Week-4

Rajesh Kalakoti, Sven nomm

2023-08-03

1 Local Outlier Factors

LOF is a normalized density based approach Denote $V^k(X)$ - distance to it's k- -nearest neighbour and $L_k(X)$ the set of points within the set of points within k-nearest neighbor distance of X

- Reachability distance of X with respect to Y is defined as:

$$R_k\left(X,Y\right) = max\left\{S(X,Y), V^k(Y)\right\}$$

- The average reachability distance:

$$AR_k(X) = \frac{\sum_{y \in L_k(X)} R_k(X,Y)}{|Y \in L_k(X)|}$$

- Local Outlier Factor:

$$LOF_k(X) = \frac{\sum_{y \in L_k(X)} \frac{AR_k(X)}{AR_k(Y)}}{|Y \in L_k(X)|}$$

1.1 K-nearest neighbor vector

```
k_nneigh_vect <- function(x,data,k) {
  temp <- as.matrix(data)
  numrow <- dim(data)[1L]
  dimnames(temp) <- NULL
  # subtract rowvector x from each row of data
  difference<- scale(temp, x, FALSE)

  # square and add all differences and then take the square root
  dtemp <- drop(difference^2 %*% rep(1, ncol(data)))
  dtemp <- sqrt(dtemp)

  # order the distances
  order_dist <- order(dtemp)
  nndist <- dtemp[order_dist]

  # find distance to k-nearest neighbor
  # uses k+1 since first distance in vector is a 0
```

```
  knndist <- nndist[k+1]

  # find neighborhood
  # eliminate first row of zeros from neighborhood
  neighborhood <- drop(nndist[nndist<=knndist])
  neighborhood <- neighborhood[-1]
  numneigh <- length(neighborhood)

  # find indexes of each neighbor in the neighborhood
  index.neigh <- order_dist[1:numneigh+1]

  # this will become the index of the distance to first neighbor
  num1 <- numneigh+3

  # this will become the index of the distance to last neighbor
  num2 <- numneigh+numneigh+2

  return(c(num1,num2,index.neigh,neighborhood))
}
```

1.2 k-distance neighborhood

```
dist_to_knn <- function(dataset,neighbors) {

  numrow <- dim(dataset)[1L]
  mxNN <- neighbors*2+2
  knndist <- matrix(0,nrow=mxNN,ncol=numrow)
  for (i in 1:numrow) {
    # find obervations that make up the k-distance neighborhood for observation dataset[i,]
    neighdist <- k_nneigh_vect(dataset[i,],dataset,neighbors)

    x <- length(neighdist)
    if (x > mxNN) {
      knndist <- rbind(knndist,matrix(rep(0,(x-mxNN)*numrow),ncol=numrow))
      mxNN <- x
    }
    knndist[1:x,i] <- neighdist
  }

  return(knndist[1:mxNN,])
}
```

1.3 Reachability Density

```
reachability_density<- function(distdata,k) {

  p <- dim(distdata)[2]
  lrd <- rep(0,p)
```

```
  for (i in 1:p) {
    j <- seq(3,3+(distdata[2,i]-distdata[1,i]))
    # compare the k-distance from each observation to its kth neighbor
    # to the actual distance between each observation and its neighbors
    numneigh <- distdata[2,i]-distdata[1,i]+1
    temp <- rbind(diag(distdata[distdata[2,distdata[j,i]],distdata[j,i]]),distdata[j+numneigh,i])

    #calculate reachability
    reach <- 1/(sum(apply(temp,2,max))/numneigh)
    lrd[i] <- reach
  }
  lrd
}
```

1.4 Calculate Local outlier Factor for each observation

```
local_outlier_factor <- function(data,k) {

  data <- as.matrix(data)

  # obtain the k nearest neighbors and their distance from each observation
  distdata <- dist_to_knn(data,k)
  p <- dim(distdata)[2L]

  # calculate the local reachability_densitydensity for each observation in data
  lrddata <- reachability_density(distdata,k)

  lof <- rep(0,p)

  # computer the local outlier factor of each observation in data
  for (i in 1:p) {
    nneigh <- distdata[2,i]-distdata[1,i]+1
    j <- seq(0,(nneigh-1))
    local.factor <- sum(lrddata[distdata[3+j,i]]/lrddata[i])/nneigh
    lof[i] <- local.factor
  }

  # return lof, a vector with the local outlier factor of each observation
  lof
}
```

2 Local Outlier Factor for Anomaly Detection in Fraud Detection

This dataset is a synthetic dataset generated using the simulator called PaySim. The dataset contains financial transactions with fraud observations. For more details, you can check here.

