Enron emails (Text analytics)

Tool: Bag of words model

~~[illegible]~~, CART, Random forest
Naive Bayes

The Analytics Edge:

Predictive coding provides an innovative way for lawyers to review large sets of documents. Lawsuits often mean the review of letters, memoranda, files, to determine which ones are relevant. With more email communications, this data has exploded. It is no longer economical to review these individually to find the typically smaller set of sensitive documents. Predictive coding uses sample set reviewed by senior attorney & analytics helps predict from the many documents, the more & less sensitive one

Overview

Enron, an American energy company was formed in 1985. It was originally involved in transmitting and distributing electricity and natural gas throughout the US. Before 2001, Enron employed nearly 20000 staff with claimed revenues of nearly $111 billion during 2000.

By the end of 2001, it was revealed that the reported financial condition came about due to systematic accounting fraud (the Enron Scandal)

Enron Corpus

The Enron Corpus is a publicly available dataset of the email messages sent or received by about 150 Seniors managers of Enron. This data was obtained by the Federal Energy Regulatory Commission during its investigation. This Corpus is one of the few publicly available mass collection of real emails available for study as such collections often are bound by privacy laws.

Predictive coding

Predictive coding allows software to take information entered by people and generalize it to a larger group of documents.
Traditionally, one searches a set of documents for a term to see and identify documents relevant to a case. Rather predictive coding considers the context. Instead of having legal experts going through all the documents, one uses a training set from a broader set which is categorized by legal experts. Then use of predictive analytics tools help predict the categorization of future documents

Californnia energy crisis

The California energy crisis in 2000-2001 was a situation in which California had a shortage of electric supply caused by market manipulations, illegal shutdown of pipllines by the Texas energy consortium Enron. The state suffered from multiple blackouts, while Enron gained the market which was deregulated.

The Federal Energy Regulatory Commission (FERC) investigated Enron's involvment in the crisis and this ultimately led to a $1.52 billion settlement

As part of the investigation FERC released the emails from some of the executies at Enron. One approach is to search for keywords like "electricity bid", "energy schedule" in emails and then carefully review which ones are responsive. Predictive coding using text analytics is an alternate & newer way to do this. The data comes from a 2010 Text Retrieval Conference - legal Track where attorneys labeled emails responsive to energy schedules or bids.

Analytics on Enron dataset

energy ← read.csv ("energy_bid .csv", $tringsAsFactors = FALSE)

Str(energy)

Head(energy)

The data frame contains a total of 851 emails where

$email = Content of email

$responsive = { 1 if email is responsive to query about energy bids & schedules ; 0 Otherwise }

energy $email [1]

dists out the entire email but this is hard to read

Strwrap (energy $email [1])

This command wraps character strings to format paragraphs to be easy to read

energy $responsive [1]

This takes a value 0 since the email is not responsive to energy bid & schedule

Strwrap (energy $ email [4])

energy $ responsive [4]

The fourth email deals with the search on the energy bids & schedules. Hence the responsive variable takes a value of 1 for this email

table (energy $ responsive)

0	1
714	137

} Many of the emails do not relate to the energy bids & schedules.

Preprocessing

library (tm)

corpus ← Corpus (VectorSource (energy $ email))

Creates the corpus

Strwrap (as.character (corpus [[4]]))

Read the email in the corpus

corpus ← tm_map(corpus, [redacted])
tolower

corpus ← tm_map(corpus, removeWords,
stopwords("english"))

corpus ← tm_map(corpus, removePunctuation)

corpus ← tm_map(corpus, removeNumbers)

corpus ← tm_map(corpus, stemDocument)

strwrap(as.character(corpus[[4]]))

Note the email is much harder for us as humans to make sense of after the preprocessing but it is much more suitable to apply analytics

freq ← DocumentTermMatrix(corpus)

The document-term matrix has 851 documents, 13770 [redacted] terms. The matrix is again very sparse (99%)

freq ← removeSparseTerms(freq, 0.97)

Removes all terms that do not occur in atleast 3% of documents.

