Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Anže Kolar

Prepoznavanje vrst letov iz zapisov GPS

diplomsko delo

interdisciplinarni univerzitetni
študijski program prve stopnje
računalništvo in matematika

Mentor: prof. dr. Janez Demšar

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Razmah pametnih telefonov in nosljivih naprav, kot so športne ure, je tudi rekreativnim športnikom omogočil, da s pomočjo GPS beležijo podatke o svojih aktivnosti. Za boljšo uporabniško izkušnjo aplikacij za delo s temi podatki in tudi za lažjo analizo podatkov potrebujemo algoritme za razlikovanje med sicer podobnimi športnimi aktivnostmi.

V diplomski nalogi s pomočjo metod strojnega učenja in na osnovi množice že zbranih podatkov sestavite model za razlikovanje med zapisi poti, ki jih opravijo jadralna letala, jadralni zmajarji in jadralni padalci. Model ustrezno analizirajte in testirajte.

Zahvaljujem se podjetju Naviter, ki je s prispevanjem izziva ter podatkov omogočilo izdelavo tega diplomskega dela.
Za strokovno pomoč se zahvaljujem tudi mentorju, prof. Janezu Demšarju, za številne napotke, nasvete ter hudomušne komentarje, s katerimi mi je olajšal pisanje te naloge.

No amount of experimentation can ever prove me right; a single experiment can prove me wrong.

— A. Einstein

Kazalo

Seznam uporabljenih kratic

kratica	**angleško**	**slovensko**
AGL	above ground level	višina nad terenom
ANN	artificial neural network	umetna nevronska mreža
CA	classification accuracy	klasifikacijska točnost
DAG	directed acyclic graph	usmerjeni aciklični graf
GPS	global positioning system	globalni sistem pozicioniranja
GLM	generalized linear model	posplošen linearni model
k**-NN**	k-nearest neighbours	metoda k najbližjih sosedov
L/D	lift-to-drag (ratio)	finesa
LR	logistic regression	logistična regresija
MLP	multi-layer perceptron	večslojni perceptron
PC	principle component	glavna komponenta
PCA	principle component analysis	analiza glavnih komponent
RF	random forest	naključni gozd
RNN	recurrent neural network	rekurenčna nevronska mreža
SVM	support vector machine	metoda podpornih vektorjev

Povzetek

Naslov: Prepoznavanje vrst letov iz zapisov GPS

Avtor: Anže Kolar

Zapisovalniki zapisov letov, ki beležijo pozicije GPS, so v letalstvu vedno pogostejši. Ustvarjene zapise iz teh naprav uporabniki navadno želijo naložiti na spletne platforme, s katerimi lahko podatke analizirajo ali pa jih delijo z drugimi uporabniki. Količina tovrstnih podatkov hitro narašča, zato se pojavlja potreba po pametnem razvrščanju naloženih podatkov, s katero se olajša delo ponudniku storitve ter pilotu zagotovi boljšo uporabniško izkušnjo. Cilj te diplomske naloge je razviti sistem za prepoznavanje vrste letalne naprave, v kateri je bil ustvarjen naložen zapis GPS. Na podmnožici celotne zbirke podatkov definiramo nabor atributov, ki jih uporabimo za grajenje in primerjanje modelov, zgrajenih z različnimi metodami strojnega učenja. Predstavimo pa tudi nekaj možnih načinov za dopolnjevanje učne množice. Rezultati testiranja so dobri: z najboljšimi modeli dosežemo točnost po meri F_1 nad 0,97, kar jih naredi uporabne tudi v produkcijskem okolju, s povečanjem števila učnih podatkov lahko uspešnost še povečamo.

Ključne besede: strojno učenje, klasifikacija, jadralno letenje.

Abstract

Title: Classification of flight types from GPS recordings

Author: Anže Kolar

Flight loggers that store GPS positions are becoming increasingly popular. Records created with such devices are usually uploaded to various web platforms that provide methods for further data exploration and integrate services for sharing the captured flights on the social media. The amount of uploaded files is continually increasing, thus creating the need for smarter classification of the data, which in turn creates a more optimised and user-friendly service. The goal of this thesis is to develop a system for the recognition of the aircraft type in which the flight was recorded. We define a set of attributes on a smaller subset of the entire flight database and use it for building and comparing the models created using different methods of machine learning. We also present a few methods that could further expand the original training set. Final results are mostly positive: most promising models achieve an F_1 score of more than 0.97 which makes them suitable for the use in a production environment. Even better scores can be attained by increasing the number of learning samples.

Keywords: machine learning, classification, soaring.

Poglavje 1

Uvod

Med rekreativnim jadralnim letenjem piloti pogosto beležijo opravljene poti z namenskimi zapisovalniki, ki temeljijo na shranjevanju pozicij GPS. Shranjene zapise lahko kasneje uporabijo za lastno evidenco ter natančnejšo analizo leta ali pa ga na tekmovanjih oddajo za izračun doseženih točk.

Tudi v letalstvu se v zadnjih letih opaža trend naraščanja pametnih naprav, tako namenskih kot v obliki mobilnih telefonov, s katerimi je mogoče ustvarjati omenjene zapise. Poleg tega se je pod okriljem organizacije Open Glider Network vzpostavil sistem talnih anten, ki lahko spremljajo letalne naprave, ki svojo pozicijo sporočajo preko namenskih protokolov [2]. Poleg tega tudi številni ponudniki tehnoloških letalskih storitev razvijajo svoje oblačne platforme, kamor lahko uporabniki shranjujejo opravljene lete [3, 1, 4, 5]. Zaradi hitro naraščajoče količine naloženih podatkov se, ne samo s stališča ponudnika storitve, temveč tudi s strani končnega uporabnika, ki želi zapise analizirati, kmalu pojavi potreba po identificiranju nekaterih ključnih lastnosti, ki so vsebovane v takih zapisih.

Kljub porastu uporabe tehnologije v aviaciji pa na področju "pametnih" algoritmov, s katerimi bi uporabniku olajšali uporabo obstoječih storitev z avtomatskim označevanjem zapisov letov, v zadnjih letih ni opaziti bistvenega napredka. Poleg izboljšanja uporabniške izkušnje, uporabniku se namreč ni treba več muditi z ročnim označevanjem vsakega izmed naloženih

letov, je z uporabo takšnih metod na boljšem tudi ponudnik storitev, saj lahko učinkoviteje poskrbi za dostop in ravnanje z naloženimi podatki.

Področje strojnega učenja v računalništvu se med drugim ukvarja tudi s sorodnimi klasifikacijskimi problemi, zato lahko reševanje naše naloge prevedemo na enega od že obstoječih pristopov. Podatke o letih črpamo iz interne zbirke podjetja Naviter ter nato nad njimi gradimo ustrezne napovedne modele, s katerimi lahko klasificiramo posamezne lete v ciljne kategorije; za vsak let določimo, v katerem tipu letalne naprave je bil zabeležen. Seveda klasifikacije ne izvajamo nad surovimi podatki, saj so v splošnem zapisi GPS lahko precej nenatančni, hkrati pa izolirana nekajsekundna podzaporedja točk pri zračnem gibanju niso dovolj zanesljiv indikator dogajanja. Posledično predstavlja velik problem definiranje ustreznih atributov na precej omejeni zbirki podatkov, ki jo imamo na voljo za učenje.

1.1 Cilji in struktura diplomske naloge

V okviru te diplomske naloge zato želimo odgovoriti na naslednja vprašanja:

1. Ali z metodami strojnega učenja lahko zanesljivo napovemo karakteristike zapisa leta in se tako izognemo ročnemu dodajanju informacij?

2. Katere metode dosežejo najvišjo točnost pri napovedovanju?

3. Kateri atributi najbolj pripomorejo k uspešnemu zaznavanju ciljnih razredov?

Vsebina diplomske naloge je razdeljena na tri dele. V uvodnem poglavju opišemo različne možne pristope h klasifikaciji vzorcev v pripadajoče razrede z uporabo metod strojnega učenja: logistične regresije, metode podpornih vektorjev, naključnih dreves, metode k najbližjih sosedov, dotaknemo pa se tudi uporabe (osnovnih) nevronskih mrež na takšnem tipu podatkov. Drugi del vsebuje pregled naše zbirke učnih podatkov skupaj z opisom metod, s

katerimi iz surovega zapisa lokacij GPS pridobimo izbrano množico atributov. V zadnjem delu predstavimo rezultate, ki smo jih uspeli doseči s prej omenjeno učno množico. V okviru diskusije skušamo interpretirati, kako se izračunani najpomembnejši atributi primerjajo s človeško klasifikacijo. Nazadnje predstavimo še možnosti za nadaljnje delo na tem področju.

Poglavje 2

Metode

V tem poglavju bralcu predstavimo kratek pregled področja klasifikacije v sklopu strojnega učenja ter nekaj najpomembnejših metod, ki smo jih uporabili za reševanje našega problema. Na koncu opišemo še eno izmed metod za primerjavo modelov strojnega učenja, zgrajenih z različnimi metodami in parametri, ter uporabljene mere uspešnosti.

Z uporabo pristopov, opisanih v tem poglavju, želimo najti čim bolj optimalen način reševanja danega problema; konkretno, poiskati želimo metodo ter njene parametre, s katerimi bomo lahko kar se da natančno klasificirali dane primerke v ciljne razrede.

2.1 Nadzorovano učenje in klasifikacija

Ko govorimo o nadzorovanem strojnem učenju, v splošnem mislimo na metode, ki za učenje uporabljajo vnaprej pripravljene vzorce atributov z označenimi ciljnimi vrednostmi. Na podlagi teh učnih primerov se inteligentni sistem sam nauči čim bolj točno razvrščati primerke v množico ciljnih vrednosti [26, 18].

Klasifikacija je eden od problemov, ki jih lahko rešujemo s prej opisanim pristopom. Podano imamo učno množico objektov, ki so opisani z n dimenzionalnimi vektorji. Iz njih želimo zgraditi model, ki bo sposoben za še

nevidene primere čim bolj pravilno napovedati, kateremu od končno mnogo ciljnih razredov le-ti pripadajo [18]. Za določanje razredov model uporabi odločitveno funkcijo, ki slika iz prostora vhodnih atributov v množico razredov. Obstaja več načinov, kako tako funkcijo definirati; lahko je podana že vnaprej ali pa se jo mora sistem naučiti iz podatkov. V okviru tega razdelka se bomo osredotočili predvsem na drugo možnost ter si pogledali nekatere od najpogosteje uporabljanih metod na tem področju: logistično regresijo, metodo podpornih vektorjev, naključne gozdove, metodo najbližjih sosedov ter umetne nevronske mreže.

Nekateri od omenjenih modelov v svoji osnovni različici podpirajo le binarno klasifikacijo in so torej sposobni določiti le, če posamezni primerek določenemu razredu pripada, oziroma ekvivalentno, kateremu izmed dveh možnih razredov pripada. Ko se ukvarjamo z večrazrednimi domenami (kot v našem primeru), se moramo odločiti za eno od dveh strategij določanja napovedanega razreda [6]. Metoda en-proti-ostalim za vsak razred ustvari svoj klasifikator; pri učenju primerke tega razreda obravnavamo kot pozitivne, ostale pa kot negativne. Strategija en-proti-enemu deluje ravno obratno: vsakega od n ciljnih razredov primerjamo s preostalimi $n - 1$ razredi in za vsakega zgradimo svoj (binarni) klasifikator. Drugi pristop je v določenih primerih lahko natančnejši, prvi pa je pri velikem številu ciljnih razredov hitrejši, saj je potrebnih bistveno manj klasifikatorjev. Strategija en-proti-ostalim je še posebej primerna pri uporabi majhnih učnih množic, saj imamo tako na razpolago večje število negativnih učnih primerov.

