Fraudulent database study

Carolina Armella, Ágatha del Olmo and María Yu García

2024-10-04

Index

1.1: Introduction

Credit card transaction fraud has become one of the most concerning problems for both consumers and financial institutions globally. This project aims to develop an effective fraud detection system using advanced machine learning techniques.

Our main goal is to create a highly accurate predictive model, capable of identifying fraudulent transactions by minimizing both false positives and false negatives, but especially false negatives. The importance of this project lies in its potential to:

• Protect consumers from financial loss. • Improve efficiency in fraud detection, allowing faster and more accurate responses. • Reduce losses for financial institutions. • Maintain customer confidence in electronic payment systems.

The database used in this project contains actual credit card transactions made by Europeans over a two-day period in September 2013. It includes a total of *284807 transactions*, of which only *492* were classified as fraudulent, which represents only *0.172%* of the total. This low proportion of frauds reflects a *significant imbalance* in the data.

Moreover, the dataset includes numerical variables that have been transformed using Principal *Component Analysis (PCA),* which means that the original variables related to transactions are not directly accessible for confidentiality reasons. These transformed variables are labeled V1 *to V28* and represent the main components of the PCA that explain most of the variability in the original data.

In addition, there are two characteristics that have not been transformed:

• *Time*: Indicates the time in seconds between each transaction and the first recorded transaction.

• *Amount*: Represents the total value of the transaction.

As the database sampling was performed over a two-day period without specifying which days they are, the variable loses some relevance in terms of its ability to identify patterns related to the time of day, week, or specific events. It would be interesting to have a more precise temporal variable indicating the day of the week, the time or even the month, to explore whether there are specific patterns related to the day or time on which transactions are made, which could help identify differences in behavior between fraudulent and non-fraudulent cases. For this reason, most of the study will ignore this variable.

Finally, the answer variable, *Class*, takes binary values:

• *1* for fraudulent transactions

• *0* for legitimate transactions

In this study, we will use classification models with the aim of identifying fraudulent transactions (class 1). We will compare the performance of three types of models: neural networks, KNN (k-nearest neighbors), and decision trees. We chose these models because of their characteristics: neural networks are very effective at capturing complex patterns, KNN is simple and effective with nonlinear data, and decision trees perform well with unbalanced data.

Although neural networks are "black box" models and their interpretation is difficult, in this case we are not concerned, since we are working with variables resulting from a PCA, and we do not know the original ones, which implies that they are no longer directly interpretable. Therefore, our approach is mainly focused on the predictive capacity of the model, without the need for an explicit interpretation of the underlying characteristics.

In this paper, fraud detection models will be evaluated using three main metrics: accuracy, recall, and F1-score. Accuracy will focus on minimizing false positives, while recall will prioritize detecting as many frauds as possible, albeit with some false positives. The F1-score will combine both aspects, providing a balance between accuracy and recall. In addition, the confusion matrix will be used to analyze errors and adjust models as needed. These metrics will allow you to effectively measure the performance of your models in a context of unbalanced data.

This project not only seeks to create a functional fraud detection system, but also to explore best practices in the handling of unbalanced data and the interpretation of models in a highly sensitive context such as financial fraud detection.

1.2: Description of the Problem

The objective of this project is to predict whether a transaction is fraudulent (class 1) or not (class 0), which is crucial to reduce economic losses and improve the security of payment systems.

We'll start with a thorough exploratory data analysis (EDA). First, we will load our selected sample and analyze each variable in detail, reviewing its types and making any adjustments as necessary. This step is essential to understand the structure of our data, identify patterns, detect anomalies, and explore the relationships between different characteristics.

During the EDA, we will focus on the analysis of the variables, as well as the identification and treatment of outliers. We'll also look at how transactions vary over time. To do this, we will use the support of graphs that will allow us to represent the complex patterns of the data in a more visual and simple way, which will help us to obtain valuable information about the nature of transactions, both fraudulent and legitimate.

As for the implementation, this will include the loading and preprocessing of the data, followed by an initial exploration and analysis. Subsequently, the predictive models will be built, evaluated and the best model will be selected based on its performance.

In a real-world environment, we anticipate several challenges, such as extreme class imbalance, rapidly evolving fraud patterns, managing false positives and consequent customer discontent, and the need for real-time detection, as well as data privacy and security considerations. To meet these challenges, we would implement periodic retraining systems, establish human review processes for doubtful cases, and optimize our models for real-time prediction, ensuring that we comply with strict privacy regulations.

