu.

Předpokládejme dva lineární modely, jeden parametrizovaný vektorem $\vec{w_1}$, který využívá první polovinu atributů a druhý $\vec{w_2}$ využívající polovinu druhou. Pak jakákoli lineární kombinace $\vec{w} = (1 - \lambda)\vec{w_1} + \lambda\vec{w_2}$, $\lambda \in (0, 1)$ bude mít menší ℓ_2 normu než w_1 nebo w_2. Z toho vyplývá, že vybráním vektoru $\vec{w}$ s vetším počtem atributů, naopak podstatně zmenší velikost ℓ_2 normy. To ukazuje na tendenci SVM modelů, které využívají atributy, které jsou redundantní v rámci datového souboru, což je jedna ze silných stránek SVM, neboť rozdělení klasifikace na větší počet redundantních atributů zapříčiní větší odolnost vůči šumu. Z hlediska selekce příznaků to však může být nevýhoda a právě ℓ_1 forma se snaží odstranit tuto vlastnost a přiřazením stejné regulační hodnoty všem $\vec{w} \in (\vec{w_1}, \vec{w_2})$.

Používá se zde regulační parametr C, který slouží pro nastavení vlastností modelu relativní k empirické chybě, tedy prioritu margin vzdálenosti vůči nesprávné klasifikaci, které je popsaná v sekci věnované SVM. Jednou z nevýhod tohoto modelu je ovšem nemožnost použití standardních nepodmíněných optimalizačních technik řešení toho problému, neboť je nediferencovatelný.

1.4.2.2 Konkávní minimalizace

V případě lineárních modelů, proces selekce atributů z datového souboru může být chápáno jako následující optimalizační problém:

$$\min_{w,b} \ell_0(w) \tag{1.39}$$

za podmínky $y_k(wx_k + b) \geq 0$. Selekce atributů spočívá v hledání váhy $\vec{w}$ s co nejméně nenulovými hodnotami, což patří do NP-těžkých úloh a není proto možné ho vyřešit přímo. Popíšeme dva způsoby, jak je možné tento optimalizační problém aproximovat.

Feature Selection Concave (FSV) je způsob, který navrhl v roce 1998 Brandley a Mangasarian a popisuje aproximaci funkce $\ell_0(w)$ jako

$$\ell_0(w) \approx \sum_{i=1}^{n} 1 - exp(-\alpha|w_i|) \tag{1.40}$$

s koeficientem α s doporučenou počáteční hodnotou 5. Tento koeficient kontroluje příkrost funkce a její blízkost k $\ell_0(w)$.

1.4.2.3 Multiplicative Update

Druhou možností je Multiplicative Update navržený Weston et el. (2003). Tato aproximace používá lehce rozdílnou funkci, kterou nahradí ℓ_0 normu.

$$\sum_{i=1}^{n} \log(\epsilon + |w_i|) \tag{1.41}$$

Ačkoli tato funkce není dobrou aproximací ℓ_0, tak její minimum je podle Wenstona et al. blízko také minimu ℓ_0 normy a vede k iterativnímu schématu, jehož každý krok odpovídá standardnímu optimalizačnímu problému SVM.

1.4.2.4 LASSO

LASSO metoda (Least Absolute Shrinkage and Selection Operator) provádí jak zmenšování, tak automatickou selekci atributů a funguje na podobném principu jako ℓ_1 SVM. Využívá ℓ_1 normu na regresní koeficienty, tedy sumu absolutních hodnot s balančním parametrem. Mějme datový soubor o n instancích a m atributech. Cílem je minimalizovat následující problém

$$min \sum_{i=1}^{n} (y_i - \sum_{j=0}^{m} (w_j x_{ij}))^2 + \alpha \sum_{k=0}^{m} |w_k| \tag{1.42}$$

za podmínky $\alpha > 0$, což je podobné minimalizaci *sumy čtverců* a *ridge regresi* za odlišných omezujících podmínek. Použití normy ℓ_1, která je konvexní, vede k vytvoření řídkého modelu a k jednoduchosti řešení, což přispěje k výrazné výpočetní výhodě a dá se využít pro regresi i klasifikaci.

Parametr α reguluje důraz na minimalizaci sumy absolutních hodnot vah vůči zbytku. Hodnoty, kterých může α dosahovat zásadně mění koeficienty a minimalizační proces.

- $\alpha = 0$ - stejné koeficienty jako u lineární regrese
- $\alpha = \infty$ - všechny koeficienty nulové
- $\alpha = (0; \infty)$ - koeficienty mezi 0 a lineární regresí

Generalizované LASSO
Zobecněná technika, která zvládá klasifikační problémy s přijatelnější ztrátou pomocí

$$min \sum_{i=1}^{n} \log(1 + e^{-y_i(wx_i+b)}) \tag{1.43}$$

za podmínky $\sum_{i=1}^{m} |w_i| \leq \sigma_0$, která má za výhodu vytvoření řídkých modelů, jehož výstupy mohou být interpretovány jako pravděpodobnosti.

1.5 Klasifikační a regresní modely

Vybraní zástupci z výše popsaných metod budou otestovány z hlediska úspěšnosti správné klasifikace, časové náročnosti i počtu vybraných atributů na třech vybraných modelech, přičemž každý se hodí na různé typy úloh. Vybrán byl algoritmus **k-Nejbližších Sousedů** (k-Nearest Neighbours), **Support Vector Machines** a **Naivní Bayesův klasifikátor**. V této kapitole projdeme funkčnost jednotlivých algoritmů. Jejich konkrétní nastavení a parametry projdeme ve spojitosti s implementací v kapitole Implementace.

1.5.1 Algoritmus k-Nejbližšších Sousedů

Algoritmus k-nejbližšších sousedů (kNN) je jedním z nejjednodušších algoritmů určených pro klasifikaci nebo regresi. kNN patří mezi *neparametrické, líné* algoritmy.

Neparametrický znamená, že nedělá žádné předpoklady o rozdělení dat v datovém souboru. To se hodí v případě, pokud máme o datech velmi málo informací, či struktura dat nedodržuje nějaký typický předpoklad rozdělení (lineární separabilita, Gaussovský mixture model).

Algoritmus patří také do kategorie líných, což znamená, že ve fázi učení se neprovádí zobecnění z případů na trénovací množině dat. Dá se tedy říci, že algoritmus nemá žádnou trénovací fázi, nebo pouze minimální. Z toho vyplývá, že celý proces učení je velmi rychlý a lze klasifikovat data téměř za běhu. Oproti tomu, testovací fáze je náročná jak paměťově, neboť je potřeba uložit celou množinu trénovacích dat, ale i časově, protože všechny datové body se mohou podílet na řešení.

KNN předpokládá, že se data nacházejí v dimenzionálním prostoru, kde lze nějakou metrikou měřit jednotlivé vzdálenosti mezi těmito datovými body (jednotlivými záznamy) a tím odvodit jejich podobnost.

Pro body $X = (x_1, x_2, \ldots, x_n)$, $Y = (y_1, y_2, \ldots, y_n)$ a $Z = (z_1, z_2, \ldots, z_n)$ musí vzdálenost d splňovat následující podmínky, aby bylo možné porovnávat jejich podobnost:

- $d(X, Y) \geq 0$ (podobnost vzorů musí být různá)
- $d(X, Y) = 0$ právě tehdy, pokud $X = Y$ (neexistují dva stejné vzory)
- $d(X, Y) = d(Y, X)$ (symetrie)
- $d(X, Y) \leq d(X, Z) + d(Z, Y)$ (trojúhelníková nerovnost)

Jako metrika vzdálenosti se používá nejčastěji *Euklidovská vzdálenost*, ale také je možné použít *Manhatanskou* (či Hammingovu), *Čebyševovu* nebo například *Minkovského vzdálenost*.

Euklidovská vzdálenost je nejpoužívanější metrika měření vzdálenosti v případě kNN. Představuje měření dvou bodů obdobně jako měření pravítkem, jedná se tedy o přímou vzdálenost mezi dvěma body. Je definována jako druhá odmocnina sumy čtvercových vzdáleností dvou bodů.

$$\begin{aligned} d(X,Y) &= \sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\ldots+(x_n-y_n)^2} \\ d(X,Y) &= \sqrt{\sum_{i=1}^{n}(x_i-y_i)^2} \end{aligned} \tag{1.44}$$

Tato vzdálenost je vždy větší nebo rovno nule.

Manhattanská vzdálenost vychází z předlohy Newyorských ulic, kde jsou ulice na sebe kolmé a pravidelné. Jedná se o sumu horizontálních a vertikálních cest z bodu X do bodu Y. Není tedy možné jít diagonálně, je nutné procházet jednotlivé křižovatky blok po bloku.

$$\begin{aligned} d(X,Y) &= |x_1-y_1|+|x_2-y_2|+\ldots+|x_n-y_n| \\ d(X,Y) &= \sum_{i=1}^{n}|x_i-y_i| \end{aligned} \tag{1.45}$$

Tato vzdálenost se hodně využívá v integrovaných obvodech, pro klasifikaci nefunguje příliš dobře. I zde platí, že vzdálenost je větší nebo rovna nule.

Čebyševova vzdálenost prozkoumává absolutní velikost podobností mezi dvěma body. Tato vzdálenost je výhodná spíše pro data, jejichž podobnost se posuzuje spíše na základě individuálních parametrů, než na základě všech dostupných. Výpočet je následující:

$$d(X,Y) = \max_i |x_i-y_i| \tag{1.46}$$

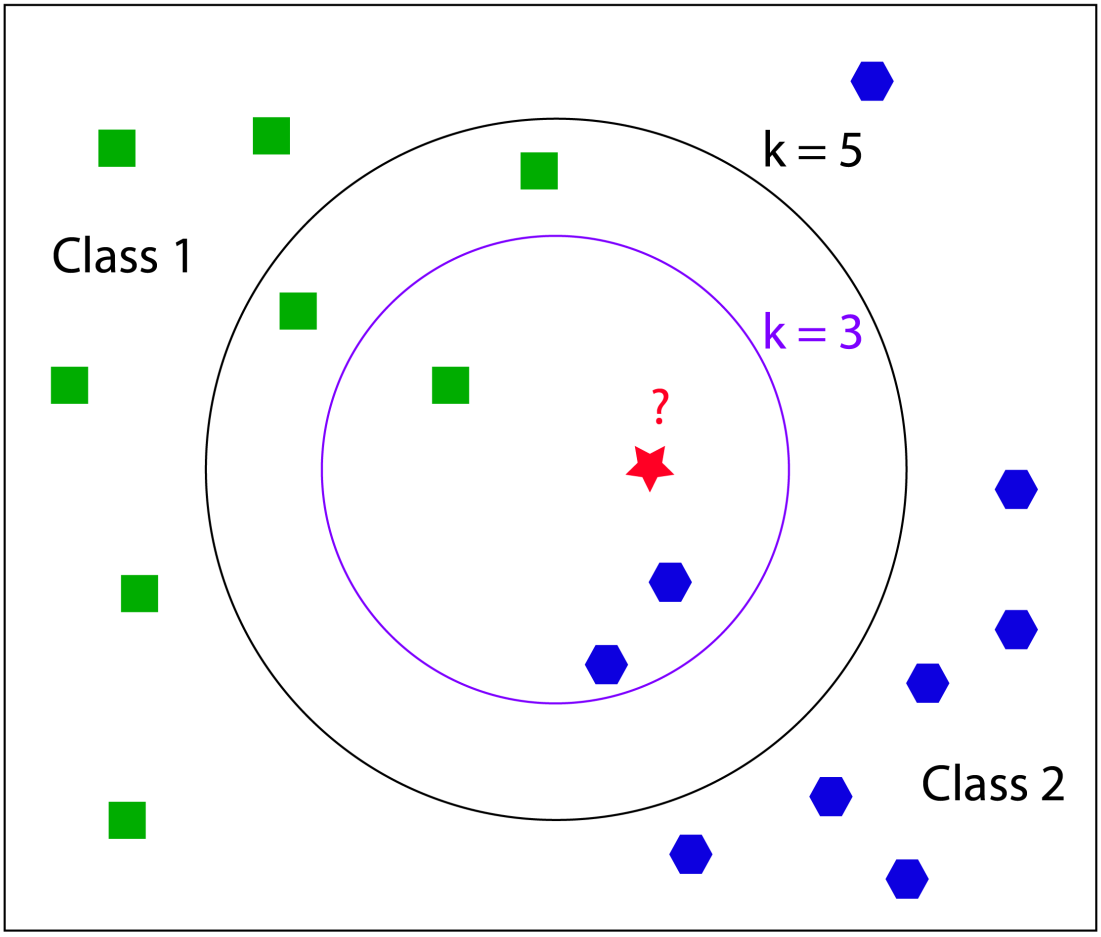
Výstupem je maximální hodnota podobnosti.

Minkowského vzdálenost představuje obecnou metriku pro měření vzdálenosti. Souvztažnost s ostatníma metrikami je zmíněna níže.

$$\begin{aligned} d(X,Y) &= \sqrt[\lambda]{|x_1-y_1|^\lambda+|x_2-y_2|^\lambda+\ldots+|x_n-y_n|^\lambda} \\ d(X,Y) &= \sqrt[\lambda]{\sum_{i=1}^{n}|x_i-y_i|^\lambda} \end{aligned} \tag{1.47}$$

V závislosti na hodnotě parametru λ je zde spojitost s ostatními metrikami.

- $\lambda = 1$: Manhattanská vzdálenost
- $\lambda = 2$: Euklidovská vzdálenost



Obrázek 1.12: Algoritmus k-nejbližších sousedů [41]

- $\lambda = \infty$: Čebyševova vzdálenost

Principem algoritmu je nalézt k nejbližších bodů podle nějaké metriky k určitému záznamu (obrázek 1.12). Výsledná třída se určí podle průměrné hodnoty sousedů nebo podle nejvíce frekventované hodnoty.

Důležitým parametrem je tedy v algoritmu k-nejbližších sousedů hodnota k, což představuje počet sousedů a výsledek je různý v závislosti na jeho velikosti.

Pro $k = 1$ se jedná o algoritmus nejbližšího souseda. Řekněme, že chceme klasifikovat bod (záznam) X. Najdeme tedy v dimenzionálním prostoru nejbližší datový bod (nejbližšího souseda) a převezme jeho třídu.

Pro $k = K$ se jedná o rozšíření metody nejbližšího souseda. Najdeme v okolí nejbližších k sousedů a hledaný bod má stejnou třídu jako většina nebo průměr z nich. Předpokládáme ale, že každý soused má stejnou váhu, což není vždy výhodné. Velmi často se algoritmus modifikuje na **vážené kNN**, kde každý soused je ohodnocen svojí váhou, která se typicky počítá pomocí vzdálenosti. To způsobí, že bližší sousedé mají na klasifikaci větší vliv než sousedi vzdálenější.

Samotné hledání nejbližších sousedů je náročný proces a existuje více způsobů, jak sousedy nalézt, případně jak samotné hledání urychlit. Jendou z možností je využití ***hrubé síly***, což je často nevyhovující a hledání by se mohlo prodloužit a právě rychlost a jednoduchost je jedna z hlavních výhod tohoto algoritmu.

Rychlejší a jednou z nejběžněji používaných metod jsou ***kd-Tree***. Hlavní podstatou tohoto algoritmu je vytvoření stromové struktury z dat. V každém uzlu jsou body rozděleny do dvou větví na základě hraniční hodnoty jednotlivých dimenzí. Body větší než tato hodnota se nacházejí v jedné části podstromu, menší v druhé části. V dalších krocích dojde k půlení na základě dalších dimenzí a po poslední dimenzi se v listech nacházejí jednotlivé body. Jedná se tedy strukturou o binární strom.

Důležitou volbou je rozhodnutí, které dimenze rozpůlit a jaká bude hraniční hodnota. Vybráním dimenze s velkou odchylkou vede k menším stromům. Hraniční hodnota je většinou určena mediánem hodnot dimenze, což vede k vyváženému rozdělení bodů.

Algoritmus dosahuje dobrých výsledků pro data o malých dimenzích, ačkoli má několik nevýhod.

1. Počet sousedů listů roste exponenciálně s dimenzí, brzy se z toho stává lineární prohledávání.

2. Rozdělení uzlů je vždy založeno na osách a nezávislé na rozložení dat, což může způsobovat slabý výpočetní výkon.

V reálné aplikaci je metoda aproximativní, tedy že nedojde vždy k nalezení nejbližších sousedů. Což způsobí zrychlení, neboť prohledávání není lineární, může mít však špatné dopady na výsledky v závislosti na aplikaci, ve které je algoritmus použitý.

Další metodou jsou ***PCA Tree***, což řeší problém *kd-Tree* použitím Principle Components Analysis na každý uzel s cílem získáním vektoru korespondující s maximálním rozptylem a rozdělit body podle něho.

Zmíním ještě metodu ***Ball Tree***, což je algoritmus efektivní hlavně na data o velkém počtu dimenzí. Obdobně jako u *kd-Trees* se jedná o binární strom s hierarchickou strukturou, avšak jeho sestavení je náročnější.

V algoritmu *Ball Tree* každý uzel („Ball“) reprezentuje množinu bodů a má dva potomky, což je opět podmnožina bodů, jejichž průnik je nulový a sjednocení je rovno množině rodiče. Každý z těchto potomků má takzvaný centroid, což představuje střed těchto bodů. Vzdálenost k nejzazšímu uzlu příslušícímu k tomuto centroidu se nazývá radius, což definuje jeho pole vlivu. Uzly, nacházející se v hlubších částí stromu mají radius menší, naopak ty na vrchu zahrnují větší prostor. Každý nový bod, který chceme přiřadit, rekurzivně připojíme k právě jednomu uzlu, jehož radius pokrývá tento bod a vzdálenost k jeho centoridu je ta nejmenší.

1.5.2 Support vector machines

Support Vector Machine (SVM) je algoritmus, který se dá využít jak pro klasifikaci, tak pro regresi, nicméně spíše se využívá pro klasifikační problémy. Obdobně jako u kNN, algoritmus se znázorňuje v dimenzionálním prostoru, kde jeden element odpovídá jednomu datovému bodu se souřadnicemi vypočítaných z jeho vlastností. Cílem SVM je nalézt takzvanou *hyperplane*, což je přímka, která nejlépe rozděluje jednotlivé třídy.

Důležitým pojem v souvislosti s SVM je vzdálenost přímky k nejbližšímu bodu, které se říká *margin vzdálenost*. Cílem je, aby byl prostor mezi hyperplane a nejbližším bodem co největší, tedy maximalizace margin vzdálenosti.

Úspěšnost klasifikace a margin vzdálenost jsou dvě různé proměnné a v modelu má správnost klasifikace větší prioritu, než maximalizace margin vzdálenosti. Ve výsledku se tedy zahrne datový bod i za cenu zmenšení vzdálenosti. Pokud to však není možné oddělit a datový bod zasahuje hluboko do prostoru jiné třídy (či jiná anomálie), teprve až pak model připustí nesprávnou klasifikaci.

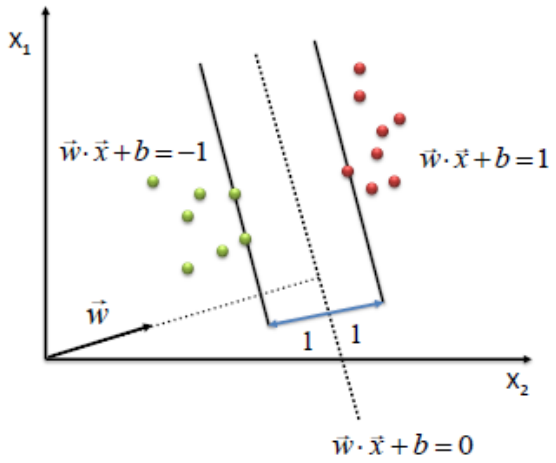
1.5.3 Lineární SVM

Pro dvě třídy, které jsou lineárně separabilní, existuje mnoho různých separátorů. U SVM se bere v potaz maximalizace margin vzdálenosti, Perceptronový algoritmus uvažuje jakoukoli separaci, Naivní Bayesův klasifikátor hledá separaci podle specifických kriterií. V praxi to znamená, že rozhodovací funkce, která určí hranici rozdělení, bude definovaná pouze malou podmnožinou dat nacházejících se na pomezí, kterým se říka *support vektory*.

Mějme normálový vektor $\vec{w}$, označovaný jako váhový vektor, který je kolmý na hyperplane. Definujme si proměnou b, jejichž specifikací spolu s vektorem $\vec{w}$ získáme přesnou polohu hyperplane. Jelikož je hyperplane kolmá na vektor $\vec{w}$, tak platí pro všechny její body, že $\vec{x}$
$\vec{w}^T\vec{x} = -b$
Pro trénovací datovou množinu $\vec{x}$ a jejich zařazení do dvou tříd y nabývající hodnot $\{-1, 1\}$ tvoří datový soubor $D = (\vec{x_i}, y_i)$. Lineární klasifikátor má potom následující podobu.

$$f(\vec{x}) \quad = \quad sign(\vec{w}^T\vec{x} + b) \tag{1.48}$$

$$f(\vec{x}) \quad = \quad \pm 1 \tag{1.49}$$



Obrázek 1.13: Vztahy vektorů v rámci SVM [2]

Lze také vypočítat Euklidovskou vzdálenost vektoru $\vec{x}$ od hyperplane. K tomu účelu si definujme hodnotu r, která značí právě nejkratší vzdálenost libovolného bodu k hyperplane. Víme, že nejkratší vzdálenost bude vždy kolmá k hyperplane a tedy rovnoběžná k normálovému vektoru $\vec{w}$. Normovaný vektor od vektoru $\vec{w}$ v tomto směru je $\frac{\vec{w}}{|\vec{w}|}$.

Bodem $\vec{x'}$ rozumíme nejbližší bod na hyperplane k bodu $\vec{x}$, tedy takový, že je lze spojit kolmicí. Potom platí:

$$\vec{x'} \quad = \quad \vec{x} - yr\frac{\vec{w}}{|\vec{w}|} \tag{1.50}$$

Výstupní třída y pouze upravuje směr přímky, neboť dosahuje hodnot 1 a -1 v tomto případě. Vzhledem tomu, že $\vec{x'}$ leží na hyperplane, musí nutně platit následující vztah

$$\vec{w^T}\vec{x'} + b \quad = \quad 0 \tag{1.51}$$

$$\vec{w^T}(\vec{x} - yr\frac{\vec{w}}{|\vec{w}|}) + b \quad = \quad 0 \tag{1.52}$$

z toho jednoduše odvodíme r a získáme naší hledanou vzdálenost.

$$r = y\frac{\vec{w^T}\vec{x} + b}{|\vec{w}|} \tag{1.53}$$

V této problematice se snažíme najít maximální margin, najít co největší vzdálenost r k support vektorům datového setu. Pro každý vzor i vypočítáme jeho vzdálenost $r_i = y_i\frac{\vec{w^T}\vec{x_i}+b}{|\vec{w}|}$. Dále pro geometrickou velikost margin platí vztah $\rho = 2\frac{1}{|\vec{w}|}$, neboť se jedná o dvě stejné délky od hyperplane. Chceme najít takové $\vec{w}$ a b, že

- $\rho = \frac{2}{\vec{w}}$ je maximální pro všechny $(\vec{x_i}, y_i) \in D$
- $y_1(\vec{w}^T\vec{x_i} + b) \geq 1$

Maximalizace ρ je to samé, jako minimalizace $\frac{1}{\rho}$, čili $\frac{\vec{w}}{2}$. To představuje spolu se snahou maximalizací úspěšnosti standardní optimalizační problém SVM - $\max_{vzdalenost} + \min_{chyba}$.

Snažíme se optimalizovat kvadratickou funkce s lineárními omezeními. Kvadratická optimalizace je známý problém a existuje mnoha algoritmů, které to řeší. V kontextu SVM se využívá Lagrangeův multiplikátor α, pro který platí:

- $\sum\limits_i \alpha_i - \frac{1}{2}\sum\limits_i\sum\limits_j \alpha_i\alpha_j y_i y_j \vec{x_i}^T\vec{x_j}$ je maximální
- $\sum\limits_i \alpha_i y_i = 0$
- $\alpha_i \geq 0,\ 1 \leq i \leq N$

tím získáme požadované řešení,představující hodnoty parametrů v následující formě:

$$\vec{w} = \sum_i \alpha_i y_i \vec{x_i} \tag{1.54}$$

$$b = y_k - \vec{w}^T\vec{x_k} \text{ pro jakékoliv } \vec{x_k} \text{ takové, že } \alpha_k \neq 0 \tag{1.55}$$

Většina multiplikátorů α_i je nulových, pokud je však hodnota nenulová, tak víme, že odpovídající hodnota x_i tohoto multiplikátoru v datovém setu je support vektor. Klasifikační funkce tedy je:

$$f(\vec{x}) = sign(\sum_i \alpha_i y_i \vec{x_i}^T\vec{x} + b) \tag{1.56}$$

1.5.4 Nelineární SVM

Existují klasifikační problémy, které nelze oddělit přímkou, tudíž by si s tím klasické SVM neporadilo. Řešení je převést datový soubor do vyšší dimenze za předpokladu zachování příbuznosti jednotlivých bodů. Takto zobrazený datový soubor je možné rozdělit lineárně. Úkolem je provést správné mapování.

Samotnou transformaci dat do vyšší dimenze zajišťuje funkce ϕ, která zobrazí datový bod $\vec{x}$ na $\phi(\vec{x})$ o vyšší dimenzi. Pokud opět dosadíme do klasifikátoru získáme:

$$f(\vec{x}) = sign(\sum_i \alpha_i y_i \phi(\vec{x_i})^T\phi(\vec{x}) + b) \tag{1.57}$$

Místo toho, aby se vektorový součin $\phi(\vec{x_i})^T\phi(\vec{x})$ počítal, tak spočítáme *kernelovou funkci*, což je jednodušší a efektivnější způsob.

Kernelová funkce $K(\vec{x_i}, \vec{x_j})$ je funkce $K : R^N \times R^N \Rightarrow R$ definovaná skalárním součinem bodů $\vec{x_i}$ a $\vec{x_j}^T$ a znázorňuje míru podobnosti mezi i-tým a j-tým vektorem složky $\vec{x}$. Jedná se tedy o funkci, která koresponduje s daným skalárním součinem v nějakém rozšířeném prostoru. Přepis vypadá tedy

$$K(\vec{x_i}, \vec{x_j}) \quad = \quad \phi(\vec{x_i})^T \phi(\vec{x_j}) \tag{1.58}$$

To nám umožní vypočítat tyto produkty bez znalosti ϕ nebo prostoru. Typické kernel funkce jsou následující:

- linearní: $K(\vec{x}, \vec{y}) = (\vec{x}^T \vec{y})$
- polynomiální: $K(\vec{x}, \vec{y}) = (\vec{x}^T \vec{y} + c)^d$
- radiální bázové (normální rozdělení): $K(\vec{x}, \vec{y}) = e^{-\frac{(\vec{x}-\vec{y})^2}{2\sigma^2}}$
- sigmoidní: $K(\vec{x}, \vec{y}) = tanh(\vec{x}^T \vec{y} + c)$

Lineární kernel funkce je nejjednodušší funkce a doporučuje se pro textovou klasifikaci, neboť většina textů je lineárně separabilní. Funkce je ve tvaru $\phi(\vec{x}) = \vec{x}$.

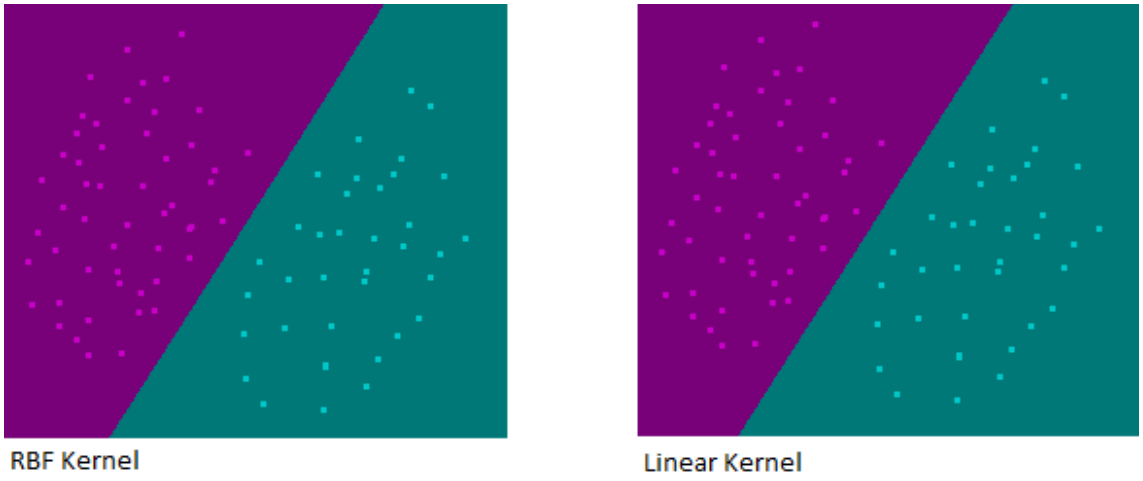
Tato funkce se hodí do prostoru s velkým počtem atributů, neboť převedením tohoto většího množství atributů do vyššího dimenzionálního prostoru příliš nezvýší přesnost klasifikátoru, pouze se zvýší výpočetní náklady funkce. U textové klasifikace je instancí celý dokument a atributy jsou jednotlivá slova, kterých je obvykle větší počet, proto se lineární kernel uvažuje v souvislosti s textovou klasifikací.

Na obrázku 1.14 je vidět srovnání radiální bázové a lineární funkce na textovém dokumentu a lze vypozorovat, že vygenerovaná hyperplane je téměř totožná v případě lineární, tak radiální bázové funkce. Mapování atributů na vyšší dimenze se v tomto případu nevyplatí. Nehledě na to, že vzhledem k jednoduchosti je lineární kernel rychlejší a trénování SVM klasifikátoru proběhne rychleji.

