Statistical Learning Project: fraud detection

Salima Tankibayeva

October 24, 2024

Abstract

Fraud detection is a critical task in the financial industry, requiring effective methods to identify fraudulent transactions. This report describes the development of a model using the XGBoost algorithm, a powerful machine learning method, to predict fraudulent credit card transactions. The analysis is based on the Credit Card Fraud Detection dataset, which includes transaction records with labels indicating fraud or non-fraud. This report explores the use of unsupervised learning techniques, specifically principal component analysis (PCA), to identify fraudulent transactions in a highly imbalanced credit card fraud detection dataset. The goal is to use unsupervised techniques to gain insights into the dataset, reduce dimensionality, and prepare for further machine learning model building. Analysis showed that while PCA helped visualize the data, it was unable to distinguish fraudulent from non-fraudulent transactions, highlighting the challenges associated with class imbalance. This paper explores the limitations of using PCA in isolation for this problem and proposes additional techniques to improve fraud detection in imbalanced datasets.

1 Introduction

The Credit Card Fraud Detection dataset offers a unique opportunity to explore both supervised and unsupervised learning approaches due to its characteristics such as class imbalance and PCA-transformed features. In a supervised learning project, we can build classification models to identify fraudulent transactions (1) versus legitimate ones (0) using different algorithms, while addressing the significant class imbalance (fraud accounts for only 0.172% of transactions) using methods such as cost-aware learning and AUPRC metrics. The V1-V28 features obtained from PCA allow for feature importance analysis, while the untransformed features, time and sum, can be used to improve model interpretability and performance. Conversely, an unsupervised learning project can focus on anomaly detection, identifying rare or novel fraud patterns without relying on labeled data. Methods such as clustering can identify unusual groupings of transactions, while dimensionality reduction methods can Ultimately, a hybrid approach combining supervised and unsupervised learning methods will result in a robust fraud detection system that can adapt to changing types of fraudulent behavior.

2 Data

The Credit Card Fraud Detection dataset contains transaction data from European cardholders over two days in September 2013, with a total of 284,807 transactions, but only 492 are fraudulent, making the data highly imbalanced (fraud accounts for just 0.172%). The features have been PCA-transformed for privacy, resulting in 28 variables labeled V1 to V28 that capture most of the original data's information. Two features, Time and Amount, are not transformed and represent the time since the first transaction and the transaction value, respectively. The target variable, Class, shows whether a transaction is fraudulent (1) or legitimate (0). Because of the imbalance, standard accuracy metrics are not suitable, and the Area Under the Precision-Recall Curve (AUPRC) was recommended for evaluation.

3 Supervised

3.1 Preprocessing

The analysis begins with loading the necessary libraries and dataset. The data.table package is used to efficiently process the data, while caret and xgboost facilitate model training and evaluation. The following code was used to load and explore the dataset:

```r
# Load necessary libraries
library(data.table)
library(caret)
library(xgboost)
library(doParallel)
library(ggplot2)
library(PRROC)

# Load the dataset using fread for faster loading
data <- fread("creditcard.csv")

# Data exploration
str(data)           # Check structure of the dataset
summary(data)       # Get basic statistics
table(data$Class)   # Check the distribution of the target variable
    (Class)

# Check for missing values
missing_percentage <- (colSums(is.na(data)) / nrow(data)) * 100
print(missing_percentage)
```

The dataset was split into training and testing sets to evaluate the model's performance accurately. The features were scaled to standardize the input data, which is essential for many machine learning algorithms, including XGBoost.