2.1 Read Data File.

```
fraud = read.csv(fraud_data_file)
str(fraud)
```

```
## 'data.frame':    6362620 obs. of  11 variables:
##  $ step          : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ type          : chr  "PAYMENT" "PAYMENT" "TRANSFER" "CASH_OUT" ...
##  $ amount        : num  9840 1864 181 181 11668 ...
##  $ nameOrig      : chr  "C1231006815" "C1666544295" "C1305486145" "C840083671" ...
##  $ oldbalanceOrg : num  170136 21249 181 181 41554 ...
##  $ newbalanceOrig: num  160296 19385 0 0 29886 ...
##  $ nameDest      : chr  "M1979787155" "M2044282225" "C553264065" "C38997010" ...
##  $ oldbalanceDest: num  0 0 0 21182 0 ...
##  $ newbalanceDest: num  0 0 0 0 0 ...
##  $ isFraud       : int  0 0 1 1 0 0 0 0 0 0 ...
##  $ isFlaggedFraud: int  0 0 0 0 0 0 0 0 0 0 ...
```

```
# top 10 rows of data frame
head(fraud,10)
```

```
##    step     type   amount    nameOrig oldbalanceOrg newbalanceOrig    nameDest
## 1     1  PAYMENT  9839.64 C1231006815      170136.0      160296.36 M1979787155
## 2     1  PAYMENT  1864.28 C1666544295       21249.0       19384.72 M2044282225
## 3     1 TRANSFER   181.00 C1305486145         181.0           0.00  C553264065
## 4     1 CASH_OUT   181.00  C840083671         181.0           0.00   C38997010
## 5     1  PAYMENT 11668.14 C2048537720       41554.0       29885.86 M1230701703
## 6     1  PAYMENT  7817.71   C90045638       53860.0       46042.29  M573487274
## 7     1  PAYMENT  7107.77  C154988899      183195.0      176087.23  M408069119
## 8     1  PAYMENT  7861.64 C1912850431      176087.2      168225.59  M633326333
## 9     1  PAYMENT  4024.36 C1265012928        2671.0           0.00 M1176932104
## 10    1    DEBIT  5337.77  C712410124       41720.0       36382.23  C195600860
##    oldbalanceDest newbalanceDest isFraud isFlaggedFraud
## 1               0           0.00       0              0
## 2               0           0.00       0              0
## 3               0           0.00       1              0
## 4           21182           0.00       1              0
## 5               0           0.00       0              0
## 6               0           0.00       0              0
## 7               0           0.00       0              0
## 8               0           0.00       0              0
## 9               0           0.00       0              0
## 10          41898       40348.79       0              0
```

The data consist of more than 2263777 million observations and 11 variables. Since the data is large, cleaning the dataset beforehand is a prerequisite for efficiency as the computer will have to deal with a large amount of data.

The first thing we need to remove the variables that do not contribute to the investigation. The variable of nameOrig and nameDest consist of a vast amount of unique values. We create a new object to differentiate from the uncleaned dataset and drop these two variables as our first step.

```r
fraud_clean <- fraud %>%  select(-c(nameOrig, nameDest))
head(fraud_clean,10)
```

```
##    step     type   amount oldbalanceOrg newbalanceOrig oldbalanceDest
## 1     1  PAYMENT  9839.64      170136.0      160296.36              0
## 2     1  PAYMENT  1864.28       21249.0       19384.72              0
## 3     1 TRANSFER   181.00         181.0           0.00              0
## 4     1 CASH_OUT   181.00         181.0           0.00          21182
## 5     1  PAYMENT 11668.14       41554.0       29885.86              0
## 6     1  PAYMENT  7817.71       53860.0       46042.29              0
## 7     1  PAYMENT  7107.77      183195.0      176087.23              0
## 8     1  PAYMENT  7861.64      176087.2      168225.59              0
## 9     1  PAYMENT  4024.36        2671.0           0.00              0
## 10    1    DEBIT  5337.77       41720.0       36382.23          41898
##    newbalanceDest isFraud isFlaggedFraud
## 1            0.00       0              0
## 2            0.00       0              0
## 3            0.00       1              0
## 4            0.00       1              0
## 5            0.00       0              0
## 6            0.00       0              0
## 7            0.00       0              0
## 8            0.00       0              0
## 9            0.00       0              0
## 10       40348.79       0              0
```

```r
fraud_clean <- fraud_clean %>%
  mutate(type = as.factor(type),
         isFraud = as.factor(isFraud))
```

2.2 Data Exploration

```r
transaction <- c("fraud","genuine")
value <- c(sum(fraud_clean$isFraud == 1), sum(fraud_clean$isFraud == 0))
percentage <- c(sum(fraud_clean$isFraud == 1)/length(fraud_clean$isFraud)*100,
                sum(fraud_clean$isFraud == 0)/length(fraud_clean$isFraud)*100)
information <- data.frame(transaction,value,percentage)
information
```

```
##   transaction   value percentage
## 1       fraud    8213   0.129082
## 2     genuine 6354407  99.870918
```

The fraud transaction is tiny compared to the genuine transaction. It only contains less than 1% of the overall data. We try to visualize the whole transactions based on the transaction type.

```r
total_fraud <- fraud_clean %>%
  filter(isFraud == 1) %>%
  select(type) %>%
  group_by(type) %>%
```

```
  count(type) %>%
  arrange(desc(n))
```

```
fraud_clean_real <- fraud_clean %>%
  filter(type == "CASH_OUT" | type == "TRANSFER")
dim(fraud_clean_real)
```

```
## [1] 2770409       9
```

```
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
set.seed(11)
library(rsample)
fraud_split <- initial_split(data = fraud_clean_real, prop = 0.7, strata = isFraud)
fraud_train <- training(fraud_split)
fraud_test <- testing(fraud_split)
```

```
fraud_scale <- fraud_clean_real %>% select(-c(isFraud, isFlaggedFraud, type))
fraud_scale <- as.data.frame(scale(fraud_scale))
```