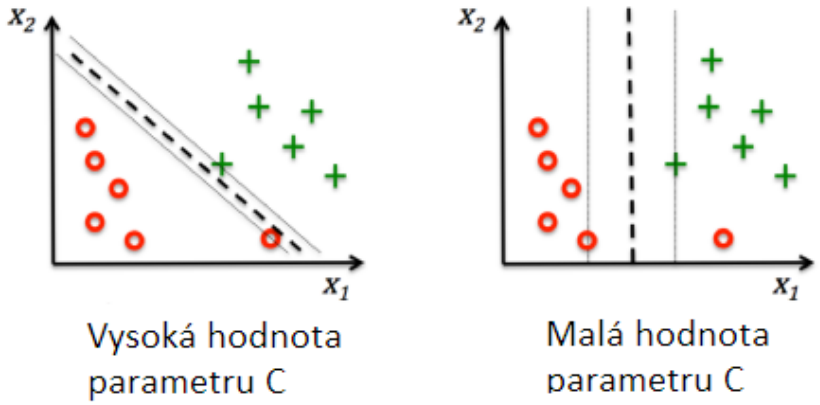
Lineární kernel se vyplatí vyzkoušet jako první z výše uvedených vzhledem ke své jednoduchosti a snadné interpretovanosti. Nelze ho však použít u všech případů.

Při nastavování SVM klasifikátoru budeme pracovat s parametrem C, který představuje jakýsi regulátor 1.15. Jeho hodnota nám definuje, jak moc se chceme vyhnout nesprávné klasifikaci. Při větší hodnotě C na trénovací množině bude mít výsledná hyperplane malou margin, tedy dojde ke správné klasifikaci u většího množství elementů. To může dobře fungovat na trénovací množině dat, nicméně je zde riziko přeučení, neboť hranice mezi třídami je z důvodu tohoto nastavení malá.

Naopak nízké hodnoty C mohou způsobit hrubější klasifikaci a tedy zvýšení počtu nesprávně ohodnocených bodů na trénovací množině, odpadá však zvýšené riziko přeučení.



Obrázek 1.14: Provnání RBF a lienární kernel funkce na textovou klasifikaci [20]



Obrázek 1.15: Závislost parametru C [2]

1.5.5 Klasifikace pomocí SVM do více tříd

SVM se používá ke klasifikace většinou do dvou tříd. Pro klasifikace do více tříd se používají dva přístupy

1. Vytvoření klasifikátoru $|C|$ - „jeden proti zbytku“
2. Vytvoření více klasifikátorů $\frac{|C|(|C|-1)}{2}$ - „jeden proti jednomu“

Zatímco u prvního přístupu je potřeba vytvořit jediný klasifikátor, který rozdělí data do více tříd a je časově náročný, naopak u druhého případu je nutné vytvořit více klasifikátorů s kratší dobou trénování, neboť trénovacích dat pro každý klasifikátor je o dost méně.

1.5.6 Bayesův klasifikátor

Jedná se o statistickou metodu, odvozené od Bayesovy věty. Bayesova věta nám udává pravděpodobnost jevu A za podmínky jevu B, kterou nazýváme „aposteriorní pravděpodobností“. Tuto pravděpodobnost odvodíme z následujících vztahů.

$$P(B \cap A) = P(B|A)\,P(A) \tag{1.59}$$

$$P(A \cap B) = P(A|B)\,P(B) \tag{1.60}$$

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)} \tag{1.61}$$

Pravděpodobnost jevu B, které se říká „normalizační konstanta“ spočítáme následovně

$$P(B) = \sum_i P(B|A_i)\,P(A_i) \tag{1.62}$$

Uvedením do kontextu klasifikátoru, mějme třídu (jev) Y_i v trénovacích datech X. Cílem je nalézt **maximální aposteriorní pravděpodobnost** (MAP). Tedy nejvíce pravděpodobnou třídu na základě trénovacích dat.

$$Y_{MAP} = \max P(Y_i|X) \tag{1.63}$$

$$Y_{MAP} = \max \frac{P(X|Y_i)\,P(Y_i)}{P(X)} \tag{1.64}$$

V reálu však neznáme závislost jednotlivých jevů, a kdy se klasifikuje na základě více než jednoho parametru. Proto si formuli upravíme, aby zahrnovala n vstupních atributů a s předpokladem, že jsou dané parametry nezávislé, tedy $P(X_i|Y, X_1, \ldots, X_n) = P(X_i, Y)$. Této úpravě se říká **Naivní Bayesův klasifikátor** a má následující podobu:

$$P(Y|X_1, \ldots, X_n) = \frac{P(X_1, \ldots, X_n|Y)P(Y)}{P(X_1, \ldots, X_n)} \tag{1.65}$$

což měří pravděpodobnost třídy Y za podmínky $X_1, \ldots, X_n$ atributů.

Kapitola 2

Implementace

Kapitola Implementace se věnuje implementaci vybraných metod v jazyce Python. Je použité množství datových souborů, které pokrývají co nejvíce možných vlastností dat, z hlediska počtu atributů, počtu tříd, velikost instance a vzájemných vazeb. Data byla základním způsobem upravena a připravena pro obecnou práci s nejrůznějšími metodami. Došlo k doplnění chybějících hodnot a diskretizaci atributů.

2.1 Datové soubory

Datové soubory 2.1, na kterých budeme testovat vybrané metody, jsou různé co se týče počtu a typu atributů, jejich velikosti, úplnosti a počtu tříd, do kterých je lze zařadit. Při menším počtu atributů je možné získat exaktní řešení hrubou silou, nicméně to by nám neposkytlo žádný náhled na využitelnost daných metod.

Vybrané datové soubory by měly pokrýt co největší rozsah možných datových schémat a poskytnout co největší variabilitu při měření výkonnosti jednotlivých metod. Výsledek dostaneme měřením jednotlivých metod na všech souborech a následným zprůměrováním hodnot.

2.2 Předzpracování dat

Před samotným vyhodnocením selekčních metod je potřeba data upravit, jednak z důvodu, aby jejich reprezentace odpovídala podobně, která je vhodná pro selekční metody, ale také aby se zvýšila přesnost modelu a zároveň se zachovalo co nejvíce informací. Nejprve je potřeba zpracovat kategorické atributy, které se převedou z textové reprezentace na numerickou. Následně je potřeba nastavit postup pro zacházení s chybějícími hodnotami a opravit případné chybné hodnoty v datech.

Tabulka 2.1: Použité datové soubory

Datový soubor	Atributů	Typ atributů	Instancí	Chybějící hodnoty	Tříd
Kosatce Originál: Iris	4	Reálné	150	Ne	3
Druhy aut Originál: Car Evaluation	6	Kategorie	1,728	Ne	4
Hledání míst Originál: Nomao	119	Reálné	34,465	Ano	2
Šachová zakončení Originál: Chess (King-Rook vs. King)	6	Kategorie, Numerické	28,056	Ne	18
Volby do kongresu Originál: Congressional Voting Records	16	Kategorie	435	Ano	2
Internetové reklamy Originál: Internet Advertisments Data	1,558	Kategorie, Numerické	3,279	Ano	2
Druhy zvířat Originál: Zoo	17	Kategorie, Numerické	101	Ne	7
Výsledky piškvorků Originál: Tic-Tac-Toe Endgame	9	Kategorie	958	Ne	2
Příjem dospělých Originál: Adult	14	Kategorie, Numerické	48,842	Ano	2
Rozdělení ecoli Originál: Ecoli	8	Reálné	336	Ne	8
Srdeční arytmie Originál: Arrhythmia	279	Reálné, Kategorie, Numerické	452	Ano	16
Druhy hub Originál: Mushroom	22	Kategorie	8,124	Ano	2
Rakovina plic Originál: Lung Cancer	56	Numerické	32	Ano	3
Kvalita vína Originál: Wine Quality	11	Reálné	4,898	Ne	11
Kategorie dokumentů Originál: CNAE-9	856	Numerické	1,080	Ne	9

2.2.1 Diskretizace atributů

Pro použití metod, které používají numerické vstupy (zejména patřící mezi statistické), musíme kategorické (nečíselné) hodnoty nahradit odpovídající číselnou hodnotou. Pod kategorický atribut spadají následující typy proměnných:

- Nominální (auto, dům, ...)
- Ordinální (datum, úrovně, ...)
- Kvantitativní (intervaly, poměry)

U nominálních atributů můžeme pouze říci, zda se jeden od druhého liší. Nahrazujeme je tedy víceméně libovolným číslem, zatímco u ordinálního jsme schopni určit pořadí, jak jdou za sebou. Jako optimální možnost se jeví nahrazením pořadovým číslem, nicméně nemáme k dispozici pořadník těchto hodnot, proto budeme tyto hodnoty nahrazovat pořadovým číslem, jakým se vyskytují v datech. Kvantitativní datové typy se ve vybraných setech nevyskytují, proto není potřeba implementovat jejich převod do numerických hodnot. Touto transformací umožním jediné implementaci zpracovávat veškeré atributy bez nutnosti speciálního zacházení.

2.2.2 Nahrazení chybějících hodnot

Dalším krokem je nahrazení chybějících hodnot. Existuje více způsobů, jakým způsobem s chybějícími daty zacházet. Mezi nejtriviálnější způsoby patří odstranění řádku s chybějící hodnotou, což je snadno implementovatelné, ale dostatečné jenom pro malý počet chybějících dat v rozsáhlé instanci.

Další možností je nahrazení dle agregace: *průměrnou, nejčastější hodnotu, mediánem, nejbližšími sousedy.* Agregovat se může po řádcích nebo sloupcích, což je častější.

V práci je použita regresní metoda nejbližších sousedů (kNN), neboť umožňuje predikovat jak diskrétní, tak spojité hodnoty. Diskrétní hodnoty nahradí nejčastějším výskytem vybraných sousedů, u spojitých je hodnota predikována průměrem. Dále kNN umožňuje pracovat s jakýmkoli atributem ve stejném smyslu jako při klasifikaci třídy, tedy predikovat hodnotu jakéhokoli atributu. Pouze je potřeba předefinovat atributy určující vzdálenost mezi body. To umožní zacházet s řádky s více chybějícími hodnotami.

Nevýhodou metody je její časová náročnost, neboť při každém hledání nejpodobnějšího výsledku, musí projít celý datový soubor. V datech, kde kNN trvá neúměrně dlouho (hledání míst), je jako nahrazovací mechanismus použito imputace průměrem (po sloupcích).

Samotné kNN je zde implementované jako regresní model s parametry:

- Počet nejbližších sousedů: *5*
- Váhy atributů: *uniformní*
- Algoritmus hledání sousedů: *Ball Tree*
- Počet listů stromu: *30*
- Metrika vzdálenosti: *Minkowského*

2.2.3 Nahrazení chybných hodnot

Veškerá úprava dat, úprava chybných atributů a překlepů probíhala manuálně, pokud to metodám neumožnilo správnou funkci. V implementaci tedy nebyl implementován žádný mechanismus, který by obstarával čištění dat. Zbývající anomálie byly ponechány a simulují tak reálný datový soubor i se šumem.

2.3 Nastavení klasifikačních modelů

Klasifikační modely, které byly použity pro měření v kombinaci s filtračními a wrapperovými metodami jsou následující. U embedded se nastavení liší jednotlivě pro každé metody a není možné použít jednotný model.

Support Vector Machines

- Regulátor C: *1*
- Kernelová funkce: *Radial Basis Function*
- Gamma koeficient kernelové funkce: $\frac{1}{features}$
- Shrinking heuristika: *Ano*
- Tolerance ukončovacího kritéria: 10^{-3}
- Velikost cache: *200MB*

k-Nejbližších sousedů

- Počet nejbližších sousedů: *Počet tříd*
- Váhy atributů: *uniformní*
- Algoritmus hledání sousedů: *Ball Tree*
- Počet listů stromu: *30*
- Metrika vzdálenosti: *Minkowského*

Naivní Bayes

- Pravděpodobnost jednotlivých tříd: *Na základě dat*

Kapitola 3

Měření

V této kapitole změříme úspěšnost a časovou náročnost vybraných selekční metod a porovnáme je na různých datech s různými parametry a tím zjistíme použitelnost a potenciál jednotlivých metod a umožní nám prozkoumávat tyto metody z různých pohledů. Také budeme testovat nastavení jednotlivých parametrů a snažit se odvodit jejich optimální hodnoty.

3.1 Sestava

Měření byla prováděna na počítači s parametry:

- Procesor: Intel(R) Core(TM) i3-6100 CPU @ 3.7GHz
- Paměť: RAM 16GB DDR4 2133MHz, CL16
- Disk: SSD 256GB SATA III, čtení a zápis maximálně 450MB/s
- Operační systém: Windows 10 64bit

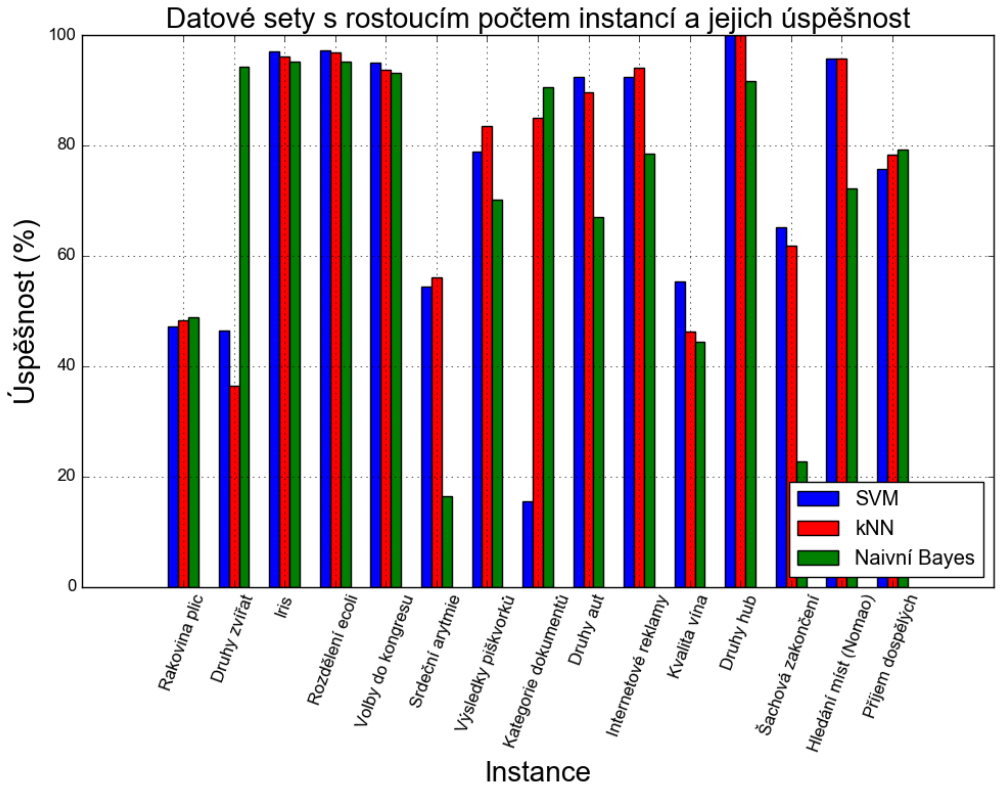
3.2 Bez selekce příznaků

Prvním krokem bude měření klasifikátorů na celé množině dat, bez jakékoli selekce atributů. Tento základní přístup nám umožní odhalit, které soubory jsou stavěné pro selekci a které nikoli a zároveň porovnáme z hlediska úspěšnosti jednotlivé varianty i s následujícími měřeními s již už provedenou selekcí. Pro vybraná dat jsou výsledky měření zachycena v tabulce 3.1.

Závislost na velikosti instance
Pokud seřadíme data podle velikosti instancí, můžeme porovnat jednotlivé úspěšnosti klasifikátorů s ohledem na tento parametr. Úspěšnost je definována procentem správně klasifikovaných řádků (záznamů) na testovací množině dat.

Tabulka 3.1: Měření za použití všech přiznaků

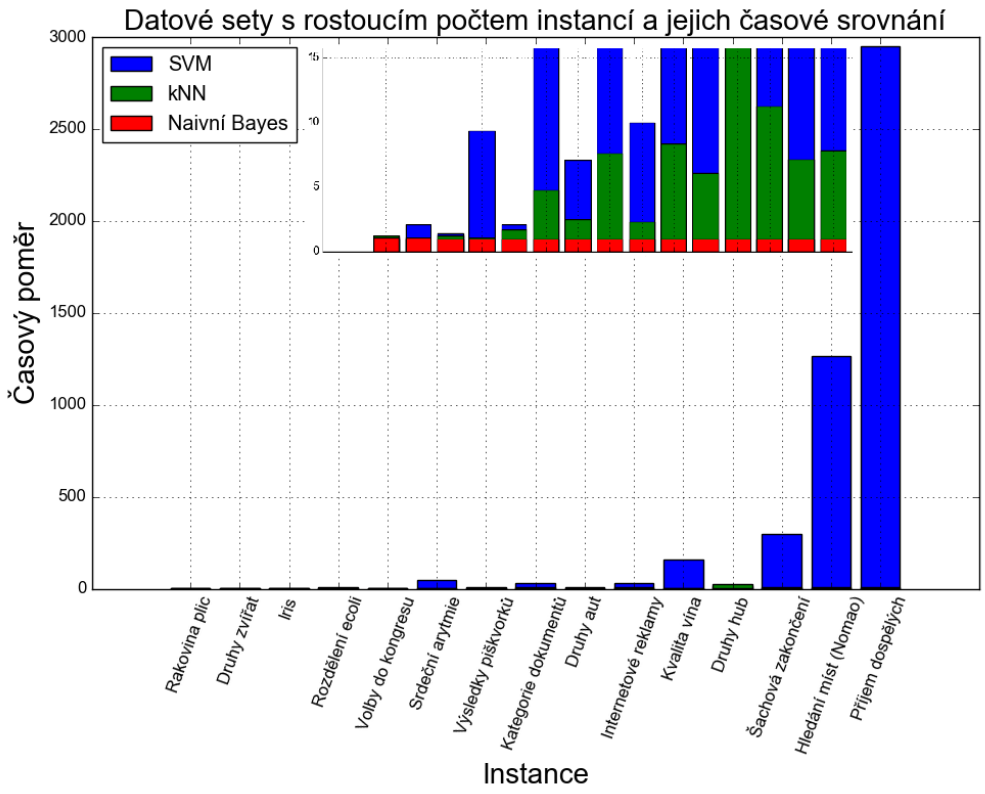
Modely / Data	SVM		kNN		Naivní Bayes	
	Čas (s)	Úspěšnost (%)	Čas (s)	Úspěšnost (%)	Čas (s)	Úspěšnost (%)
Iris	0,0010	97,10	0,0009	96,06	0,0007	95,24
Druhy aut	0,0282	92,40	0,0066	89,63	0,0028	67,09
Hledání míst (Nomao)	158,9867	95,72	0,8937	95,68	0,1254	72,24
Šachová zakončení	12,9188	65,18	0,4848	61,75	0,0431	22,79
Volby do kongresu	0,0025	94,99	0,0020	93,69	0,0012	93,14
Internetové reklamy	3,6961	92,34	0,9119	94,13	0,1098	78,55
Druhy zvířata	0,0016	46,44	0,0007	36,37	0,0008	94,23
Výsledky piškvorků	0,01333	78,93	0,0046	83,57	0,0019	70,17
Příjem dospělých	136,3877	75,81	0,3605	78,36	0,0462	79,29
Rozdělení ecoli	0,0109	97,31	0,0012	96,84	0,0012	95,14
Srdeční arytmie	0,1704	54,48	0,0178	56,06	0,0037	16,38
Druhy hub	0,1685	99,97	0,2791	99,98	0,0111	91,74
Rakovina plic	0,0007	47,20	0,0008	48,39	0,0007	48,82
Kvalita vína	1,1760	55,38	0,0451	46,29	0,0075	44,46
Kategorie dokumentů	1,2162	15,50	0,2872	84,91	0,0377	90,58



Obrázek 3.1: Úspěšnost klasifikátorů v závislosti na velikosti instance

Z měření 3.1 zjistíme, že vazba mezi mírou úspěšnosti a počtem instancí není významná a nenacházíme žádnou spojitost mezi těmito vztahy. Klasifikátor SVM dosahuje až na pár výjimek nejlepších úspěšností. Naopak Naivní Bayes vyšel z tohoto měření jako nejslabší, ač se většinou výsledky příliš neliší.

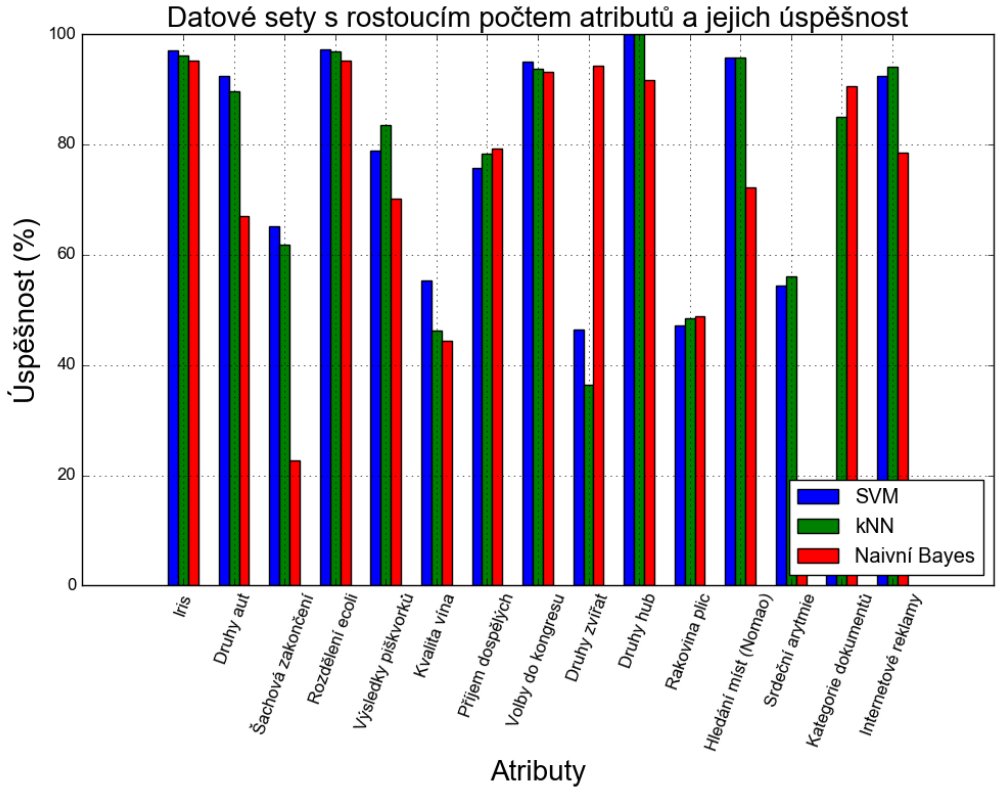
Z časového hlediska je vizualizace provedena poměrově 3.2, neboť výchylky v době trvání by nešlo přehledně vizualizovat. Zobrazena je tedy hodnota, kolikrát déle trval výpočet oproti nejrychlejšímu modelu. Vidíme, že největší podíl času zabírá právě SVM. Pokud nevybereme žádné atributy, tak pro lepší úspěšnost ohodnocení musíme počítat s větší časovou dotací. Naproti tomu Naivní Bayes si udržuje přibližně stejnou náročnost napříč všemi daty.



Obrázek 3.2: Doba výpočtu klasifikátorů v závislosti na velikosti instance

Závislost na počtu atributů
Druhým pohledem je porovnat výsledky s ohledem na počet atributů, vizualizováno na 3.3, jejichž počtem roste komplexnost dat. Sledujeme ale, že pouhý počet atributů také nemá přímou vazbu na úspěšnost klasifikátoru. Tuto informaci jsme si chtěli ověřit, neboť za daty se skrývá spousta dalších vzájemných vztahů, jiné míry důležitosti atributů a jiné typy

atributů. Právě měření selekčních metod nám pomůže odhalit více z těchto vazeb.



Obrázek 3.3: Úspěšnost klasifikátorů v závislosti na počtu atributů

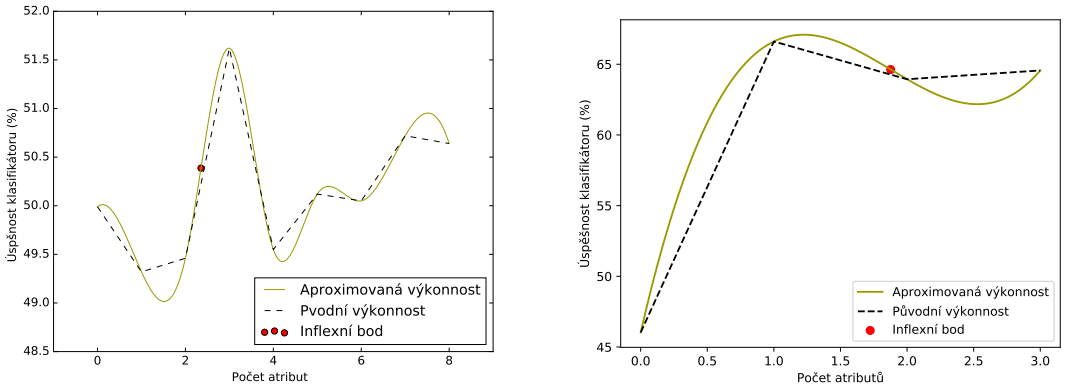
3.3 Vybírání podmnožiny atributů

U metod, ohodnocující jednotlivé příznaky podle jejich důležitosti musíme určit postup, jaké příznaky do výsledného počtu atributů zařadit.

Máme možnost do toho procesu hledání *zapojit klasifikátor*, tím nám splynou filtrační a wrapperové kategorie, lépe řečeno kombinujeme oba přístupy. Ohodnocení atributů nám sestaví pomyslný žebříček seřazených atributů dle důležitosti, které se přidávají do výsledné množiny atributů a od klasifikátoru získáváme zpětnou vazbu v podobě skóre, která v závislosti na nastavení zastaví přidávání dalších atributů v pořadí a prohlásí výběr za ukončený. Pro tento způsobem jsme použili algoritmy, které ukončí přidávání přístupu při:

- Zhoršení skóre klasifikátoru (*FS*)
- Existenci inflexního bodu (*IP*)

V první variantě se do množiny přidávají atributy od nejlepšího po nejhorší, dle ohodnoceného seznamu atributů. Výběr končí, pokud přidáním dalšího atributu v pořadí se procentuální úspěšnost nezlepší. Tento hladový přístup začíná tedy od prázdné množiny a výsledný počet atributů je co nejmenší, ač se najdou určitě lepší varianty z wrapper rodiny metod pro kombinaci s filtračními.



Obrázek 3.4: Selekce příznaku na základě inflexního bodu

U druhé varianty je postup podobný, také dochází k přidávání nejlepších příznaků, ale v každém kroku se aproximuje křivka počtu atributů se závislostí na skóre nějakou funkcí. Pokud tato funkce má inflexní bod (mění se z konvexní na konkávní nebo naopak), výběr se ukončí. Aproximace křivky a jejich inflexní body můžeme vidět na příkladech v 3.4. Právě existence tohoto bodu je pro nás ukazatel, že se mění stupeň výkonnosti pro model (ač nemusí jít pro pokles). Aby bylo možné aproximovat křivku, je nutné mít definované

alespoň tři body, tedy musí být přítomny alespoň tři atributy, což je podmínka, která se dotkne pouze menších datových souborů. Nicméně aproximace diskrétních hodnot nějakou spojitou funkcí není zcela vypovídající a výběr atributů se občas ukončí dříve, než je vhodné.

Pokud do výběru *klasifikátor nezapojíme*, jedná se o čistě filtrační metodu a výsledný počet atributů můžeme být omezen

- Definováním konstanty k
- Procentuálním zastoupením z celku (RS)
- Definováním hranice užitečnosti (TS)

Definování konstantního počtu k atributů, které budou použity, není příliš šikovné v kombinaci s různými daty s rozdílným počtem atributů. Lepší možností je ale určení procentuálního zastoupení příznaků, jehož hodnotu změříme v následujícím experimentů.

Hranicí užitečnosti rozumíme práh skóre, který čárou rozdělí atributy na ty, co budou a nebudou součástí výsledné podmnožiny. Definováním konkrétní hodnoty prahu se budeme také zabývat níže.

3.3.1 Procentuální zastoupení atributů

Procentuální zastoupení bylo měřeno na pěti vzorových datových souborech, přičemž každý z nich se liší od ostatních nějakou charakteristikou.

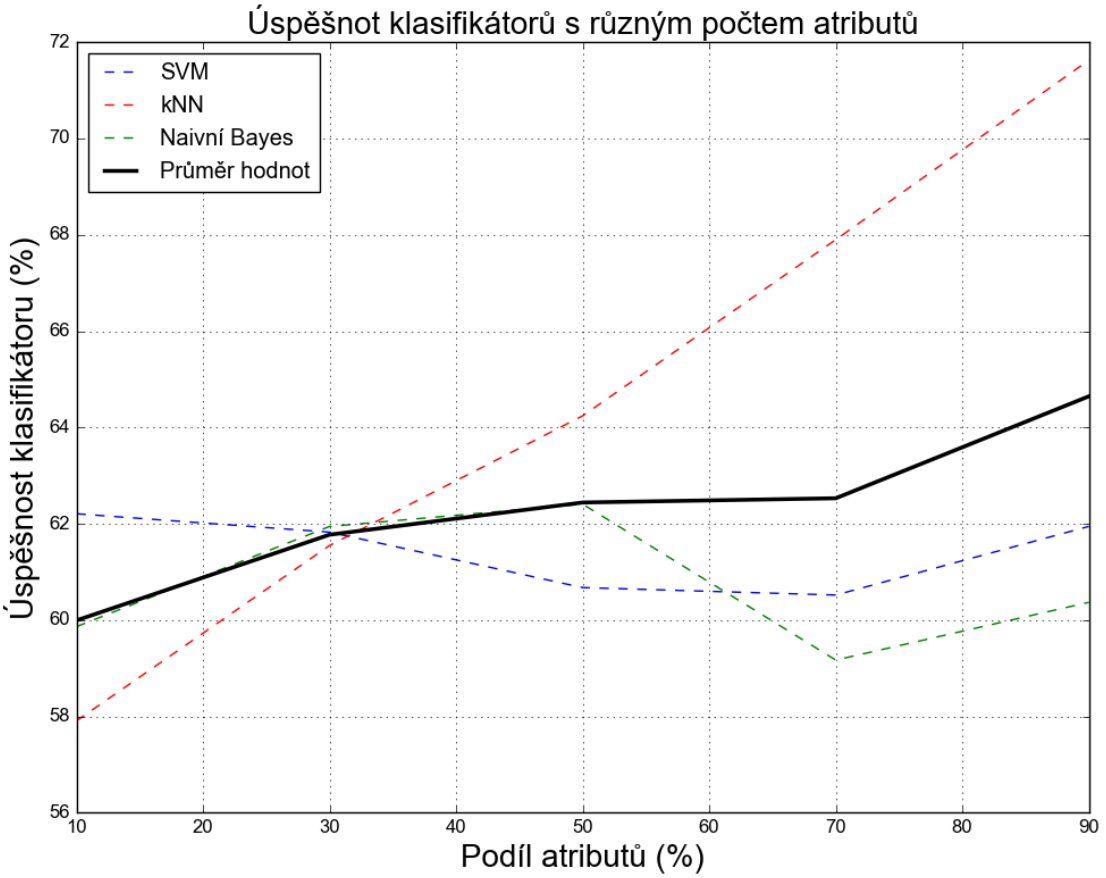
- Druhy aut: *Kategorické datové typy s menším počtem atributů*
- Šachová zakončení: *Vyšší počet instancí a tříd*
- Internetové reklamy: *Vyšší počet atributů s chybějícími hodnotami*
- Kvalita vína: *Průměrný počet atributů, tříd a instancí*
- Kategorie dokumentů: *Vyšší počet atributů s menším počtem instancí*

Výsledné procentuálně úspěšnosti klasifikátorů představují průměrné hodnoty z opakovaných měření na těchto datech.

