UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA

DIPLOMOVÁ PRÁCE

Metody klasifikace

Katedra matematické analýzy a aplikací matematiky
Vedoucí bakalářské práce: **doc. RNDr. Karel Hron, Ph.D.**
Vypracoval: **Bc. Tadeáš Václavek**
Studijní program: N1103 Aplikovaná matematika
Studijní obor Aplikace matematiky v ekonomii
Forma studia: prezenční
Rok odevzdání: 2019

BIBLIOGRAFICKÁ IDENTIFIKACE

Autor: Bc. Tadeáš Václavek

Název práce: Metody klasifikace

Typ práce: Diplomová práce

Pracoviště: Katedra matematické analýzy a aplikací matematiky

Vedoucí práce: doc. RNDr. Karel Hron, Ph.D.

Rok obhajoby práce: 2019

Abstrakt: Klasifikace dat je jednou z metod mnohorozměrné statistické analýzy, při které je úkolem zařadit testovací data do dvou či více skupin. Tato diplomová práce se zabývá metodami, které se pro klasifikaci využívají. Cílem je čtenáři představit některé metody, jež se v současnosti nejvíce používají, a na konkrétním datovém souboru představit jejich praktickou implementaci v softwaru R.

Klíčová slova: klasifikace dat, logistická regrese, LDA, QDA, KNN, klasifikační stromy, SVM, baseball

Počet stran: 75

Počet příloh: 1

Jazyk: český

BIBLIOGRAPHICAL IDENTIFICATION

Author: Bc. Tadeáš Václavek

Title: Classification methods

Type of thesis: Masters's

Department: Department of Mathematical Analysis and Application of Mathematics

Supervisor: doc. RNDr. Karel Hron, Ph.D.

The year of presentation: 2019

Abstract: Classification of data is one of the multivariate statistical methods, when we want to categorize test observations to two or more groups. This thesis present methods, which we can use for classification. The goal is to show some classification methods which are widely used these days, apply them to the particular data set and show how to implement them in R software.

Key words: classification, logistic regression, LDA, QDA, KNN, classification trees, SVM, baseball

Number of pages: 75

Number of appendices: 1

Language: Czech

Prohlášení

Prohlašuji, že jsem diplomovou práci zpracoval samostatně pod vedením pana doc. RNDr. Karla Hrona, Ph.D. a všechny použité zdroje jsem uvedl v seznamu literatury.

V Olomouci dne ....

podpis

Poděkování

Rád bych poděkoval zejména vedoucímu diplomové práce doc. RNDr. Karlu Hronovi, Ph.D. za obětavost a čas, který mi věnoval při konzultacích. Dále bych rád poděkoval své rodině a blízkým za trpělivost, kterou se mnou měli při tvorbě této práce.

Obsah

Úvod **7**

1 Základy baseballu a představení datového souboru **8**
1.1 Baseballová pravidla 8
1.2 Rozdělení hráčů do skupin 9
1.3 Použitý datový soubor a proměnné 9

2 Základní metody klasifikace dat **11**
2.1 Logistická regrese 11
2.2 Lineární a kvadratická diskriminační analýza 14
2.2.1 Lineární diskriminační analýza 15
2.2.2 Kvadratická diskriminační analýza 19
2.3 Klasifikace podle nejbližších sousedů 21

3 Metody využívající stromy **26**
3.1 Klasifikační stromy 26
3.2 Pytlování 32
3.3 Náhodné lesy 34

4 Metoda SVM **37**
4.1 Maximální okrajový klasifikátor 37
4.2 Podpůrný vektorový klasifikátor 42
4.3 SVM 46

5 Porovnání metod klasifikace dat **56**
5.1 Výsledky binární klasifikace 56
5.2 Výsledky klasifikace do tří skupin 58

Závěr **60**

Příloha **61**

Literatura **74**

Úvod

Tématem diplomové práce je představit různé metody, s jejichž pomocí lze klasifikovat data. Klasifikací chápeme rozdělení daných pozorování do dvou či více skupin nebo tříd vykazujících určité společné vlastnosti.

Než se dostaneme k samotné klasifikaci dat a metod k ní použitých, naznačíme v první kapitole základy baseballu, které nám pomohou s lepší orientací v použitém datovém souboru. Následně si tento datový soubor představíme a popíšeme jednotlivé proměnné. Ve druhé kapitole se začneme zabývat základními metodami, kterými jsou logistická regrese, lineární a kvadratická diskriminační analýza a KNN neboli metodě k-nejbližších sousedů. V další kapitole si ukážeme, jak vytvořit klasifikační strom nebo dokonce celý les. Posledním představeným přístupem ke klasifikaci dat bude metoda SVM (z anglického *support vector machines*). V poslední kapitole potom budou jednotlivé metody porovnány zejména z hlediska úspěšnosti klasifikace.

Cílem diplomové práce je představit metody používané pro klasifikaci dat, vysvětlit je čtenáři a v neposlední řadě pak tyto metody srovnat podle toho, jak si vedly na použitých datech.

Závěrem je třeba zdůraznit, že ke každé použité metodě bude představen i příslušný kód v softwaru R, čtenář tedy bude mít možnost si všechny metody vyzkoušet a popřípadě aplikovat na své vlastní data.

Diplomová práce byla vysázena v systému LaTeX 2_ε.

Kapitola 1

Základy baseballu a představení datového souboru

V této úvodní kapitole si velmi stručně vysvětlíme baseballová pravidla a seznámíme se s rozdělením jednotlivých hráčů do skupin. Na jejím konci si ještě představíme použitý datový soubor, na kterém budeme veškeré metody předvádět. Čtenáře, který by chtěl pravidla, jednotlivé pozice, charakteristiku hráčů a baseballové statistiky více přiblížit, odkážu na první kapitolu práce [16]. Při tvorbě této kapitoly byla použita literatura [1], [14] a [16].

1.1. Baseballová pravidla

Baseball lze zařadit do kategorie pálkovacích kolektivních sportů. Velké oblibě se těší zejména v Jižní a Severní Americe nebo v Asii. Za každý tým nastupuje devět hráčů, kteří zastávají jak obrannou, tak útočnou činnost. Zápas je rozdělen na devět směn a každý tým má tedy devět pokusů na skórování. Aby mohl hráč doběhnout na domácí metu a zajistit tak svému týmu bod, je nutné nejprve odpálit míč a přitom nebýt vyautován. O to se naopak snaží bránící tým. Týmy se střídají, jakmile zahraje tým v obraně tři auty. Vyautovat hráče je možné příhozem na první metu, chycením míče ze vzduchu, tečováním mezi metami nebo strikeoutem, kdy hráč třikrát netrefí dobře nadhozený míč. Baseball se hraje na neobvyklém hřišti, které má tvar čtvrtkruhu. Zápas končí po odehrání všech devíti směn vítězstvím týmu, který získal více bodů.

1.2. Rozdělení hráčů do skupin

Jelikož úkolem klasifikace dat je zařadit pozorování do určitých skupin, rozdělíme za tímto účelem hráče do tří skupin podle jejich obranné činnosti. První skupinou budou vnitřní polaři, tzv. infieldeři, kam zařadíme druhou metu, spojku a chytače. Tito hráči by měli vykazovat slabší útočné schopnosti. Do druhé skupiny zařadíme vnější polaře neboli outfieldery, konkrétně tedy levého, středního a pravého polaře. Jelikož jejich obranná hra není tak náročná, mohou se více věnovat tréninku útočení a očekáváme tak lepší útočné statistiky. Poslední skupinou jsou hráči hrající na první a třetí metě, které můžeme označit jako corners. Tito hráči jsou většinou špičkami ve svých týmech, co se útoku týče a jejich útočná hra by měla být výrazněji lepší, než u předešlých skupin.

1.3. Použitý datový soubor a proměnné

Data, která budeme používat při klasifikaci, pocházejí ze slavné MLB (*Major League Baseball*) [1]. Jedná se o baseballové statistiky 240 hráčů s nejvyšším počtem odehraných zápasů v letech 2015 a 2016. Celkem tedy máme 480 pozorování, která jsou popsána deseti proměnnými. Jmenovitě pak již zmíněným rokem (year), počtem odehraných zápasů (games), počtem startů na pálce (AB), dvojmetovými odpaly (2B), homeruny (HR), úspěšnými odpaly (hits), staženými body (RBI), úspěšností na pálce (AVG) a úspěšností získání mety (OBP). Poslední, ale zřejmě nejvíce důležitou proměnnou, je pozice (pos), na které hráč nastupuje. Tato proměnná je jako jediná kvalitativní a není tedy reprezentována číselnou hodnotou, ale slovním popisem. Nabývá tří hodnot podle skupin popsaných v předešlé podkapitole. Jelikož baseball není v našich krajinách příliš rozšířeným sportem, popíšeme si jednotlivé proměnné podrobněji, aby čtenář lépe pochopil použitý datový soubor.

- **games** - počet odehraných zápasů za jednu sezónu.
- **AB** - počet startů na pálce.

- **2B** - počet dvojmetových odpalů. Při takovémto odpalu hráč doběhne rovnou až na druhou metu. Odpal tedy musí být dostatečně dlouhý, aby hráč na druhou metu stihnul doběhnout před příhozem protihráče.
- **HR** - počet homerunů. Homerun zaznamená hráč, který odpálí míč až za hrazení baseballového hřiště a může tedy oběhnout všechny mety, čímž získá pro svůj tým bod.
- **hits** - počet úspěšných odpalů. Úspěšný odpal je takový, při kterém se hráč dostane na první, druhou, třetí nebo rovnou zpět na domácí metu.
- **RBI** - počet stažených bodů. Stažený bod může hráč zaznamenat tak, že po jeho odpalu spoluhráč, který již stojí na některé z met, doběhne na metu domácí.
- **AVG** - průměrná úspěšnost odpalu. Vypočítá se jako podíl úspěšných odpalů ku počtu startů na pálce.
- **OBP** - úspěšnost dosáhnutí mety. Dosáhnout některé z met může hráč například po dobrém odpalu, metě zdarma nebo po chybě týmu v obraně.

Kapitola 2

Základní metody klasifikace dat

Ve druhé kapitole si představíme základní metody využívané pro klasifikaci dat. Začneme s logistickou regresí, která je naprosto základním nástrojem klasifikace. Poté se zaměříme na lineární diskriminační analýzu a její kvadratickou modifikaci, na závěr této kapitoly si řekneme něco o metodě k-nejbližších sousedů. Tato kapitola byla vypracována s pomocí literatury [3], [4], [5], [7], [8], [9], [11], [12], [15] a [17].

2.1. Logistická regrese

Logistická regrese, jako nástroj pro odhad pravděpodobnosti výskytu nějakého jevu, je čtenáři zcela jistě známa a přistoupíme tedy rovnou k jejímu využití při klasifikaci dat. Pro lepší pochopení si vše vysvětlíme přímo na našem datovém souboru. Řekněme, že na základě již dříve zmíněných baseballových statistik jako HR, 2B atd. chceme každého hráče zařadit do jedné ze tří skupin. Připomeňme si, že tyto skupiny jsou vnitřní polaři, vnější polaři a corners. Vysvětlovaná proměnná tedy bude kvalitativní a vysvětlující proměnné naopak kvantitativní. Jelikož se logistická regrese používá zejména pro binární klasifikaci [4], použijeme podsoubor, který obsahuje pouze vnitřní polaře a corners. Zařazovat tedy budeme pouze do dvou tříd. Nyní se přeci jen na chvíli vraťme k podstatě logistické regrese. Můžeme říci, že modeluje pravděpodobnost, se kterou určité pozorování (v našem případě hráč), patří do nějaké třídy. Například pravděpodobnost toho,

že hráč bude patřit do skupiny cornes na základě počtu HR, se dá zapsat následovně:

$$P(pos = corners|HR).$$

Nyní je nutno zvolit si nějaký práh, řekněme 0,5 a na základě toho, zda pravděpodobnost $P(pos = corners|HR)$ překročila tento práh, můžeme každé pozorování do třídy zařadit či nikoliv. Pokud vysvětlovanou proměnnou označíme jako Y, pak výskytu jevu (pos=corners) přiřadíme hodnotu 1, jinak bude Y nabývat nulové hodnoty.

Přirozeně nás dále bude zajímat, jak takovou pravděpodobnost pro vektor prediktorů $\boldsymbol{X} = (X_1, \ldots, X_p)^T$, označme ji například $p(\boldsymbol{X})$, spočítat. V logistické regresi používáme funkci, nazývající se logistická funkce a vypadající následovně:

$$p(\boldsymbol{X}) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1 + e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}.$$

Nyní tedy víme, jak pravděpodobnost $p(\boldsymbol{X})$ vypočítat, co ale neznáme, jsou koeficienty $\beta_0, \beta_1, \ldots, \beta_p$, které je třeba z dat odhadnout. K tomu využijeme metodu maximální věrohodnosti. Jelikož se tato práce zaměřuje na popsání metod klasifikace, dovolíme si zde tuto metodu pouze krátce představit v kontextu odhadu parametrů logistické regrese. Zájemce o tuto problematiku odkazujeme na [9]. Princip spočívá ve vytvoření věrohodnostní funkce pro n pozorování y_i
a $x_i = (x_{i1}, \ldots, x_{ip})^T, i = 1, \ldots, n$, jejíž předpis vypadá následovně:

$$\ell(\beta_0, \beta_1, \ldots, \beta_p) = \prod_{i:y_i=1} p(\boldsymbol{x_i}) \prod_{i':y_{i'}=0} (1 - p(\boldsymbol{x_{i'}})).$$

Dále by se tato funkce vhodně upravila (například logaritmováním) a hledaly by se takové hodnoty parametrů $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$, pro které by byla hodnota věrohodnostní funkce maximální.

V další části této podkapitoly si ukážeme, jak logistickou regresi aplikovat na reálná data v softwaru R. Na začátek je však třeba zmínit několik důležitých skutečností. Jako datový soubor zde budeme používat ten, který obsahuje pouze

hráče klasifikované jako vnitřní polaři a corners, protože, jak již bylo řečeno, logistická regrese se využívá hlavně při binární klasifikaci. Soubor s názvem *Baseballcorinf* tedy bude obsahovat pouze 300 hráčů, z nichž je 180 vnitřních polařů a 120 corners. Při všech zde zmíněných metodách byla vyzkoušena původní i normovaná data, u logistické regrese to však na výsledek nemělo vliv. Datový soubor je rozdělen na trénovací a testovací část, s tím, že na trénovacím souboru je vždy metoda natrénována a následně s použitím testovacího souboru otestována její přesnost. Posledním velmi důležitým bodem, který je zde nutné zmínit, je nekonzistence výsledků v důsledku různého rozdělení datového souboru. Rozdělíme-li data vždy náhodně, je velmi pravděpodobné, že naše výsledky se budou značně lišit. Z tohoto důvodu je u každé metody ukázán vždy pouze jeden z výsledků a v poslední kapitole, ve které metody porovnáváme je tato nekonzistence vyřešena opakováním metod a následným zprůměrováním výsledků. Pojďme se tedy podívat přímo na logistickou regresi. Nejprve je nutné si do softwaru nainstalovat knihovnu `ISLR`. Dále rozdělíme náš datový soubor na trénovací a testovací data pomocí funkce *sample.split*, která je implementovaná v knihovně `caTools`, při našem nastavení je 70% pozorování zařazeno jako trénovacích a zbylých 30% jako testovacích. Následně již pomocí funkce *glm* s parametrem *family = binomial* provedeme logistickou regresi. Pomocí příkazu *summary$coef* si také můžeme vypsat hodnoty odhadnutých parametrů $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$ společně s dalšími charakteristikami. Vše je podrobně vidět v následujícím kódu.

```
> library(ISLR)
> library(caTools)
> set.seed(123)
> train = sample.split(Baseballcorinf$pos, SplitRatio = 0.7)
> test=Baseballcorinf[,2:10][!train,]
> pos.test=Baseballcorinf$pos[!train]
> glm.fit=glm(pos~year+games+at.bats+doubles+hr+hits+rbi+avg+obp,
+             data=Baseballcorinf,family=binomial, subset=train)
> summary(glm.fit)$coef
                Estimate   Std. Error    z value     Pr(>|z|)
(Intercept) -5.164089e+02 725.07546382 -0.7122140 4.763323e-01
year         2.627848e-01   0.36146660  0.7269961 4.672284e-01
games       -7.758862e-02   0.01985823 -3.9071265 9.340025e-05
at.bats      1.466283e-02   0.01375730  1.0658220 2.865041e-01
doubles     -5.270146e-02   0.03490472 -1.5098664 1.310775e-01
hr          -1.311541e-01   0.04011516 -3.2694386 1.077611e-03
hits         7.829565e-03   0.05168373  0.1514899 8.795893e-01
rbi          5.641498e-03   0.02041191  0.2763827 7.822541e-01
avg          1.228791e+01  25.25387708  0.4865750 6.265595e-01
obp         -2.346032e+01   9.57114709 -2.4511506 1.424004e-02
```

Jako další nás zajímá, jak si tato metoda vedla na našich testovacích datech. Jako práh zvolíme hodnotu 0,5 a níže je opět vidět příslušný kód s výsledkem klasifikace.

```
> glm.probs=predict(glm.fit,test, type="response")
> glm.pred=rep("cor",length(pos.test))
> glm.pred[glm.probs>.5]="inf"
> table(glm.pred,pos.test)
        pos.test
glm.pred COR INF
     cor  21  14
     inf  15  40
> (21+40)/90
[1] 0.6777778
```

Závěrem této podkapitoly můžeme říci, že logistická regrese správně klasifikovala přes 67% testovacích pozorování.

2.2. Lineární a kvadratická diskriminační analýza

Jako další metodu klasifikace si představíme lineární diskriminační analýzu (*linear discriminant analysis*) a kvadratickou diskriminační analýzu (*quadratic discriminant analysis*). Začněme ale nejdříve krátkou motivací. Stejně tak jako u logistické regrese předpokládejme, že chceme nějaké pozorování zařadit do jedné z K tříd, kde nyní je $K \geq 2$. Zavedeme tedy apriorní pravděpodobnost $\pi_k, k = 1, \ldots, K$, která nám udává pravděpodobnost, že náhodně zvolené pozorování pochází z k-té skupiny. Dále nechť $f_k(\boldsymbol{x})$ značí hustotu $\boldsymbol{x}$ pro p-rozměrné

pozorování z k-té skupiny. Bayesova věta se pak používá k získání aposteriorní pravděpodobnosti $P(k|\boldsymbol{x})$ toho, že pozorování $\boldsymbol{x}$ patří do k-té skupiny, což můžeme zapsat následovně

$$P(k|\boldsymbol{x}) = \frac{\pi_k f_k(\boldsymbol{x})}{\sum_{l=1}^{K} \pi_l f_l(\boldsymbol{x})}.$$

Nyní již budeme používat zkrácený zápis $p_k(\boldsymbol{x}) = P(k|\boldsymbol{x})$. Připomeňme si, že v případě logistické regrese popsané v předchozí podkapitole jsme $p_k(\boldsymbol{x})$ počítali přímo. Nyní však můžeme pouze odhadnout π_k a $f_k(\boldsymbol{x})$ a dosadit je do předchozího vztahu. Odhad apriorní pravděpodobnosti π_k je velmi snadný. Spočítáme ho jako podíl trénovacích pozorování, která patří do k-té třídy ku celkovému počtu trénovacích pozorování. Na druhou stranu odhady hustot $f_k(\boldsymbol{x})$ se jeví jako podstatně náročnější úkol, pakliže nepředpokládáme tyto hustoty v nějaké známé rodiny rozdělení.