2.1.1 Logistična regresija

Logistična regresija (angl. *logistic regression, LR*) je eden od načinov posplošitve linearne regresije za reševanje klasifikacijskih problemov [18, 22]. Uporablja se na učnih problemih, kjer lahko ciljna spremenljivka zavzame samo dva razreda; kot smo omenili že v uvodu v to poglavje, pa to ne omejuje uporabe tudi pri več razredih, temveč le pomeni, da moramo zgraditi dovolj veliko število različnih modelov.

Za razumevanje delovanja metode si moramo najprej pogledati osnovno idejo delovanja linearne regresije. Posamezni učni primerek, ki ga predstavimo kot vektor atributov, označimo z $\mathbf{x}_i \in \mathbb{R}^d$, pripadajoče ciljne vrednosti pa z $y_i \in \{-1, 1\}$. S poljubno minimizacijsko metodo poiščemo take koeficiente $\mathbf{w}^*$ linearne funkcije

$$f(\mathbf{x}_i) = \mathbf{w}^T \mathbf{x}_i = \hat{y}_i, \tag{2.1}$$

da je vsota kvadratnih napak med napovedanimi in dejanskimi vrednostmi napak učnih primerkov minimalna. To storimo z minimizacijo kriterijske funkcije, ki je definirana kot

$$J(\mathbf{w}) = \sum_i \sqrt{(y_i - \mathbf{x}_i^T \mathbf{w})^2}. \tag{2.2}$$

Pri posplošenih linearnih modelih (angl. *generalized linear models, GLM*) na levi strani enačbe (2.1) nad $\hat{y}_i$ uporabimo poljubno povezovalno funkcijo (angl. *link function*) l. Če $l(x) = x$ dobimo navadno enačbo linearne regresije, pri klasifikacijskih problemih pa običajno posežemo po posebni obliki logistične funkcije, ki zgladi prehod med obema podprostoroma, ki jih ustvari regresijska premica. Enačba logistične funkcije se glasi

$$l_L(x) = \frac{1}{1 + e^{-x}}. \tag{2.3}$$

Taka funkcija ima nekaj lepih lastnosti, ki jih lahko uporabimo:

- je odvedljiva,
- zaloga vrednosti je na intervalu $[0, 1]$,
- definirana je na celotni realni množici.

Enačba logistične regresije se torej glasi

$$l_L(f(\mathbf{x}_i)) = \frac{1}{1 + e^{-\mathbf{w}^\mathbf{T}\mathbf{x}}} = \hat{y}_i, \tag{2.4}$$

kjer $\hat{y}_i$ predstavlja verjetnost, da i-ti vektor atributov pripada prvemu izmed ciljnih razredov.

Pri uporabi metod, ki temeljijo na linearni regresiji lahko hitro pride do pretiranega prilagajanja modela učnim podatkom. To v praksi pomeni, da bo dobljena funkcija f preveč prilagojena učnim podatkom in se bo zaradi tega slabše posploševala na nove še nevidene primerke. Za zmanjšanje vpliva tega pojava se lahko poslužimo regularizacije (angl. *regularization*), ki skrbi za to, da elementi vektorja $\mathbf{w}$ ne postanejo preveliki. To storimo tako, da v kriterijsko funkcijo, definirano v enačbi (2.2), uvedemo dodatni člen, ki se glasi

$$\lambda \mathbf{w}^T \mathbf{w}. \tag{2.5}$$

Parameter λ, ki ga navadno določamo s prečnim preverjanjem, imenujemo stopnja regularizacije.

2.1.2 Metoda podpornih vektorjev

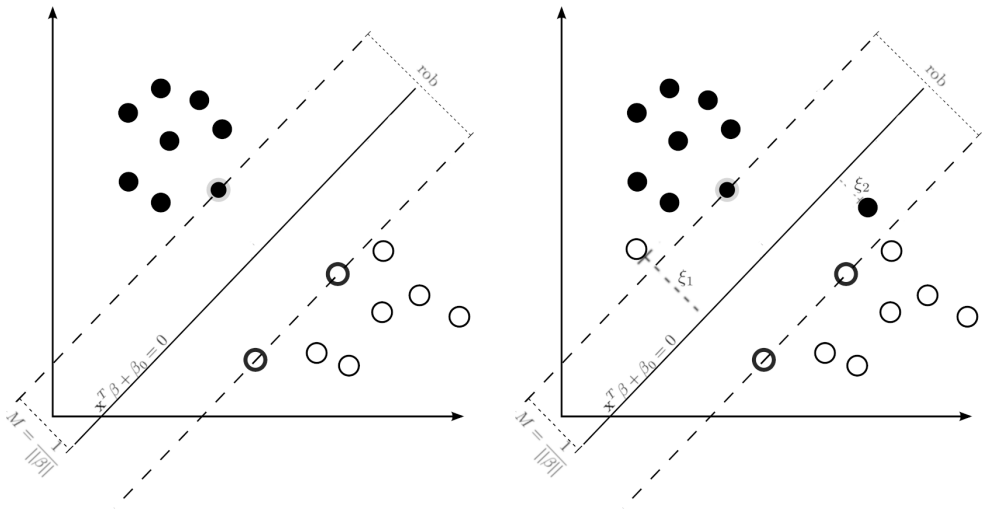
Metoda podpornih vektorjev (angl. *Support Vector Machine, SVM*) v večini primerov spada med točnejše metode strojnega učenja [25, 18]. Podobno kot pri logistični regresiji tudi pri tej metodi poskušamo poiskati hiperravnino, ki celotni prostor razmeji na dva manjša, od katerih vsak čim bolje opiše posamezni klasifikacijski razred. Za razliko od večine preostalih pristopov pa ima metoda podpornih vektorjev nekaj prednosti [22]:

1. Zaradi načina postavitve ločilne hiperravnine se dobro posploši tudi na nove podatke.

2. Podatkov pred učenjem ni potrebno pretirano obdelovati, saj je algoritem zasnovan tako, da dobro deluje neodvisno od količine atributov.

3. Metoda je dovolj splošna, da lahko predstavi kompleksnejše funkcije, hkrati pa je verjetnost pretiranega prilagajanja podatkom majhna.

Osnovni linearni algoritem določanja meje med razredoma je relativno preprost. Algoritem poišče enačbo take hiperravnine, ki prostor razdeli tako, da je razdalja med hiperravnino in najbližjimi predstavniki obeh razredov maksimalna. Primerke, ki ležijo najbliže hiperravnini, imenujemo tudi podpornimi vektorji. Primer takega problema je prikazan na sliki 2.1a. Pri izvajanju napovedi s SVM so vektorji, ki ležijo v enem podprostoru, označeni z -1, tisti v drugem pa z 1.



(a) Rešitev problema z ločljivimi razredi z uporabo SVM.

(b) Rešitev problema, ki s SVM ni idealno rešljiv.

Slika 2.1: Primer rešitev dveh problemov, ki sta z metodo podpornih vektorjev idealno rešljiva (slika 2.1a) oziroma ne (slika 2.1b). Z rdečimi oziroma zelenimi točkami so predstavljeni učni primeri dveh razredov. Sredinska črta s formulo $\mathbf{x}^T\beta + \beta_0$ predstavlja hiperravnino, ki ločuje oba razreda, črtkane črte pa so ji vzporedne in oddaljene za velikost roba M. Točke, ki ležijo na robu, so t.i. podporni vektorji. Razdalje, označene z ξ_i na sliki 2.1b, so vrednosti dopolnilnih spremenljivk, ki pripadajo napačno uvrščenim točkam.

Zgornjo intuitivno razlago lahko zapišemo tudi formalno [14]. Ponovno imamo podane z vektorji predstavljene učne primerke $\mathbf{x}_i \in \mathbb{R}^d$ ter pripadajoče oznake razredov $y_i \in \{1, -1\}$. Definiramo funkcijo f, ki bo predstavljala

oddaljenost posamezne točke $\mathbf{x}$ od dane hiperravnine.

$$f(\mathbf{x}) = \mathbf{x}^T\beta + \beta_0, \tag{2.6}$$

kjer je $\beta \in \mathbb{R}^d$ enotski vektor, ki služi kot normala hiperravnine, $\beta_0 \in \mathbb{R}$ pa prosti člen.

Na podlagi f lahko tudi določimo točke H, ki ležijo na sami hiperravnine:

$$H = \{\mathbf{x} \in \mathbb{R}^d : f(\mathbf{x}) = \mathbf{x}^T\beta + \beta_0 = 0\}. \tag{2.7}$$

Iz definicije funkcije f sledi tudi klasifikacijsko pravilo

$$G(\mathbf{x}) = \operatorname{sign} f(\mathbf{x}). \tag{2.8}$$

Če privzamemo, da so razredi ločljivi (tj. vsak od obeh podprostorov vsebuje natanko en ciljni razred, torej velja $y_i f(\mathbf{x}_i) > 0 \ \forall i$), potem obstaja hiperravnina, ki med obema ciljnima razredoma ustvari največji rob (angl. *biggest margin*). Iskanje rešitve lahko ponazorimo z naslednjim optimizacijskim problemom:

$$\begin{aligned} &\max_{\beta,\beta_0,||\beta||=1} M \\ &\text{pri pogojih} \quad y_i(\mathbf{x}_i^T\beta + \beta_0) \geq M \ \forall i. \end{aligned} \tag{2.9}$$

Ekvivalentno lahko zapišemo tudi

$$\begin{aligned} &\min_{\beta,\beta_0} ||\beta|| \\ &\text{pri pogojih} \quad y_i(\mathbf{x}_i^T\beta + \beta_0) \geq 1 \ \forall i. \end{aligned} \tag{2.10}$$

Pri tem je velikost robu enaka $M = 1/||\beta||$.

Recimo, da problem ni idealno rešljiv in bodo pri vsaki rešitvi vsaj nekatere od točk ležale v napačnem podprostoru. Primer take učne množice je prikazan na sliki 2.1b. Potem v problem, definiran z enačbo (2.9), uvedemo dopolnilne spremenljivke ξ_i, ki merijo relativne oddaljenosti vektorja $\mathbf{x}_i$ od pravilnega prostora glede na velikost robu M. Problem popravimo tako, da dopušča tudi rešitve, kjer pride do takšnega prekrivanja: obstoječe pogoje

spremenimo v $y_i(\mathbf{x}_i\beta + \beta_0) \geq M(1 - \xi_i)\ \forall i$, dodamo pa še pogoje za dopolnilne spremenljivke, ki zagotavljajo, da $\xi_i \geq 0\ \forall i$ in $\sum_i \xi_i \leq C$. Dopolnilne spremenljivke ξ_i predstavljajo delež velikosti robu, za katerega je (napačna) klasifikacija $f(\mathbf{x}_i)$ oddaljena od delilne hiperravnine. Do napačne klasifikacije pride, če $\xi_i > 1$. Parameter C je izbran tako, da zagotavlja, da je skupna relativna napaka $\sum_i \xi_i$ napačno napovedanih primerov manjša ali enaka C. Matematično gledano gre za konveksni optimizacijski problem, ki ga lahko zapišemo v naslednji obliki:

$$\begin{aligned} & \min_{\beta} ||\beta|| \\ \text{pri pogojih} \quad & y_i(\mathbf{x}_i^T\beta + \beta_0) \geq 1 - \xi_i \quad \forall i, \\ & \xi_i \geq 0 \quad \forall i, \\ & \sum_i \xi_i \leq C \end{aligned} \tag{2.11}$$

2.1.3 Naključni gozdovi

Naključni gozdovi (angl. *random forests, RF*) se pri napovedovanje naslanjajo na uporabo več odločitvenih dreves, ki z glasovanjem odločajo o uvrstitvi primerka v enega izmed možnih razredov [14, 18]. V splošnem so drevesa dobra izbira za izgradnjo modelov, saj lahko z njimi uspešno zajamemo tudi lastnosti bolj kompleksnih podatkov, hkrati pa ob zadostni globini delujejo relativno nepristransko. To potrjujejo tudi opažanja na realnih učnih problemih, kjer so naključna drevesa pogosto ena najtočnejših uporabljenih metod [11]. S povezovanjem več dreves v ansamble skušamo zmanjšati vpliv variance, saj lahko z vsakim na nekoliko drugačen način zajamemo lastnosti v podatkih. V ozadju takega načina učenja se skriva pristop *bagging*.