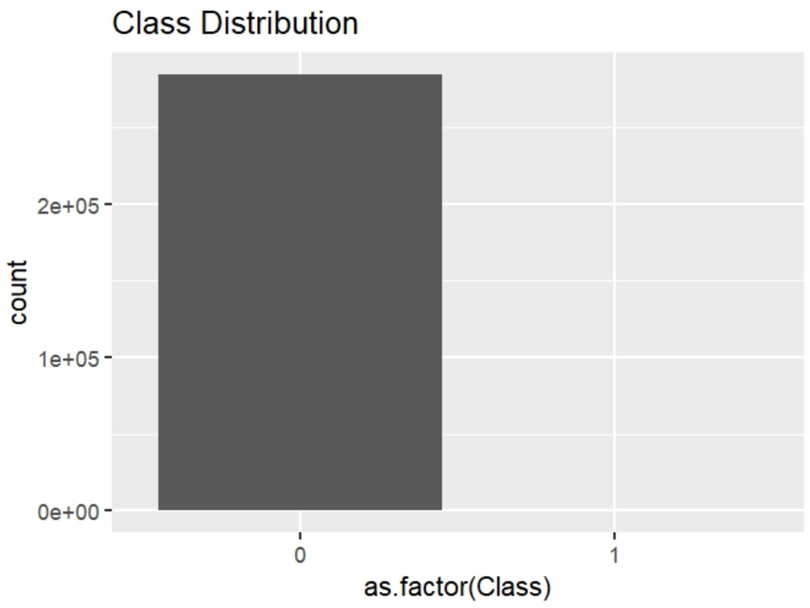

Figure 1: class distribution

3.2 Model training

The model was trained using the XGBoost algorithm with specified parameters to handle the binary classification problem. The scale_pos_weight parameter was utilized to address class imbalance.

```r
# Prepare data for XGBoost
train_matrix <- xgb.DMatrix(data = as.matrix(train[, -ncol(train),
    with = FALSE]), label = train$Class)
test_matrix <- xgb.DMatrix(data = as.matrix(test[, -ncol(test),
    with = FALSE]), label = test$Class)

# Set parameters for XGBoost
params <- list(
  objective = "binary:logistic",
  eval_metric = "logloss",
  max_depth = 6,
  eta = 0.1,
  nthread = numCores,
  scale_pos_weight = sum(train$Class == 0) / sum(train$Class == 1)
)

# Train the XGBoost model with cross-validation
set.seed(123)
model_xgb <- xgb.train(
  params = params,
  data = train_matrix,
  nrounds = 100,
  watchlist = list(train = train_matrix),
  early_stopping_rounds = 10,
  verbose = 0
)
```

3.3 Model Evaluation

The trained model was evaluated on the test set, and the performance was summarized using a confusion matrix. The Precision-Recall curve was also plotted to visualize the model's ability to differentiate between classes.

```r
# Load necessary libraries
library(data.table)
library(caret)
library(xgboost)
library(doParallel)
library(ggplot2)
library(PRROC)

# Load the dataset using fread for faster loading
data <- fread("creditcard.csv")

# Data exploration
str(data)           # Check structure of the dataset
summary(data)       # Get basic statistics
table(data$Class)   # Check the distribution of the target variable
    (Class)

# Check for missing values
missing_percentage <- (colSums(is.na(data)) / nrow(data)) * 100
print(missing_percentage)
```

Preparing Data for XGBoost and Training the Model

```r
# Prepare data for XGBoost
train_matrix <- xgb.DMatrix(data = as.matrix(train[, -ncol(train),
    with = FALSE]), label = train$Class)
test_matrix <- xgb.DMatrix(data = as.matrix(test[, -ncol(test),
    with = FALSE]), label = test$Class)

# Set parameters for XGBoost
params <- list(
  objective = "binary:logistic",
  eval_metric = "logloss",
  max_depth = 6,
  eta = 0.1,
  nthread = numCores,
  scale_pos_weight = sum(train$Class == 0) / sum(train$Class == 1)
)

# Train the XGBoost model with cross-validation
set.seed(123)
model_xgb <- xgb.train(
  params = params,
  data = train_matrix,
  nrounds = 100,
  watchlist = list(train = train_matrix),
  early_stopping_rounds = 10,
  verbose = 0
)
```

Making Predictions and Evaluating the Model

```
# Make predictions on the test set
predictions_xgb <- predict(model_xgb, test_matrix)
predictions_xgb_class <- ifelse(predictions_xgb > 0.5, 1, 0)

# Evaluate the model performance
confusion_result <- confusionMatrix(factor(predictions_xgb_class),
    factor(test$Class))
print(confusion_result)

# Precision-Recall Curve and AUPRC calculation
pr_curve <- pr.curve(scores.class0 = predictions_xgb[test$Class ==
    1],
                     scores.class1 = predictions_xgb[test$Class ==
                         0],
                     curve = TRUE)

# Plot the Precision-Recall Curve
plot(pr_curve, main="Precision-Recall Curve")

# Display AUPRC value
cat("AUPRC:", pr_curve$auc, "\n")
```

3.4 Feature importance

To gain insights into which features contributed most to the model's predictions, the feature importance was visualized.