2.3 compute Local outlier factor values for 1000

```
fraud_clean_real_1000 <- head(fraud_clean_real,1000)
fraud_scale_1000 <- head(fraud_scale,1000)
```

lof values for 1000 data points.

```
local_factor_values = local_outlier_factor(fraud_scale_1000,3)
```

appendind lof values to original data frame.

```
fraud_clean_real_1000$lof <- local_factor_values
```

To gain more understanding of the data, we will see the distribution between fraud and genuine transactions. Since the dataset consists of more than two variables, we can use PCA and use the first two dimensions of the PCA.

```
library(FactoMineR)
fraud_pca <- PCA(fraud_scale_1000, scale.unit = F, ncp = 6, graph = F)
summary(fraud_pca)
```

```
##
## Call:
## PCA(X = fraud_scale_1000, scale.unit = F, ncp = 6, graph = F)
```

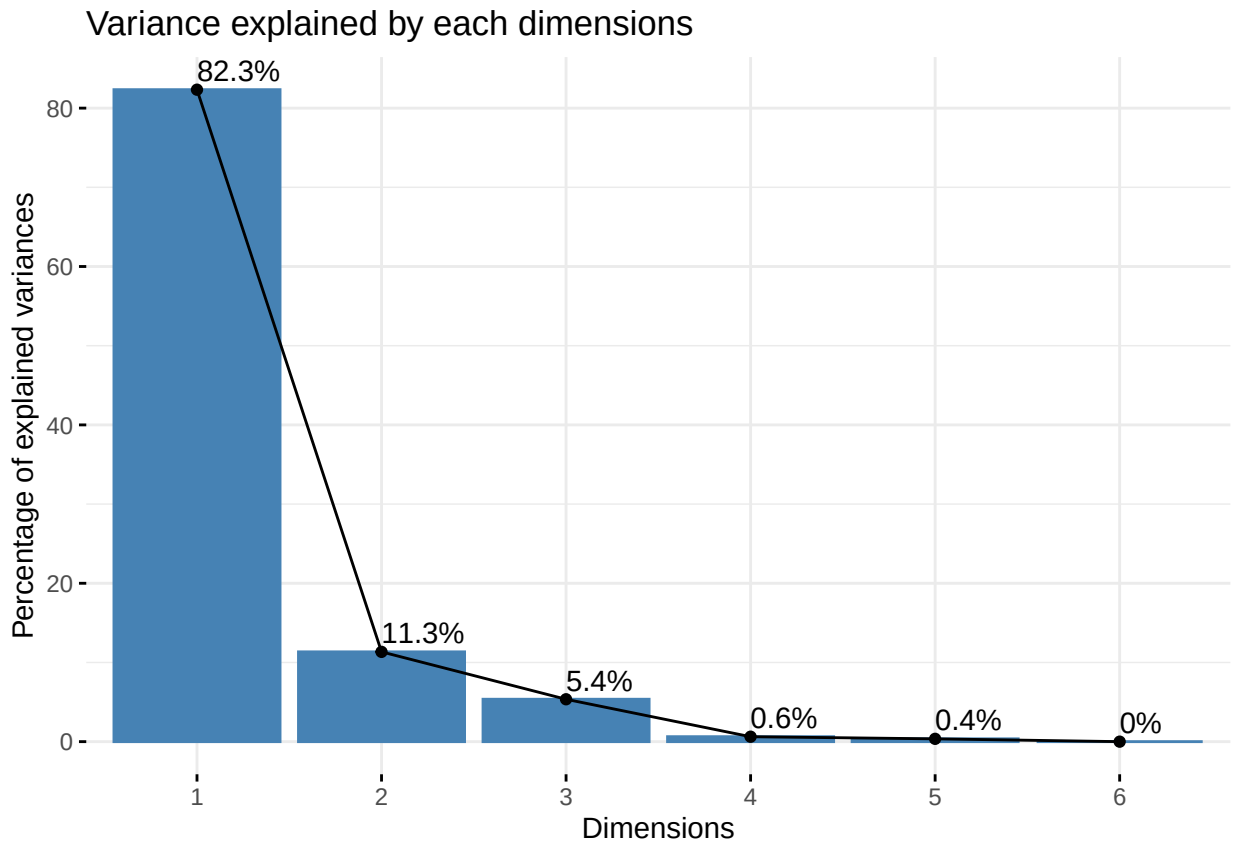
```
##
##
## Eigenvalues
##                        Dim.1   Dim.2   Dim.3   Dim.4   Dim.5   Dim.6
## Variance              17.791   2.450   1.157   0.135   0.077   0.000
## % of var.             82.326  11.339   5.353   0.626   0.356   0.000
## Cumulative % of var.  82.326  93.665  99.018  99.644 100.000 100.000
##
## Individuals (the 10 first)
##                     Dist    Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3
## 1                |  1.003 | -0.718  0.003  0.512 | -0.208  0.002  0.043 | -0.654
## 2                |  1.001 | -0.718  0.003  0.514 | -0.208  0.002  0.043 | -0.651
## 3                |  0.913 | -0.665  0.002  0.531 | -0.080  0.000  0.008 | -0.615
## 4                |  0.948 | -0.704  0.003  0.552 | -0.126  0.001  0.018 | -0.616
## 5                |  0.784 | -0.673  0.003  0.737 | -0.016  0.000  0.000 | -0.166
## 6                |  0.891 | -0.642  0.002  0.519 | -0.091  0.000  0.010 | -0.605
## 7                |  0.967 | -0.710  0.003  0.539 | -0.181  0.001  0.035 | -0.625
## 8                |  1.199 | -0.721  0.003  0.361 | -0.091  0.000  0.006 |  0.489
## 9                |  0.954 | -0.663  0.002  0.484 | -0.144  0.001  0.023 | -0.655
## 10               |  0.804 | -0.425  0.001  0.279 | -0.036  0.000  0.002 | -0.667
##                     ctr   cos2
## 1                 0.037  0.425 |
## 2                 0.037  0.423 |
## 3                 0.033  0.453 |
## 4                 0.033  0.422 |
## 5                 0.002  0.045 |
## 6                 0.032  0.461 |
## 7                 0.034  0.418 |
## 8                 0.021  0.166 |
## 9                 0.037  0.472 |
## 10                0.038  0.688 |
##
## Variables
##                      Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr
## step              |  0.000  0.000  0.000 |  0.001  0.000  0.021 |  0.000  0.000
## amount            |  0.209  0.245  0.097 |  0.520 11.040  0.600 |  0.155  2.068
## oldbalanceOrg     |  2.785 43.591  0.866 |  1.086 48.135  0.132 | -0.116  1.158
## newbalanceOrig    |  3.161 56.163  0.909 | -0.991 40.059  0.089 |  0.095  0.778
## oldbalanceDest    | -0.006  0.000  0.000 |  0.056  0.128  0.006 |  0.667 38.486
## newbalanceDest    | -0.008  0.000  0.000 |  0.125  0.638  0.022 |  0.816 57.509
##                    cos2
## step              0.000 |
## amount            0.053 |
## oldbalanceOrg     0.001 |
## newbalanceOrig    0.001 |
## oldbalanceDest    0.887 |
## newbalanceDest    0.933 |
```