Bagging (krajše za *bootstrap aggregation*) je način ravnanja z učnimi podatki (t.i. meta-algoritem) pri učenju s poljubno ansambelsko metodo. Recimo, da imamo podano celotno učno množico $\mathcal{Z}$. Iz nje v fazi učenja generiramo nove manjše učne podmnožice $\mathcal{Z}^{*b}, b = 1, \ldots, B$, ki jih uporabimo za izgradnjo več modelov. Vsak od njih do svoje končne napovedi pride na

nekoliko drugačen način, odločitev celotnega ansambla pa je sprejeta z glasovanjem. Ker so modeli med seboj večinoma neodvisni, s takim pristopom znižujemo varianco, tj. občutljivost metode na razpoložljive učne podatke, končne napovedi.

Tak način učenja je še posebej primeren za odločitvena drevesa in njihovo povezovanje v naključne gozdove. Z *baggingom* namreč uspemo izpovprečiti napovedi več modelov, od katerih je vsak približno nepristranski, a občutljiv na osamelce (angl. *outliers*), ki so se nahajali v pripadajočih učnih podatkih.

Algoritem za izgradnjo naključnega gozda z algoritmom *bagging* je naslednji [14]:

1. Za $b = 1, \ldots, B$:
   (a) Z algoritmom *bagging* generiramo učno množico $\mathcal{Z}^{*b}$.
   (b) Zgradimo drevo T_b na učnih podatkih $\mathcal{Z}^{*b}$ tako da v listih rekurzivno ponavljamo naslednje korake, dokler ne dosežemo omejitve globine ali najmanjšega dovoljenega števila vzorcev v listu:
      i. Izmed vseh p atributov jih izberemo $m < p$.
      ii. Izmed izbranih m atributov poiščemo takega, da ponuja najboljšo delitev vozlišča.
      iii. Razdelimo vozlišče.
2. Z množico $\{T_b\}_{b=1}^{B}$ tvorimo ansambelski model.

2.1.4 Metoda najbližjih sosedov

Za razliko od do sedaj opisanih pristopov metoda k najbližjih sosedov (angl. *k-nearest neighbours*, *k-NN*) deluje na nekoliko drugačen način: za delovanje ne potrebuje vnaprej zgrajenega modela, temveč se napovedi računajo ob dejanskih poizvedbah glede na dane učne podatke [14, 18]. Tak način napovedovanja je poznan pod imenom leno učenje (angl. *lazy learning*, tudi *memory-based learning*). Prednost takega pristopa je, da za učenje ne porabimo nič dodatnega časa. Po drugi strani pa lahko kot dve največji slabosti

navedemo, da moramo v pomnilniku ves čas hraniti celotno učno množico, saj ta ostaja neposplošena, posledica česar pa je tudi velika časovna kompleksnost napovedovanja. Velja pa omeniti tudi, da nam zaradi takega načina napovedovanja, ne glede na število možnih razredov, ni potrebno graditi več modelov, v kar nas sicer prisilita tehniki en-proti-ostalim in en-proti-enemu.

Postopek določanja razreda neke točke je naslednji. Recimo, da želimo točko $\mathbf{x}_0$ klasificirati v enega izmed razredov iz $\mathcal{C} = \{C_1, \ldots, C_m\}$. Poiščemo k točk, za katere poznamo ciljne razrede in ki so najbližje $\mathbf{x}_0$ glede na vnaprej določeno metriko (npr. evklidsko razdaljo). Naj bodo to točke $\mathbf{x}_1, \ldots, \mathbf{x}_k$, ki pripadajo razredom $c_1, \ldots, c_k$. Napovedani razred c_0 za točko $\mathbf{x}_0$ določimo z glasovanjem k najbližjih točk, kar lahko formalno zapišemo kot

$$c_0 = \arg\max_{c \in \mathcal{C}} \sum_{i=1}^{k} \delta(c, c_k). \tag{2.12}$$

V zgornjem zapisu funkcija $\delta(i, j)$ ustreza Kroneckerjevi delta funkciji, torej

$$\delta(i, j) = \begin{cases} 0, & \text{če } i \neq j, \\ 1, & \text{če } i = j. \end{cases} \tag{2.13}$$

Tako kot ostali pristopi je tudi metoda najbližjih sosedov občutljiva na preveliko oziroma premajhno prilagajanje podatkom. Za razliko od linearne regresije tu učinke tega pojava izničujemo z ustrezno izbiro parametra k; premajhne vrednosti (npr. $k = 1$, ki obravnava samo najbližjega soseda) lahko vodijo v pretirano prilagajanje, med tem ko prevelike vrednosti aproksimirajo napovedi z večinskim razredom. Navadno izberemo $k = 5$ ali $k = 10$, pri večjem številu atributov pa se splača razmisliti tudi o $k = \sqrt{d}$, kjer je d dimenzija prostora učnih primerkov.

Podobno lahko na kvaliteto napovedi vpliva tudi izbira metrike. Dve najpogostejši metriki sta manhattanska in evklidska razdalja. Naj bosta $x, y \in \mathbb{R}^n$. Potem je evklidska razdalja med točkama enaka

$$d_2(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}, \tag{2.14}$$

manhattanska pa

$$d_1(x, y) = \sum_{i=1}^{n} |x_i - y_i|. \tag{2.15}$$

Zgornji enačbi sta posebna primera razdalje Minkowskega. Ta je v odvisnosti od parametra p definirana kot

$$d_p(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}. \tag{2.16}$$

2.1.5 Umetne nevronske mreže

Umetne nevronske mreže (angl. *artificial neural networks*, *ANNs*) so področje v metodah strojnega učenja, ki se je posebej uveljavilo šele v zadnjih letih. Temeljijo na simulaciji človeških celic – nevronov [22, 18, 24]. Zaradi obsežnosti področja se bomo v okviru te naloge ukvarjali le z eno izmed preprostejših oblik, to je z večslojnimi perceptronskimi (angl. *Multi-layer Perceptron*, *MLP*) mrežami.

Osnovni gradnik vsake nevronske mreže so takoimenovane enote (angl. *units*), ki ponazarjajo nevrone. Posamezne enote v strukturi celotne mreže predstavimo kot grafovska vozlišča, ki jih med seboj povežemo z uteženimi povezavami, ki predstavljajo sinapse, služijo pa prenašanju aktivacij (angl. *activation*) med vozlišči. Naj bo vozlišče i povezano z j, $w_{i,j}$ pa utež na tej povezavi, a_i pa vrednost signala, ki ga oddaja i. Potem je vhodna vrednost vozlišča j enaka uteženi vsoti nevronov na prejšnji plasti, torej

$$\text{in}_j = \sum_i w_{i,j} a_i + w_j, \tag{2.17}$$

kjer i teče po vseh vozliščih, ki imajo izhodno povezavo z j, w_j pa je lastna aktivacijska vrednost vozlišča j. Nad izračunano vrednostjo in_j nato

uporabimo aktivacijsko funkcijo (angl. *activation function*) g, da dobimo izhodno vrednost enote j:

$$a_j = g(\text{in}_j). \tag{2.18}$$

Tipično za funkcijo g vzamemo pragovno funkcijo (angl. *treshold function*), ki pretvori vhodno vrednost v število na intervalu med 0 in 1, odvisno od tega, če je bila mejna vrednost dosežena ali ne. Ena izmed možnosti je, da vzamemo nezvezno funkcijo $g(x) = \operatorname{sign} a_j$, vendar pa v praksi večkrat vzamemo sigmoidno funkcijo, definirano v enačbi (2.3), iz podobnih razlogov kot pri logistični regresiji. Pogosto se uporablja tudi funkcija ReLu (okrajšava za *rectified linear unit*), definirana kot

$$\text{ReLu}(x) = \begin{cases} 0, & \text{če } x < 0, \\ x, & \text{če } x \geq 0. \end{cases} \tag{2.19}$$

Glede na način povezave med vozlišči se nevronske mreže delijo na dva tipa: usmerjene (angl. *feed-forward*) ter rekurenčne nevronske mreže (angl. *recurrent neural networks, RNN*). Prve lahko predstavimo v obliki usmerjenega acikličnega grafa (angl. *directed acyclic graph, DAG*), vsako vozlišče pa kot vhode dobiva samo podatke vozlišč iz prejšnjih plasti in jih pošilja kasneje ležečim vozliščem. Rekurenčna mreža deluje ravno nasprotno – svoje izhode vrača tudi vozliščem, od katerih prejema vhode. Rezultati takih oblik mrež so odvisni tudi od vhodnih atributov, ki so jih prejele v prejšnjih iteracijah, zato pravimo, da imajo lastnost imenovano kratkoročni spomin (angl. *short-term memory*).

Pri (linearnih) klasifikacijskih problemih po navadi zadoščajo že nevronske mreže z dvema plastema brez dodatnih skritih nivojev, v praksi pa se pogosto uporabljajo tudi tri- ali štirinivojske mreže, saj lahko slednje aproksimirajo katero koli ciljno funkcijo s poljubno natančnostjo.

2.2 Normalizacija podatkov

Večina od opisanih metod deluje ne deluje najbolje, če vrednosti na različnih oseh med seboj niso primerljive. Do težav pride, če so podatki na različnih oseh navedeni v različnih skalah. Temu se lahko izognemo s postopkom normalizacije (angl. *normalization*), ki podatke v vsaki od dimenzij porazdeli približno standardno normalno [22]. Postopek normalizacije j-te dimenzije z n učnimi primerki je naslednji:

1. Izračunamo povprečje $\mu_j = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_{j,i}$.
2. Izračunamo standardno deviacijo $\sigma_j = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_{j,i} - \mu_j)^2}$.
3. Točke transformiramo v skladu z ravnokar določeno standardno normalno porazdelitvijo osi j: $\mathbf{x}_{j,i}$ postane $(\mathbf{x}_{j,i} - \mu_j)/\sigma_j$.

