Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matej Kopar

Klasifikacija virusnih zaporedij s strojnim učenjem

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Tomaž Curk

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Uporabite in ovrednotite uspešnost metod strojnega učenja za klasifikacijo genomskih zaporedij virusov v pripadajoče taksonomske enote. Iz podatkovne zbirke NCBI pridobite genomska zaporedja ter podatke o taksonomski pripadnosti posameznih virusov. Določite najbolj ustrezen atributni opis genomskega zaporedja, ki omogoča uspešno modeliranje. Preučite in ovrednotite uporabnost predznanja o odnosih med taksonomskimi enotami, kar je zapisano v filogenetskem drevesu (ontologiji). Primerjajte uspešnost standardnih metod za večznačno napovedovanje in metod, ki eksplicitno modelirajo strukturo (ontologijo) relacij med razredi.

Implement and evaluate standard machine learning methods for the classification of viral genome sequences into corresponding taxonomy groups. Obtain genome sequence and taxonomy membership data on viruses from the NCBI database. Empirically determine the most appropriate feature representation of genomic sequences to be used for modeling. Apply and evaluate the impact of previous knowledge on phylogenetic relationships on the predictive performance. Evaluate and compare the performance of standard multi-label approaches with specialized approaches to structured class prediction.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Matej Kopar, z vpisno številko **63120219**, sem avtor diplomskega dela z naslovom:

Klasifikacija virusnih zaporedij s strojnim učenjem

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Tomaža Curka,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 15. september 2015 Podpis avtorja:

Iskreno se zahvaljujem svojim staršem in Manci za podporo ter potrpežljivost. Posebna zahvala pa gre tudi mentorju doc. dr. Tomažu Curku za vso podporo, pomoč in usmeritve pri izdelavi diplomskega dela.

Okiju.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
SVM	support vector machine	metoda podpornih vektorjev
NCBI	National Center for Biotechnology Information	center za biotehnološke podatke
CDS	coding sequence	kodirajoči del sekvence
PS	PyStruct	metoda za strukturirano napovedovanje
RF	random forest	naključni gozd
LR	logistic regression	logistična regresija
DC	dummy classifier	večinski klasifikator
DNA	deoxyribonucleic acid	deoksiribonukleinska kislina
A	adenine	adenin
T	thymine	timin
C	cytosine	citozin
G	guanine	gvanin
NGS	next generation sequencing	visokozmogljivo sekvenciranje

Povzetek

V diplomskem delu smo uporabili različne metode strojnega učenja za uvrščanje virusnih zaporedij v ustrezne taksonomske skupine. Z dostopanjem do podatkovne zbirke NCBI, ki hrani biološke in biotehnološke podatke, smo najprej sestavili celotno taksonomsko strukturo znanih virusnih zaporedij. Podatke smo ustrezno filtrirali in tako zgradili množico učnih primerov. Nato smo uporabili klasične metode strojnega učenja in metodo strukturiranega napovedovanja in ovrednotili uspešnost napovedovanja v taksonomske skupine. V delu smo preučili, kateri načini opisovanja genomskih zaporedij so najprimernejši. Opis genomskih zaporedij s k-terkami ne zajame vseh podrobnosti genomov, zato so najboljši doseženi rezultati le nekoliko boljši od večinskega klasifikatorja. Predznanje o evolucijski povezanosti taksonomskih skupin nekoliko izboljša napovedi modelov, ki to znanje lahko uporabijo.

Ključne besede: strojno učenje, klasifikacija, metoda podpornih vektorjev, naključni gozdovi, virusna zaporedja, strukturirano strojno učenje.

Abstract

In this diploma thesis our goal was to classify viral sequences into taxonomic groups by using different machine learning methods. We assembled the taxonomic structure by collecting data from NCBI web site. To clean the data we applied several filtering steps. We then evaluated the predictive performance of classical and structured machine learning methods on the task of classification in taxonomy groups. We wanted to determine the most suitable way to describe genomic sequences. Using k-mers to describe the genomic composition yielded poor predictive models, with best performance slightly above the performance of the majority classifier. Methods, which are able to use prior knowledge on the taxonomic relationships between classes, performed slightly better than methods, which did not use such information.

Keywords: machine learning, classification, support vector machine, random forest, viral sequences, structured machine learning.

Poglavje 1

Uvod

Virusi so del vseh ekosistemov, kjer mikrobi poganjajo ključne energetske in osnovne transformacije, vključno z oceani, človeško populacijo ter industrijskimi fermentacijami [9].

Ljudje se od nekdaj spopadamo z virusi, saj nam v večini primerov leti predstavljajo grožnjo. Virusi so zelo majhni patogeni. Ker sami nimajo celičnih mehanizmov za lastno reprodukcijo, se reproducirajo v celicah živih bitij. Okužijo lahko vse vrste organizmov - tako arheje kot tudi bakterije, glive, rastline, živali in ljudi [17].

Ta prabitja so na našem planetu od samega začetka življenja. Glede na to, da so nekatere vrste preživele vse do danes, so se morale na drugačno življenje prilagajati z mutacijami. Zaradi mutacij se pojavljajo drugačna genomska zaporedja, lahko tudi povsem novi virusi, kar predstavlja glavno težavo pri uvrščanju. V večini primerov s sekvenciranjem pridobimo zgolj delčke genoma določenega virusa, ki ga želimo čim natančneje uvrstiti v taksonomsko skupino.

Področje klasifikacije virusov je relativno aktivno, razvijajo se novi pristopi ter aplikacije, ki na podlagi krajših delov genoma uvrščajo že zelo uspešno. Eden najnovejših programov je naprimer VirSorter, ki uspe s 100% natančnostjo in občutljivostjo $\geq 95\%$ identificirati virusne signale v združenih zaporedjih dolgih zgolj nekaj 10K baz [9]. Zanimiv pripomoček za detekcijo

virusov je tudi program VirFind. Z njim lahko iščemo po neznanih genomskih zaporedjih, določimo in filtriramo vhodne podatke ter iz njih pripravimo datoteke za nadaljnjo obdelavo. Algoritem za program je bil razvit z uporabo NGS (angl. next generation sequencing) [2]. Z NGS lahko hitro resekvenciramo celoten genom rastline ali pa vzorčimo transkriptome bolj učinkovito, globlje in cenejše [10].

Implementacija tako obsežnega programa bi bila časovno zahtevna in tako bolj primerna za višje stopnje. Zato smo se odločili, da za diplomsko nalogo vzamemo zgolj del celotnega postopka - klasifikacija virusov v taksonomske skupine.

Iz podatkov, ki so na voljo v podatkovni bazi NCBI, gradimo napovedne modele, s katerimi klasificiramo viruse v taksonomske skupine. Za vsak virus, kateremu je določen celoten genom, iz podatkovne baze NCBI najprej pridobimo njegovo taksonomsko pripadnost (red, družino, poddružino, rod in vrsto). Nato za vsak virus pridobimo ustrezne atribute iz sekvenčnih podatkov, ki jih imamo na voljo. V našem primeru smo se odločili za opis genomskega zaporedja s *k-terkami*, kjer je $k \in [1, 7]$ ter za opis frekvenc pojavitve posameznih kodonov (to je, zaporedij treh baz, ki se pojavijo v kodirajočem delu genov).

Eden večjih izzivov uporabe pristopov podatkovnega rudarjenja, še posebej na bioloških podatkih, je priprava ustreznega nabora atributov. Biološki podatki običajno vsebujejo precej šuma. Zaradi tega so lahko zelo specifični, kar oteži iskanje zakonitosti v njih. Za uspešnejše rezultate modeliranja moramo zato najprej določiti najbolj informativen nabor atributov.

1.1 Cilji diplomske naloge

V diplomski nalogi želimo odgovoriti na naslednja vprašanja:

1. Ali lahko z metodami strojnega učenja zgradimo napovedni model za uvrščanje virusnih zaporedij v taksonomijo?
2. Kateri opis genomskih zaporedij je najbolj ustrezen?
3. Ali je metoda za strukturirano napovedovanje, implementirana v PyStruct, primerna za uvrščanje virusnih zaporedij?
4. Ali predznanje o taksonomski povezanosti virusov pirpomore k boljši napovedi?

Poglavje 2

Podatki

Podatki so pridobljeni iz podatkovne baze na spletni strani NCBI , ki vsebuje več prosto dostopnih podatkovnih baz z biotehnološkimi in biomedicinskimi podatki. Za zapis genomskih zaporedij uporabljajo format GenBank. Baze podatkov so razdeljene na več sklopov - nukleotidi, taksonomije, proteini ipd. Za naš problem smo črpali podatke iz podatkovne baze o nukleotidih [14].

Podatke smo prenesli v formatu GenBank. GenBank je prosto dostopna podatkovna baza sekvenc, pripisov njihovih funkcij in njihovih proteinskih translacij. NCBI skrbi za ažurnost podatkov, ki jih sprejema od laboratorijev iz vsega sveta, in tako vsebuje podatke o že več kot 100000 različnih organizmih. V zadnjih 30 letih je to postala najpomembnejša podatkovna baza za raziskovanje v skoraj vseh bioloških sferah [13]. V našem delu smo uporabili ta format zaradi boljše strukturiranosti podatkov. Za implementacijo smo uporabili tudi knjižnico BioPython [1]. Knjižnica ima implementiran bralnik formata GenBank, kar omogoča enostaven dostop do vseh podatkov v formatu in njihovo obdelavo.

Iz pridobljenih podatkov smo pripravili taksonomsko strukturo glede na družine, rodove in vrste virusov. Vseh zapisov je bilo 4909 (6. september 2015), od tega je bilo primerov v listih (pred pripravo) 2774, vseh taksonomskih enot pa 558.