2.2.1. Lineární diskriminační analýza

Jak již bylo řečeno, stěžejním krokem bude odhadnout $f_k(\boldsymbol{x})$ a následně tento odhad dosadit do $p_k(\boldsymbol{x})$. Pozorování následně budeme klasifikovat do takové třídy, pro kterou bude $p_k(\boldsymbol{x})$ největší. Předpokládejme nyní, že vektor prediktorů $\boldsymbol{X} = (X_1, X_2, \ldots, X_p)^T$ pochází z vícerozměrného normálního rozdělení, což můžeme zapsat jako $\boldsymbol{X} \sim N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, kde $\boldsymbol{\mu}$ je vektor středních hodnot a $\boldsymbol{\Sigma}$ je varianční matice. Hustota vícerozměrného normálního rozdělení je ve tvaru

$$f(\boldsymbol{x}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right).$$

Lineární diskriminační pravidlo, které si vzápětí zadefinujeme, předpokládá, že pozorování z k-té třídy pocházejí z vícerozměrného normálního rozdělení se střední hodnotou $\boldsymbol{\mu}_k$ a varianční maticí $\boldsymbol{\Sigma}$, která je stejná pro všech K tříd. Dosadíme-li hustotu pro k-tou třídu $f_k(\boldsymbol{x})$ do vztahu pro výpočet $p_k(\boldsymbol{x})$ a upravíme, dostaneme již zmíněné lineární diskriminační pravidlo, které vypadá následovně

$$\delta_k(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \ln \pi_k.$$

Každé pozorování pak zařadíme do třídy, pro kterou bude mít lineární diskriminační pravidlo největší hodnotu. Dále nás budou zajímat odhady parametrů $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \pi_1, \ldots, \pi_K$ a $\boldsymbol{\Sigma}$. Za odhady $\boldsymbol{\mu}_k$ a $\boldsymbol{\Sigma}$ budeme považovat výběrový průměr, respektive výběrovou společnou varianční matici, které dostaneme jako

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{n_k} \sum_{i=1}^{n_k} \boldsymbol{x}_{ki},$$

a z výběrových variančních matic v jednotlivých skupinách

$$\hat{\boldsymbol{\Sigma}_k} = \frac{1}{n_k - 1} \sum_{i=1}^{n_k} (\boldsymbol{x}_{ki} - \hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_{ki} - \hat{\boldsymbol{\mu}}_k)^T,$$

následně

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{n - K} \sum_{k=1}^{K} (n_k - 1)\hat{\boldsymbol{\Sigma}_k},$$

za odhad π_k považujeme, jak bylo již zmíněno dříve,

$$\hat{\pi}_k = n_k / n$$

kde n je celkový počet pozorování a n_k je počet pozorování v k-té třídě. Nyní, když máme zkonstruované odhady, můžeme je dosadit do lineárního diskriminačního pravidla a provést klasifikaci. Abychom mohli klasifikovat pomocí LDA, je potřeba nainstalovat do softwaru R knihovnu `MASS`. Data si rozdělíme na trénovací a testovací část stejným způsobem, jako tomu bylo u logistické regrese a tuto část kódu zde tedy vynecháme. Nejprve si ukážeme binární klasifikaci a použijeme opět soubor, který obsahuje pouze vnitřní polaře a corners. Příkaz *lda* provede lineární diskriminační analýzu a její shrnutí můžeme vidět níže.

```
> library(MASS)
> lda.fit=lda(pos~games+at.bats+doubles+hr+hits+rbi+avg+obp,
+             data=Baseballcorinf,subset = train)
> lda.fit
Call:
lda(pos ~ games + at.bats + doubles + hr + hits + rbi + avg +
    obp, data = Baseballcorinf, subset = train)

Prior probabilities of groups:
COR INF
0.4 0.6

Group means:
       games  at.bats  doubles       hr     hits      rbi       avg       obp
COR 140.3929 486.8571 26.53571 21.78571 129.3095 71.15476 0.2627619 0.3346071
INF 129.5635 454.8730 23.42063 12.49206 121.4524 54.14286 0.2630317 0.3194841

Coefficients of linear discriminants:
                LD1
games   -0.063612843
at.bats  0.008529119
doubles -0.027417322
hr      -0.081850192
hits     0.025812518
rbi     -0.012458938
avg     -8.317849230
obp     -7.897236411
```

Lze si všimnout, že z výstupu metody LDA můžeme vyčíst například apriorní pravděpodobnosti π_k, průměry jednotlivých prediktorů v každé třídě, které metoda využívá jako odhady $\boldsymbol{\mu}_k$ nebo také koeficienty lineárního diskriminačního pravidla. Nyní se podívejme, jak si metoda vedla na testovacích datech.

```
> lda.pred=predict(lda.fit,test)
> lda.class=lda.pred$class
> table(lda.class,pos.test)
         pos.test
lda.class COR INF
      COR  25  12
      INF  11  42
> (25+42)/90
[1] 0.7444444
```

Lineární diskriminační analýza správně klasifikovala více než 74% testovacích pozorování, což můžeme považovat za dobrý výsledek. Jednou z hlavních výhod lineární diskriminační analýzy oproti logistické regresi je skutečnost, že LDA lze použít i na klasifikaci do více než dvou skupin (což je sice možné i pomocí logistické regrese, nicméně se pro tuto úlohu nepoužívá).

Ukážeme si zde tedy i klasifikaci do všech tří skupin, které náš datový soubor obsahuje. Zadání do softwaru je obdobné, jako tomu bylo u binární klasifikace,

s rozdílem použitého datového souboru a proto si zde již okomentujeme pouze výsledek následujícího kódu.

```
> lda.fit=lda(pos~games+at.bats+doubles+hr+hits+rbi+avg+obp,
+             data=Baseball,subset = train)
> lda.fit
Call:
lda(pos ~ games + at.bats + doubles + hr + hits + rbi + avg +
    obp, data = Baseball, subset = train)

Prior probabilities of groups:
  COR   INF    OF
0.250 0.375 0.375

Group means:
       games  at.bats  doubles       hr     hits      rbi       avg       obp
COR 140.3929 486.8571 26.53571 21.78571 129.3095 71.15476 0.2627619 0.3346071
INF 129.5635 454.8730 23.42063 12.49206 121.4524 54.14286 0.2630317 0.3194841
OF  134.6508 453.0079 23.27778 15.29365 120.2857 56.47619 0.2615000 0.3281508



Coefficients of linear discriminants:
                LD1          LD2
games    -0.059896758  0.058144009
at.bats   0.003356759 -0.007247554
doubles  -0.019335875 -0.043760962
hr       -0.088001172  0.034673389
hits      0.034364671  0.014957016
rbi      -0.001480812 -0.057011586
avg      -8.628054897 -3.939864587
obp     -10.792639904 13.111834738

Proportion of trace:
   LD1    LD2
0.8719 0.1281



> lda.pred=predict(lda.fit,test)
> lda.class=lda.pred$class
> table(lda.class,pos.test)
         pos.test
lda.class COR INF OF
      COR  18   7 15
      INF   8  30 22
      OF   10  17 17
> (18+30+17)/144
[1] 0.4513889
```

Po zhlédnutí výsledku můžeme říci, že metoda LDA správně klasifikovala jen lehce přes 45% pozorování. Závěrem je nutno říci, že metoda byla použita i na normovaná data, ale stejně tak, jako tomu bylo u logistické regrese, to na výsledcích klasifikace nic nezměnilo.

2.2.2. Kvadratická diskriminační analýza

Lineární diskriminační analýza předpokládala, že pozorování v každé třídě pochází z vícerozměrného normálního rozdělení s odlišnou střední hodnotou pro každou třídu a společnou varianční maticí. Kvadratická diskriminační analýza místo společné varianční matice předpokládá, že každá třída má svou vlastní varianční matici a můžeme tedy psát, že $\boldsymbol{X} \sim N_p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, kde $\boldsymbol{\mu}_k$ je střední hodnota a $\boldsymbol{\Sigma}_k$ je varianční matice pro k-tou třídu. Kvadratické diskriminační pravidlo pak definujeme následovně

$$\delta_k(\boldsymbol{x}) = -\frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}_k^{-1}\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{\Sigma}_k^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}_k^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\ln|\boldsymbol{\Sigma}_k| + \ln\pi_k.$$

Kvadratické diskriminační pravidlo každé pozorování zařadí do takové třídy, pro kterou je hodnota tohoto pravidla největší. Proč nás vlastně zajímá, zda mají jednotlivé třídy společnou či rozdílnou varianční matici? Respektive, proč preferovat QDA před LDA či naopak? Řekněme, že máme p prediktorů a pro odhad varianční matice tedy bude potřeba odhadnout $p(p+1)/2$ parametrů. Jestliže bychom ale uvažovali metodu QDA, pak bychom museli odhadnou $Kp(p+1)/2$ parametrů, což je při větším počtu prediktorů zásadní nevýhoda metody QDA. V praxi se dle [4] pro menší rozsahy trénovací množiny používá spíše LDA. Naopak pro dostatečně velký rozsah trénovací množiny, popřípadě je-li zcela evidentní, že varianční matice jsou pro každou skupiny odlišné, je vhodné použít QDA. Nakonec této podkapitoly si ještě uvedeme praktickou ukázku použití metody QDA v softwaru R. Použijeme stejné rozdělení dat na trénovací a testovací množinu, jako tomu bylo u LDA a opět ukážeme klasifikaci jak do dvou, tak do tří tříd. Funkce *qda* pro zavolání metody se nachází ve stejném balíčku jako LDA. Začněme nejdříve s binární klasifikací.

```
> qda.fit=qda(pos~games+at.bats+doubles+hr+hits+rbi+avg+obp,
+             data=Baseballcorinf,subset = train)
> qda.fit
Call:
qda(pos ~ games + at.bats + doubles + hr + hits + rbi + avg +
    obp, data = Baseballcorinf, subset = train)

Prior probabilities of groups:
COR INF
0.4 0.6

Group means:
       games  at.bats  doubles       hr     hits      rbi       avg       obp
COR 140.3929 486.8571 26.53571 21.78571 129.3095 71.15476 0.2627619 0.3346071
INF 129.5635 454.8730 23.42063 12.49206 121.4524 54.14286 0.2630317 0.3194841


> qda.class=predict(qda.fit,test)$class
> table(qda.class,pos.test)
         pos.test
qda.class COR INF
      COR  24  15
      INF  12  39
> (24+39)/90
[1] 0.7
```

Vidíme, že funkce *qda* nám stejně tak jako *lda* vypsala apriorní pravděpodobnosti a skupinové průměry. Dále můžeme pozorovat, že QDA si na našich datech vedla dobře, jelikož správně klasifikovala 70% testovacích pozorování. Dále si obdobně jako u LDA ukážeme ještě klasifikaci do tří skupin.

```
> qda.fit=qda(pos~games+at.bats+doubles+hr+hits+rbi+avg+obp,
+             data=Baseball,subset = train)
> qda.fit
Call:
qda(pos ~ games + at.bats + doubles + hr + hits + rbi + avg +
    obp, data = Baseball, subset = train)

Prior probabilities of groups:
  COR   INF    OF
0.250 0.375 0.375

Group means:
       games  at.bats  doubles       hr     hits      rbi       avg       obp
COR 140.3929 486.8571 26.53571 21.78571 129.3095 71.15476 0.2627619 0.3346071
INF 129.5635 454.8730 23.42063 12.49206 121.4524 54.14286 0.2630317 0.3194841
OF  134.6508 453.0079 23.27778 15.29365 120.2857 56.47619 0.2615000 0.3281508
```

```
> qda.class=predict(qda.fit,test)$class
> table(qda.class,pos.test)
         pos.test
qda.class COR INF OF
      COR  16  10 18
      INF  11  31 20
      OF    9  13 16
> (16+31+16)/144
[1] 0.4375
```

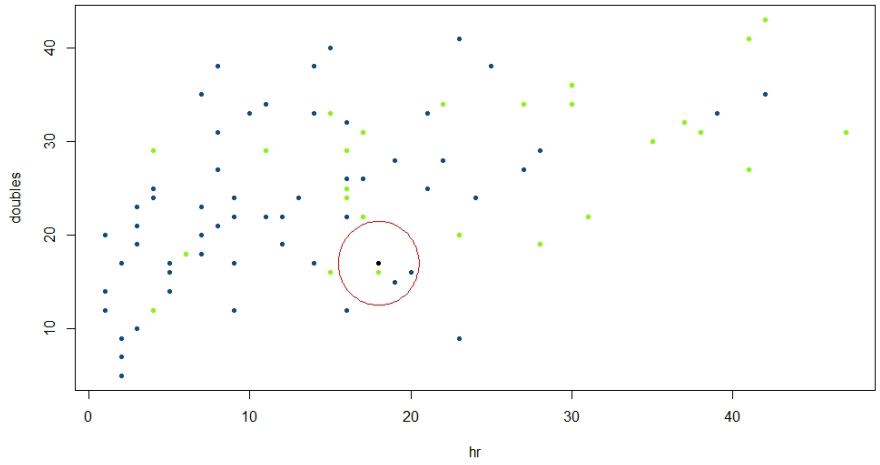
Pro tři skupiny metoda obdobně, jako tomu bylo u LDA, nedopadla nejlépe a klasifikovala správně jen přes 43% pozorování. Opět bylo vyzkoušeno normování a též v tomto případě nemělo na výsledek klasifikace vliv. Pojďme si ještě na závěr srovnat výsledky metod LDA a QDA. Při binární klasifikaci si lépe vedla metoda LDA, která správně klasifikovala přes 74% testovacích pozorování, naproti tomu metoda QDA správně klasifikovala přesně 70% testovacích pozorování. Stejně tomu bylo i u klasifikace do tří tříd, kde LDA správně zařadila přes 45%, oproti téměř 44% u QDA.

2.3. Klasifikace podle nejbližších sousedů

Jako poslední ze základních přístupů ke klasifikaci si představíme metodu klasifikace podle nejbližších sousedů. Tato metoda je velmi intuitivní a v kontextu klasifikace často používaná. Zařadit bychom ji mohli do kategorie učení s učitelem. Dále budeme v textu využívat zkratku KNN (z anglického *K-Nearest Neighbours*). Nyní si představíme základní princip této metody. Nejprve data rozdělíme na trénovací a testovací množinu. Následně pozorování z testovací množiny dosadíme do prostoru trénovací množiny a hledáme k nejbližších sousedů, tedy takových trénovacích pozorování, která mají k našemu testovacímu pozorování nejmenší vzdálenost. Následně toto pozorování zařadíme do takové třídy, ze které pochází většina k sousedů. Jako vzdálenost se obvykle používá euklidovská vzdálenost. Připomeňme si, že euklidovská vzdálenost pozorování $\boldsymbol{x} = (x_1, \ldots, x_p)^T$ a $\boldsymbol{y} = (y_1, \ldots, y_p)^T$ je definovaná jako

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2}.$$

Metoda je tedy opravdu velmi intuitivní a snad jediným úskalím je volba parametru k. Nejprve je třeba si uvědomit, že při volbě $k = 1$ budeme testovací pozorování klasifikovat vždy do takové třídy, do které patří nejbližší trénovací pozorování. Tento přístup by však mohl vést k přílišnému přeučení a naopak vysoké hodnoty k povedou k vyšší výpočetní náročnosti. Dalším doporučením při volbě parametru k u binární klasifikace je používat lichá čísla, čímž se vyhneme případu, kdy například ze čtyř nejbližších sousedů budou dva z první skupiny a dva ze druhé. Nyní si ještě celý algoritmus stručně popíšeme pomocí následujícího obrázku.



Obrázek 2.1: Ilustrace metody KNN

Na obrázku jsou vidět reálná pozorování z našeho datového souboru. Trénovací množina obsahuje 30 pozorování zelené barvy, která patří skupině corners a 60 modrých pozorování, která patří vnitřním polařům. Černou barvou je označeno testovací pozorování, které chceme zařadit. Jak si lze všimnout, klasifikovat budeme podle tří nejbližších sousedů, tedy $k = 3$. Testovací pozorování bude zřejmě klasifikováno do třídy vnitřních polařů, jelikož dva ze tří nejbližších sousedů patří právě do druhé skupiny.

Dále si ukážeme metodu KNN v softwaru R. Nejprve budeme potřebovat knihovny `class` a `caTools`. Začneme s binární klasifikací a budeme potřebovat soubor *Baseballcorinf*, obsahující pouze vnitřní polaře a corners. Ze všeho nejdříve

je nutné rozdělit soubor na trénovací a testovací množinu, metodu pak provedeme pomocí příkazu *knn*.

```
> library(class)
> library(caTools)
> set.seed(1)
> train = sample.split(Baseballcorinf$pos, SplitRatio = 0.7)
> train.x=Baseballcorinf[,2:10][train,]
> test.x=Baseballcorinf[,2:10][!train,]
> pos.train=Baseballcorinf$pos[train]
> pos.test=Baseballcorinf$pos[!train]
> knn.pred=knn(train.x,test.x,pos.train,k=3)
> table(knn.pred,pos.test)
        pos.test
knn.pred COR INF
     COR  21  10
     INF  15  44
> (21+44)/90
[1] 0.7222222
> knn.pred=knn(train.x,test.x,pos.train,k=5)
> table(knn.pred,pos.test)
        pos.test
knn.pred COR INF
     COR  20   7
     INF  16  47
> (20+47)/90
[1] 0.7444444
```

Nejprve jsme vyzkoušeli $k = 3$ a následně $k = 5$. Jak vidíme, tak obě volby parametru dopadly velmi podobně. Konkrétně pro $k = 3$ bylo správně klasifikováno přes 72% a pro $k = 5$ přes 74% testovacích pozorování. U logistické regrese, LDA a QDA normování dat nemělo význam. Dále se podíváme, jestli bude mít normování vliv na metodu KNN. Pomocí příkazu *scale* data znormujeme a dále bude celý postup stejný jako výše.

```
> set.seed(1)
> standardized.x=scale(Baseballcorinf[,3:10])
> train.x1=standardized.x[train,]
> test.x1=standardized.x[!train,]
> pos.train=Baseballcorinf$pos[train]
> knn.pred=knn(train.x1,test.x1,pos.train,k=3)
> table(knn.pred,pos.test)
        pos.test
knn.pred COR INF
     COR  19  17
     INF  17  37
> (19+37)/90
[1] 0.6222222
```

```
> knn.pred=knn(train.x1,test.x1,pos.train,k=5)
> table(knn.pred,pos.test)
        pos.test
knn.pred COR INF
     COR  20  16
     INF  16  38
> (20+38)/90
[1] 0.6444444
```

Vidíme, že normování mělo na výsledek klasifikace značný vliv a je nutné vzít tuto skutečnost v potaz. Nakonec vyzkoušíme klasifikaci na celém našem souboru, budeme tedy klasifikovat do tří skupin. Jak soubor rozdělit na testovací, respektive trénovací data a jak data normovat, jsme si již ukázali, a tudíž to zde nebudeme opět uvádět.

```
> knn.pred=knn(train.x,test.x,pos.train,k=3)
> table(knn.pred,pos.test)
        pos.test
knn.pred COR INF OF
     COR  13   8 19
     INF   6  26 18
     OF   17  20 17
> (13+26+17)/144
[1] 0.3888889
> knn.pred=knn(train.x,test.x,pos.train,k=5)
> table(knn.pred,pos.test)
        pos.test
knn.pred COR INF OF
     COR  12   8 11
     INF   8  23 27
     OF   16  23 16
> (12+23+16)/144
[1] 0.3541667
> knn.pred=knn(train.x,test.x,pos.train,k=7)
> table(knn.pred,pos.test)
        pos.test
knn.pred COR INF OF
     COR  15   5 12
     INF   9  25 24
     OF   12  24 18
> (15+25+18)/144
[1] 0.4027778
```

Nyní jsme zkusili klasifikovat postupně dle třech, pěti a sedmi nejbližších sousedů. Vidíme, že výsledky zde nedosahují příliš uspokojivých hodnot, nicméně můžeme konstatovat, že nejlépe dopadla klasifikace při použití sedmi nejbližších sousedů. Nakonec opět zkusíme ta samá nastavení parametru k pro normovaná data.

```
> knn.pred=knn(train.x1,test.x1,pos.train,k=3)
> table(knn.pred,pos.test)
        pos.test
knn.pred COR INF OF
     COR  15   8 15
     INF  11  30 23
     OF   10  16 16
> (15+30+16)/144
[1] 0.4236111
> knn.pred=knn(train.x1,test.x1,pos.train,k=5)
> table(knn.pred,pos.test)
        pos.test
knn.pred COR INF OF
     COR  14  11 13
     INF  10  26 25
     OF   12  17 16
> (14+26+16)/144
[1] 0.3888889
> knn.pred=knn(train.x1,test.x1,pos.train,k=7)
> table(knn.pred,pos.test)
        pos.test
knn.pred COR INF OF
     COR  14   9  8
     INF  10  27 27
     OF   12  18 19
> (14+27+19)/144
[1] 0.4166667
```

Zde se zcela překvapivě jeví jako nejlepší použití tří nejbližších sousedů. Dále můžeme konstatovat, že oproti binární klasifikaci, kde normování dat vedlo ke zhoršení výsledků metody, nyní normování naopak výsledky většinově vylepšilo.

