Machine Learning for Predicting Bad Loans

by yhat

August 16, 2013

New and creative applications for machine learning are cropping up all over the place. Who knew that agriculturalists are using image recognition to evaluate the health of plants? Or that researchers are able to generate music imitating the styles of masters from Chopin to Charlie Parker? While there's a ton of interest in applying machine learning in new fields, there's no shortage of creativity among analysts solving age-old prediction problems.

This is a post exploring one of the oldest prediction problems--predicting risk on consumer loans.

Predicting Bad Loans

We're going to be using the publicly available dataset of Lending Club loan performance. It's a real world data set with a nice mix of categorical and continuous variables.

LendingClub makes several datasets available on their website. We're going to use the 2007 to 2011 file (`LoanStats3a.csv`), and our goal will be to build a web app which can approve and decline new loan applications.

Read and Clean the Data

```
library(stringr)
library(plyr)
library(lubridate)
library(randomForest)
library(reshape2)
library(ggplot2)

df <- read.csv("~/Downloads/LoanStats3a.csv", h=T, stringsAsFactors=F,

#take a peak...
head(df)

#annoying column; just get rid of it
df[,'desc'] <- NULL

summary(df)
#almost all NA, so just get rid of it
df[,'mths_since_last_record'] <- NULL

#get rid of fields that are mainly NA
poor_coverage <- sapply(df, function(x) {
  coverage <- 1 - sum(is.na(x)) / length(x)
  coverage < 0.8
})
df <- df[,poor_coverage==FALSE]
```

import_lending_club.R hosted with ❤ by GitHub view raw

Let's load the data into R and look at the `status` column. Notice that there are several different statuses which seem to indicate good repayment behavior and several more that indicate less than perfect repayment

behavior.

We're going to approach this as a binary classification problem, so our first step is to decide what statuses we'll consider good and which bad.

```
bad_indicators <- c("Late (16-30 days)", "Late (31-120 days)", "Default

df$is_bad <- ifelse(df$loan_status %in% bad_indicators, 1,
                    ifelse(df$loan_status=="", NA,
                           0))
table(df$loan_status)
table(df$is_bad)

head(df)
df$issue_d <- as.Date(df$issue_d)
df$year_issued <- year(df$issue_d)
df$month_issued <- month(df$issue_d)
df$earliest_cr_line <- as.Date(df$earliest_cr_line)
df$revol_util <- str_replace_all(df$revol_util, "[%]", "")
df$revol_util <- as.numeric(df$revol_util)

outcomes <- ddply(df, .(year_issued, month_issued), function(x) {
  c("percent_bad"=sum(x$is_bad) / nrow(x),
    "n_loans"=nrow(x))
})

plot(outcomes$percent_bad, main="Bad Rate")
outcomes
```

define_bads.R hosted with ❤ by GitHub view raw

I didn't do anything too crazy here. If a loan was ever delinquent or if it is currently active but behind schedule, I considered it "bad". Conversely, I treated other active loans, loans in a grace period, and fully paid off loans to be "good".

Certain statuses were ambigious, and, without a data dictionary, we need to choose how to deal with them. If it wasn't clear that an applicant actually

recieved a loan (e.g. `df$status==""`), I categorized it as NA.

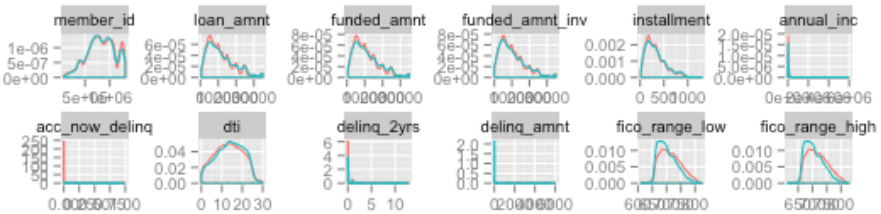
Also, one of the reasons I chose to use the file from 2007 to 2011 was to limit the build sample to mature vintages only. In other words, if we were to look at loans originated in 2012, some would only be part-way through repayment and therefore would appear to be performing better than mature vintages which have had more time to go bad.