Jako testovací metody jsou použity zástupci jednotlivých kategorií filtračních metod. První metodou je Spearmenův korelační faktor, který reprezentuje metody založené na korelaci. Dále je použit test Chi-kvadrátu, který patří do nezávislých statistických testů a také metrika Vzájemná informace, což je zástupce informačně-teoretických metrik.

Cílem je experimentálně zjistit, jakou hodnotu parametru je vhodné použít s ohledem na výkonost jednotlivých modelů, ale také časovou složitost. Následně pak bude parametr

použit v následujících sekcích, kde tento přístup hledání podmnožiny atributů porovnáváme s ostatními přístupy.

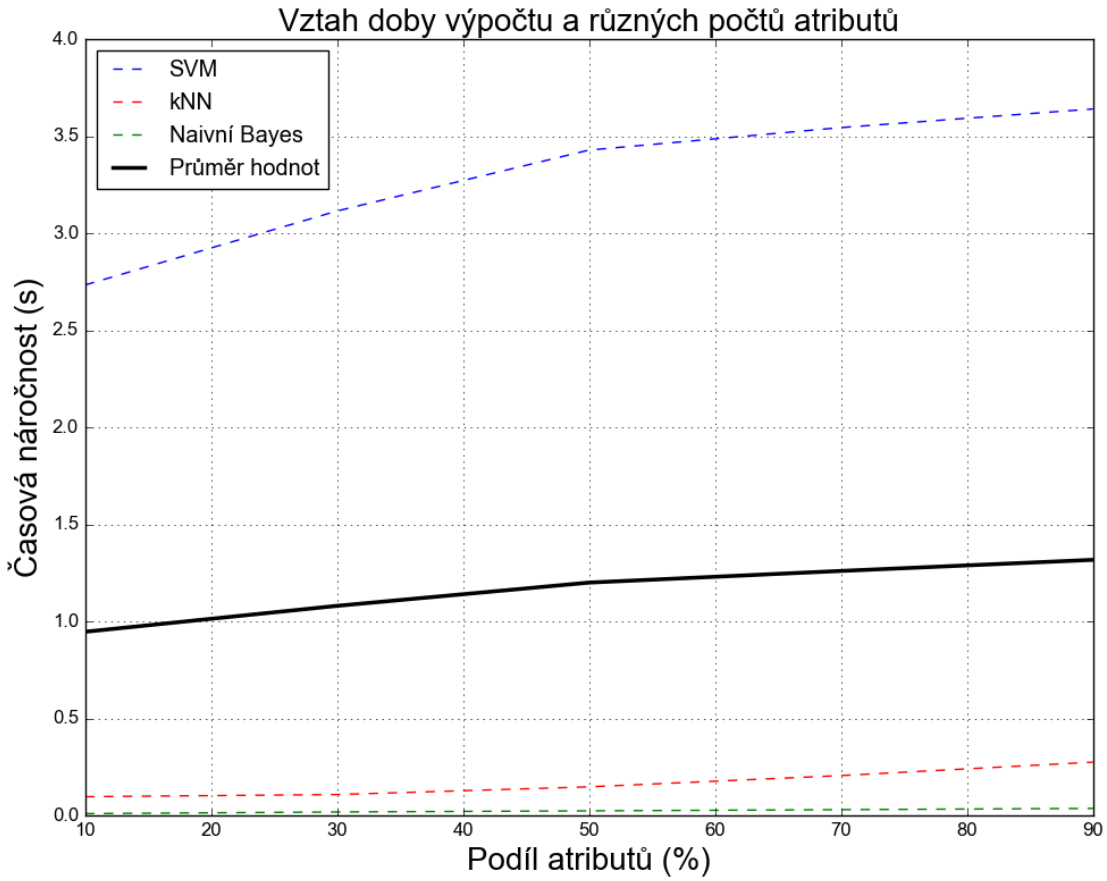


Obrázek 3.5: Úspěšnost klasifikátorů v závislosti na podílu atributů

Výstup ze srovnání je zobrazen na obrázku 3.5, kde je vidět výkonnost jednotlivých klasifikátoru v závislosti na počtu atributů. Například u kNN je vidět téměř lineární nárůst, což ale nemusí odpovídat skutečnosti, že více atributů zvyšuje přesnost pro tento model. U ostatních se úspěšnost snižuje s vyšším počtem atributů a průměrná úspěšnost se pohybuje kolem 62%. Pro měření se mi zdá jako vhodná hodnota z intervalu od 30-50%, kde dochází stále k nárůstu úspěšnosti a klasifikátory mají obdobné vlastnosti, což je důležité pro následné srovnávání metod. V tomto intervalu je i hodnota 50%, která je populární i intuitivní vzhledem k problému a přináší přijatelný kompromis mezi cenou a výkonem a udržuje klasifikátory ve srovnatelné výkonnosti.

Z časového hlediska na obrázku 3.6 lze vyčíst lehký nárůst časové složitosti s přibývajícími atributy, což není nic nečekaného. Tento nárůst je pro všechny klasifikátory téměř rovno-

měrný.

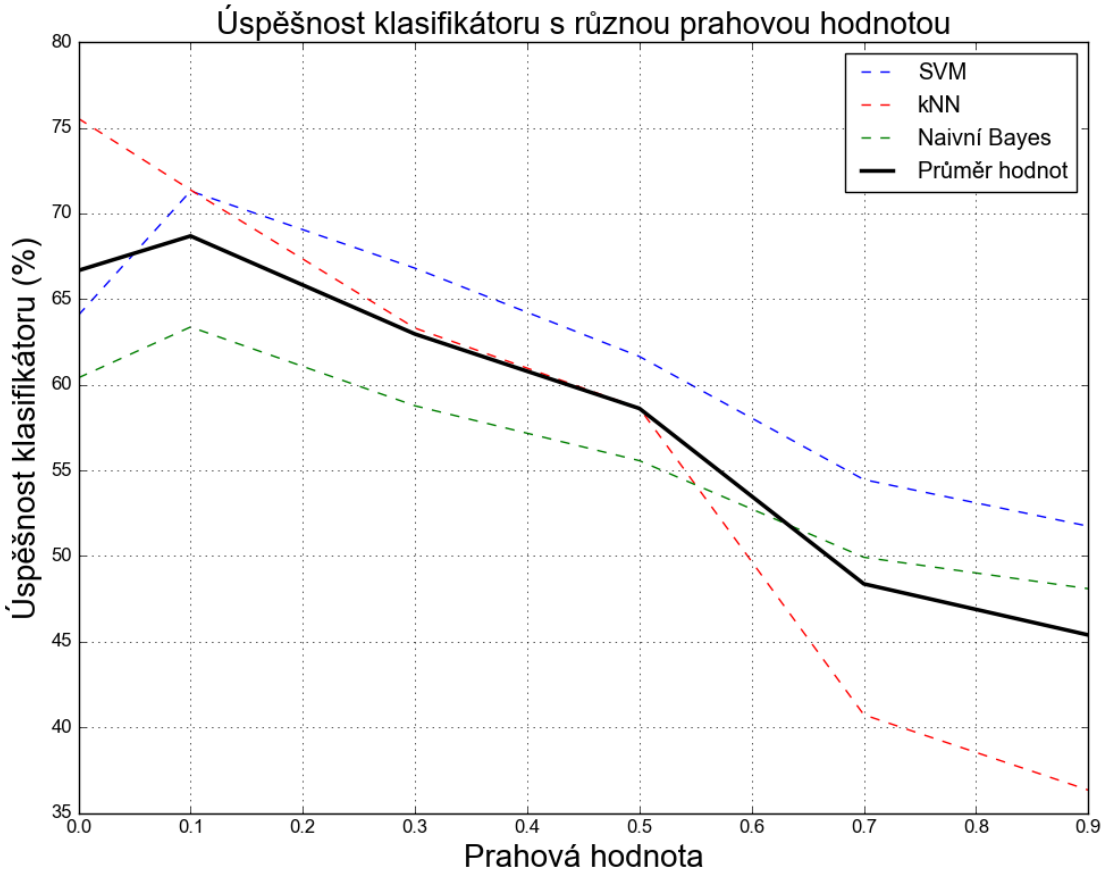


Obrázek 3.6: Časová složitost klasifikátorů v závislosti na podílu atributů

3.3.2 Hranice užitečnosti atributů

Hranicí užitečnosti rozumíme nějakou prahovou hodnotu, která určuje, zda atribut bude ve výsledné množině (vyskytuje se nad prahem), či zda bude vyloučen (užitečnost atributu je pod prahem). Nejprve je potřeba mít skóre všech metod normalizované tím způsobem, že nejužitečnější atribut má skóre 1 a zbývající jsou vypočítány poměrem. To aplikujeme i na korelační metody, jenž jsou normalizované bez zásahu, aby byly všechny metody nastavené jednotným způsobem. Tím docílíme jednak zobecněním prahu pro všechny metody, ale také zaručením, že vždy alespoň jeden atribut bude vybrán do výsledku.

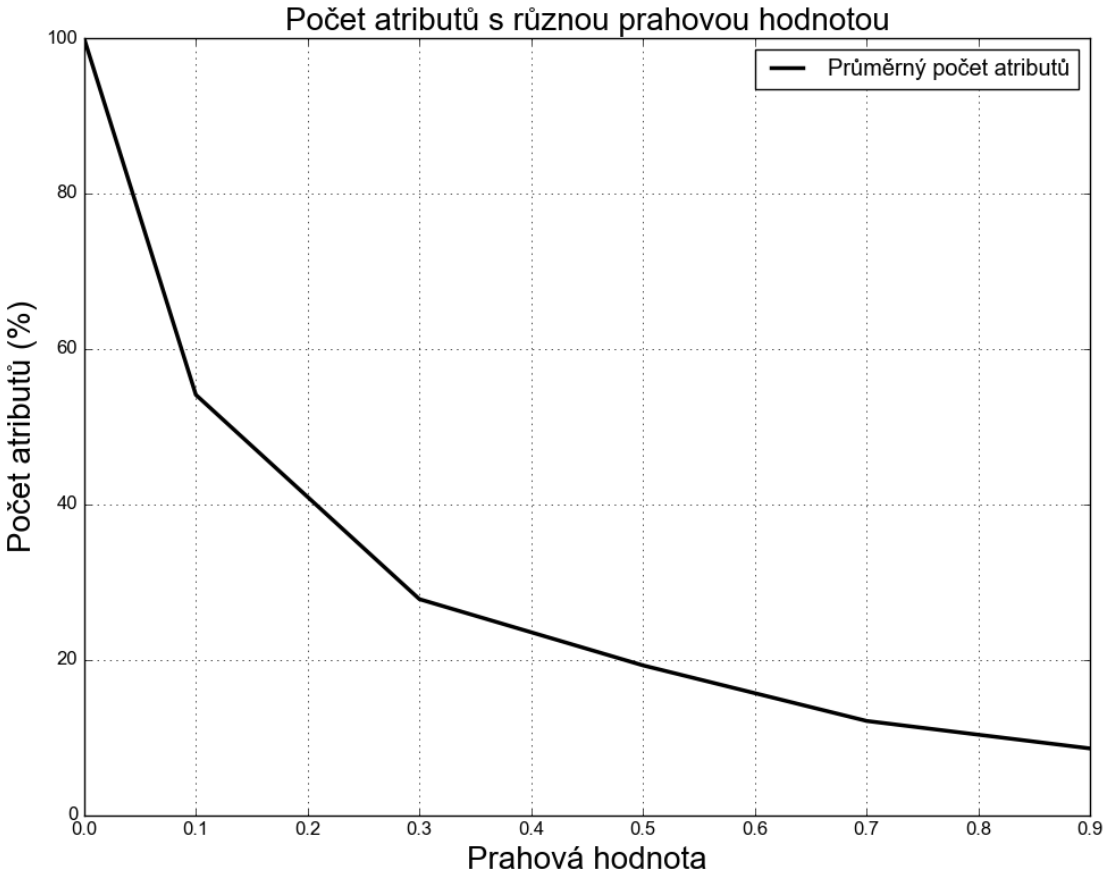
Experimentální měření probíhala na stejných datech se stejnými metodami, jako u hledání procentuálního zastoupení atributů. Z měření na grafu 3.7 vyšlo najevo, že dobrou



Obrázek 3.7: Úspěšnost klasifikátorů v závislosti na různém prahu

volbou nastavení prahu se zdá být hodnota 0,1, která obsahuje přibližně 50% atributů dle 3.8. Což odpovídá i předchozímu měření, kde jsme nastavovali poměr atributů zapojených do výsledné množiny také na hodnotu 50%.

Rozdíl nastává hlavně u dat, kde je velký rozdíl mezi počty atributů. Data s menším počet atributů se normalizují obvykle na hodnoty s větším průměrným skórem, tudíž je jich procentuálně zastoupeno více, než u souborů s více atributy.



Obrázek 3.8: Procentuální zastoupení atributů s různým prahem

3.4 Metriky založené na korelaci

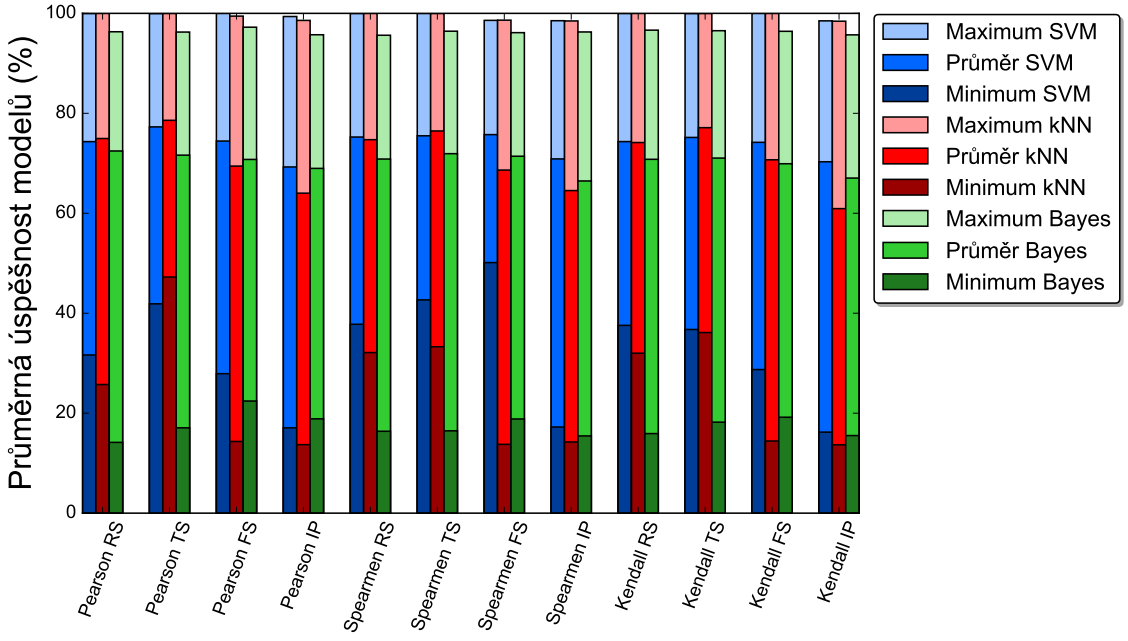
Mezi metriky založené na korelaci patří

- Pearsonův korelační koeficient
- Spearmanův korelační koeficient
- Kendallův korelační koeficient (tau-b verze)

Předpoklady na data

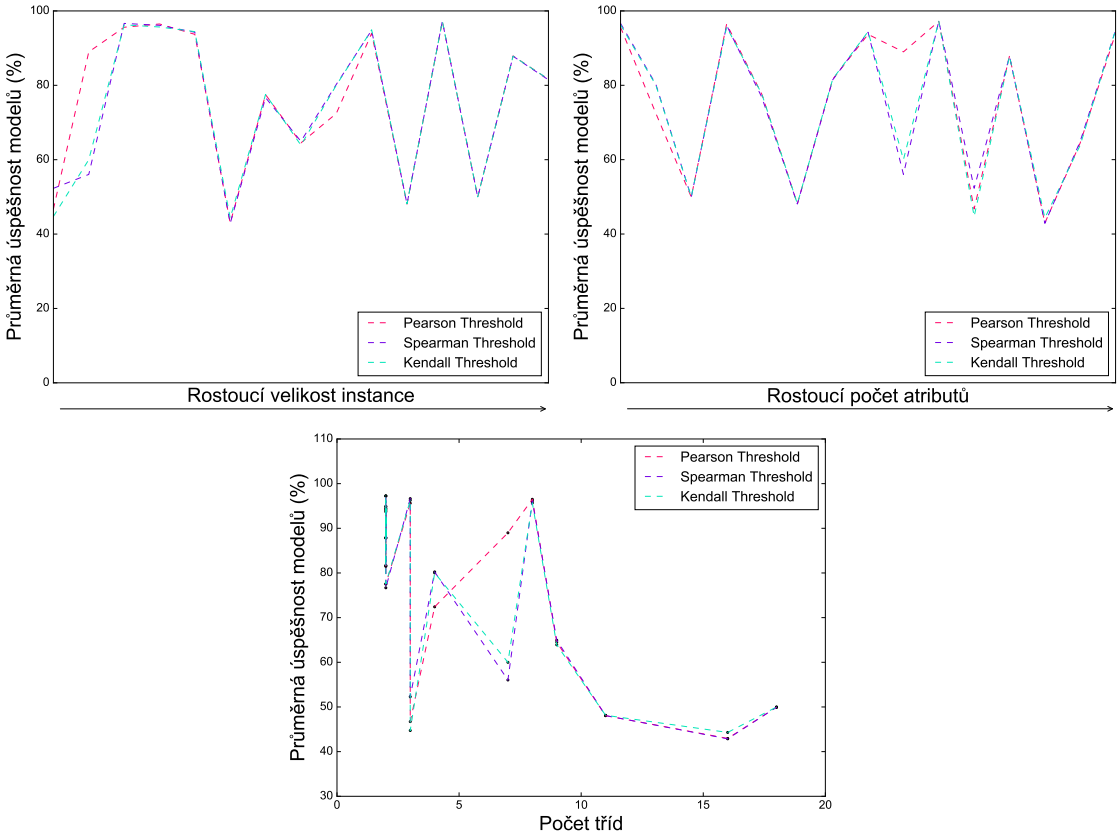
Požadavky na atributy / Metody	Rozdělení	Datový typ	Závislost
Pearsonův korelační koeficient	*normální*	*intervalový, poměrový*	*lineární*
Spearmanův korelační koeficient	*žádné*	*ordinální, intervalový, poměrový*	*monotónní*
Kendallův korelační koeficient	*žádné*	*ordinální, reálný*	*monotónní*

Výkonnost
Výkonnost měření probíhala porovnáním jednotlivých korelačních metod v kombinaci s různými způsoby výběru finální množiny atributů, což je procentuální zastoupení (PS), hranice užitečnosti (TS), dopředný výběr (FS) a výběr na základě inflexního bodu (IP).



Obrázek 3.9: Úspěšnost korelačních metod na jednotlivých klasifikátorech

Na obrázku 3.9 je vidět procentuálního úspěšnost jednotlivých klasifikátorů měřených na testovacích datech, na kterých proběhla různá selekce atributů. V tmavších barvách je znázorněn nejhorší výsledek dosažený klasifikátorem a světlou barvou naopak nejlepší výsledek. Průměrná hodnota je zobrazena mezi extrémy. Měření probíhala na všech 15 referenčních datových sadách. Minimum a maximum odpovídá tedy nejméně, respektive nejvíce vhodnému setu pro danou metodu.

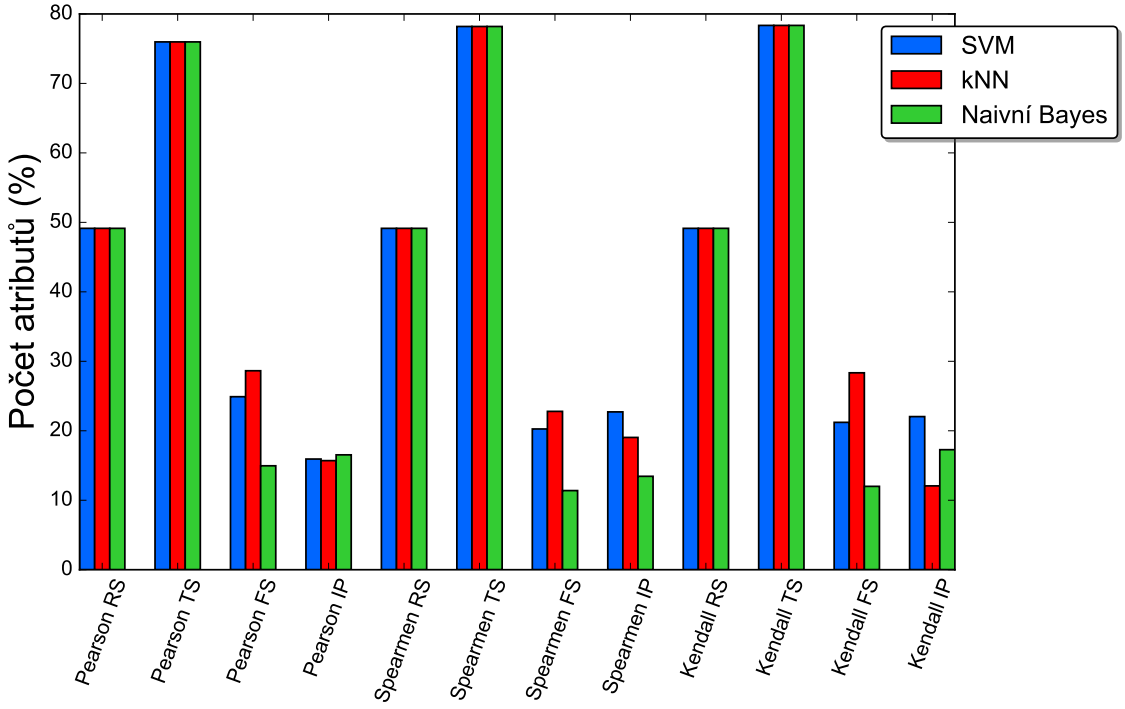


Obrázek 3.10: Porovnání závislostí korelačních metod

Můžeme sledovat, že vzhledem k typu metod jsou výsledky dost podobné, pouze s malou odchylkou v řádu jednotek procent. Nejhorších výsledků obvykle obsahovaly v kombinaci s modelem Naivního Bayese, který ve většině případů byl nejslabším článkem a dosahoval nejnižších hodnot. Nejlepších výsledků dosahovalo SVM, neboť v žádném datovém setu vyloženě nepropadlo a v nadpoloviční většině dokonce dosahoval i nejlepších průměrných výsledků. Nicméně časová dotace pro tento model je vyšší.

Prahová selekce se jeví jako nejúspěšnější z hlediska výkonnosti a oproti ostatním nabízí vyšší minima, než u ostatních. Obzvláště v kombinaci s Pearsonem se jeví její použití jako

příhodné, vzhledem k populárnosti této metody.



Obrázek 3.11: Procento vybraných atributů korelačními metodami

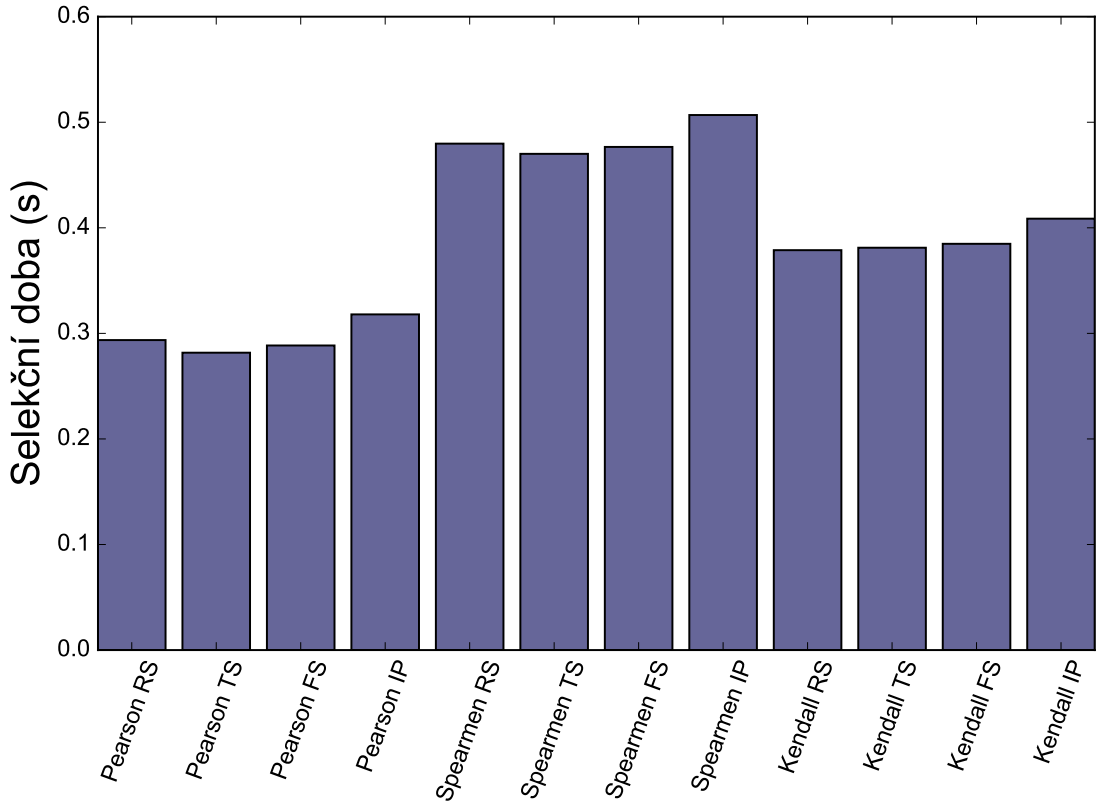
Právě prahová selekce vyšla v měření s nejlepší úspěšností. Její porovnání v kombinaci s korelačními metodami nabízí obrázek 3.10. Hodnoty na vodorovné ose (vyjma počtu tříd) jsou upravené, aby byl graf lépe přehledný, tudíž každý krok reprezentuje další datový set s vyšším počtem sledovaného parametru. Vztah mezi velikostí instance a úspěšností není zřejmý a není viditelná žádná souvislost mezi těmito parametry. To samé v případě počtu atributů. Ovšem v případě počtu tříd můžeme sledovat mírnou sestupnou tendenci, což je způsobeno větším počtem možností, ve kterých se může klasifikátor mýlit.

Počet atributů

Co se týče počtu atributů, které byly do klasifikace zapojeny, tak nejméně atributů sledujeme u filter-wrapperových metod, obzvláště u inflexního výběru, který u všech metod dosahoval také nejnižší úspěšnosti, obzvláště v kombinaci s kNN modelem. Nejvíce atributů zahrnul do výsledku prách užitečnosti, kde ač všechny klasifikátory dosahují vysokých výsledků, výsledná podmnožina rozhodně nepatří mezi minimální a blíží se k 80% zahrnutých příznaků. Celé procentuální zastoupení atributů je znázorněno v grafu 3.11.

Časová náročnost

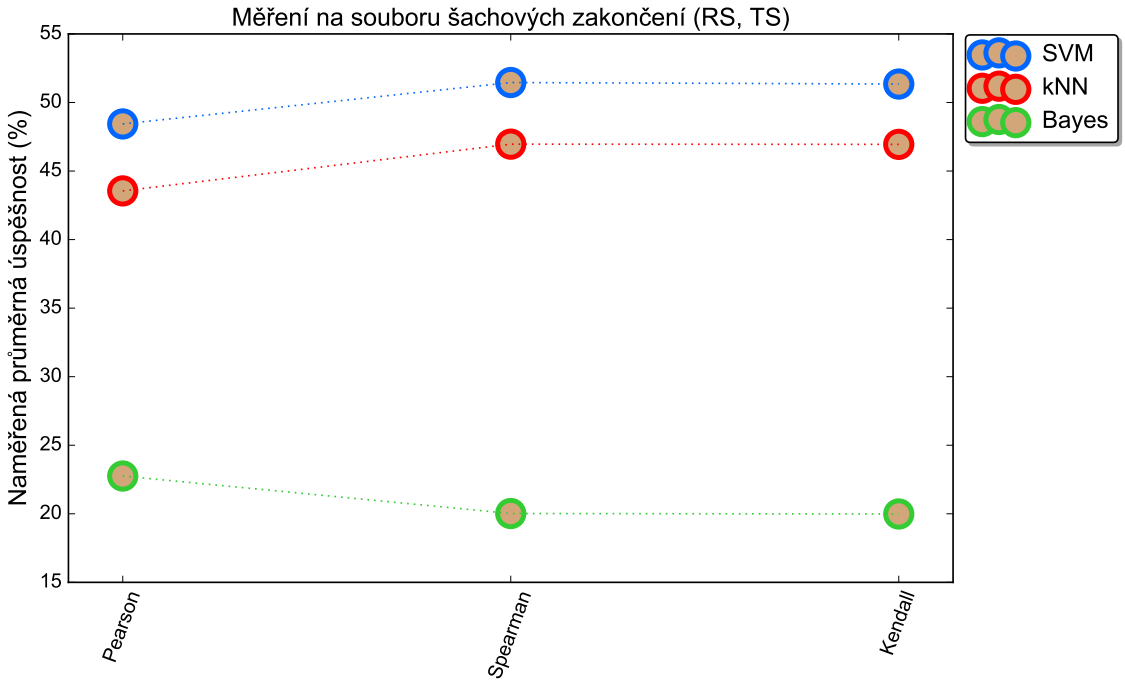
Další faktor, který by se měl brát v úvahu, je samotná doba trvání vyhodnocovacího pro-



Obrázek 3.12: Výpočetní čas jednotlivých korelačních metod

cesu. Výpočetní čas jednotlivých metod je znázorněno na 3.12. Zde nezáleží, o jaký způsob hledání podmnožiny atributů se jedná, neboť ohodnocení je pro stejné metody společné. Naměřené rozdíly jsou tedy způsobeny odchylkou měření. Nejméně náročný na ohodnocení atributů je hledání závislostí na základě Pearsonova korelačního koeficientu. Další v pořadí je Kendall implementovaný ve verzi s časovou složitostí $\mathcal{O}(n \log n)$) a nárůstem přibližně o třetinu a v testu jako nejnáročnější vyšel Spearmenův korelační koeficient se složitostí také $\mathcal{O}(n \log n)$, který však potřeboval v průměru o další třetinu více času než Kendall. Vzhledem k tomu, že Pearson měl dobrou i úspěšnost klasifikátoru (respektive nebyl zaznamenán žádný výrazný rozdíl), lze se přiklonit k této metodě při absenci analýzy vlastností datového setu.