2.1 Priprava taksonomije virusov

Iz pridobljenih podatkov smo pripravili taksonomsko strukturo celotnega nabora podatkov. Zaradi izredno velike razpršenosti podatkov, predvsem pa velikega števila razredov je bilo potrebno del podatkov urediti, del pa združiti v večje skupine.

2.1.1 Urejanje podatkov

Po pregledu taksonomske strukture podatkov, s katerimi operiramo, smo naredili seznam razredov, katere smo izločili v končnem naboru: *bakterije (angl. bacteria), neklasificirane (angl. unclassified)* ter *nerazvrščene (angl. unassigned)*. Ker smo želeli uvrščati viruse, smo izključili bakterije, saj bi nam podatki o bakterijah vnašali šum v nabor podatkov. Taksonomske skupine bakterij smo popolnoma odstranili. Delež bakterij je skupno predstavljal $0,47\%$ podatkov.

Za izločitev neklasificiranih ter nerazvrščenih razredov pa smo se odločili že zaradi imen, ker ne želimo napovedovati razrede, katerih na NCBI še niso uspeli nedvoumno določiti. Takšnih primerov ne odstranimo iz podatkovnega nabora ampak jih razvrstimo v starše. Večina nerazvrščenih ter neklasificiranih primerov se je znašla prav v končnih vozliščih in nam s tem zelo razširila število razredov. Ker bi nam v nadaljevanju to otežilo napovedovanje, smo jih odstranili. Pred urejanjem je bilo vseh razredov 558, kar smo znižali na 454.

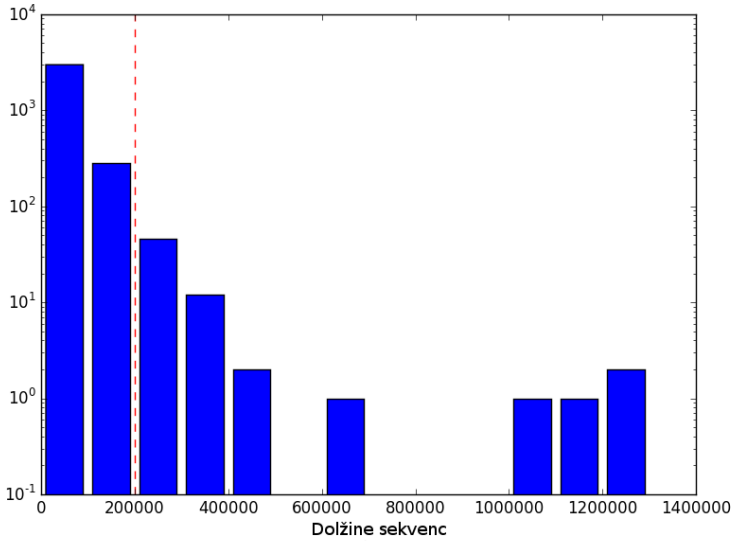
2.1.2 Filtriranje glede na število primerov v listih ter glede na dolžino sekvence primera

V podatkovnem naboru je bilo nekaj razredov, ki so bili zastopani z izredno malo primeri. Ker bi bila napoved z nizkim številom primerov težja in posledično uspešnost slabša, smo število primerov v razredu omejili. Razrede z manj kot 10 primeri smo odstranili in jih prestavili v novo ustvarjeni razred

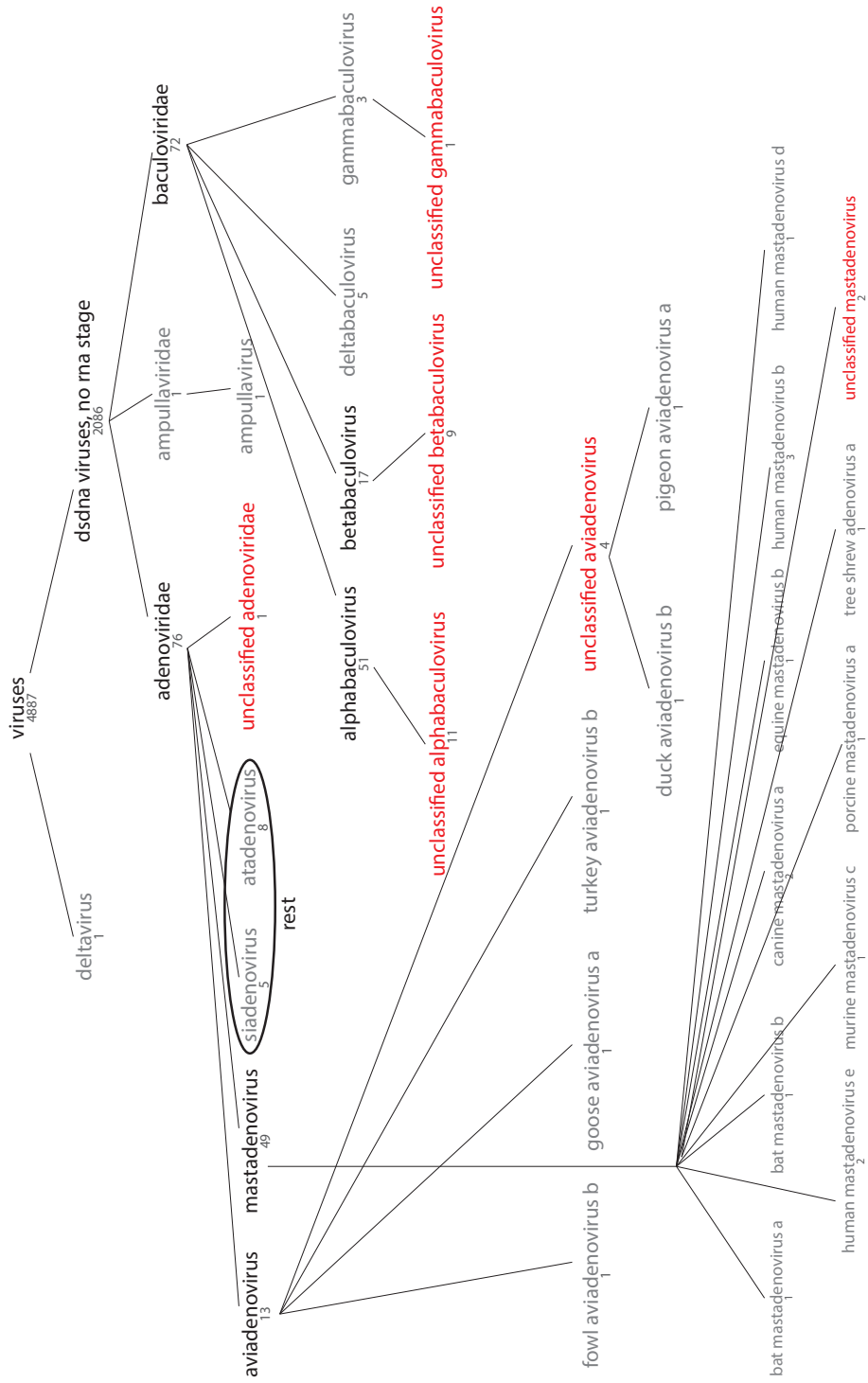
ostali (angl. rest). Po združevanju razredov v nov razred *ostali*, smo implementirali še odstranjevanje edincev. Tako smo zmanjšali globino določene veje in število napovednih razredov.

Ker obstajajo tudi primeri, kjer so sekvence izredno dolge (tudi po milijon znakov), smo nastavili še zgornjo mejo dolžin sekvenc. Iz slike 2.1 je razvidno, da primeri z daljšimi sekvencami od 200000 baz predstavljajo $1,99\%$ vseh primerov v listih.

Po vseh zgoraj opisanih postopkih (združevanjem premajhnih razredov, odstranjevanjem edincev in ponovitvijo teh dveh korakov po odstranjevanju predolgih sekvenc) smo število razredov zmanjšali na 120. Urejanje celotne taksonomije iz začetnega v končno stanje je prikazano na sliki 2.2. Meja za obstanek razreda je vsaj 10 elementov. Vozlišča, ki so zastopana v premajhnem številu (npr. *atadenovirus* in *siadenovirus*) smo prestavili v novo vozlišče *ostali* (angl. *rest*). Znotraj *aviadenovirus* smo najprej odstranili *uncalssified aviadenovirus*. To je korak urejanja, ki ga opisujemo v 2.1.1. Nato smo začeli z združevanjem vozlišč, ki niso zastopana v dovolj velikem številu. Vozlišča *fowl aviadenovirus b*, *goose aviadenovirus a* ter ostala, ki imajo premalo elementov, smo najprej prestavili v novo vozlišče *ostali*. Ker je vozlišče *ostali* znotraj *aviadenovirus* edinec, smo ga odstranili in s tem zmanjšali globino.



Slika 2.1: Graf distribucije dolžin sekvenc. Na x-osi so dolžine sekvenc, na y-osi pa frekvence pojavitev določene dolžine. Graf je narisan v logaritemski lestvici. Rdeča črta prikazuje najdaljše še obravnavano genomsko zaporedje.



Slika 2.2: Delna struktura taksonomije. Črna vozlišča ostanejo, siva so odstranjena zaradi premalo primerov, rdeča pa zaradi kriterijev urejanja podatkov.

2.2 Priprava genomskih zaporedij virusov

Podatke, ki smo jih pridobili do sedaj, je bilo potrebno dodatno pretvoriti:

1. pretvoriti genomska zaporedja v atributni zapis,
2. pretvoriti razrede v večznačni zapis,
3. pripraviti podatke za uporabo v napovednih modelih.