3.5 Results

The confusion matrix output indicated the model's accuracy and performance metrics, which are crucial for evaluating the effectiveness of the fraud detection model. The AUC for the Precision-Recall curve was calculated to be 0.8374, suggesting that the model has a good ability to balance precision and recall. While the model maintains a good balance between precision and recall, it is important to evaluate whether this trade-off is appropriate for the goals of your application (for example, fraud detection may prioritize recall over precision).

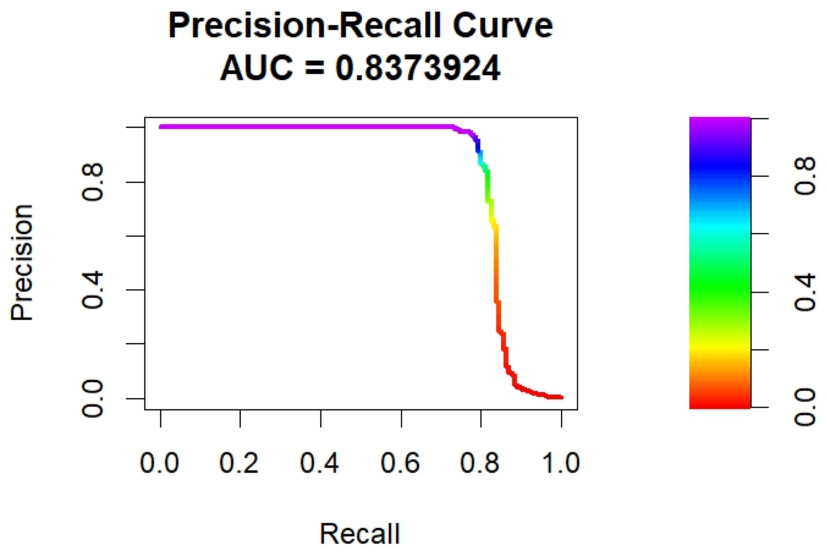

Figure 2: precision-recall curve

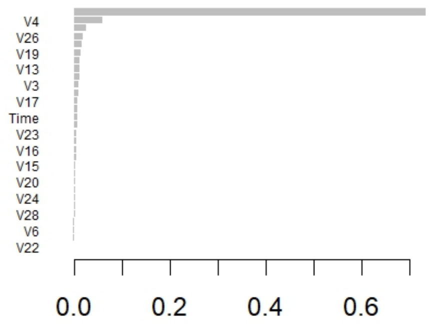

Figure 3: feature importance

4 Unsupervised

4.1 Data Preprocessing

Data preprocessing steps involved:

- **Handling Missing Data:** Any missing data was removed using the function `na.omit`.
- **Removing Constant and Zero-Variance Features:** Features that had constant values or zero variance across all transactions were identified and removed to:
  - Prevent computational inefficiency.
  - Improve the PCA results.
- **Feature Scaling:** Since PCA is sensitive to the scale of the input data, all features were standardized (centered and scaled) to have:
  - Zero mean.
  - Unit variance.

Dimensionality Reduction Using PCA PCA was applied to reduce the dimensionality of the dataset while retaining most of the variance in the original features. The following steps were performed:

- **Fit PCA:** The scaled data was fed into PCA, and the principal components were extracted.
- **Explained Variance Analysis:** The explained variance for each principal component was calculated to assess how much information is captured by each component.
- **Cumulative Explained Variance:** This analysis helped determine the number of principal components required to retain a high proportion of the total variance.

4.2 Evaluation of PCA Results

- **Variance Explained by Each Component:** The explained variance of each principal component was examined to understand how much information (variance) is retained by each.
- **Cumulative Variance:** The cumulative variance was analyzed to decide the optimal number of components for dimensionality reduction, aiming to retain approximately 95% of the original variance.
- **PCA Visualization:** The first two principal components were used to visualize the dataset, focusing on whether fraudulent and non-fraudulent transactions could be separated in the reduced space.