2.3 Prečno preverjanje

Po končanem učenju si želimo naše modele seveda primerjati. To sicer lahko storimo kar na celotni učni množici, vendar pa to ni najboljša ideja, naš model je namreč prilagojen prav tem podatkom. Če imamo na voljo dovolj podatkov, lahko manjši delež le-teh izločimo in jih uporabljamo samo za testiranje. Včasih pa ni možno niti to. V takih primerih se poslužimo tehnike, imenovane k-kratno prečno preverjanje (angl. *k-fold cross-validation*) [14]. Postopek testiranja je naslednji:

1. Ustvarimo razbitje množice učnih podatkov $\mathcal{Z}$ na k manjših enako velikih podmnožic $\mathcal{Z}_i, i = 1, \ldots, k$. Če poleg tega zahtevamo, da so posamezni razredi približno enako zastopani, takemu pristopu pravimo stratificirano (angl. *stratified*) k-kratno prečno preverjanje.
2. Za $i = 1, \ldots, k$:
   (a) Na učni množici $\bigcup_{j \neq i} \mathcal{Z}_j$ zgradimo model M_i.

(b) Točnost modela acc_i preverimo s klasifikacijo učnih primerkov iz množice $\mathcal{Z}_i$.

3. Predvidena točnost modela je povprečje posameznih točnosti: $\text{acc} = \frac{1}{k}\sum_{i=1}^{k}\text{acc}_k$.

Najpogostejše vrednosti za parameter k so $k = 5$ ali $k = 10$, kar da dovolj velike vzorce, da so statistično gledani dober približek dejanske točnosti, hkrati pa še vedno ohranja sprejemljive čase testiranja [22].

Nadgradnja tega pristopa, ki omogoča primerjanje modelov, zgrajenih z različnimi metodami, je interno prečno preverjanje (angl. *nested cross-validation*, tudi *internal cross-validation*) [18]. Pri tem pristopu poleg učne in testne množice uvedemo še tretjo, validacijsko. To storimo tako, da celotno množico razdelimo na dva dela, notranjega in testnega. Nad notranjim delom podatkom v okviru klasičnega k-kratnega prečnega preverjanja izberemo najboljše parametre za izbrano metodo. Z zunanjim prečnim preverjanjem nato nad najboljšim modelom izvedemo še testiranje, ki predvidi napovedno točnost. S takim pristopom se izognemo problemu, ko zaradi uporabe istih podatkov za izbiro parametrov in testiranje točnosti zaradi prilagajanja podatkom vračamo pretirano optimistične ocene [7]. Celotni postopek lahko nekoliko bolj formalno zapišemo z naslednjim algoritmom:

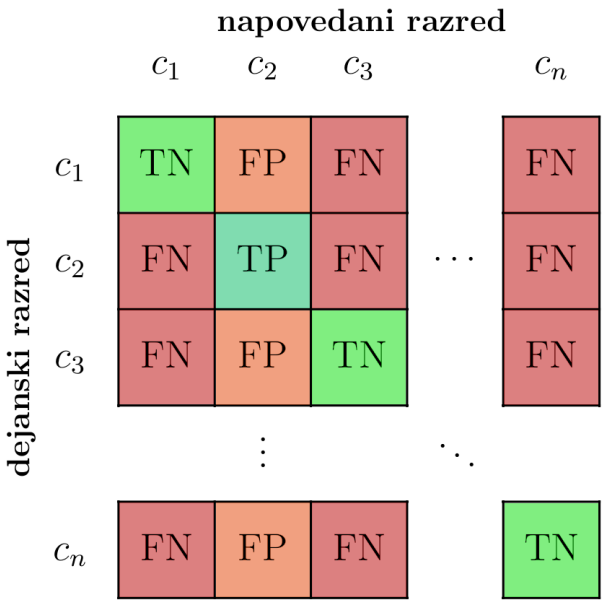
1. Ustvarimo razbitje celotne množice $\mathcal{Z}$ na k podmnožic $\mathcal{Z}_i$.

2. Za $i = 1, \ldots, k$:

   (a) Definiramo notranjo množico $\mathcal{I} = \bigcup_{j \neq i} \mathcal{Z}_j$ in testno $\mathcal{T} = \mathcal{Z}_i$.

   (b) Izvedemo k-kratno prečno preverjanje nad $\mathcal{I}$, s katerim določimo optimalne parametre modela.

   (c) Z učno množico $\mathcal{T}$ preizkusimo točnost najboljšega modela.



Slika 2.2: Matrika zmot za klasifikacijski problem z n razredi pri obravnavi razreda c_2. S TP označimo pravilno identificirane pozitivne primerke, s FP pozitivno označene negativne vzorce, s FN nepravilno identificirane pozitivne primere in s TN pravilno določene negativne osebke.

2.4 Mere uspešnosti

Za primerjavo modelov moramo najprej podati nekaj smiselnih mer uspešnosti. Uporaba vseh opisanih mer temelji na matriki zmot (angl. *confusion matrix*), ki prikaže povezavo med dejanskimi in napovedanimi razredi. Primer take matrike je prikazan na sliki 2.2.

V okviru te naloge smo uporabili dve različni meri uspešnosti, klasifikacijsko točnost in mero F_1 [18]. V nadaljevanju bodo opisane mere za dva klasifikacijska razreda. Pri obravnavi večjega števila le-teh izračunamo želeno mero m_{c_i} za vsak razred c_i posebej in rezultat povprečimo, torej je skupna mera za vse razrede enaka

$$m = \frac{1}{n} \sum_{i=1}^{n} m_{c_i}. \tag{2.20}$$

V preostanku razdelka vzorce glede na pravilnost njihove klasifikacije in dejanski razred delimo v štiri skupine:

1. TP (*true positive*) predstavlja število pravilno identificiranih pozitivnih vzorcev.

2. FP (*false positive*) predstavlja število nepravilno identificiranih negativnih vzorcev.

3. FN (*false negative*) predstavlja število nepravilno identificiranih pozitivnih vzorcev.

4. TN (*true negative*) predstavlja število pravilno identificiranih negativnih vzorcev.

Klasifikacija točnost (angl. *classification accuracy*) CA je definirana kot

$$\text{CA} = \frac{TP + TN}{TP + FP + FN + TN}. \tag{2.21}$$

Pove nam, kolikšen delež vseh primerkov je bil razvrščen pravilno.

Kot alternativo lahko uporabimo mero priklica (angl. *recall*) in natančnosti (angl. *precision*). Priklic pove, kolikšen delež izmed vseh pozitivnih primerov smo pravilno identificirali, kar formalno zapišemo kot

$$\text{recall} = \frac{TP}{TP + FN}. \tag{2.22}$$

Nasprotno kot priklic nam natančnost pove, kolikšen delež pozitivno identificiranih primerov dejansko pripada temu razredu. Z uporabo že znanih oznak lahko to zapišemo kot

$$\text{precision} = \frac{TP}{TP + FP}. \tag{2.23}$$

Družina mer, ki združuje natančnost in priklic, se imenuje F_β (angl. *F-score*, tudi *F-measure*). Pozitivni realni parameter β določa vpliv natančnosti. Enačba metrike F_β se glasi

$$F_\beta = (1+\beta) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}. \tag{2.24}$$

Navadno uporabimo $\beta = 1$, torej

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}. \tag{2.25}$$

2.5 Programska orodja

Večina kode za reševanje zastavljenega klasifikacijskega problema je napisana v programskem jeziku Python, ki se v zadnjih letih veliko uporablja za reševanje problemov, povezanih s strojnim učenjem. Za učenje uporabimo knjižnico *scikit-learn* [21], ki ponuja že implementirane nekatere od metod, ki smo jih navedli v tem poglavju. Temelji na paketih *SciPy* [17] in *NumPy* [20], ki uporabniku omogočata dostop do širokega spektra matematičnih funkcionalnosti. Zaradi hitrejšega izvajanja je jedro obeh knjižnic napisano v programskem jeziku C oziroma Fortran.

V začetni fazi odkrivanja lastnosti v podatkih se poslužimo tudi paketa *pandas* [19], del vizualizacij je ustvarjen s paketom *matplotlib* [16], del pa v okolju *Orange* [9].

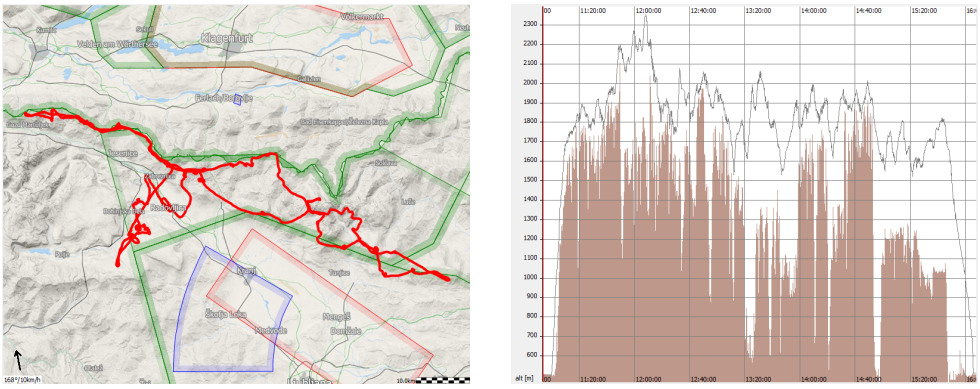
Poglavje 3

Podatki

3.1 Opis podatkov

Podatki v obliki surovih neoznačenih zapisov GPS so pridobljeni iz interne baze podjetja Naviter. Celotna baza obsega približno 300 tisoč zapisov v formatu IGC. Gre za tekstovni format, kjer vsaka vrstica predstavlja svoj zapis določenega tipa; nekateri so namenjeni beleženju meta-podatkov o letu, drugi zapisu dejanskih lokacij, spet tretji pa predstavljajo kontrolne zapise, s katerimi je mogoče preveriti integriteto datoteke [13]. Za beleženje tovrstnih zapisov so bili včasih potrebni namenski zapisovalniki, namenjeni tekmovalcem, do danes pa se je ta tehnologija že bistveno razširila, poleg tega pa je zapisovanje podprto z uporabo specializiranih programov tudi na pametnih telefonih. Format že od samega začetka podpira dodajanje opomb v obliki zapisov tipa *H* (angl. *H records*), v katerih lahko zabeležimo model letalne naprave (konkretno to storimo z vrstico `HFGTY`), vendar pa v praksi to polje velikokrat ostane neuporabljeno ali pa so v njem zapisani zastareli podatki. Poleg tega je bil format v svoji prvi različici leta 1993 razvit primarno za jadralce, med tem ko so bili padalci in zmajarji v drugem planu. Ker se standard od takrat ni bistveno spreminjal, še vedno manjka standardizirano polje za označevanje vrste letalne naprave.

Zaradi navedenih razlogov se pri razvrščanju letov glede na tipe plovil ne



(a) Lokacije zapisa IGC (b) Višinski profil in višina terena

Slika 3.1: Primer rekonstruiranega zapisa IGC. Slika 3.1a prikazuje opravljeno pot (na sliki označena z rdečo), slika 3.1b pa pripadajoč višinski profil skupaj z dodanimi višinami terena v vsaki izmed zapisanih točk.

moremo zanašati na podatke, ki jih vnese uporabnik. Še vedno pa imamo na voljo vrstice, v katerih je zapisana pozicija GPS, t.i. zapisi tipa *B* (angl. *B records*). V osnovni različici ti zapisi vsebujejo naslednje podatke:

- čas v dnevu (datum je naveden samo na začetku datoteke kot zapis tipa H),
- geografska širina,
- geografska dolžina,
- oznaka veljavnosti zapisa,
- višina po barometričnem višinomeru (angl. *pressure altitude*) ter višina po modelu GPS (angl. *GPS altitude*).