2.2.1 Pretvorba genomskih zaporedij v atributni zapis

Priprava kvalitetnih atributov je ključnega pomena pri strojnem učenju. Za začetek smo si pripravili atribute s štetjem *k-terk* znotraj vsake sekvence, kjer je $k = [1, 7]$. Šteli smo prekrivajoče *k-terke* nukleotidov *A, T, C* in *G*. Tako smo dobili strukturo *k-terk* v določeni sekvenci. Učenje je tako temeljilo na primerjavi in opisovanju podobnosti sestave genomov. Ker smo želeli odstraniti vpliv velikosti sekvenc, smo vsako od vrstic tudi normalizirali tako, da smo vse elemente delili z vsoto vrstice.

$$\vec{v} = \frac{\vec{v}}{\sum \vec{v}} \tag{2.1}$$

Normalizacija delno omili razlike, ki nastanejo zaradi različnih velikosti sekvenc oz. virusnih genomov.

Primer. Imamo virus A, ki ima sekvenco dolžine 100, razporeditev nukleotidov pa je enakomerna (torej vsak od *k-terk* zavzema delež 25%) in virus B, ki ima sekvenco dolžine 1000 z enako razporeditvijo nukleotidov kot A.

$$\begin{bmatrix} 25 & 25 & 25 & 25 \\ 250 & 250 & 250 & 250 \end{bmatrix}$$

Brez normalizacije vrstice, bi v metodo skaliranja vključili zgornjo matriko. Rezultat oz. izhod pa bi bil naslednji:

$$\begin{bmatrix} 0,1 & 0,1 & 0,1 & 0,1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Na ta način smo porušili prvotno razmerje *k-terk*, saj matrika trdi, da si virusa glede strukture *k-terk* nista podobna. To seveda ne drži, saj vemo, da je razporeditev enaka.

Normalizacija po vrsticah s formulo 2.1, pred skaliranjem podatkov nam da naslednjo matriko:

$$\begin{bmatrix} 0,25 & 0,25 & 0,25 & 0,25 \\ 0,25 & 0,25 & 0,25 & 0,25 \end{bmatrix}$$

Ker sta vrstici pred skaliranjem identični, bosta taki tudi po skaliranju. Tako smo zagotovili, da se začetna razporeditev *k-terk* ohrani. Posledično so tudi rezultati napovednih algoritmov bolj točni. □

Različni organizmi in celo geni istega organizma, se lahko zelo razlikujejo glede na sestavo kodonov. Kodon je zaporedje treh baz, ki se nahaja v kodirajočem delu gena, in določa aminokislino, ki se bo pojavila v proteinu. Protein je tako zaporedje aminokislin, ki ga določa genski zapis. Več različnih kodonov lahko določa isto aminokislino. Tovrstne razlike v preferenci izbire med redundantnimi kodoni so posledica evolucijskih procesov (selekcija, mutacija, kot tudi genska izguba) [8]. Zgodovina evolucije določenega genoma je tako tesno povezana s sestavo kodonov. Zato smo med atribute vključili tudi takšne, s katerimi opišemo preferenco do posameznih kodonov.

Pogostost kodonov smo prebrali iz zapisov kodirajočih regij CDS (angl. coding sequence). Podatke o CDS pa smo pridobili v formatu GenBank. Kodone štejemo z naslednjo psevdokodo:

```
1  data = []
2  for each genome G_i:
3    kmer_freqs = {}
4    for each gene g_i in genome G_i:
5      CDS_seq = concatenate_all_CDS(g_i)
6      while CDS_seq:
7        codon = CDS_seq[:3]
8        assert len(codon) == 3
9        kmer_freqs[codon] = kmers_freqs.get(codon, 0) + 1
10       CDS_seq = CDS_seq[3:]
11   data.append(correctly_order(kmer_freqs))
```

Premikamo se po 3 baze v zaporedju CDS posameznega gena in štejemo pojavitve trojk. Frekvence pojavitev hranimo v slovarju *kmer_freqs*. Trojke nato v metodi *correctly_order* ustrezno uredimo, da ustrezajo zgolj naboru trojk iz nukleotidov *A, T, C* in *G* ter jih dodamo v spremenljivko *data*. Tudi te podatke, tako kot podatke o *k-terkah*, normaliziramo po vrsticah s formulo 2.1.

V atributni zapis dodamo še dolžino celotne sekvence. S tem podatkom učnim algoritmom podamo informacijo o velikosti genoma, kar smo sicer z normalizacijo *k-terk* implicitno odstranili.

2.2.2 Pretvorba v večznačni zapis

V naši problemski domeni smo z več iteracijami, filtriranji in urejanjem podatkov uspeli zmanjšati število razredov. Končnih razredov je 120, zato binarna klasifikacija ne pride v poštev. Ker smo želeli napovedovati oz. pri učenju uporabiti tudi celotno taksonomijo, smo vse razrede pretvorili v večznačni zapis (*angl. multilabel*).

```
1  def get_path_row(node, path_attributes):
2      path = node.split("->")[1:]
3      if "rest" in path:
4          if "rest" != path[-1]:
5              ValueError("rest is not list node.")
6          path[-1] = path[-2] + "->" + path[-1]
7
8          vector = np.array([1 if attr in path else 0
9                             for attr in path_attributes],
10                            dtype=np.float32)
11
12     if sum(vector) != len(path):
13         print "problems in get_path_row..."
14
15     return vector
```

Zaradi težav z uporabo metode *MultiLabelBinarizer* iz knjižnice *scikit-learn*, smo implementirali lastno metodo za pretvorbo v večznačni zapis. Metoda na podlagi taksonomske pripadnosti ustvari vektor, kjer stolpci predstavljajo vsa vozlišča. Za vsako prečkanje določenega notranjega vozlišča v taksonomski pripadnosti virusa, to označimo v spremenljivki *vector* s številom 1.

Primer. Če primer *virus - dsdna virus* predstavlja razred 1, primer *virus - ssrna virus* pa razred 2, je razredni vektor v tem primeru:

$$\begin{bmatrix} 1 & 2 \end{bmatrix}$$

Po pretvobi v večznačni zapis pa je ta vektor predstavljen v matriki:

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Stolpci v matriki so trije, ker predstavljajo vozlišča *virus, dsdna virus* in *ssrna virus*, vrednosti v posamezni vrstici pa prečkanje vozlišča od korenskega do končnega vozlišča. □

2.2.3 Priprava podatkov za uporabo v napovednih modelih

Večina uporabljenih atributov temelji na štetju, zato so podatki praviloma večji od 1. Taki podatki lahko v metodi, ki ni skalirno invariantna, privedejo do težav. Primer takšne metode je SVM. Zaradi tega je priporočljivo skaliranje vsakega atributa na interval $[0, 1]$ ali $[-1, +1]$ [7]. Podatke smo skalirali z metodo *MinMaxScaler* iz knjižnice *scikit-learn*. Metoda skalira po naslednjih formulah:

$$X_{std} = \frac{X - min(X)}{max(X) - min(X)} \tag{2.2}$$

$$X_{scaled} = X_{std} * (feature\_range_{max} - feature\_range_{min}) + feature\_range_{min} \tag{2.3}$$

V našem delu smo uporabili privzeti skalirni interval $[0, 1]$.

Poglavje 3

Metode

Ključni del diplomske naloge je uporaba metod strojnega učenja in metod odkrivanja znanj iz podatkov za modeliranje in napovedovanje na podlagi pripravljenih učnih podatkov. Osnovno vodilo strojnega učenja je opisovanje vzorcev v podatkih [18]. Za reševanje našega problema smo uporabili nadzorovane metode strojnega učenja za klasifikacijo. Tu se želimo naučiti funkcijo oziroma model, ki dani primer (genomsko zaporedje) preslika v enega izmed vnaprej določenih razredov [3].

3.1 Standardni napovedni modeli

V standardnih napovednih modelih smo uporabili podatke, v katerih nismo eksplicitno zapisali predznanja o taksonomiji. Uporabili smo naslednje metode:

1. metodo podpornih vektorjev (angl. Support Vector Machine (SVM)),
2. naključne gozdove (angl. Random Forest (RF)),
3. večinski klasifikator,
4. ostale metode.

V standardnih napovednih modelih uporabimo tudi večznačni klasifikator, ki poskrbi za to, da vsako od metod lahko uporabimo za napovedovanje na večznačni (angl. multilabel) domeni. V našem primeru uporabimo klasifikator eden-proti-ostalim (angl. OneVsRestClassifier), ker deluje hitreje kot klasifikator eden-proti-enemu. Strategija eden-proti-ostalim deluje tako, da uporabimo en klasifikator za en razred, kjer so primeri tega razreda pozitivni primeri, vsi ostali primeri pa so negativni. Pri strategiji eden-proti-enemu pa se zgradi $\frac{n \cdot (n-1)}{2}$ binarnih klasifikatorjev, kjer je n število razredov.

Pri strategiji eden-proti-ostalim je v našem primeru število klasifikatorjev 175. V primeru eden-proti-enemu pa je potrebnih 15225 klasifikatorjev. Iz tega je več kot razvidno, da je strategija eden-proti-ostalim časovno manj zahtevna.

3.1.1 Metoda podpornih vektorjev

Metoda podpornih vektorjev (angl. Support Vector Machine ali SVM) predstavlja eno izmed najbolj uspešnih metod za klasifikacijo. Deluje na podlagi deljenja primerov v razrede. Večina algoritmov stremi k temu, da minimalizira število atributov, uporabljenih v napovednem modelu. Metoda podpornih vektorjev pa uporabi vse atribute, tudi če niso pretirano pomembni.