1.3: Pre-process

Loading the data

After loading the necessary libraries, we load the credit card transaction dataset.

```
fraud <- read_csv("creditcard.csv")
dim(fraud)
```

```
[1] 284807     31
```

Stratified sampling

To begin with, we seek to reduce computational complexity, since it is a database with many observations (284807). We selected a random sample of 10000 observations using stratified sampling, i.e. we divided the set of observations into subsets according to whether they are class 0 or 1, guaranteeing the same percentage of cases of each type as in the original database (proportional fixation). This is essential because although it is much less common for a movement to be fraudulent, we need the model

to learn and evaluate taking into account that these cases exist, as they are precisely the ones we are interested in finding. Within each stratum we choose at random, using set.seed(1313) to ensure reproducibility, allowing future analyses with the same code to generate the same random sample. As we saw at the end, the sample and the original database have the same structure according to the Class.

```
prop_original <- sum(fraud$Class==1)/nrow(fraud)

prop_0 <- sum(fraud$Class == 0) / nrow(fraud)
prop_1 <- sum(fraud$Class == 1) / nrow(fraud)
prop_0+prop_1==1
```

```
[1] TRUE
```

```
muestra_clase_1 <- round(prop_1 * 10000)
muestra_clase_0 <- round(prop_0 * 10000)
muestra_clase_1+muestra_clase_0==10000
```

```
[1] TRUE
```

```
set.seed(1313)
muestra_clase_1 <- fraud %>%
  filter(Class == 1) %>%
  sample_n(muestra_clase_1, replace = TRUE)

set.seed(1313)
muestra_clase_0 <- fraud %>%
  filter( Class == 0) %>%
  sample_n(muestra_clase_0, replace = FALSE)

fraud <- bind_rows(muestra_clase_0, muestra_clase_1)

table(fraud$Class)
```

```
   0 1
9983 17
```

```
prop_nueva <- sum(fraud$Class==1)/nrow(fraud)
round(prop_original,4)==prop_nueva
```

```
[1] TRUE
```

```
dim(fraud)
```

```
[1] 10000    31
```

Then, we check for missing values in the dataset using the anyNA(fraud) function, in this case there are none, so you don't need to process them. We save the sample of the dataset in a . RData for later uses.

```
anyNA(fraud)
```

```
[1] FALSE
```

```
save(fraud, file = "fraud. RData")
```

Variable factor

The Class variable is the predictor variable in this dataset, indicating whether a transaction is fraudulent or not. Originally it is numerical, but since it represents categories (0 normal transaction, 1 fraud), we converted it into a factor for its correct use in classification models.

```
load("fraud. RData")

class(fraud$Class)
```

```
[1] "numeric"
```

```
fraud$Class <- as.factor(fraud$Class)

class(fraud$Class)
```

```
[1] "factor"
```

High correlations

Now we begin to analyze the relationships between predictors. First, we look at correlation, a statistical measure that measures the strength and direction of the relationship between two variables. It is useful to detect high correlations between variables, as strongly correlated variables can cause problems such as multicollinearity. However, we expect them to be low since the variables come from a Principal Component analysis and we hope that all redundancy has been eliminated.

```
corr <- abs(cor(fraud[,1:30])[upper.tri(cor(fraud[,1:30]))]) > .9
corr_altos <- sum(corr)
corr_altos
```

```
[1] 0
```

As expected, we found no correlations greater than 0.9, which means that there are no variables with extremely strong linear relationships. This is positive, as the risk of multicollinearity and overfitting in the model is low.

Linear dependencies

We now assess whether any of the variables may be an exact linear combination of other variables, which can cause problems in certain models. If a variable is a linear combination of others, we could be introducing redundancy into the model, which would affect interpretability and could cause some algorithms to not work properly, as the information would be duplicated.

```
findLinearCombos(fraud)
```

```
$linearCombos
list()

$remove
NULL
```

We have not found exact linear combinations (extreme multicollinearity), which supports previous results that there are no excessively high correlations between variables, as the variables come from a PCA process. We can conclude that each of the variables provides unique information, that is, they maximize variability.

Typification

We may be interested in carrying out a typification or standardization, that is, the process of transforming the variables so that they have a mean of 0 and a standard deviation of 1. This procedure is particularly useful for models that assume that variables have a normal distribution, as is the case of Linear Discriminant Analysis (LDA) and other linear models.