Doporučení
Výhodou metod založených na korelaci je právě jejich jednoduchost a snadnost použití. Není potřeba nastavovat žádné parametry a jejich výpočet je relativně rychlý.



Obrázek 3.13: Porovnání korelačních metod na datovém souboru šachových zakončení

Metody byly úspěšné především u méně komplexních datových souborů, špatných výsledků naopak dosahovali data s větším množství tříd či s nelineárními a nemonotónními vztahy. Způsoby selekce zahrnující zpětnou vazbu dosahují přinejlepším stejných výsledků za delší časový úsek, a u dat s více atributy (Internetové reklamy) nejčastěji v kombinaci s kNN úplně propadnou s úspěšností menší než 20%.

Je vhodné tedy použít nějaký z jednodušších způsobů výběru atributů jako třeba procentuální zastoupení atributů, případně výběr na základě prahové hodnoty, které je pouze lehce náročnější na výpočetní čas, avšak dosahuje dobrých výsledků a pokud nemáme žádné informace o použitých datech, tak použít všechny tyto metody (nebo alespoň Perason a Spearmena), který nám pomohou odhalit, zda jsou vazby mezi atributy lineární, či monotónní a jejich výpočet příliš nestojí.

Rozdíl mez metodami je znázorněn například na datovém souboru *šachových zakončení* 3.13, jež obsahuje velký počet instancí a tříd. Uvažujeme zde pouze poměrovou a prahovou selekci a můžeme pozorovat nárůst úspěšnosti u Spearmanového a Kendallova koeficientu oproti Pearsonovi, a to v případě ohodnocení kNN či SVM klasifikátorem. To je způsobené skutečností, že popsání závislostí těchto dat lze zachytit lépe monotónními vztahy, než-li lineárními.

Bayesův klasifikátor, který se hodí pro malá data, dosahuje menších úspěšností v případě většiny datových souborů, respektive přínos časově náročnějších korelačních metod je téměř nulový. Hodí se tedy v kombinaci s Pearsonovým korelačním koeficientem.

3.5 Metriky založené na statistickém ohodnocení

Mezi implementované metriky, které jsou založené na statistickém ohodnocení patří

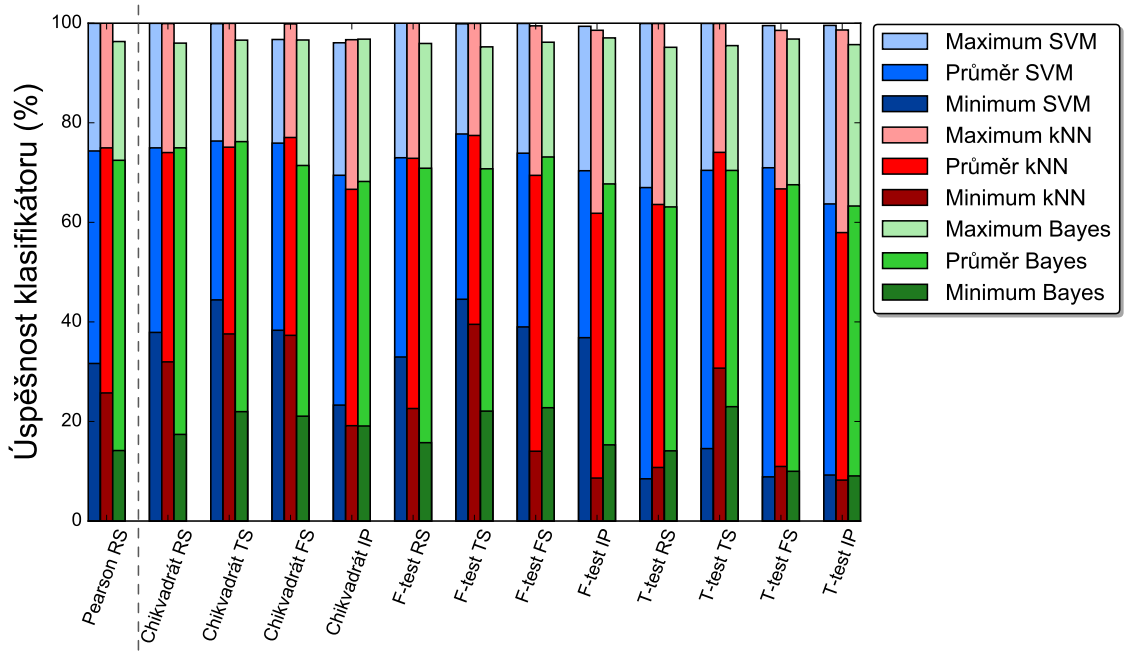
- Chi-kvadrát test
- F-test (verze ANOVA)
- T-test (verze studentův T-test)

Předpoklady na data

Požadavky na atributy / Metody	Rozdělení	Vztah mezi třídami	Velikost instance
Chi-kvadrát	*normální*	*nezávislé*	*větší*
F-test	*normální*	*nezávislé*	*neomezeno*
T-test	*normální*	*nezávislé*	*neomezeno*

Chi-kvadrát test také nelze provést na záporné atributy, tudíž nebyl test proveden na datovém setu Srdeční arytmie. T-test také předpokládá shodnost rozptylů dat.

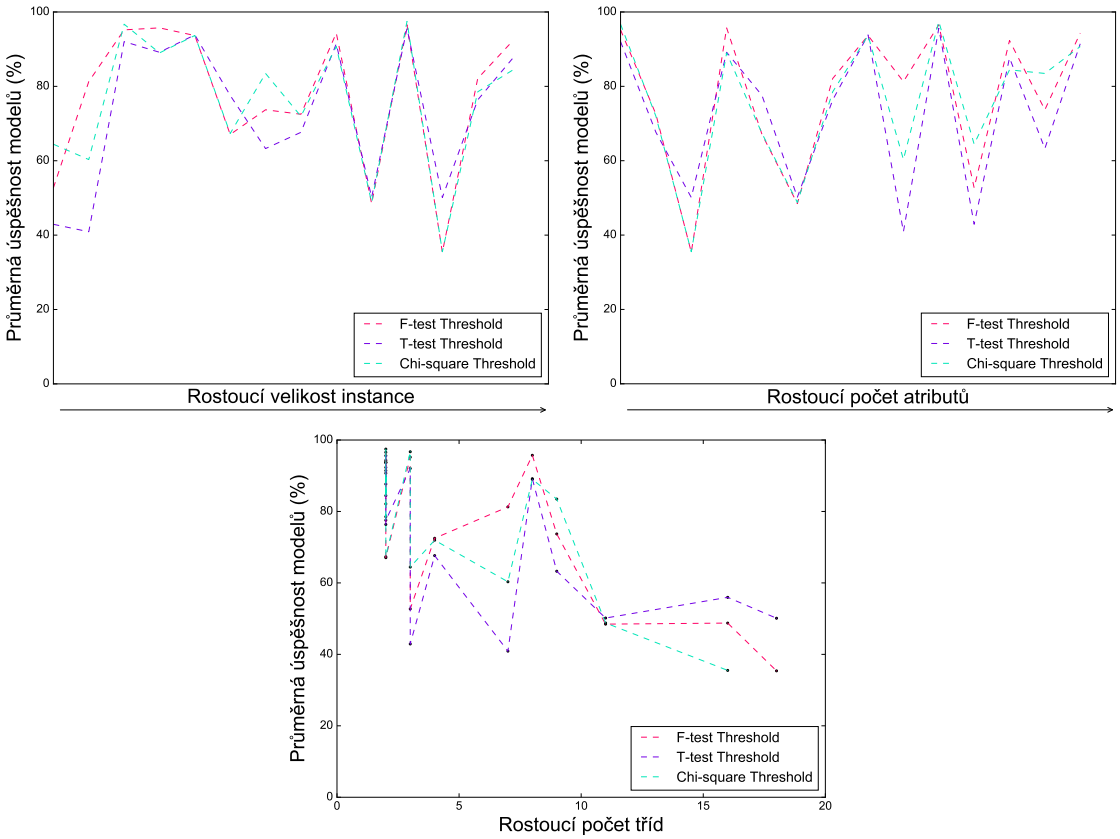
Naměřenou úspěšnost porovnáme i s *Pearsonovým korelačním koeficientem* (oddělené svislou čárou) s poměrovým způsobem vybírání atributů.



Obrázek 3.14: Úspěšnost statistických metod na jednotlivých klasifikátorech

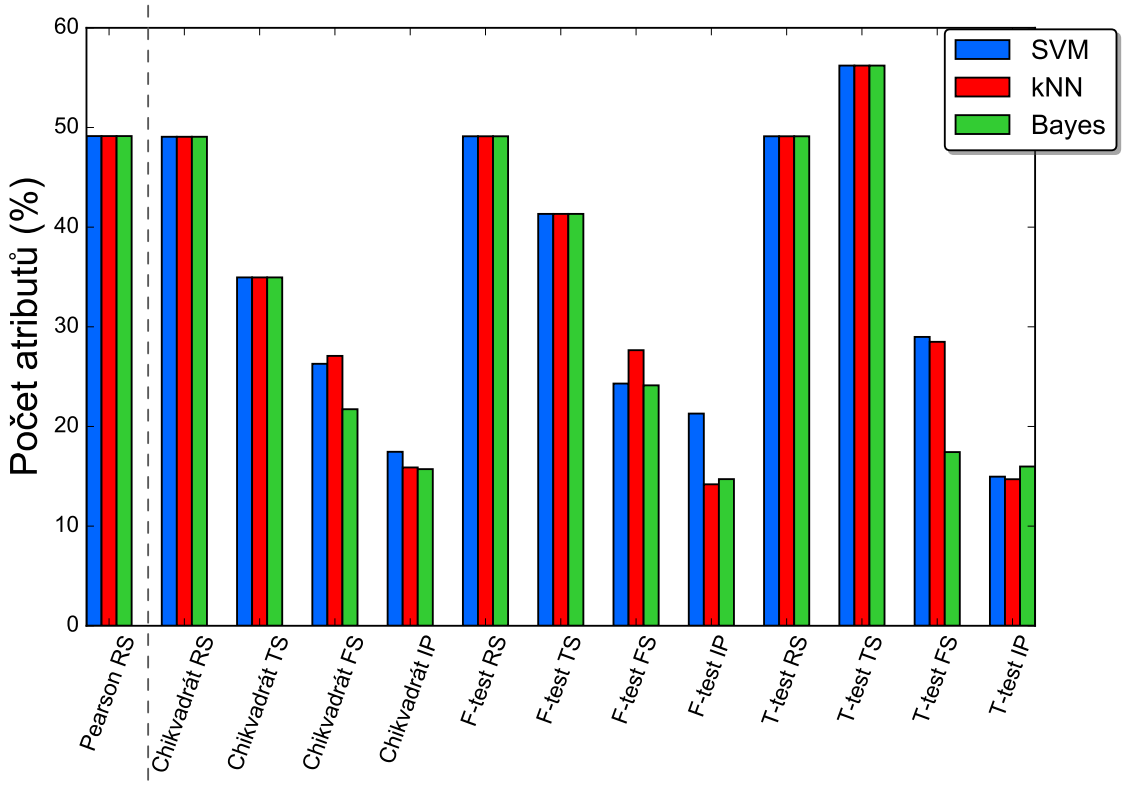
Výkonnost
Z porovnání na 3.17 lze vyčíst, že v rámci metod dopadli obstojně všechny verze Chí-kvadrátu, který dosáhl více jak 70% ve většině způsobů selekce atributů napříč klasifikátory. Není viditelný velký rozdíl mezi způsoby RS, TS a FS. Ačkoli obdobně v jako minulém testu, IP dosáhl nejhorších výsledků, nehledě na metodu.

Celkově nejlepšího skóre dosáhla statistika F-test s více jak 77% úspěšností jak pro SVM, tak pro kNN. Klasifikátor Naivní Bayes dosahoval avšak lepších výsledku zpravidla v kombinaci právě s Chí-kvadrátem. Můžeme si všimnout, že způsoby selekce obsahující zpětnou vazbu od klasifikátoru dosahují ještě výraznějších minim (obzvláště u F-testu a T-testu), než je tomu u korelačních metod, což je způsobeno jejich hladovým přístupem v kombinaci s normalizací skóre těchto metod.



Obrázek 3.15: Porovnání závislostí statistických metod

V porovnání úspěšnosti mezi 3.15 můžeme pozorovat menší výchylku Chí-kvadrátu ve výkonnosti, tedy patří mezi stabilní pro různá data. Obdobně jako u korelačních metod, i zde pozorujeme závislost v počtu tříd na výslednou úspěšnost modelu.

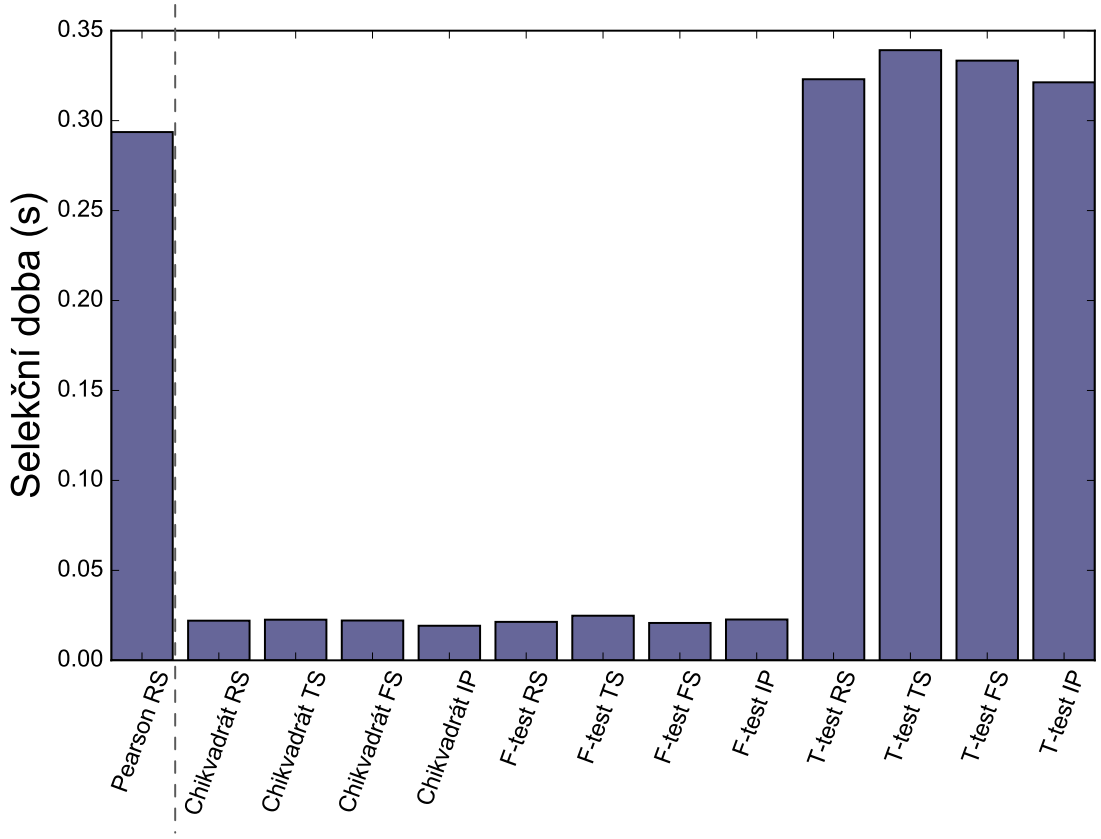


Obrázek 3.16: Procento vybraných atributů statistickými metodami

Počet atributů
Počet atributů 3.16 je výrazně nižší u prahové selekce, než je tomu u korelačních metod vzhledem k odlišnému rozdělení skóre. Například v případě Chí-kvadrátu je prahem vyselektováno přibližně 35% atributů, avšak výkonnost je stále jedna z nejlepších. Oproti tomu u korelačních metod to bylo kolem 75%, což má vliv na klasifikační čas.

Časová náročnost
Dobu ohodnocení jednotlivých atributů můžeme vidět na 3.17. Metody Chí-kvadrát a F-test jsou časově velmi nenáročné, což z nich dělá velmi flexibilní metody. Časová složitost Chí-kvadrátu je $\mathcal{O}(mn)$, kde m je počet tříd a n je celkový počet atributů. Avšak v naměřených výsledcích vychází k velmi nízké průměrné hodnotě. T-testu trvá ohodnotit atributy lehce déle, než v případě Pearsona, ale v obdobném měřítku.

Doporučení
Statistické metody nabízejí co se týče výkonnosti obdobné výhody jako metody korelační. Jejich síla tkví v jejich statistických vlastnostech, ať je to metrika průměru v případě T-testu, rozptylu v případě F-testu či vztah mezi atributem a třídou u Chí-kvadrátu,

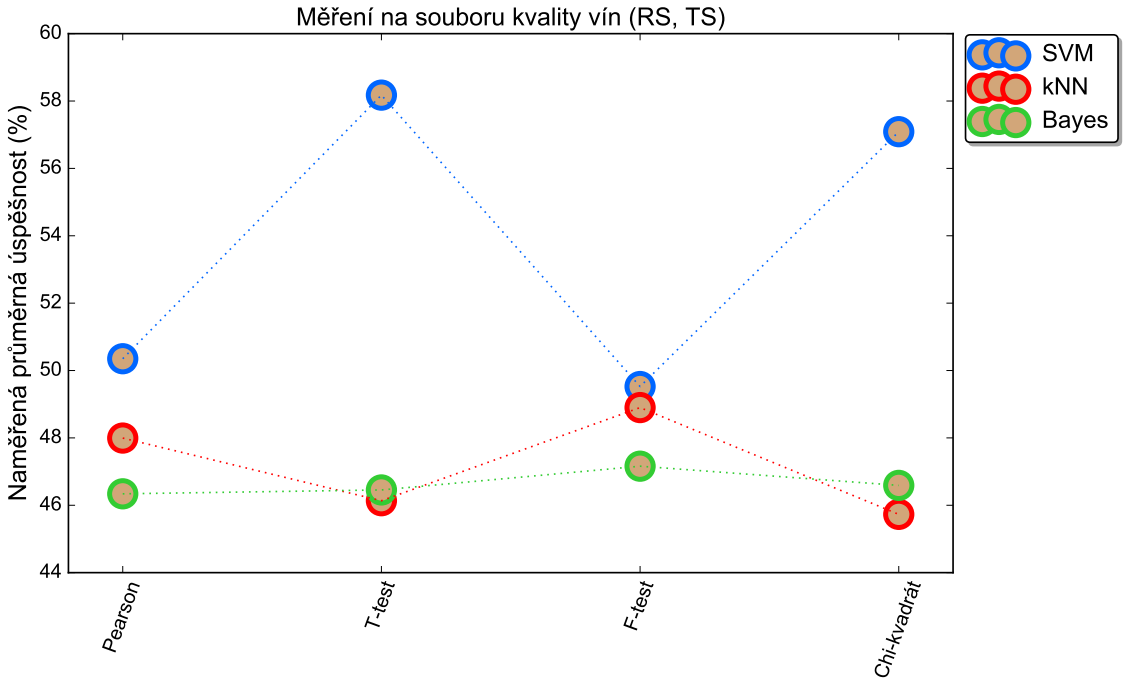


Obrázek 3.17: Výpočetní čas jednotlivých statistických metod

což způsobí odlišná ohodnocení pro jednotlivé metody znázorněné na 3.18 a je potřeba přizpůsobit statistickou vlastnost danému datovému souboru.

Mezi omezení Chí-kvadrát testu patří nemožnost provést hodnocení na záporných atributech, z tohoto důvodu nebyl test proveden na všech souborech, nicméně silnou stránkou je v univerzálnosti použití s různými klasifikačními modely a také jejich dobrá úspěšnost na různé škále dat a patří tedy mezi základní metody k ověření vztahu proměnných. Na základě jeho skóre se lze dozvědět podstatné informace o zpracovaných datech, které se mohou hodit pro další zpracování.

V dalším porovnání, T-test předpokládá shodu rozptylů populací a porovnává jejich průměry, zatímco F-test porovnává rozptyly. Koncepčně je však F-test velmi podobný většímu počtu dvou-výběrových T-testů. Z toho důvodu je F-test výhodnější pro větší škálu problémů a i výpočetní doba je kratší.



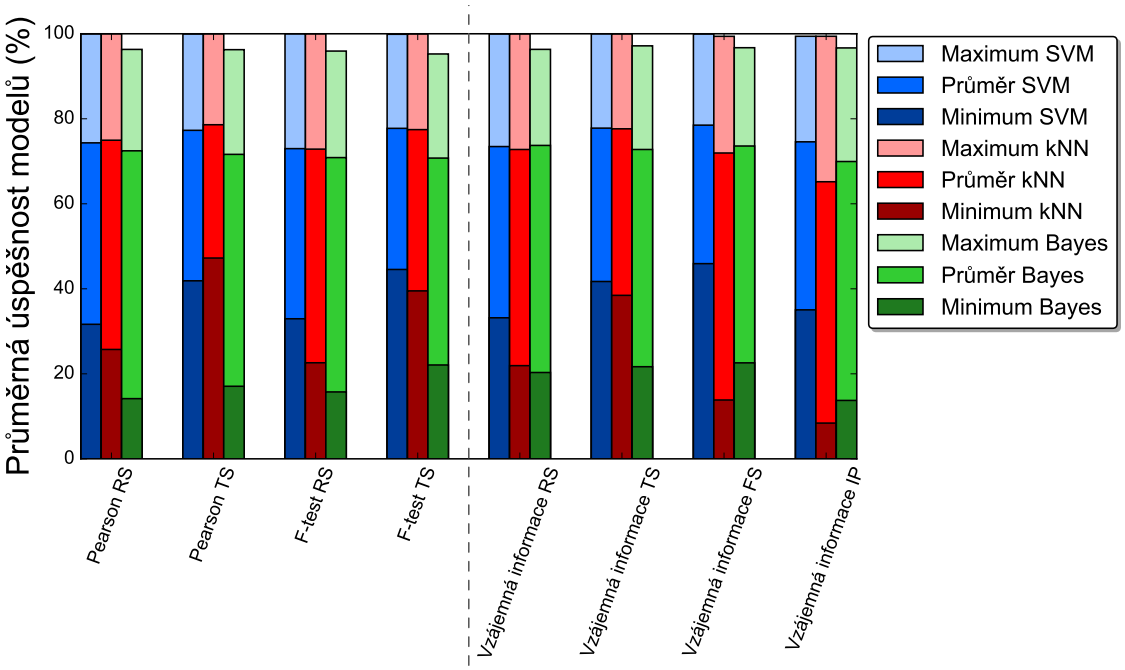
Obrázek 3.18: Porovnání metod na datovém souboru kvality vína

3.6 Metrika založená na entropii

Implementovanou metodou, založenou na entropii je

- Vzájemná informace

Naměřenou úspěšnost srovnáme i s předchozími měřeními - *Pearsonovým korelačním koeficientem* a *ANOVA F-testem* (oddělené svislou čárou) s poměrovým a prahovým způsobem vybírání atributů.

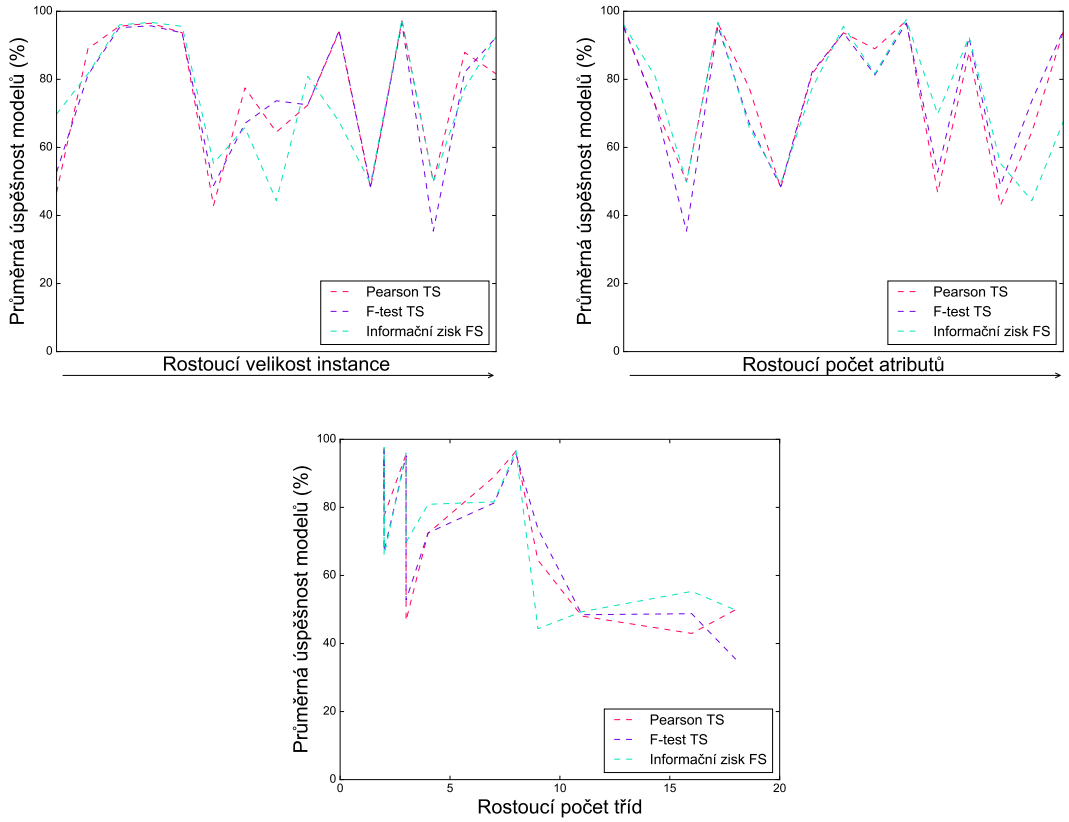


Obrázek 3.19: Úspěšnost metrik založených na entropii na jednotlivých klasifikátorech

Výkonnost
Výkonnost na 3.19 ukazuje na vysokou úspěšnost SVM klasifikátoru, který má nejlepší úspěšnost průměrnou, maximální i minimální v rámci metod Vzájemné informace. Nedobrých výsledků získáme v kombinaci s kNN, jehož minimum dosahuje nižších hodnot, než u korelačních či statistických metod a zaostává vůči SVM. V případě inflexního výběru je to dokonce pouze něco málo přes 8% (data šachových zakončení), což je soubor dat, kde SVM dosahuje úspěšnosti 65%. Naivní Bayes dosahuje výsledků obdobných bez větších výchylek.

Wrapperové metody v kombinaci s Vzájemnou informací nedosahují nižší výkonnosti, jako je tomu u korelačním a statistických metod, což zvyšuje jejich použitelnost právě s touto metodou. Dokonce v kombinaci s dopřednou selekcí a SVM klasifikátorem dosahovala me-

toda nejlepšího skóre.



Obrázek 3.20: Porovnání závislostí metod založených na entropii

Zobrazení vzájemných vztahů je na 3.20 a metoda Vzájemné informace se v tomto kotextu příliš neliší od ostatních porovnávaných metod.

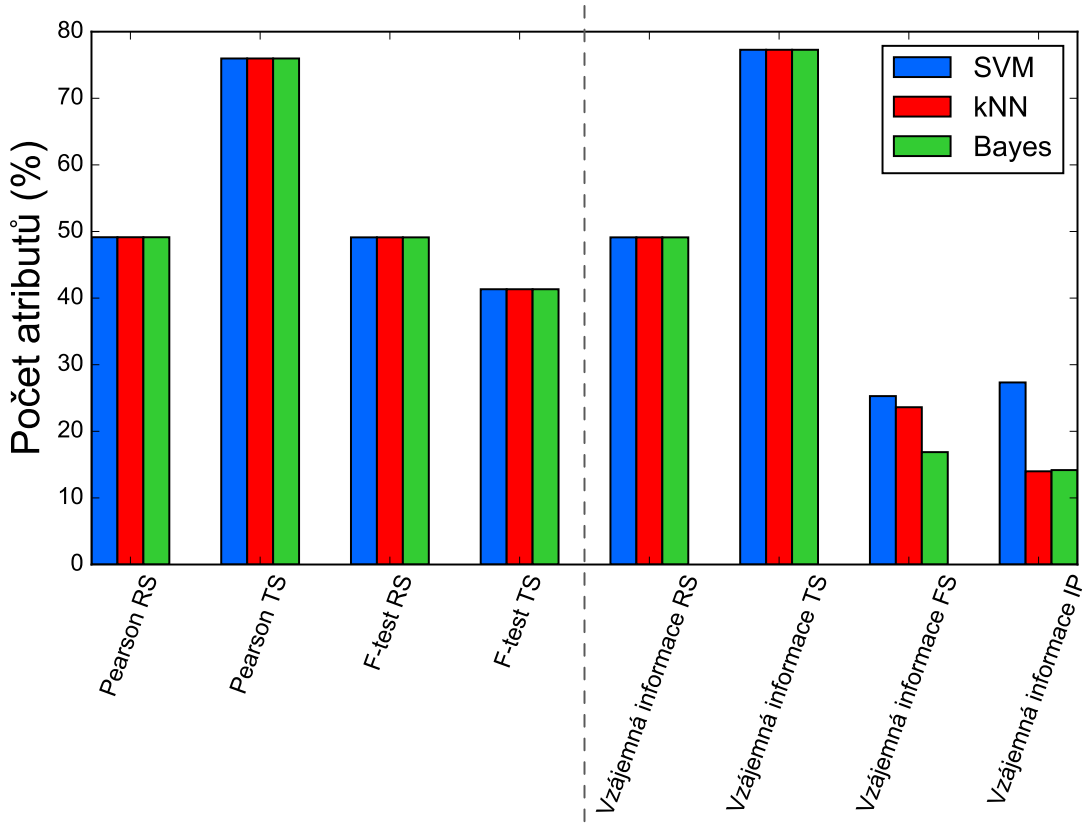
Počet atributů

Počet atributů 3.21 je výrazně nižší opět u wrapperových metod (FS, IP), avšak skóre dosahují obdobného, jako ostatní. U prahové selekce dochází k opět nárůstu počtu atributů na téměř 80%, což je téměř dvojnásobek oproti F-testu.