We now have 612 terms left.

energysparse ← as.data.frame(as.matrix(freq))

energysparse$responsive ← energy$responsive

colnames(energysparse) ← make.names(colnames(energysparse))

library(caTools)

set.seed(1978)

spl ← sample.split(energysparse$responsive, SplitRatio = 0.7)

train ← subset(energySparse, spl == TRUE)

test ← subset(energysparse, spl == FALSE)

Base model

table(train$responsive)

0	1
500	96

} Majority of emails are non-responsive

Note false negatives are more critical here than false positives.

table(test$responsive)

0	1
214	41

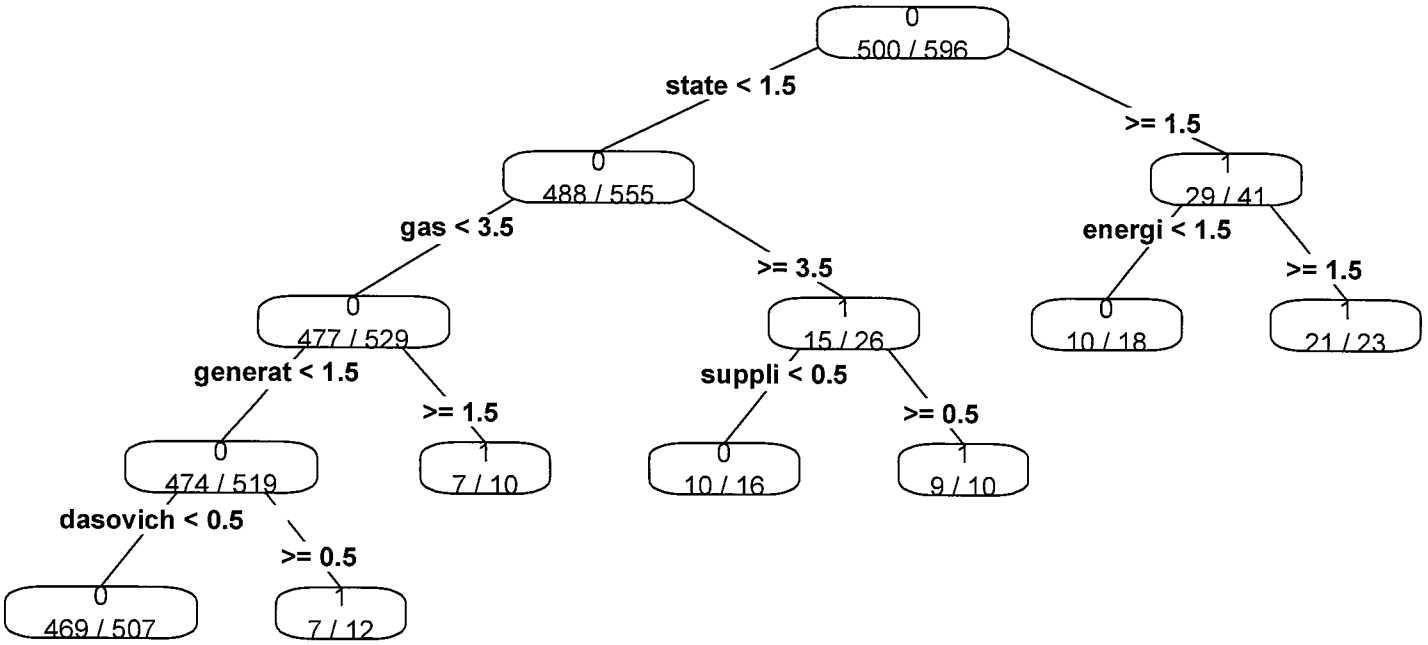
} Baseline model will give accuracy = $\frac{214}{214+41} = 0.83$

CART

```
library(rpart)
library(rpart.plot)
set.seed(1)
model1 <- rpart(as.factor(responsive) ~ ., data = train)

prp(model1, type = 4, extra = 2)

Summary(model1)
predict1 <- predict(model1, newdata = test, type = "class")
table(predict1, test$responsive)
```

	0	1
0	199	26
1	15	15

Same accuracy as baseline but true positives have increased.