To verify that the typing has been carried out correctly, we check that each variable has a mean close to 0 and a standard deviation of 1. This ensures that variables are on a similar scale and that no variable has a disproportionate impact due to their original scale.

```
fraud_tip <- fraud

for (i in 1:30) {
  fraud_tip[[i]] <- (fraud[[i]] - mean(fraud[[i]], na.rm = TRUE)) / sd(fr
aud[[i]], na.rm = TRUE)
}

head(fraud_tip,2)

# A tibble: 2 × 31
    Time     V1    V2     V3     V4    V5     V6    V7     V8     V9    V
10
   <dbl>  <dbl> <dbl>  <dbl>  <dbl> <dbl>  <dbl> <dbl>  <dbl>  <dbl>  <db
l>
1  0.269 -0.246 0.529 -0.190 -0.131 0.168 -0.636 1.18  -0.507  1.39  -0.2
85
2 -1.16  -0.270 0.502  0.574  0.166 0.768 -0.106 0.937 -0.258 -0.713  0.1
58
# ℹ 20 more variables: V11 <dbl>, V12 <dbl>, V13 <dbl>, V14 <dbl>, V15 <d
bl>,
#   V16 <dbl>, V17 <dbl>, V18 <dbl>, V19 <dbl>, V20 <dbl>, V21 <dbl>,
#   V22 <dbl>, V23 <dbl>,  V24 <dbl>, V25 <dbl>, V26 <dbl>, V27 <dbl>,
#   V28 <dbl>, Amount <dbl>, Class <fct>

comp_media <- numeric(30)
for (i in 1:30) {
```

```
  comp_media[i] <- mean(fraud_tip[[i]])
}
round(comp_media,2)
```

```
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
comp_sd <- numeric(30)
for (i in 1:30) {
  comp_sd[i] <- sd(fraud_tip[[i]])
}
comp_sd
```

```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Normalization to [0,1]

In some cases, it may also be desirable to normalize the variables to a range of [0, 1], which implies a different transformation of the variables. This type of adjustment is particularly useful for models that depend on distances, such as Neural Networks or K Nearest Neighbors (KNN).

Normalization to [0, 1] is especially relevant when variables have very different units from each other, as is the case with the Amount variable (measured in euros) compared to the others. The standard deviation of this variable is considerably higher than that of other variables, which could result in the model dominating if not adjusted.

In this normalization process, we transform the variables so that they are all in a common range of [0, 1], preventing variables with larger scales (such as Amount) from influencing more than they should. This step helps all variables to have a similar weight when they are entered into the model.

```
round(apply(fraud[,2:30], 2, sd),2)
```

```
    V1     V2     V3     V4     V5     V6     V7     V8     V9    V10
V11
  2.05   1.87   1.54   1.44   1.41   1.35   1.23   1.08   1.10   1.10   1
.01
   V12    V13    V14    V15    V16    V17    V18    V19    V20    V21
V22
  0.98   1.00   0.91   0.92   0.86   0.79   0.84   0.82   0.92   0.68   0
.72
   V23    V24    V25    V26    V27    V28 Amount
  0.59   0.60   0.53   0.48   0.42   0.38 302.61
```

```
fraud_nor <- fraud

for (i in 1:30) {
    fraud_nor[[i]] <- (fraud[[i]] - min(fraud[[i]], na.rm = TRUE)) / (max
(fraud[[i]], na.rm = TRUE) - min(fraud[[i]], na.rm = TRUE))
  }
```

```
head(fraud_nor, 2)
```

```
# A tibble: 2 × 31
   Time    V1    V2    V3    V4    V5    V6    V7    V8    V9   V10   V11
V12
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
<dbl>
1 0.628 0.923 0.798 0.869 0.218 0.373 0.568 0.584 0.681 0.635 0.634 0.323
0.599
2 0.236 0.922 0.798 0.903 0.238 0.389 0.586 0.579 0.690 0.502 0.648 0.423
0.793
# ℹ 18 more variables: V13 <dbl>, V14 <dbl>, V15 <dbl>, V16 <dbl>, V17 <d
bl>,
#   V18 <dbl>, V19 <dbl>, V20 <dbl>, V21 <dbl>, V22 <dbl>, V23 <dbl>,
#   V24 <dbl>, V25 <dbl>, V26 <dbl>, V27 <dbl>, V28 <dbl>, Amount <dbl>,
#   Class <fct>
```