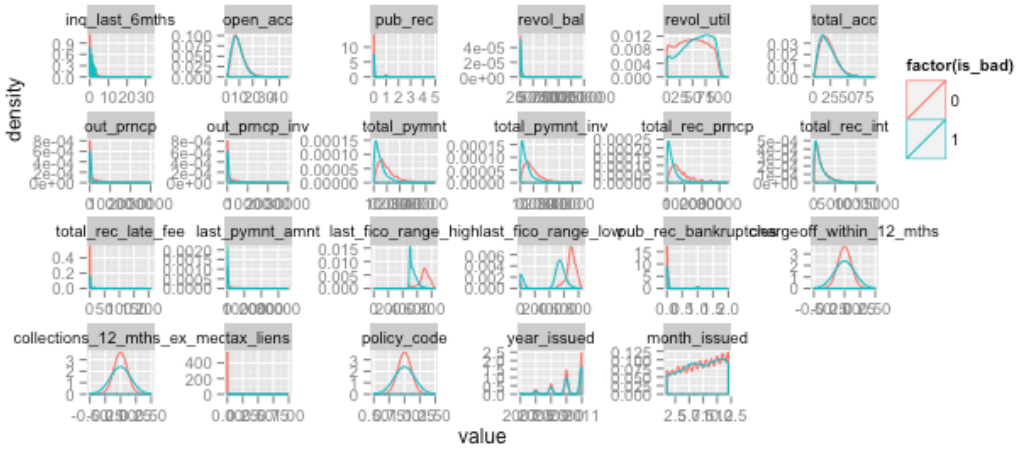
A Strategy for Finding Risky Applicants

For feature selection, I kept it quick and dirty. You can use `ggplot2` to quickly compare the default rates and the distributions of each variable. From there, visually inspect the distributions and pick out a few variables that appear to have significant differences in the bad and good populations.

```
#figure out which columns are numeirc (and hence we can look at the dis
numeric_cols <- sapply(df, is.numeric)
#turn the data into long format (key->value esque)
df.lng <- melt(df[,numeric_cols], id="is_bad")
head(df.lng)

#plot the distribution for bads and goods for each variable
p <- ggplot(aes(x=value, group=is_bad, colour=factor(is_bad)), data=df.
#quick and dirty way to figure out if you have any good variables
p + geom_density() +
  facet_wrap(~variable, scales="free")

#NOTES:
# - be careful of using variables that get created AFTER a loan is issu
# - any ID variables that are numeric will be plotted as well. be sure
```

lending_club_find_features.R hosted with ♥ by GitHub view raw





After getting rid of loans issued after 2012, I was left with approximately 30,000 loan applications. From there I split the data into training (75%) and test (25%) sets.

Random Forest does a pretty outstanding job with most prediction problems (if you're interested, read our post on random forest using python), so I decided to use `R`'s Random Forest package.

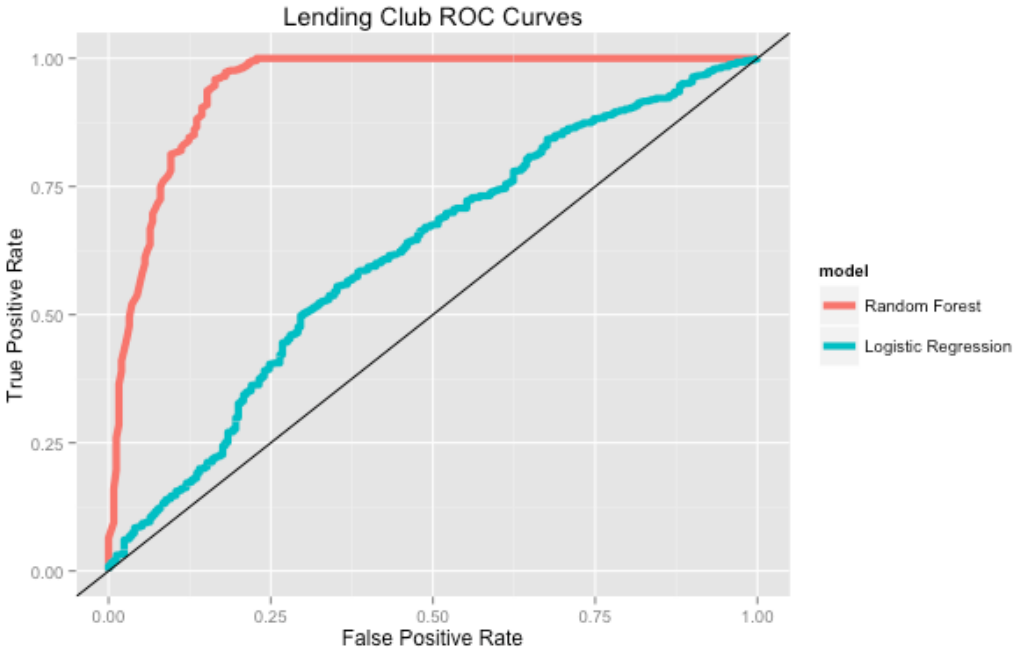
```
# only evaluate w/ vintages that have come to term
df.term <- subset(df, year_issued < 2012)
df.term$home_ownership <- factor(df.term$home_ownership)
df.term$is_rent <- df.term$home_ownership=="RENT"
df.term$fico_range <- factor(df.term$fico_range)
df.term$fico_ordered <- as.numeric(df.term$fico_range)

idx <- runif(nrow(df.term)) > 0.75
train <- df.term[idx==FALSE,]
test <- df.term[idx==TRUE,]



rf <- randomForest(factor(is_bad) ~ last_fico_range_high + last_fico_ra
                     pub_rec_bankruptcies + revol_util + inq_last_6mths
                   type="classification", data=train, importance=TRUE,
```