Vrstice tega tipa je mogoče razširiti s še dodatnimi uporabniško definiranimi polji, vendar pa se ta razlikujejo med zapisovalniki in zato niso nujno merodajna.

Gostota zapisa se v zadnjem času giblje med eno in štirimi sekundami, v nekaterih ekstremnih primerih, ki se navadno pojavljajo predvsem pri starejših zapisih, pa lahko znaša tudi do 20 sekund. Upoštevajoč dejstvo, da

tudi same lokacije pridobljene s sistemom GPS niso popolnoma natančne, s hitrejšim zapisovanjem lokacij ne pridobimo bistveno več podatkov. Vse naštete lastnosti nam omogočajo, da natančno rekonstruiramo let, iz njega računamo določene nove atribute za posamezne točke zapisa (kot so smer in jakost vetra ter višina nad terenom (angl. *above ground level, AGL*) ter na poljubne načine konstruiramo atribute iz razpoložljivih podatkov. Primer rekonsturiranega leta je prikazan na sliki 3.1.

3.2 Priprava podatkov

3.2.1 Klasifikacijski razredi

V okviru te naloge želimo z uporabo metod strojnega učenja razviti model, ki bo podatke sposoben razvrstiti v enega izmed naslednjih štirih razredov:

1. jadralno letalo
2. jadralno padalo
3. jadralni zmaj
4. pešec[1]

Posamezni razredi se med seboj izključujejo, zato določen zapis pripada natanko eni kategoriji.

3.2.2 Preobdelava podatkov

Kot smo že omenili, so vsi naši podatki neoznačeni. Zaradi tega iz celotne množice naključno pridobimo približno 150 vzorcev iz vsakega razreda ter jih ročno označimo. V nadaljevanju bomo te podatke obravnavali kot našo učno

[1]Medtem ko pešci niso ravno ciljna publika proizvajalcev naprav za beleženje letalskih letov, se včasih, še posebno pri padalcih, zgodi, da se na hrib odpravijo peš (t.i. *Hike & Fly*) in ta del poti tudi posnamejo. Zaradi tega želimo take zapise zaznati in primerno označiti.

množico. Menimo, da izbrani podatki kar se da dobro predstavljajo različne možne zunanje dejavnike, ki lahko vplivajo na karakteristike leta. Med drugim se razlikujejo v trajanju, nivoju znanja pilota, vremenu, zmogljivostih letalne naprave in namenu leta.

Preobdelava podatkov iz surovega zapisa v datoteki IGC poteka v dveh fazah. V prvi fazi podatke iz tekstovnega zapisa preoblikujemo v obliko, ki je primerna za nadaljnjo obdelavo z računalnikom. Dodali smo tudi nekatere naknadno izračunljive podatke (npr. višino nad terenom).

V drugi nekoliko obsežnejši fazi iz dopolnjenih zapisov pridobimo atribute, ki jih uporabimo pri strojnem učenju. Zaradi nenatančnosti, ki je že v osnovi prisotna v sistemu GPS, se osredotočamo predvsem na značilnosti, ki jih je mogoče izračunati v daljših časovnih intervalih (npr. dolgih vsaj minuto), da se lahko napake med seboj vsaj nekoliko izpovprečijo.

Omenimo še, da zaradi majhnega števila označenih vzorcev poskusimo tudi umetno generirati nove primerke za učenje z delitvijo originalnih zapisov na manjše enote z začetkom ob naključnih intervalih. Na ta način ustvarimo še nekoliko bolj raznolike podatke, saj vsaka novo ustvarjena datoteka vsebuje le manjši in nekoliko bolj specifičen del leta, ter povečamo skupno velikost učne množice iz šeststo na nekaj tisoč zapisov.

3.3 Prostor učnih podatkov

V nadaljevanju tega razdelka predstavimo osnovne atribute, ki so prisotni v končnem modelu. Takih spremenljivk je 32 in so navedene v tabeli 3.2, dodamo pa jim še nekatere sestavljene atribute, ki jih dobimo s kombiniranjem več osnovnih. Slika 3.2 prikazuje podatke, opisane z navedenimi atributi, po opravljeni normalizaciji in izvedeni projekciji PCA v dve dimenziji. Kot lahko opazimo, so pri dani izbiri atributov podatki med seboj precej lepo ločeni: dokaj jasno je definirana meja med jadralnimi letali oziroma pešci ter preostalima razredoma, medtem ko pa so podatki zmajarjev in padalcev med seboj nekoliko bolj pomešani.

Tabela 3.1: Zastopanost najvplivnejših atributov v dvokomponentni projekciji PCA.

PC1		PC2	
Atribut	Delež	Atribut	Delež
75. kvantil hitrosti	0,270	25. kvantil vert. hitrosti	0,330
povprečna hitrost	0,260	povprečna hitrost izgube višine	0,298
povprečna hitrost planiranja	0,259	povprečna hitrost pridobitve višine	0,296
25. kvantil hitrosti	0,258	25. kvantil izgube višine	0,291
standardni odklon hitrosti	0,253	st. odklon hitrosti (30-min okno)	0,284

Obe komponenti PCA se izražata kot kombinaciji velikega števila atributov: v prvi ima 13 spremenljivk zastopanost večjo od 20 %, v drugih pa je takih 12. Projekcija uspe opisati 69 % skupne variance v podatkih. Poleg tega lahko opazimo, da je PC1 najbolj definirajo s horizontalnim gibanjem povezani atributi, PC2 pa tisti, ki opisujejo vertikalne spremembe. Točnejši pregled najvplivnejših atributov je prikazan v tabeli 3.1.

Nekoliko bolj napredna vizualizacija z algoritmom *FreeViz* [10], ki je namenjen dvodimenzionalni projekcij mnogoatributnih podatkov z upoštevanjem razredov vsebovanih vzorcev, je prikazana na sliki 3.3. Podobno kot že pri PCA lahko tudi tu opazimo lepo ločenost razredov, še posebej jadralnih letal in pešcev. Iz velikosti baznih vektorjev lahko razberemo, da v vodoravni smeri posamezne razrede najlepše ločujejo spremenljivke, ki temeljijo na hitrosti gibanja ter hitrosti spuščanja. K vertikalni separaciji, ki je sicer zaradi lege hiperravnin, s katerimi so razredi ločeni, manj pomembna, pa najbolj pripomorejo delež premočrtnega leta, povprečni čas kroženja (ki ločuje predvsem pešce od padalcev) ter povprečna finesa. Iz grafa lahko razberemo tudi, da večina atributov ne ustvari izrazite meje med razredi, saj so si njihovi bazni vektorji po smeri in velikosti zelo podobni; kljub temu pa med seboj še vedno nekoliko razlikujejo in zato vseeno pozitivno pripomorejo h kvaliteti ločevanja.

Tabela 3.2: Seznam uporabljenih osnovnih atributov.

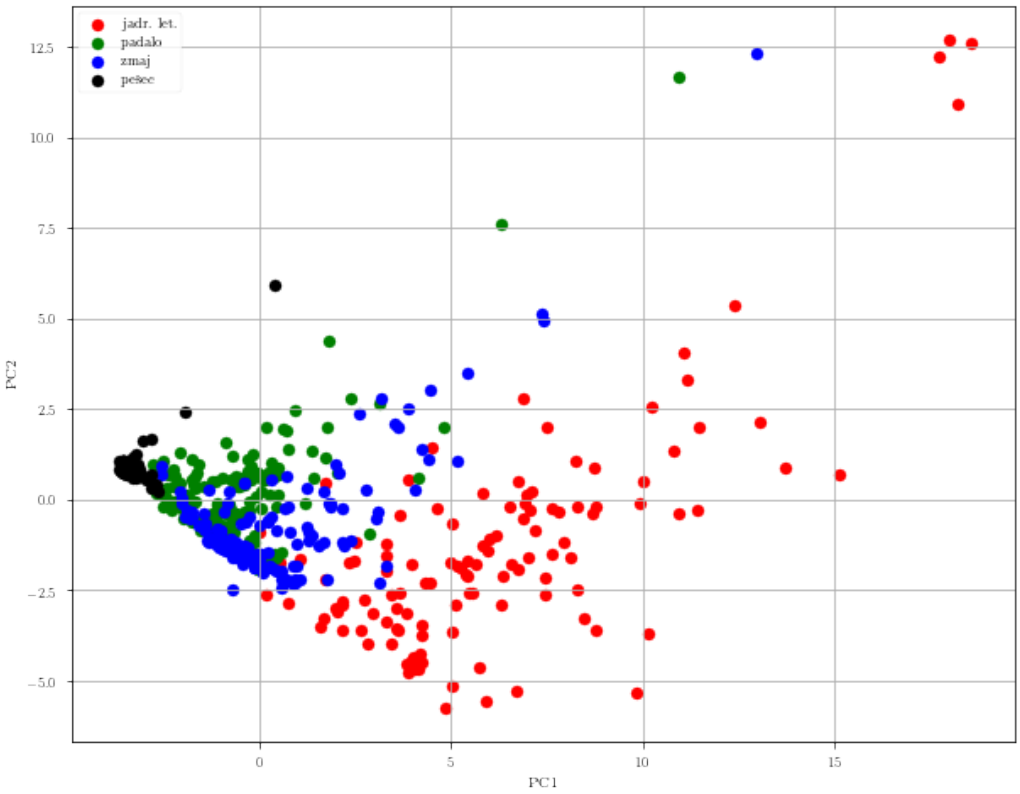
Atributi	
povprečna hitrost	najnižja hitrost v 5-minutnem oknu
najvišja hitrost	najvišja hitrost v 5-minutnem oknu
25. kvantil hitrosti	povprečna hitrost v 5-minutnem oknu
75. kvantil hitrosti	standardna deviacija hitrosti v 5-minutnem oknu
95. kvantil hitrosti	najnižja hitrost v 10-minutnem oknu
standardna deviacija hitrosti	najvišja hitrost v 10-minutnem oknu
razlika med najvišjo in najnižjo točko	povprečna hitrost v 10-minutnem oknu
povprečna hitrost pridobitve višine	standardna deviacija hitrosti v 10-minutnem oknu
povprečna hitrost izgube višine	najnižja hitrost v 30-minutnem oknu
25. kvantil pridobitve višine	najvišja hitrost v 30-minutnem oknu
25. kvantil izgube višine	povprečna hitrost v 30-minutnem oknu
standardna deviacija spremembe višine	standardna deviacija hitrosti v 30-minutnem oknu
povprečna hitrost planiranja[a]	delež premočrtnega[b] leta
povprečna finesa[c]	povprečna vertikalna hitrost pri premočrtnem letu
povprečna višina nad terenom	povprečno trajanje kroženja
standardna deviacija višine nad terenom	povprečna vertikalna hitrost v kroženju[d]

[a]Planiranje je poseben primer premočrtnega leta, pri katerem (navadno) letimo popolnoma v ravni črti proti ciljni točki, ne da bi se ozirali na možna dviganja.