```
round(apply(fraud_nor[,2:30], 2, sd),2)
```

```
    V1     V2     V3     V4     V5     V6     V7     V8     V9    V10
V11
  0.05   0.02   0.04   0.07   0.03   0.03   0.02   0.04   0.06   0.03   0
.09
   V12    V13    V14    V15    V16    V17    V18    V19    V20    V21
V22
  0.06   0.12   0.06   0.12   0.07   0.04   0.08   0.07   0.02   0.03   0
.06
   V23    V24    V25    V26    V27    V28 Amount
  0.02   0.10   0.04   0.09   0.02   0.02   0.02
```

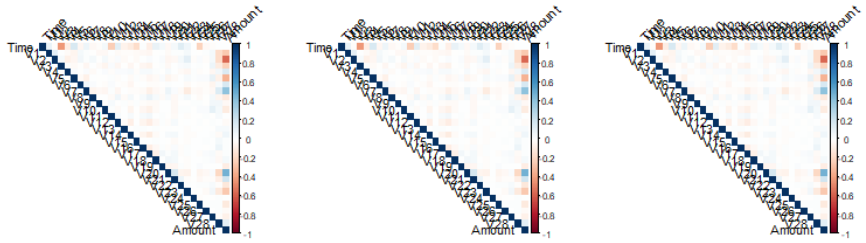
At the end of normalization, all variables have more similar standard deviations, ensuring that no variable is unbalanced in terms of its influence on the model.

Check

Finally, we use correlation graphs to compare the relationships between variables in the different versions of the data:

• Original data. • Data normalized to [0.1]. • Typed data (mean 0, standard deviation 1).

Correlation graphs allow us to visually see how the relationships between variables remain consistent even after preprocessing. This is important to make sure that scaling and standardization have not distorted the relationships between variables.



1.4: Initial Exploratory Data Analysis

We have decided to typify the variables for this study, since the KNN and Neural Network models depend on distances and weights. If the variables have different scales, some may dominate the model, skewing the results. Typing ensures that all variables have equal weight. In the case of Decision trees, typing is not required, as they divide observations based on thresholds of variables, regardless of their scale. However, typifying can help improve the consistency of divisions across nodes.

Division into training set and test set

Now we divide the data into training set and test set, we use createDataPartition from the caret package to do the division making sure that both sets maintain the same Class ratio, as we need it to be taken into account in the construction of the model and its predictive capacity analysis.

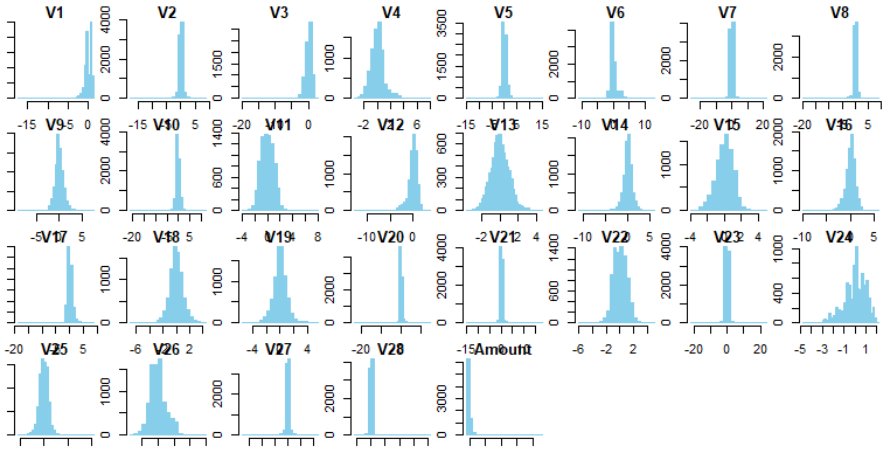
```
set.seed(1313)
trainIndex <- createDataPartition(fraud_tip$Class, p = .8,
                                  list = FALSE,
                                  times = 1)

fraud_Train <- fraud_tip[ trainIndex,]
fraud_Test  <- fraud_tip[-trainIndex,]

prop_train <- sum(fraud_Train$Class==1)/nrow(fraud_Train)
prop_test <- sum(fraud_Test$Class==1)/nrow(fraud_Test)

round(prop_train,3)==round(prop_test,3)

[1] TRUE

round(prop_test,3)==round(prop_original,3)

[1] TRUE
```

From here we completely forget about the test set, and work entirely with the training set.

Distributions of variables

Having divided the data, we analyzed the distributions of the variables in the dataset to better understand their characteristics.

```
par(mfrow = c(4, 8), mar = c(1, 1, 1, 1))

for (var in names(fraud_Train)[2:30]) {
  hist(fraud_Train[[var]], main = paste( var),
       xlab = var, col = "skyblue", border = "skyblue", breaks = 30)
}