U inflexního výběru Vzájemné informace je zajímavé, že počet atributů je výrazně vyšší (téměř dvojnásobný) pro klasifikátor SVM, než pro zbylé dva, avšak podíl zastoupení atributů je stále menší jak 30%.

Časová náročnost

Vzájemná informace patří mezi časově náročnější filtrační metody, což je znázorněné

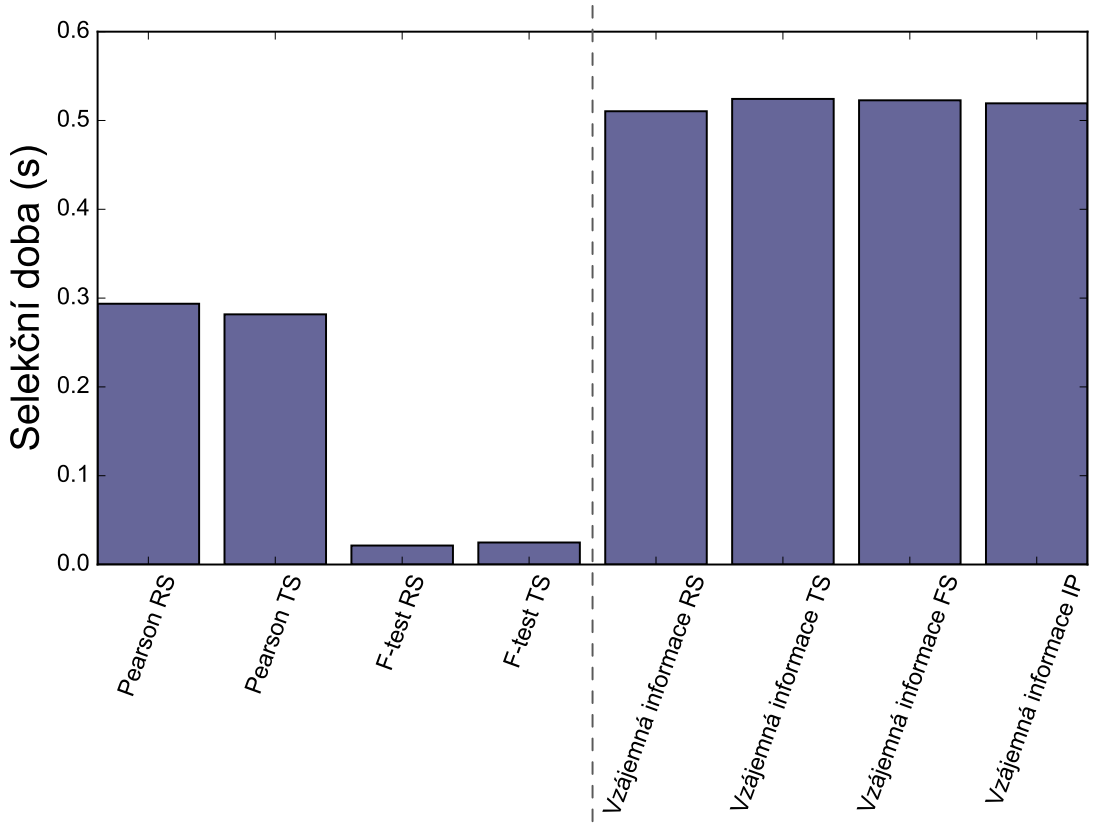


Obrázek 3.21: Procento vybraných atributů metodami založených na entropii

na 3.22. Obzvláště s porovnáním s F-testem, jehož vyhodnocení patří mezi nejrychlejší.

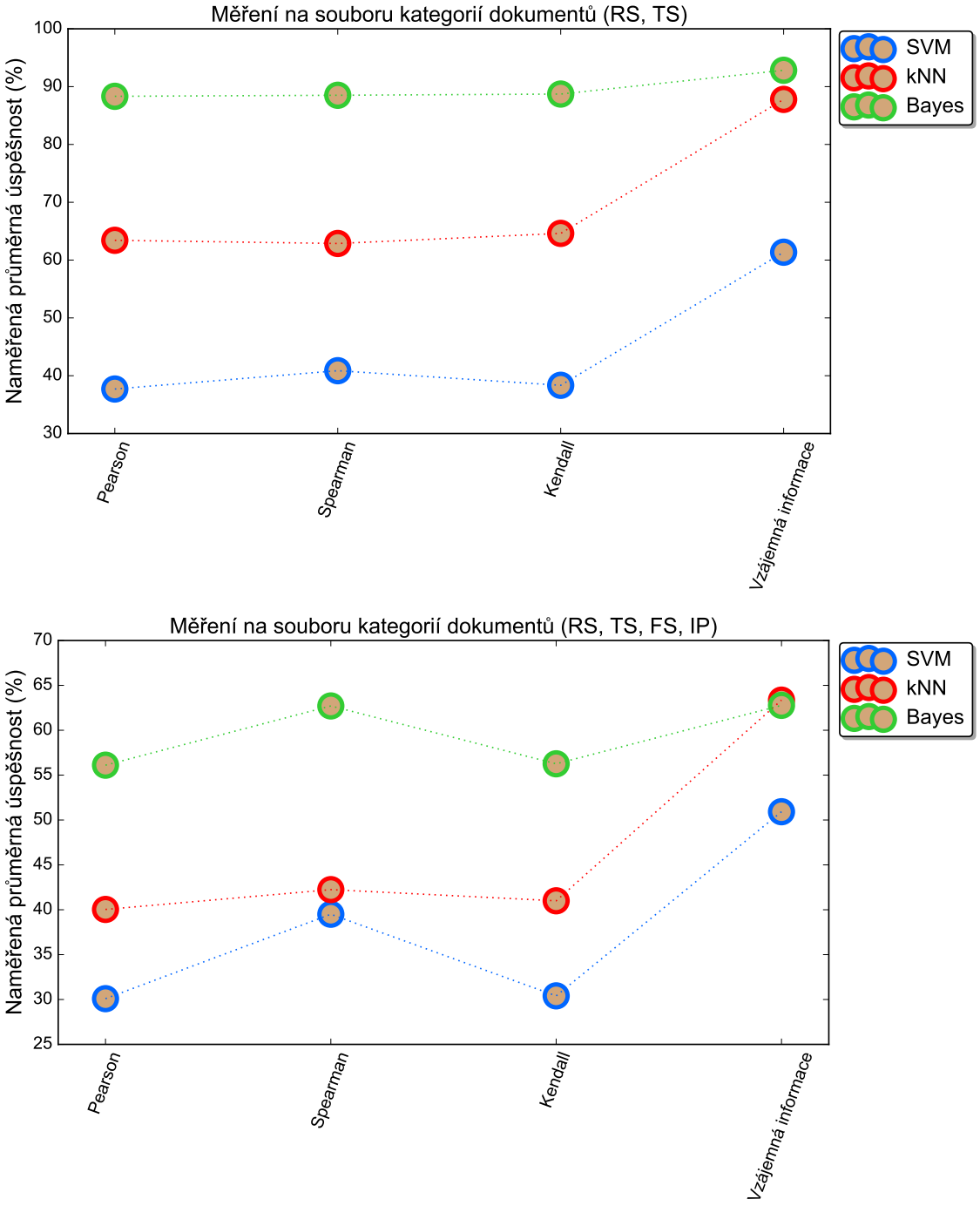
Doporučení
Informační zisk představuje dobrý způsob měření důležitosti atributů a přináší větší robustnost pro nelineární vztahy, neboť umožňuje měřit všechny vztahy mezi atributem a třídou, ne pouze lineární. Musíme však počítat s větší časovou dotací pro výpočet. Na obrázku 3.23 je znázorněno zlepšení výkonu při použití této metody na datovém souboru *kategorií dokumentů*, což jsou data s vetším počtem atributů a tříd, kde obzvláště SVM v kombinaci s korelačními metodami odvedlo špatnou práci. V porovnání s korelačními metodami dochází k signifikantnímu nárůstu úspěšnosti u SVM a kNN, což můžeme považovat za indikátor odlišného přístupu hledání vztahů mezi atributy a třídami. Obdobně by se dal nalézt i datový soubor s horšími výsledky.

Ve spojení s Bayesovým klasifikátorem, při kterém dosahoval obstojných výsledků, se výpočetní čas sníží (oproti SVM) a následné vybírání finální podmnožiny atributů nabízí velkou variabilitu a lze kombinovat i s wrapperovými způsoby výběru, které jsou daleko



Obrázek 3.22: Výpočetní čas jednotlivých metod založených na entropii

více použitelné právě v kombinaci s touto metodou.



Obrázek 3.23: Porovnání metod na datovém souboru kategorií dokumentů

3.7 Genetický algoritmus

Genetický algoritmus obsahuje několik nastavitelných parametrů ovlivňující jeho výkonnost a časovou náročnost. Při měření jsme zafixovali hodnoty těchto parametrů na doporučené či standardní hodnotě a se zbyvajícím hýbali a pozorovali změny ve výsledcích. Tímto způsobem sice neodhalíme vzájemné závislosti v nastavení, avšak umožní nám ucelený pohled na parametr a zjištění jeho vlastností.

K měření algoritmu byly vybráno 5 souborů testovacích dat, stejných jako při měření prahu či poměru atributů. Jedná se o data s různým počtem atributů a instancí, na což je genetický algoritmus velmi citlivý, ale vzhledem ke snaze najít nějaké obecné nastavení, tyto hodnoty zprůměrujeme a budeme se snažit najít nastavení s obstojným výsledkem pro nejrůznější typy dat.

Pravděpodobnost křížení

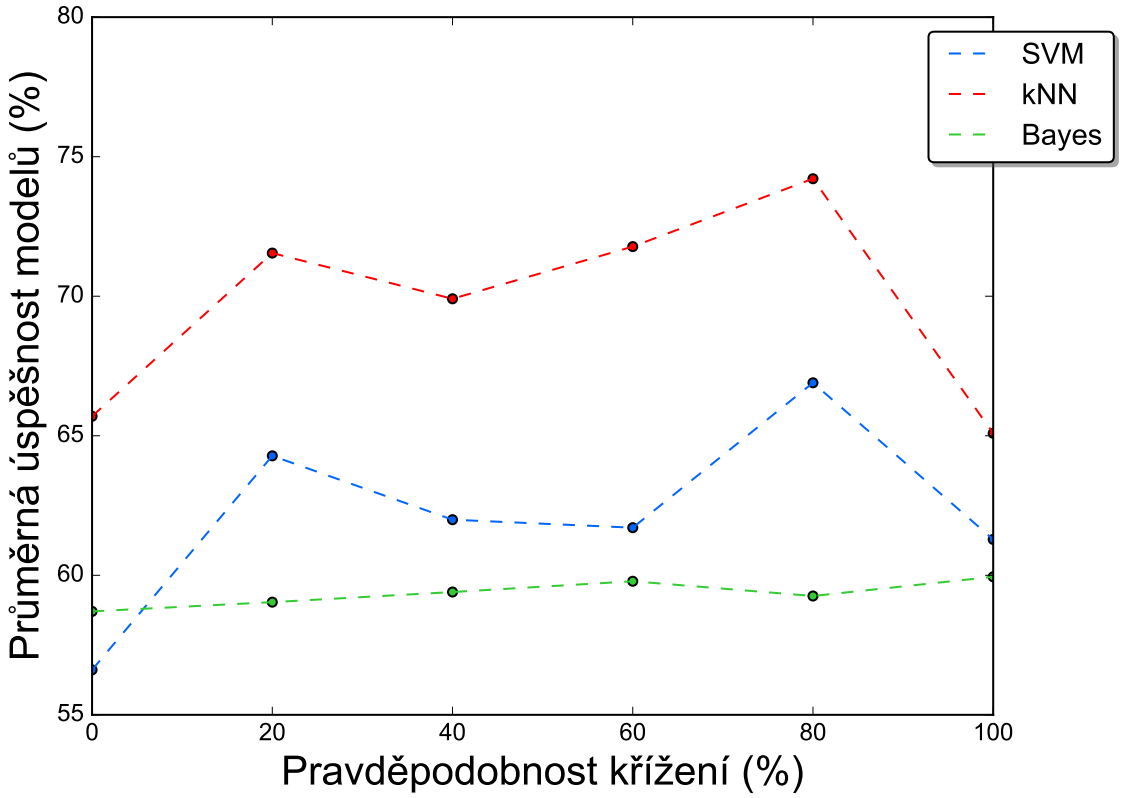
Pravděpodobnost křížení	***0 - 100%***
Pravděpodobnost mutace	*1%*
Velikost populace	*10*
Počet generací	*10*

Křížení je implementováváno **dvoubodové**, tedy náhodně se vyberou dva body v chromozomu rodičů (rozdělí se na tři části) a nakombinují se mezi sebou, čímž vzniknou dva potomci. Následně se při selekce, která je implementovaná ruletovým výběrem, vybere množství jedinců odpovídající definované velikosti populace. Zbytek jedinců selekcí neprošlo a jejich řešení zde zaniká.

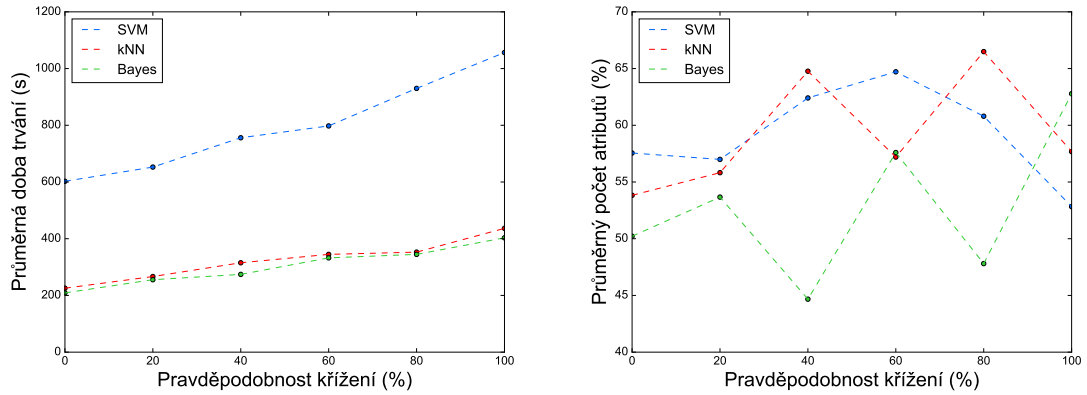
Při změně hodnoty pravděpodobnosti selekce se přechází od varianty, kdy se populace neobměňuje vůbec (0%), tedy veškeré změny v chromozomech musela obstarat mutace, která má však nízkou pravděpodobnost. Naopak při 100% pravděpodobnosti selekce měl každý z rodičů potomka a selekční tlak je vyšší.

Z grafu 3.24 je vidět růst úspěšnosti s rostoucí pravděpodobností křížení s maximem někde kolem 80%, což spadá i do rozmezí doporučených hodnot. Křížení by mělo být v algoritmus časté, aby docházelo co nejvíce k tvorbě nových řešení, které je na počátku vygenerované náhodně.

Co se týče časové závislosti na 3.25, tak dochází k nárůstu výpočetní doby s přibývající pravděpodobností křížení, což je logické, neboť dochází častěji k vytváření nových potomků. Ač není tento proces vytváření nijak složitý, rozdíl znatelný je. Avšak tento nárůst není nijak velký, proto se nemusíme obávat zvolit vyšší pravděpodobnost a pro další měření použijeme **pravděpodobnost křížení 80%** jako výchozí hodnotu.



Obrázek 3.24: Úspěšnost klasifikátorů s měnící se pravděpodobností křížení



Obrázek 3.25: Doba trvání a procentuální počet atributů křížení

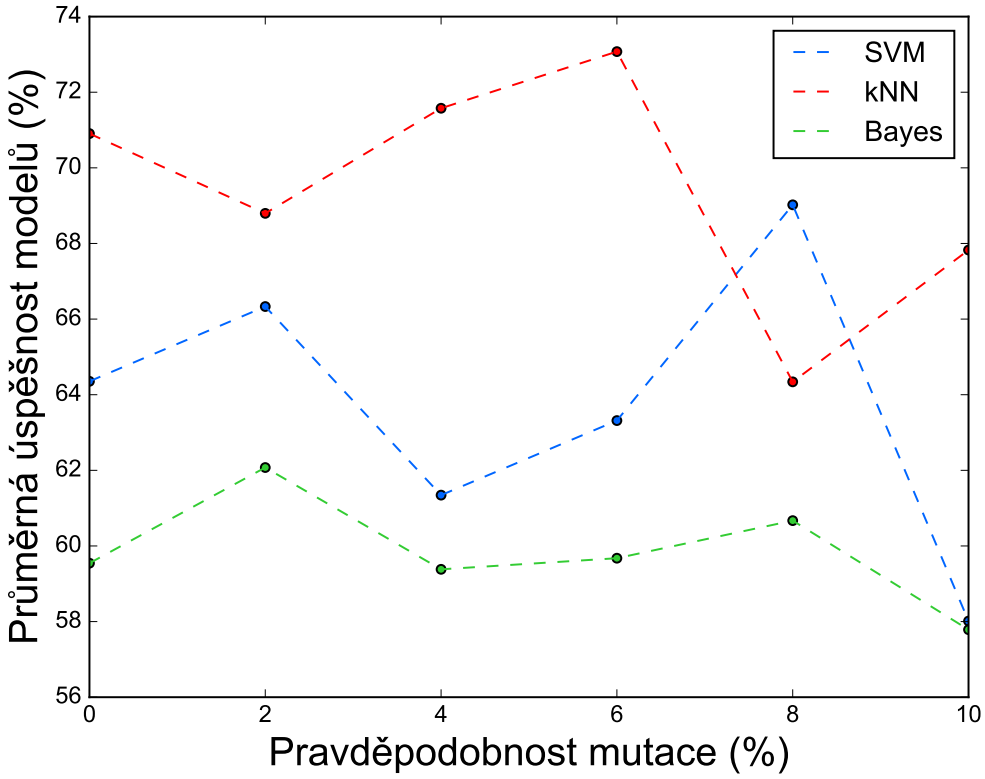
Procento vybraných atributů nevykazuje žádný trend a velmi se liší pro jednotlivé klasifikátory. Jedná se spíše o hodnoty náhodné a slouží spíše k porovnání s ostatními metodami.

Pravděpodobnost mutace

Pravděpodobnost křížení	*80%*
Pravděpodobnost mutace	***0 - 10%***
Velikost populace	*10*
Počet generací	*10*

Mutace je implementována jako jednoduchá inverze bitu na jedné pozici. Nejprve se u jedince vybraného k mutaci náhodně vybere jeden bit v chromozomu, který představuje konkrétní atribut z datového souboru. Následně pokud řešení atribut neobsahovalo, tak se přidá do řešení, pokud ho řešení obsahovalo, z řešení se odstraní.

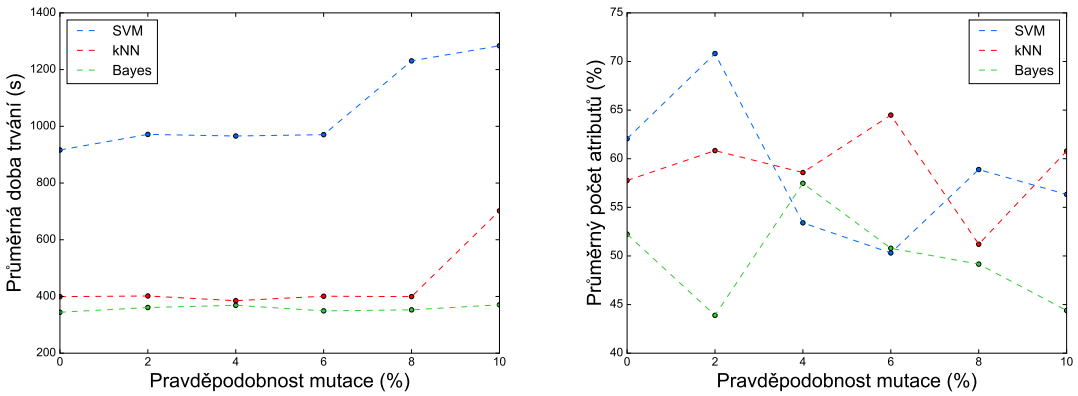
Úkolem mutace je zabránit předčasně konvergenci a zachování diverzity populace. Její pravděpodobnost by se měla vyskytovat spíše v nižších hodnotách, neboť vysoké hodnoty neodvolí populaci konvergovat k nějakému řešení a dosáhnout lokálních minim.



Obrázek 3.26: Úspěšnost klasifikátorů s měnící se pravděpodobností mutace

Vzhledem ke zvolené nízké velikosti populace z výpočetních důvodů a všeobecně nižší prav-

děpodobnosti mutace nemusí být výsledky statisticky příliš významné a výstupy z měření 3.26 neukazují nejlépe vlastnosti tohoto parametru. Měření byla také prováděna v intervalu 0 - 10%, což jsou běžné hodnoty. V případě měření až do 100% by byl patrný větší vliv náhodnosti nalezeného řešení a pravděpodobně i pokles úspěšnosti řešení. Proto nebude chybou zvolit **pravděpodobnost 1%**, což je bývá často doporučováno pro tento algoritmus.



Obrázek 3.27: Doba trvání a procentuální počet atributů mutace

Časový nárůst znázorněný na 3.27 je minimální vzhledem k celkově nízké pravděpodobnosti mutace, které se navíc často nemusí ani provést s nastavenou velikostí populace a počtem generací. Samotná mutace je implementovaná jednoduchou záměnou hodnoty, důsledek mutace na celkový výsledek je proto minimální.

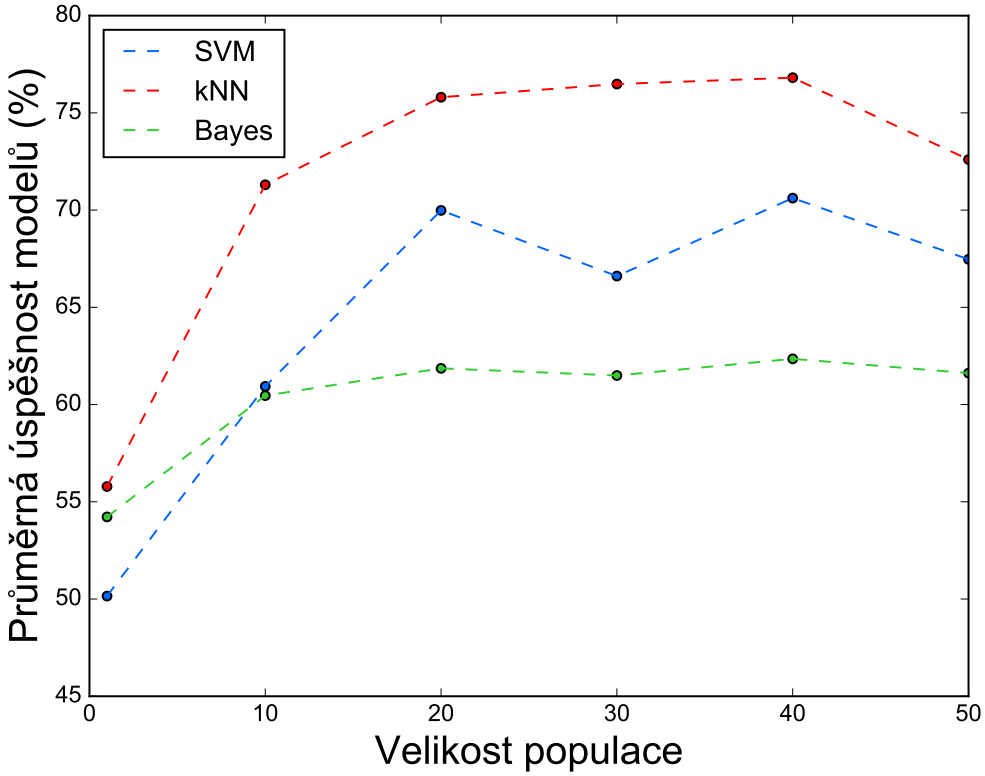
Velikost populace

Pravděpodobnost křížení	*80%*
Pravděpodobnost mutace	*1%*
Velikost populace	***1 - 50***
Počet generací	*10*

Velikost populace je parametr, který je velmi závislý jak na kvalitě řešení, tak výpočetní době. Vzhledem k náročnosti fitness funkce, která spočívá v ohodnocení stávajícího řešení, je to část výpočtu nejnáročnější na čas. Neboť čím větší je počet jedinců, tím více řešení musí být ohodnoceno. Nicméně s příliš malým počtem jedinců se nemusíme optimu ani přiblížit.

Počáteční řešení jedinců je náhodně vygenerovaná posloupnost atributů. Každý atribut má 50% šanci být zahrnut do počátečního řešení. Takto získané řešení se ohodnotí dle nějakého klasifikačního modelu a získaná úspěšnost představuje fitness jedince. S větším

počtem jedinců dosáhneme větší variability populace, což je znát obzvláště u datových souborů s větším počtem atributů. U dat s nízkým počtem atributů se nevyplatí vytvářet populaci s velkým počtem jedinců a prakticky za stejnou dobu můžeme mít k dispozici řešení hrubou silou.

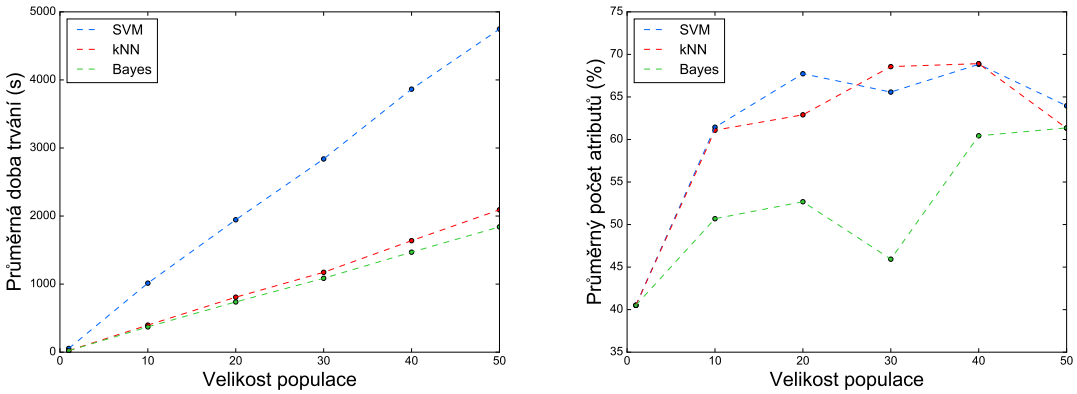


Obrázek 3.28: Úspěšnost klasifikátorů s měnící se velikostí populace

V obrázku 3.28 dochází k nárůstu úspěšnosti se zvětšující se populací, což je chování, které čekáme ve spojitosti se změnou tohoto parametru. Samozřejmě musíme počítat s nějakým rozptylem, ale standardně, úspěšnost klasifikátorů by měla konvergovat k nějaké hodnotě až nakonec bude docházet k žádnému či malému zlepšení úspěšnosti na úkor stále rostoucí časové dotace. Cílem je tedy nalézt bod, ve kterém i malé změny ve velikosti populace stále způsobují znatelný nárůst výkonnosti, což v našem případě je někdy kolem populace o velikosti 20. Jedná se ale o průměrné hodnoty několika různých datových souborů, přičemž každý z nich má odlišnou strukturu dat a rozdílný počet atributů. Kdybychom test prováděli individuálně na jednotlivá data, výsledek by nám vyšel pokaždé jiný.

Ve snaze zobecnit nastavení tohoto algoritmu budeme uvažovat **velikost populace 20**,

což je sice příliš mnoho pro malá data a naopak nedostačující pro větší, avšak dochází k nejlepšímu vyvážení cena/výkon napříč datovými soubory.



Obrázek 3.29: Doba trvání a procentuální počet atributů u různé populace

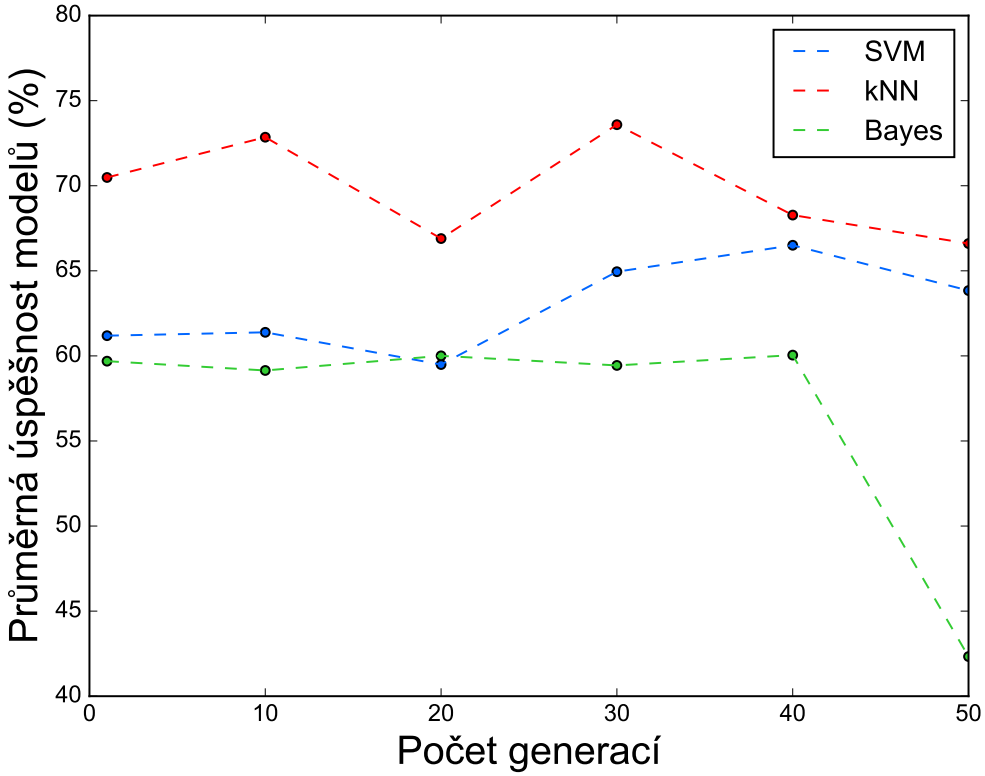
Obzvláště u SVM (obrázek 3.29) dochází ke stálému nárůstu doby výpočtu, která se pohybuje průměrně kolem 80 minut s populací o velikosti 50, což je v porovnání s 10 jedinci za 15 minut velký skok. Můžeme si všimnou, že rozdíl ve výkonu modelů není prakticky žádný. Méně strmý nárůst má kNN a Bayes, jejichž náročnost je téměř totožná.