Kapitola 3

Metody využívající stromy

V této kapitole si ukážeme, jak pomocí metod využívajících rozhodovací stromy klasifikovat data. Nejprve se budeme zabývat jednoduchými klasifikačními stromy, které následně vylepšíme pomocí metody zvané pytlování, popřípadě použitím náhodných lesů. Všechny tyto metody, ačkoliv většinou nedosahují takových výsledků jako složitější metody, jsou velmi názorné a jednoduché na interpretaci. V této kapitole byla použita literatura [2], [4], [6], [7] a [13].

3.1. Klasifikační stromy

Klasifikační stromy jsou speciálním případem tzv. rozhodovacích stromů. Nejprve si povíme, jak takový klasifikační strom vytvořit. Proces výstavby klasifikačního stromu se dá popsat pomocí dvou kroků. V prvním kroku rozdělíme prostor nezávislých proměnných (prediktorů) $X_1, X_2, \cdots, X_p$ do J odlišných a vzájemně se nepřekrývajících regionů $R_1, R_2, \cdots, R_J$. Ve druhém kroku už pouze každé pozorování, které padlo do regionu R_j, zařadíme do takové třídy, která byla v trénovacích datech v tomto regionu nejčastěji zastoupena. Nyní si první krok probereme podrobněji. Čtenáře určitě napadne otázka, jak vytvořit regiony $R_1, R_2, \cdots, R_J$? Tento proces se nazývá rekurzivní binární dělení. Nejprve vezmeme v úvahu všechny možné prediktory $X_1, X_2, \cdots, X_p$ a jeden z nich, řekněme X_i, zvolíme a dále pomocí nějakého bodu řezu s rozdělíme prostor X_i na dva regiony $\{X|X_i < s\}$ a $\{X|X_i \geq s\}$ tak, aby klasifikační chyba byla co nejmenší.

Klasifikační chyba je podíl trénovacích pozorování, která nepatří do nejčastější třídy a vypadá následovně:

$$E = 1 - \max_k(\hat{p}_{jk}),$$

kde $\hat{p}_{jk}$ je poměr trénovacích pozorování v j-tém regionu pocházejících z k-té třídy. Místo klasifikační chyby je dále možné použít například Giniho index, definovaný jako:

$$G = \sum_{k=1}^{K} \hat{p}_{jk}(1 - \hat{p}_{jk}),$$

nebo míru neuspořádanosti (z anglického *cross-entropy*):

$$D = -\sum_{k=1}^{K} \hat{p}_{jk} \log \hat{p}_{jk}.$$

Dále celý proces opakujeme a hledáme opět nejlepší prediktor a nejlepší bod řezu v rámci každého z výsledných regionů. Nyní ale rozdělíme pouze jeden ze dvou již vytvořených regionů. Tímto způsobem vytváříme regiony, dokud nedosáhneme zastavovacího kritéria, například dokud každý z regionů neobsahuje méně než deset pozorování.

Takto vytvořený klasifikační strom by však mohl být příliš hustý, což by mohlo vést k přílišnému přeučení a výsledný strom by pak dosahoval špatných výsledků na testovacím souboru. Menší strom tvořený menším počtem regionů bude navíc snadnější na intepretaci. Jedním z možných způsobů, jak vytvořit menší strom, je nastavit práh pro jedno z použitých klasifikačních kritérií tak, že když dojde k pouze malému zmenšení tohoto kritéria, proces výstavby stromu se zastaví. Je ovšem možné, že v dalším kroce po zastavení výstavby by nám klasifikační kritérium mohlo výrazně klesnout a přišli bychom tím o důležité zlepšení modelu. Daleko lepším přístupem bude, když nejprve vytvoříme velký strom s velkým počtem regionů, který označíme T_0 a následně ho budeme zpětně prořezávat, čímž dostane nový podstrom. Jedním ze způsobů, jak takovéto prořezávání provést, je použít metodu *cost complexity prunning*. Nejprve budeme uvažovat posloupnost

podstromů indexovanou nezáporným ladícím parametrem α. Pro každou hodnotu parametru α dostaneme podstrom $T \subset T_0$ takový, že klasifikační chyba

$$E = 1 - \max_k(\hat{p}_{jk}) + \alpha|T|$$

bude co nejmenší, kde $|T|$ je počet konečných větví klasifikačního stromu. Parametr α kontroluje kompromis mezi složitostí stromu a výslednou klasifikační chybou. Je třeba zdůraznit, že pokud nastavíme parametr α roven nule, strom zůstane neprořezán. Se zvětšujícím se α tedy bude strom více a více řídnout. Parametr α můžeme určit například pomocí křížové validace.

Na konci této podkapitoly si uvedeme praktický příklad na našem datovém souboru a ukážeme si, jak klasifikační strom vytvořit a následně prořezat pomocí softwaru R. Jelikož se klasifikační stromy používají k binární klasifikaci, použijeme datový soubor, který obsahuje pouze dvě skupiny hráčů, a to corners a vnitřní polaře. Vhodné je ještě čtenáře upozornit, že v této kapitole nebudou vyzkoušena normovaná data, jelikož by se klasifikační strom stal neinterpretovatelným. Ještě než začneme, je třeba si nainstalovat do softwaru R knihovnu s názvem `tree`. Dále si soubor náhodně rozdělíme na trénovací a testovací část. Nyní již můžeme příkazem *tree* vytvořit klasifikační strom a následně si pomocí *summary* vypsat důležité vlastnosti stromu.

```
> library(tree)
> set.seed(2)
> train=sample(1:nrow(Baseballcorinf), 210)
> test=Baseballcorinf[-train,]
> pos.test=Baseballcorinf$pos[-train]
> tree.base=tree(pos~.,data = Baseballcorinf,subset = train)
> summary(tree.base)

Classification tree:
tree(formula = pos ~ ., data = Baseballcorinf, subset = train)
Variables actually used in tree construction:
[1] "hr"      "games"   "at.bats" "rbi"     "doubles" "obp"     "hits"    "avg"
Number of terminal nodes:  22
Residual mean deviance:  0.4893 = 91.98 / 188
Misclassification error rate: 0.1095 = 23 / 210
```

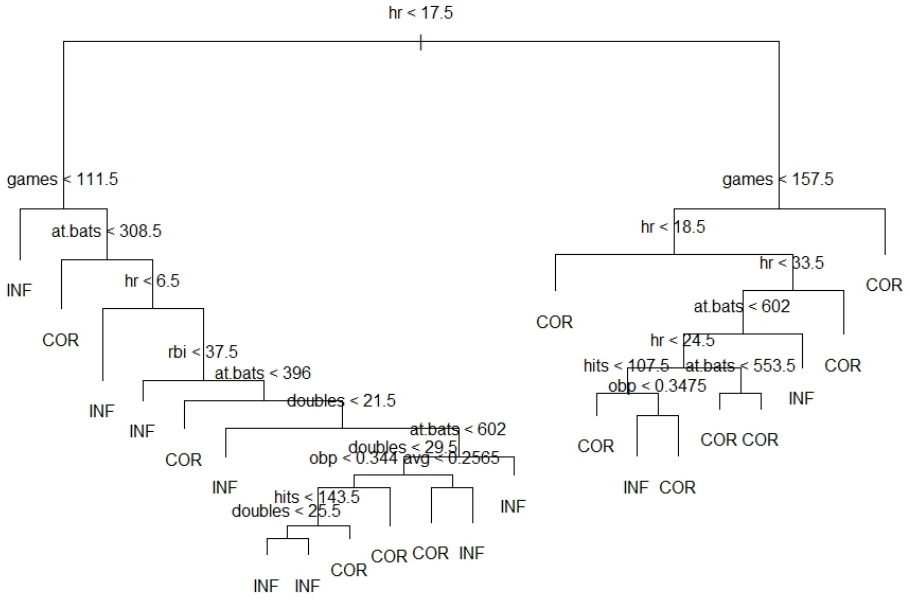
Ze *summary* si můžeme všimnout, kolik proměnných bylo při konstrukci stromu použito, počtu konečných uzlů nebo také jaká byla trénovací chyba modelu. Více nás však bude zajímat testovací chyba, kterou si vypíšeme pomocí následujícího

příkazu. Vyčíst ji opět můžeme z příslušné tabulky, stačí, když podělíme součet prvků na diagonále součtem všech prvků v tabulce. Tímto však získáme pouze poměr správně zařazených prvků, který jsme užívali v předchozích kapitolách. Proto toto číslo odečteme od jedničky a vynásobíme stem, jak můžeme vidět v posledním řádku příkazu. Výsledkem tedy je, že jsme více než dvě třetiny testovacích pozorování zařadili správně.

```
> tree.pred=predict(tree.base,test,type="class")
> table(tree.pred,pos.test)
         pos.test
tree.pred COR INF
      COR  21  16
      INF  17  36
> (1-(21+36)/90)*100
[1] 36.66667
```

Zbývá nám už jen si vytvořený klasifikační strom vykreslit. Příkazem *plot* vytvoříme jednotlivé větve stromu a pomocí *text* k nim přidáme popisky.

```
> plot(tree.base)
> text(tree.base)
```



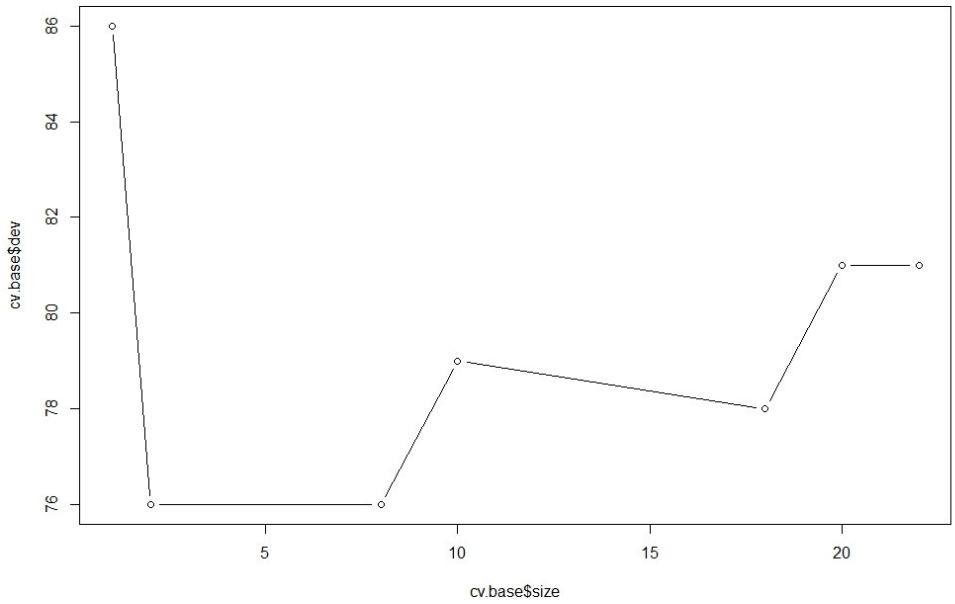
Obrázek 3.1: Klasifikační strom

Takto vypadá výsledný klasifikační strom. Jak již bylo na začátku kapitoly řečeno, klasifikační stromy jsou velmi jednoduché na interpretaci, a pojďme si nyní tento strom projít. Když se podíváme na první větvení, můžeme říci, že všichni hráči, kteří mají více než 17 homerunů, putují stromem doprava a ti, co mají 17 a méně, doleva. Takto bychom prošli celý strom, až dojdeme ke konečným větvím, kde je pozorování přiřazeno do jedné ze dvou skupin. Zkusíme dále tento strom prořezat a zároveň porovnat výsledné grafy a přesnost klasifikace. Pro prořezávání použijeme již dříve zmíněnou metodu *cost complexity prunning*. Nejprve si zavoláme funkci *cv.tree*, která provede křížovou validaci, za účelem zjistit posloupnost uvažovaných podstromů.

```
> set.seed(1)
> cv.base=cv.tree(tree.base,FUN=prune.misclass)
> cv.base
$size
[1] 22 20 18 10  8  2  1

$dev
[1] 81 81 78 79 76 76 86

$k
[1]      -Inf  0.000000  1.000000  1.375000  2.500000  2.666667 25.000000

$method
[1] "misclass"

attr(,"class")
[1] "prune"         "tree.sequence"
```

Parametr *size* zastupuje počet konečných uzlů uvažovaných podstromů a parametr *dev* udává míru chyby křížové validace. Nejmenší naměřená chyba je pro podstrom s osmi konečnými uzly a tuto hodnotu tedy využijeme při prořezávání. Posledním parametrem, který jsme pomocí této funkce získali, je k, který odpovídá již dříve zmíněnému parametru α. Pro zajímavost si ještě vztah parametrů *size* a *dev* vykresleme pomocí příkazu *plot*.

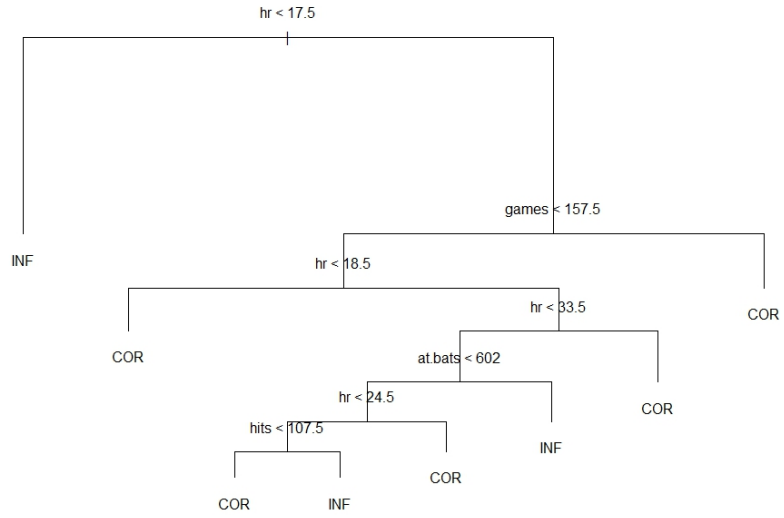
```
> plot(cv.base$size,cv.base$dev,type="b")
```



Obrázek 3.2: Vztah parametrů *size* a *dev*

Nyní již můžeme náš strom prořezat pomocí příkazu *prune.misclass* a vykreslit si ho. Spočítáme si opět testovací chybu.

```
> prune.base=prune.misclass(tree.base,best=8)
> plot(prune.base)
> text(prune.base)
> tree.pred=predict(prune.base,test,type="class")
> table(tree.pred,pos.test)
         pos.test
tree.pred COR INF
      COR  15  10
      INF  23  42
> (1-(15+42)/90)*100
[1] 36.66667
```



Obrázek 3.3: Prořezaný strom

Závěrem lze říci, že testovací chyba u prořezaného stromu zůstala stejná jako u neprořezaného, nicméně prořezaný strom je mnohem přehlednější a tudíž i snadnější na interpretaci. Prořezávání se tedy ukázalo jako velmi užitečné.

3.2. Pytlování

Klasifikační stromy, představené v minulé podkapitole, velmi často trpí vysokou variabilitou. Kdybychom například náš datový soubor rozdělili na dvě část a na každou z nich použili klasifikační strom, dostali bychom pravděpodobně velmi

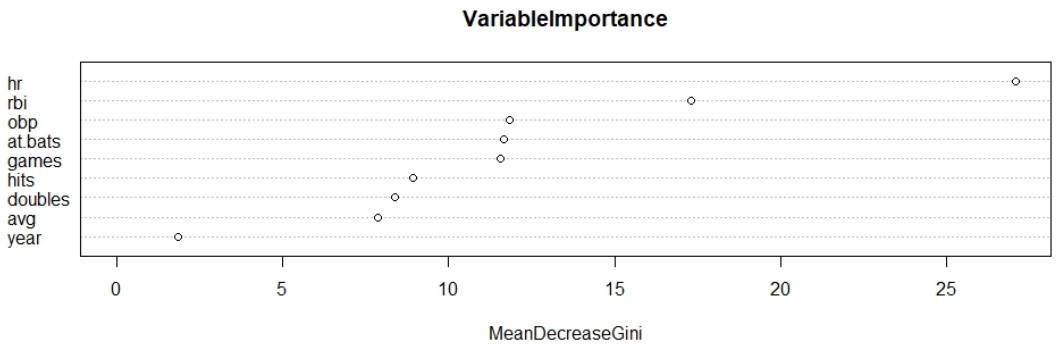
odlišné výsledky. Jedním ze způsobů, jak si s tímto problémem poradit, je takzvané pytlování (z anglického *bagging*), které si zde představíme. Tato metoda vychází z principu metody zvané *bootstrap*. Podstatou je, že z našeho datového souboru vždy náhodně vygenerujeme (vybíráme s opakováním) nějaký podsoubor o rozsahu původního souboru, na který následně aplikujeme zvolenou metodu. Toto provedeme B-krát a získáme tak B odlišných podsouborů. Následně natrénujeme klasifikační strom na b-tý podsoubor ($b \in \{1, \ldots, B\}$) a obdržíme model $\hat{f}^{*b}(\boldsymbol{x})$. Nakonec zprůměrujeme výsledky těchto modelů a dostaneme

$$\hat{f}_{bag}(\boldsymbol{x}) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(\boldsymbol{x}),$$

což se nazývá pytlování. Tím, že zprůměrujeme těchto B stromů, dosáhneme menší variability, což bylo naším cílem. Výsledkem této metody v kontextu klasifikace je, že pro všechna pozorování z testovacího souboru zjistíme, do jaké třídy je každý z B stromů klasifikoval a výslednou třídou bude taková, která byla klasifikována nejčastěji. Stanovení parametru B, tedy počtu vytvořených podsouborů, není pro pytlování kritické. V praxi volíme B dostatečně velké, aby byla výsledná testovací chyba co nejmenší. Tato chyba se označuje jako OOB chyba (z anglického *out-of-bag error*). Určíme ji tak, že pro každý z B podsouborů spočítáme klasifikační chybu pro ta pozorování, která se v podsouboru nevyskytla a následně zprůměrujeme. Tato pozorování se také někdy označují jako OOB pozorování. Praktickou ukázku v softwaru R a přesnost klasifikace si ukážeme v následující podkapitole, neboť pytlování a náhodné lesy, které budou obsahem další podkapitoly, spolu velmi úzce souvisí a pro jejich výstavbu se v softwaru používá stejná funkce.

