Anomaly detection using the PCA algorithm.

Monika Baloda

Contents

#Problem Motivation#

Anomaly detection has many different applications. You can use it to find bad quality data. when it comes to customer surveys anomalies can pull out surveys where people spend more time taking the survey and contain the most informative information (most survey respondents tend to rush through surveys, it's odd to say but anomaly respondents tend to think more through the survey). Normally surveys have a huge amount of multicollinearity as people tend to click through them quickly, interestingly enough much of that multicollinearity is reduced through this anomaly algorithm. This makes building insightful supervised models with survey data much easier.

Our Strategy

We will be doing anomaly detection using PCA.

- The outline of the process:

1) Calculate the Principal Components
2) Remove the weakest principal components
3) Reconstruct the data with removing the weakest principal components
4) Calculate all the distances from the reconstructed data from the original data, the data that is the farthest away from the original data are considered the anomalies

Setup

- Initial Setup, there is no need to change anything here

```
library(tidyverse) # dplyr, tidyr, stringr, ggplot2, ...
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.1     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.0     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(ISLR2)

data(Caravan) # We will be the Caravan

Caravan <-
  Caravan %>%
  select(-MOSTYPE,-Purchase) #Remove Categorical Columns for simplicity

#Ensure results are consistent, do not change this number
set.seed(208) # do NOT change this number
```

Step 1: Run a PCA analysis

- Use the prcomp() function to run a PCA analysis

```
#Insert answer here
comp <- prcomp(Caravan,scale=F,center = F)
```

- The PCA analysis has several outputs

comp$rotation will output the rotation matrix that is used to transform the original data into the principal component form of the data

comp$sdev will output the eigenvalues, this will be useful in understanding how much information each principal component offers.

comp$x will output the principal component version of the data (will have same number of rows and columns as original data)

- You can use the principal component data and rotation matrix to transform back into the original data if you use all the principal components you should get the full original data back by doing the matrix multiplication Principal*Transpose(Rotation)

Step 2: Verify you got the matrix multiplying correct by checking you get the original data back

- You may need to round the data since these computations can be off by an extremely small amount

```
rot_matrix=comp$rotation

pc_data=comp$x

retrived_orig_data = pc_data%*%t(rot_matrix)



##Testing : check if retrived original data is same as original data#
num_of_columns_to_test=20

#randomly select 5 columns to test
random_indices = sample(0:ncol(Caravan), num_of_columns_to_test, replace = F)


for (i in 1:length(random_indices)){
  col1=retrived_orig_data[, i]
  col2=Caravan[, i]

  #let's do a t-test to know whether two columns are the same
  p_value = t.test(col1, col2)$p.value
  alpha <- 0.05  # Set the significance level (alpha)

  # Print the conclusion based on the p-value and significance level
  if (p_value <= alpha) {
  cat("The column number ", random_indices[i], " is significantly different in
                                  original and pca retrived dataframes", "\n")
  }
  else {
  cat("The column number ", random_indices[i], " is NOT significantly different
                                in original and pca  retrived dataframes","\n")
  }
}
```

```
## The column number  12  is NOT significantly different
##                                  in original and pca  retrived dataframes
## The column number  17  is NOT significantly different
##                                  in original and pca  retrived dataframes
## The column number  72  is NOT significantly different
##                                  in original and pca  retrived dataframes
## The column number  70  is NOT significantly different
##                                  in original and pca  retrived dataframes
## The column number  18  is NOT significantly different
##                                  in original and pca  retrived dataframes
## The column number  26  is NOT significantly different
##                                  in original and pca  retrived dataframes
## The column number  40  is NOT significantly different
##                                  in original and pca  retrived dataframes
```

```
## The column number  2  is NOT significantly different
##                                  in original and pca  retrived dataframes
## The column number  65  is NOT significantly different
##                                  in original and pca  retrived dataframes
## The column number  62  is NOT significantly different
##                                  in original and pca  retrived dataframes
## The column number  28  is NOT significantly different
##                                  in original and pca  retrived dataframes
## The column number  16  is NOT significantly different
##                                  in original and pca  retrived dataframes
## The column number  31  is NOT significantly different
##                                  in original and pca  retrived dataframes
## The column number  41  is NOT significantly different
##                                  in original and pca  retrived dataframes
## The column number  52  is NOT significantly different
##                                  in original and pca  retrived dataframes
## The column number  57  is NOT significantly different
##                                  in original and pca  retrived dataframes
## The column number  36  is NOT significantly different
##                                  in original and pca  retrived dataframes
## The column number  46  is NOT significantly different
##                                  in original and pca  retrived dataframes
## The column number  68  is NOT significantly different
##                                  in original and pca  retrived dataframes
## The column number  51  is NOT significantly different
##                                  in original and pca  retrived dataframes
```

Step 3: Use (comp$sdev)^2 to figure out how to remove the bottom weakest principal components.