2.4 visualize variance of PCA

```
library("factoextra")
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

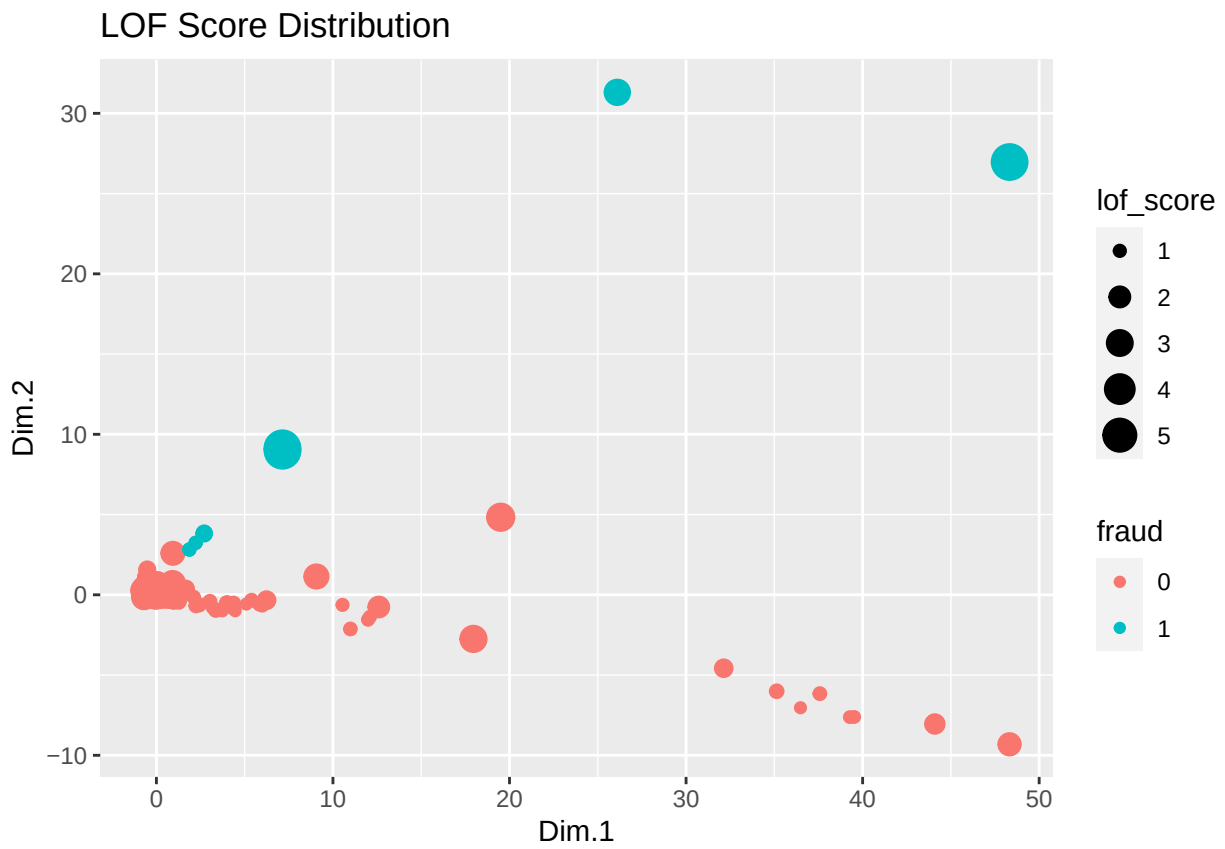
```
fviz_eig(fraud_pca, ncp = 6, addlabels = T, main = "Variance explained by each dimensions")
```



The result from PCA above shows that if we use the first two dimensions of the data, we still retain 98% variance from the original data. The first three dimensions along with the LOF score and fraud label is obtained and stored in the new data frame.