0
500 / 596
state < 1.5
>= 1.5
0
488 / 555
1
29 / 41
gas < 3.5
energi < 1.5
>= 3.5
>= 1.5
0
477 / 529
1
15 / 26
0
10 / 18
1
21 / 23
generat < 1.5
suppli < 0.5
>= 1.5
>= 0.5
0
474 / 519
1
7 / 10
0
10 / 16
1
9 / 10
dasovich < 0.5
>= 0.5
0
469 / 507
1
7 / 12

Random forest

```
library (randomForest)
set.seed (1979)
model2 <- randomForest (as.factor (responsive) ~ .,
                        data = train)
predict2 <- predict (model2, newdata = test,
                     type = "class")
table (predict2, test$responsive)
```

Predict \ Actual	0	1
0	204	26
1	10	15

Accuracy = 0.858

has improved.

Also we now do a better job of identifying true negatives.

```
varUsed (model2)
```

This helps identify by frequency which predictors are used in the random forest.

```
Order2 <- Sort (varUsed (model2), index.return = TRUE)
```

Returns sorted frequency & indices

```
names (test[, tail (order2$ix)])
```

deff richard pleas attach forward Subject

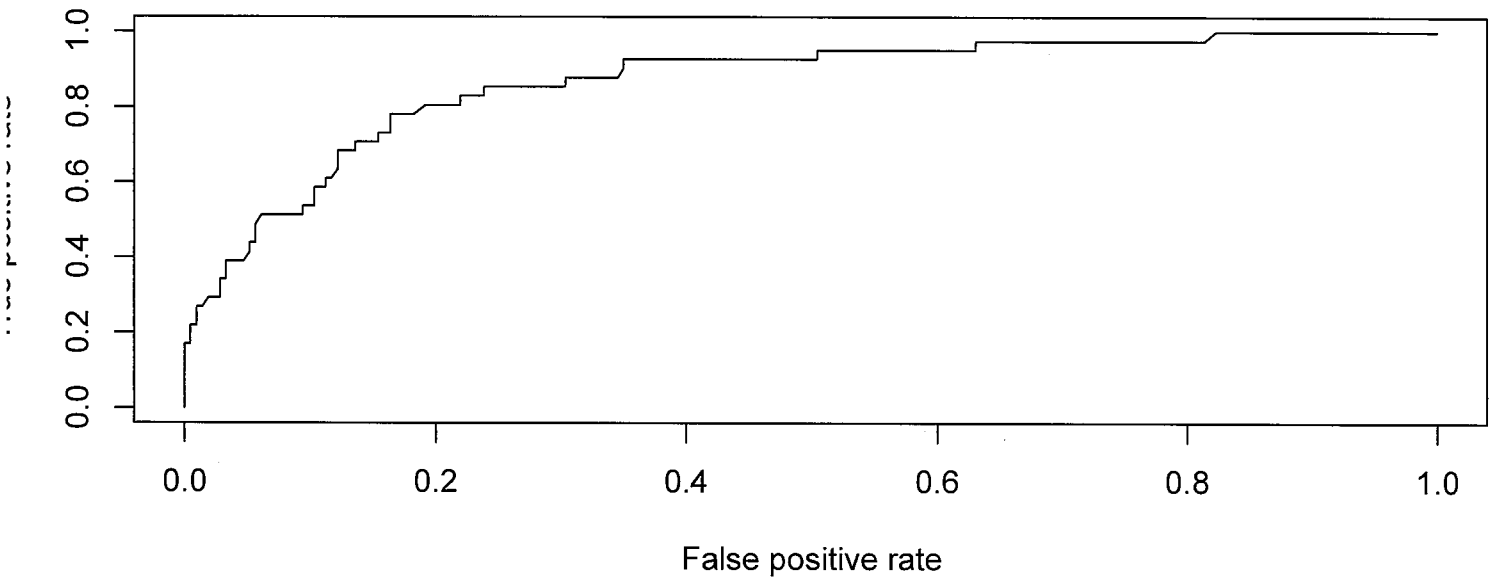
```
library(ROCR)

predict prob1 <- predict(model1, newdata = test,
                         type = "prob")

predict prob 2 <- predict(model2, newdata = test,
                          type = "prob")

pred1 <- prediction(predict prob1[,2], test$responsive)

pred2 <- prediction(predict prob2[,2], test$responsive)

performance(pred1, measure = "auc")

performance(pred2, measure = "auc")
```

AUC of random forest is very good

```
plot(performance(pred2, measure = "tpr", x.measure = "fpr"),
```

Here AUC is around 0.86.