- The best way to do this is to create a data frame storing all the squared values
- Add an id number to identify which principal components it corresponds to. Use row_number() to help with this
- Use the cumsum() function to calculate the cumulative sum of all the squared values
- Divide the cumulative sum column by the max cumulative sum value to convert into %'s
- Find the principal component number that corresponds to retaining at least 95% of the information
- Use the filter command to filter Percent >= 0.95 and check which ID is the first of that filtered dataframe

```
squared_values <- (comp$sdev)^2

# Create a dataframe to store the squared values with an ID number
df <- data.frame(ID = 1:length(squared_values), Squared_Value = squared_values)

# Calculate the cumulative sum of the squared values and percentage of cumulative sum
df$Cumulative_Sum <- cumsum(df$Squared_Value)
df$Percent <- df$Cumulative_Sum / max(df$Cumulative_Sum)

# Find the principal component number that corresponds to retaining at least 95% of the information
```

```r
percent_of_info_to_be_retained=95  #my code is general and you can set the percentage to be retained
retained_pc <- min(df$ID[df$Percent >= percent_of_info_to_be_retained/100])

# Filter the dataframe to find the first ID that meets the condition
filtered_df <- df %>% filter(Percent >= percent_of_info_to_be_retained/100)

# Get the ID of the first row in the filtered dataframe
retained_pc_id <- filtered_df$ID[1]

# Print the retained principal component number and ID
print(paste("Minimum Number of Retained Principal Components required to retain"
            ,percent_of_info_to_be_retained, "percent of information are:",retained_pc))
```

```
## [1] "Minimum Number of Retained Principal Components required to retain 95 percent of information ar
```

```r
print(paste("The number of removed Weakest PCs are:",  ncol(comp$x)-retained_pc))
```

```
## [1] "The number of removed Weakest PCs are: 69"
```

Step 4: Reconstruct the data with only using the top principal components

- Copy the working matrix multiplication from above, but this time only use the first X columns in the matrix algebra where X is the number of principal components chosen above

```r
number_of_pcs_to_keep=retained_pc

rot_matrix=comp$rotation[1:number_of_pcs_to_keep,1:number_of_pcs_to_keep]
pc_data=comp$x[,1:number_of_pcs_to_keep]

top_pc_data = pc_data%*%t(rot_matrix)

dim(pc_data)
```

```
## [1] 5822   15
```

Step 5: For each observation, calculate the distance from the original data to the reconstructed data

- You can either write a loop to calculate distances or by doing matrix algebra
- Matrix Algebra formula: sqrt(rowSums((A - B)^2))
- Save the distances into a new data frame

```r
#compute distances for each row (observation) in all columns
distances=sqrt(rowSums((Caravan - top_pc_data)^2))

# Create a new data frame to store the distances
distance_df <- data.frame(Distance = distances)
print(head(distance_df)) #Print the distance data frame
```

```
##   Distance
## 1 32.75832
## 2 28.78374
## 3 26.23306
## 4 24.60535
## 5 31.57419
## 6 24.61225
```

Step 6: Calculate the 95th percentile of the distances, anything greater than this value will be anomalies.

- You can use the quantile() function to calculate the 95th percentile
- Then write an ifelse() statement flagging anything greater than this value

```
# Calculate the 95th percentile
percentile_95 <- quantile(distance_df, probs = 0.95, na.rm=T)

# Flag anomalies using ifelse statement
anomaly_flags <- ifelse(distance_df > percentile_95, "Anomaly", "Normal")

# Create a new data frame with distances and anomaly flags
anamoly_df <- data.frame(Distance = distance_df, Anomaly = anomaly_flags)

# Count the number of anomalies
anomaly_count <- table(anomaly_flags)["Anomaly"]
cat("The percentage of anamolies present in the data are:",
                              round(100*(anomaly_count/length(distances)),2))
```

