Classification

Code

02/17/2023

Introduction to Classification

Definition:

- Classification is a machine learning algorithm that assignes/categorizes a given input as one of several predefined classes.

- Classification involves training our machine learning algorithm to recognize differnt classes and the attribures that distiguaish them. A polpulr implementation of this concept is object recognition software

Strengths of Classification:

1. Classification algorithms are highly accurate and improve significantly the larger the training data set is

2. Classification algorithms are easily interpreted an often include reason for certain decisions so one can trace the logic that the algorithm used to make a decision

3. Classification algorithms are robust and can handle corrupted/obfuscated or noisy observation while maintaining the consistency of its classification

Weaknesses of Classification:

1. Some classification algorithms overfit data meaning that it creates unnecessary associations within the training data. As such, its performs poorly with unseen data

2. Classification algorithms require huge amount of reprocessed data to accurately classify unknown observations

3. Some classification are computationally complex meaning that the have high space and/or time requirements

Goal:

- In this notebook, I intend to demonstrate the life cycle of a machine learning project using classification and analyze accurate the model is at at identifying unseen data.

Machine Learning Life Cycle

Data Set:

- For this project, I'm using a data set that represents a marketing campaign by a Portuguese bank. The campaign was conducted by phone. The goal was to call a client multiple times and see if they'd like to subscribe to a bank term deposit

Hide

```
bank.data <- read.csv(file="bank-full.csv", head= TRUE,sep=";")
str(bank.data)
```

```
'data.frame':   45211 obs. of  17 variables:
 $ age      : int  58 44 33 47 33 35 28 42 58 43 ...
 $ job      : chr  "management" "technician" "entrepreneur" "blue-collar" ...
 $ marital  : chr  "married" "single" "married" "married" ...
 $ education: chr  "tertiary" "secondary" "secondary" "unknown" ...
 $ default  : chr  "no" "no" "no" "no" ...
 $ balance  : int  2143 29 2 1506 1 231 447 2 121 593 ...
 $ housing  : chr  "yes" "yes" "yes" "yes" ...
 $ loan     : chr  "no" "no" "yes" "no" ...
 $ contact  : chr  "unknown" "unknown" "unknown" "unknown" ...
 $ day      : int  5 5 5 5 5 5 5 5 5 5 ...
 $ month    : chr  "may" "may" "may" "may" ...
 $ duration : int  261 151 76 92 198 139 217 380 50 55 ...
 $ campaign : int  1 1 1 1 1 1 1 1 1 1 ...
 $ pdays    : int  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
 $ previous : int  0 0 0 0 0 0 0 0 0 0 ...
 $ poutcome : chr  "unknown" "unknown" "unknown" "unknown" ...
 $ y        : chr  "no" "no" "no" "no" ...
```

- The data comprises factors such as age, job status, balance, housing status, etc. Our job is to determine some relathionship between the people that sign up for the term and those that don't

Step1: Data Cleaning

- Before performing any analysis or computation on data, it must be cleaned. It involves

    1. **Properly naming columns and rows**
       In this case, the default, balance, housing, loan, contact, campagin, and y columns are ambiguous in meaning

Hide

```
colnames(bank.data)[5] = "hasDefaultedB4"
colnames(bank.data)[6] = "avgYearlyBal"
colnames(bank.data)[7] = "hasHouseLoan"
colnames(bank.data)[8] = "hasPersonalLoan"
colnames(bank.data)[9] = "contactType"
colnames(bank.data)[16] = "prevOutcome?"
colnames(bank.data)[17] = "subscribed?"
str(bank.data)
```

```
'data.frame':   45211 obs. of  17 variables:
 $ age            : int  58 44 33 47 33 35 28 42 58 43 ...
 $ job            : chr  "management" "technician" "entrepreneur" "blue-collar" ...
 $ marital        : chr  "married" "single" "married" "married" ...
 $ education      : chr  "tertiary" "secondary" "secondary" "unknown" ...
 $ hasDefaultedB4 : chr  "no" "no" "no" "no" ...
 $ avgYearlyBal   : int  2143 29 2 1506 1 231 447 2 121 593 ...
 $ hasHouseLoan   : chr  "yes" "yes" "yes" "yes" ...
 $ hasPersonalLoan: chr  "no" "no" "yes" "no" ...
 $ contactType    : chr  "unknown" "unknown" "unknown" "unknown" ...
 $ day            : int  5 5 5 5 5 5 5 5 5 5 ...
 $ month          : chr  "may" "may" "may" "may" ...
 $ duration       : int  261 151 76 92 198 139 217 380 50 55 ...
 $ campaign       : int  1 1 1 1 1 1 1 1 1 1 ...
 $ pdays          : int  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
 $ previous       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ prevOutcome?   : chr  "unknown" "unknown" "unknown" "unknown" ...
 $ subscribed?    : chr  "no" "no" "no" "no" ...
```

2. Removing duplicates from a data set. In this case there are none

Hide

```
print (sum(duplicated(heart.data)))
```

```
[1] 0
```

3. Addressing NA's (unavailable data) if required. In this case there are none

Hide

```
print (sum(is.na(bank.data)))
```

```
[1] 0
```

Step 2: Creating Linear Models Test Data

A. Creating Test Data

- To ensure that the model is trained accurately, it is necessary to randomize the order of entries in the data frame

	a..	job	marital	education	hasDefaultedB4	avgYearlyBal	hasHouseL...
	<int>	<chr>	<chr>	<chr>	<chr>	<int>	<chr>
12744	27	management	single	secondary	no	1609	yes
19914	54	management	divorced	tertiary	no	-125	no
8016	31	blue-collar	married	secondary	no	0	yes
35132	42	blue-collar	married	primary	no	2111	yes
23880	48	technician	divorced	secondary	no	105	no
29663	59	management	single	tertiary	no	705	no

6 rows | 1-8 of 17 columns

- Next, we assign 80% (36169) of the data to train set and the rest to the test set

Hide

```
bank.train <- reordered.bank.data[1:36169, ]
bank.test <- reordered.bank.data[36179: 45211, ]
paste("Training set Size: ", nrow(bank.train))
```

```
[1] "Training set Size:  36169"
```

Hide

```
paste("Test set Size:     ", nrow(bank.test))
```

```
[1] "Test set Size:      9033"
```

B. Exploring the data

- The Average Yearly balance of people at different education levels:

Hide

```
paste("Average age of particiants with tertiary education participants: ", mean(bank.train
$avgYearlyBal[bank.train$education=="tertiary"]))
```

```
[1] "Average age of particiants with tertiary education participants:  1777.43590951932"
```

Hide

```
paste("Average age of particiants with secondary education participants: ", mean(bank.trai
n$avgYearlyBal[bank.train$education=="secondary"]))
```

```
[1] "Average age of particiants with secondary education participants:  1161.92719647274"
```

Hide

```
paste("Average age of particiants with primary education participants: ", mean(bank.train
$avgYearlyBal[bank.train$education=="primary"]))
```

```
[1] "Average age of particiants with primary education participants:  1235.61054172767"
```

- Number of people per educational group that have defauted on a loan

Hide

```
paste("Number of particiants with tertiary education that have defaulted: ", nrow(bank.tra
in[bank.train$hasDefaultedB4=="yes" & bank.train$education=="tertiary",]))
```

```
[1] "Number of particiants with tertiary education that have defaulted:  160"
```

Hide

```
paste("Number of particiants with secondary education that have defaulted: ", nrow(bank.tr
ain[bank.train$hasDefaultedB4=="yes" & bank.train$education=="secondary",]))
```

```
[1] "Number of particiants with secondary education that have defaulted:  381"
```

Hide

```
paste("Number of particiants with primary education that have defaulted: ", nrow(bank.trai
n[bank.train$hasDefaultedB4=="yes" & bank.train$education=="primary",]))
```

```
[1] "Number of particiants with primary education that have defaulted:  96"
```

- The effectiveness of the campaign (How many people subscribed)

Hide

```
paste("Number of subscribers: ",nrow(bank.train[bank.train$`subscribed?`=="yes",]))
```

```
[1] "Number of subscribers:  4247"
```

- Number of people who smoke/drink/both a housing loan

Hide

```
paste("Number of people with housing loans: ",nrow(bank.train[bank.train$`hasHouseLoan?`=
='no',]))
```

```
[1] "Number of people with housing loans:  0"
```

Step 3: Models

A. Creating Logistic regression Model

Hide

```
bank2 <- bank.data
bank2[bank2 == "NULL"] = NA
bank2 <- na.omit(bank2)
logistic.bank.model3 <- glm(bank2$`subscribed?` ~., data = bank2, family = "gaussian")
summary(logistic.bank.model)
```

B. Creating Naive Bayes Model

Hide

```
# Loading packages for naive Bayes
library(e1071)
library(caTools)
library(caret)
classifier_cl <- naiveBayes(train$cardio ~ ., data = train)
classifier_cl
```

C. Bayes vs Logistic regression

Naïve Bayes is a probabilistic algorithm that works well for high-dimensional data sets and is simple to implement. Additionally, naive Bayes is noise averse and can handle classification of data sets with plenty variables not related to the classification. While it's easy to implement, it assumes that attributes are independent and affect the predicate in the same way making it unsuitable for tasks that don't follow that assumption.

On the other hand, Logistic Regression is a statistical algorithm that handles both linear and nonlinear relationships between attributes and the target variable. It is easy to set up and efficient. The primary drawback of Logistic Regression is its propensity to overfit meaning that the model performs well for the data set that it was trained with but fails for unseen data. Additionally, logistic regression models require large data sets to create models with high accuracy.