V úvodu této kapitoly bylo řečeno, že jednou z hlavních výhod klasifikačních stromů je jejich názornost a snadná interpretovatelnost. Tato vlastnost klasifikačních stromů ovšem při použití pytlování odpadá, neboť kvůli velkému počtu stromů je již není možné graficky vyobrazit. Není též možné ze stromu vyčíst, které proměnné měly na klasifikaci největší podíl. Můžeme tedy říci, že pytlování vylepšuje přesnost modelu výměnou za snadnou interpretovatelnost. Je

však možné graficky znázornit důležitost každého z prediktorů, a to tak, že pro každý prediktor určíme, jaký podíl měl na průměrném snížení klasifikační chyby, popřípadě pomocí Giniho indexu.



Obrázek 3.4: Důležitost proměnných

Na grafu je vidět, že nejdůležitější proměnnou pro výstavbu stromů je počet homerunů následovaný počtem RBI. Naopak nejméně významnou proměnnou je rok, ze kterého statistiky pocházejí.

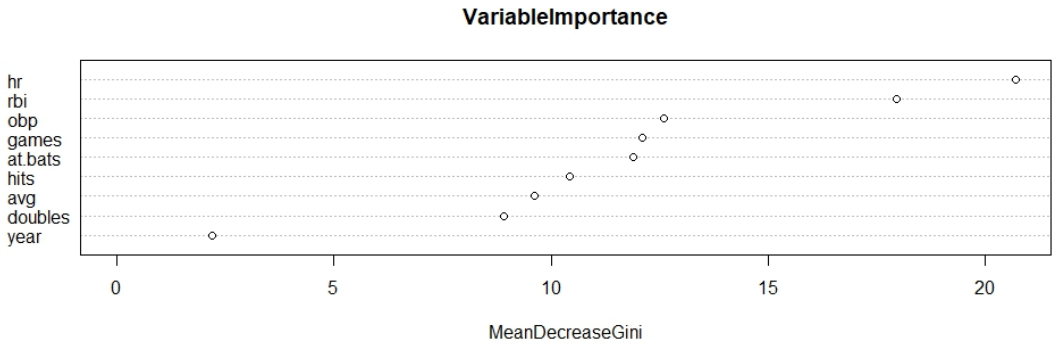
3.3. Náhodné lesy

Poslední metodou, kterou si v kontextu klasifikačních stromů představíme, jsou náhodné lesy (z anglického *random forests*). Jak již bylo zmíněno, náhodné lesy jsou velmi podobné metodě pytlování, přináší však jisté vylepšení. Připomeňme, že metoda pytlování uvažovala vždy B náhodných podsouborů, na kterých vytvořila klasifikační stromy. Při výstavbě těchto stromů byly uvažovány všechny možné prediktory, ale je zřejmé, že u většiny stromů hrály hlavní roli pouze ty nejvýznamnější. Představme si, že v našem souboru bude jeden velmi výrazný prediktor, pak je velmi pravděpodobné, že téměř každý ze stromů bude vypadat velmi podobně a výsledky podstromů tak budou silně korelovány. Náhodné lesy tento problém řeší tak, že při každém rozdělení stromu do dalších větví uvažují pouze některé náhodně zvolené prediktory. Takto vytvořené stromy budou daleko různorodější, než kdybychom použili metodu pytlování. Dále nás zajímá, kolik

prediktorů má být při každém dělení náhodně vybráno. Označme počet všech prediktorů jako p a počet náhodně vybraných jako m. Pak dle [4] standardně bereme $m \approx \sqrt{p}$. Je také vhodné si uvědomit, že jestliže zvolíme $m = p$, pak se z náhodných lesů stane metoda pytlování. Tuto vlastnost využijeme při praktické ukázce v softwaru R. Nejprve je nutné si nainstalovat knihovnu s názvem `randomForest`. Následně pomocí funkce *randomForest* vytvoříme náhodný les. Do funkce musíme zadat, kolik stromů má vytvořit a jaký bude počet uvažovaných prediktorů. Jelikož náš datový soubor obsahuje devět prediktorů, tak podle vztahu $m \approx \sqrt{p}$ v každém dělení stromu budeme uvažovat pouze tři náhodně zvolené prediktory.

```
> library(randomForest)
> set.seed(1)
> rf=randomForest(pos~.,data=Baseballcorinf,ntree=500,subset=train,mtry=3)
> rf

Call:
 randomForest(formula = pos ~ ., data = Baseballcorinf, ntree = 500,
mtry = 3, subset = train)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 26.67%
Confusion matrix:
    COR INF class.error
COR  51  36   0.4137931
INF  24 114   0.1739130
```

Vidíme, že pro vytvoření tohoto náhodného lesu jsme použili 500 stromů, počet uvažovaných prediktorů byl, jak již bylo zmíněno, tři a lehce pod 27% pozorování bylo zařazeno špatně. Připomeňme, že jednoduchý i prořezaný klasifikační strom dosahoval horších výsledků. Dále můžeme pozorovat, která proměnná měla největší podíl na průměrném zmenšení Giniho indexu.



Obrázek 3.5: Důležitost proměnných

Stejně tak, jako tomu bylo u metody pytlování, je nejvýznamnějším prediktorem počet homerunů. Čtenář by si však měl všimnout, že i když oba grafy vypadají velmi podobně, hodnoty se v mnoha případech liší. Nakonec se podívejme, jak dopadne metoda pytlování. V softwaru R použijeme stejný příkaz, jako při vytváření náhodného lesu s tím rozdílem, že nyní použijeme všech devět prediktorů.

```
> set.seed(1)
> bg=randomForest(pos~.,data=Baseballcorinf,ntree=500,subset=train,mtry=9)
> bg

Call:
 randomForest(formula = pos ~ ., data = Baseballcorinf, ntree = 500,
mtry = 9, subset = train)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 9

        OOB estimate of  error rate: 28.44%
Confusion matrix:
    COR INF class.error
COR  48  39   0.4482759
INF  25 113   0.1811594
```

Závěrem této kapitoly lze říci, že nejlépe dopadla metoda zvaná náhodné lesy, která špatně zařadila pouze lehce pod 27% trénovacích pozorování. Jako druhá skončila metoda pytlování s 28,44% špatně zařazenými pozorováními. Nakonec klasifikační i prořezaný klasifikační strom špatně zařadil třetinu pozorování.

Kapitola 4

Metoda SVM

Jako poslední si v této práci představíme metodu SVM (z anglického *support vector machines*). Než však začneme, je třeba čtenáře upozornit, že v této kapitole nebudeme některé termíny překládat a raději je nahradíme zkratkou. SVM je zobecnění velmi intuitivní metody maximálního okrajového klasifikátoru (z anglického *maximal margin classifier*), kterou si zde představíme také. Jelikož je metoda SVM náročnější na porozumění, bude jí v tomto textu věnováno více prostoru a představíme si ji od úplných základů. Při zpracování této kapitoly byla použita literatura [4], [7], [10], [15] a [17].

4.1. Maximální okrajový klasifikátor

Ještě než se seznámíme s klasifikováním pomocí maximálního okrajového klasifikátoru, ukážeme si jednoduchou motivaci s použitím nadrovin. V p-dimenzionálním prostoru se nadrovinou rozumí afinní podprostor s dimenzí $p-1$. Matematicky to můžeme zapsat následně

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0. \tag{4.1}$$

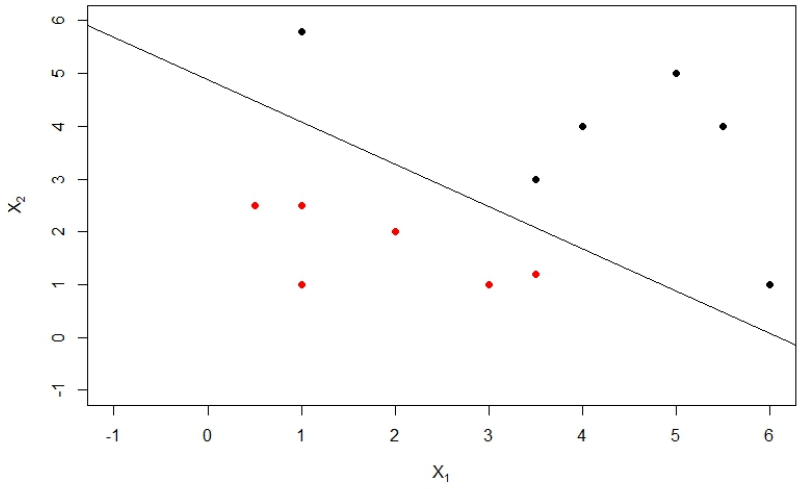
Dále předpokládejme, že $\boldsymbol{X} = (X_1, \ldots, X_p)^T$ nevyhovuje (4.1), ale místo toho platí, že po dosazení

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p > 0. \tag{4.2}$$

To nám říká, že $\boldsymbol{X}$ leží na jedné straně nadroviny, zatímco při

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p < 0 \tag{4.3}$$

leží bod na druhé straně nadroviny. O nadrovině se tedy dá říci, že p-dimenzionální prostor rozděluje na dvě poloviny. Nadrovina ve dvoudimenzionálním prostoru (tedy přímka) je pro ukázku vyobrazena níže.



Obrázek 4.1: Dvoudimenzionální nadrovina

Řekněme nyní, že bychom tedy pomocí nadroviny chtěli data klasifikovat. To bude možné provést pouze pro dvě pěkně rozdělené třídy, například tak, jako je tomu na obrázku výše. Pozorováním označeným červenou barvou můžeme přiřadit hodnotu $y_i = 1$ a černým $y_i = -1$. Rozdělující nadroviny mají tu vlastnost, že pro $\boldsymbol{x} = (x_{i1}, \ldots, x_{ip})^T$

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} > 0 \text{ jestliže } y_i = 1, \tag{4.4}$$

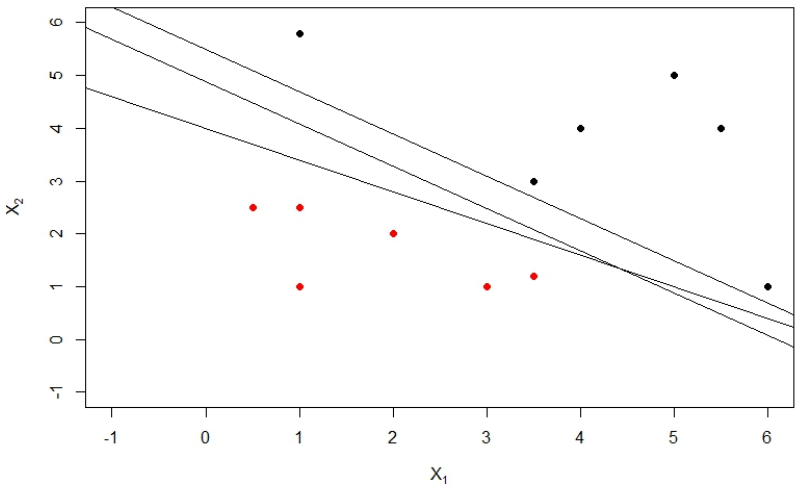
a

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} < 0 \text{ jestliže } y_i = -1. \tag{4.5}$$

To můžeme ekvivalentně zapsat jako

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) > 0. \tag{4.6}$$

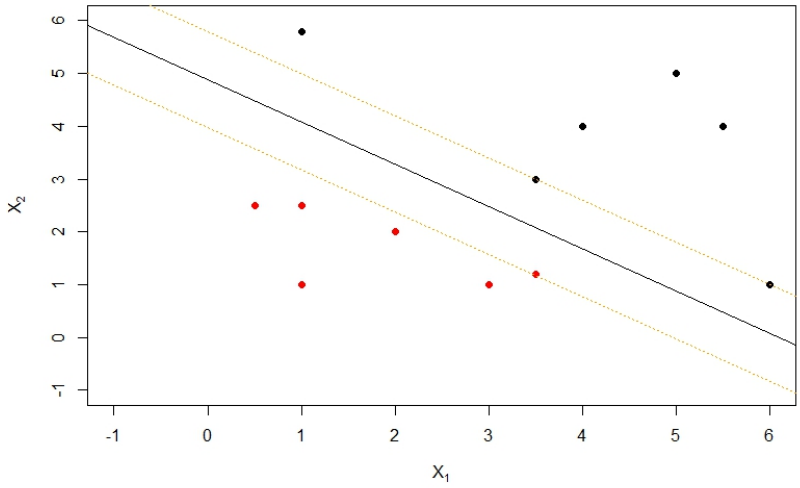
Pokud tyto rozdělující nadroviny existují, můžeme je velmi jednoduše použít pro klasifikaci dat. Nové testovací pozorování bude zařazeno do takové třídy, na jaké straně nadroviny se nachází. Jestliže je tedy možné naše data takto perfektně rozdělit pomocí nadrovin, pak bude existovat nekonečně mnoho možností, jak data rozdělit. S každou nadrovinou je totiž možné lehce pootočit, popřípadě ji o kousek přesunout tak, aby opět data perfektně rozdělila. Tato situace je znázorněna na následujícím grafu.



Obrázek 4.2: Možná rozdělení pomocí různých nadrovin

Kterou z nadrovin tedy vybrat? Přístup, který si zde popíšeme, se nazývá maximální okrajová nadrovina, což je taková rozdělující nadrovina, která je nejdále od trénovacích pozorování. Vypočítáme tedy vzdálenost od každého trénovacího pozorování k dané rozdělující nadrovině a nejmenší vzdálenost, kterou nalezneme, se nazývá okraj. Maximální okrajová nadrovina je pak taková, která má okraj největší. Lze tedy říci, že maximální okrajová nadrovina je rozdělující nadrovina, která má největší minimální vzdálenost od trénovacích pozorování. Na jejím základě pak můžeme testovací pozorování klasifikovat podobně, jako tomu bylo

u rozdělujících nadrovin. Tato metoda je známá jako maximální okrajový klasifikátor. Ještě než si ukážeme, jak takový maximální okrajový klasifikátor zkonstruovat, se podíváme na jeho grafické znázornění.



Obrázek 4.3: Maximální okrajový klasifikátor

Na obrázku vidíme maximální okrajovou nadrovinu vyznačenou černou čarou. Okraj je dán vzdáleností mezi oranžovou přerušovanou čarou a černou čarou. Dále zde vidíme několik trénovacích pozorování pocházejících ze dvou tříd. Dvě černá a jedno červené pozorování, která leží na oranžové přerušované čáře, se nazývají podpůrné vektory (z anglického *support vector*).

Nyní se podíváme, jak zkonstruovat maximální okrajový klasifikátor. Předpokládejme, že máme množinu trénovacích pozorování $\boldsymbol{x_1}, \ldots, \boldsymbol{x_n} \in \mathbb{R}^p$ a příslušné označení třídy $y_1, \ldots, y_n \in \{-1, 1\}$. Najít maximální okrajovou nadrovinu vede k řešení podmíněné optimalizace

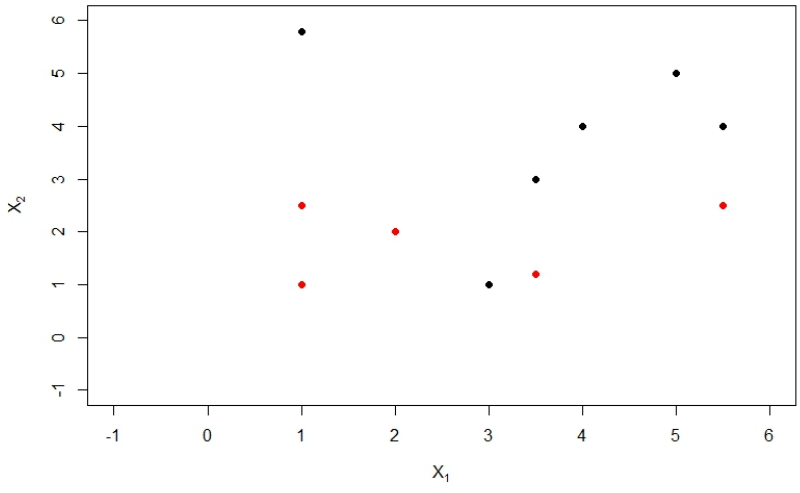
$$\underset{\beta_0, \beta_1, \ldots, \beta_p}{\text{maximalizovat}} \quad M \tag{4.7}$$

$$\text{za podmínky} \quad \sum_{j=1}^{p} \beta_j^2 = 1, \tag{4.8}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M \ \forall \ i = 1, \ldots, n. \tag{4.9}$$

Podmínky, respektive omezení (4.8) a (4.9) zajišťují, aby bylo každé pozorování na správné straně nadroviny ve vzdálenosti rovné nejméně M od této nadroviny. Dále M zde představuje již dříve zmíněný okraj naší nadroviny a chceme zvolit takov8 $\beta_0, \beta_1, \ldots, \beta_p$, kter8 maximalizují M, což je přesně definice maximální okrajové nadroviny. Řešení takovéto optimalizační úlohy je nicméně nad rámec této diplomové práce.

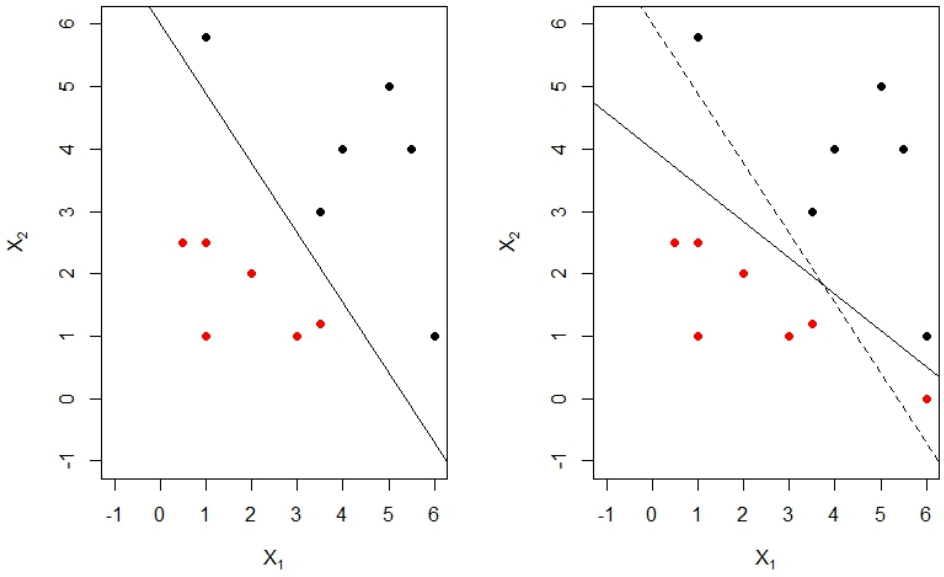
Dosud jsme předpokládali, že naše data se dají perfektně rozdělit pomocí nadrovin. Je však zřejmé, že v praxi se s takto rozdělenými daty pravděpodobně nesetkáme. Optimalizační úloha (4.7) - (4.9) tedy nebude mít řešení pro $M > 0$. V další podkapitole si tedy maximální okrajový klasifikátor zobecníme pro případ, kdy nadrovinou data *téměř* rozdělíme do tříd. Tento přístup nazveme podpůrný vektorový klasifikátor (z anglického *support vector classifier*). Nejdříve si však pro orientaci ukážeme případ, kdy data nedokážeme pomocí nadrovin rozdělit.