```
fraud_a <- data.frame(fraud_pca$ind$coord[,1:3])
fraud_b <- cbind(fraud_a, fraud = fraud_clean_real_1000$isFraud, lof_score = fraud_clean_real_1000$lof)
fraud_lof_visual <- ggplot(fraud_b, aes(x=Dim.1 ,y=Dim.2, color=fraud)) +
    geom_point(aes(size=lof_score)) +
  ggtitle("LOF Score Distribution")

fraud_lof_visual
```



From the visualization above, we can see that genuine and fraudulent transactions have different patterns. The higher the lof score it has, the dot is bolder and more prominent.

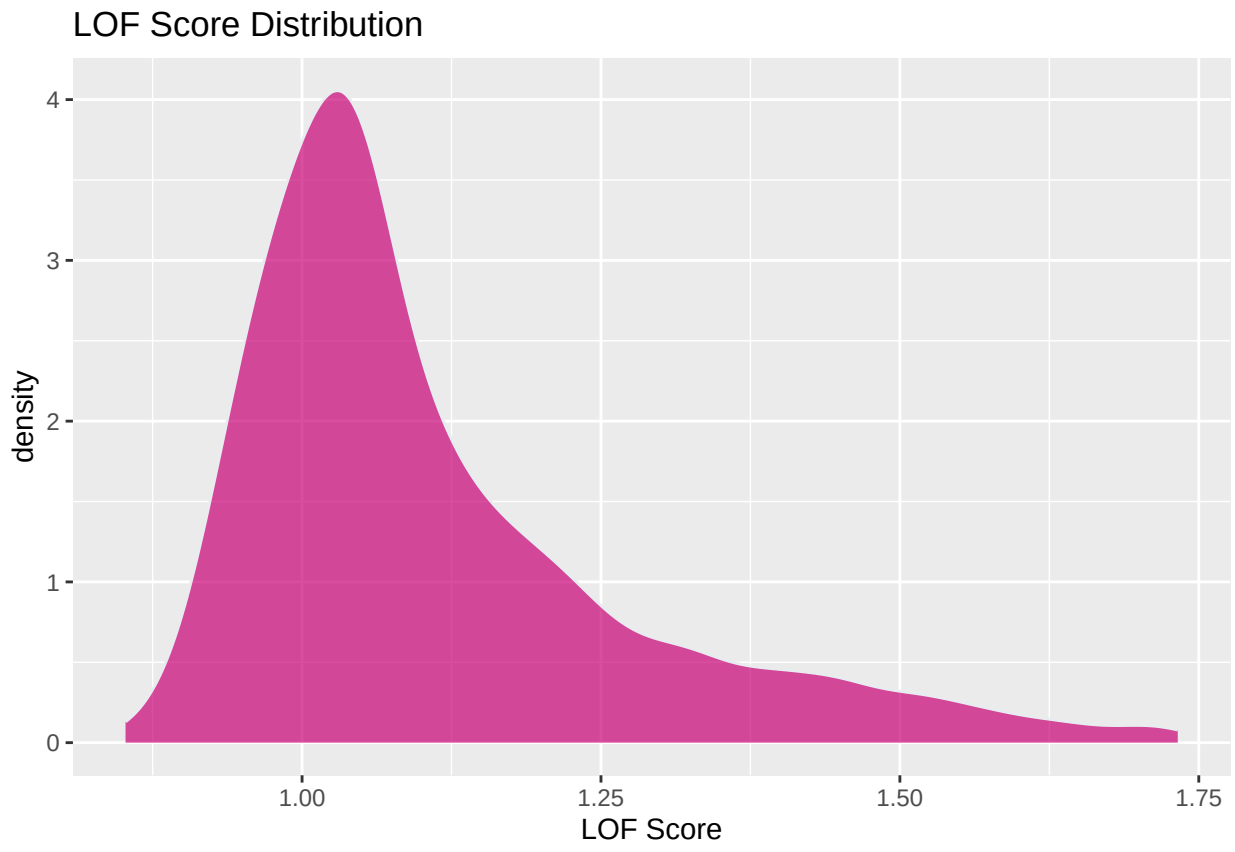
The rule of thumb of the lof score says that if the LOF score is more than 1, it is likely to be an outlier. Somehow, a threshold can be adjusted with the distribution of the data. Let's see the statistics of the LOF score first.

