Can we block internet advertisements?

Rommel Bartolome

Summary

Advertisements has always been an essential part of all businesses around the world. In particular, internet advertisements in the form of banners in websites has been on the rise especially at the turn of the century. However, many users do not want to be cluttered with these images. In this exercise, we explore a dataset where we have several image features, such as size, if the site is local and on what site it came from. Exploring the dataset, we found that it had a lot of components (1,559 variables). As such, we performed Principal Components Analysis (PCA) to the data to decrease the number of components. We found that (1) 110 components can explain 75% of the variance and (2) 73 components can explain 60% of the variance, which is a massive decrease from the original number of components. We also classified the images using Support Vector Machines, where indeed, we found that it is possible to classify images as ads or non-ads with an accuracy of 96.2% using a Linear Kernel. This is similar to what the original authors got at 97% accuracy.

Data Reading

From the source, it was described that this dataset represents a set of possible advertisements on Internet pages. The features encode the geometry of the image (if as well as phrases occuring in the URL, the image's URL and alt text, the anchor text, and words occuring near the anchor text. The task is to predict whether an image is an advertisement ("ad") or not ("nonad").

Before anything else, we will need the following tools to do our analysis:

```
library(tidyverse)
library(FactoMineR)
library(pROC)
library(e1071)
library(caret)
```

We now read the data that was provided. We open the names of the columns first and remove all the non-names. Then, we add another column name called, "ad" which will signify if the particular entry is an ad or not.

```
# remove non-names columns
name <- read.csv('ad.names', sep = ',',
                 header = F)[-c(1,2,7,465,961,1434,1546), ][1]

# add column for ad
name[nrow(name) + 1,] <- "ad"
name <- gsub(":.*","", name[[1]])
data <- read.csv('ad.data', sep = ',')

#rename the ad data with the names file
names(data) <- name
```

Here, we have successfully assembled the data needed for our exploration and knowledge discovery.

The following is the data attributes given by the original creators of the dataset:

1. height: continuous
2. width: continuous
3. aratio: continuous

4. local: 0,1.
5. 457 features from url terms, each of the form "url*term1+term2..."
6. 495 features from origurl terms, in same form
7. 472 features from ancurl terms, in same form
8. 111 features from alt terms, in same form
9. 19 features from caption terms
10. tagging if the image is an ad or a non-ad

Data Exploration

We check the data dimensions:

```
data %>% nrow()
```

```
## [1] 3278
```

```
data %>% length()
```