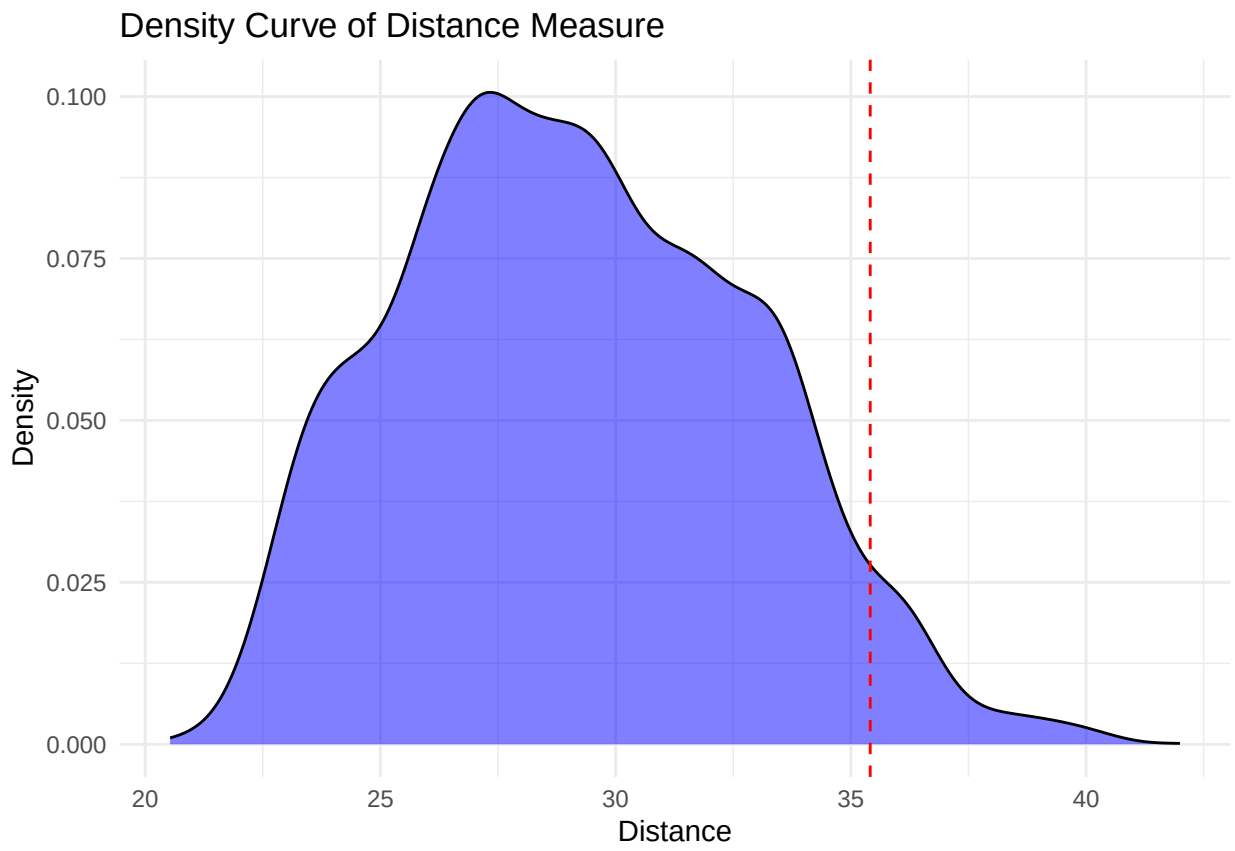
```
## The percentage of anamolies present in the data are: 5.02
```

Step 7: Use ggplot() to plot a density curve of the distances

- You can also append a vertical line at the 95th percentile mark

```
library(ggplot2)

# Plot density curve and vertical line
ggplot(data = distance_df, aes(x = Distance)) +
  geom_density(fill = "blue", alpha = 0.5) +
  geom_vline(xintercept = percentile_95, color = "red", linetype = "dashed") +
  labs(x = "Distance", y = "Density") +
  ggtitle("Density Curve of Distance Measure") +
  theme_minimal()
```



Step 8: Append the original data with the anomalies

- You can use bind_cols() to bind the anomalies

```
# Convert anomaly_flags to a data frame
anomalies_df <- data.frame(Anomaly = anomaly_flags)

# Append the original data with the anomalies
Appended_Caravan<- bind_cols(Caravan, anomalies_df)
head(Appended_Caravan[,80:85])  #making sure Distance anamoly flag column is added
```

```
##   AZEILPL APLEZIER AFIETS AINBOED ABYSTAND Distance
## 1       0        0      0       0        0   Normal
## 2       0        0      0       0        0   Normal
## 3       0        0      0       0        0   Normal
## 4       0        0      0       0        0   Normal
## 5       0        0      0       0        0   Normal
## 6       0        0      0       0        0   Normal
```

Step 9: We can group by anomalies and summarise_all to see if anything stands out with the group

- Do these observations stand out with any particular variables?

Summarizing the anomalies

```
library(dplyr)
library(tidyr)

grouped_data <- Appended_Caravan %>%group_by(Distance)
summary_data <- as.data.frame(grouped_data %>%summarise_all(funs(mean)))

# Convert from wide to long format\
cols=names(Caravan)
long_df <- as.data.frame(pivot_longer(summary_data, cols = cols,
                                      names_to = "variable", values_to = "value"))

#We now divide the mean value of a variable in Anomaly to total value.
        #If there is no difference, this should be 0.5
ratio_df <- as.data.frame(long_df %>%  group_by(variable) %>% mutate(ratio = value / sum(value)) )

standout_variables=ratio_df %>%filter(ratio > 0.75) %>%pull(variable)

print("The names of standout variables are:")
```

```
## [1] "The names of standout variables are:"
```

```
print(standout_variables)
```

```
##  [1] "PWERKT"   "PZEILPL"  "AWERKT"   "AZEILPL"  "MRELSA"   "MRELOV"
##  [7] "MFALLEEN" "MOPLHOOG" "MSKD"     "MINK123M" "PVRAAUT"  "AVRAAUT"
```