```
summary(fraud_b)
```

```
##      Dim.1              Dim.2              Dim.3          fraud
##  Min.   :-0.7260   Min.   :-9.31453   Min.   :-2.86211   0:968
##  1st Qu.:-0.7130   1st Qu.:-0.16675   1st Qu.:-0.60722   1: 32
##  Median :-0.7018   Median :-0.09728   Median :-0.44904
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.00000
##  3rd Qu.:-0.6443   3rd Qu.: 0.03268   3rd Qu.: 0.07354
##  Max.   :48.3272   Max.   :31.30889   Max.   : 5.32294
##    lof_score
##  Min.   :0.8524
##  1st Qu.:1.0026
##  Median :1.0631
##  Mean   :1.1723
##  3rd Qu.:1.2059
##  Max.   :5.9601
```

```
fraud_b %>%
  filter(lof_score <= 1.75) %>%
```

```
  ggplot( aes(x=lof_score)) +
    geom_density( color="#e9ecef", fill = "#c90076", alpha=0.7) +
    scale_fill_manual(values="#8fce00") +
    xlab("LOF Score")+
  ggtitle("LOF Score Distribution")
```




```
    labs(fill="")
```

```
## $fill
## [1] ""
##
## attr(,"class")
## [1] "labels"
```

We see above, the LOF score have many points with the score more than 1. To classify a point as an outlier or not, we can set the threshold higher.

One method to determine threshold is calculating the quantile point. Here, we will set threshold 90% as the normal points, while the last 10% is considered as outlier. The threshold proportion can be adjusted depend on the business case. If user wish to more cautios with the LOF score, user can set the threshold higher.

```
quantile(fraud_b$lof_score, probs = c(0, 0.9))
```

```
##        0%       90%
## 0.8523534 1.4561876
```

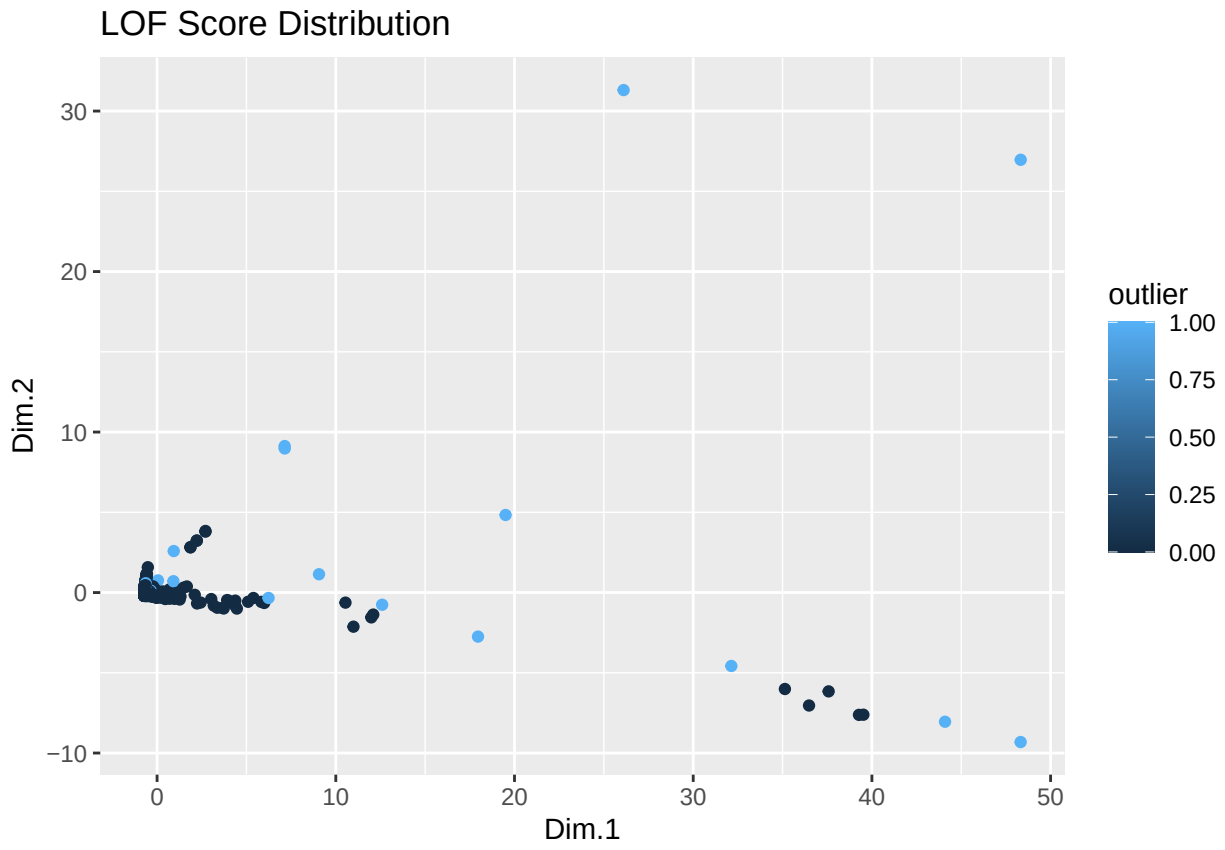
The 90% proportion of the LOF score falls under below 1.4561876. We will use this threshold to determine if a point falls under the threshold; we categorize that point as an outlier.

```
fraud_b <- fraud_b %>%
  mutate(outlier = ifelse(lof_score > 1.4561876 , 1, 0))
```

We can once again visualize the distribution of the outlier for all observations.