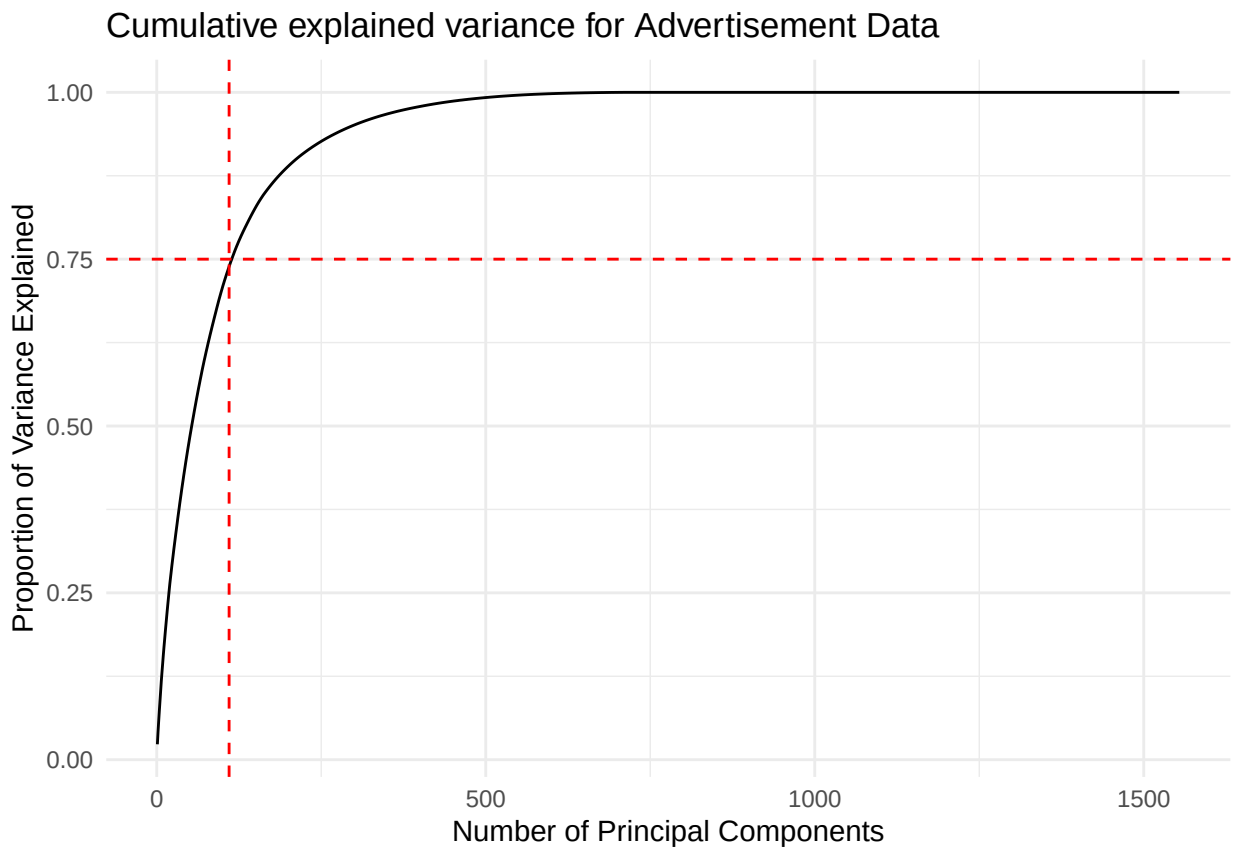
```
## [1] 1559
```

From here, we can see that our data is 3278 entries with 1559 columns. Our data has too many columns. To solve these too much dimensionality, we shall use principal components analysis. Below we run an initial PCA fit on the full data set and display the summary head.

```
n <- data %>% length()
data1_pca <- prcomp(data[,6:n-1], scale = TRUE, center = TRUE)
```

