Laborator 3.1 - Tehnologii NoSQL

Metode probabilistice pentru clasificare - Naive Bayes

Gheorghe Cosmin Silaghi

Universitatea Babeș-Bolyai

March 31, 2023

Metoda Naive Bayes

Rata de eroare de test este minimizata, in medie, daca, pe baza atributelor independente, atribuim fiecarei observatii din setul de test cea mai plauzibila clasa j pentru care.

$$Pr(Y = j|X = x_0) \tag{1}$$

este cea mai mare

Metoda Naive Bayes

Rata de eroare de test este minimizata, in medie, daca, pe baza atributelor independente, atribuim fiecarei observatii din setul de test cea mai plauzibila clasa j pentru care.

$$Pr(Y = j|X = x_0) \tag{1}$$

este cea mai mare

Pentru o problema de clasificare binara

Se prezice clasa 1 daca $Pr(Y = 1|X = x_0) > 0.5$ si clasa opusa in caz contrar

Metoda Naive Bayes

Rata de eroare de test este minimizata, in medie, daca, pe baza atributelor independente, atribuim fiecarei observatii din setul de test cea mai plauzibila clasa j pentru care.

$$Pr(Y = j|X = x_0) \tag{1}$$

este cea mai mare

Pentru o problema de clasificare binara

Se prezice clasa 1 daca $Pr(Y = 1|X = x_0) > 0.5$ si clasa opusa in caz contrar

Metoda de clasificare naive Bayes estimeaza distributiile de probabilitati conditionate $P(X_i|Y)$ si dupa accea, atribuie fiecarei instante de test clasa care maximizeaza ec. 1

Metoda Naive Bayes

Rata de eroare de test este minimizata, in medie, daca, pe baza atributelor independente, atribuim fiecarei observatii din setul de test cea mai plauzibila clasa j pentru care.

$$Pr(Y = j|X = x_0) \tag{1}$$

este cea mai mare

Pentru o problema de clasificare binara

Se prezice clasa 1 daca $Pr(Y = 1|X = x_0) > 0.5$ si clasa opusa in caz contrar

Metoda de clasificare naive Bayes estimeaza distributiile de probabilitati conditionate $P(X_i|Y)$ si dupa accea, atribuie fiecarei instante de test clasa care maximizeaza ec. 1

Presupuneri de baza

- atributele sunt independente statistic
- atributele sunt au acceeasi importanta

Modelul Naive Bayes

$$Pr(y_{value}|X = (x_1, x_2, \ldots, x_p)) \approx \prod_{i=1}^{p} Pr(X_i = x_i|Y = y_{value}) \quad (2)$$

- pentru atribute nominale, $Pr(X_i = value_i|Y = y_{value})$ va fi estimata prin numararea aparitiei perechilor $(value_i, y_{value})$ in setul de antrenament
- probabilitatile apriori $Pr(Y = y_{value})$ vor fi estimate numarand y_{value} in setul de antrenament
- pentru atribute numerice, $Pr(X|Y)$ va fi estimata pe baza presupunerii ca $Pr(X|Y = y_{value})$ urmeaza o distributie gausiana cu media si abaterea standard estimata din instantele (x, y_{value}) in setul de antrenament.

Modelul Naive Bayes

$$Pr(y_{value}|X = (x_1, x_2, \ldots, x_p)) \approx \prod_{i=1}^{p} Pr(X_i = x_i|Y = y_{value}) \quad (2)$$

- pentru atribute nominale, $Pr(X_i = value_i|Y = y_{value})$ va fi estimata prin numararea aparitiei perechilor $(value_i, y_{value})$ in setul de antrenament
- probabilitatile apriori $Pr(Y = y_{value})$ vor fi estimate numarand y_{value} in setul de antrenament
- pentru atribute numerice, $Pr(X|Y)$ va fi estimata pe baza presupunerii ca $Pr(X|Y = y_{value})$ urmeaza o distributie gausiana cu media si abaterea standard estimata din instantele (x, y_{value}) in setul de antrenament.

Corectii

- se foloseste metoda de netezire Laplace prin care se incepe numararea de la o valoare mica pentru a evita situatia lipsei de informatii in setul de antrenament pentru o anume pereche de valori $(value_i, y_{value})$
- functia de distributie pentru atribute numerice poate fi estimata si in mod ne-parametric, folosind estimatori de tip kernel.

Atitudinea si performanta angajatilor

- set propus initial de IBM
- contine factori care influenteaza atitudinea si performanta angajatilor
- 1470 instante
- 31 atribute
- trebuie sa incarcam pachetul *library(modelsdata)* pentru a obtine setul de date

Cod sursa disponibil pe Moodle

Primii pasi

Primii pasi

Incarcarea datelor in R

```
library(modelsdata)
data(attrition)
```

Primii pasi

Incarcarea datelor in R

```
library(modelsdata)
data(attrition)
```

Inspectarea variabilelor numerice

```
attrition %>%
select_if(is.numeric) %>%
gather(metric, value) %>%
ggplot(aes(value, fill=metric)) +
geom_density(show.legend = FALSE) +
facet_wrap(~metric, scales = "free")
```

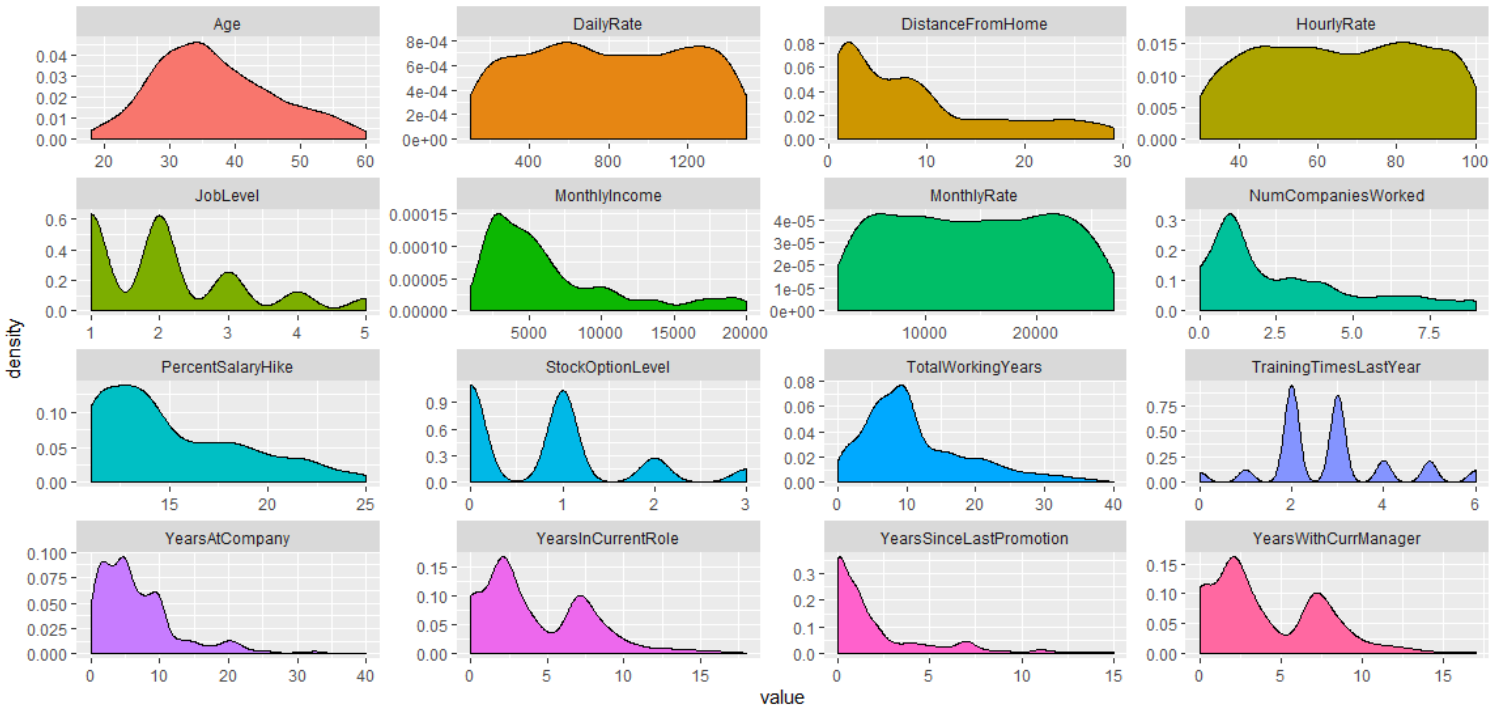
Primii pasi

Incarcarea datelor in R

```
library(modelsdata)
data(attrition)
```

Inspectarea variabilelor numerice

```
attrition %>%
select_if(is.numeric) %>%
gather(metric, value) %>%
ggplot(aes(value, fill=metric)) +
geom_density(show.legend = FALSE) +
facet_wrap(~metric, scales = "free")
```



Preprocesarea datelor

- observam faptul ca variabilele *StockOptionLevel*, *JobLevel*, *TrainingTimeLastYear* arata ca niste variabile nominale, dar in setul de date sunt marcate ca si numerice
- se convertesc aceste variable in nominale - factori

Cod R

```
attrition <- attrition %>%
mutate( JobLevel = factor(JobLevel),
StockOptionLevel = factor(StockOptionLevel),
TrainingTimesLastYear = factor(TrainingTimesLastYear) )
```

Invatarea unui model - packageul *Caret*

- `http://topepo.github.io/caret/index.html`
- permite automatizarea operatiilor fluxului de procesari Machine Learning, pentru invatarea unui model (pentru clasificare si predictie numerica)
- pune la dispozitie un numar mare de modele (238 modele disponibile la acest moment)

Invatarea unui model - packageul *Caret*

- `http://topepo.github.io/caret/index.html`
- permite automatizarea operatiilor fluxului de procesari Machine Learning, pentru invatarea unui model (pentru clasificare si predictie numerica)
- pune la dispozitie un numar mare de modele (238 modele disponibile la acest moment)

Functia train - parametri

- data: setul de date pe care se face invatarea
- method: metoda de invatare
- trControl: procedura aleasa pentru validarea invatarii. Poate fi CrossValidation, Bootstrap, Holdout sau altele
- tuneGrid: permite cautare exhaustiva pe combinatia de valori ai parametrilor metodei, furnizati la intrare

Impartirea setului de date in set de antrenare si set de test

```
set.seed(123)
split <- initial_split(attrition, prop = 0.7, strata = "Attrition")
train <- training(split)
test <- testing(split)
```

Impartirea setului de date in set de antrenare si set de test

```
set.seed(123)
split <- initial_split(attrition, prop = 0.7, strata = "Attrition")
train <- training(split)
test <- testing(split)
```

extragerea esantioanelor s-a realizat cu stratificare dupa atributul Attrition: atat setul de antrenare cat si cel de test au proportie similara a valorilor Attrition

```
table(train$ Attrition)
table(test$ Attrition)
```

Impartirea setului de date in set de antrenare si set de test

```
set.seed(123)
split <- initial_split(attrition, prop = 0.7, strata = "Attrition")
train <- training(split)
test <- testing(split)
```

extragerea esantioanelor s-a realizat cu stratificare dupa atributul Attrition: atat setul de antrenare cat si cel de test au proportie similara a valorilor Attrition

```
table(train$ Attrition)
table(test$ Attrition)
```

Reproductibilitatea rezultatelor in cazul folosirii randomizarii

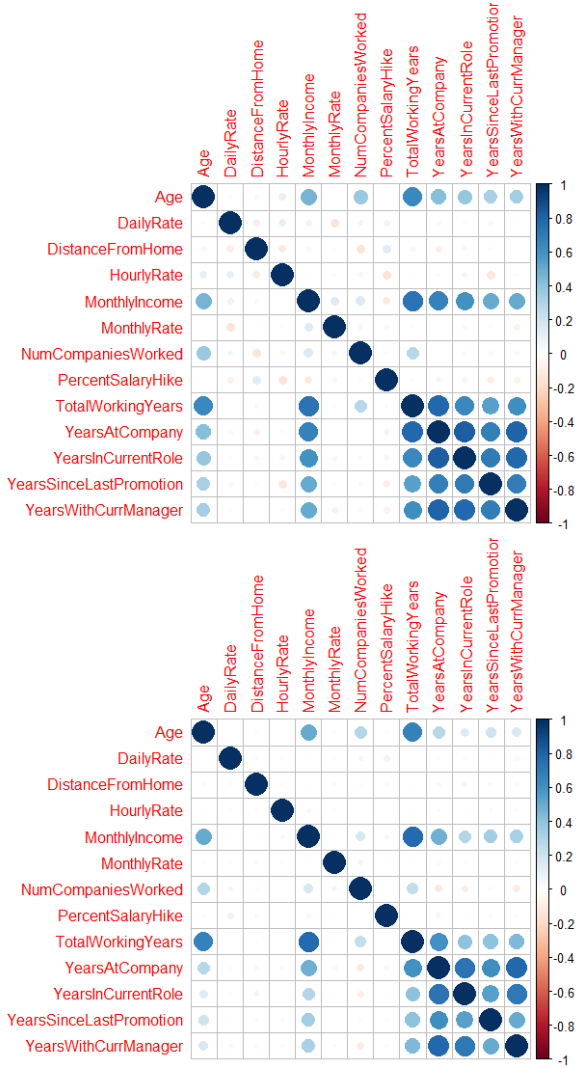
- se foloseste *set.seed* pentru reproductibilitatea rezultatelor
- la realizarea reala a modelului, *set.seed* trebuie inlaturat

Corelarea atributelor numerice

R code

```
train %>%
filter(Attrition == "Yes") %>%
select_if(is.numeric) %>%
cor() %>%
corrplot::corrplot()

train %>%
filter(Attrition == "No") %>%
select_if(is.numeric) %>%
cor() %>%
corrplot::corrplot()
```



Antrenarea modelului de clasificare

- setam atributul Attrition ca si clasa tinta
- celelalte atribute vor constitui variabilele independente (features)
- impartim setul de antrenament intr-un set doar cu features si unul doar cu clasa tinta

Cod R

```
features <- setdiff(names(train), "Attrition")
x <- train[,features]
y <- train$Attrition
```

Stabilim o metoda de validare: 10-folds Cross Validation

```
train_control <- trainControl(
method = "cv",
number = 10 )
```

Invatarea modelului Naive Bayes

```
mod_nb1 <- train(
x = x,
y = y,
method = "nb",
trControl = train_control )

confusionMatrix(mod_nb1)
```

- Acuratete: 0.832 - calculata ca si medie a celor 10 acurateti obtinute pe cele 10 procese de validare din cadrul procedurii 10-folds CV
- Matricea de confuzie:

		True values	
		No	Yes
Predicted values	No	75.3	8.3
	Yes	8.5	7.9

Imbunatatirea clasificatorului

- putem realiza un grid de cautare cu optiunile posibile care sa fie considerate la invatarea modelului
- se va calcula un model pentru fiecare combinatie posibila de optiuni, si vom putea selecta combinatia cu cea mai buna performanta

```
Define sets of model parameter values to evaluate
for each parameter set do
    for each resampling iteration do
        Hold–out specific samples
        [Optional] Pre–process the data
        Fit the model on the remainder
        Predict the hold–out samples
    end
    Calculate the average performance across hold–out predictions
end
Determine the optimal parameter set
Fit the final model to all the training data using the optimal parameter set
```

Parametrii de optimizat la NB

- usekernel: folosirea functiilor kernel pentru estimarea distributiilor atributelor numerice
- adjust: ajustarea latimii de banda a functiilor kernel. valori mai mari inseamna o flexibilitate mai mare la estimarea acestora
- fL: folosirea metodei Laplace pentru netezire. Se indica nivelul de unde se porneste numararea

R code

```
search_grid <- expand.grid(
usekernel = c(TRUE, FALSE),
fL = 0.5,
adjust = seq(0, 5, by = 1) )
```

Re-antrenarea modelelor folosind gridul de cautare

```
mod_nb2 = train(
x = x,
y = y,
method = "nb",
trControl = train_control,
tuneGrid = search_grid,
)
confusionMatrix(mod_nb2)
```

		True values	
		No	Yes
Predicted values	No	81.7	11.5
	Yes	2.2	4.7

Accuracy: 0.8631

Modelele rezultate in urma cautarii

Cod R

```
mod_nb$results %>%
top_n(5, wt = Accuracy) %>%
arrange(desc(Accuracy))
```

Cele mai bune 5 modele

	usekernel	fL	adjust	Accuracy	Kappa	AccuracySD	KappaSD
1	TRUE	0.5	4	0.8631068	0.3382185	0.02069680	0.11189822
2	TRUE	0.5	3	0.8601942	0.3985884	0.02865494	0.11287270
3	TRUE	0.5	5	0.8592233	0.2683726	0.02059534	0.11899273
4	TRUE	0.5	2	0.8524272	0.4310778	0.03028960	0.09673828
5	TRUE	0.5	1	0.8242718	0.3858349	0.04960015	0.15101812

Realizarea predictiilor

- cu cei mai buni parametri identificati in pasul anterior se construieste un model folosind tot setul de antrenament
- acest model se aplica pe setul de test, care nu a fost deloc utilizat in pasul de invatare

R code

```
pred <- predict(mod_nb2, test)
confusionMatrix(pred, test$Attrition)
```

		True values	
		No	Yes
Predicted values	No	362	45
	Yes	7	26

- Accuracy: 0.8818
- 95% confidence interval: (0.8479, 0.9105)
  - daca intervalul de incredere contine acuratetea Zero Information Rate classifier, atunci modelul obtinut de noi nu este mai bun decat cel bazat exclusiv pe informatia din variabila tinta, deci metoda nu a putut extrage informatie utila din variabilele independente pentru imbunatatirea clasificarii.
- p-value: 0.0066 - probabilitatea ca performanta clasificarii sa fie obtinuta datorita sansei
- sensitivity: 0.981 specificity: 0.3662

Desenarea curbei ROC

Pachetul pROC

```
library(pROC)
roc.val <- roc(actual.class ~probability, dataset)
adf <- data.frame(x <- roc.val$specificities, y <- roc.val$sensitivities)
ggplot(adf, aes(x, y)) + geom_line() + scale_x_reverse()
```