Obrázek 4.4: Případ nerozdělitelných dat pomocí nadrovin

4.2. Podpůrný vektorový klasifikátor

Na obrázku 4.4 v předchozí podkapitole jsme si ukázali případ, kdy není možné data perfektně rozdělit pomocí nějaké nadroviny. Problémů však může nastat více. Řekněme například, že přidáme do naší trénovací množiny jedno pozorování navíc. Toto jedno trénovací pozorování může mít obrovský vliv na to, jak bude nakonec například maximální okrajová nadrovina vypadat. Vše je ilustrováno na následujícím obrázku.

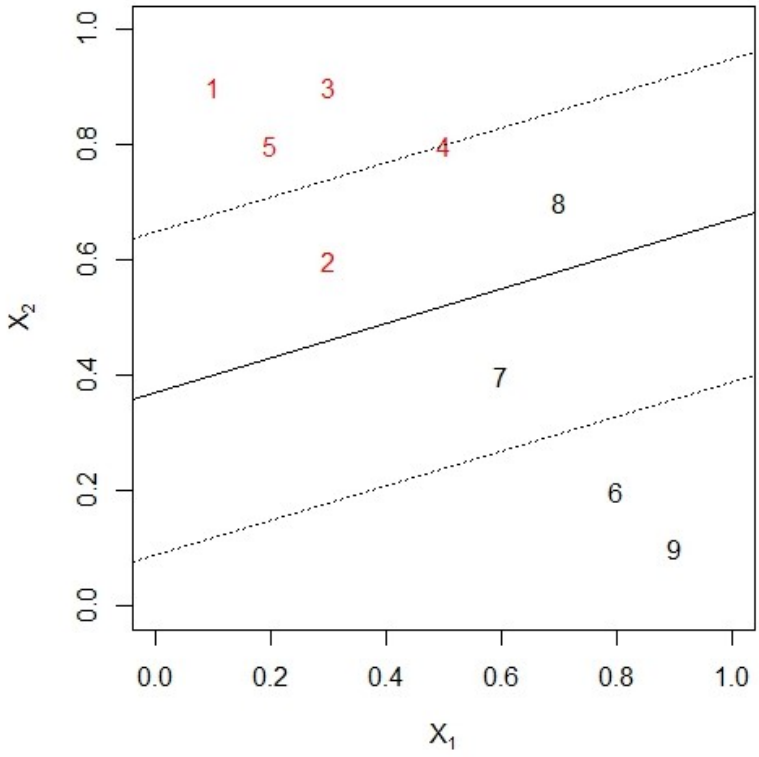


Obrázek 4.5: Změna nadroviny po přidání jednoho pozorování

Obrázek vlevo ilustruje rozdělení dat pomocí maximální okrajové nadroviny. Na pravém obrázku si můžeme všimnout, že přibylo jedno červené trénovací pozorování. Přidáním tohoto jednoho pozorování nám původní maximální okrajovou nadrovinu označenou přerušovanou černou čarou posunulo na novou nadrovinu označenou plnou čarou. Toto posunutí by pak mohlo hrát velkou roli při následné klasifikaci.

Abychom těmto problémům předešli, ukážeme si podpůrný vektorový klasifikátor, který již nerozděluje data perfektně, ale za účelem lepší klasifikace testo-

vacích pozorování raději pár trénovacích pozorování zařadí chybně. Někdy se také dle [4] nazývá mírný okrajový klasifikátor (z anglického *soft margin classifier*). Tento klasifikátor nejenže povolí, aby pozorování leželo na opačné straně okraje, ale dokonce aby leželo na opačné straně nadroviny.



Obrázek 4.6: Podpůrný vektorový klasifikátor

Na obrázku výše můžeme opět vidět pozorování pocházející ze dvou tříd. Dále zde černou čarou máme označenu nadrovinu a přerušovanými čarami příslušné okraje. Nejprve se zaměříme na červená pozorování. Pozorování 1, 3 a 5 jsou na správné straně okraje, zatímco pozorování číslo 2 je na špatné straně okraje a číslo 4 je přesně na okraji. Následně se podíváme na černá pozorování. Čísla 6 a 9 jsou na správné straně okraje, kdežto pozorování číslo 7 je na straně špatné. Pozorování číslo 8 je dokonce na špatné straně nadroviny, čímž je samozřejmě

i na špatné straně okraje.

Nyní se podíváme na samotnou konstrukci podpůrného vektorového klasifikátoru. Ten klasifikuje testovací pozorování podle toho, na jaké straně nadroviny leží. Nadrovina je zvolena tak, aby správně rozdělila většinu trénovacích pozorování do dvou tříd, nicméně je možné, že některá pozorování budou zařazena špatně. Najít tuto nadrovinu vede na následující úlohu podmíněné optimalizace

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximalizovat}} \quad M \tag{4.10}$$

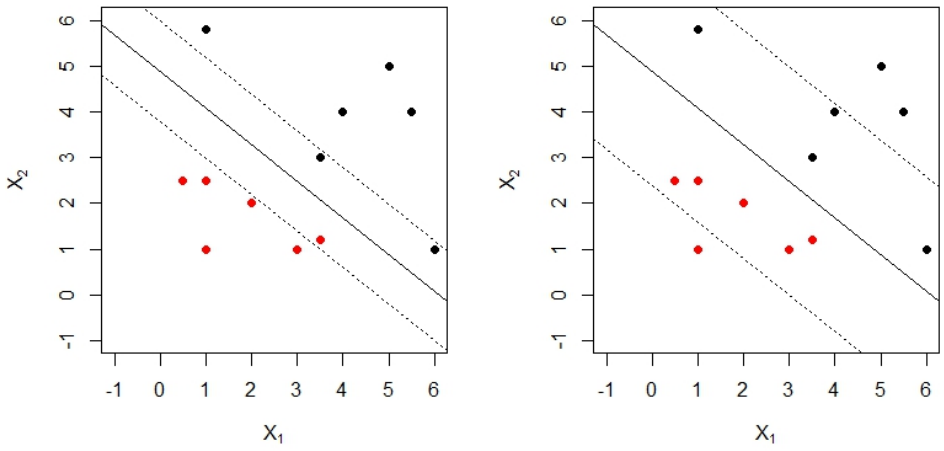
$$\text{za podmínky} \quad \sum_{j=1}^{p} \beta_j^2 = 1, \tag{4.11}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i), \tag{4.12}$$

$$\epsilon_i \geq 0, \ \sum_{i=1}^{n} \epsilon_i \leq C, \tag{4.13}$$

kde C je nezáporný ladící parametr. Hodnoty $\epsilon_1, \ldots, \epsilon_n$ v (4.12) jsou takzvané volné proměnné. Podívejme se na tyto volné proměnné podrobněji. Volné proměnné nám dávají informaci o tom, kde se pozorování vzhledem k nadrovině a vzhledem k okraji nachází. Jestliže $\epsilon_i = 0$, pak i-té pozorování je na správné straně okraje. Dále pokud je $\epsilon_i > 0$, pak řekneme, že i-té pozorování je na špatné straně okraje. V této situaci řekneme, že i-té pozorování porušuje okraj. A konečně $\epsilon_i > 1$ značí, že i-té pozorování je na špatné straně nadroviny. Nyní si povíme něco o funkci parametru C. Když se podíváme na podmínku (4.13), zjistíme, že C omezuje součet ϵ_i a udává nám tedy, jak moc budeme porušovat okraj, popřípadě nadrovinu. Jestliže $C = 0$, pak žádné pozorování nesmí porušit okraj nebo být na špatné straně nadroviny a úloha (4.10) - (4.13) se zjednoduší na hledání maximální okrajové nadroviny dané předpisem (4.7) - (4.9). Pro $C > 0$ nesmí být více než C pozorování na špatné straně nadroviny. Čím se C zvyšuje,

tím jsme tolerantnější k porušení okraje a okraj tedy bude širší. Jaký vliv by mělo různé nastavení parametru C ilustruje následující obrázek.



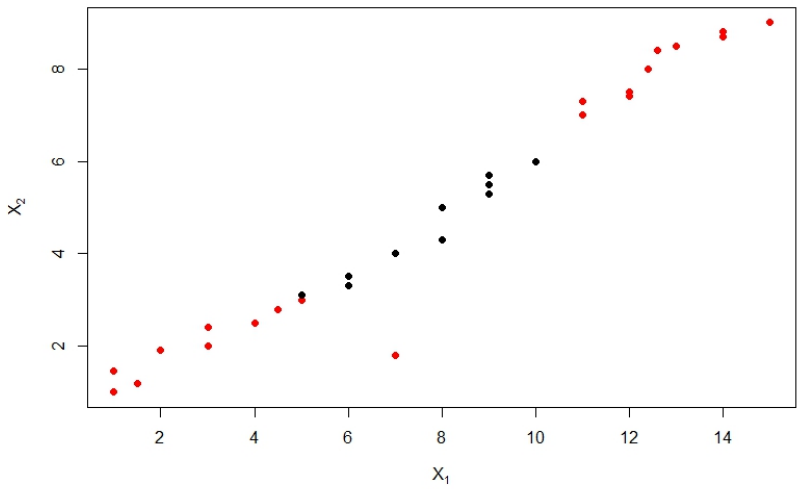
Obrázek 4.7: Různé nastavení parametru C

Na obrázku lze vidět, jak nám nastavení parametru C ovlivní šířku okraje a tudíž i počet pozorování, která okraj porušují. Na levém obrázku porušilo okraj pouze jedno červené a dvě černá pozorování, zatímco na tom pravém pět červených a čtyři černé.

Na konci této podkapitoly se ještě krátce zamyslíme nad optimalizační úlohou (4.10) - (4.13). Když se na tuto úlohu podíváme podrobněji, zjistíme, že pouze pozorování, která leží na okraji nebo tento okraj porušují, mají vliv na výslednou nadrovinu a tudíž i na výsledný klasifikátor. Nově přidané trénovací pozorování, které bude ležet na správné straně okraje, nám tedy nezmění výsledný klasifikátor, což je oproti maximálnímu okrajovému klasifikátoru velkou výhodou. Podpůrnými vektory budeme nazývat ta pozorování, která leží přímo na okraji nebo na jeho špatné straně.

4.3. SVM

V této poslední podkapitole si nejdříve ukážeme, co dělat, když naše data nebude možné rozdělit pomocí podpůrného vektorového klasifikátoru v podobě, jak jsme jej uvažovali dříve, a následně se seznámíme s metodou SVM. Dosud jsme předpokládali, že data lze rozdělit pomocí lineární hranice. V praxi se nicméně často setkáme s daty, která budeme muset rozdělit pomocí nelineárních hranic. Tato situace je ilustrována na následujícím obrázku.



Obrázek 4.8: Situace, kdy bude data nutné rozdělit nelineární hranicí

Abychom mohli data rozdělit jinak než lineárně, pozměníme již dříve zmíněný podpůrný vektorový klasifikátor, a to tak, že rozšíříme prostor proměnných pomocí kvadratických, kubických nebo dalších polynomiálních funkcí prediktorů. Například místo toho, abychom při konstrukci podpůrného vektorového klasifikátoru použili p proměnných

$$X_1, X_2, \ldots, X_p,$$

můžeme použít $2p$ proměnných

$$X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2.$$

Potom optimalizační úlohu (4.10) - (4.13) můžeme zapsat jako

$$\underset{\beta_0,\beta_{11},\beta_{12},\ldots,\beta_{p1},\beta_{p2},\epsilon_1,\ldots,\epsilon_n}{\text{maximalizovat}} \quad M \tag{4.14}$$

$$\text{za podmínky} \quad y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_{j1} x_{ij} + \sum_{j=1}^{p} \beta_{j2} x_{ij}^2 \right) \geq M(1 - \epsilon_i), \tag{4.15}$$

$$\sum_{i=1}^{n} \epsilon_i \leq C, \ \epsilon_i \geq 0, \ \sum_{j=1}^{p} \sum_{k=1}^{2} \beta_{jk}^2 = 1. \tag{4.16}$$

V rozšířeném prostoru proměnných je hranice daná (4.10) - (4.13) opět lineární. V původním prostoru je však hranice daná rovnicí $q(\boldsymbol{x}) = 0$, kde q je kvadratický polynom a řešení tak tedy bude nelineární.

Nyní se podíváme na samotnou metodu SVM, která je rozšířením podpůrného vektorového klasifikátoru. Opět budeme rozšiřovat prostor proměnných, nyní však pomocí takzvaných jader. Nejprve zavedeme skalární součin (z anglického *inner product*). Skalární součin mezi dvěmi p-rozměrnými pozorováními je definován jako

$$\langle \boldsymbol{x_i}, \boldsymbol{x_{i'}} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}. \tag{4.17}$$

Dále můžeme ukázat, že lineární podpůrný vektorový klasifikátor může být reprezentován jako

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle \boldsymbol{x}, \boldsymbol{x_i} \rangle. \tag{4.18}$$

Je třeba ještě zmínit, že pro získání funkce $f(\boldsymbol{x})$ potřebujeme spočítat skalární součin mezi novým pozorováním $\boldsymbol{x}$ a každým trénovacím pozorováním $\boldsymbol{x_i}$. Ukazuje

se však, že α_i je různé od nuly pouze pro podpůrné vektory. Pokud tedy trénovací pozorování není podpůrným vektorem, potom se α_i rovná nule. Nechť $\mathcal{S}$ je nyní množina indexů podpůrných vektorů. Pak můžeme (4.18) přepsat jako

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle \boldsymbol{x}, \boldsymbol{x_i} \rangle, \tag{4.19}$$

což obvykle vede ke značnému zjednodušení. K získání lineárního podpůrného klasifikátoru $f(\boldsymbol{x})$ tedy potřebujeme pouze skalární součiny. Dále předpokládejme, že pokaždé, když se skalární součin (4.17) objeví v (4.18), nahradíme ho následujícím zobecněním

$$K(\boldsymbol{x_i}, \boldsymbol{x_{i'}}), \tag{4.20}$$

kde K je tzv. jádro. Jádro je funkce, která určuje míru podobnosti dvou pozorování. Uveďme zde například

$$K(\boldsymbol{x_i}, \boldsymbol{x_{i'}}) = \sum_{j=1}^{p} x_{ij} x_{i'j}, \tag{4.21}$$

čímž získáme zpět podpůrný vektorový klasifikátor. Jádro (4.21) označujeme jako lineární jádro, jelikož podpůrný vektorový klasifikátor je lineární funkcí parametrů. Dalším příkladem může být

$$K(\boldsymbol{x_i}, \boldsymbol{x_{i'}}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d, \tag{4.22}$$

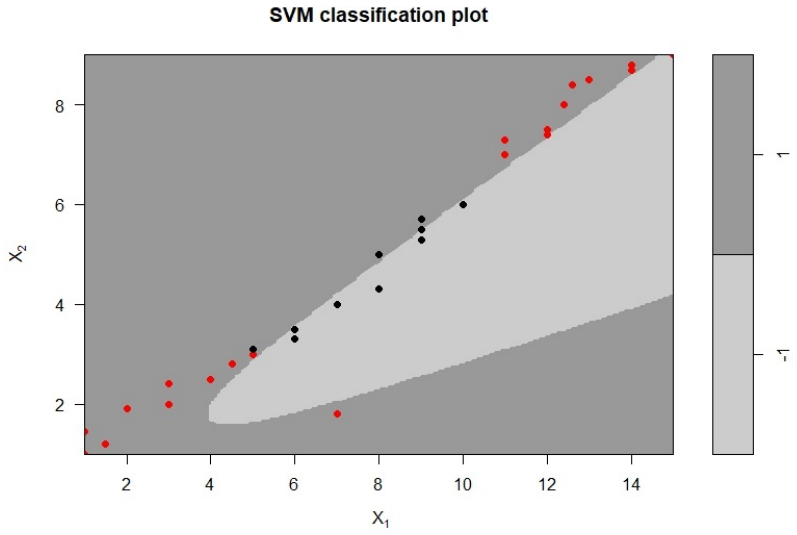
které je známé jako polynomiální jádro stupně d. Pakliže je podpůrný vektorový klasifikátor kombinován s nelineárním jádrem, výsledný klasifikátor označíme jako SVM. Funkce $f(\boldsymbol{x})$ bude v tomto případě ve tvaru

$$f(\boldsymbol{x}) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(\boldsymbol{x}, \boldsymbol{x_i}). \tag{4.23}$$

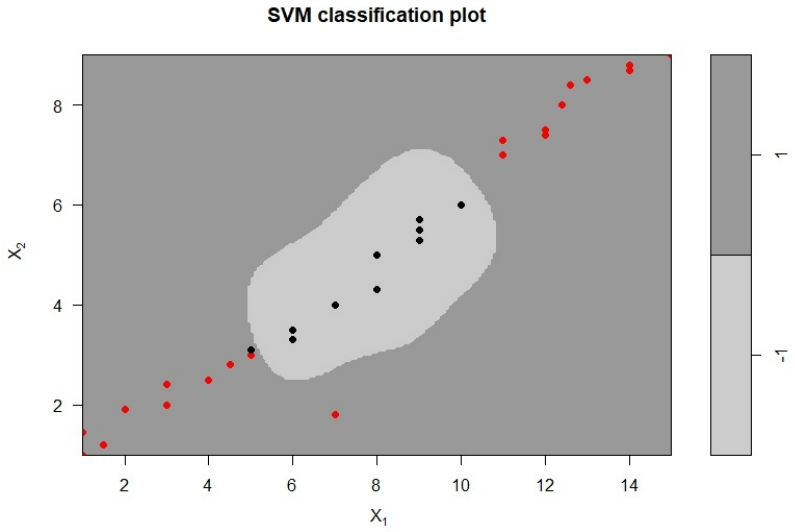
Další jádro, které se hojně využívá, se nazývá Gaussovo jádro. V některých literaturách jej lze najít také pod názvem *radial kernel*. Jeho předpis je následující

$$K(\boldsymbol{x_i}, \boldsymbol{x_{i'}}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2), \tag{4.24}$$

kde γ je kladná konstanta. Nyní si ukážeme pomocí speciálního grafu pro metodu SVM, jak by vypadalo polynomiální, respektive Gaussovo jádro na datech, která byla představena na obrázku (4.8).



Obrázek 4.9: Ukázka polynomiálního jádra se stupněm 3



Obrázek 4.10: Ukázka Gaussova jádra

Vidíme, že obě jádra data rozdělí podstatně lépe, než jádro lineární. Tímto by byla teorie zabývající se odvozením metody SVM a metodou samotnou téměř hotová. Ještě než se však pustíme do praktické ukázky na našich datech, je třeba zmínit ještě jednu podstatnou věc. Zatím jsme v našich úvahách uvažovali pouze případ, kdy data rozdělujeme do dvou skupin a provádíme tak binární klasifikaci. Metodu SVM lze rozšířit tak, aby bylo možné ji použít na klasifikování do více než dvou tříd. Uvedeme si zde dva nejpoužívanější způsoby.