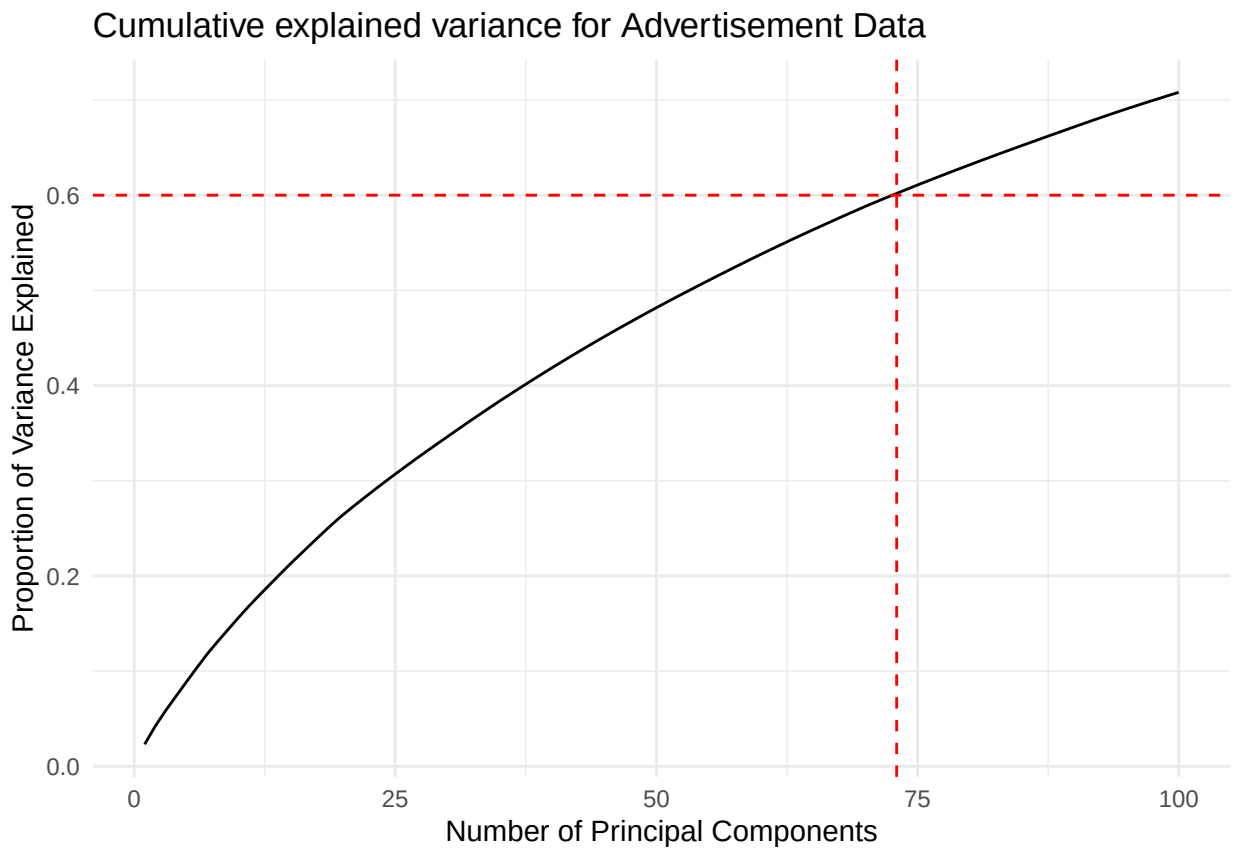
Now, we visualize the importance of the components, we plot cumulative variance of proportion explained:

```
data.frame(
  cev=summary(data1_pca)$importance[3,]
  ) %>% ggplot(aes(x=1:length(cev), y=cev)) +
  geom_line() +
  theme_minimal() +
  labs(title="Cumulative explained variance for Advertisement Data") +
  xlab("Number of Principal Components") +
  ylab("Proportion of Variance Explained") +
  geom_hline(aes(yintercept=0.75), color='red', linetype='dashed') +
  geom_vline(aes(xintercept=110), color='red', linetype='dashed')
```



We find that up to ~110 components, we achieve around 75% of the cumulative proportion of variance explained. We try to lower it since 110 components is still very large:

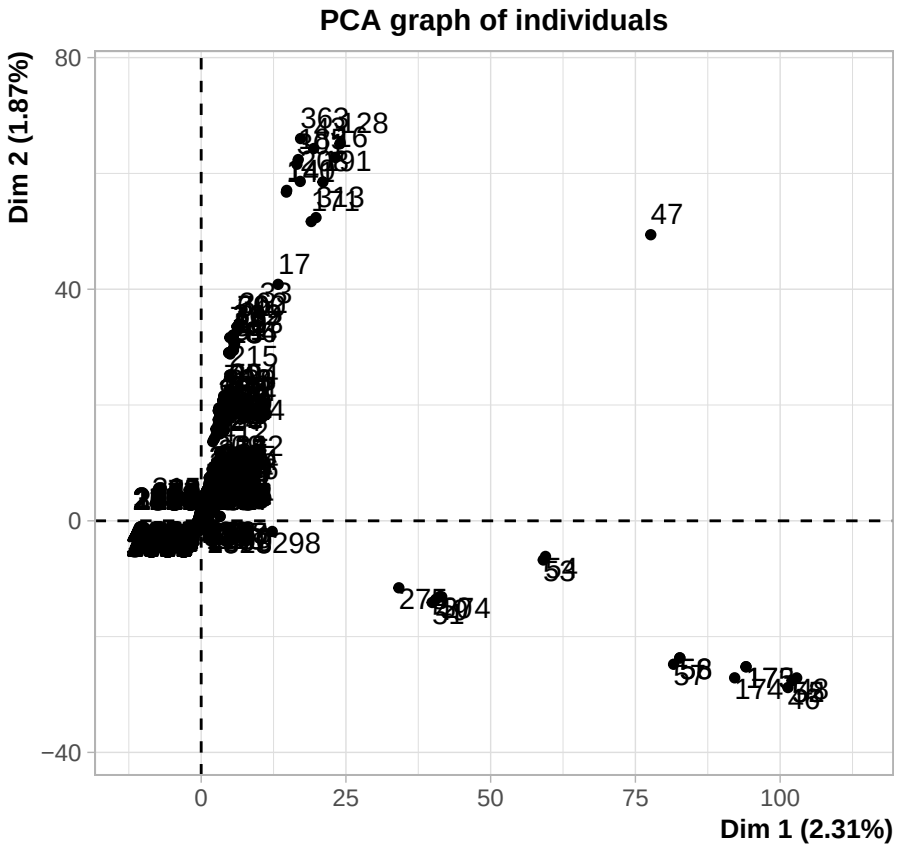
```
data.frame(
  cev <- summary(data1_pca)$importance[3,] %>% head(100)
  ) %>% ggplot(aes(x=1:100, y=cev)) +
  geom_line() +
  theme_minimal() +
  labs(title="Cumulative explained variance for Advertisement Data") +
  xlab("Number of Principal Components") +
  ylab("Proportion of Variance Explained") +
  geom_hline(aes(yintercept=0.60), color='red', linetype='dashed') +
  geom_vline(aes(xintercept=73), color='red', linetype='dashed')
```