```
fraud_lof_visual_b <- ggplot(fraud_b, aes(x=Dim.1 ,y=Dim.2, color=outlier)) +
    geom_point() +
  ggtitle("LOF Score Distribution")

fraud_lof_visual_b
```



3 ISOLATION forest for Fraud detection.

The algorithm of Isolation Forest works the same by targeting the observation that stands alone after the isolation is applied to the observation. The observation far from the rest will be identified as unusual rather than gathered in groups that share the similarity. The more points travel from the groups, the more indication we can consider that as an anomaly. The isolation forest will choose attributes originating from the data in random and recursively partition the observation of the maximum and minimum value of the selected features. The observation that requires less partition will likely be an anomaly than the observation that requires more separation.

```
library(isotree)
fraud_isotree <- isolation.forest(fraud_train %>% select(-isFraud), sample_size = 64)
```

The code will build the algorithm with the default parameter. The default parameters are:

- **sample_size:** The number of sample-size of data sub-samples with which each binary tree will be built. The default for this parameter is the length of the dataset. For memory management, we will try to use 64 sample sizes as the first attempt.
- **ntrees:** Number of binary trees to build for the model. The default for these parameters is 10.

The Isolation Forest Algorithm will produce an anomaly score within the range of 0 to 1. The interpretation of an anomaly score can be determined here:

- If the anomaly score is close to 1, it can be interpreted that the point is an anomaly.
- If the anomaly score is minimal compared to 0.5, it can be interpreted as a regular data point.
- If the anomaly score produced, all of them, is 0.5, then it can be confirmed that there is no anomaly for the sampling dataset.

```
fraud_score <- predict(fraud_isotree, newdata = fraud_train %>% select(-isFraud))
fraud_train$score <- fraud_score
```

we will divide the anomaly score with the a spesific threshold. If the anomaly score is more than 80% quantile for the rest of data, we will clasify the point as an anomaly. This threshold can be tolerated with respective subject business matter.

```
quantile(fraud_train$score, probs = c(0, 0.8))
```

```
##        0%       80%
## 0.3512429 0.4865171
```

```
fraud_train <- fraud_train %>%
  mutate(fraud_detection = as.factor(ifelse(fraud_score >= 0.4821847 , 1, 0)))
```

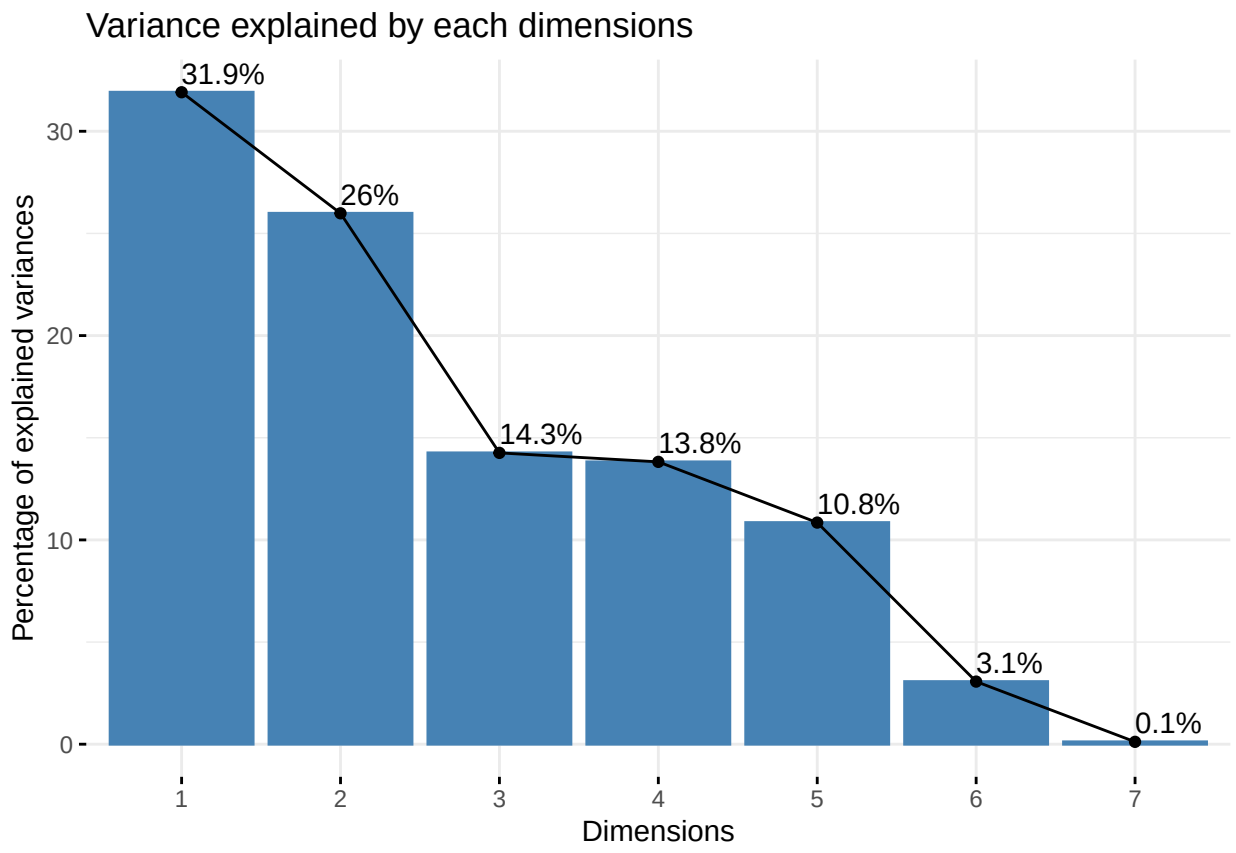
3.1 Visualizing The Isolation Forest Score

A contour plot can visualize the score produced by the Isolation Forest algorithm. The visualization will give us more insight into the polarization of the anomaly score. Since the visualization is limited to fewer dimensions, we will compress the data into two-dimension using PCA; then, we will take two dimensions produced by the PCA and visualize it in a contour plot provided by the lattice library.