Počet generací

Pravděpodobnost křížení	*80%*
Pravděpodobnost mutace	*1%*
Velikost populace	*10*
Počet generací	***1 - 50***

Poslední parametrem je počet generací v životním cyklu jedinců, tedy kolik iterací bude algoritmus hledat nejlepší řešení. Obdobně jako u velikosti populace, i zde budeme srovnávat výkonnost s náročností v jednotlivých generacích a budeme se pokoušet odhadnout moment, kdy ukončit výpočet a spokojit se se současným nejlepším řešení za cenu ušetření výpočetního času.

V algoritmu je implementována podmínka, která ukončí výpočet pokud již nějaký jedinec obsahuje globální optimum, tedy úspěšnost klasifikátoru je 100%. To je velmi nepravděpodobné u běžných datových souborů, nicméně u těch jednodušších (Iris) může tato situace nastat a urychlí to výpočet. Úspěšnost klasifikátoru nezávisí ovšem pouze na konkrétní podmnožině atributů, ale také na výběru testovacích dat a konkrétním sestavením modelu.

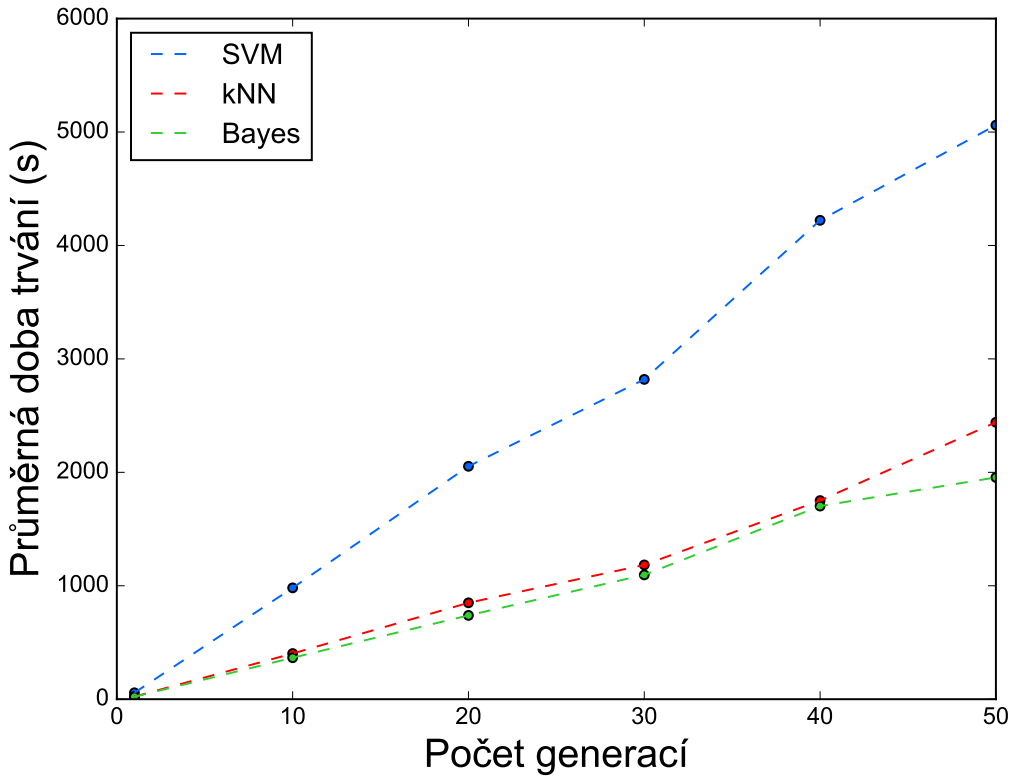


Obrázek 3.30: Úspěšnost klasifikátorů s měnícím se počtem generací

Výsledky naměřené v 3.30 jsou zajímavé tím způsobem, že s přibývajícím počtem generací nedochází k prakticky žádnému zlepšení. K tom došlo pravděpodobně z toho důvodu, že velikost populace byla nastavena na 10 jedinců, což neumožnilo populaci konvergovat k nějakému řešení. Bohužel s větším počtem jedinců stoupá výpočetní doba, díky čemuž je genetický algoritmus s rozsáhlejší populací a vyšším počtem generací méně použitelnější pro všeobecné uplatnění. Uvažujme pro srovnání s ostatními metodami **10 generací**, které lze spočítat v rozumném čase a porovnáme s ostatními metodami.

Z časové hlediska na 3.31 sledujeme obdobný trend jako se změnou velikosti populace, což je lineární nárůst časové složitosti, se strmějším průběhem pro SVM.

Doporučení
Genetický algoritmus v souvislosti s hledáním příznaků je specifická metoda, hodící se na extrémnější datové soubory. Vzhledem k závislosti a citlivosti parametrů nelze aplikovat jedno konkrétní nastavení všeobecně pro jakákoli data. Výpočetně zajímavých výsledků dosahuje pro data s větším počtem atributů, kdy se smaže časová režie způsobená chodem genetic-



Obrázek 3.31: Doba trvání a procentuální počet atributů s různým počtem generací

kého algoritmu a těží z postupného zlepšování řešení.

Kritickým bodem algoritmu je ohodnocovaní fitness funkce, která při každé iteraci ohodnocuje každého jedince. Obecné požadavky na fitness funkci je, aby byla rychlá, což se v kontextu se selekcí příznaků dá těžko dodržet. Proto klasifikátor spojený s genetickým algoritmem by měl být co nejrychlejší, přičemž SVM vyšel v našem nastavení se svojí dlouhou klasifikační dobou jako nevyhovující.

Nastavení parametrů, obzvláště velikosti populace a počtu generací, je velmi individuální a spojené s konkrétními daty. Lze použít nějakou z heuristik k jejich nastavení, většinou je však potřeba mít nějaké informace o zpracovaných datech a provést testy s různou kombinací těchto parametrů.

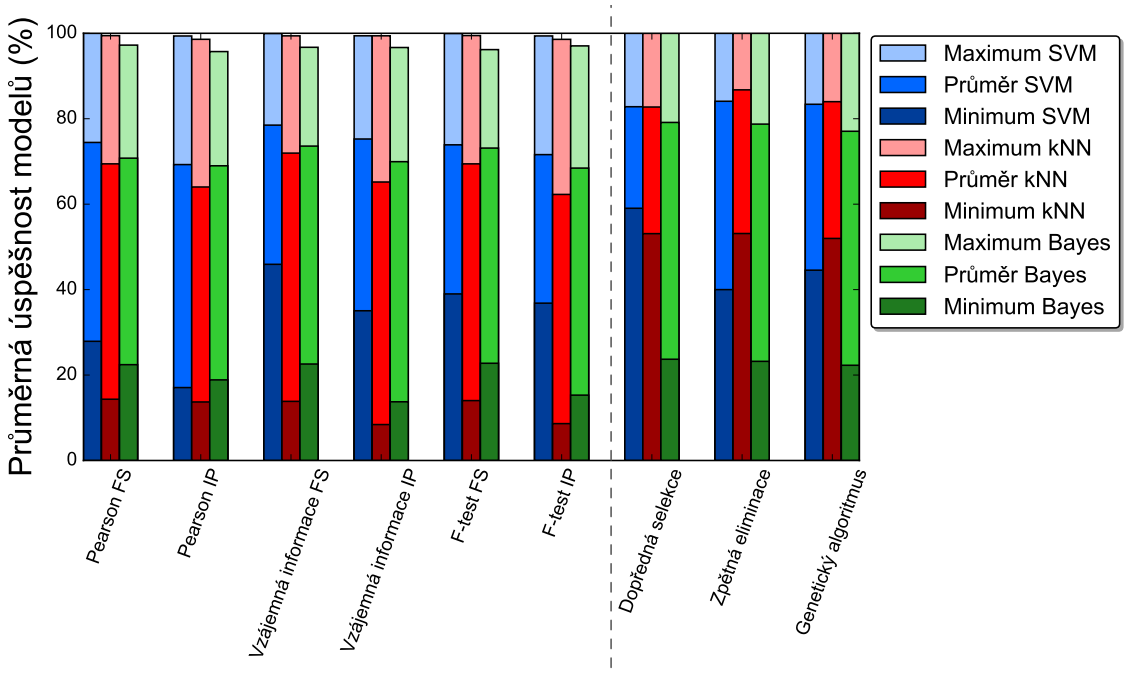
3.8 Wrapperové metody

V této sekci srovnáme metody, které získávají finální podmnožinu atributu na základě zpětné vazby atributů. Patří sem metody z předchozích sekcí s dopředným hladovým (TS) a inflexním (IP) způsobem selekce.

- Pearsonův korelační koeficient
- F-test (verze ANOVA)
- Vzájemná informace
- Sekvenční dopředná selekce
- Sekvenční zpětná eliminace
- Genetický algoritmus (s parametry výše)

Pro srovnání vybereme také populární *Pearsonův korelační koeficient*, *Vzájemnou informaci* a *F-test*, jakožto reprezentanty všech kategorií.

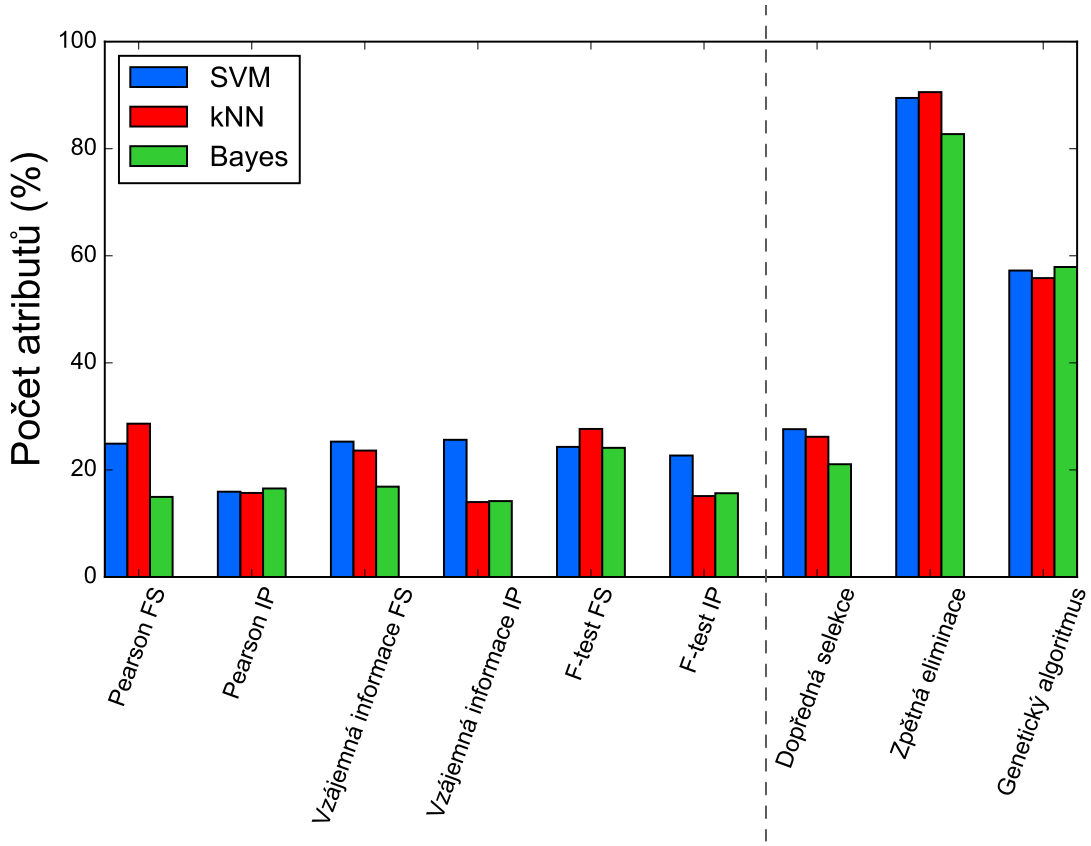
Výkonnost



Obrázek 3.32: Úspěšnost wrapperových metod na jednotlivých klasifikátorech

Výkonnost znázorněna na 3.32 ukazuje na vyšší úspěšnost čistě wrapperových metod oproti filter-wapperovém výběru. Co se týče minimálních a maximálních hodnot, i zde je výsledek

lepší, obzvláště v kombinace s kNN, kde minimum dosahuje několikanásobných hodnot. Nejlepšího výsledku v rámci jednotlivých klasifikátorů dosáhla sekvenční zpětná eliminace, s maximem v kombinaci s modelem kNN. Bayesův klasifikátor dosáhl oproti ostatním nízkých minimálních hodnot na datových souborech šachového zakončení a srdeční arytmie, což jsou soubory s větším počtem atributů. Úspěšnost se zde pohybovala pouze okolo 20%, zatímco modely SVM a kNN dosahovaly o poznání více (k 60%), navíc rozdíl mezi minimem a průměrem je malý.



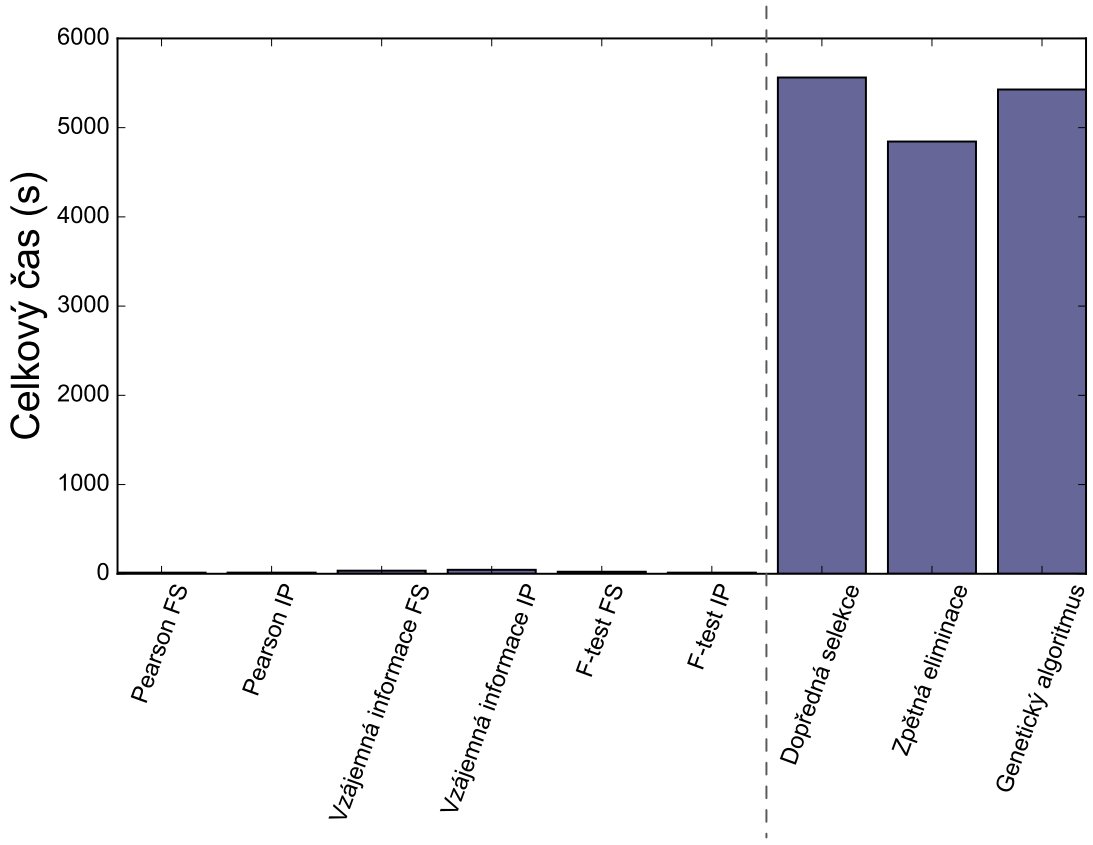
Obrázek 3.33: Procento vybraných atributů wrapperovými metodami

Počet atributů

Procentuální zastoupení atributů 3.33 je vysoké u sekvenční zpětné eliminace, což se dalo očekávat, jelikož klasifikace začíná s kompletním počtem atributů a postupně se ubírají podle výkonnosti. Zde dochází k riziku, že ve výsledné podmnožině atributů zůstanou i ty nepotřebné, avšak na druhou stranu dokáže zachovat atributy, které jsou užitečné pouze dohromady

Opačného chování sledujeme u sekvenční dopředné selekce, kde je počet atributů obdobný jako u filter-wrapperových metod, což je kolem 20% atributů.

U genetického algoritmu se počet atributů vyvíjel na základě mutace a selekce, jejich počet se tedy dynamicky mění a proto počet atributů kolem 50% odpovídá průměrné evoluci jedinců.



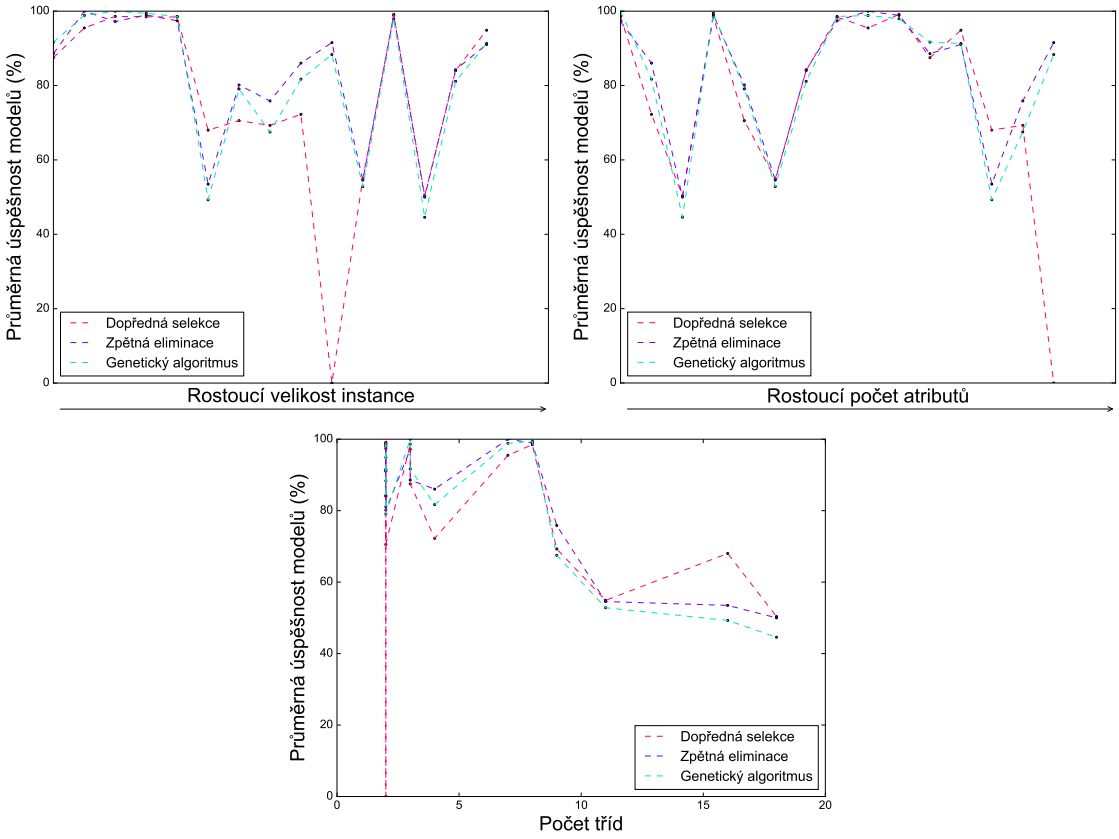
Obrázek 3.34: Výpočetní čas jednotlivých wrapperových metod

Časová náročnost
Srovnání celkového času jednotlivých metod zobrazené na 3.34 se u filter-wrapperových metod skládá z doby ohodnocení atributů, doby hledání podmnožiny atributů (FS nebo IP) a následná klasifikace pomocí SVM, kNN a Bayesem. U wraperrových metod je vše obsaženo v jednom kroku.

Je zde vidět velký rozdíl mezi jednotlivými typy metod. Například sekvenční dopředná selekce pro datový soubor Internetových reklam nenašla řešení ani po několika desítkách hodin vypočtu. Tento datový soubor obsahuje velké množství atributů a doba výpočtu

dopřednou selekcí trvá neúměrně dlouho.

Ač dosáhly wrapperové metody lepších úspěšností na testovacích datech, časová dotace pro výpočet je několikanásobně větší, než u filtračních metodách s wrapperových přístupem hledání atributů. Genetický algoritmus vzhledem k náročnosti fitness funkce vyhodnocuje atributy velmi dlouhou dobu a to i s malým počtem generací a nízkou velikostí populace, která byla nastavena.



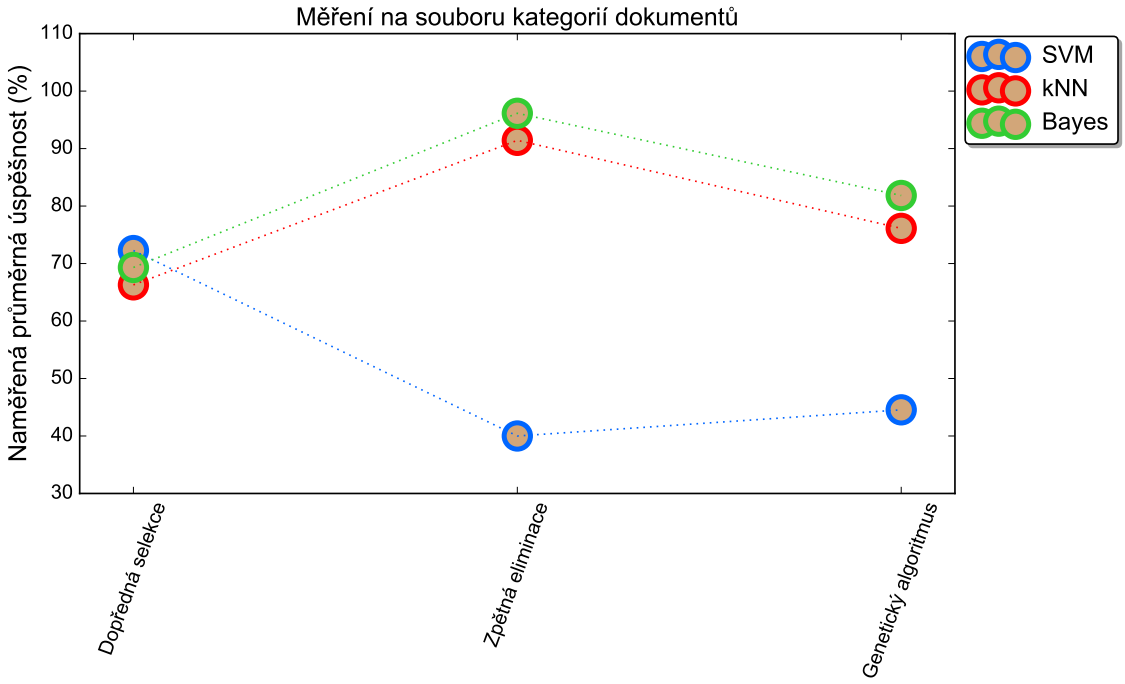
Obrázek 3.35: Porovnání závislostí wrapperových metod

Ve vzájemných závislostí na velikosti instance, počtu atributů a počtu tříd na 3.35 vidíme pokles s rostoucím počtem atributů v případě sekvenční dopředné selekce až k výsledné nule, což představuje hodnotu nedoběhlého měření datového souboru Internetových reklam, který obsahuje nejvíce atributů. Chování metod je jinak velmi podobné ve sledovaných parametrech.

Doporučení

Genetický algoritmus nepatří mezi metody, které by se měly vyzkoušet jako jedny z prv-

ních. Naopak využití tohoto algoritmu je spíše experimentální, hodící se v případech, kdy jiné metody selžou. Je potřebné nastavit pravděpodobnosti křížení a mutace na základě zvolených experimentů a hlavně velikost populace a počet generací, což jsou parametry přímo ovlivňující dobu běhu a konvergenci řešení v čase. V obecném nastavení použitým v tomto měření je algoritmus neefektivní a měl by se optimalizovat na konkrétní typy datových souborů.



Obrázek 3.36: Porovnání metod na datovém souboru kategorií dokumentů

Sekvenční metody patří mezi základní wrapperové metody, jsou velmi jednoduché a snadno reprezentovatelné. Dopředná selekce umožňuje vybrat malý počet užitečných atributů, avšak neodhalí kombinace méně důležitých atributů, které jsou pospolu velmi užitečné. Dopředná selekce nabízí obdobné výsledky napříč klasifikátory 3.36, úspěšnost zpětné eliminace se liší na základě specifik jednotlivých datových souborů a použitého klasifikačního modelu, kde můžeme dosáhnout dosti rozdílných výsledků.

Doba běhu je také velká pro datové soubory s větším počtem méně signifikantních atributů, menší v případě výskytu důležitých atributů. Hodí se použít pro data, která omezí prostor hledaných atributů co nejvíce důležitými příznaky v kombinaci s libovolným klasifikátorem, který poskytuje dostačující výkon za rozumný čas. Nezávislost klasifikátoru je výhoda wrapperových metod a lze optimalizovat pro konkrétní problém.

U zpětné eliminace je naopak časová dotace delší v počátku (zahrnují se všechny atributy) avšak lépe nalezne jednotlivě vztahy mezi atributy, hodí se tedy použít pro data s komplexními vazbami mezi atributy, případně pokud je žádoucí zahrnutí většího počtu příznaků.

3.9 Porovnání s hrubou silou

Po vybrání 8 datových souborů s menším počtem atributů otestujeme vybrané zástupce filtračních, filtračně-wrapperových i wrapperových metod s hrubou silou a zjistíme tím jejich efektivnost. Mezi **filtračními metodami** porovnáme

- Pearsonův korelační koeficient - poměrová selekce
- F-test (verze ANOVA) - poměrová selekce
- F-test (verze ANOVA) - prahová selekce
- Vzájemná informace - prahová selekce

Zástupci **filter-wrapperových metod** jsou

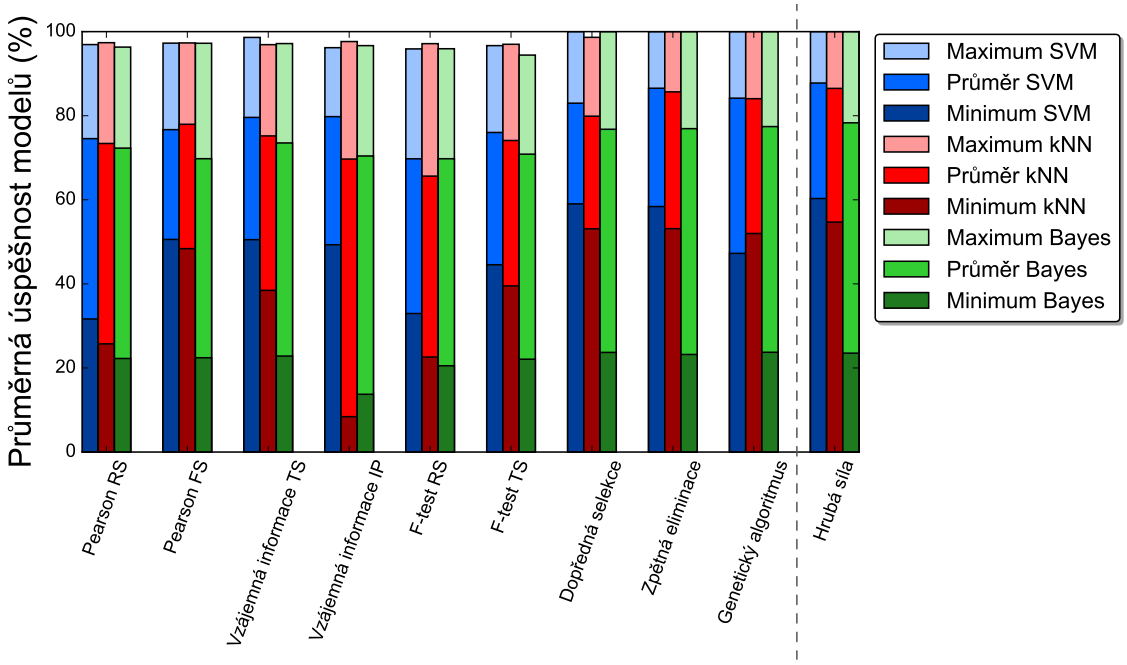
- Pearsonův korelační koeficient - hladová dopředná selekce
- Vzájemná informace - inflexní selekce

Z čistě **wrapperových metod** vybereme

- Sekvenční dopředná selekce
- Sekvenční zpětná eliminace
- Genetický algoritmus

Výkonnost
Výkonnost metod je znázorněna na 3.37. Všimněme si, že rozdíl mezi hrubou silou, které poskytuje exaktní řešení vůči ostatním na 8 nejmenších datových souborech, je malý. Například zpětná eliminace dosahuje téměř totožných hodnot a nejedná se o exaktní metodu. Za ostatními lehce zaostávají obě verze F-testu, které při zprůměrování těchto datových souborů mají od ostatních odstup (obzvláště pro SVM). Je vidět, že nelze dosáhnout 100% úspěšnosti ani prozkoušením všech možností, a dosažená úspěšnost má své limity a většina metod se k tomuto limitu přibližuje, alespoň při aplikaci na data s menším počtem atributů.

Počet atributů
Co se týče počtu atributů 3.38, hrubá síla v průměru dosahuje lehce přes 50% vybraných atributů, což je velmi podobné jako u například u genetického algoritmu či odpovídá poměrové selekci. Nejvíce atributů bylo vybráno pro klasifikátor kNN, nejméně pak pro Naivního



Obrázek 3.37: Úspěšnost vybraných metod vůči hrubé síle

Bayese.