První bude takzvaná "jeden vůči jednomu" klasifikace (z anglického *one-versus-one classification*), která je někdy také označována jako klasifikace "všech párů" (z anglického *all-pairs classification*). Máme-li tedy $K > 2$ tříd, tak tento přístup zkonstruuje $\binom{K}{2}$ SVM, kde každý porovnává dvě třídy. Testovací pozorování pak zařadíme do takové třídy, do které byla nejčastěji klasifikována v těchto $\binom{K}{2}$ binárních klasifikacích.

Druhým způsobem je takzvaná "jeden vůči všem" klasifikace (z anglického *one-versus-all classification*). Zde nejprve vytvoříme K SVM a pokaždé porovnáváme jednu z K tříd oproti ostatním $K-1$ třídám. Nechť $\beta_{0k}, \beta_{1k}, \ldots, \beta_{pk}$ označují parametry, které získáme z výsledného SVM pro k-tou třídu. Dále nechť $\boldsymbol{x}^*$ je

testovací pozorování. Potom toto pozorování zařadíme do takové třídy, pro kterou je $\beta_{0k} + \beta_{1k}x_1^* + \cdots + \beta_{pk}x_p^*$ největší.

Nakonec si ukážeme praktickou ukázku v softwaru R. Nejprve se budeme zabývat binární klasifikací, kterou vyzkoušíme i na normovaná data. Za tímto účelem budeme potřebovat nahrát knihovnu `e1071`. Dále data opět rozdělíme za pomoci knihovny `caTools` a pomocí funkce *svm* provedeme klasifikaci metodou SVM. Začněme nejprve s lineárním jádrem.

```
> library(e1071)
> library(caTools)
> set.seed(1)
> data=Baseballcorinf
> data$spl=sample.split(data,SplitRatio = 0.7)
> train=subset(data, data$spl==TRUE)
> test=subset(data, data$spl==FALSE)
> train=train[,-11]
> test=test[,-11]
> svm_model <- svm(pos~ ., data=train, method="C-classification", kernel="linear")
> summary(svm_model)

Call:
svm(formula = pos ~ ., data = train, method = "C-classification", kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1
      gamma:  0.1111111

Number of Support Vectors:  132

 ( 64 68 )


Number of Classes:  2

Levels:
 COR INF
```

Pomocí příkazu *summary* si opět můžeme vypsat některé zajímavé informace. Zde například vidíme, že jsme jako jádro opravdu použili jádro lineární, a že parametr $C = 1$. Dále si můžeme všimnout počtu podpůrných vektorů a počtů tříd, do kterých klasifikujeme. Dalším krokem bude otestovat tento SVM model na našich testovacích datech.

```
> pred_test <-predict(svm_model,test)
> mean(pred_test==test$pos)
[1] 0.7555556
```

Při použití lineárního jádra jsme tedy správně zařadili více než 75% testo-

vacích pozorování. Mohlo by nás však zajímat, jak by se změnila klasifikace při změně parametru C. Za tímto účelem použijeme příkaz *tune*, který pomocí křížové validace vybere optimální parametry pro náš model. Nejdříve však musíme velmi jednoduše připravit vstupní data, což je ilustrováno níže.

```
> x=data[,-1]
> y=data[,1]
> svm_tune <- tune(svm, train.x=x, train.y=y, kernel="linear", ranges=list(cost=10^(-1:2)))
> print(svm_tune)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
    1

- best performance: 0.2666667
```

Při použití k složkové křížové validace zjistíme, že optimální hodnota parametru je $C = 1$, což je i hodnota, kterou jsme použili. Dále si ukážeme klasifikaci použitím polynomiálního jádra.

```
> svm_model <- svm(pos~ ., data=train, method="C-classification", kernel="polynomial")
> summary(svm_model)

Call:
svm(formula = pos ~ ., data = train, method = "C-classification", kernel = "polynomial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  1
     degree:  3
      gamma:  0.1111111
     coef.0:  0

Number of Support Vectors:  151

 ( 74 77 )


Number of Classes:  2

Levels:
 COR INF



> pred_test <-predict(svm_model,test)
> mean(pred_test==test$pos)
[1] 0.7777778
```

Za pomoci příkazu *summary* si opět můžeme zjistit hodnoty použitých parametrů, stupeň použitého polynomu, popřípadě počet podpůrných vektorů. Polynomiálním jádrem třetího stupně jsme nyní správně klasifikovali více než 77%

testovacích pozorování a můžeme tedy sledovat lehké zlepšení oproti lineárnímu jádru. Opět bylo ověřeno i jiné (optimální) nastavení parametrů, ale také v tomto případě to nevedlo k jejich změně. Posledním jádrem, které jsme v této kapitole uvedli, je Gaussovo jádro (v softwaru R označeno jako *radial*) a ukážeme tedy i jeho použití.

```
> svm_model <- svm(pos~ ., data=train, method="C-classification", kernel="radial")
> summary(svm_model)

Call:
svm(formula = pos ~ ., data = train, method = "C-classification", kernel = "radial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.1111111

Number of Support Vectors:  156

 ( 76 80 )


Number of Classes:  2

Levels:
 COR INF



> pred_test <-predict(svm_model,test)
> mean(pred_test==test$pos)
[1] 0.7555556
```

Gaussovo jádro nám správně klasifikovalo stejné procento testovacích pozorování jako jádro lineární. Zde se však ukázalo, že nastavení parametru $\gamma = 0,5$ vedlo k vylepšení klasifikace, konkrétně pak tento model správně zařadil 76,67% pozorování. Tato hodnota parametru γ byla opět zjištěna pomocí příkazu *tune*. Všechna tři jádra byla použita i na normovaná data, výslednou klasifikaci to však neovlivnilo. Nyní se ještě zaměříme na klasifikaci do všech tří tříd. Použijeme zde opět lineární, polynomiální a Gaussovo jádro na původní i normovaná data. Jelikož je postup téměř identický, ukážeme si rovnou příslušné kódy, které na konci rozebereme.

```
> svm_model <- svm(pos~ ., data=train, method="C-classification", kernel="linear")
> summary(svm_model)

Call:
svm(formula = pos ~ ., data = train, method = "C-classification", kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1
      gamma:  0.1111111

Number of Support Vectors:  314

 ( 82 104 128 )


Number of Classes:  3

Levels:
 COR INF OF



> pred_test <-predict(svm_model,test)
> mean(pred_test==test$pos)
[1] 0.5208333
```

```
> svm_model <- svm(pos~ ., data=train, method="C-classification", kernel="polynomial")
> summary(svm_model)

Call:
svm(formula = pos ~ ., data = train, method = "C-classification", kernel = "polynomial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  1
     degree:  3
      gamma:  0.1111111
     coef.0:  0

Number of Support Vectors:  327

 ( 84 115 128 )


Number of Classes:  3

Levels:
 COR INF OF



> pred_test <-predict(svm_model,test)
> mean(pred_test==test$pos)
[1] 0.4513889
```

```
> svm_model <- svm(pos~ ., data=train, method="C-classification", kernel="radial")
> summary(svm_model)

Call:
svm(formula = pos ~ ., data = train, method = "C-classification", kernel = "radial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.1111111

Number of Support Vectors:  325

 ( 83 114 128 )


Number of Classes:  3

Levels:
 COR INF OF



> pred_test <-predict(svm_model,test)
> mean(pred_test==test$pos)
[1] 0.4861111
```

Po prozkoumání výše zmíněných kódů a hlavně jejich výsledků můžeme říci, že nejlépe nám testovací data do tří tříd zařadil lineární SVM (52,08%). Na pomyslném druhém místě skončil SVM využívající Gaussova jádra (48,62%) a nejhůře dopadl polynomiální SVM třetího stupně, který správně klasifikoval pouze 45,14% pozorování. Dále bylo pro každé jádro otestováno i jiné nastavení parametrů, nicméně v tomto případě to ke zlepšení klasifikace nevedlo. Normování dat na výslednou klasifikaci opět nemělo vliv.

Kapitola 5

Porovnání metod klasifikace dat

Všechny dříve získané výsledky jsou značně ovlivněny rozdělením datového souboru na trénovací a testovací množinu, čehož si může čtenář povšimnout například u logistické regrese nebo u klasifikačních stromů. Díky této skutečnosti byla každá metoda provedena stokrát, vždy na náhodně rozděleném souboru. Autor diplomové práce uvažoval i o větším počtu opakování metod, nicméně výsledky klasifikace se stabilizovaly již při dvaceti opakování. Podíly správně zařazených pozorování pak byly zprůměrovány a jejich pomocí následně metody porovnáme. Dále budou zmíněny klady a zápory některých metod. Autor této práce by však rád zmínil, že výsledky zde prezentované jsou platné pouze pro použitý datový soubor a na tuto skutečnost je tedy třeba při vlastní analýze dat myslet. Metody, které v této práci dosahovaly nejlepších výsledků, mohou pro jiné datové soubory dávat naopak výsledky nejhorší. Je tedy vždy vhodné při klasifikaci vyzkoušet více metod.

5.1. Výsledky binární klasifikace

V této podkapitole si pomocí tabulky porovnáme všechny metody použité při binární klasifikaci. Uvedeme zde také vliv normování dat na klasifikaci a klady a zápory metod. V tabulce bude také pro zajímavost zmíněno pořadí metod (pro původní data), které bude stanoveno podle průměrného podílu správně zařazených testovacích pozorování.

Metoda	Přesnost původní data	Přesnost normovaná data	Pořadí
Logistická regrese	73,89%	73,89%	4.
LDA	74,29%	74,29%	3.
QDA	70,21%	70,21%	6.
KNN (k=5)	69,00%	67,72%	9.
Klasifikační strom	77,98%	-	2.
Prořezaný klasifikační strom	78,49%	-	1.
Náhodný les	68,99%	-	10.
Pytlování	68,46%	-	11.
SVM - lineární	73,09%	73,09%	5.
SVM - polynomiální	69,07%	69,07%	8.
SVM - Gaussovo	69,89%	69,89%	7.

Tabulka 5.1: Výsledky binární klasifikace

První informací, kterou můžeme z tabulky vyčíst, je vliv normování dat na jednotlivé metody. Normováním dat se nám výsledná klasifikace změnila pouze u metody KNN. Je také vhodné dodat, že toto normování mělo na výsledek klasifikace negativní dopad. Dále se podíváme na jednotlivé metody podle jejich pořadí úspěšnosti klasifikace. Nejlépe data klasifikoval prořezaný klasifikační strom, mezi jehož výhody patří zejména grafická reprezentace a interpretovatelnost. Hned v závěsu skončil klasický klasifikační strom. Mezi velkou nevýhodu těchto dvou metod však patří možnost pouze binární klasifikace. Na třetím místě se umístila LDA s více než 74% správně zařazených testovacích pozorování. Jednou z předností této metody je možnost klasifikovat data do více skupin. Čtvrtou v pořadí je logistická regrese, u které jsme však limitování binární klasifikací, jelikož pro klasifikaci do více skupin se metoda nevyužívá. Na pátém místě skončila metoda SVM s lineárním jádrem, která klasifikovala více než 73% testovacích pozorování správně. Mezi nevýhody této metody patří poměrně složitější teoretická stránka a nutnost nastavit některé parametry. Další místo obsadila metoda QDA s úspěšností více než 70%. U této metody je však třeba myslet na dostatečný rozsah datového souboru s čímž úzce souvisí nutnost odhadu varianční matice pro každou třídu zvlášť, což zejména při velkém počtu tříd může být problém.

Sedmé a osmé patří metodě SVM s Gassovým respektive polynomiálním jádrem. Metoda KNN s pěti nejbližšími sousedy pak obsadila deváté místo. Mezi výhody této metody patří zejména velmi intuitivní postup klasifikace. Na předposledním, respektive posledním místě skončila metoda náhodných lesů a pytlování. U těchto metod bohužel ztrácíme možnost grafické reprezentace. Jak jsem si však ukázali, lze graficky znázornit důležitost jednotlivých proměnných na výstavbě těchto modelů.

5.2. Výsledky klasifikace do tří skupin

V minulé podkapitole jsme se podívali, jak si jednotlivé metody vedly při binární klasifikaci a uvedli si také některá možná omezení, popřípadě výhody. V této podkapitole si pak již pouze ukážeme, jak jednotlivé metody dopadly při klasifikaci do všech tří skupin. Ještě jednou je vhodné zmínit, že pořadí jednotlivých metod platí pouze na našem uvedeném datovém souboru a při klasifikaci vlastního datového souboru by měl čtenář vyzkoušet metod více a zvážit i různá nastavení parametrů.

Metoda	**Přesnost původní data**	**Přesnost normovaná data**	**Pořadí**
LDA	48,47%	48,47%	1.
QDA	47,51%	47,51%	3.
KNN (k=7)	41,85%	42,76%	6.
SVM - lineární	48,31%	48,31%	2.
SVM - polynomiální	43,54%	43,54%	5.
SVM - Gaussovo	47,02%	47,02%	4.

Tabulka 5.2: Výsledky klasifikace do tří skupin

Nejlépe i v tomto případě dopadla metod LDA. Na druhém místě skončila metoda SVM využívající lineárního jádra. Jako třetí skončila metoda QDA s více než 47% správně zařazených testovacích pozorování. Na čtvrtém a pátém místě se umístila metoda SVM s použitím Gaussova, respektive polynomiálního jádra. Na posledním místě skončila metoda KNN pro sedm nejbližších sousedů. U tuto

metody mělo stejně tak jako u binární klasifikace vliv normování dat. Tentokrát však byl tento vliv pozitivní ve smyslu úspěšnosti klasifikace.

Závěr

Cílem této diplomové práce bylo čtenáře seznámit jak se základními, tak s pokročilými nebo méně známými metodami klasifikace dat. Velmi podrobně pak tato práce popisuje metody využívající klasifikační stromy a SVM. Mám za to, že tyto ač zatím možná méně známé metody, se čtenáři mohou nejednou hodit. Byl jsem také velmi příjemně překvapen výsledky zejména binární klasifikace, kde výsledky dosahovaly téměř 80% úspěšnosti.

U každé metody bylo nutno seznámit s principy jejího fungování. Tato teoretická část je obsáhlejší zejména pak u metod klasifikačních stromů a metody SVM. Každá metoda pak zahrnovala i praktickou ukázku v softwaru R. Praktická část mi zabrala možná více času než samotné sepsání teorie, jsem za to ovšem ve výsledku velmi rád, jelikož práce měla být od začátku praktičtějšího charakteru.

Doufám, že tato práce pomůže zejména zájemcům o téma klasifikace dat pochopit jednotlivé metody a zároveň může posloužit jako návod pro implementaci metod v softwaru R.

Příloha

Statistiky hráčů nastupujících v základní části MLB v sezónách 2015 a 2016.

Player	Year	Pos	G	AB	2B	HR	H	RBI	AVG	OBP
Machado, M	2015	COR	162	633	30	35	181	86	0,286	0,359
Seager, K	2015	COR	161	623	37	26	166	74	0,266	0,328
Andrus, E	2015	INF	160	596	34	7	154	62	0,258	0,309
Davis, C	2015	COR	160	573	31	47	150	117	0,262	0,361
Longoria, E	2015	COR	160	604	35	21	163	73	0,27	0,328
Rizzo, A	2015	COR	160	586	38	31	163	101	0,278	0,387
Calhoun, K	2015	OF	159	630	23	26	161	83	0,256	0,308
Cespedes, Y	2015	OF	159	633	42	36	184	105	0,291	0,328
Goldschmidt, P	2015	COR	159	567	38	33	182	110	0,321	0,435
Pillar, K	2015	OF	159	586	31	12	163	56	0,278	0,314
Trout, M	2015	OF	159	575	32	41	172	90	0,299	0,402
Cabrera, M	2015	OF	158	629	36	12	172	77	0,273	0,314
Donaldson, J	2015	COR	158	620	41	41	184	123	0,297	0,371
Fielder, P	2015	COR	158	613	28	23	187	98	0,305	0,378
Hosmer, E	2015	COR	158	599	33	18	178	93	0,297	0,363
Martinez, J	2015	OF	158	596	33	38	168	102	0,282	0,344
Mauer, J	2015	COR	158	592	34	10	157	66	0,265	0,338
Votto, J	2015	COR	158	545	33	29	171	80	0,314	0,459
Arenado, N	2015	COR	157	616	43	42	177	130	0,287	0,323
Blackmon, C	2015	OF	157	614	31	17	176	58	0,287	0,347
Bruce, J	2015	OF	157	580	35	26	131	87	0,226	0,294
Dozier, B	2015	INF	157	628	39	28	148	77	0,236	0,307
Frazier, T	2015	COR	157	619	43	35	158	89	0,255	0,309
Granderson, C	2015	OF	157	580	33	26	150	70	0,259	0,364
McCutchen, A	2015	OF	157	566	36	23	165	96	0,292	0,401
Pollock, A	2015	OF	157	609	39	20	192	76	0,315	0,367
Pujols, A	2015	COR	157	602	22	40	147	95	0,244	0,307

Player	Year	Pos	G	AB	2B	HR	H	RBI	AVG	OBP
Aybar, E	2015	INF	156	597	30	3	161	44	0,27	0,301
Bogaerts, X	2015	INF	156	613	35	7	196	81	0,32	0,355
Cano, R	2015	INF	156	624	34	21	179	79	0,287	0,334
Desmond, I	2015	INF	156	583	27	19	136	62	0,233	0,29
Fowler, D	2015	OF	156	596	29	17	149	46	0,25	0,346
Gonzales, A	2015	COR	156	571	33	28	157	90	0,275	0,35
Headley, C	2015	COR	156	580	29	11	150	62	0,259	0,324
Markakis, N	2015	OF	156	612	38	3	181	53	0,296	0,37
Gregorius, D	2015	INF	155	525	24	9	139	56	0,265	0,318
Parra, G	2015	OF	155	547	36	14	159	51	0,291	0,328
Peralta, J	2015	INF	155	579	26	17	159	71	0,275	0,334
Semien, M	2015	INF	155	556	23	15	143	45	0,257	0,31
Abreu, J	2015	COR	154	613	34	30	178	101	0,29	0,347
Altuve, J	2015	INF	154	638	40	15	200	66	0,313	0,353
Carpenter, M	2015	COR	154	574	44	28	156	84	0,272	0,365
Castellanos, N	2015	COR	154	549	33	15	140	73	0,255	0,303
Heyward, J	2015	OF	154	547	33	13	160	60	0,293	0,359
Kemp, M	2015	OF	154	596	31	23	158	100	0,265	0,312
Kinsler, I	2015	INF	154	624	35	11	185	73	0,296	0,342
Ramirez, A	2015	INF	154	583	33	10	145	62	0,249	0,285
Santana, C	2015	COR	154	550	29	19	127	85	0,231	0,357
Bautista, J	2015	OF	153	543	29	40	136	114	0,25	0,377
Eaton, A	2015	OF	153	610	28	14	175	56	0,287	0,361
Forsythe, L	2015	INF	153	540	33	17	152	68	0,281	0,359
Gonzales, C	2015	OF	153	554	25	40	150	97	0,271	0,325
Harper, B	2015	OF	153	521	38	42	172	99	0,33	0,46
Marte, S	2015	OF	153	580	30	19	167	81	0,288	0,337
Polanco, G	2015	OF	153	593	35	9	152	52	0,256	0,32
Suzuki, I	2015	OF	153	398	5	1	91	21	0,229	0,282
Cruz, N	2015	OF	152	590	22	44	178	93	0,302	0,369
Peterson, J	2015	INF	152	528	23	6	126	52	0,239	0,314
Plouffe, T	2015	COR	152	573	35	22	140	86	0,244	0,307
Revere, B	2015	OF	152	592	22	2	181	45	0,306	0,342
Solarte, Y	2015	COR	152	526	33	14	142	63	0,27	0,32
Bryant, K	2015	COR	151	559	30	26	153	99	0,274	0,368
Castro, S	2015	INF	151	547	24	11	146	68	0,267	0,298
Galvis, F	2015	INF	151	559	14	7	147	50	0,263	0,302
Gardner, B	2015	OF	151	571	26	16	148	66	0,259	0,343
Kiermaier, K	2015	OF	151	505	25	10	133	40	0,263	0,298

Player	Year	Pos	G	AB	2B	HR	H	RBI	AVG	OBP
Pederson, J	2015	OF	151	480	19	26	101	54	0,21	0,346
Walker, N	2015	INF	151	543	32	16	145	71	0,267	0,327
Alvarez, P	2015	COR	150	437	18	27	106	76	0,243	0,32
LeMahieu, D	2015	INF	150	564	21	6	170	61	0,301	0,358
Posey, B	2015	INF	150	557	28	19	177	95	0,318	0,379
Upton, J	2015	OF	150	542	26	26	136	81	0,251	0,336
Wong, K	2015	INF	150	557	28	11	146	61	0,262	0,321
Choo, S	2015	OF	149	555	32	22	153	82	0,276	0,375
Duffy, M	2015	COR	149	573	28	12	169	77	0,295	0,334
Lawrie, B	2015	COR	149	562	29	16	146	60	0,26	0,299
Lind, A	2015	COR	149	502	32	20	139	87	0,277	0,36
Peralta, D	2015	OF	149	462	26	17	144	78	0,312	0,371
Reddick, J	2015	OF	149	526	25	20	143	77	0,272	0,333
Coghlan, C	2015	OF	148	440	25	16	110	41	0,25	0,341
Escobar, A	2015	INF	148	612	20	3	157	47	0,257	0,293
Garcia, A	2015	OF	148	553	17	13	142	59	0,257	0,309
Phillips, B	2015	INF	148	588	19	12	173	70	0,294	0,328
Herrera, O	2015	OF	147	495	30	8	147	41	0,297	0,344
Moustakes, M	2015	COR	147	549	34	22	156	82	0,284	0,348
Norris, D	2015	INF	147	515	33	14	129	62	0,25	0,305
Owings, C	2015	INF	147	515	27	4	117	43	0,227	0,264
Simmons, A	2015	INF	147	535	23	4	142	44	0,265	0,321
Encarnacio, E	2015	COR	146	528	31	39	146	111	0,277	0,372
Morrison, L	2015	COR	146	457	15	17	103	54	0,225	0,302
Betts, M	2015	OF	145	597	42	18	174	77	0,291	0,341
Gordon, D	2015	INF	145	615	24	4	205	46	0,333	0,359
Moss, B	2015	COR	145	469	24	19	106	58	0,226	0,304
Miller, B	2015	INF	144	438	22	11	113	46	0,258	0,329
Rollins, J	2015	INF	144	517	24	13	116	41	0,224	0,285
Beltre, A	2015	COR	143	567	32	18	163	83	0,287	0,334
Cabrera, A	2015	INF	143	505	28	15	134	58	0,265	0,315
Crawford, B	2015	INF	143	507	33	21	130	84	0,256	0,321
Lagares, J	2015	OF	143	441	16	6	114	41	0,259	0,289
Ethier, A	2015	OF	142	395	20	14	116	53	0,294	0,366
Perez, S	2015	INF	142	531	25	21	138	70	0,26	0,28
Russel, A	2015	INF	142	475	29	13	115	54	0,242	0,307
Segura, J	2015	INF	142	560	16	6	144	50	0,257	0,281
Trumbo, M	2015	OF	142	508	23	22	133	64	0,262	0,31
Bourn, M	2015	OF	141	425	15	0	101	30	0,238	0,31

Player	Year	Pos	G	AB	2B	HR	H	RBI	AVG	OBP
Kipnis, J	2015	INF	141	565	43	9	171	52	0,303	0,372
Maybin, C	2015	OF	141	505	18	10	135	59	0,267	0,327
Braun, R	2015	OF	140	506	27	25	144	84	0,285	0,356
Cain, L	2015	OF	140	551	34	16	169	72	0,307	0,361
Gose, A	2015	OF	140	485	24	5	123	26	0,254	0,321
Reynolds, M	2015	COR	140	382	21	13	88	48	0,23	0,315
Young, C	2015	OF	140	318	20	14	80	42	0,252	0,32
Escobar, Y	2015	COR	139	535	25	9	168	56	0,314	0,375
Hunter, T	2015	OF	139	521	22	22	125	81	0,24	0,293
Rodriguez, S	2015	COR	139	224	12	4	55	17	0,246	0,281
Taylor, M	2015	OF	138	472	15	14	108	63	0,229	0,282
Belt, B	2015	COR	137	492	33	18	138	68	0,28	0,356
Brantley, M	2015	OF	137	529	45	15	164	84	0,31	0,379
Flores, W	2015	INF	137	483	22	16	127	59	0,263	0,295
Jones, A	2015	OF	137	546	25	27	147	82	0,269	0,308
Ramirez, Ar	2015	COR	137	475	31	17	117	75	0,246	0,297
Rasmus, C	2015	OF	137	432	23	25	103	61	0,238	0,314
Jackson, A	2015	OF	136	491	25	9	131	48	0,267	0,311
Molina, Y	2015	INF	136	488	23	4	132	61	0,27	0,31
Smith, S	2015	OF	136	395	31	12	98	42	0,248	0,33
Vogt, S	2015	INF	136	445	21	18	116	71	0,261	0,341
Byrd, M	2015	OF	135	506	25	23	125	73	0,247	0,29
Duda, L	2015	COR	135	471	33	27	115	73	0,244	0,352
McCann, B	2015	INF	135	465	15	26	108	94	0,232	0,32
Venable, W	2015	OF	135	349	13	6	85	33	0,244	0,32
Ahmed, N	2015	INF	134	421	17	9	95	34	0,226	0,275
Beltran, C	2015	OF	133	478	34	19	132	67	0,276	0,337
Marisnick, J	2015	OF	133	339	15	9	80	36	0,236	0,281
Napoli, M	2015	COR	133	407	20	18	91	50	0,224	0,324
Pagan, A	2015	OF	133	512	21	3	134	37	0,262	0,303
Inciarte, E	2015	OF	132	524	27	6	159	45	0,303	0,338
Moreland, M	2015	COR	132	471	27	23	131	85	0,278	0,33
Murphy, D	2015	OF	132	361	18	10	102	50	0,283	0,318
Smoak, J	2015	COR	132	296	16	18	67	59	0,226	0,299
Valbuena, L	2015	COR	132	434	18	25	97	56	0,224	0,31
Drew, S	2015	INF	131	383	16	17	77	44	0,201	0,271
Schumaker, S	2015	OF	131	244	20	1	59	21	0,242	0,306
Suzuki, K	2015	INF	131	433	17	5	104	50	0,24	0,296
Cervelli, F	2015	INF	130	450	17	7	133	44	0,296	0,37

Player	Year	Pos	G	AB	2B	HR	H	RBI	AVG	OBP
Hechavarria, A	2015	INF	130	470	17	5	132	48	0,281	0,315
Murphy, Dan	2015	INF	130	499	38	14	140	73	0,281	0,322
Asche, C	2015	OF	129	425	22	12	104	39	0,245	0,294
Bour, J	2015	COR	129	409	20	23	107	73	0,262	0,321
Carter, C	2015	COR	129	391	17	24	78	64	0,199	0,307
Giavotella, J	2015	INF	129	453	25	4	123	49	0,272	0,318
Holt, B	2015	INF	129	454	27	2	127	45	0,28	0,349
Howard, R	2015	COR	129	467	29	23	107	77	0,229	0,277
Martin, R	2015	INF	129	441	23	23	106	77	0,24	0,329
Prado, M	2015	COR	129	500	22	9	144	63	0,288	0,338
Goins, R	2015	INF	128	376	16	5	94	45	0,25	0,318
Guyer, B	2015	OF	128	332	21	8	88	28	0,265	0,359
Gyorko, J	2015	INF	128	421	15	16	104	57	0,247	0,297
Ramos, W	2015	INF	128	475	16	15	109	68	0,229	0,258
Tulowitzki, T	2015	INF	128	486	27	17	136	70	0,28	0,337
Escobar, E	2015	INF	127	409	31	12	107	58	0,262	0,309
Hernandez, C	2015	INF	127	405	20	1	110	35	0,272	0,339
Kang, J	2015	COR	126	421	24	15	121	58	0,287	0,355
Realmuto, J	2015	INF	126	441	21	10	114	47	0,259	0,29
Robinson, C	2015	COR	126	309	15	10	84	34	0,272	0,358
Sandoval, P	2015	COR	126	470	25	10	115	47	0,245	0,292
Turner, J	2015	COR	126	385	26	16	113	60	0,294	0,37
Yelich, C	2015	OF	126	476	30	7	143	44	0,3	0,366
Zobrist, C	2015	INF	126	467	36	13	129	56	0,276	0,359
Burns, B	2015	OF	125	520	18	5	153	42	0,294	0,334
Canha, M	2015	OF	124	441	22	16	112	70	0,254	0,315
Infante, O	2015	INF	124	440	23	2	97	44	0,22	0,234
Ozuna, M	2015	OF	123	459	27	10	119	44	0,259	0,308
Rosario, E	2015	OF	122	453	18	13	121	50	0,267	0,289
Davis, K	2015	OF	121	392	16	27	97	66	0,247	0,323
DeSchields, D	2015	OF	121	425	22	2	111	37	0,261	0,344
Freese, D	2015	COR	121	424	27	14	109	56	0,257	0,323
Fuld, S	2015	OF	120	290	16	2	57	22	0,197	0,276
Gonzales, M	2015	INF	120	344	18	12	96	34	0,279	0,317
Iglesias, J	2015	INF	120	416	17	2	125	23	0,3	0,347
Ordor, R	2015	INF	120	426	21	16	111	61	0,261	0,316
Sanchez, C	2015	INF	120	389	23	5	87	31	0,224	0,268
Sogard, E	2015	INF	120	372	12	1	92	37	0,247	0,294
Cabrera, M	2015	COR	119	429	28	18	145	76	0,338	0,44

Player	Year	Pos	G	AB	2B	HR	H	RBI	AVG	OBP
Uribe, J	2015	COR	119	360	17	14	91	43	0,253	0,32
Amarista, A	2015	INF	118	324	10	3	66	30	0,204	0,257
Espinosa, D	2015	INF	118	367	21	13	88	37	0,24	0,311
Francoeur, J	2015	OF	118	326	16	13	84	45	0,258	0,286
Freeman, F	2015	COR	118	416	27	18	115	66	0,276	0,37
Tomas, Y	2015	OF	118	406	19	9	111	48	0,273	0,305
Bourjos, P	2015	OF	117	195	8	4	39	13	0,2	0,29
Cuddyer, M	2015	OF	117	379	18	10	98	41	0,259	0,309
Kendrick, H	2015	INF	117	464	22	9	137	54	0,295	0,336
Hill, A	2015	COR	116	313	18	6	72	39	0,23	0,295
Mercer, J	2015	INF	116	394	21	3	96	34	0,244	0,293
Paulsen, B	2015	COR	116	325	19	11	90	49	0,277	0,326
Reyes, J	2015	INF	116	481	25	7	132	53	0,274	0,31
Tejada, R	2015	INF	116	360	23	3	94	28	0,261	0,338
Blanco, G	2015	OF	115	327	19	5	95	26	0,291	0,368
Gomez, C	2015	OF	115	435	29	12	111	56	0,255	0,314
Grandal, Y	2015	INF	115	355	12	16	83	47	0,234	0,353
De Aza, A	2015	OF	114	325	17	7	85	35	0,262	0,333
Gennett, S	2015	INF	114	375	18	6	99	29	0,264	0,294
Hamilton, B	2015	OF	114	412	8	4	93	28	0,226	0,274
Hardy, J	2015	INF	114	411	14	8	90	37	0,219	0,253
Harrison, J	2015	COR	114	418	29	4	120	28	0,287	0,327
McCann, J	2015	INF	114	401	18	7	106	41	0,264	0,297
Cron, C	2015	COR	113	378	17	16	99	51	0,262	0,3
Montero, M	2015	INF	113	347	11	15	86	54	0,248	0,345
Pierzinsky, A	2015	INF	113	407	24	9	122	49	0,3	0,339
Davis, R	2015	OF	112	341	16	8	88	30	0,258	0,306
DeJesus, D	2015	OF	112	288	9	5	67	30	0,233	0,297
Flowers, T	2015	INF	112	331	12	9	79	12	0,239	0,295
Perez, H	2015	COR	112	263	15	1	64	21	0,243	0,257
Zunino, M	2015	INF	112	350	11	11	61	28	0,174	0,23
Ellsbury, J	2015	OF	111	452	15	7	116	33	0,257	0,318
Johnson, K	2015	INF	111	310	11	14	82	47	0,265	0,314
Teixeira, M	2015	COR	111	392	22	31	100	79	0,255	0,357
Castillo, W	2015	INF	110	342	15	19	81	57	0,237	0,296
Rivera, R	2015	INF	110	298	14	5	53	26	0,178	0,213
Souza Jr., S	2015	OF	110	373	15	16	84	40	0,225	0,318
Ackley, D	2015	OF	108	238	11	10	55	30	0,231	0,284

Player	Year	Pos	G	AB	2B	HR	H	RBI	AVG	OBP
Pena, B	2015	INF	108	333	17	0	91	18	0,273	0,334
Spangenberg, C	2015	INF	108	303	17	4	82	21	0,271	0,333
Utley, C	2015	INF	107	373	21	8	79	39	0,212	0,286
Barnes, B	2015	OF	106	255	13	2	64	17	0,251	0,314
Blanco, A	2015	INF	106	233	22	7	68	25	0,292	0,36
Chisenhall, L	2015	OF	106	333	19	7	82	44	0,246	0,294
Guerrero, A	2015	OF	106	219	9	11	51	36	0,233	0,261
Pennington, C	2015	INF	105	210	6	3	44	21	0,21	0,298
Ramirez, H	2015	OF	105	401	12	19	100	53	0,249	0,291
Rios, A	2015	OF	105	385	22	4	98	32	0,255	0,287
Castro, J	2015	INF	104	337	19	11	71	31	0,211	0,283
Gordon, A	2015	OF	104	354	18	13	96	48	0,271	0,377
Grichuk, R	2015	OF	103	323	23	17	89	47	0,276	0,329
Hundley, N	2015	INF	103	366	21	10	110	43	0,301	0,339
Lucroy, J	2015	INF	103	371	20	7	98	43	0,264	0,326
Descalso, D	2015	INF	101	185	3	5	38	22	0,205	0,283
Joseph, C	2015	INF	100	320	16	11	75	49	0,234	0,299
Panik, J	2015	INF	100	382	27	8	119	37	0,312	0,378
Correa, C	2015	INF	99	387	22	22	108	68	0,279	0,345
Alcides Escobar	2016	INF	162	637	24	7	166	55	0,261	0,292
Jonathan Schoop	2016	INF	162	615	38	25	164	82	0,27	0,30
George Springer	2016	OF	162	644	29	29	168	82	0,261	0,359
Jose Altuve	2016	INF	161	640	42	24	216	96	0,338	0,396
Robinson Cano	2016	INF	161	655	33	39	195	103	0,298	0,35
Nolan Arenado	2016	COR	160	618	35	41	182	133	0,294	0,362
Chris Carter	2016	COR	160	549	27	41	122	94	0,222	0,321
Evan Longoria	2016	COR	160	633	41	36	173	98	0,273	0,318
Jose Abreu	2016	COR	159	624	32	25	183	100	0,293	0,353
Odubel Herrera	2016	OF	159	583	21	15	167	49	0,286	0,361
Marcus Semien	2016	INF	159	568	27	27	135	75	0,238	0,300
Eugenio Suarez	2016	COR	159	565	25	21	140	70	0,248	0,317
Mike Trout	2016	OF	159	549	32	29	173	100	0,315	0,441
Mark Trumbo	2016	OF	159	613	27	47	157	108	0,256	0,316
Mookie Betts	2016	OF	158	672	42	31	214	113	0,318	0,363
Miguel Cabrera	2016	COR	158	595	31	38	188	108	0,316	0,393
Todd Frazier	2016	COR	158	590	21	40	133	98	0,225	0,302
Freddie Freeman	2016	COR	158	589	43	34	178	91	0,302	0,400
Freddy Galvis	2016	INF	158	584	26	20	141	67	0,241	0,274
Paul Goldschmidt	2016	COR	158	579	33	24	172	95	0,297	0,411

Player	Year	Pos	G	AB	2B	HR	H	RBI	AVG	OBP
Eric Hosmer	2016	COR	158	605	24	25	161	104	0,266	0,328
Francisco Lindor	2016	INF	158	604	30	15	182	78	0,301	0,358
Nick Markakis	2016	OF	158	599	38	13	161	89	0,269	0,346
Carlos Santana	2016	COR	158	582	31	34	151	87	0,259	0,366
Kyle Seager	2016	COR	158	597	36	30	166	99	0,278	0,359
Joey Votto	2016	COR	158	556	34	29	181	97	0,326	0,434
Xander Bogaerts	2016	INF	157	652	34	21	192	89	0,294	0,356
Kole Calhoun	2016	OF	157	594	35	18	161	75	0,271	0,348
Chris Davis	2016	COR	157	566	21	38	125	84	0,221	0,332
Adam Eaton	2016	OF	157	619	29	14	176	59	0,284	0,362
Danny Espinosa	2016	INF	157	516	15	24	108	72	0,209	0,306
Manny Machado	2016	COR	157	640	40	37	188	96	0,294	0,343
Wil Myers	2016	COR	157	599	29	28	155	94	0,259	0,336
Corey Seager	2016	INF	157	627	40	26	193	72	0,308	0,365
Yonder Alonso	2016	COR	156	482	34	7	122	56	0,253	0,316
Brandon Belt	2016	COR	156	542	41	17	149	82	0,275	0,394
Jackie Bradley	2016	OF	156	558	30	26	149	87	0,267	0,349
Ian Desmond	2016	OF	156	625	29	22	178	86	0,285	0,335
Adrian Gonzalez	2016	COR	156	568	31	18	162	90	0,285	0,349
Matt Kemp	2016	OF	156	623	39	35	167	108	0,268	0,304
Jason Kipnis	2016	INF	156	610	41	23	168	82	0,275	0,343
Anthony Rendon	2016	COR	156	567	38	20	153	85	0,270	0,348
Jonathan Villar	2016	INF	156	589	38	19	168	63	0,285	0,369
Kris Bryant	2016	COR	155	603	35	39	176	102	0,292	0,385
Brandon Crawford	2016	INF	155	553	28	12	152	84	0,275	0,342
Josh Donaldson	2016	COR	155	577	32	37	164	99	0,284	0,404
Brian Dozier	2016	INF	155	615	35	42	165	99	0,268	0,340
Adeiny Hechavarria	2016	INF	155	508	17	3	120	38	0,236	0,283
Cesar Hernandez	2016	INF	155	547	14	6	161	39	0,294	0,371
Anthony Rizzo	2016	COR	155	583	43	32	170	109	0,292	0,385
Christian Yelich	2016	OF	155	578	38	21	172	98	0,298	0,376
Dustin Pedroia	2016	INF	154	633	36	15	201	74	0,318	0,376
Adrian Beltre	2016	COR	153	583	31	32	175	104	0,300	0,358
Carlos Correa	2016	INF	153	577	36	20	158	96	0,274	0,361
Didi Gregorius	2016	INF	153	562	32	20	155	70	0,276	0,304
Ian Kinsler	2016	INF	153	618	29	28	178	83	0,288	0,348
Andrew McCutchen	2016	OF	153	598	26	24	153	79	0,256	0,336
Stephen Piscotty	2016	OF	153	582	35	22	159	85	0,273	0,343
Martin Prado	2016	COR	153	600	37	8	183	75	0,305	0,359
Jean Segura	2016	INF	153	637	41	20	203	64	0,319	0,368

Player	Year	Pos	G	AB	2B	HR	H	RBI	AVG	OBP
Justin Upton	2016	OF	153	570	28	31	140	87	0,246	0,310
Maikel Franco	2016	COR	152	581	23	25	148	88	0,255	0,306
Adam Jones	2016	OF	152	619	19	29	164	83	0,265	0,310
Brad Miller	2016	INF	152	548	29	30	133	81	0,243	0,304
Jose Ramirez	2016	COR	152	565	46	11	176	76	0,312	0,363
Melky Cabrera	2016	OF	151	591	42	14	175	86	0,296	0,345
Starlin Castro	2016	INF	151	577	29	21	156	70	0,270	0,300
Jake Lamb	2016	COR	151	523	31	29	130	91	0,249	0,332
Addison Russell	2016	INF	151	525	25	21	125	95	0,238	0,321
Justin Turner	2016	COR	151	556	34	27	153	90	0,275	0,339
Khris Davis	2016	OF	150	555	24	42	137	102	0,247	0,307
Adam Duvall	2016	OF	150	552	31	33	133	103	0,241	0,297
Carlos Gonzalez	2016	OF	150	584	42	25	174	100	0,298	0,350
Curtis Granderson	2016	OF	150	545	24	30	129	59	0,237	0,335
Mike Napoli	2016	COR	150	557	22	34	133	101	0,239	0,335
Rougned Odor	2016	INF	150	605	33	33	164	88	0,271	0,296
Jordy Mercer	2016	INF	149	519	22	11	133	59	0,256	0,328
Melvin Upton	2016	OF	149	492	15	20	117	61	0,238	0,291
Corey Dickerson	2016	OF	148	510	36	24	125	70	0,245	0,293
Jacoby Ellsbury	2016	OF	148	551	24	9	145	56	0,263	0,330
Brett Gardner	2016	OF	148	547	22	7	143	41	0,261	0,351
Marcell Ozuna	2016	OF	148	557	23	23	148	76	0,266	0,321
Elvis Andrus	2016	INF	147	506	31	8	153	69	0,302	0,362
Jay Bruce	2016	OF	147	539	27	33	135	99	0,250	0,309
Bryce Harper	2016	OF	147	506	24	24	123	86	0,243	0,373
Yadier Molina	2016	INF	147	534	38	8	164	58	0,307	0,360
Mitch Moreland	2016	COR	147	460	21	22	107	60	0,233	0,298
Ben Zobrist	2016	INF	147	523	31	18	142	76	0,272	0,386
Howie Kendrick	2016	OF	146	487	26	8	124	40	0,255	0,326
DJ LeMahieu	2016	INF	146	552	32	11	192	66	0,348	0,416
Kevin Pillar	2016	OF	146	548	35	7	146	53	0,266	0,303
Buster Posey	2016	INF	146	539	33	14	155	80	0,288	0,362
Nomar Mazara	2016	OF	145	516	13	20	137	64	0,266	0,320
Alexei Ramirez	2016	INF	145	478	22	6	115	48	0,241	0,277
Travis Shaw	2016	COR	145	480	34	16	116	71	0,242	0,306
Gregory Polanco	2016	OF	144	527	34	22	136	86	0,258	0,323
Charlie Blackmon	2016	OF	143	578	35	29	187	82	0,324	0,381
Leonys Martin	2016	OF	143	518	17	15	128	47	0,247	0,306
Denard Span	2016	OF	143	572	23	11	152	53	0,266	0,331
Ichiro Suzuki	2016	OF	143	327	15	1	95	22	0,291	0,354

Player	Year	Pos	G	AB	2B	HR	H	RBI	AVG	OBP
Jayson Werth	2016	OF	143	525	28	21	128	69	0,244	0,335
Javier Baez	2016	INF	142	421	19	14	115	59	0,273	0,314
Jason Heyward	2016	OF	142	530	27	7	122	49	0,230	0,306
Jonathan Lucroy	2016	INF	142	490	24	24	143	81	0,292	0,355
Daniel Murphy	2016	INF	142	531	47	25	184	104	0,347	0,390
Asdrubal Cabrera	2016	INF	141	521	30	23	146	62	0,280	0,336
David Freese	2016	COR	141	437	23	13	118	55	0,270	0,352
Marwin Gonzalez	2016	OF	141	484	26	13	123	51	0,254	0,293
Eduardo Nunez	2016	COR	141	553	24	16	159	67	0,288	0,325
Brandon Phillips	2016	INF	141	550	34	11	160	64	0,291	0,320
Chase Headley	2016	COR	140	467	18	14	118	51	0,253	0,331
Matthew Joyce	2016	OF	140	231	10	13	56	42	0,242	0,403
Sean Rodriguez	2016	COR	140	300	16	18	81	56	0,270	0,349
Michael Saunders	2016	OF	140	490	32	24	124	57	0,253	0,338
Yasmany Tomas	2016	OF	140	530	30	31	144	83	0,272	0,313
Salvador Perez	2016	INF	139	514	28	22	127	64	0,247	0,288
Chase Utley	2016	INF	138	512	26	14	129	52	0,252	0,319
Jose Iglesias	2016	INF	137	467	26	4	119	32	0,255	0,306
Russell Martin	2016	INF	137	455	16	20	105	74	0,231	0,335
Joc Pederson	2016	OF	137	406	26	25	100	68	0,246	0,352
J.T. Realmuto	2016	INF	137	509	31	11	154	48	0,303	0,343
Seth Smith	2016	OF	137	378	15	16	94	63	0,249	0,342
Stephen Vogt	2016	INF	137	490	30	14	123	56	0,251	0,305
Scooter Gennett	2016	INF	136	498	30	14	131	56	0,263	0,317
Ryan Braun	2016	OF	135	511	23	30	156	91	0,305	0,365
Rajai Davis	2016	OF	134	454	23	12	113	48	0,249	0,306
Brandon Drury	2016	INF	134	461	31	16	130	53	0,282	0,329
Adonis Garcia	2016	COR	134	532	29	14	145	65	0,273	0,311
Joe Mauer	2016	COR	134	494	22	11	129	49	0,261	0,363
Yoenis Cespedes	2016	OF	132	479	25	31	134	86	0,280	0,354
Yunel Escobar	2016	COR	132	517	28	5	157	39	0,304	0,355
Randal Grichuk	2016	OF	132	446	29	24	107	68	0,240	0,289
John Jaso	2016	OF	132	380	25	8	102	42	0,268	0,353
Josh Harrison	2016	INF	131	487	25	4	138	59	0,283	0,311
Ender Inciarte	2016	OF	131	522	24	3	152	29	0,291	0,351
Travis Jankowski	2016	OF	131	335	13	2	82	12	0,245	0,332
Kelly Johnson	2016	COR	131	304	14	10	75	34	0,247	0,306
Wilson Ramos	2016	INF	131	482	25	22	148	80	0,307	0,354
Troy Tulowitzki	2016	INF	131	492	21	24	125	79	0,254	0,318
Alejandro De Aza	2016	OF	130	234	9	6	48	25	0,205	0,297
Brian McCann	2016	INF	130	429	13	20	104	58	0,242	0,335

Player	Year	Pos	G	AB	2B	HR	H	RBI	AVG	OBP
Danny Valencia	2016	COR	130	471	22	17	135	51	0,287	0,346
Matt Carpenter	2016	COR	129	473	36	21	128	68	0,271	0,380
Starling Marte	2016	OF	129	489	34	9	152	46	0,311	0,362
Angel Pagan	2016	OF	129	495	24	12	137	55	0,277	0,331
Cheslor Cuthbert	2016	COR	128	475	28	12	130	46	0,274	0,318
Derek Dietrich	2016	COR	128	351	20	7	98	42	0,279	0,374
Evan Gattis	2016	INF	128	447	19	32	112	72	0,251	0,319
Alex Gordon	2016	OF	128	445	16	17	98	40	0,220	0,312
Jedd Gyorko	2016	COR	128	400	9	30	97	59	0,243	0,306
Brandon Moss	2016	COR	128	413	19	28	93	67	0,225	0,300
Paulo Orlando	2016	OF	128	457	24	5	138	43	0,302	0,329
Logan Forsythe	2016	INF	127	511	24	20	135	52	0,264	0,333
Joe Panik	2016	INF	127	464	21	10	111	62	0,239	0,315
Erick Aybar	2016	INF	126	415	19	3	101	34	0,243	0,303
Lonnie Chisenhall	2016	OF	126	385	25	8	110	57	0,286	0,328
Yasmani Grandal	2016	INF	126	390	14	27	89	72	0,228	0,339
Adam Lind	2016	COR	126	401	17	20	96	58	0,239	0,286
Justin Smoak	2016	COR	126	299	10	14	65	34	0,217	0,314
Dexter Fowler	2016	OF	125	456	25	13	126	48	0,276	0,393
Jeff Francoeur	2016	OF	125	307	15	7	78	34	0,254	0,297
Aaron Hill	2016	COR	125	378	14	10	99	38	0,262	0,336
Kirk Nieuwenhuis	2016	OF	125	335	18	13	70	44	0,209	0,324
Derek Norris	2016	INF	125	415	17	14	77	42	0,186	0,255
Andrelton Simmons	2016	INF	124	448	22	4	126	44	0,281	0,324
Matt Wieters	2016	INF	124	423	17	17	103	66	0,243	0,302
Peter Bourjos	2016	OF	123	355	20	5	89	23	0,251	0,292
Aaron Hicks	2016	OF	123	327	13	8	71	31	0,217	0,281
Hernan Perez	2016	OF	123	404	18	13	110	56	0,272	0,302
Miguel Rojas	2016	COR	123	194	12	1	48	14	0,247	0,288
Coco Crisp	2016	COR	122	446	27	13	103	55	0,231	0,302
Philip Gosselin	2016	INF	122	220	12	2	61	13	0,277	0,324
Cristhian Adames	2016	INF	121	225	7	2	49	17	0,218	0,304
Zack Cozart	2016	INF	121	464	28	16	117	50	0,252	0,308
Kolten Wong	2016	INF	121	313	7	5	75	23	0,240	0,327
Avisail Garcia	2016	OF	120	413	18	12	101	51	0,245	0,307
J.D. Martinez	2016	OF	120	460	35	22	141	68	0,307	0,373
Steven Souza	2016	OF	120	430	17	17	106	49	0,247	0,303
Billy Hamilton	2016	OF	119	411	19	3	107	17	0,260	0,321
Ketel Marte	2016	INF	119	437	21	1	113	33	0,259	0,287
Chris Owings	2016	INF	119	437	24	5	121	49	0,277	0,315
Giancarlo Stanton	2016	OF	119	413	20	27	99	74	0,240	0,326

Player	Year	Pos	G	AB	2B	HR	H	RBI	AVG	OBP
Brett Wallace	2016	COR	119	217	10	6	41	20	0,189	0,309
Matt Adams	2016	OF	118	297	18	16	74	54	0,249	0,309
Norichika Aoki	2016	OF	118	417	24	4	118	28	0,283	0,349
Carlos Gomez	2016	OF	118	411	22	13	95	53	0,231	0,298
Jake Marisnick	2016	OF	118	287	18	5	60	21	0,209	0,257
Mark Reynolds	2016	COR	118	393	24	14	111	53	0,282	0,356
Jose Bautista	2016	OF	116	423	24	22	99	69	0,234	0,366
C.J. Cron	2016	COR	116	407	25	16	113	69	0,278	0,325
Tyler Naquin	2016	OF	116	321	18	14	95	43	0,296	0,372
Miguel Sano	2016	COR	116	437	22	25	103	66	0,236	0,319
Mark Teixeira	2016	COR	116	387	16	15	79	44	0,204	0,292
Tucker Barnhart	2016	INF	115	377	23	7	97	51	0,257	0,323
J.J. Hardy	2016	INF	115	405	29	9	109	48	0,269	0,309
Jace Peterson	2016	OF	115	350	16	7	89	29	0,254	0,350
Josh Reddick	2016	OF	115	398	17	10	112	37	0,281	0,345
Ryan Zimmerman	2016	COR	115	427	18	15	93	46	0,218	0,272
Jeremy Hazelbaker	2016	OF	114	200	7	12	47	28	0,235	0,295
Michael Bourn	2016	OF	113	375	13	5	99	38	0,264	0,314
Welington Castillo	2016	INF	113	416	24	14	110	68	0,264	0,322
Jason Castro	2016	INF	113	329	16	11	69	32	0,210	0,307
Chris Johnson	2016	COR	113	243	11	5	54	24	0,222	0,281
Max Kepler	2016	OF	113	396	20	17	93	63	0,235	0,309
Ryan Raburn	2016	OF	113	223	10	9	49	30	0,220	0,309
Neil Walker	2016	INF	113	412	9	23	116	55	0,282	0,347
Ryan Howard	2016	COR	112	331	10	25	65	59	0,196	0,257
Aledmys Diaz	2016	INF	111	404	28	17	121	65	0,300	0,369
Ezequiel Carrera	2016	OF	110	270	9	6	67	23	0,248	0,323
Nicholas Castellanos	2016	OF	110	411	25	18	117	58	0,285	0,331
Matt Holliday	2016	OF	110	382	20	20	94	62	0,246	0,322
Michael Conforto	2016	OF	109	304	21	12	67	42	0,220	0,310
Enrique Hernandez	2016	OF	109	216	8	7	41	18	0,190	0,283
Yangervis Solarte	2016	INF	109	405	26	15	116	71	0,286	0,341
Andrew Romine	2016	INF	108	174	5	2	41	16	0,236	0,304
Rickie Weeks	2016	OF	108	180	9	9	43	27	0,239	0,327
Jarrod Dyson	2016	OF	107	299	14	1	83	25	0,278	0,340
Tommy Joseph	2016	COR	107	315	15	21	81	47	0,257	0,308
Colby Rasmus	2016	OF	107	369	10	15	76	54	0,206	0,286
Matt Szczur	2016	OF	107	185	9	5	48	24	0,259	0,312
Gregor Blanco	2016	OF	106	241	10	1	54	18	0,224	0,309
Kurt Suzuki	2016	INF	106	345	24	8	89	49	0,258	0,301
James McCann	2016	INF	105	344	9	12	76	48	0,221	0,272

Player	Year	Pos	G	AB	2B	HR	H	RBI	AVG	OBP
Trevor Story	2016	INF	97	372	21	27	101	72	0,272	0,341
Brett Lawrie	2016	INF	94	351	22	12	87	36	0,248	0,310
Tyler Saladino	2016	INF	93	298	14	8	84	38	0,282	0,315
Jarrod Saltalamacchia	2016	INF	92	246	5	12	42	38	0,171	0,284
Matt Duffy	2016	INF	91	333	14	5	86	28	0,258	0,310
Nick Ahmed	2016	INF	90	284	9	4	62	20	0,218	0,265
Andres Blanco	2016	INF	90	190	15	4	48	21	0,253	0,316
Shawn O'Malley	2016	INF	89	210	9	2	48	17	0,229	0,299
Gregorio Petit	2016	INF	89	204	13	2	50	17	0,245	0,299
Cameron Rupp	2016	INF	105	389	26	16	98	54	0,252	0,303
Darwin Barney	2016	INF	104	279	13	4	75	19	0,269	0,322
Ivan De Jesus	2016	INF	104	221	10	1	56	20	0,253	0,311
Francisco Cervelli	2016	INF	101	326	14	1	86	33	0,264	0,377
Dioner Navarro	2016	INF	101	304	13	6	63	35	0,207	0,265
Devon Travis	2016	INF	101	410	28	11	123	50	0,300	0,332
Tim Anderson	2016	INF	99	410	22	9	116	30	0,283	0,306
Johnny Giavotella	2016	INF	99	346	20	6	90	31	0,260	0,287

Literatura

[1] *Baseball stats* [online]. [cit. 2018-07-07]. Dostupné z: mlb.mlb.com/stats/sortable.jsp.

[2] Breiman, L., Friedman, J.H., Olshen, R., Stone, C.J., *Classification and Regression Trees*. Wadsworth & Brooks/Cole Advanced Books & Software, Pacific California, 1984.

[3] Brodinová, Š.: *Diskriminační analýza pro kompoziční data*. Bakalářská práce, Přírodovědecká fakulta UP, Olomouc, 2012.

[4] James, G., Witten, D., Hastie, T., Tibshirani, R.: *An Introduction to Statistical Learning*. Springer, New York, 2013.

[5] James, G., Witten, D., Hastie, T., Tibshirani, R.: *Package 'ISLR'* [online]. 2017, [cit. 2018-10-26]. dostupné z: https://cran.r-project.org/web/packages/ISLR/ISLR.pdf.

[6] Liaw, A.: *Package 'randomForest'* [online]. 2018, [cit. 2018-11-14]. dostupné z: https://cran.r-project.org/web/packages/randomForest/randomForest.pdf.

[7] Filzmoser, P.: *Klassifikation und Diskriminanzanalyse*. TU Wien, Wien, 2009.

[8] Holčík, J.: *Analýza a klasifikace dat*. Masarykova Univerzita, Brno, 2012.

[9] Hron, K., Kunderová, P.: *Základy počtu pravděpodobnosti a metod matematické statistiky*. Univerzita Palackého v Olomouci, Olomouc, 2013.

[10] Meyer, D.: *Package 'e1071'* [online]. 2018, [cit. 2018-12-08]. dostupné z: https://cran.r-project.org/web/packages/e1071/e1071.pdf.

[11] Ripley, B.: *Package 'MASS'* [online]. 2018, [cit. 2018-08-17]. dostupné z: https://cran.r-project.org/web/packages/MASS/MASS.pdf.

[12] Ripley, B.: *Package 'class'* [online]. 2015, [cit. 2018-09-22]. dostupné z: https://cran.r-project.org/web/packages/class/class.pdf.

[13] Ripley, B.: *Package 'tree'* [online]. 2018, [cit. 2018-11-14]. dostupné z: https://cran.r-project.org/web/packages/tree/tree.pdf.

[14] Süss, V.: *Softball a baseball*. Grada, Praha, 2003.

[15] Tuszynski, J.: *Package 'caTools'* [online]. 2018, [cit. 2018-11-11]. dostupné z: https://cran.r-project.org/web/packages/caTools/caTools.pdf.

[16] Václavek, T.: *Vizualizace mnohorozměrných dat s R*. Bakalářská práce, Přírodovědecká fakulta UP, Olomouc, 2017.

[17] Varmuza, K., Filzmoser, P.: *Introduction to multivariate statistical analysis in chemometrics*. CRC Press, New York, 2009.