par(mfrow = c(1, 1))
```



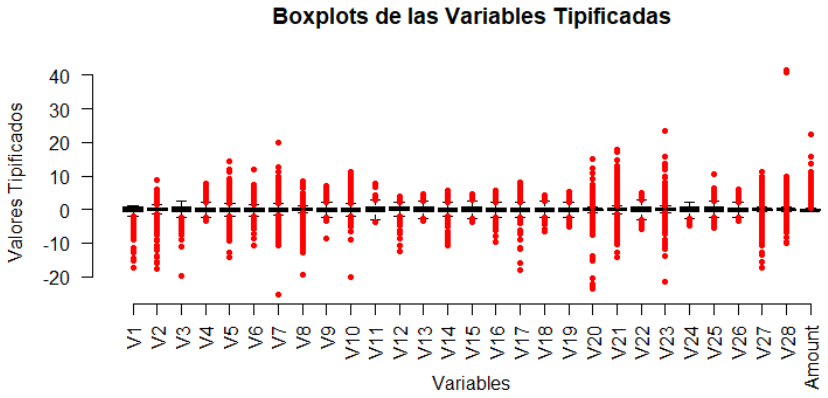
It seems that most of the variables have fairly symmetrical distributions, although there are some with a clear bias to the left or right. Several variables show peaks in specific values, these concentrations could be indicative of recurring behaviors in normal or fraudulent transactions. The variability between the distributions is high, which suggests that each variable provides unique information to the predictive model, which makes sense considering the previous PCA process.

It is relevant to note that some histograms reveal the presence of possible outliers, points that are very far from the general distribution, which we have decided to study further. Identifying these outliers is crucial, as they can have a major impact on the accuracy of the fraud detection model.

Outlier detection

Outliers Outliers (or outliers) are data that deviate significantly from the rest of the observations in a dataset, and can represent anomalies or unusual behaviors, essential for fraud detection. Statistically, observations are three times greater in absolute values than the standard deviation. We see them first on a plot.

```r
boxplot(fraud_Train[ , 2:30],
        col = "skyblue",
        outcol = "red",
        pch = 20,
        frame.plot = FALSE,
        main = "Boxplots de las Variables Tipificadas",
        ylab = "Valores Tipificados",
        xlab = "Variables",
        las = 2)

outliers_count <- sapply(fraud_Train[, 2:30],
                         function(x) sum(x < (mean(x, na.rm = TRUE) - 3 *
sd(x, na.rm = TRUE)) |
                                                 x > (mean(x, na.rm = TRUE) + 3
* sd(x, na.rm = TRUE))))
```



We now see the total number of outliers per variable, and what percentage they represent over the total number of observations.

```r
outliers_count
```

```
    V1     V2     V3     V4     V5     V6     V7     V8     V9    V10
V11
   102    117     56     85     80    120     93    154     61     90
19
   V12    V13    V14    V15    V16    V17    V18    V19    V20    V21
V22
   109     39    104     33     58     85     56    110    120    129
33
   V23    V24    V25    V26    V27    V28 Amount
   116     13     79     28    141     59    123
```

```r
round(outliers_count/nrow(fraud_Train)*100,2)
```

```
    V1     V2     V3     V4     V5     V6     V7     V8     V9    V10
V11
  1.27   1.46   0.70   1.06   1.00   1.50   1.16   1.92   0.76   1.12   0
```

```
.24
   V12    V13    V14    V15    V16    V17    V18    V19    V20    V21
V22
  1.36   0.49   1.30   0.41   0.72   1.06   0.70   1.37   1.50   1.61   0
.41
   V23    V24    V25    V26    V27    V28 Amount
  1.45   0.16   0.99   0.35   1.76   0.74   1.54
```

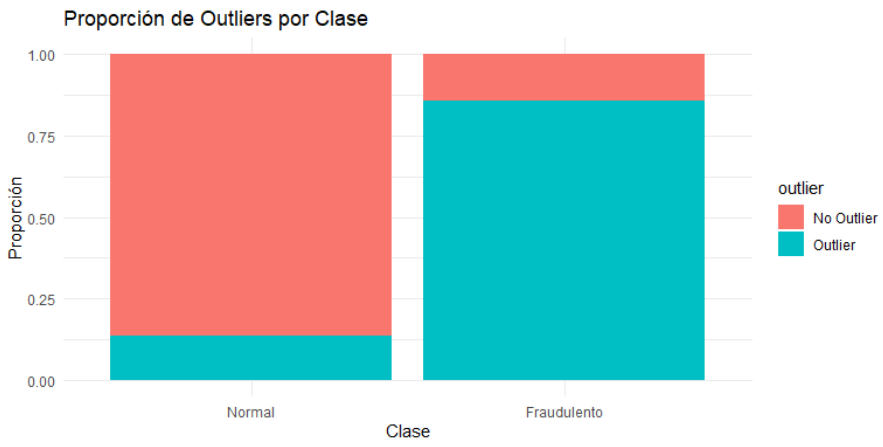
Most variables have a 0-2% percentage of outliers, with V8 and V27 showing the highest number with 1.92% (154) and 1.76% (141) respectively. This percentage is very small, and generally shouldn't have a big impact on predictive models, especially with a high amount of data. However, in the specific example we are studying, the effect that outliers can have on the model is especially interesting because frauds tend to have different or atypical characteristics compared to normal transactions. In this sense, as we have already mentioned, outliers could represent precisely those unusual behaviors that we are trying to identify. Therefore, we study which of these outliers are related to fraud.