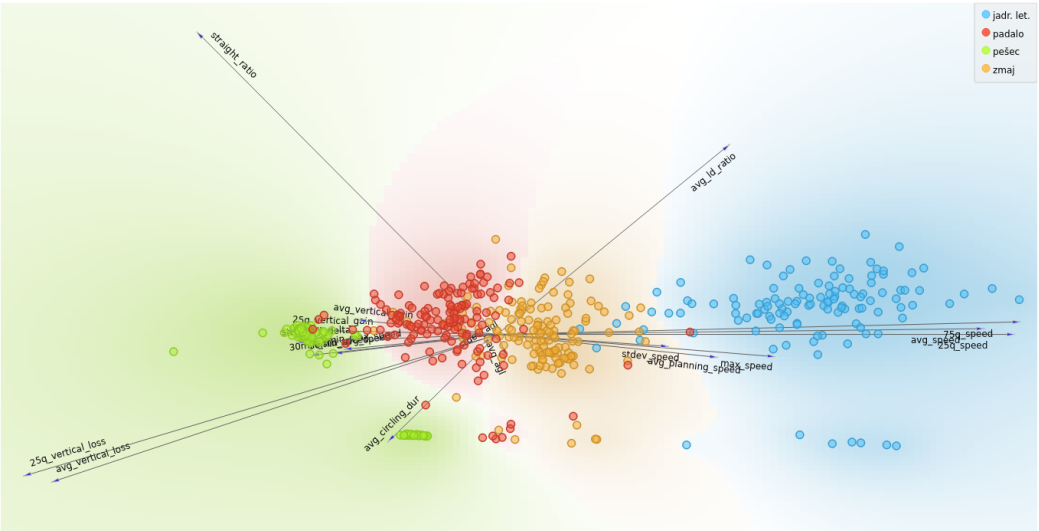
[b]Del leta, pri katerem ne prihaja do naglih odstopanj v smeri. Pri naši implementaciji dopuščamo spremembe do $8°/\,\mathrm{min}$.

[c]Finesa (angl. *lift-to-drag ratio*, okrajšano *L/D ratio*) pove, koliko kilometrov poti lahko opravimo za vsak izgubljeni kilometer višine.

[d]Letalne naprave brez motorja višino (navadno) pridobivajo v termičnih stolpih, ki se dvigajo zaradi temperaturnih razlik zraka. Ti so navadno ozki (premer do nekaj sto metrov), zato morajo letala krožiti na mestu, če želijo ostati v njihovem območju delovanja.



Slika 3.2: Vizualizacija podatkov po izvedeni normalizaciji in po uporabi dvokomponentne projekcije PCA. Iz grafa lahko vidimo, da je meja med jadralnimi letali ter ostalim delom podatkov precej jasno definirana, podobno lahko rečemo tudi za pešče. Po drugi strani pa so padalski in zmajarski podatki med seboj precej pomešani.



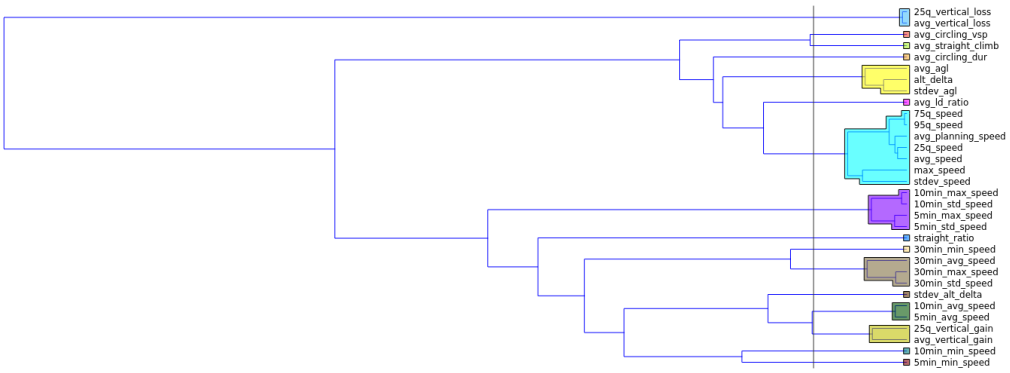
Slika 3.3: Večdimenzionalna vizualizacija podatkov z algoritmom *FreeViz* po izvedeni normalizaciji. Podobno kot pri PCA tudi tu opazimo relativno dobro ločenost razredov; do izrazitega prekrivanj prihaja le na meji med razredoma padal in zmajev.

3.4 Koreliranost atributov

V prejšnjem izdelku smo navedli, da je pri uporabi projekcije *FreeViz* prostor ključno definirajo le nekateri izmed izbranih atributov, ostali pa kažejo v podobne smeri ter so primerljivih velikosti. Vse to nakazuje na dejstvo, da so določeni pari spremenljivk med seboj visoko korelirani ter zato bistveno ne pripomorejo k opisovanju učnih vzorcev.

Našo tezo potrjuje tudi dendrogram gručenja na sliki 3.4, ki prikazuje združevanje atributov na podlagi Pearsonove razdalje med njimi ter načinom iskanja najbližjih gruč z uporabo Wardove razdalje. Drevo na sliki je odrezano dovolj zgodaj, da še ponuja dovolj majhne in podobne skupine, ki jih je mogoče tudi interpretirati. Navedimo samo nekatere najopaznejše lastnosti:

1. Najopaznejša skupina združuje spremenljivke, ki opisujejo hitrosti med celotnim letom: povprečno hitrost ter povprečno hitrost planiranja, vse izbrane kvantile hitrosti, njeno standardno deviacijo ter najvišjo



Slika 3.4: Dendrogram gručenja uporabljenih atributov z uporabo Pearsonove razdalje ter združevanjem na podlagi Wardove razdalje. Drevo odrežemo v točki, ki je dovolj zgodnja, da še ponuja dovolj veliko število skupin, hkrati pa zagotavlja dovolj velik razmak med zaporednima vejitvama.

vrednost. Podobno so med seboj povezani so tudi višinski atributi.

2. Korelirana sta tudi povprečna hitrost spuščanja ter 25. kvantil iste količine.

3. Visoka je tudi povezanost med atributi, ki opisujejo najvišjo hitrost in standardno deviacijo le-te v pet- oziroma desetminutnem oknu. Zanimivo je, da se povprečna hitrost v omenjenih oknih ne nahaja v ločeni gruči, kar kaže na velike odklone tudi v tako kratkih obdobjih merjenja.

4. V nasprotju s prejšnjo točko pa je tridesetminutno okno že dovolj dolgo, da se omenjeni atributi že združijo v eno samo skupino.

5. Minimalne hitrosti v oknih različnih dolžin so med seboj zelo raznolike, kar je predvidoma odvisno od nihanja kvalitete signala GPS.

6. Med atributi, ki predstavljajo različne količine, je korelacija majhna.

Poglavje 4

Rezultati

V tem poglavju opišemo uporabljene klasifikacijske modele, postopek iskanja optimalnih hiperparametrov ter rezultate testiranja. Posamezne metode primerjamo med seboj ter poskušamo razložiti, zakaj nekatere izmed njih delujejo bolje od drugih. Interpretirati poskušamo tudi smiselnost odvisnosti ciljnega razreda od najvplivnejših atributov pri logistični regresiji. Nazadnje zapišemo še opažanja glede uporabe različnih pristopov konstrukcije učnih primerov v procesu predobdelave podatkov.

4.1 Opis klasifikacijskih modelov

Zadanega klasifikacijskega problema smo se lotili z naslednjimi modeli.

Logistična regresija zgradi najenostavnejši model, ki ga je za razliko od ostalih pristopov enostavno interpretirati. To je tudi razlog, da metodo uporabimo za iskanje atributov, ki najbolj pozitivno na točnost razvrščanja.

Metoda podpornih vektorjev združuje hitrost klasifikacije, ki jo ponuja logistična regresija, hkrati pa na večini problemov deluje podobno dobro kot naključni gozdovi [11], zaradi česar so tovrstni modeli zelo primerni za praktično uporabo.

Naključni gozdovi za dan tip podatkov morda niso najprimernejša metoda, saj je zaradi zveznosti atributov težko določiti točno mejo delitve dimenzij. Kljub temu pa z dovolj velikimi gozdovi uspemo izpovprečiti delilne točke in tako dvigniti uspešnost metode na pričakovano stopnjo.

Metoda najbližjih sosedov utegne delovati relativno dobro, saj se podatki, kot smo opazili že na sliki 3.2, združujejo v relativno velike homogene skupine. V praksi ta metoda sicer ni najbolj uporabna, saj zahteva, da vse učne podatke vedno držimo v pomnilniku.

Nevronske mreže na strukturiranih linearnih podatkih niso najbolj običajna izbira, saj lahko večino karakteristik zajamemo že s kvalitetno konstruiranimi atributi. Kljub temu pa metodo uporabimo za primerjavo, če vseeno uspe zajeti kakšno novo zvezo med vhodnimi spremenljivkami, ki je ostali pristopi niso zaznali.

4.2 Rezultati testiranja

4.2.1 Nabor hiperparametrov

Izbranim pristopom smo z internim prečnim preverjanjem poiskali najuspešnejšo kombinacijo parametrov modela. Razpoložljivi parametri za posamezne metode so navedeni v tabeli 4.1. Pri logistični regresiji je v večini primerov edini parameter, ki se ga splača nastavljati, stopnja regularizacije; preverili smo vrednosti med 0 in 2 v korakih po 0,02, pri čemer te vrednosti določajo inverz stopnje regularizacije λ. Metoda podpornih vektorjev se lahko klasifikacije loti na način en-proti-ostalim, ki je opisan v Poglavju 2, ali pa poseže po namenskem algoritmu Crammerja in Singerja [8], ki lahko z enim modelom opravlja večrazredno klasifikacijo. Pri naključnih gozdovih smo preizkusili točnost pri uporabi manjšega števila dreves v gozdu (5 in 10) ter z večjimi $(1000, 5000, 10000)$, za katerega predvidevamo, da zaradi povprečenja delitvenih napak ponuja boljše rezultate. Metodo najbližjih sosedov smo preizkusili za relativno standardne vrednosti parametra $k = 5$ in

Tabela 4.1: Nabor parametrov posameznih metod za pri iskanju optimalne konfiguracije modelov.

Metoda	**Parameter**	**Nabor vrednosti**
LR	stopnja regularizacije	0,02, 0,04, ..., 2,00
SVM	tip večrazredne klasifikacije	en-proti-ostalim, Crammer-Singer
	omejitev skupne napake C	0,02, 0,04, ..., 2,00
RF	št. dreves	5, 10, 1000, 2500, 5000, 10000
k-NN	k	5, 10
	mera razdalje	d_1, d_2
ANN	št. slojev	1, 2
	velikost sloja	25, 50, 75, 100

$k = 10$. Preverili smo tudi smiselnost uporabe manhattanske oziroma evklidske razdalje. Pri nevronskih mrežah smo preizkusili eno- in dvoslojna modela različnih velikosti.

V tabeli lahko tudi opazimo odsotnost različnih jeder pri metodi podpornih vektorjev. Odločili smo se, da jih pri testiranju izpustimo, ker so pri začetnih preizkusih polinomska (do stopnje 4) ter krožna jedra (angl. *radial basis function kernel*) dosledno dosegala slabše rezultate kot linearne različice.

4.2.2 Rezultati testiranja

Rezultati najboljših konfiguracij omenjenih modelov z uporabo petkratnega internega prečnega preverjanja so navedeni v tabeli 4.2. Povprečja posameznih metod so vizualno prikazana tudi na sliki 4.1. Najuspešnejši model temelji na uporabi metode podpornih vektorjev z linearnim jedrom in načinom večrazredne klasifikacije Crammer-Singer. Tak rezultat je na nek način pričakovan: kot smo videli že na sliki 3.2, so razredi med seboj relativno dobro ločljivi, z metodo pa prostor uspemo razdeliti tako, da se lepo

Tabela 4.2: Povprečni rezultati najboljših metod v različnih iteracijah internega prečnega preverjanja ter njihovi pripadajoči odkloni. Metoda RF10 predstavlja naključne gozdove z največ deset drevesi, RF10k pa z največ 10 tisoč.