training_lending_club.R hosted with ♥ by GitHub view raw

There really are lots of ways to skin this cat, so you can and should explore a few. Checkout this post exploring the best modeling techniques among Kaggle participants in the Give Me Some Credit competition

Kaggle participants in the Give Me Some Credit competition.

I horse raced Random Forest against other models, and Random Forest consistently outperformed the other algorithms like logistic regression.



Deploying to Yhat

So I've got this script on my laptop which is cool in an academic sort of way. But these insights would be more useful in a live application. Let's turn our R script into a routine that can be called via REST.

First, wrap your model variables in the `model.transform` and `model.predict` functions. Be sure to handle any categorical variables. You can do this by using the `levels` function in `R`.

Execute the `yhat.deploy` function, and your model is deployed and exposed as a RESTful API that you can call from anywhere!

```
library(yhatr)

model.require <- function() {
  library(randomForest)
}

model.transform <- function(df) {
  df$is_rent <- df$home_ownership=="RENT"
```

```
  df
}

model.predict <- function(df) {
  df$prob_default <- predict(rf, newdata=df, type="prob")[,2]
  df
}
```

lending_club_deploy.R hosted with ♥ by GitHub view raw

Automatically Generate API Docs

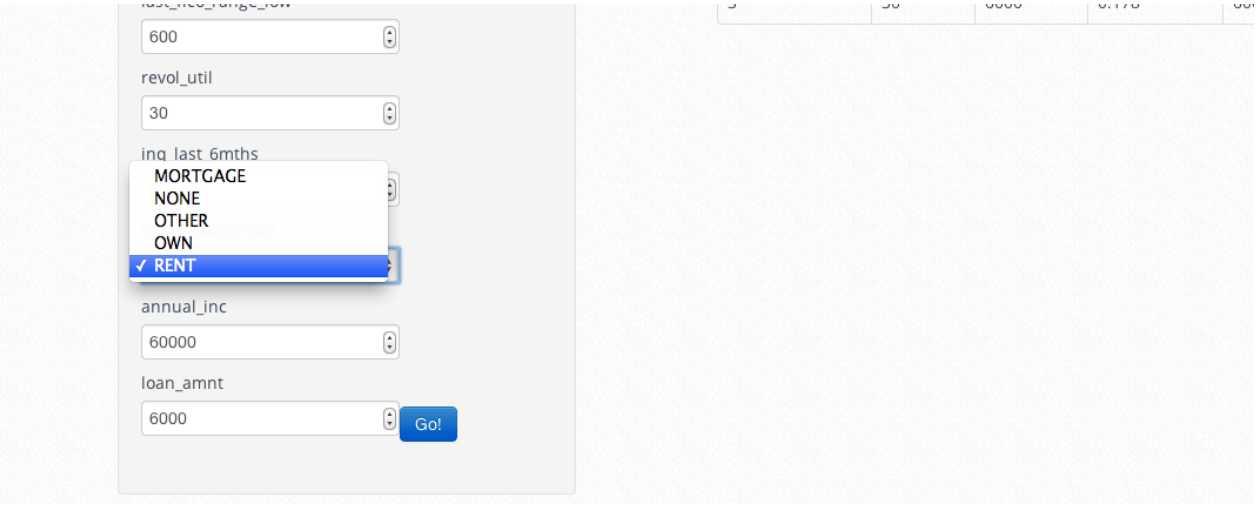
Your model is deployed and can be called via REST. Because you might not be the only person using your API (i.e. others on your team might need to call it to make predictions), you probably want to add some documentation around its usage. Yhat can generate a documentation page for you.