```
outlier_class <- apply(fraud_Train[ , 2:30], 1, function(row) {
  any(row < -3 | row > 3)
})

fraud_Train %>%
  mutate(outlier_class = outlier_class) %>%
  group_by(Class, outlier_class) %>%
  summarise(outliers = n()) %>%
  mutate(pct = outliers / sum(outliers))

# A tibble: 4 × 4
# Groups:   Class [2]
  Class outlier_class outliers   pct
  <fct> <lgl>            <int> <dbl>
1 0     FALSE             6911 0.865
2 0     TRUE              1076 0.135
3 1     FALSE                2 0.143
4 1     TRUE                12 0.857
```

We conclude that, indeed, outliers play a crucial role in the detection of fraudulent cases. 86% of outliers correspond to fraudulent transactions, which highlights their importance as possible indications of fraud. This finding reinforces the need to pay special attention to outliers, as they not only represent outliers, but can also be a key signal for identifying fraudulent behaviors within the dataset. We can see this very clearly in the following plot:

```
fraud_Train %>%
  mutate(outlier = ifelse(rowSums(fraud_Train[,2:30] > 3 | fraud_Train[,2
:30] < -3) > 0, "Outlier", "No Outlier"),  Class = recode(Class, `0` = "N
ormal", `1` = "Fraudulento")) %>%
  ggplot(aes(x = Class, fill = outlier)) +
  geom_bar(position = "fill") +
```

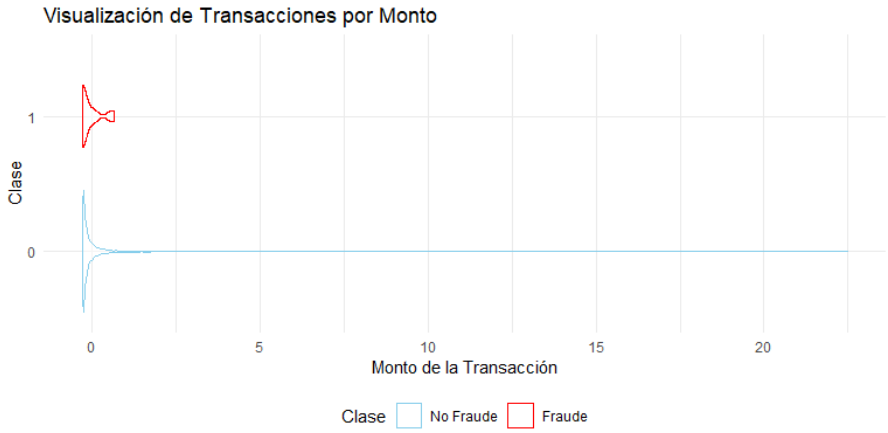
```
  labs(title = "Proporción de Outliers por Clase", y = "Proporción", x =
"Clase") +
  theme_minimal()
```



Relationship with the target variable

In this analysis, we explore the relationship between the transaction amount (Amount) and the transaction class (Class) using a violin chart, very useful for studying the distribution of numerical data categorized by classes.

```
ggplot(fraud_Train, aes(x = Amount, y = Class, color = as.factor(Class)))
+
  geom_violin(alpha = 0.5) +
  scale_color_manual(values = c("skyblue", "red"),
                     labels = c( "No Fraude", "Fraude")) +
  labs(title = "Visualización de Transacciones por Monto",
       x = "Monto de la Transacción",
       y = "Clase",
       color = "Clase") +
  theme_minimal() +
  theme(legend.position = "bottom")
```

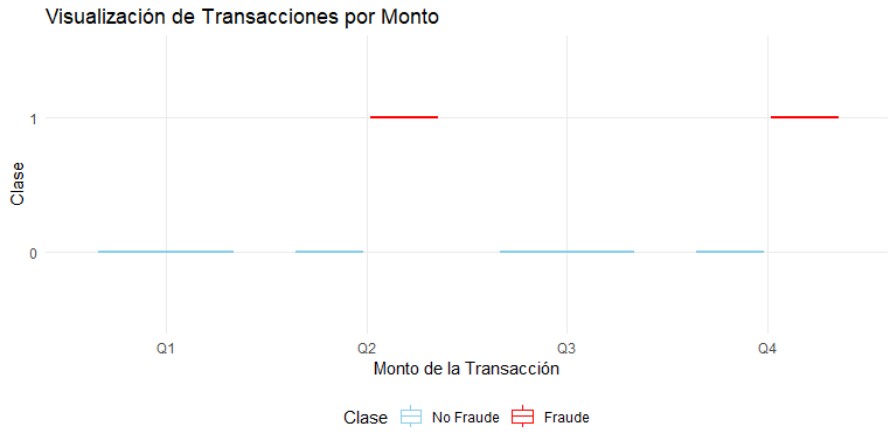


We note that fraudulent transactions (in red) tend to be concentrated in low amounts, while non-

fraudulent transactions (in blue) are distributed over a much wider range of amounts, although it is also quite concentrated in low amounts. This suggests that the transaction amount could be an important feature for fraud detection, as frauds seem to be more related to lower-value transactions.

Since the relationship between the amount and the class is not completely linear, we see if discretizing the Amount variable improves the clarity of the relationship between both classes according to the amount.