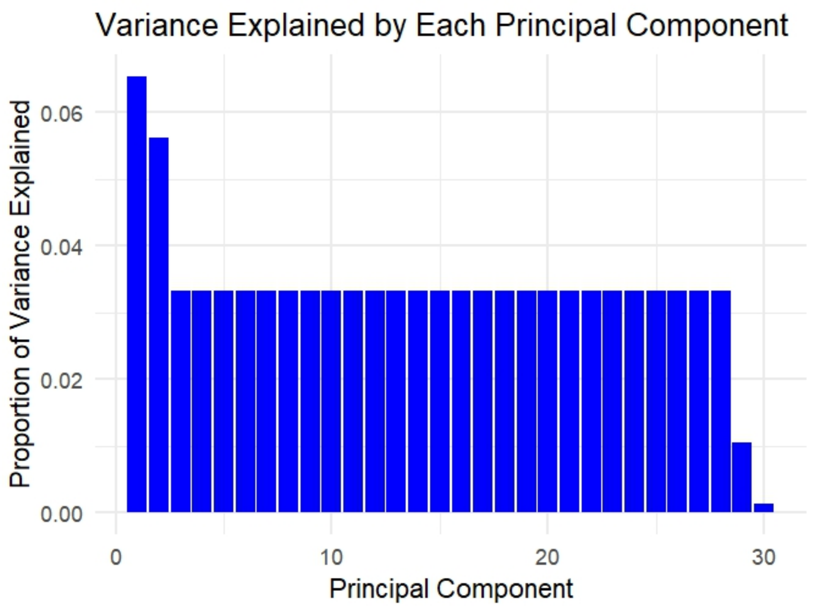

Figure 4: variance explained by each principal component

4.3 Results

- **Explained Variance:**
    - Principal Component 1 (PC1) explained only 6.5% of the total variance in the dataset.
    - Principal Component 2 (PC2) contributed an additional 5.6% of the variance.
    - Beyond the first two components, most components explained an even smaller portion of the variance (approximately 3.33%).

  The cumulative variance showed that:

    - The first 22 principal components were required to retain around 78.8% of the total variance.
    - Achieving 95% of the cumulative variance required the inclusion of nearly all 30 principal components, indicating that dimensionality could not be significantly reduced without substantial information loss.

- **PCA Visualization:** Despite the dimensionality reduction:
    - No clear separation was observed between the fraudulent and non-fraudulent transactions in the first two principal components; both classes were highly overlapping in the PCA plot.

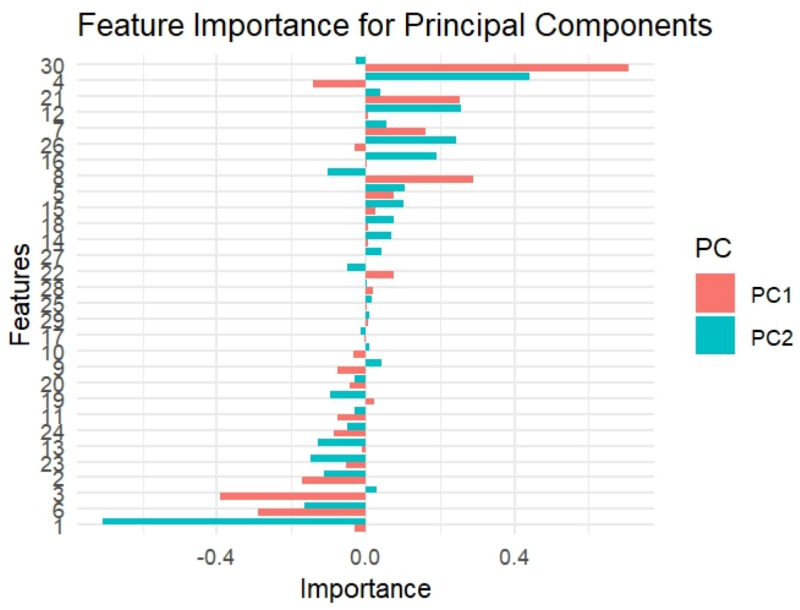

Figure 5: feature importance for principal components

  - The PCA-transformed dataset still failed to distinguish between the two classes, highlighting the limitation of PCA as a tool for fraud detection when applied in isolation to this problem.