First, set up a test case with raw data as it appears in the wild. In other words, take real observations from your data set or generate some realistic sample data. For us, this will be a few raw loan applications. I'm just using the rows we used for training the model.

Pass that data to the `yhat.document` function along with the model name and model version you want to document (for us this is version number 1). Yhat generates HTML docs and return a URL that looks like this:

```
features <- c("last_fico_range_high", "last_fico_range_low", "revol_uti
              "inq_last_6mths", "home_ownership", "annual_inc", "loan_a
sample.data <- df.term[,features]
sample.data$annual_inc[sample.data$annual_inc > 200000] <- 200000
head(sample.data)
yhat.document(model="LendingClubRandomForest", version=2, df=sample.dat
```

lending_club_doc.R hosted with ♥ by GitHub view raw

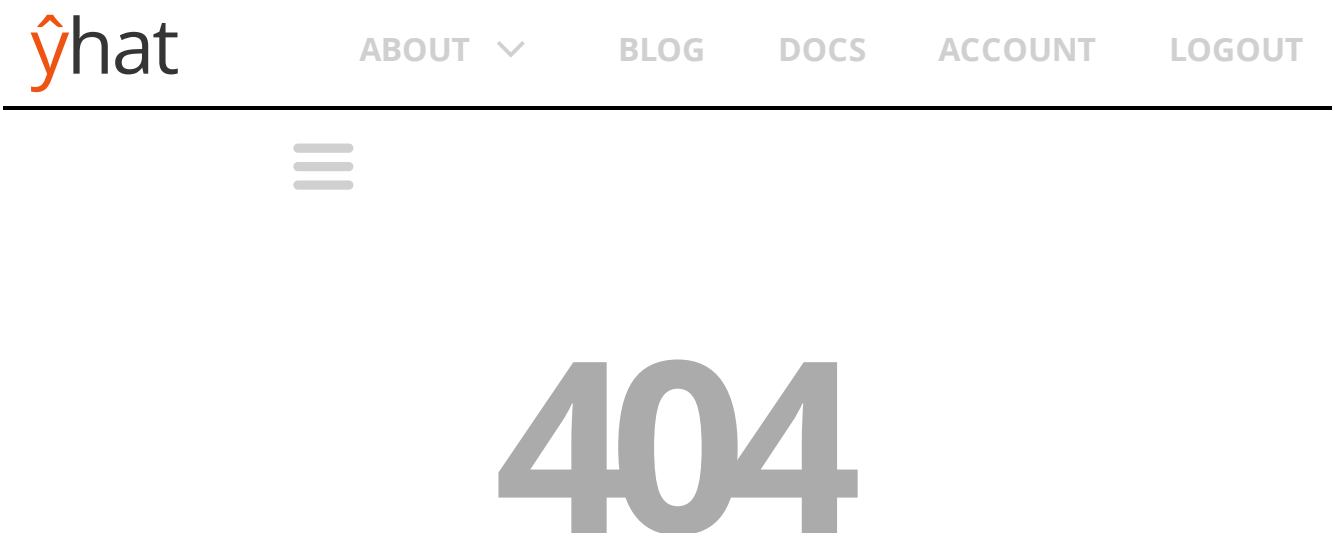


Using the test data you provided, Yhat will identify the input parameters that your model expects when making new predictions. From there, it produces a web app that lets you test the model using a UI. Input some test values and click "Go!" to execute the model in real time.

The App

You can visit my app here, or you can use it in the iframe below.

Use the Results dropdown to display the prediction in HTML or JSON. The JSON format is especially helpful to anybody using your model from other software applications.



Final Thoughts

- Lending Club Stats
- Lending Club Modeling
- Lending Club Loan Analysis: Making Money with Logistic Regression by Dr. Jason Davis
- The Complete Guide to Investor Risks at Lending Club & Prosper by Simon Cunningham
- Lending Club Review - How to Become a Bank
- Big Data + Machine Learning = Scared banks by Jeremy Liew
- Random Forest of 'Give Me Some Credit' Survey Results by Margit Zwemer
- Analysis of Survey Data for the ?Give Me Some Credit? Competition Hosted on Kaggle (PDF whitepaper)

« 10 BOOKS FOR DATA ENTHUSIASTS ESTIMATING USER LIFETIMES - TH... »

info@yhathq.com
support@yhathq.com
(917) 719-5959

Our Products

Enterprise
Cloud
Get the ŷhat AMI »
Enterprise Signup »

Learn More

About
Blog
FAQ
Jobs
Terms of Service

Newsletter

Email Address

Get Updates

Connect With Us