```
fraud_Train2 <- fraud_Test

Amount_discreta <-quantile(fraud_Train2$Amount, probs = seq(0, 1, by = 0.
25))

fraud_Train2$Amount <- cut(fraud_Train2$Amount,
                           breaks = Amount_discreta,
                           include.lowest = TRUE,
                           labels = c("Q1", "Q2", "Q3", "Q4"))


ggplot(fraud_Train2, aes(x = Amount, y = Class, color = as.factor(Class))
) +
  geom_boxplot(alpha = 0.5) +
  scale_color_manual(values = c("skyblue", "red"),
                     labels = c("No Fraude", "Fraude")) +
  labs(title = "Visualización de Transacciones por Monto",
       x = "Monto de la Transacción",
       y = "Clase" ,
       color = "Clase") +
  theme_minimal() +
  theme(legend.position = "bottom")

rm(fraud_Train2)
```



After discretizing the variable "Amount" into quartiles, we observe that fraudulent transactions (in red) tend to be concentrated in medium and high amounts. This

suggests that frauds are more associated with high values. In contrast, normal transactions (in blue) are more evenly distributed across all amount ranges, especially in the low and medium-high values. The discretization of the variable "Amount" facilitates the visualization of these differences and highlights the importance of this variable to detect fraud.

Correlations with the target variable

We study the correlations between the variables and the target variable Class, which allow us to identify which are more relevant to predict fraud. If a variable has a high correlation (positive or negative) with class, it is an indication that that variable may be a good predictor.

```
correlations<-numeric(29)

for (i in 2:30) {
  correlations[i-1] <- cor(fraud_Train[[i]], as.numeric(fraud_Train$Class
), method = "pearson")
}
round(correlations,2)
```

```
 [1] -0.07 0.07 -0.13 0.09 -0.06 -0.04 -0.13 -0.03 -0.07 -0.16 0.11 -0.20
[13] -0.01 -0.24 0.00 -0.11 -0.21 -0.06 0.01 0.01 -0.01 0.01 0.00 0.00
[25] -0.01 -0.01 -0.06 -0.01 0.00
```

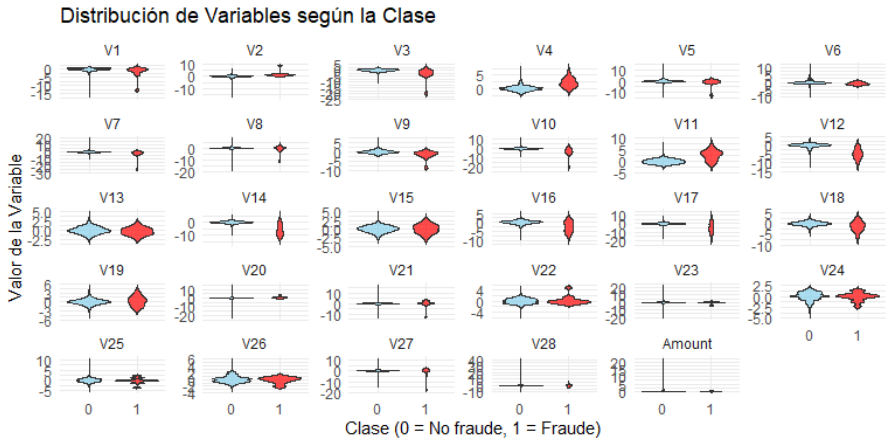
Although some variables seem to have some relationship to the class such as V7 (0.16), V11 (0.24), and V14 (0.21), the correlation is not strong enough to make clear predictions from them alone. This highlights the importance of using more advanced modeling methods (such as the ones we are considering, KNN, decision trees, and neural networks), which do not rely exclusively on linear correlation to make predictions.

Class distributions

Finally, we are interested in analyzing how the variables are distributed within each class. If we observe that a variable has a very different distribution between fraudulent and non-fraudulent transactions, we can deduce that this variable is interesting for classification.