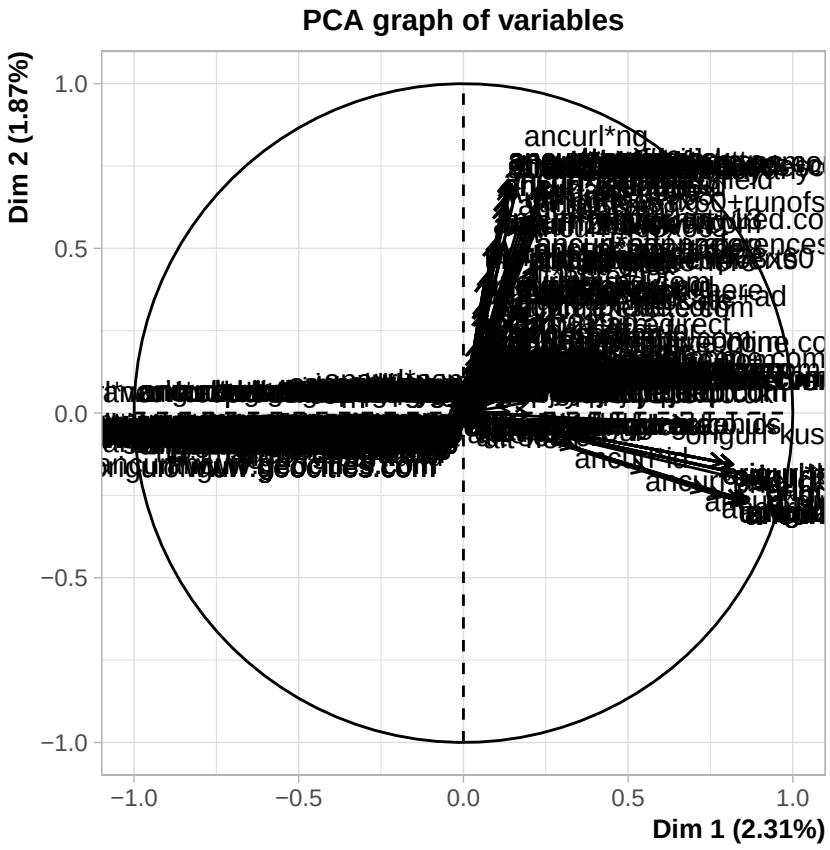


At 73 components, we can explain 60% of the variance.