Uporabljeni so za napoved ciljne spremenljivke, ki je lahko diskretna ali zvezna. Pri metodi podpornih vektorjev je torej pomembno, kako združiti atribute in ne kako izbrati najprimernejše [5]. Metoda podpornih vektorjev je uporabna za učenje velikih naborov podatkov, opisanih z velikim številom atributov [16].

Definicija 3.1 *Ideja metode podpornih vektorjev je, da določimo optimalni razred, ki razlikuje eno hiperravnino od druge v prostoru atributov. Če je učna množica linearno neodvisna, lahko sklepamo, da obstaja nekaj ločnih hiperravnin. Optimalna hiperravnina je enakomerno oddaljena od najbližjih primerov obeh razredov. Učne primere, ki so bližji hiperravnini, imenujemo podporni vektorji. Optimalna hiperravnina je izbrana tako, da maksimizira razdaljo med hiperravnino in podpornimi vektorji [5].*

Primer. Naj bo n število učnih primerov in a število atributov. Nabor učnih primerov je podan s pari $(t_j, y_j), j = 1 \dots n$, kjer je $y_j = 1$, če učni primer pripada prvemu razredu in $y_j = -1$, če učni primer pripada drugemu razredu. Predpostavimo, da so vsi atributi zvezni ali pa so vsaj obravnavani kot takšni. Zatorej, $t_j \in \mathbb{R}^a$, $j = 1 \dots n$. Enačba hiperravnine je [5]:

$$(\boldsymbol{w} \cdot \boldsymbol{t}) - b = 0 \tag{3.1}$$

Optimalna hiperravnina mora pravilno klasificirati vse učne primere.

$$\forall j \in 1 \dots n : y_j(\boldsymbol{w} \cdot \boldsymbol{t}_j - b) \geq 1 \tag{3.2}$$

Z minimizacijo magnitude koeficientov minimiziramo tudi kompleksnost rešitve. Težavo omejene optimizacije lahko transformiramo v funkcijsko maksimizacijo:

$$W(\alpha) = \sum_{j=1}^{n} \alpha_j - \frac{1}{2} \sum_{i,j}^{n} \alpha_i \alpha_j y_i y_j (\boldsymbol{t}_i \cdot \boldsymbol{t}_j) \tag{3.3}$$

z omejitvijo

$$\forall j \in 1 \dots n : \alpha_j \geq 0 \tag{3.4}$$

in

$$\sum_{j=1}^{n} \alpha_j y_j = 0 \tag{3.5}$$

□

Prednosti SVM so učinkovitost v visoko dimenzionalnih prostorih, tudi če je število dimenzij večje od števila primerov in učinkovitost pri porabi spomina. Uporabimo lahko različna jedra. V našem primeru smo uporabili linearno jedro, ker omogoča hitrejše delovanje.

3.1.2 Naključni gozdovi

Naključni gozdovi (angl. Random Forest (RF)) predstavljajo izboljšavo napovedne točnosti klasifikacijskih dreves. Prvotno so bili razviti zgolj za izboljšanje le-teh, vendar so lahko uporabljeni tudi za izboljšanje regresijskih dreves. Naključni gozd sestavlja skupina klasifikacijskih dreves, ki deluje tako, da pri uvrščanju primera v razred glasuje. Naključni gozd napove tisti razred, kamor večina klasifikacijskih dreves uvrsti določen primer. Naključni gozd bi dosegel najboljše rezultate takrat, ko bi gozd sestavljala drevesa, ki so si med seboj najbolj različna. Tako bi vsak od dreves našel kakšen poseben koncept, ki ga druga drevesa niso odkrila in se je skrival v podatkih. Napovedne točnosti te metode so ene najboljših, zato smatramo metodo naključnih dreves za eno od najbolj zanesljivih metod uvrščanja. Metoda prevzame vse dobre strani klasifikacijskih dreves ter zavrže možnost interpretacije modela. Ta ni več mogoča, ker metodo sestavlja vrsta modelov, katerih vpliv je odvisen od konteksta [15, 4].

3.1.3 Ostale metode

Uporabili smo tudi večinski klasifikator (angl. Dummy Classifier (DC)) ter logistično regresijo (angl. Logistic Regression (LR)). Večinski klasifikator služi kot osnovna, referenčna metoda za testiranje in primerjanje rezultatov. Deluje tako, da vsem primerom napove večinski razred. S tem vidimo, koliko boljše (če sploh) napovedujejo ostale metode. Logistično regresijo vpeljemo

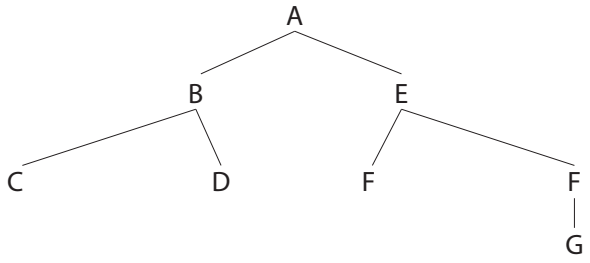
za primerjavo, vendar se nato izkaže kot relativno dobra metoda, in v nekaj primerih zelo konkurira SVM.

3.2 Napovedovanje strukturiranih razredov s programom PyStruct

PyStruct je program za strukturirano napovedovanje podatkov. Strukturirano napovedovanje je generalizacija standardnih paradigm nadzorovanega učenja, klasifikacije in regresije. V klasifikaciji so lahko ciljna domena diskretni razredi, kjer je izguba med 0 in 1 (če štejemo napačno napovedane). Pri regresiji so ciljna domena zvezna števila, izguba pa je običajno povprečna kvadratna napaka. V strukturiranem napovedovanju sta tako ciljna domena kot tudi izguba manj samovoljna - cilj ni napovedati, na primer, število ampak veliko bolj strukturiran objekt kot sta zaporedje ali graf. V strukturiranem napovedovanju se pogosto srečamo s končnim, vendar ogromnim prostorom Y. To lahko omejimo z večrazredno klasifikacijo. Ideja strukturiranega napovedovanja je torej, da pri napovedi uporabimo strukturo izhodnega prostora [6]. Strukturo lahko pridobimo na več načinov. Lahko jo izračunamo z algoritmi za pridobivanje vzajemne informacije, ali pa jo zgradimo sami. Tukaj smo funkcijo, ki zgradi stukturo prilagodili našim potrebam.

```
1  def chow_liu_tree(y_):
2    n_labels = y_.shape[1]
3    mi = np.zeros((n_labels, n_labels))
4    for i in range(n_labels):
5      row_i = np.zeros(n_labels)
6      for ex in y_:
7        prev = -1
8        next = -1
9        if ex[i] == 1:
10         temp_occur = [x for x in np.where(ex == 1)[0] if x
               != i]
11         less_than_i = [x for x in temp_occur if x < i]
12         more_than_i = [x for x in temp_occur if x > i]
13         if len(less_than_i) > 0:
14           prev = max(less_than_i)
15         if len(more_than_i) > 0:
16           next = min(more_than_i)
17         if prev == -1 and next != -1:
18           prev = next
19         elif next == -1 and prev != -1:
20           next = prev
21         elif next == -1 and prev == -1:
22           raise ValueError("alkdfjasd")
23         row_i[prev] = 1
24         row_i[next] = 1
25     mi[i, :] = row_i
26   if (mi.transpose() == mi).all() == False:
27     ValueError("Matrix is not symmetric!")
28   mst = minimum_spanning_tree(csr_matrix(-mi))
29   edges = np.vstack(mst.nonzero()).T
30   edges.sort(axis=1)
31   return edges
```

Primer.



Slika 3.1: Struktura taksonomije primera.

Iz drevesa, podanega v sliki 3.1, dobimo sledečo matriko Y:

$$\begin{bmatrix} A & B & C & D & E & F & G \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Z zgornjo metodo *chow_liu_tree* bi jo pretvorili v matriko, ki bi predstavljala sosede (A je sosed od B in E, B je sosed od A, C in D ipd.). Dobimo sledečo matriko:

$$\begin{bmatrix} & A & B & C & D & E & F & G \\ A & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ B & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ C & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ D & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ E & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ F & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ G & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

□

S predznanjem v obliki relacij med razredi naj bi se PyStruct lažje naučil in napovedal bolj točne vrednosti.

3.3 Vrednotenje napovedne uspešnosti s prečnim preverjanjem

Če imamo na voljo malo vhodnih podatkov, ne smemo prikrajšati učnega algoritma za podmnožico testnih primerov. Za učenje potrebujemo vse razpoložljive primere, vendar želimo vseeno oceniti uspešnost avtomatsko zgrajene teorije. Ocenjevanje uspešnosti hipoteze na učni množici je zelo slaba ocena. V našem primeru uporabimo K-kratno prečno preverjanje. Število K določa število modelov, ki jih moramo zgraditi [4, 12].

Na začetku celotno množico podatkov razdelimo na K (v našem primeru 5) približno enako velikih podmnožic. Pri vsakem od korakov nato uporabimo eno podmnožico za testiranje, ostale pa za učenje in gradnjo napovednega modela. Uspešnost končne hipoteze ocenimo kot povprečno uspešnost vseh zgrajenih napovednih modelov.

3.3.1 Večznačno ocenjevanje

Napovedi smo ocenjevali z merami povprečne natančnosti (angl. Average Precision Score), površine pod krivuljo ROC oz. mero AUC ter merjenjem Hammingove napake (angl. Hamming Loss). Zaradi večznačne narave problema bi bilo neprimerno uporabiti mero natančnosti (angl. accuracy), saj ta predvideva, da so napovedi pravilne v vseh značkah, sicer napoved oceni za popolnoma neuspešno.