Časová náročnost

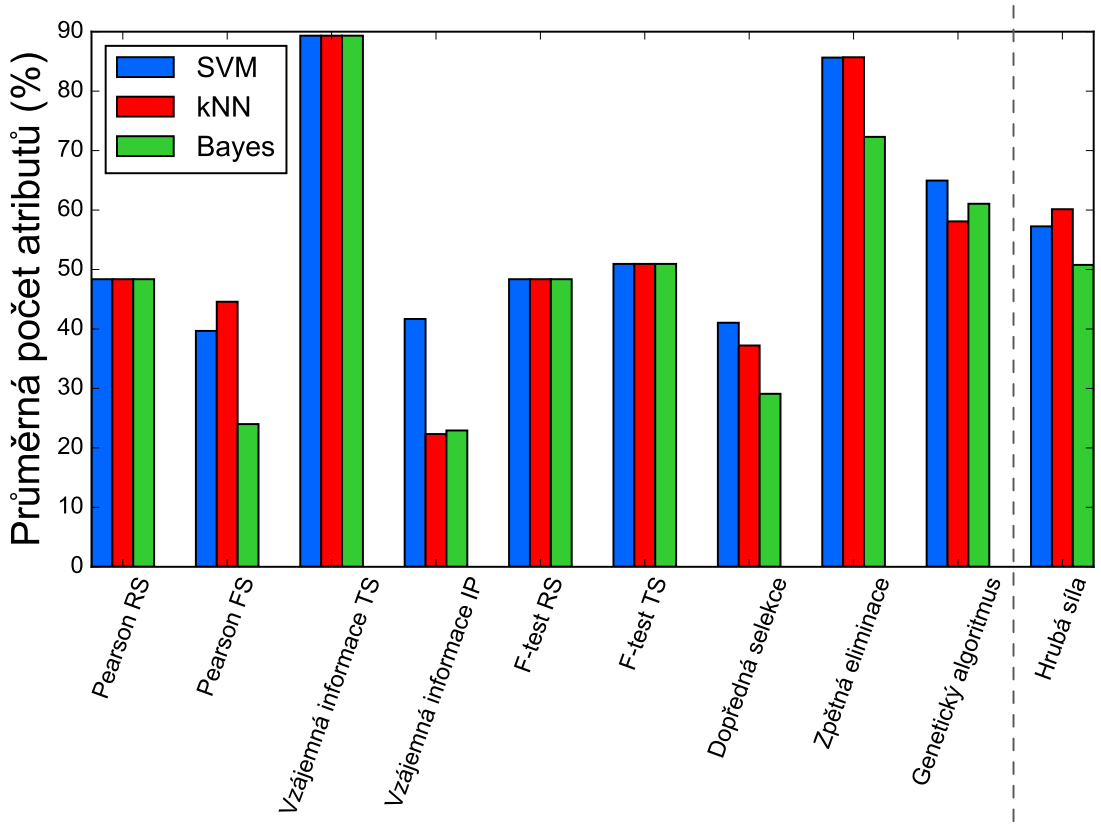
Časové porovnání metod vůči hrubě síle 3.39 je velmi důležitý ukazatel jejich použitelnosti a náročnosti. Vidíme, že většina metod se drží řádově v nižších hodnotách než hrubá síla s výjimkou genetického algoritmu.

Zde se prokázalo, že genetický algoritmus není vhodný pro běžné datové soubory s menším počtem atributů, neboť časová dotace pro výpočet je dokonce i větší než u hrubé síly. Síla algoritmu tedy není v běžném použití, nýbrž pro specializované datové soubory s velkým počtem atributu a vyladěnými atributy právě pro konkrétní datový soubor. V případě zvýšení velikosti populace či počtu generací by byla doba výpočtu ještě delší, nicméně výkonnostní výstupy algoritmu jsou srovnatelné s metodou hrubé síly, neboť v tomto omezeném rozsahu atributů jedinci v závislosti na náhodě s velkou pravděpodobností vyzkouší všechna řešení.

Dopředná selekce v testu malých dat vyšla dobře, neboť hlavní nedostatky, s kterými se potýkala ve srovnání s wrapperovými metodami byly u rozsáhlých datových souborů a náročnost byla nižší než u filtračně-wrapperových metod.

Doporučení

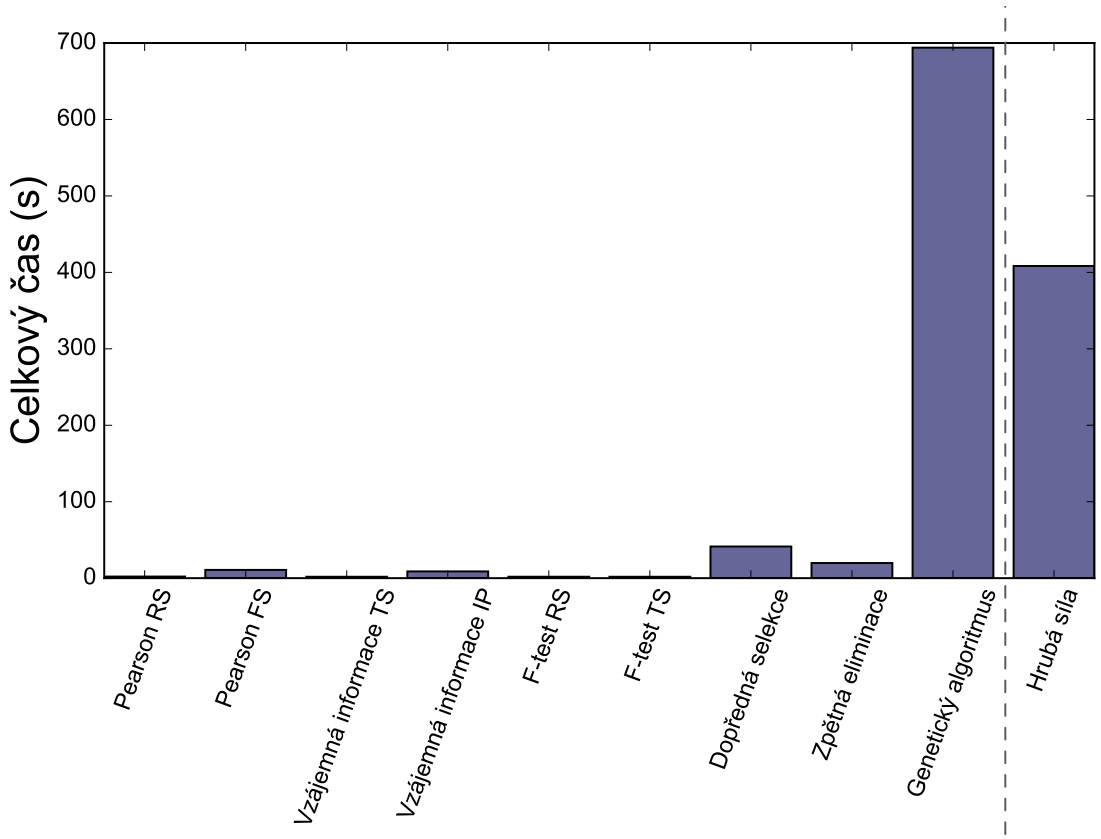
Na malých datových souborech fungují dobře téměř všechny metody. Dobré výsledky za



Obrázek 3.38: Procento vybraných atributů s porovnáním hrubé síly

krátký čas měly obě sekvenční selekce, které jsou stavěny pro obdobné typy problémů, nicméně charakter filtračních metod je schopný odhalit statistické vlastnosti jednotlivých atributů s tím, že je možné jich vyzkoušet více za cenu wrapperové selekce a za cenu hrubé síly jsme schopni otestovat téměř všechny dostupné.

Genetický algoritmus se k výpočtu příliš nehodí vzhledem ke svému velkému nároku na výpočetní čas a je lepší ho využít v případě selhání jiných možností.



Obrázek 3.39: Výpočetní čas vybraných metod a hrubé síly

3.10 Embedded metody

Embedded metody se koncepčně více odlišují od ostatních, neboť klasifikátor je přímo optimalizován jako součástí učení a tedy má rozdílná nastavení než v předchozích metodách. Metody vybrané ke srovnání jsou:

- Rozhodovací stromy
- Rekurzivní eliminace příznaků (RFE)
- ℓ_1 SVM
- LASSO

Porovnání se zbylými metodami naznačíme, avšak je potřeba si uvědomit rozdílnost nastavení. Výsledky jsou tedy spíše názorné a mohou se velmi lišit v případě rozdílných implementací. Nejprve zmíníme nastavení jednotlivých metod, se kterými pracujeme a budeme testovat.

Rozhodovací stromy

- Funkce měřící kvalitu rozdělení: *Entropie*
- Strategie rozdělení: *Nejlepší*
- Maximální počet atributů: *Neomezeno*
- Maximální hloubka stromu: *Neomezeno*
- Minimum vzorků potřebných k rozdělení uzlu: *2*
- Minimální počet vzorků obsažených v listu: *1*
- Maximální počet listů: *Neomezeno*
- Váha tříd: *Stejná*
- Práh pro zastavení růstu stromu: 10^{-7}
- Předtřídění dat: *Ne*

Rekurzivní eliminace příznaků

- Velikost kroku algoritmu: *1*
- Validace: *3-násobná křížová validace*
- Klasifikační model: ***Support Vector Machines***
  - Regulátor C: *1*

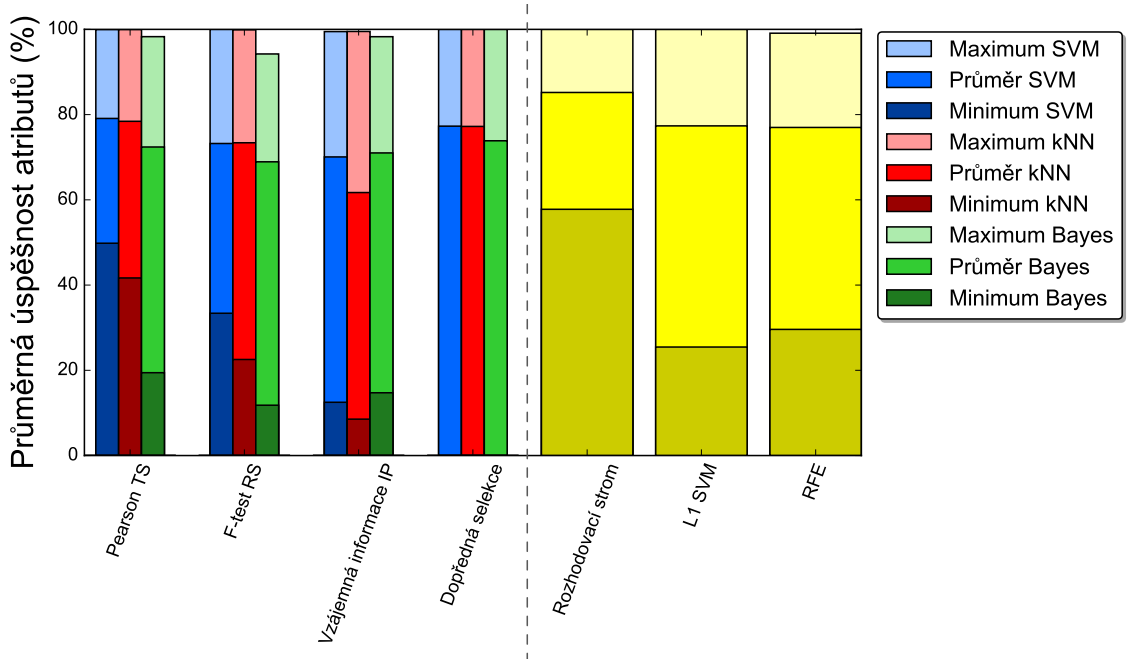
  - Kernelová funkce: *Lineární*
  - Váha tříd: *Stejná*
  - Rozhodovací funkce: *Jeden-proti-jednomu*
  - Gamma koeficient kernelové funkce: $\frac{1}{atributů}$
  - Shrinking heuristika: *Ano*
  - Tolerance ukončovacího kritéria: 10^{-3}
  - Velikost cache: *200MB*
  - Počet iterací: *Neomezeně*

ℓ_1 ***SVM***

- Penalizace: ℓ_1
- Ztráta: ℓ_1 *čtvereční*
- Optimalizační algoritmus: *Primární optimalizační problém*
- Regulátor C: *1*
- Kernelová funkce: *Lineární*
- Váha tříd: *Stejná*
- Rozhodovací funkce: *Jeden-proti-všem*
- Tolerance ukončovacího kritéria: 10^{-4}
- Centrování dat: *Ano*
- Počet iterací: *1000*

LASSO

- Parametr α: *1*
- Selekce: *Cyklická, sekvenční smyčka přes atributy*
- Tolerance ukončovacího kritéria: 10^{-4}
- Normalizace: *Ano*
- Centrování dat: *Ano*
- Počet iterací: *1000*



Obrázek 3.40: Úspěšnost embedded metod

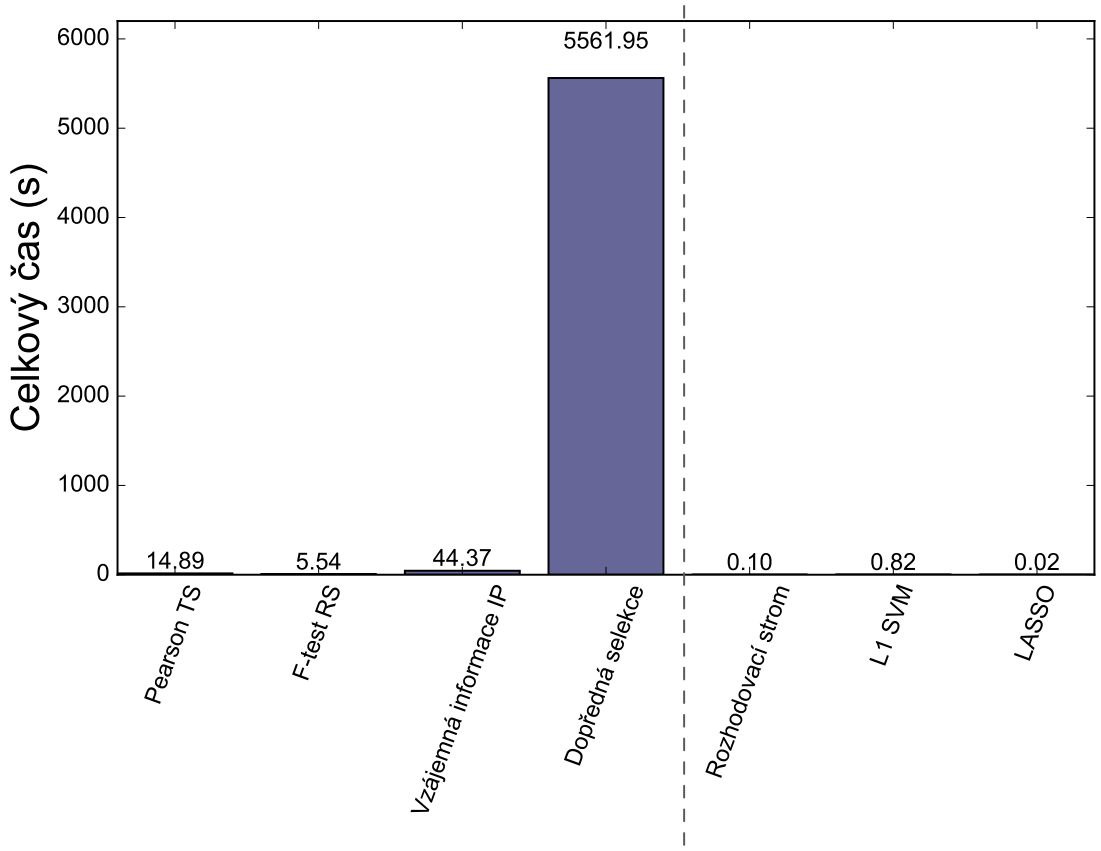
Výkonnost

Z hlediska výkonnosti nelze do porovnání zahrnout LASSO, neboť jeho skóre se určuje jiným způsobem, nikoli procentuální úspěšností. Tabulka s úspěšnostmi je k dispozici v příloze. Nejlepší skóre značí skóre 1, avšak muže dosahovat i záporných hodnot, pokud si model vede hůř, než konstantní model s predikcí té samé hodnoty. Datové soubory získaly záporné či nulové hodnoty, což značí špatně natavený model či malé předzpracování dat a nehodí se tedy pro srovnání.

Na obrázku 3.40 je znázorněno porovnání zbylých metod i se zástupci ostatních kategorií. V průměru se tedy držíme v stále stejných hodnotách, nicméně metoda RFE přes svojí velkou časovou náročnost nebyla použita na všech datových souborech, porovnání je tedy velmi orientační. Dobrých výsledků dosáhl rozhodovací strom, což je jedna z nejpopulárnějších metod ke klasifikace a regresi. Obzvláště minimální úspěšnost držící se okolo 60% je velmi nadprůměrná.

Časová složitost

Délku trvání klasifikace porovnáme nejprve bez RFE na všech datových souborech, znázorněno na 3.41, kde se projevuje velká neefektivita wrapperové dopředné selekce. Naopak vybrané embedded metody dosahují velmi dobrých časů, avšak u filtračních metod je započítáno ohodnocení všemi třemi klasifikátory.

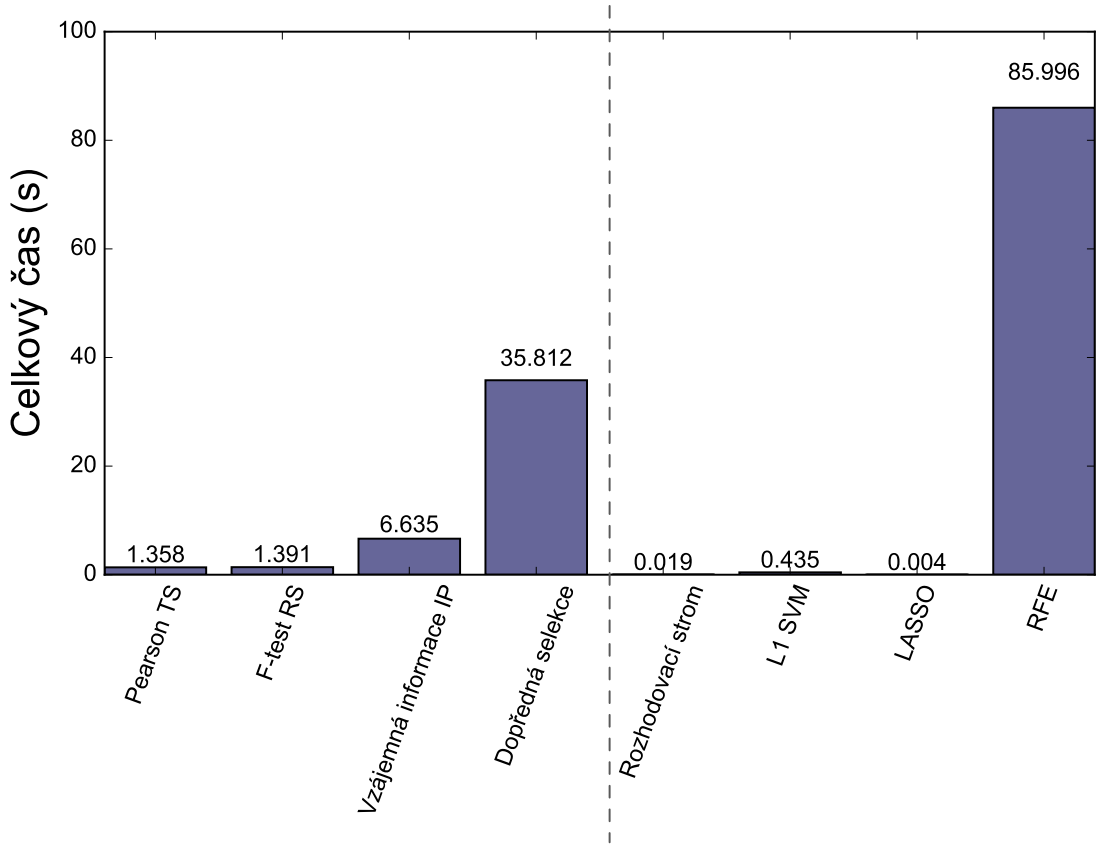


Obrázek 3.41: Výpočetní čas embedded metod na vybraných datových souborech

Se započítáním pouze souborů, které byly použity v kombinaci s metodou RFE 3.42, vidíme že RFE v tomto nastavení je více jak 2x náročnější než Sekvenční dopředná eliminace. Zbylé embedded metody vyžadují téměř zanedbatelný čas k natrénování.

Doporučení
Rozhodovací stromy patří mezi snadno interpretovatelné metody, které odhalí nelineární vztahy. Trénování modelu na základě trénovacích dat je rychlé a pochopitelné (whitebox). Nicméně nevýhodou je malá stabilita, tedy že malé změny v datech způsobí větší změny ve stromu. Model postrádá tedy robustnost v tomto směru a hodí použít více stromů (ensemble), vytvořit *náhodný les*. Horších výsledků dosahuje také v případě menšího vzorku testovacích dat (rozdělení ecoli, rakovina plic), kde dosažená úspěšnost je spíše podprůměrná.

Velmi náročnou metodou spadající pod wrapper i embedded metody se v testech ukázala Rekurzivní eliminace příznaků. V implementaci používá k ohodnocení SVM klasifikátor a při každé iteraci dochází k přenastavení váhy. Pro rychlejší běh se vyplatí omezit počet



Obrázek 3.42: Výpočetní čas embedded metod na všech datových souborech

iterací nějakou horní mezí, neboť výpočet trvá příliš dlouho. Druhou možností je zvýšit počet příznaků, které se každým krokem odebírají. Dobrá optimalizace parametrů je klíčová pro použitelnost metody.

ℓ_1 SVM získalo například u datového souboru šachových zakončení pouze 25%, což je velký rozdíl oproti rozhodovacímu stromu s 81% či ostatních v kombinaci SVM či kNN pohybující se kolem 60%. Způsobeno to mohlo být omezením počtu iterací či volbou ztrátové normy. Jedná se o metodu s velkým počtem parametrů, které je potřeba optimalizovat pro daný problém. To samé se dá říci v případě LASSO, kde nemáme k dispozici přímá porovnání z hlediska úspěšnosti, avšak hodnocení bylo často záporné, což značí špatně nastavený model. Vyskytují se i skóre nulové, které značí konstantní model, tedy predikce stejné hodnoty nezávisle na vstupních atributech.

3.11 Náhled na metody

Celkové zařazení jednotlivých metod je znázorněno v tabulce 3.2. Existuje samozřejmě spousta rozšíření, která pozmění vlastnosti jednotlivý metod, avšak uvažujeme pouze základní formy metod.

Tabulka 3.2: Různé pohledy na metody selekce příznaků

Metody	**Filtrační**	**Wrapper**	**Embedded**	**Univarietní**	**Multivarietní**	**Parametrické**	**Neparametrické**	**Nastavitelné**	**Vztah**
Pearsonův koeficient	✓	✓[a]	✗	✓	✗	✓	✗	✗	Lineární
Spearmanův koeficient	✓	✓[a]	✗	✓	✗	✗	✓	✗	Monotónní
Kendallův koeficient	✓	✓[a]	✗	✓	✗	✗	✓	✗	Monotónní
F-test	✓	✓[a]	✗	✓	✗	✓	✗	✗	Lineární
T-test	✓	✓[a]	✗	✓	✗	✓	✗	✗	Lineární
Chí-kvadrát	✓	✓[a]	✗	✓	✗	✗	✓	✗	Lineární
Vzájemná informace	✓	✓[a]	✗	✓	✗	✗	✓	✗	Nelineární
Dopředná selekce	✗	✓	✗	✗	✓	✗	✓	✗	Nelineární
Zpětná eliminace	✗	✓	✗	✗	✓	✗	✓	✗	Nelineární
Genetický algoritmus	✗	✓	✗	✗	✓	✗	✓	✓	Nelineární
Rekurzivní zpětná eliminace	✗	✗	✓	✗	✓	✗	✓	✓	Lineární
Rozhodovací strom	✗	✗	✓	✓[b]	✓[c]	✗	✓	✓	Nelineární
L1 SVM	✗	✗	✓	✗	✓	✓	✗	✓	Lineární
LASSO	✗	✗	✓	✗	✓	✓	✗	✓	Lineární

[a] U způsobů vybrání příznaků, kde je klasifikátor zahrnutý
[b] Za použití algoritmu ID3
[c] Za použití algoritmu CART

Závěr

Cílem diplomové práce bylo nejprve seznámit se z hlavními myšlenkami metod pro selekci atributů a identifikovat jejich kategorie a následně vybrané zástupce implementovat ve skriptovacím jazyce Python a porovnat z hlediska úspěšnosti, časové náročnosti na referenčních klasifikačních algoritmech a odhalit možnosti a omezení jednotlivých metod.

Cíl se podařilo splnit. Teoretická část práce je obsažena v části analýza, rozdělená podle základních kategorií metod, která obsahuje poznatky získané z rešeršní analýzy. Samotné implementaci je věnovaná samostatná kapitola obsahující náhled na použitá data a zahrnuje i způsoby úpravy atributů a doplnění chybějících hodnot. Následná porovnání jsou prováděna v části měření se zástupci metod jednotlivých kategorií, které byly implementovány v jazyce Python a je možné nahlédnout na porovnání těchto metod případně odvodit vhodné parametry pro jejich použití.

Metody selekce příznaků založené na korelaci jsou rychlé a snadno použitelné. Jejich síla tkví právě v jednoduchosti a vyplatí se použít více korelačních metod k získání informací o datovém souboru. Z měření vyšlo také najevo, že je vhodnější použít tyto metody s poměrným počtem atributů nastaveným na 50% či podle definovaného prahu úspěšnosti, který z měření vyšel nejlépe na hodnotě 0,1 u normalizovaného skóre.

Statistické metody mají naměřenou výkonnost obdobnou jako korelační a každá nabízí specifickou vlastnost, kterou je potřeba co nejvíce přizpůsobit datovému souboru. Ať už se jedná o metriku průměru v případě *T-testu*, či rozptylu v případě *F-testu*, nicméně pro obecnější typy úloh je lepší použít právě F-test. Dále *Chí-kvadrát* metrika je jedna se základních a často používaných metod k měření vztahů mezi atributem a třídou, nicméně pro lepší relevanci je doporučeno mít datový soubor o větší velikosti.

Metoda *Vzájemné informace* je jediná nelineární filtrační metoda, která dosahovala slušného skóre i v kombinaci s dopřednou či inflexní selekcí, což jsou způsoby výběru podmnožin atributů, kde měly korelační i statistické metody nejhorší výsledky. Obzvláště v kombinaci s SVM bylo procento úspěšnosti v průměru nad 70%. Tyto klady mají cenu

větší časové náročnosti ve srovnání se zbylými metody založenými na ohodnocení.

Z rodiny wrapperových metod implementace *Genetického algoritmu* neprokázala příliš velkou použitelnost na obecných případech, neboť na menších datových souborech časová složitost signifikantně přesahovala složitost hrubé síly a vzhledem k omezené možnosti velikosti populace i počtu iterací nebyl algoritmus schopný příliš konvergovat k optimálnímu řešení. V kombinaci s tímto algoritmem je potřeba vybrat klasifikační model s co nejrychlejší odezvou, neboť fitness funkce algoritmu byla právě úzkým hrdlem cyklu algoritmu. Oba sekvenční algoritmy dosahovaly obdobných výsledků s mírnou preferencí *zpětné eliminace*, neboť pro datové soubory s velkým počtem atributů byl rozdíl časové složitosti razantní. Zpětná eliminace byla také schopna lépe odhalit vazby mezi atributy, naopak podmnožina atributů vrácená *dopřednou selekcí* byla vždy menší, tedy proběhla větší redukce dimenze dat.

Posledním typem měřených metod jsou metody embedded, které vzhledem ke své struktuře nemohou být ohodnoceny stejným způsobem a za stejných podmínek jako metody předchozí, neboť mají naprosto odlišný způsob pracování s klasifikátory, a tím i jiné jejich nastavení případně jiný typ. Velmi dobrého průměrného i minimálního skóre dosáhly *Rozhodovací stromy*, jež se trénují i krátkou dobu. Jejich riziko spočívá v přeučení, a tím spojeným nárůstem komplexity, proto se vyplatí v těchto případech vytvořit nějaký typ „ensemble“ selekce příznaků, například Náhodný les. *Rekurzivní eliminace příznaků* v kombinaci s lineárním SVM byla časově náročnější než sekvenční dopředná selekce, čímž byla nepoužitelná pro větší data. ℓ_1 *SVM* a *LASSO* získaly nízké skóre na většině datových souborů, což bylo způsobeno buď nízkou úrovní předzpracování dat, či nepřizpůsobením parametrů typům dat.

Vypracování této práce mi dalo povědomí o různých kategorií selekce příznaků, pomohlo mi zorientovat se v jejich použitelnosti, uvědomění si silných a slabých stránek vybraných metod ale také jejich omezení, která mají jednotlivé metody, ať už z hlediska struktury či závislosti dat.

Literatura

[1] *Machine learning datasets*, [vid. 2016-11-12]. Dostupné z: https://archive.ics.uci.edu.

[2] *Scicit learn*, [vid. 2016-11-12]. Dostupné z: http://scikit-learn.org/stable.

[3] HANG, Yin-Wen, HSIEH, Cho-Jui a CHANG, Kai-Wei, *Training and testing low-degree polynomial data mappings via linear svm [online]*, Journal of Machine Learning Research (2010), [vid. 2017-01-26]. Dostupné z: http://www.jmlr.org/papers/volume11/chang10a/chang10a.pdf.

[4] KALÁTOVÁ, Eva a DOBIÁŠ, Jaroslav, *Evoluční algoritmy [online]*, Duben 2000, [vid. 2017-02-13]. Dostupné z: http://www.kiv.zcu.cz/studies/predmety/uir/gen_alg2/E_alg.htm.

[5] VERGARA, Jorge R. a ESTEVÉZ, Pablo A., *A review of feature selection methods based on mutual information [online]*, (2013), [vid. 2017-01-12]. Dostupné z: http://repositorio.uchile.cl/bitstream/handle/2250/126533/A-review-of-feature-selection-methods-based-on-mutual-information.pdf?sequence=1.

[6] KOHAVI, Ron a JOHN, George H., *Wrappers for feature subset selection [online]*, (1996), [vid. 2017-02-12]. Dostupné z: http://ai.stanford.edu/~ronnyk/wrappersPrint.pdf.