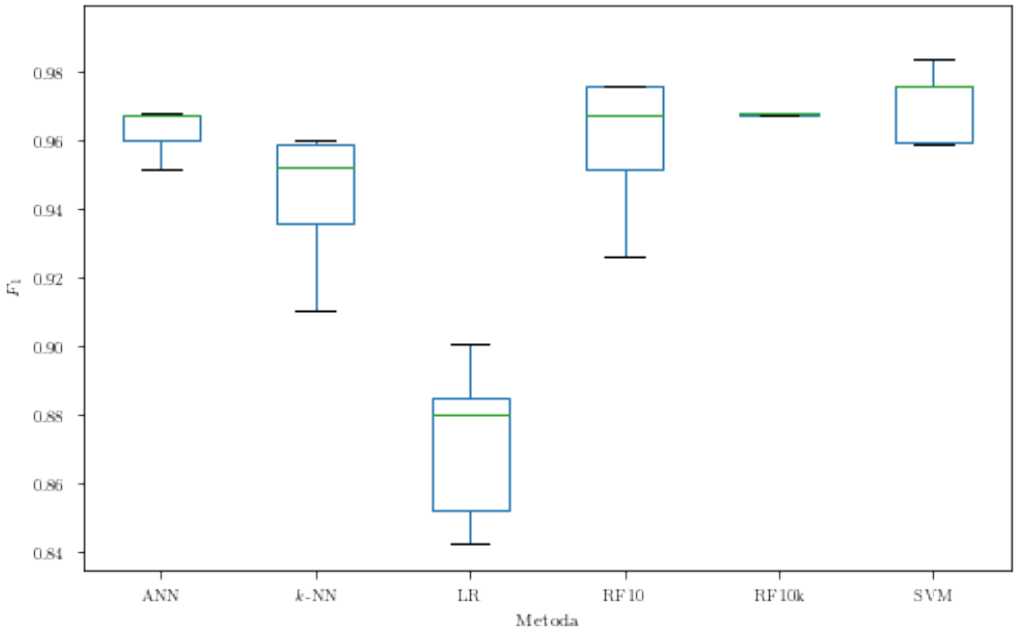
Metoda	**Povp. F_1**	**St. dev. F_1**	**Povp. CA**	**St. dev. CA**
k-NN	0,943	0,021	0,943	0,021
ANN	0,963	0,007	0,950	0,009
LR	0,872	0,024	0,873	0,021
RF10	0,959	0,021	0,963	0,014
RF10k	0,969	0,015	0,968	0,015
SVM	0,971	0,011	0,969	0,014

ločijo tudi pokrivanja med razredi padalcev in zmajarjev.

Podobno uspešna sta tudi oba modela, ki temeljita na naključnih gozdovih. Zanimivo je predvsem to, da klasifikator z desetimi drevesi relativno uspešno konkurira svojemu večjemu bratu z 2500-timi. Iz tega sklepamo, da so ne glede na začetni izbor podmnožice učnih primerov in atributov razredi lepo ločljivi. Model s tisoči dreves izvaja izredno konsistentne napovedi, kar načeloma pomeni, da dobro aproksimira funkcijo, ki ločuje posamezne razrede. Pretirano prilagajanje je sicer možno, a zaradi načina testiranja tudi precej neverjetno.

Uspešna je tudi metoda najbližjih sosedov, za $k = 5$ in ob uporabi manhattanske razdalje prav tako dosega uspešnost okrog 95 %, ki nepričakovano dobro zadane tudi napovedi v izhodišču grafa na sliki 3.2, kjer pride do velikega prekrivanja dveh razredov. Uspešnost za razreda peščev in letal pa po drugi strani ni vprašljiva, saj je v obeh primerih dominacija pripadnikov obeh razredov na pripadajočih podprostorih očitna.

Logistična regresija pri vseh delitvah učne množice dosega najslabše rezultate. Menimo, da je za to kriva lega podatkov v prostoru (razreda padal in zmajev sta ukleščena med preostala dva razreda) ter relativno malo učnih po-

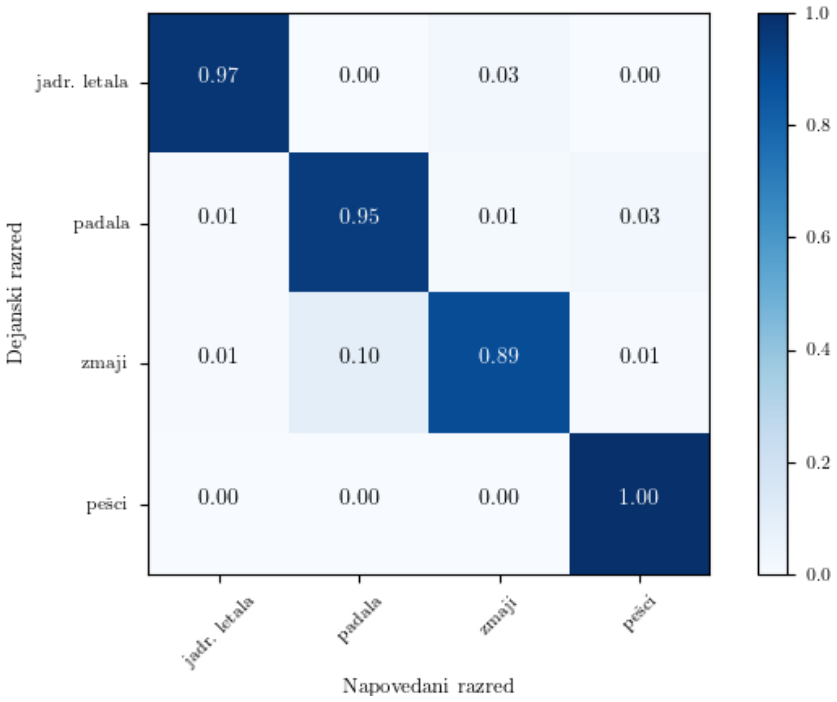


Slika 4.1: Vizualna predstavitev povprečnih rezultatov posameznih metod iz internega prečnega preverjanja iz tabele 4.2.

datkov. Zaradi prvega razloga odpove zanesljivost metode en-proti-ostalim, saj je težko potegniti mejo med prostoroma, kjer se ciljni razred nahaja na sredini, z ostalih strani pa ga obdajajo ostali. Zaradi majhne učne množice odpove tudi pristop en-proti-enemu, ki dosega še slabše rezultate od prej omenjenega pristopa.

Tudi alternativna metoda z uporabo preprostih nevronskih mrež doseže precej dobre rezultate. Mreža z dvema nivojema, od katerih je vsak velik 100 enot, po povprečni uspešnosti konkurira modelu naključnih gozdov. Odstopanja od povprečja so relativno stabilna napram ostalim modelom (v tem pogledu je boljša le klasifikacija z velikimi naključnimi gozdovi). Zaradi tega je tak način klasifikacije primeren, če si želimo uporabiti metodo, ki ima relativno visoko natančnost ter vrednosti konsistentno dobro napoveduje tudi za različne nabore vhodnih podatkov.

Menimo, da je za uporabo v končnih aplikacijah najbolj smiselna uporaba modela, ki uporablja algoritem SVM; klasifikacija je hitra (časovna komple-



Slika 4.2: Kontingenčna matrika deležev napovedanih razredov pri prečnem preverjanju metode podpornih vektorjev z linearnim jedrom, parametrom $C = 0,6$ in načinom večrazredne klasifikacije Crammer-Singer.

ksnost je linearna glede na število atributov), rezultati pa so primerljivi z velikimi naključnimi gozdovi. Po drugi strani pa je prav slednji model najprimernejši, če v prvi vrsti želimo zagotovljeno čim višjo točnost napovedi.

Slika 4.2 prikazuje kontingenčno matriko testiranja optimalne konfiguracije metode podpornih vektorjev. Opazimo lahko, da ima model visok delež (nad 95 %) pozitivnih identifikacij pri treh od štirih razredov; slabši rezultat dosežemo le pri določanju razreda zmajev. Večina nepravilno klasificiranih primerkov tega razreda konča v razredu padal; to se zgodi, ker sta si razreda v prostoru atributov precej blizu in se vzorci med seboj prekrivajo (kot že nakazano na sliki 3.2). Zaradi počasnosti pride tudi do prekrivanja razreda padalcev in pešcev, vendar pa je tu napačno označenih vzorcev manj.

Tabela 4.3: Najvplivnejši atributi za posamezne razrede pri uporabi univariatne logistične regresije in načinom klasifikacije en-proti-ostalim.

Razred	**Atribut**	**Uspešnost**
jadralno letalo	75. kvantil hitrosti	0,993
	25. kvantil hitrosti	0,989
	povp. hitrost planiranja	0,985
	povprečna hitrost	0,972
	standardna deviacija hitrosti	0,967
jadralno padalo	standardna deviacija spremembe višine	0,671
	povprečna hitrost pridobitve višine	0,668
	delež premočrtnega leta	0,666
	standardna deviacija hitrost	0,666
	najvišja hitrost	0,666
jadralni zmaj	povprečni čas kroženja	0,670
	najvišja hitrost v 10 minutnem oknu	0,667
	delež premočrtnega leta	0,666
	standardna deviacija hitrosti v 30 minutnem oknu	0,666
	najnižja hitrosti v 30 minutnem oknu	0,666
pešec	75. kvantil hitrosti	0,992
	standardna deviacija hitrosti	0,989
	povprečna hitrost planiranja	0,987
	povprečna hitrost	0,986
	standardna deviacija višine nad terenom	0,979

4.3 Najpomembnejši atributi

Tabela 4.3 prikazuje naučene najpomembnejše atribute, ki smo jih določili z uporabo univariatne logistične regresije. Dobljeni rezultati so v veliki primeri pričakovani.

Jadralna letala lahko za razliko od ostalih sredstev za vzdrževanje zadostnega vzgona potrebujejo precej visoke hitrosti: povprečne se gibljejo okrog 100 km/h, minimalne pa so večini primerov višje od 60 km/h, razmerje L/D pa je navadno nad 30. Oba kvantila hitrosti dobro opišeta počasne in hitre

faze leta. Hitrost planiranja pove, kako hitro se je letalo gibalo v fazi leta, ko na letalo vplivata samo sila vzgona in upora in pilot navadno želi maksimizirati razmerje med prepotovano razdaljo in izgubljeno višino. Deviacija hitrosti je prav tako pomemben faktor, saj lahko odstopanja od povprečja znašajo več kot 100 km/h.