Površina pod krivuljo ROC meri diskriminativnost klasifikatorjev. Prvotno so jo uporabljali v ameriški vojski in z njo ocenjevali radariste ter njihovo točnost. Danes jo uporabljamo na področju statistike in podatkovnem rudarjenju oz. strojnem učenju. Mera je zanimiva, saj se iz njenih rezultatov hitro vidi uspeh napovednega modela. Rezultat $0,50$ pomeni, da so napovedne vrednosti naključne. Zato pričakujemo rezultate, ki so boljši od $0,50$.

Večznačno uvrščanje zahteva drugačno ocenjevanje kot klasično enoznačno [11]. Ena od ocenjevanj, ki jih v drugih merah običajno ni smiselno opazovati, je tudi Hammingova napaka. S to oceno preverimo, v koliko značkah

je naša napoved napačna.

Definicija 3.2 *Če je D večznačni nabor podatkov, je $|D|$ število primerov. $|L|$ predstavlja število vseh značk, y_i dejansko vrednost in x_i napoved [11].*

$$HammingLoss(x_i, y_i) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{xor(x_i, y_i)}{|L|} \tag{3.6}$$

Uporabimo tudi povprečno natančnost. Mera se od klasične natančnost razlikuje po tem, da ne upošteva samo razreda kot celote, ampak tudi zaporedje značk.

3.4 Filtriranje atributov

Z rastjo vrednosti k pri *k-terkah* število atributov narašča eksponentno. Tako imamo, na primer, pri $k = 1$ vsega skupaj 69 atributov, pri $k = 2$ imamo 81 atributov, pri $k = 7$ pa že 16449 atributov. Število atributov je opisano z naslednjo formulo:

$$attr\_no_k = 4^k + 1 + 4^3 \tag{3.7}$$

kjer 4^k predstavlja število atributov iz *k-terk*, konstanti pa predstavljata dolžino sekvence ter število kodonov. Zaradi velikega števila atributov smo se odločili, da izberemo le nekaj najboljših. To smo storili z metodo *SelectKBest*. Metodi moramo podati število želenih končnih atributov ter funkcijo po kateri izbira najustrežnejše. Testirali smo funkcije *chi2, f_classif* in *f_regression* za 30 najboljših atributov. Opazovali smo rezultat AUC za različne k. Rezultate smo opazovali zgolj za metodo naključnih gozdov.

V tabeli 3.1 je očitno, da so rezultati s funkcijo *chi2* boljši, zato število atributov pred vhodom v napovedne modele zmanjšamo na najboljših 30.

k	**chi2**	**f_classif**	**f_regression**
1	**0,572**	0,566	0,567
2	**0,575**	0,566	0,568
3	**0,580**	0,567	0,560
4	**0,577**	0,564	0,545
5	0,569	**0,571**	0,540
6	**0,576**	0,569	0,551
7	**0,577**	0,564	0,562

Tabela 3.1: Primerjava rezultatov AUC naključnih gozdov za različne funkcije ocenjevanja.

Poglavje 4

Rezultati

4.1 Uspešnost različnih načinov atributnega opisovanja

Tukaj želimo ovrednotiti ciljno vprašanje o uspešnosti uvrščanja virusnih zaporedij v taksonomijo. Vsi uporabljeni podatki vsebujejo tudi atribut, ki podaja dolžino genoma. Opozoriti želimo še na to, da so vsa testiranja potekala brez iskanja najboljših parametrov za vsako od metod. Pri merjenju rezultatov smo odstranili korensko vozlišče, ker je prisotno v vseh primerih. Zato je neprimerno, da ocenjujemo pravilnost napovedovanja tega vozlišča. Za učenje ga kljub vsemu uporabimo, saj je pri metodi za strukturirano napovedovanje ključno ravno to, da iz vozlišč sestavimo matriko sosedov (primer 3.2), ki pomaga metodi pri uvrščanju.

4.1.1 Uspešnost atributnega opisovanja s *k-terkami*

Sprva smo preverili uspešnost atributnega opisovanja zgolj s *k-terkami* baz $A,\ T,\ C$ in G, kjer je $k \in [1, 7]$.

Slika 4.1 pokaže, da so napovedne točnosti AUC izredno slabe. Najboljše rezultate dobimo z metodo naključnih gozdov, ki doseže AUC $0,57$, vendar so tudi ti rezultati pogojno uspešni.

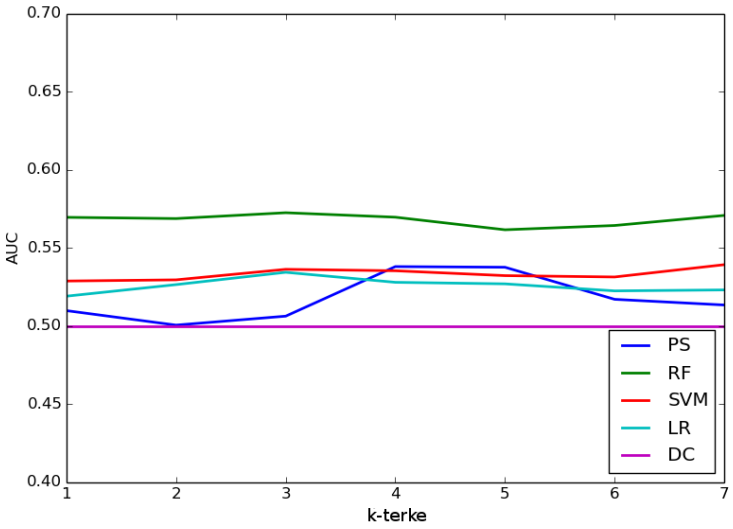
Slika 4.2 nam pokaže, da je napaka naših napovedi v približno 2%, kar pomeni v $3,5$ značkah. Zaradi slabih rezultatov smo se nato odločili, da poskušamo najti atribute, ki so bolj informativni in pomagajo napovednemu modelu napovedovati z večjo točnostjo in manjšo napako.

4.1.2 Uspešnost atributnega opisovanja s kodoni in *k-terkami*

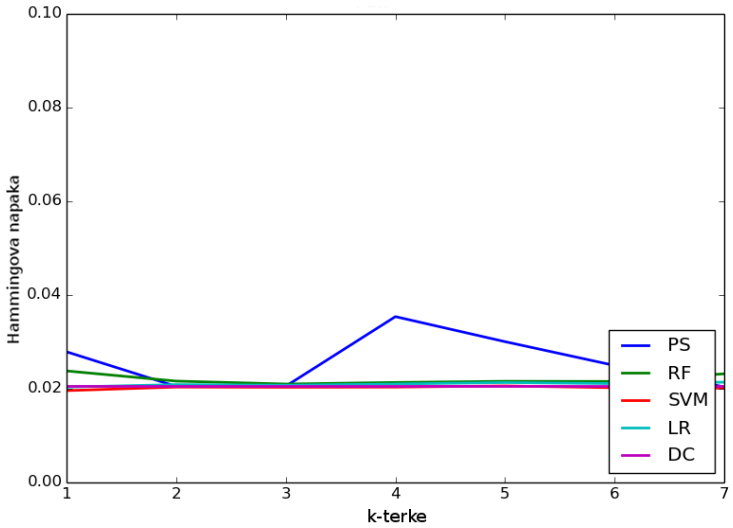
Struktura kodonov naj bi v določenem genomu nekoliko bolj opredelila za kater genom gre. Zato dodamo vsakemu podatkovnemu naboru *k-terk* še atributni opis kodonov.

Na slikah 4.1 in 4.3 je težko opaziti raziko med različnimi podatkovni nabori. Še najbolj vidna razlika je pri strukturiranem napovedovanju s PyStruct, ki se mu rezultat izboljša v $k = 2$ in $k = 3$. Za ostale razlike lahko pregledamo natančnejše rezultate v tabelah 4.1 in 4.2. Za $k = 1$ in $k = 2$ je napovedovanje na podatkovnem naboru s kodoni zanesljivo boljše, saj pri vseh štirih metodah napove bolje kot brez kodonov. Za $k = 3$ sta podatkovna nabora še najbolj enakovredna, saj je dvakrat boljša napoved v podatkih brez kodonov in dvakrat boljša pri podatkih s kodoni. Podobno je za $k = 7$, vendar v tem primeru dvakrat bolje napovemo na podatkih s kodoni, dvakrat pa je rezultat enak. Za $k = 5$ in $k = 6$ je rezultat pri treh metodah boljši pri podatkih s kodoni, pri eni metodi je boljši za podatke brez kodonov ter pri eni enak. Opazimo, da se rezultati nekoliko izboljšajo.

Od tu naprej delamo primerjave in merimo uspešnost zgolj na podatkovnem naboru s kodoni in *k-terkami*, saj smo s tem naborom napovedali boljše v 75% primerov, medtem ko smo brez kodonov boljše napovedali v 14% primerov. Rezultat je bil približno enak v 11% primerov. Zagotovo je iskanje primernih atributov z drugimi postopki eno od zelo zanimivih vprašanj, ki se je odprlo za nadaljnje delo.



Slika 4.1: AUC za atributno opisovanje s *k-terkami*.