```
library(lattice)
library(FactoMineR)
library(factoextra)

fraud_pca_train <- PCA(fraud_train %>% select(step, amount, oldbalanceOrg, newbalanceOrig, oldbalanceDes
```

```
fviz_eig(fraud_pca_train, ncp = 9, addlabels = T, main = "Variance explained by each dimensions")
```




```
pca_grid <- as.data.frame(fraud_pca_train$ind)
```

```
pca_grid <- pca_grid %>%
  select(coord.Dim.1, coord.Dim.2)
```

```
d1_seq <- seq(min(pca_grid$coord.Dim.1), max(pca_grid$coord.Dim.1), length.out = 100)
d2_seq <- seq(min(pca_grid$coord.Dim.2), max(pca_grid$coord.Dim.2), length.out = 100)

fraud_train_grid <- expand.grid(d1 = d1_seq, d2 = d2_seq)
head(fraud_train_grid)
```

```
##           d1        d2
## 1 -0.7894322 -7.939856
## 2  0.2520342 -7.939856
## 3  1.2935005 -7.939856
## 4  2.3349669 -7.939856
## 5  3.3764333 -7.939856
## 6  4.4178996 -7.939856
```

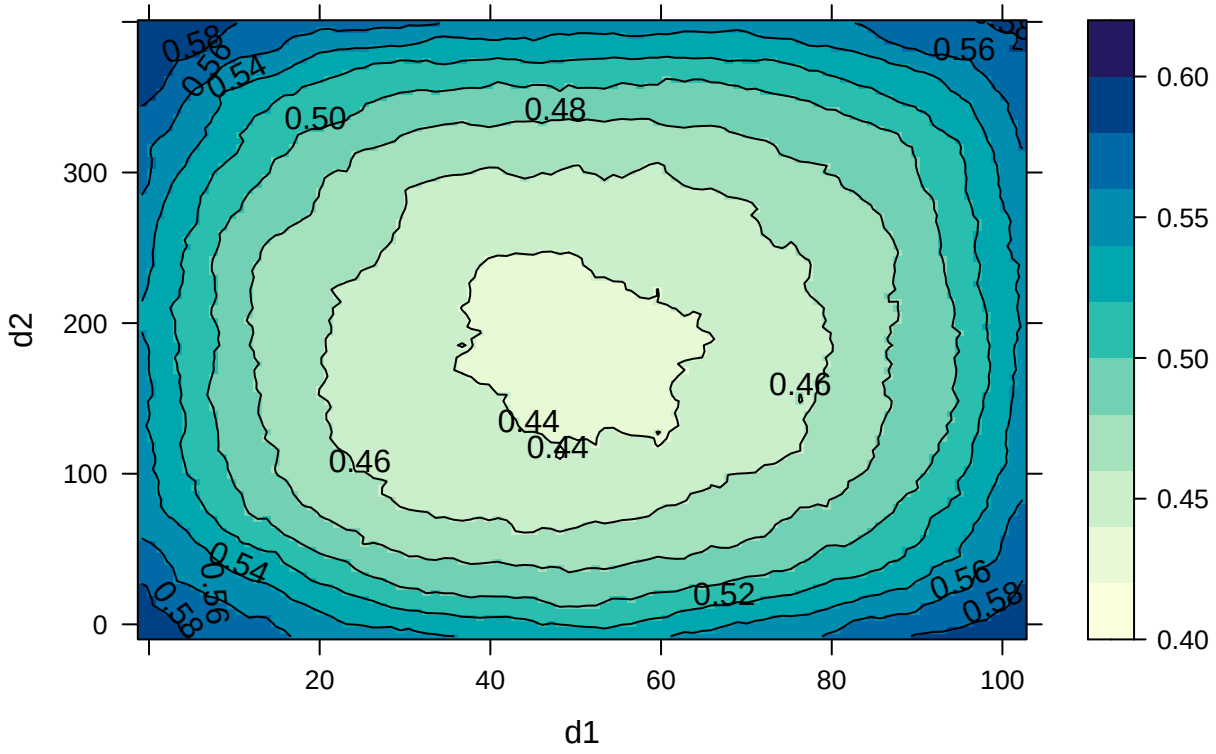
```
library(isotree)
```

```r
pca_isotree <- isolation.forest(fraud_train_grid, sample_size = 64)

fraud_train_grid$score <- predict(pca_isotree, fraud_train_grid)
```

```r
library(lattice)
# Define a custom color palette

# Create the contour plot with the custom color palette
contourplot(score ~ d1 + d2, fraud_train_grid, region = TRUE)
```



middle white space represents the points polarizing as the normal data transactions; here, we called genuine transactions. In contrast, the darker blue area is the region that far from the normal instances. The Isolation Forest will take the points that fall far from the normal point polarization as an anomaly.