AUC is a good measure across thresholds.


0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate

Naive Bayes Classifier

The naive Bayes classifier is a simple classification rule based on Bayes rule and it can be used with the simple bag of words representation of text.

Bag of words representation of document $\longrightarrow$ Classify the document

$$P\left(\text{Document } d \in \underbrace{\text{Class } k}_{C_k} \,\middle|\, \underbrace{x_{i1}, \ldots, x_{ip}}_{\text{Features of document}}\right)$$

$$P(C_k | x_{i1}, \ldots, x_{ip}) = \frac{P(C_k \cap (x_{i1}, \ldots, x_{ip}))}{P(x_{i1}, \ldots, x_{ip})}$$

Naive bayes classifier assumes independence of the features given the category

$$P(C_k | x_{i1}, \ldots, x_{ip}) \propto P(C_k) P(x_{i1} | C_k) \ldots P(x_{ip} | C_k)$$

For example: A fruit is an apple (category) if it is red, round and around 3 cm in diameter.

A Naive Bayes classifier considers each of these features to contribute to the probability that the fruit is an apple

$$\therefore P(C_k | x_{i1}, \ldots x_{ip}) = \frac{1}{Z} P(C_k) \prod_{i=1}^{p} P(x_{ki} | C_k)$$

where $Z = P(x_{i1}, \ldots, x_{ip})$ depends only on factors

The prediction rule is to assign a class k such that

$$\underset{k=1,\ldots,K}{\operatorname{argmax}} \underbrace{P(C_k)}_{\downarrow} \underbrace{\prod_{i=1}^{p} P(x_{ki} | C_k)}_{\downarrow}$$

estimated from training set and then applied on test set

For example, assuming gaussian prediction variables:

$$P(x_{ki} | C_k) = \frac{1}{\sqrt{2\pi \sigma_{ki}^2}} e^{-\frac{(x_{ki} - \mu_{ki})^2}{2\sigma_{ki}^2}}$$

where μ_{ki} and σ_{ki}^2 are mean and variances of values in category k for variable x_{ki}.

Naive Bayes

$$P(C_k | x_1, \ldots, x_p) = \frac{P(C_k) P(x_1 | C_k) \ldots P(x_p | C_k)}{\sum_{\ell} P(C_\ell) P(x_1 | C_\ell) \ldots P(x_p | C_\ell)}$$

In general,

$$P(C_k | X_1, \ldots X_p) = \frac{P(C_k)\ P(X_1, \ldots, X_p | C_k)}{P(X_1, \ldots, X_p)}$$

$$= \frac{P(C_k)\ P(X_1 | X_2, \ldots, X_p, C_k)\ P(X_2 | X_3, \ldots, X_p, C_k) \ldots P(X_p | C_k}{P(X_1, \ldots, X_p)}$$

To estimate $P(X_1, \ldots, X_p | C_k)$ where each X_i takes 2 values and C_k takes 2 values, we need to estimate approximately $2(2^p)$ parameters (probabilities)

In Naive Bayes, we use & assume:

$$P(X_1, \ldots, X_p | C_k) = P(X_1 | C_k) \ldots P(X_p | C_k),$$

Now we need to estimate approximately

$2p(2)$ parameters (probabilities)

Naive Bayes makes the problem much smaller to solve & estimate. It works with less data too.

Naive Bayes classifier

install.packages("e1071")
library(e1071

This provides a package to use a naive Bayes classifier.

model3 ← naiveBayes(as.factor(responsive) ~ . , data = train)

naiveBayes(.) in the e1071 package computes the conditional posterior probabilities of a categorical variable given independent predictor variables using Bayes rule

Summary(model3)
model3 $ apriori

0	1
500	96

} Distribution of dependent variable

model3 $ tables — lists tables one for each predictor. For each numeric variable, it gives target class, mean & standard deviation of variable

predict3 ← predict(model3, newdata = test, type = "class")
table(predict3, test$responsive)

	0	1
0	184	15
1	30	26

Accuracy is lower than baseline but true positives has increased.