Now, we plot the PCA graph of individuals and of variables below:

```
n <- length(data)
res.pca = PCA(data[,6:n-1])
```


PCA graph of individuals
Dim 2 (1.87%)
Dim 1 (2.31%)
80
40
0
−40
0
25
50
75
100
47
17
215
298
54
58
57



Knowledge Discovery

We try Support Vector Machines to predict if an image is an ad or not. We split the data to train and test sets:

```
smp_size <- floor(0.75 * nrow(data))
set.seed(123)
train_ind <- sample(seq_len(nrow(data)), size = smp_size)

data_train <- data[train_ind, ]
data_test <- data[-train_ind, ]
```

```
svm_radial <- tune(svm, ad ~., data = data_train, kernel = "radial")
svm_radial$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = ad ~ ., data = data_train, kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.0003528582
##
```

```
## Number of Support Vectors:  798
```

```
confusionMatrix(predict(svm_radial$best.model, data_test),
                data_test$ad)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction ad. nonad.
##     ad.     93      2
##     nonad.  28    697
##
##                Accuracy : 0.9634
##                  95% CI : (0.9482, 0.9752)
##     No Information Rate : 0.8524
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8404
##
##  Mcnemar's Test P-Value : 5.01e-06
##
##             Sensitivity : 0.7686
##             Specificity : 0.9971
##          Pos Pred Value : 0.9789
##          Neg Pred Value : 0.9614
##              Prevalence : 0.1476
##          Detection Rate : 0.1134
##    Detection Prevalence : 0.1159
##       Balanced Accuracy : 0.8829
##
##        'Positive' Class : ad.
##
```

From here, we can see that we have achieved 96% accuracy for our model, as we only failed to classify 30 of the ads. We try a linear kernel:

```
svm_linear <- tune(svm, ad ~., data = data_train, kernel = "linear")
svm_linear$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = ad ~ ., data = data_train, kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##       gamma:  0.0003528582
##
## Number of Support Vectors:  328
```

```
confusionMatrix(predict(svm_linear$best.model, data_test),
                data_test$ad)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction ad. nonad.
##     ad.    100     10
##     nonad.  21    689
##
##                Accuracy : 0.9622
##                  95% CI : (0.9468, 0.9742)
##     No Information Rate : 0.8524
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8439
##
##  Mcnemar's Test P-Value : 0.07249
##
##             Sensitivity : 0.8264
##             Specificity : 0.9857
##          Pos Pred Value : 0.9091
##          Neg Pred Value : 0.9704
##              Prevalence : 0.1476
##          Detection Rate : 0.1220
##    Detection Prevalence : 0.1341
##       Balanced Accuracy : 0.9061
##
##        'Positive' Class : ad.
##
```

Lastly, we try a polynomial kernel:

```
svm_poly <- tune(svm, ad ~., data = data_train, kernel = "polynomial")
svm_poly$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = ad ~ ., data = data_train, kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  3
##       gamma:  0.0003528582
##      coef.0:  0
##
## Number of Support Vectors:  865
```

```
confusionMatrix(predict(svm_poly$best.model, data_test),
                data_test$ad)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction ad. nonad.
##     ad.     50      1
##     nonad.  71    698
##
```

```
##                Accuracy : 0.9122
##                  95% CI : (0.8907, 0.9307)
##     No Information Rate : 0.8524
##     P-Value [Acc > NIR] : 1.809e-07
##
##                   Kappa : 0.5413
##
##  Mcnemar's Test P-Value : 4.232e-16
##
##             Sensitivity : 0.41322
##             Specificity : 0.99857
##          Pos Pred Value : 0.98039
##          Neg Pred Value : 0.90767
##              Prevalence : 0.14756
##          Detection Rate : 0.06098
##    Detection Prevalence : 0.06220
##       Balanced Accuracy : 0.70590
##
##        'Positive' Class : ad.
##
```

We check the best model in terms of the accurancy. For the SVM technique, we have the following accuracy metrics:

Model	**Accuracy**
Radial Kernel	96.3%
Linear Kernel	96.2%
Polynomial Kernel	91.2%

From here, we can see that the best model in terms of accuracy is the SVM model with a Linear kernel, having 96.3% accuracy. However, this is a close call with the one with a Radial Kernel. Now, we look at the value of specificity:

Model	**Specificity**
Radial Kernel	0.9971
Linear Kernel	0.9857
Polynomial Kernel	0.9986

From the table above, we can see that the Radial Kernel has a better specificity. As such, since the accuracy of the Radial and Linear kernel difference is very small, we would choose the Linear Kernel SVM to classify ads based on the attibutes provided.