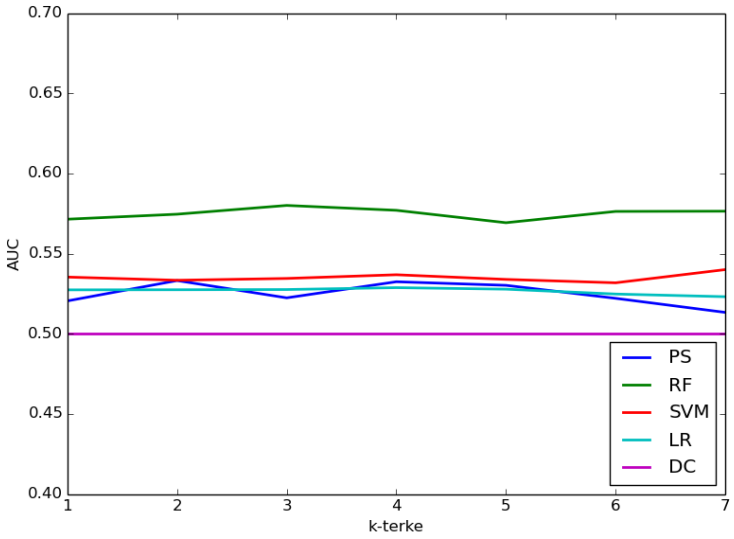
Slika 4.2: Hammingova napaka za atributno opisovanje s *k-terkami*.

	1	**1+K**	**2**	**2+K**	**3**	**3+K**	**4**	**4+K**
PS	0,510	**0,521**	0,501	**0,533**	0,506	**0,522**	**0,538**	0,532
RF	0,570	**0,572**	0,569	**0,575**	0,573	**0,580**	0,570	**0,577**
SVM	0,529	**0,536**	0,530	**0,533**	**0,536**	0,534	0,535	**0,537**
LR	0,520	**0,527**	0,526	**0,528**	**0,534**	0,528	0,528	**0,529**
DC	0,500	0,500	0,500	0,500	0,500	0,500	0,500	0,500

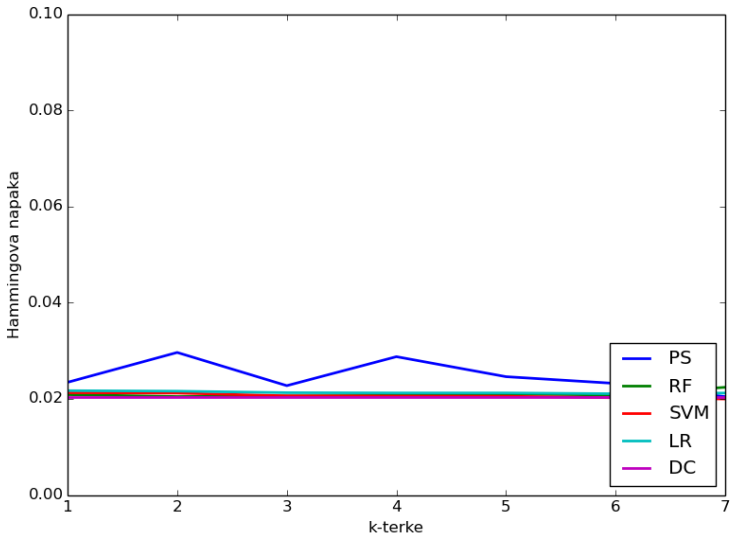
Tabela 4.1: Primerjava rezultatov atributov za podatkovni nabor samo s *k-terkami* (samo število) ter za podatkovni nabor s *k-terkami* in kodoni (K zraven števila). Predstavljen je samo del *k-terk* za $k = [1, 4]$.

	5	**5+K**	**6**	**6+K**	**7**	**7+K**
PS	**0,538**	0,530	0,517	**0,522**	0,513	0,513
RF	0,562	**0,569**	0,564	**0,576**	0,571	**0,577**
SVM	0,532	**0,533**	0,532	0,531	0,539	**0,540**
LR	0,527	**0,528**	0,522	**0,525**	0,523	0,523
DC	0,500	0,500	0,500	0,500	0,500	0,500

Tabela 4.2: Primerjava rezultatov atributov za podatkovni nabor samo s *k-terkami* (samo število) ter za podatkovni nabor s *k-terkami* in kodoni (K zraven števila). Predstavljen je samo del *k-terk* za $k = [5, 7]$.



Slika 4.3: AUC za atributno opisovanje s *k-terkami* in kodoni.

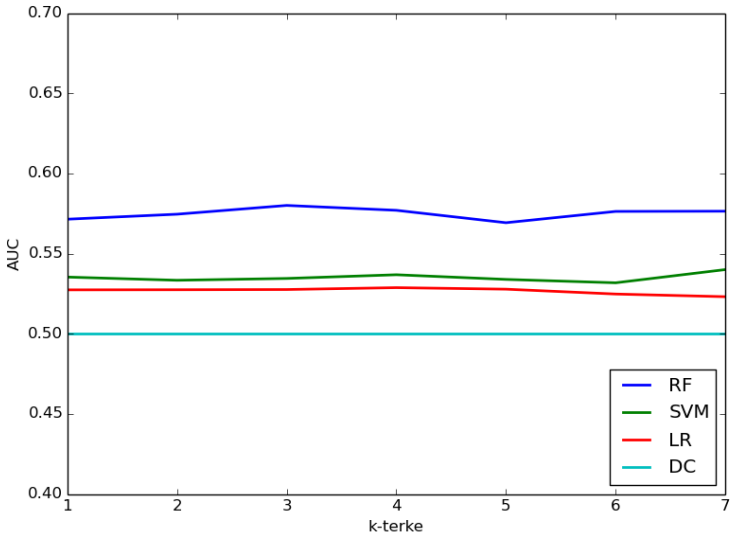


Slika 4.4: Hammingova napaka za atributno opisovanje s *k-terkami* in kodoni.

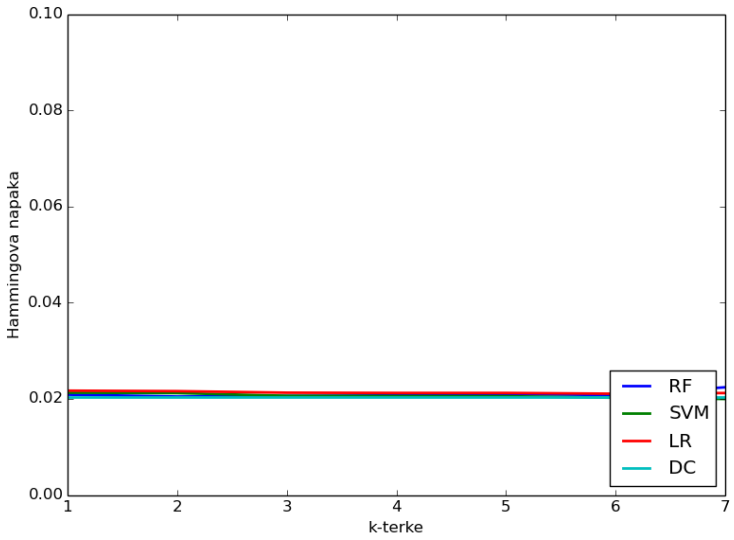
4.2 Uspešnost napovedovanja s standardnimi pristopi za večznačno napovedovanje

V tem podpoglavju odgovorimo na ciljno vprašanje o upešnosti uvrščanja virusnih zaporedij v taksonomijo z metodami strojnega učenja. Sliki 4.5 in 4.6 prikazujeta, kako se izbrani nabor podatkov obnaša pri večznačnem napovedovanju s klasičnimi metodami. Opazimo lahko, da metoda naključnih gozdov prednjači z rezultatom AUC $0,580$ za $k = 3$. Rezultati RF tudi pri ostalih k niso bistveno drugačni, saj je razlika med najvišjo in najnižjo mero AUC $0,0108$. Metoda LR je izredno blizu metode SVM, kar nas preseneti. Rezultati se v povprečju razlikujejo za približno $0,007$ v korist SVM, razen za $k = 7$, kjer uspešnost LR rahlo pade, uspešnost SVM pa se izboljša. V povprečju so rezultati pričakovani. Najbolje napovedujejo RF, kar se lahko pripiše njihovim značilnostim pri uvrščanju. Najbolj preseneti metoda LR, ki se od SVM-ja razlikuje minimalno.

Hammingova napaka je pri klasičnih metodah skoraj enaka - vse metode imajo napako okrog 2%, kar pomeni napačno napoved v približno $3,5$ značkah.



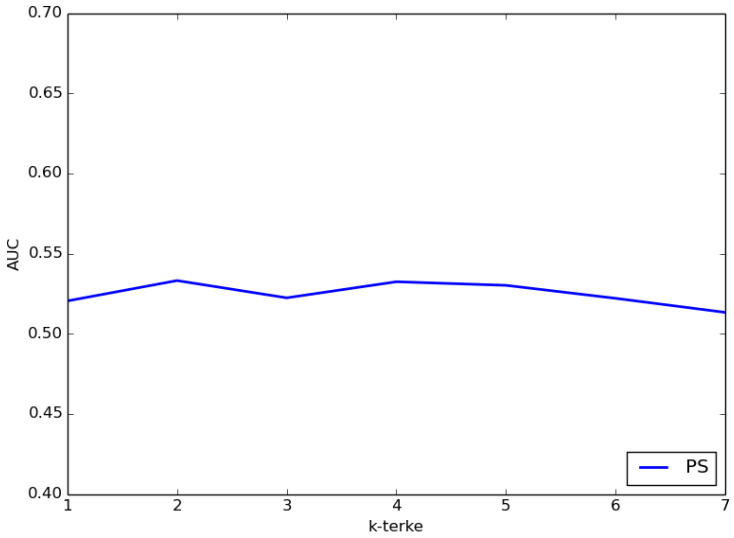
Slika 4.5: AUC za atributno opisovanje s *k-terkami* in kodoni.



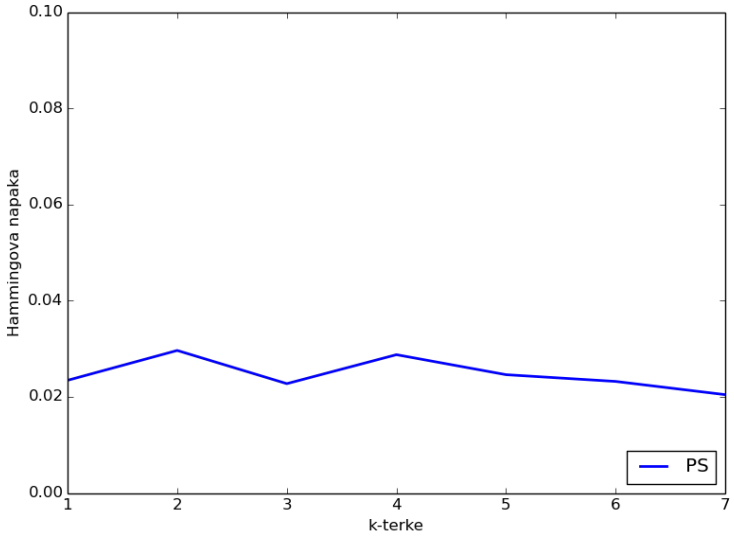
Slika 4.6: Hammingova napaka za atributno opisovanje s *k-terkami* in kodoni.

4.3 Uspešnost strukturiranega napovedovanja z metodo PyStruct

Sliki 4.7 in 4.8 prikazujeta uspešnost metode PyStruct. Opazimo lahko, da je uspešnost nižja kot pri klasičnih metodah na sliki 4.5, Hammingova napaka pa višja. Tega nismo pričakovali, saj PyStruct uporablja dodatno predznanje v obliki relacij med sosedi v taksonomiji, česar klasične metode nimajo, zato smo pričakovali pri PyStructu boljše rezultate. Opozoriti moramo, da smo zaradi časovne potratnosti metodo omejili z največ 500 iteracijami. Nastavili smo tudi nekatere druge parametre, ki naj bi pohitrile delovanje. Še vedno obstaja verjetnost, da je težava za tako časovno potratnost v neustreznih parametrih, zato bi jih bilo smiselno bolje raziskati in najti najbolj ustrezno možno kombinacijo. Pri časovni potratnosti verjetno pripomore tudi število značk, ki jih je 175.



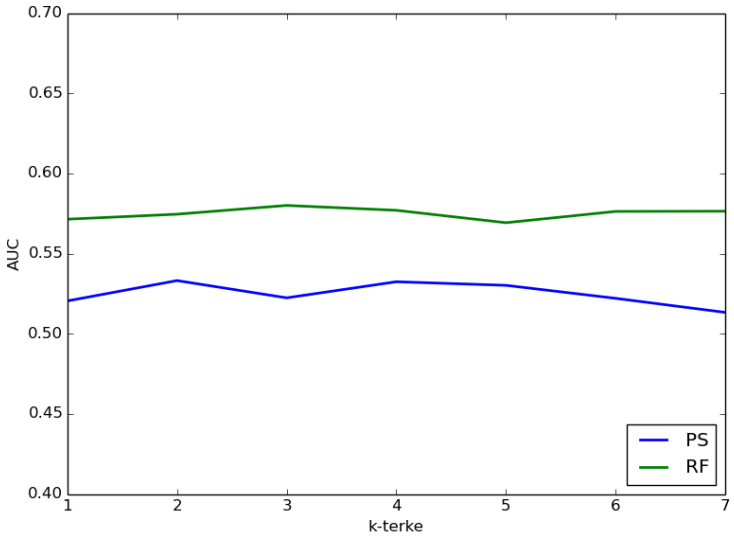
Slika 4.7: AUC uspešnosti strukturiranega napovedovanja z metodo PyStruct.



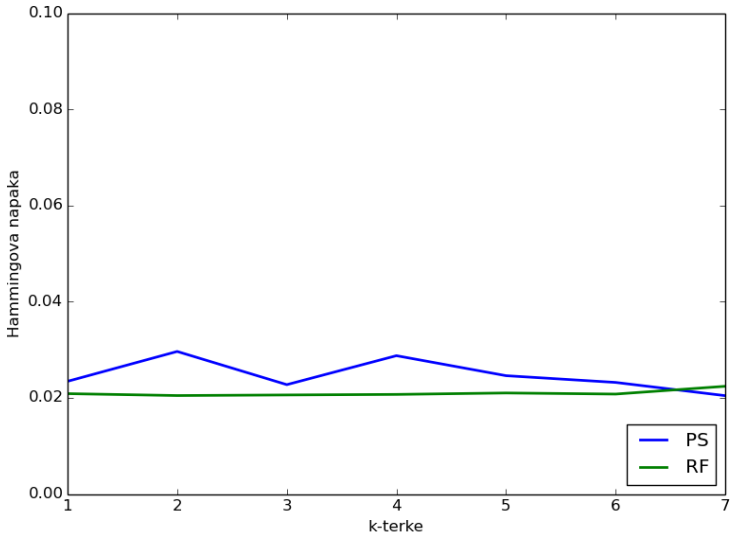
Slika 4.8: Hammingova napaka za strukturirano napovedovanje z metodo PyStruct.

4.4 Primernost metode PyStruct za napovedovanje taksonomije

V tem poglavju odgovorimo na ciljno vprašanje o primernosti metode PyStruct za napovedovanje taksonomije. Rezultata na slikah 4.9 in 4.10 pokažeta, da je metoda za strukturirano napovedovanje manj primerna od naključnih gozdov. Težava se lahko skriva v tem, da smo metodo PyStruct omejili zgolj z največ 500 iteracijami (kar je privzeto nastavljeno na 10000). Najbolj verjetna možnost za tako slab rezultat pa je tudi izbira atributov. Če bi izbrali nekatere bolj kompleksne atribute, bi metoda PyStruct lahko napovedala bolje od naključnih gozdov, ni pa nujno. Vsekakor bi bil ta problem primeren za nadaljnje delo. Vzrok za tak rezultat lahko tiči tudi v tem, da uporablja PyStruct za učenje zgolj linearne relacije med atributi. Naključni gozdovi imajo sposobnost zajeti lokalne, nelinearne relacije med atributi in razredom, zaradi česar se ta metoda izkaže za boljšo na mnogo domenah. Empirično dokazano lahko trdimo, da je metoda PyStruct z našim naborom podatkov za napovedovanje taksonomije manj primerna od naključnih gozdov.



Slika 4.9: Ocena AUC o primernosti metode PyStruct za napovedovanje taksonomije.



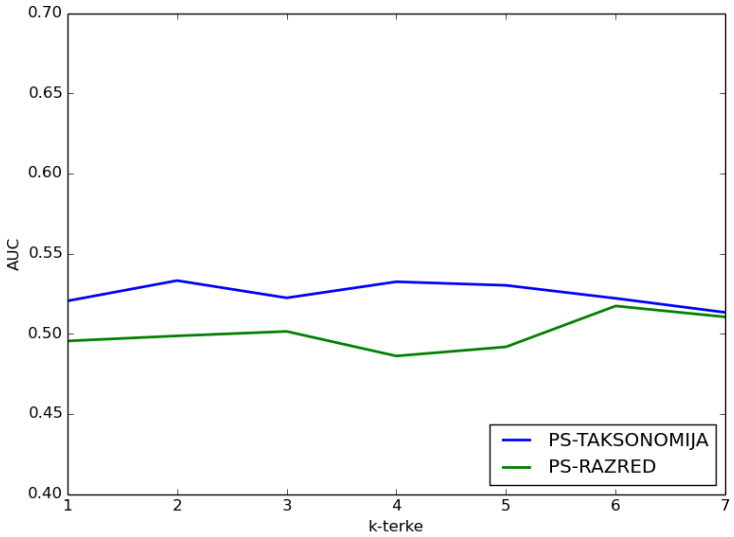
Slika 4.10: Ocena Hammingove napake o primernosti metode PyStruct za napovedovanje taksonomije.

4.5 Vpliv predznanja o taksonomiji na uspešnost napovedovanja

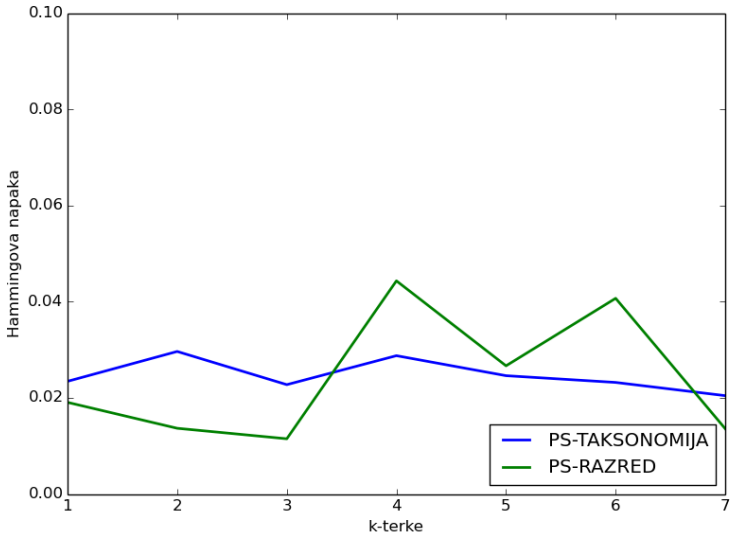
Preverili smo, kakšen je vpliv predznanja o taksonomiji za uspešnost napovedovanja. To smo storili tako, da smo primerjali napovedne rezultate metode PyStruct, kjer smo za napovedovanje uporabili samo končne razrede in kjer smo za napoved uporabili celotno taksonomijo.

Na slikah 4.11 in 4.12, ki predstavljata vpliv predznanja o taksonomiji za metodo PyStruct opazimo, da predznanje pozitivno vpliva na uspešnost napovedovanja. S predznanjem je napoved pričakovano boljša, kot če uporabimo zgolj liste.

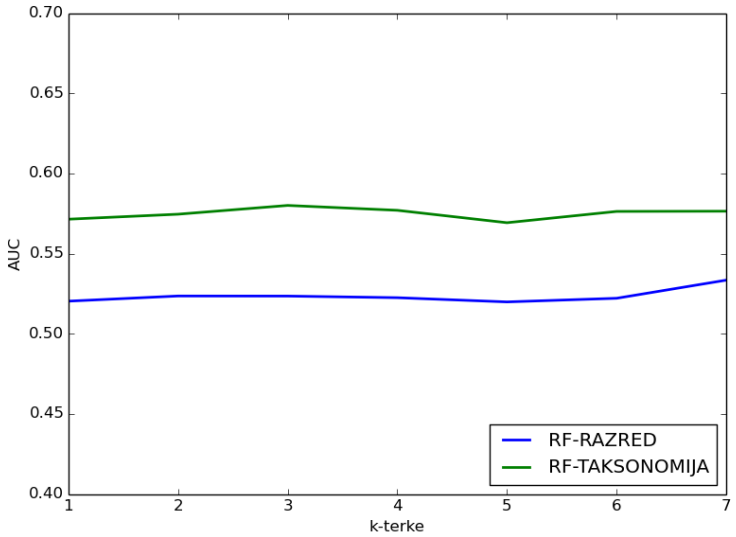
Tudi v primeru, ko napovedujemo z metodo naključnih gozdov, se rezultati vidno izboljšajo, glej sliki 4.13 in 4.14. Rezultati se izboljšajo, če uporabimo predznanje o taksonomiji. Rezultat je pričakovan, saj uporabimo znanje, ki ga napovedovanje v posamezne liste nima.



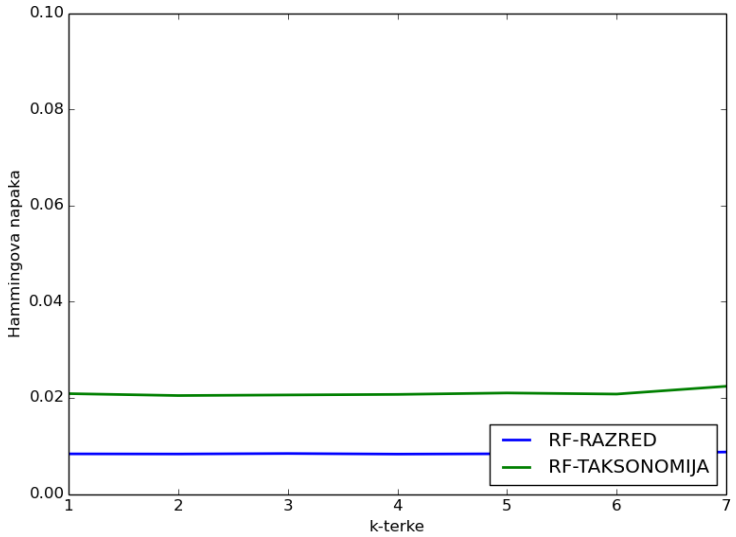
Slika 4.11: Ocena AUC o vplivu predznanja o taksonomiji na uspešnost napovedovanja za PyStruct.



Slika 4.12: Ocena Hammingove napake o vplivu predznanja o taksonomiji na uspešnost napovedovanja za PyStruct.



Slika 4.13: Ocena AUC o vplivu predznanja o taksonomiji na uspešnost napovedovanja za naključni gozd.



Slika 4.14: Ocena Hammingove napake o vplivu predznanja o taksonomiji na uspešnost napovedovanja za naključni gozd.

Poglavje 5

Zaključek

V taksonomiji, zgrajeni iz podatkov s spletne strani NCBI, smo uporabili štiri klasične algoritme za strojno učenje (logistično regresijo, naključne gozdove, referenčni večinski klasifikator ter metodo podpornih vektorjev) ter enega za strukturirano napovedovanje (PyStruct). Ugotovili smo, da z metodami strojnega učenja lahko zgradimo napovedni model za uvrščanje virusnih zaporedij v taksonomijo. Za uspešno operiranje z biološkimi podatki je potreben bolj premišljen pristop. V našem primeru so sicer rezultati le nekoliko boljši od večinskega klasifikatorja. Uporabiti bi bilo potrebno še več danega predznanja ali pa najti daljša, bolj specifična zaporedja, katerih pojavitev bi iskali v ostalih zaporedjih.

V naši implementaciji je najbolj ustrezen način iskanja značilk združitev strukture *k-terk* za $k = [1, 7]$ in podatkov o strukturi kodonov. Kodone pridobimo iz kodirnih delov genomskih zaporedij in preštejemo njihovo frekvenco. Pri primernosti metode za strukturirano napovedovanje lahko rečemo, da je metoda naključnih gozdov bolj primerna od metode PyStruct. Tega nismo pričakovali, saj metoda PyStruct uporablja prednost v obliki eksplicitno podanega predznanja sosedov celotne taksonomije. Zelo verjetno je težava tudi v omejitvi števila iteracij. Predznanje o taksonomski povezanosti virusov pripomore k boljši napovedi tako v primeru uporabe PyStructa, kot tudi v primeru uporabe naključnih gozdov. Iz tega lahko sklepamo, da je priso-

tnost taksonomije pri učenju uporabna, zato bi bilo smiselno v nadaljnjem delu poizkusiti z iskanjem kompleksnejših atributov, povezanih s taksonomsko pripadnostjo virusov.

Kompleksnost problema je (pričakovano) presegla okvire diplomske naloge in hkrati odprla vrsto novih vprašanj. Slab izbor atributov je nakazal na to, da bi bilo potrebno izbrati in uporabiti več vnaprej znanih podatkov, saj je s tako grobimi atributi v takšnih podatkih težko izluščiti kaj konkretnega. Zanimivo bi bilo preučiti, kako se obnaša napovedovanje, kadar poravnamo sekvence in iščemo daljša, specifična zaporedja in kako kadar uporabimo in upoštevamo več podatkov o strukturi genoma in genov, kar je shranjeno v zapisih GenBank. Smiselno bi bilo tudi poiskati kakšne dodatne atribute, povezane s taksonomsko pripadnostjo.

Literatura

[1] Peter J. A. Cock, Tiago Antao, Jeffrey T. Chang, Brad A. Chapman, Cymon J. Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, and Michiel J. L. de Hoon. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, 2009.

[2] Thien Ho and Ioannis E Tzanetakis. Development of a virus detection and discovery pipeline using next generation sequencing. *Virology*, 471:54–60, 2014.

[3] M. Kantardzic. *Data Mining: Concepts, Models, Methods, and Algorithms*. John Wiley & Sons, 2011.

[4] Igor Kononenko. Strojno ucenje. *Fakulteta za racunalništvo in informatiko, Ljubljana, Slovenija*, 41, 1997.

[5] Igor Kononenko and Matjaž Kukar. *Machine learning and data mining: introduction to principles and algorithms*. Horwood Publishing, 2007.

[6] Andreas C. Müller and Sven Behnke. pystruct - learning structured prediction in python. *Journal of Machine Learning Research*, 15:2055–2060, 2014.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Du-

chesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] Yosef Prat, Menachem Fromer, Nathan Linial, and Michal Linial. Codon usage is associated with the evolutionary age of genes in metazoan genomes. *BMC Evolutionary Biology*, 9(1):285, 2009.

[9] Simon Roux, Francois Enault, Bonnie L Hurwitz, and Matthew B Sullivan. Virsorter: mining viral signal from microbial genomic data. *PeerJ*, 3:e985, 2015.

[10] Rajeev K Varshney, Spurthi N Nayak, Gregory D May, and Scott A Jackson. Next-generation sequencing technologies and their implications for crop genetics and breeding. *Trends in biotechnology*, 27(9):522–530, 2009.

[11] J. Wang. *Data Warehousing and Mining: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications*. Contemporary research information science and technolog book series. Information Science Reference, 2008.

[12] Wikipedia. Cross-validation (statistics) — wikipedia, the free encyclopedia, 2015. [Online; accessed 9-September-2015].

[13] Wikipedia. Genbank — wikipedia, the free encyclopedia, 2015. [Online; accessed 9-September-2015].

[14] Wikipedia. National center for biotechnology information — wikipedia, the free encyclopedia, 2015. [Online; accessed 9-September-2015].

[15] Wikipedia. Random forest — wikipedia, the free encyclopedia, 2015. [Online; accessed 9-September-2015].

[16] Wikipedia. Support vector machine — wikipedia, the free encyclopedia, 2015. [Online; accessed 9-September-2015].

[17] Wikipedia. Virus — wikipedia, the free encyclopedia, 2015. [Online; accessed 12-September-2015].

[18] Ming Xue and Changjun Zhu. A study and application on machine learning of artificial intellligence. In *Artificial Intelligence, 2009. JCAI '09. International Joint Conference on*, pages 272–274, April 2009.