# Spoznavanje s podatki o biološkem odgovoru na učinkovine

Gregor Majcen (63070199)

26. april 2012

## 1 Uvod

Šesta domača naloga je namenjena preučevanju podatkov, ki so s področja kemoinformatike in jih bomo potrebovali za naslednje tekmovanje. Cilj je napoved biološkega odgovora na učinkovino. Tokrat imamo binarni razred, napovedati pa moramo s kolikšno verjetnostjo trdimo pozitivno napoved.

## 2 Podatki in opis problemske domene

Učna množica (podatki) vsebuje **1776 atributov**, **3751 primerov** in **1 razred**. Kot je že napisano v uvodu je **razred binaren**, **atributi** so pa **mešani** in **normalizirani** (od vključno 0 do vključno 1). Če pogledamo vsak atribut posebej in preštejemo različne vrednosti atributa, lahko s pomočjo tega na grobo postavimo mejo med diskretnim in zveznim atributom. Izbrana **meja, ki loči tip je 10** in rezultat tega je **1357 diskretnih atributov** ter **419 zveznih**.

Učno množico imamo sedaj tipizirano, sedaj pa poglejmo še splošne opazke. Opazil sem, da je matrika zopet kar **redka** in sicer je le **16% neničelnih elementov**. Kljub temu je **28 atributov brez ničelnih elementov**. Ti popolnoma polni atributi so vsi zvezni. Seveda pa obstajajo tudi skoraj prazni atributi in če zopet postavimo mejo 10, imamo **46 atributov, ki imajo 10 ali manj neničelnih vrednosti**. Najbolj prazni so trije atributi, ki vsebujejo **le eno neničelno vrednost** in sicer atribut številka **71**, **882** in **857**.

Vsi ti podatki so implementirani v datoteki *main.py*.

## 3 Informativnost atributov

Informativnost atributov sem preveril z dvema testoma, to sta **ReliefF** in **InformationGain**.

ReliefF je eden izmed ocenjevalcev atributov, ki zna delati z zveznimi atributi in tudi zna najti odvisnost med atributi. Ta zadnja lastnost velja večinoma za malo število atributov, pri veliki količini pa to lastnost deloma izgubi, saj je preveč možnih kombinacij (gleda le nekaj sosednjih). Uporabil sem implementacijo, ki je napisana v Orange.

InformationGain, ki sem ga implementiral ze v drugi domači nalogi, je prepisan s pomočjo numpy, ki je zelo hiter pri računaju z vektorji.

Da sem določil, ali je ocena dobra ali ne, sem uporabil **permutacijski test** in sicer s **stotimi permutacijami** in $\alpha = \mathbf{0.05}$. S temi mejami sem dobil, da je **1072 atributov primernih**.

Algoritma sta na voljo v *ig.py* in *relieff.py*.

# 4 Ocenjevanje kakovosti napovednih modelov

Modele ocenjujemo z funkcijo logLoss:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N} y_i \ln(\hat{y}_i) + (1 - y_i) \ln(1 - \hat{y}_i),$$

kjer je $N$ število primerov, $ln$ naravni logaritem, $\hat{y}_i$ naša verjetnost za i-ti primer in $y_i$ rezultat razreda za i-ti primer (ki je 0 ali 1).

Naš cilj je minimizirati oceno logLoss.

Da sem ugotovil, kateri že znani algoritmi za klasifikacijo ugajajo našim podatkom sem si izbral pet različnih klasifikatorjev, kateri so že implementirani v Orange:

- Logistic regression
- Random Forest (50 dreves)
- Naivni bayes
- K-Nearest Neighbors
- Linear Support Vector Machine

Implementiral sem 10-kratno prečno preverjanje in testiral zgornje klasifikatorje. Kot mejo sem si izbral najboljšo konstantno verjetnost, to je 0.542400, kjer sem dobil logLoss enak 0.690. To mejo sta prekoračila le Random Forest (z $logLoss = 0.467$) in K-Nearest Neighbors (z $logLoss = 0.540$).

Prečno preverjanje in logLoss algoritem je na voljo v *ocene.py*.

# 5 Zaključek

Najboljši rezultat prečnega preverjanja je bil RandomForest s 50 drevesi. S tem modelom bi na realnem tekmovanju prišel na približno 160-to mesto. Ta rezultat je pod benchmarkom RandomForesta. Temu bi se lahko približal z večjim številom dreves, celo višje pa bi mogoče lahko prišel z selekcijo in dodajanjem atributov in z uporabo metode stacking.

# 6 Izjava o izdelavi domače naloge

Domačo nalogo in pripadajoče programe sem izdelal sam.