[7] ZIELINSKI, Karin, PETERS, Dagmar a LAUR, Rainer, *Stopping criteria for single-objective optimization [online]*, Institute for Electromagnetic Theory and Microelectronics, [vid. 2017-02-20]. Dostupné z: https://pdfs.semanticscholar.org/29ad/325712735f8c0c617f7430e2092630abac7e.pdf.

[8] ZHU, Weidong a LIN, Yongmin, *Using gini-index for feature weighting in text categorization [online]*, Journal of Computational Information Systems (2013), [vid. 2017-01-22]. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.353.3216&rep=rep1&type=pdf.

[9] YANG, Jieming, QU, Zhaoyang a LIU Liu, *Improved feature-selection method considering the imbalance problem in text categorization [online]*, Scientific World Journal (2014), [vid. 2017-01-22]. Dostupné z: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4058251/.

[10] TIEMANN, Thomas K. a MAHBOBI, Mohammad, *Introductory business statistics with interactive spreadsheets – 1st canadian edition*, Cambridge University Press, 2010.

[11] KORDÍK, Pavel a MOTL, Jan, *Vytěžování znalostí z dat [online]*, [vid. 2017-01-19]. Dostupné z: https://edux.fit.cvut.cz/oppa/BI-VZD/prednasky/.

[12] KUMAR, Neeraj, ZHANG, Li a NAYAR, Shree, *What is a good nearest neighbors algorithm for finding similar patches in images? [online]*, [vid. 2017-01-25]. Dostupné z: http://www1.cs.columbia.edu/CAVE/publications/pdfs/Kumar_ECCV08_2.pdf.

[13] MANNING, Christopher D. , RAGHAVAN, Prabhakar a SCHÜTZE, Hinrich, *Introduction to information retrieval*, Cambridge University Press, 2008.

[14] PERKINS, Simon, LACKER, Kevin a THEILER, James, *Grafting: Fast, incremental feature selection by gradient descent in function space [online]*, Journal of Machine Learning Research (2003), [vid. 2016-12-10]. Dostupné z: http://www.jmlr.org/papers/volume3/perkins03a/perkins03a.pdf.

[15] ŘEZÁNKOVÁ, Hana, MAREK, Luboš a VRABEC, Michal, *Interaktivní učebnice statistiky: Typy proměnných [online]*, [vid. 2017-01-12]. Dostupné z: http://iastat.vse.cz/typy_promennych.html.

[16] TU, Chung-Jui, CHUANG, Li-Yeh, CHANG, Jun-Yang, a YANG, Cheng-Hong, *Feature selection using pso-svm [online]*, IAENG International Journal of Computer Science, [vid. 2017-02-20]. Dostupné z: http://www.iaeng.org/IJCS/issues_v33/issue_1/IJCS_33_1_18.pdf.

[17] GUYON, Isabelle, GUNN, Steve, NIKRAVESH, Masoud a ZADEH, Lofti A., *Feature extraction*, vol. 207, Springer, 2006.

[18] BUREŠ, Lukáš a ZIMERMANN, Petr, *Metody počítačového vidění 8. přednáška: Strojové učení [online]*, (2014), [vid. 2017-01-26]. Dostupné z: http://www.kky.zcu.cz/uploads/courses/mpv/08/lesson08_materialy.pdf.

[19] Saylor Academy, *Correlation and regression [online]*, 2012, [vid. 2016-12-12]. Dostupné z: https://saylordotorg.github.io/text_introductory-statistics/s14-correlation-and-regression.html.

[20] KOWALCZYK, Alexandre, *Linear kernel: Why is it recommended for text classification? [online]*, Říjen 2014, [vid. 2017-01-26]. Dostupné z: https://www.svm-tutorial.com/2014/10/svm-linear-kernel-good-text-classification/.

[21] Algobeans, *Decision trees tutorial [online]*, 2016, [vid. 2017-04-20]. Dostupné z: https://annalyzin.files.wordpress.com/2016/07/decision-trees-titanic-tutorial.png.

[22] YU, Cheng, *Numerical optimization [online]*, [vid. 2017-02-08]. Dostupné z: http://www.jade-cheng.com/au/coalhmm/optimization/.

[23] ADAMI, Christoph, *What is information? [online]*, 2016, [vid. 2017-01-16]. Dostupné z: http://rsta.royalsocietypublishing.org/content/roypta/374/2063/20150230/F2.large.jpg?width=800&height=600&carousel=1.

[24] LAKENS, Daniel, *Always use welch's t-test instead of student's t-test [online]*, January 2015, [vid. 2017-01-11]. Dostupné z: http://daniellakens.blogspot.cz/2015/01/always-use-welchs-t-test-instead-of.html.

[25] ROZENBERG, David, *Úprava nástroje dmvisual [online]*, Master's thesis, České vysoké učení technické v Praze, 2012, [vid. 2017-01-29]. Dostupné z: https://dip.felk.cvut.cz/browse/pdfcache/rozendav_2012bach.pdf.

[26] Plant & Soil Sciences eLibrary, *Chi-square test for goodness of fit in a plant breeding example [online]*, [vid. 2017-01-05]. Dostupné z: https://passel.unl.edu/pages/informationmodule.php?idinformationmodule=1130447119&topicorder=14&maxto=15&minto=1.

[27] KARÁSEK, J., *Citlivost metod pro měření podobnosti kvantitativních proměnných [online]*, Září 2012, [vid. 2017-01-18]. Dostupné z: http://access.feld.cvut.cz/view.php?cisloclanku=2012090003.

[28] AARSHAY, Jain, *A complete tutorial on ridge and lasso regression in python [online]*, Leden 2015, [vid. 2017-03-12]. Dostupné z: https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/.

[29] PANUŠ, Jan, *Evoluční algoritmy v optimallizačních problémech veřejné správy*, Master's thesis, Univerzita Pardubice, 2008, [vid. 2017-02-20], Dostupné z: https://dk.upce.cz/bitstream/handle/10195/30356/text.pdf?sequence=1&isAllowed=y.

[30] BROWNLEE, Jason, *An introduction to feature selection [online]*, October 2014, [vid. 2016-12-10]. Dostupné z: http://machinelearningmastery.com/an-introduction-to-feature-selection.

[31] ———, *Classification and regression trees for machine learning [online]*, Duben 2016, [vid. 2017-02-25]. Dostupné z: http://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/.

[32] FROST, Jim, *Choosing between a nonparametric test and a parametric test [online]*, February 2015, [vid. 2017-03-02]. Dostupné z: http://blog.minitab.com/blog/adventures-in-statistics-2/choosing-between-a-nonparametric-test-and-a-parametric-test.

[33] BECKER, Jonathan, *An introduction to particle swarm optimization (pso) with applications to antenna optimization problems [online]*, Červen 2013, [vid. 2017-02-20]. Dostupné z: http://wirelesstechthoughts.blogspot.cz/2013/06/an-introduction-to-particle-swarm.html.

[34] OBITKO, Marek, *Introduction to genetic algorithms [online]*, [vid. 2017-02-08]. Dostupné z: http://www.obitko.com/tutorials/genetic-algorithms/about.php.

[35] Math and Stats Support Centre, *Pearsonův korelační koeficient [online]*, October 2014, vid. 2017-01-20]. Dostupné z: http://mathstat.econ.muni.cz/media/12657/pear_cor.pdf.

[36] Changing Minds, *Parametric vs. non-parametric tests [online]*, [vid. 2017-03-02]. Dostupné z: http://changingminds.org/explanations/research/analysis/parametric_non-parametric.htm.

[37] Lipo Wang Nina ZHOU, *A modified t-test feature selection method and its application on the hapmap genotype data*, February 2008, Dostupné z: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5054219/. [vid. 2017-01-11].

[38] University of Florida, *The id3 algorithm [online]*, [vid. 2017-02-25]. Dostupné z: https://www.cise.ufl.edu/~ddd/cap6635/Fall-97/Short-papers/2.htm.

[39] University of Washington, *Information gain*, [vid. 2017-01-17]. Dostupné z: https://courses.cs.washington.edu/courses/cse455/10au/notes/InfoGain.pdf.

[40] WINSTON, P., *C4.5 tutorial [online]*, 1992, [vid. 2017-02-25]. Dostupné z: http://www2.cs.uregina.ca/~dbd/cs831/notes/ml/dtrees/c4.5/tutorial.html.

[41] Perseus, *Classification parameter optimization [online]*, 2015, [vid. 2017-02-02]. Dostupné z: http://www.coxdocs.org/lib/exe/fetch.php?media=perseus:user:activities:matrixprocessing:learning:knn.png.

[42] Tutorials Point, *Genetic algorithms - crossover [online]*, [vid. 2017-02-08]. Dostupné z: https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_crossover.htm.

[43] PORKODI, R., *Comparison of filter based feature selection algorithms: An overview [online]*, Department of Computer Science, [vid. 2017-01-22]. Dostupné z: http://ijirts.org/volume2issue2/IJIRTSV2I2034.pdf.

[44] CAPARRINI, Fernando Sancho, *Local search algorithms in netlogo [online]*, 2016, [vid. 2017-04-20]. Dostupné z: http://www.cmbi.ru.nl/redock/images/SimulatedAnnealing.jpg.

[45] Statistics Solutions, *Correlation (pearson, kendall, spearman) [online]*, [vid. 2017-01-30]. Dostupné z: http://www.statisticssolutions.com/correlation-pearson-kendall-spearman/.

[46] Statstutor, *Spearman's correlation [online]*, [vid. 2017-01-21]. Dostupné z: `http://www.statstutor.ac.uk/resources/uploaded/spearmans.pdf`.

Příloha **A**

Seznam použitých zkratek

CART Classification and regression trees

FS Forward selection

FSV Feature selection concave

ID3 Iterative dichotomiser 3

IG Information gain

IP Inflection point selection

IR Information gain ratio

kNN k-nearest neighbor

LASSO Least absolute shrinkage and selection operator

MAP Maximální aposteriorní pravděpodobnost

MI Mutual information

PCA Principal component analysis

PTA Plus ℓ-take away

RFE Recursive feature elimination

RMSE Root-mean-square error

RS Rank selection / Ratio selection

RWS Roulette wheel selection

SDS Sekvenční dopředná selekce

SUS Stochastin universal sampling

SVM Support vector machines

SZE Sekvenční zpětná eliminace

TL Tabu list

TS Threshold selection / Tournament selection

Příloha **B**

Naměřené hodnoty

Tabulka B.1: Úspěšnost Pearsonova koeficientu (SVM/kNN/Bayes)

Datové soubory	**Pearson RS** *(%)*	**Pearson TS** *(%)*	**Pearson FS** *(%)*	**Pearson IP** *(%)*
Kosatce	96.92/94.56/95.74	95.64/96.38/94.81	97.26/95.63/96.39	94.92/94.58/95.72
Druhy aut	73.32/72.56/65.19	75.92/76.44/64.89	70.21/90.58/57.99	69.19/65.47/57.09
Hledání míst	94.78/94.54/87.32	95.76/95.67/72.3	84.71/81.62/85.65	84.63/86.04/85.44
Šachová zakončení	31.65/25.73/22.25	65.22/61.37/23.26	65.15/61.56/22.43	23.16/26.07/22.38
Volby do kongresu	95.38/92.17/94.25	94.77/94.2/92.13	95.06/95.23/95.7	95.58/95.64/95.24
Internetové reklamy	92.45/94.38/92.81	93.32/92.39/96.19	95.45/14.34/94.44	95.48/13.7/93.39
Druhy zvířat	82.45/86.57/89.28	90.44/80.44/96.08	69.45/69.39/68.4	66.61/67.41/64.63
Výsledky piškvorků	72.48/69.44/68.22	80.57/81.2/70.72	69.37/65.62/70.79	69.27/53.9/68.14
Příjem dospělých	83.25/83.56/79.88	81.17/83.53/80.14	79.15/83.72/79.96	78.05/80.94/79.99
Rozdělení ecoli	94.77/97.37/96.32	95.89/97.33/96.26	96.4/97.31/97.23	95.3/97.44/95.64
Srdeční arytmie	54.41/55.44/14.16	53.7/58.09/17.07	53.54/56.02/55.47	53.43/55.01/57.94
Druhy hub	99.97/100/89.83	99.95/99.99/91.64	100/99.45/93.73	99.37/98.59/93.58
Rakovina plic	60.31/46.34/55.76	43.74/50.85/45.56	62.54/64.88/65.21	47.45/60.83/59.02
Kvalita vína	49.4/48.75/47.06	51.3/47.24/45.62	50.57/48.38/49.27	49.51/49.29/47.66
Kategorie dokumentů	33.52/62.97/88.88	41.88/63.87/87.77	27.9/17.87/28.91	17.07/15.43/18.87

Tabulka B.2: Úspěšnost Spearmanova koeficientu (SVM/kNN/Bayes)

Datové soubory	**Spearman RS** *(%)*	**Spearman TS** *(%)*	**Spearman FS** *(%)*	**Spearman IP** *(%)*
Kosatce	94.33/95.43/95.43	96.52/97.13/96.29	97.36/96.53/95.64	97.57/96.87/96.28
Druhy aut	73.49/71.37/64.1	87.86/87.05/65.81	69.92/91.33/57.96	93.69/90.65/56.82
Hledání míst	95.59/95.59/72.14	95.69/95.61/72.15	85.36/82.0/85.72	85.37/83.13/85.76
Šachová zakončení	37.79/32.13/17.15	65.12/61.79/22.89	65.1/61.58/18.84	20.13/20.15/15.45
Volby do kongresu	95.81/91.71/93.71	95.06/93.39/94.37	96.41/96.32/95.11	95.34/93.4/94.86
Internetové reklamy	91.56/92.5/91.54	93.0/94.98/96.43	94.32/13.77/94.33	94.22/14.24/94.21
Druhy zvířat	82.63/85.75/89.54	45.08/33.29/89.74	59.34/54.13/60.79	61.8/53.03/60.69
Výsledky piškvorků	69.75/60.36/69.52	78.9/80.64/70.5	70.27/62.87/69.75	68.57/61.37/69.95
Příjem dospělých	82.61/83.25/79.73	80.99/83.33/80.1	82.0/83.44/79.39	81.9/80.99/79.14
Rozdělení ecoli	96.08/97.65/95.62	96.44/96.26/95.7	97.12/97.07/96.14	93.75/95.97/95.37
Srdeční arytmie	55.92/55.72/16.38	56.28/55.85/16.47	54.95/55.88/56.44	51.83/54.17/24.86
Druhy hub	99.97/100/89.91	99.99/99.98/91.81	98.6/98.64/94.64	98.53/98.48/95.09
Rakovina plic	64.74/48.14/52.24	48.19/56.36/52.36	56.15/60.44/61.62	53.01/60.81/61.99
Kvalita vína	49.5/48.99/47.47	50.71/47.93/45.56	50.13/48.33/49.19	50.21/49.02/48.32
Kategorie dokumentů	39.06/62.3/88.3	42.67/63.44/88.71	59.04/27.37/55.63	17.23/15.86/18.19

Tabulka B.3: Úspěšnost Kendallova koeficientu (SVM/kNN/Bayes)

Datové soubory	**Kendall RS** *(%)*	**Kendall TS** *(%)*	**Kendall FS** *(%)*	**Kendall IP** *(%)*
Kosatce	97.99/94.66/96.64	97.32/95.21/96.27	96.18/97.04/96.41	96.06/95.25/95.64
Druhy aut	73.79/70.43/63.44	87.62/86.63/65.78	69.78/90.37/58.24	92.52/60.18/57.05
Hledání míst	95.68/95.68/72.4	95.61/95.64/72.27	84.71/84.05/85.47	84.62/76.82/81.69
Šachová zakončení	37.57/32.0/17.02	65.12/61.89/22.95	65.34/61.52/19.19	20.23/19.21/15.52
Volby do kongresu	95.61/91.5/94.48	94.6/94.05/94.66	96.17/95.42/95.81	94.41/94.96/95.7
Internetové reklamy	91.76/89.64/91.18	92.86/95.41/96.53	94.34/14.43/94.48	94.32/13.67/94.62
Druhy zvířat	85.8/85.17/90.01	50.54/36.14/93.32	67.09/86.65/68.43	55.14/49.81/57.02
Výsledky piškvorků	69.9/60.92/69.72	80.22/81.83/70.65	70.12/66.89/70.23	68.59/52.55/68.61
Příjem dospělých	82.57/83.38/79.42	81.27/83.47/80.29	81.73/83.44/79.25	81.72/80.44/79.32
Rozdělení ecoli	95.86/96.41/96.44	96.33/95.53/95.19	96.21/97.07/96.02	89.87/96.95/95.58
Srdeční arytmie	54.41/54.39/15.92	55.55/59.15/18.2	52.51/55.2/55.8	54.89/56.22/33.47
Druhy hub	99.98/100/89.72	99.96/99.98/91.81	100/100/94.28	98.5/98.43/94.41
Rakovina plic	44.75/46.14/49.55	42.13/58.25/33.81	59.95/61.02/56.03	57.84/53.82/71.74
Kvalita vína	49.55/49.12/47.22	51.74/47.37/45.3	50.06/49.3/49.19	49.7/48.98/47.55
Kategorie dokumentů	39.93/62.89/88.8	36.75/66.35/88.63	28.73/18.09/29.85	16.21/16.66/17.77

Tabulka B.4: Úspěšnost F-testu (SVM/kNN/Bayes)

Datové soubory	**F-test RS** *(%)*	**F-test TS** *(%)*	**F-test FS** *(%)*	**F-test IP** *(%)*
Kosatce	95,89/96,01/95,93	94,6/96,55/94,42	96,85/96,03/96,05	97,06/95,78/96,86
Druhy aut	73,9/71,82/63,99	77,08/75,5/64,92	69,39/90,84/66,75	69,58/64,62/63,34
Hledání míst	94,76/94,68/87,34	93,92/93,95/89,12	84,82/84,01/85,61	84,76/83,4/85,64
Šachová zakončení	32,95/22,6/20,52	44,55/39,51/22,08	65,26/61,75/22,76	65,33/8,63/15,3
Volby do kongresu	94,86/92,34/94,45	95,0/92,17/93,83	95,94/95,29/95,38	96,24/92,11/95,85
Internetové reklamy	92,7/93,84/92,91	93,33/94,39/95,25	95,31/14,02/94,27	93,59/13,77/94,33
Druhy zvířat	43,7/35,69/75,98	80,26/82,17/81,41	49,25/38,07/79,77	42,21/35,05/36,36
Výsledky piškvorků	71,83/60,62/67,23	70,54/61,01/69,64	69,64/62,03/70,87	68,1/58,21/70,32
Příjem dospělých	82,96/83,54/79,81	83,05/83,36/79,93	79,15/83,86/80,06	78,09/80,61/79,84
Rozdělení ecoli	95,28/97,14/92,73	96,67/96,99/93,52	97,34/96,53/96,18	95,64/97,29/97,05
Srdeční arytmie	53,68/54,22/15,74	54,86/64,11/27,3	53,64/57,06/62,15	51,87/54,89/56,79
Druhy hub	100,0/100,0/89,63	99,88/99,97/89,82	99,95/99,47/93,58	99,38/98,58/93,7
Rakovina plic	59,87/55,78/47,35	55,82/58,53/43,8	61,49/70,1/63,91	45,13/62,75/56,81
Kvalita vína	49,63/48,85/47,31	49,42/48,95/47,01	51,34/50,81/49,36	50,17/51,48/47,15
Kategorie dokumentů	52,67/85,77/92,12	77,26/74,59/69,26	38,99/41,69/40,27	36,83/37,05/37,33

Tabulka B.5: Úspěšnost T-testu (SVM/kNN/Bayes)

Datové soubory	**T-test RS** *(%)*	**T-test TS** *(%)*	**T-test FS** *(%)*	**T-test IP** *(%)*
Kosatce	80,49/74,06/78,01	93,74/95,01/87,34	96,07/96,32/96,23	68,38/63,85/75,68
Druhy aut	67,5/64,95/70,68	69,9/62,84/70,18	70,91/63,08/70,93	69,6/62,52/70,1
Hledání míst	95,5/95,34/72,27	95,51/95,38/72,1	71,51/63,03/71,48	49,42/24,58/30,0
Šachová zakončení	27,68/15,34/14,82	65,4/61,97/22,97	65,16/61,67/13,71	27,79/8,22/12,34
Volby do kongresu	95,45/92,84/94,53	95,76/92,22/93,54	94,8/94,71/95,74	96,02/92,6/95,09
Internetové reklamy	92,36/94,69/64,8	95,14/88,19/91,0	94,91/88,57/90,53	95,03/83,03/90,53
Druhy zvířat	46,96/35,97/91,32	45,25/30,71/46,75	45,05/33,64/69,09	42,07/32,04/49,04
Výsledky piškvorků	69,71/62,34/69,0	79,48/83,03/70,15	70,62/67,33/70,43	69,97/57,2/68,88
Příjem dospělých	75,81/74,39/78,0	75,97/74,24/78,86	79,43/76,78/80,09	76,0/74,0/77,9
Rozdělení ecoli	96,07/96,84/95,16	74,57/97,43/95,5	97,24/98,35/96,82	89,43/97,65/95,7
Srdeční arytmie	52,58/54,49/18,3	54,62/56,03/57,24	55,96/56,14/62,2	53,01/54,54/62,07
Druhy hub	99,94/99,94/86,67	99,96/99,95/86,94	99,52/98,56/95,47	99,56/98,64/94,7
Rakovina plic	37,89/35,74/52,25	38,74/42,25/47,73	56,03/42,52/41,41	46,26/44,81/71,12
Kvalita vína	58,31/46,14/46,58	58,04/46,13/46,33	58,36/49,1/49,16	49,34/46,19/47,91
Kategorie dokumentů	8,48/10,76/14,1	14,55/85,48/89,79	8,87/10,96/9,99	9,24/10,25/9,06

Tabulka B.6: Úspěšnost Chi-kvadrát testu (SVM/kNN/Bayes)

Datové soubory	**Chi-kv. RS** *(%)*	**Chi-kv. TS** *(%)*	**Chi-kv. FS** *(%)*	**Chi-kv. IP** *(%)*
Kosatce	95,62/96,61/96,0	97,27/96,27/96,6	96,15/96,68/96,63	95,76/95,44/96,8
Druhy aut	73,38/72,06/64,34	76,17/75,76/63,94	69,78/90,64/67,86	69,42/66,41/63,36
Hledání míst	95,73/95,62/72,13	85,82/95,27/72,22	93,51/95,6/72,27	85,87/95,36/72,1
Šachová zakončení	37,89/31,97/17,4	44,42/40,15/21,97	64,98/61,55/21,06	23,29/19,15/19,09
Volby do kongresu	95,43/91,39/94,26	95,02/92,63/93,39	96,09/95,34/94,96	
Internetové reklamy	92,65/93,56/92,39	92,36/89,08/90,68	96,23/91,36/90,67	95,78/88,97/90,37
Druhy zvířat	47,87/39,07/88,22	49,99/37,58/93,38	47,88/37,3/54,9	45,72/34,72/35,84
Výsledky piškvorků	71,19/62,4/69,77	71,06/60,31/70,63	70,51/64,89/69,73	67,46/55,96/70,33
Příjem dospělých	75,89/78,41/79,57	81,04/75,45/79,11	83,15/83,1/79,48	83,19/82,97/79,57
Rozdělení ecoli	96,87/96,95/94,94	74,22/96,36/96,4	96,73/96,87/96,11	95,64/96,69/96,11
Srdeční arytmie	N/A	N/A	N/A	N/A
Druhy hub	99,98/99,93/91,64	99,91/99,97/92,5	93,77/99,86/87,71	87,15/76,49/83,94
Rakovina plic	56,68/46,19/48,44	63,32/64,05/65,95	57,82/65,18/75,74	96,07/94,02/95,83
Kvalita vína	57,82/46,18/48,48	56,36/45,28/44,7	57,85/46,84/45,93	45,54/44,06/44,97
Kategorie dokumentů	52,44/85,89/91,98	81,7/83,2/85,51	38,32/53,4/46,86	35,21/37,82/35,53

Tabulka B.7: Úspěšnost Vzájemné informace (SVM/kNN/Bayes)

Datové soubory	**Vz. inf. RS** *(%)*	**Vz. inf. TS** *(%)*	**Vz. inf. FS** *(%)*	**Vz. inf. IP***(%)*
Kosatce	96,81/95,97/96,33	98,62/95,53/97,16	95,35/96,15/96,49	96,18/95,61/96,67
Druhy aut	95,89/96,01/95,93	93,65/91,93/67,83	94,01/90,85/57,85	77,09/74,99/57,52
Hledání míst	95,67/95,54/72,2	95,69/95,54/72,21	95,14/95,54/86,81	84,73/78,88/75,91
Šachová zakončení	33,18/21,92/20,31	65,23/61,67/22,83	65,1/61,72/22,58	65,1/8,41/13,74
Volby do kongresu	94,41/91,81/94,46	95,4/94,06/94,09	96,13/94,78/95,81	95,48/95,24/95,1
Internetové reklamy	92,52/93,15/91,3	93,6/94,61/96,53	94,43/13,84/94,46	94,32/14,03/94,47
Druhy zvířat	45,26/34,42/88,4	50,52/38,46/96,82	80,93/76,51/87,42	85,98/75,58/89,18
Výsledky piškvorků	76,36/67,21/70,92	80,69/77,57/69,46	69,3/57,73/70,52	79,46/60,12/69,64
Příjem dospělých	75,95/78,38/79,32	75,87/78,48/79,55	74,93/77,8/79,28	76,1/77,8/79,17
Rozdělení ecoli	95,95/96,36/94,29	96,67/96,9/94,9	96,44/97,08/96,72	89,63/97,64/95,04
Srdeční arytmie	56,13/56,07/58,4	53,14/57,08/21,67	55,1/56,22/54,63	54,76/54,85/52,89
Druhy hub	99,96/99,98/90,06	100,0/99,97/89,12	99,95/99,4/93,34	99,41/99,41/93,73
Rakovina plic	55,04/48,13/50,11	41,72/48,87/51,43	62,61/71,83/74,91	46,23/57,96/70,81
Kvalita vína	51,57/47,39/43,63	55,98/45,44/45,07	52,2/49,34/46,48	49,31/49,93/46,5
Kategorie dokumentů	52,48/87,08/92,68	70,24/88,5/92,98	45,92/40,49/46,61	35,05/37,36/18,83

Tabulka B.8: Úspěšnost wrapper metod

Datové soubory	Sekvenční dopředná selekce	Sekvenční zpětná eliminace	Genetický algoritmus
Kosatce	100,0/97,92/97,87	97,37/98,0/96,23	100,0/100,0/100,0
Druhy aut	73,63/70,02/73,1	93,71/90,18/74,21	88,4/82,49/74,21
Hledání míst	95,61/95,86/93,12	96,18/96,1/80,87	96,19/95,87/81,85
Šachová zakončení	65,39/62,02/23,7	64,64/62,16/23,21	47,26/62,75/23,73
Volby do kongresu	98,51/98,64/98,08	98,46/96,69/97,1	97,83/98,57/99,31
Internetové reklamy	N/A	95,02/97,6/81,99	93,18/95,21/76,6
Druhy zvířat	96,15/90,32/100,0	100,0/100,0/100,0	100,0/96,55/100,0
Výsledky piškvorků	72,13/68,82/70,72	81,75/85,22/73,47	82,51/80,06/74,67
Příjem dospělých	85,96/85,97/80,13	86,46/84,24/82,04	82,86/80,32/80,18
Rozdělení ecoli	99,0/98,28/98,17	97,94/100,0/98,98	100,0/100,0/98,28
Srdeční arytmie	66,67/71,14/66,25	65,71/66,67/28,1	60,58/65,0/22,29
Druhy hub	100,0/100,0/97,34	100,0/100,0/96,57	100,0/100,0/93,89
Rakovina plic	75,0/100,0/87,5	85,71/80,0/100,0	100,0/75,0/100,0
Kvalita vína	59,04/53,1/52,55	58,39/53,13/52,18	57,39/51,98/49,12
Kategorie dokumentů	72,26/66,29/69,32	40,0/91,43/96,14	44,55/76,12/81,84

Tabulka B.9: Úspěšnost embedded metod

Datové soubory	**Rozhodovací strom** *(%)*	**RFE** *(%)*	ℓ_1 **SVM** *(%)*	**LASSO**
Kosatce	93,33	97,96	85,11	-3,27
Druhy aut	95,49	70,23	69,03	-0,02
Hledání míst	96,93	N/A	94,87	0,0
Šachová zakončení	80,59	29,6	25,46	-0,02
Volby do kongresu	95,08	94,67	97,06	-1,71
Internetové reklamy	97,31	N/A	97,55	-0,05
Druhy zvířat	96,67	96,67	86,49	-6,77
Výsledky piškvorků	85,45	70,27	68,23	-2,29
Příjem dospělých	81,72	N/A	82,43	0,0
Rozdělení ecoli	90,62	99,1	93,07	-0,03
Srdeční arytmie	59,54	67,74	60,47	-1,9
Druhy hub	100	99,0	100,0	0,0
Rakovina plic	60,0	70,0	54,55	-3,06
Kvalita vína	57,78	51,59	52,41	0,0
Kategorie dokumentů	87,37	N/A	93,58	-0,15

Tabulka B.10: Úspěšnost hrubé síly

Datové soubory	**Hrubá síla** *(%)*
Kosatce	100.0/100.0/98.11
Druhy aut	94.44/90.86/74.69
Hledání míst	N/A
Šachová zakončení	64.69/61.88/23.52
Volby do kongresu	100.0/99.36/100.0
Internetové reklamy	N/A
Druhy zvířat	100.0/100.0/100.0
Výsledky piškvorků	82.75/85.17/76.76
Příjem dospělých	N/A
Rozdělení ecoli	100.0/100.0/99.09
Srdeční arytmie	N/A
Druhy hub	N/A
Rakovina plic	N/A
Kvalita vína	60.3/54.7/54.33
Kategorie dokumentů	N/A

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
  - `data` ..... adresář s testovacími daty
  - `feature_selection.py` ..... implementace selekce příznaků
  - `method` ..... adresář s jednotlivými metodami
  - `README` ..... popis spuštění implementace
  - `results` ..... adresář s výstupy implementace
  - `schedule.bat` ..... spustitelný skript pro Windows
  - `schedule.sh` ..... spustitelný skript pro Unix
  - `settings` ..... adresář s konfiguračním souborem
  - `util` ..... adresář s pomocnými třídami
- `src`
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF