Machine Learning with Caret

Dieudonne Ouedraogo

11/15/2018

We want to predict loan status based on some other characteristics

Load Data and package Caret.

We want to predict loan status based on some other characteristics

```
library(caret)
library(knitr)
library(pander)
train<-read.csv("train_u6lujuX_CVtuZ9i.csv",stringsAsFactors = T)
str(train)
```

```
## 'data.frame':    614 obs. of  13 variables:
##  $ Loan_ID          : Factor w/ 614 levels "LP001002","LP001003",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ Gender           : Factor w/ 3 levels "","Female","Male": 3 3 3 3 3 3 3 3 3 3 ...
##  $ Married          : Factor w/ 3 levels "","No","Yes": 2 3 3 3 2 3 3 3 3 3 ...
##  $ Dependents       : Factor w/ 5 levels "","0","1","2",..: 2 3 2 2 2 4 2 5 4 3 ...
##  $ Education        : Factor w/ 2 levels "Graduate","Not Graduate": 1 1 1 2 1 1 2 1 1 1 ...
##  $ Self_Employed    : Factor w/ 3 levels "","No","Yes": 2 2 3 2 2 3 2 2 2 2 ...
##  $ ApplicantIncome  : int  5849 4583 3000 2583 6000 5417 2333 3036 4006 12841 ...
##  $ CoapplicantIncome: num  0 1508 0 2358 0 ...
##  $ LoanAmount       : int  NA 128 66 120 141 267 95 158 168 349 ...
##  $ Loan_Amount_Term : int  360 360 360 360 360 360 360 360 360 360 ...
##  $ Credit_History   : int  1 1 1 1 1 1 1 0 1 1 ...
##  $ Property_Area    : Factor w/ 3 levels "Rural","Semiurban",..: 3 1 3 3 3 3 3 2 3 2 ...
##  $ Loan_Status      : Factor w/ 2 levels "N","Y": 2 1 2 2 2 2 2 1 2 1 ...
```

Pre-processing using Caret

```
sum(is.na(train))
```

```
## [1] 86
```

Next, let us use Caret to impute these missing values using KNN algorithm. We will predict these missing values based on other attributes for that row. Also, we'll scale and center the numerical data by using the convenient preprocess() in Caret.

```
#Imputing missing values using KNN.Also centering and scaling numerical columns
preProcValues <- preProcess(train, method =c("knnImpute","center","scale"))
library('RANN')
train_processed <- predict(preProcValues, train)
sum(is.na(train_processed))
```

```
## [1] 0
```

It is also very easy to use one hot encoding in Caret to create dummy variables for each level of a categorical variable. But first, we'll convert the dependent variable to numerical.

```r
#Converting outcome variable to numeric
train_processed$Loan_Status<-ifelse(train_processed$Loan_Status=='N',0,1)

id<-train_processed$Loan_ID
train_processed$Loan_ID<-NULL

#Checking the structure of processed train file
str(train_processed)
```

```
## 'data.frame':    614 obs. of  12 variables:
##  $ Gender           : Factor w/ 3 levels "","Female","Male": 3 3 3 3 3 3 3 3 3 3 ...
##  $ Married          : Factor w/ 3 levels "","No","Yes": 2 3 3 3 2 3 3 3 3 3 ...
##  $ Dependents       : Factor w/ 5 levels "","0","1","2",..: 2 3 2 2 2 4 2 5 4 3 ...
##  $ Education        : Factor w/ 2 levels "Graduate","Not Graduate": 1 1 1 2 1 1 2 1 1 1 ...
##  $ Self_Employed    : Factor w/ 3 levels "","No","Yes": 2 2 3 2 2 3 2 2 2 2 ...
##  $ ApplicantIncome  : num  0.0729 -0.1343 -0.3934 -0.4617 0.0976 ...
##  $ CoapplicantIncome: num  -0.554 -0.0387 -0.554 0.2518 -0.554 ...
##  $ LoanAmount       : num  0.0162 -0.2151 -0.9395 -0.3086 -0.0632 ...
##  $ Loan_Amount_Term : num  0.276 0.276 0.276 0.276 0.276 ...
##  $ Credit_History   : num  0.432 0.432 0.432 0.432 0.432 ...
##  $ Property_Area    : Factor w/ 3 levels "Rural","Semiurban",..: 3 1 3 3 3 3 3 2 3 2 ...
##  $ Loan_Status      : num  1 0 1 1 1 1 1 0 1 0 ...
```

Now, creating dummy variables using one hot encoding:

```r
#Converting every categorical variable to numerical using dummy variables
dmy <- dummyVars(" ~ .", data = train_processed,fullRank = T)
train_transformed <- data.frame(predict(dmy, newdata = train_processed))

#Checking the structure of transformed train file
str(train_transformed)
```

```
## 'data.frame':    614 obs. of  19 variables:
##  $ Gender.Female          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Gender.Male            : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Married.No             : num  1 0 0 0 1 0 0 0 0 0 ...
##  $ Married.Yes            : num  0 1 1 1 0 1 1 1 1 1 ...
##  $ Dependents.0           : num  1 0 1 1 1 0 1 0 0 0 ...
##  $ Dependents.1           : num  0 1 0 0 0 0 0 0 0 1 ...
##  $ Dependents.2           : num  0 0 0 0 0 1 0 0 1 0 ...
##  $ Dependents.3.          : num  0 0 0 0 0 0 0 1 0 0 ...
##  $ Education.Not.Graduate : num  0 0 0 1 0 0 1 0 0 0 ...
##  $ Self_Employed.No       : num  1 1 0 1 1 0 1 1 1 1 ...
##  $ Self_Employed.Yes      : num  0 0 1 0 0 1 0 0 0 0 ...
##  $ ApplicantIncome        : num  0.0729 -0.1343 -0.3934 -0.4617 0.0976 ...
##  $ CoapplicantIncome      : num  -0.554 -0.0387 -0.554 0.2518 -0.554 ...
##  $ LoanAmount             : num  0.0162 -0.2151 -0.9395 -0.3086 -0.0632 ...
##  $ Loan_Amount_Term       : num  0.276 0.276 0.276 0.276 0.276 ...
##  $ Credit_History         : num  0.432 0.432 0.432 0.432 0.432 ...
##  $ Property_Area.Semiurban: num  0 0 0 0 0 0 0 1 0 1 ...
##  $ Property_Area.Urban    : num  1 0 1 1 1 1 1 0 1 0 ...
##  $ Loan_Status            : num  1 0 1 1 1 1 1 0 1 0 ...
```

```r
#Converting the dependent variable back to categorical
train_transformed$Loan_Status<-as.factor(train_transformed$Loan_Status)
```