```
fraud_long <- fraud_Train %>%
  pivot_longer(cols = 2:30, names_to = "Variable", values_to = "Value")

order_of_variables <- names(fraud_Train)[2:31]
fraud_long$Variable <- factor(fraud_long$Variable, levels = order_of_vari
ables)

ggplot(fraud_long, aes(x = Class, y = Value, fill = Class)) +
  geom_violin(trim = FALSE, alpha = 0.7) +
  facet_wrap(~ Variable, scales = "free_y") +
  theme_minimal() +
```

```
  labs(title = "Distribución de Variables según la Clase",
       x = "Clase (0 = No fraude, 1 = Fraude)",
       y = "Valor de la Variable") +
  scale_fill_manual(values = c(" skyblue", "red")) +
  theme(legend.position = "none")
```



When analyzing the violin charts, significant differences are observed in the distribution of fraudulent and non-fraudulent transactions. Variables such as V3, V4, V12, V14, V16 and V17 show clear disparities in terms of density and range between classes, suggesting that they could be relevant for fraud detection. In contrast, variables such as V2, V13, V15, and V26 show no obvious differences between fraudulent and non-fraudulent transactions.

1.5: Conclusion

In conclusion, our analysis of financial transactions for fraud detection reveals that, despite the unbalanced distribution of classes, relevant patterns can be identified that allow distinguishing between normal and fraudulent transactions. The standardization and normalization of variables, along with the analysis of outliers and correlations, have been crucial to prepare the data and minimize the impact of possible biases. Variables such as V7, V11 and V14, along with the amount of the transaction, emerged as key factors in the fraud classification. This process of data cleansing and transformation, together with the focus on the relationship between variables and the target class, establishes a solid foundation for the development of more accurate and efficient predictive models in fraud detection.

1.6: References

Credit card fraud Detection. (2018, March 23). https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/data

Kuhn, M. (2019, March 27). 4 Data Splitting | The caret Package. https://topepo.github.io/caret/data-splitting.html#simple-splitting-with-important-groups

Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson and Gianluca Bontempi. Calibrating Probability with Undersampling for Unbalanced Classification. In Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2015

Dal Pozzolo, Andrea; Caelen, Olivier; Le Borgne, Yann-Ael; Waterschoot, Serge; Bontempi, Gianluca. Learned lessons in credit card fraud detection from a practitioner perspective, Expert systems with applications,41,10,4915-4928,2014, Pergamon

Dal Pozzolo, Andrea; Boracchi, Giacomo; Caelen, Olivier; Alippi, Cesare; Bontempi, Gianluca. Credit card fraud detection: a realistic modeling and a novel learning strategy, IEEE transactions on neural networks and learning systems,29,8,3784-3797,2018,IEEE

Dal Pozzolo, Andrea Adaptive Machine learning for credit card fraud detection ULB MLG PhD thesis (supervised by G. Bontempi)

Carcillo, Fabrizio; Dal Pozzolo, Andrea; Le Borgne, Yann-Aël; Caelen, Olivier; Mazzer, Yannis; Bontempi, Gianluca. Scarff: a scalable framework for streaming credit card fraud detection with Spark, Information fusion,41, 182-194,2018,Elsevier

Carcillo, Fabrizio; Le Borgne, Yann-Aël; Caelen, Olivier; Bontempi, Gianluca. Streaming active learning strategies for real-life credit card fraud detection: assessment and visualization, International Journal of Data Science and Analytics, 5,4,285-300,2018,Springer International Publishing

Bertrand Lebichot, Yann-Aël Le Borgne, Liyun He, Frederic Oblé, Gianluca Bontempi Deep-Learning Domain Adaptation Techniques for Credit Cards Fraud Detection, INNSBDDL 2019: Recent Advances in Big Data and Deep Learning, pp 78-88, 2019

Fabrizio Carcillo, Yann-Aël Le Borgne, Olivier Caelen, Frederic Oblé, Gianluca Bontempi Combining Unsupervised and Supervised Learning in Credit Card Fraud Detection Information Sciences, 2019

Yann-Aël Le Borgne, Gianluca Bontempi Reproducible machine Learning for Credit Card Fraud Detection - Practical Handbook

Bertrand Lebichot, Gianmarco Paldino, Wissam Siblini, Liyun He, Frederic Oblé, Gianluca Bontempi Incremental learning strategies for credit cards fraud detection, IInternational Journal of Data Science and Analytics