- **Class Imbalance and Model Performance:** The confusion matrix after performing supervised learning on the PCA-transformed data showed that:

  - Sensitivity for non-fraudulent transactions (Class 0) was 100%, meaning that all non-fraudulent transactions were correctly identified.
  - Specificity for fraudulent transactions (Class 1) was 0%, indicating that no fraudulent transactions were correctly identified, even after PCA transformation.
  - Balanced accuracy was 50%, confirming that the model was not better than random guessing in detecting fraud.

4.4 Discussion

- **Limitations of PCA for Fraud Detection:** The results demonstrate the limitations of PCA for detecting fraudulent transactions in highly imbalanced datasets. Specifically:

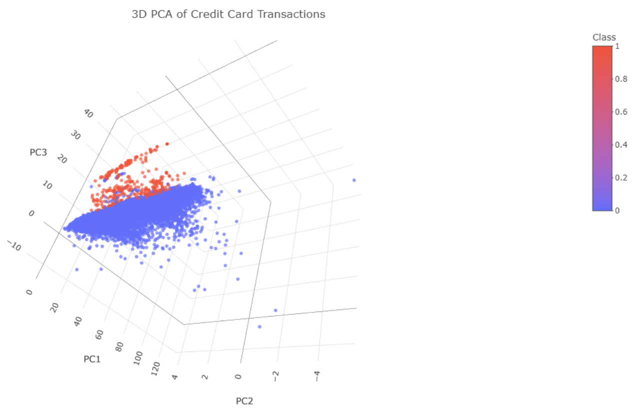

Figure 6: PCA in 3d

- **Variance Distribution:** PCA was unable to compress the data effectively into a small number of components while retaining the variance needed for accurate fraud detection.
- **No Class Separation:** PCA failed to differentiate between fraudulent and non-fraudulent transactions when visualized, likely due to the overwhelming class imbalance and the nature of the dataset, where class differences may not be linearly separable.

PCA is a powerful tool for dimensionality reduction and data visualization, but for this problem, it was not sufficient to capture the nuanced patterns necessary for fraud detection.

5 Conclusion

This analysis highlights the challenges of using PCA for fraud detection in imbalanced datasets. Although PCA is useful for reducing dimensionality and understanding the data, it is not sufficient on its own for detecting fraud. Future efforts should focus on combining PCA with more sophisticated techniques that can handle the class imbalance problem and improve the identification of fraudulent transactions.

This report examined both supervised and unsupervised learning methods for credit card fraud detection, highlighting the strengths and limitations of each approach.

The XGBoost model used in supervised learning was effective in eliminating the significant class imbalance in the dataset, achieving high performance with an AUPRC of 0.8374. By adjusting the parameters to eliminate the imbalance, the model demonstrated a good balance between precision and recall, making it a suitable choice for fraud detection. Feature importance analysis further provided information about the key variables driving the model's predictions, contributing to better interpretability of the fraud detection process.

In contrast, using Principal Component Analysis (PCA) as an unsupervised learning method for dimensionality reduction revealed several limitations. Although PCA was able to reduce the dimensionality of the data, it was unable to separate fraudulent transactions from non-fraudulent ones in the reduced space. The first two principal components capture only a small fraction of the total variance, and cumulative variance analysis showed that almost all components are necessary to preserve meaningful information. As a result, PCA does not effectively distinguish between the two classes, highlighting its limited usefulness for fraud detection in highly imbalanced datasets.

To summarize, supervised learning models, especially XGBoost, are well suited for fraud detection in imbalanced datasets, while PCA and similar unsupervised methods are insufficient on their own. Future work should focus on combining these methods into a hybrid approach that leverages the strengths of both. Additionally, incorporating advanced anomaly detection techniques and improved methods for handling class imbalance, such as cost-aware learning or oversampling, can further improve the effectiveness of fraud detection systems.