Jadralna padala so izmed vseh letalnih naprav najpočasnejša in najmanj učinkovita v smislu pretvorbe višine v prepotovano razdaljo; povprečne hitrosti se gibljejo med 15 in 25 km/h, najvišje dovoljene pa se začnejo pri 60 km/h. Finese so nizke, okrog 10. Najdene pomembne spremenljivke, ki so značilne za padalce, so zato nekoliko nenavadne; po drugi strani pa to ni presenetljivo, saj jih želimo ločiti predvsem od jadralnih zmajev, ki so jim po hitrostnih karakteristikah precej podobni. Standardna deviacija spremembe višine ter povprečna hitrost pridobitve višine dobro opišeta, kako efektivne so letalne naprave v dviganjih; visoki odkloni padalcev pomenijo, da dosegajo precej višje vertikalne hitrosti. To je razumljivo, saj so padala počasnejša, zaradi česar lahko natančneje najdejo središča dviganj, ter lažja. Nekateri padalski leti imajo tudi majhen delež premočrtnega leta, saj se piloti samo spustijo ob grebenu do vznožja vzpetine. Med pomembnimi atributi najdemo tudi standardno deviacijo hitrosti ter najvišjo hitrost: ta je navadno nizka z majhnimi odkloni.

Tudi nabor najpomembnejših atributov pri zmajih je določen tako, da jih uspešno razlikujemo predvsem od padal. Presenetljivo je najučinkovitejša spremenljivka za določanje tega razreda povprečni čas v kroženju. Časi so tu namreč daljši kot pri ostalih skupinah, kar posledično vpliva tudi na delež premočrtnega leta. Prav tako jih dobro identificirajo tudi atributi, povezani s hitrostjo, ter njenimi odkloni v deset- oziroma tridesetminutnih oknih; hiter zmaj namreč lahko doseže hitrosti počasnega jadralnega letala, počasen pa leti približno enako hitro kot hiter padalec. Z uporabo različno dolgih obdobij merjenja zagotavljamo, da se že pokažejo značilne razlike med letalnimi napravami.

Identificiranje peščev je po drugi strani enostavno: vse hitrosti so v skladu

z zmogljivostmi človeka omejene na dobrih 10 km/h (hitrost teka), zaradi česar jih je preprosto ločiti od ostalih naprav. Tudi standardna deviacija hitrosti je tipično majhna. Ker je njihovo gibanje omejeno na zemeljsko površje, je majhen tudi standardni odklon višine nad terenom; po drugi strani pa povprečna višina nad terenom ni zanesljiv indikator, saj so tovrstni podatki na določenih področjih nenatančni in povzročijo znatna nihanja v vrednosti atributa.

4.4 Ostala opažanja

Za konec zapišimo še nekaj opažanj, ki med seboj ter z ostalimi razdelki v tem poglavju niso preveč povezana.

Začnimo s smiselnostjo nižanja števila dimenzij učne množice, ki jo preverjamo tako, da primerjamo rezultate testiranj nad nespremenjeno množico s takimi, ki jih pridobimo po izvedeni redukciji z uporabo PCA na različno število komponent. Poročamo lahko, da se pri dani množici uporaba PCA pred samim učenjem ne izplača; razlik med različnim številom parametrov namreč ni. Predvidevamo, da do tega pride, ker je število parametrov že v osnovi majhno in posledično vsi vsaj nekoliko pripomorejo k izboljšanju točnosti.

Večje razlike pa opazimo, če originalne datoteke IGC, iz katerih konstruiramo značilke, razdelimo na več skrajšanih. Na ta način za nekajkrat umetno povečamo število učnih primerov in v podatke uvedemo nekaj dodatne raznolikosti, saj tako nekatere skrajšane datoteke opisujejo samo začetno fazo leta, druge vmesno, spet tretje pa zaključno. Ideja za takšno dopolnjevanje podatkov (angl. *data augmentation*) izvira iz tehnik učenja nevronskih mrež, kjer se tehnika pogosto uporablja nad slikovnimi in zvočnimi (spektralnimi) podatki [15, 23].

Naš algoritem dopolnjevanja je preprost. Recimo, da želimo zapise IGC razdeliti na m minut dolge zapise. Uporabljen algoritem za delitev lahko zapišemo kot:

Tabela 4.4: Rezultati prečnega preverjanja pri dopolnjevanju podatkov z izrezovanjem manjših delov leta in uporabi učne metode podpornih vektorjev. V posameznem stolpcu je prikazana uspešnost in število učnih primerov pri deljenju datotek na 20-, 30- oziroma 60-minutne zapise ter primerjava z originalno množico.

	orig.	**20 min**	**30 min**	**60 min**
CA	0,968	0,974	0,977	0,978
$\mathbf{F}_1$	0,967	0,974	0,978	0,978
št. učnih primerov	632	9469	6135	2735

1. Za vsako datoteko IGC v učnih podatkih z dolžino zapisa m_i minut:
   (a) Če $m_i \leq 2m$: zapis lahko brez prekrivanja razdelimo na kvečjemu dva dela (prvi se začne točno ob pričetku, drugi pa na natanko polovici), zato datoteke ne spreminjamo in za učenje uporabimo celotni zapis.
   (b) Če $m_i > 2m$: datoteko razdelimo na $\lfloor m_i/m \rfloor$ zapisov z naključno izbranimi začetki (dopuščamo tudi prekrivanja). Vsak izmed teh zapisov je dolg (približno) m minut.
2. Konstuirane podatke uporabimo namesto originalne učne množice.

V tabeli 4.4 so prikazani rezultati takšnega dopolnjevanja pri uporabi metode podpornih vektorjev pri delitvi na 20, 30 in 60 minutne intervale. Opazimo lahko rahlo povečanje uspešnosti, ki jo po našem mnenju lahko pripišemo predvsem povečanju učne množice, ne pa tudi večji raznolikosti v podatkih. Vredno je dodati tudi, da v tem primeru večja učna množica ne pomeni tudi kvalitetnejših učnih podatkov, saj delitev na 20 minutne zapise daje slabše rezultate kot na 30 oziroma 60 minutne. Menimo, da se razlika pojavi, ker dvajset minut ni dovolj, da bi se v zapisu pokazale vse značilnost leta ter so za to potrebni daljši segmenti.

Poglavje 5

Sklepne ugotovitve

V diplomskem delu razvijemo sistem za razvrščanje zapisov letalskih letov na podlagi sledi GPS s pomočjo različnih metod strojnega učenja (z uporabo logistične regresije, metode podpornih vektorjev, naključnih gozdov, metode najbližjih sosedov ter umetnih nevronskih mrež). Poročamo lahko, da je večina modelov, zgrajenih z omenjenimi pristopi, primerna za uvrščanje zapisov v pripadajoče razrede, saj smo z njimi konsistentno dosegali uspešnost nad 95 %. Med vsemi pristopi najbolj izstopa uporaba metode podpornih vektorjev, ki je zaradi svoje hitrosti in med vsemi modeli najvišje natančnosti primerna za uporabo v primerih iz resničnega življenja. Omeniti velja tudi, da smo z velikimi (z nekaj tisoč drevesi) naključnimi gozdovi uspeli dobro aproksimirati funkcijo, ki ločuje posamezne razrede in tako dobili klasifikator, ki, ne glede na vhode, z nespremenljivo natančnostjo napoveduje ciljne razrede.

Glavna slabost vseh uporabljenih pristopov je slabo ločevanje med dvema izmed štirih razredov. Vzrok za slabo razlikovanje padalcev in zmajarjev je v tem, da sta si obe letalni napravi po tehničnih karakteristikah med seboj precej podobni. Težava je odpravljiva s konstrukcijo novih, dodatnih atributov, s katerimi bi lahko oba razreda jasneje ločili.

Primer atributov, ki jih zaradi tehničnih razlogov v diplomsko nalogo še nismo uspeli vključiti, bi bili denimo vremenski podatki in podatki o

vzletiščih in pristajališčih. O njihovem vplivu na uspešnost modelov je težko govoriti, vsekakor pa bi bilo zanimivo preveriti, če povečajo ločitveno mejo med razredi.

Težav nam ne olajša niti majhna učna množica, ki obsega le dobrih 600 učnih vzorcev. Do neke mere problem rešujemo z razdeljevanjem daljših zapisov IGC na več krajših ter s tem umetno povečujemo število učnih primerkov. Je pa ta problem rešljiv tudi z označevanjem še neznanih vzorcev. To je v našem primeru še posebej praktično, saj razpolagamo z veliko bazo podatkov, ki so sicer neoznačeni, drugih zadržkov pred njihovo uporabo pa ni. Tu se lahko uporabimo pristop aktivnega učenja (angl. *active learning*), kjer računalniški algoritem s pomočjo uporabnikovega domenskega znanja označuje primerke, ki najpozitivneje vplivajo na izboljšanje klasifikacijske uspešnosti [12].

Literatura

[1] Online Contest. Dosegljivo: `https://www.onlinecontest.org/olc-3.0/segelflugszene/index.html`. [Dostopano: 23. 3. 2018].

[2] Open Glider Network. Dosegljivo: `http://www.glidernet.com`. [Dostopano: 23. 3. 2018].

[3] SeeYou – Naviter.com. Dosegljivo: `https://www.naviter.com/products/seeyou/`. [Dostopano: 23. 3. 2018].

[4] XC Globe. Dosegljivo: `http://xcglobe.com/`. [Dostopano: 23. 3. 2018].

[5] XContest. Dosegljivo: `https://www.xcontest.org/world/en/`. [Dostopano: 23. 3. 2018].

[6] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[7] Gavin C. Cawley and Nicola L.C. Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11:2079–2107, August 2010.

[8] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, 2:265–292, March 2002.

[9] Janez Demšar, Tomaž Curk, Aleš Erjavec, Črt Gorup, Tomaž Hočevar, Mitar Milutinovič, Martin Možina, Matija Polajnar, Marko Toplak,

Anže Starič, Miha Štajdohar, Lan Umek, Lan Žagar, Jure Žbontar, Marinka Žitnik, and Blaž Zupan. Orange: Data mining toolbox in python. *Journal of Machine Learning Research*, 14:2349–2353, 2013.

[10] Janez Demšar, Gregor Leban, and Blaž Zupan. Freeviz — an intelligent multivariate visualization approach to explorative analysis of biomedical data. *Journal of Biomedical Informatics*, 40(6):661 – 671, 2007. Intelligent Data Analysis in Biomedicine.

[11] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15(1):3133–3181, January 2014.

[12] Yifan Fu, Xingquan Zhu, and Bin Li. A survey on instance selection for active learning. *Knowledge and Information Systems*, 35(2):249–283, May 2013.

[13] Fédération Aéronautique Internationale. *Technical Specification for GNSS Flight Recorders*, april 2016. Second edition.

[14] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[15] S. Hauberg, O. Freifeld, A. Boesen Lindbo Larsen, J. W. Fisher, III, and L. K. Hansen. Dreaming More Data: Class-dependent Distributions over Diffeomorphisms for Learned Data Augmentation. *ArXiv e-prints*, October 2015.

[16] John D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[17] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python. Dosegljivo: `https://scipy.org/`, 2001–. [Dostopano: 15. 5. 2018].

[18] Igor Kononenko and Matjaž Kukar. *Machine Learning and Data Mining: Introduction to Principles and Algorithms*. Horwood Publishing Limited, 2007.

[19] Wes McKinney. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[20] Travis E. Oliphant. *Guide to NumPy*. CreateSpace Independent Publishing Platform, USA, 2nd edition, 2015.

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

[22] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2nd edition, 2003.

[23] J. Salamon and J. P. Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283, March 2017.

[24] Toby Segaran. *Programming Collective Intelligence: Building Smart Web 2.0 Applications*. O'Reilly, Beijing, 2007.

[25] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, pages 45–66, 2001.

[26] M. Xue and C. Zhu. A study and application on machine learning of artificial intellligence. In *2009 International Joint Conference on Artificial Intelligence (JCAI)*, volume 00, pages 272–274, 04 2009.