Defying Defaults

Analyzing Key Trends to Safeguard Your Business

Xe Yuin Chong

Introduction

Unlocking Insights for a Secure Financial Future

In a world where financial stability is paramount, the ability to mitigate payment defaults can make or break a business. Today, we embark on a journey to uncover the insights that will empower us to secure our financial future.

As we navigate through data-rich landscapes, we'll decode the patterns and trends hidden within our payment transactions. These insights hold the keys to strengthening our business and fostering growth.

Datasets

Payments Dataset: Contains essential information about transaction payments, including unique identifiers (TransactionID, ContractID, ClientID), transaction dates, repayment amounts, and payment codes. Each row represents a transaction payment associated with a contract.

Clients Dataset: Provides valuable insights into our clients. It includes client identifiers (ClientID), their entity types, and the year their businesses were established. This dataset allows us to contextualize payment behaviors based on client characteristics.

Objective

Our primary objective is to gain a deep understanding of payment defaults and identify key trends within our data. Payment defaults are not merely financial setbacks; they are critical cost factors that impact our business's overall health and growth potential.

By harnessing the power of data analysis, we aim to:

- Uncover patterns and correlations that shed light on why defaults occur.
- Identify potential default-prone clients and transaction trends.
- Develop actionable recommendations to mitigate future payment defaults.

Data Preprocessing

Data Cleaning

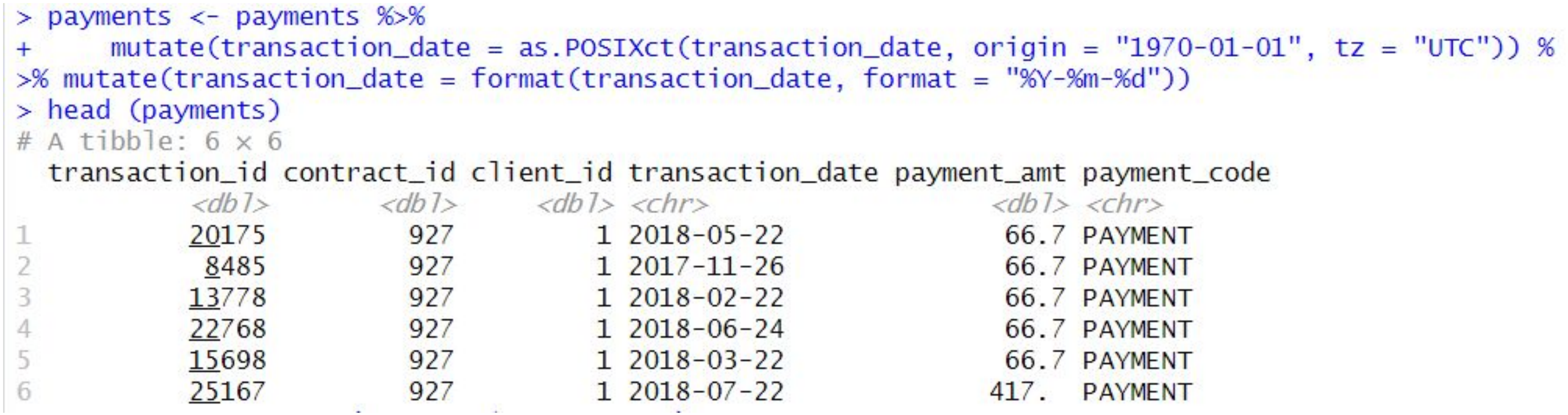
1. **Handling Missing Data:** A thorough check for missing data was conducted and no missing values were identified in the dataset.

2. **Correcting Typos and Inconsistencies:** The data was carefully reviewed for typographical errors and inconsistencies but no typos or inconsistencies were found in the dataset.

3. **Managing Duplicate Values:** Duplicate values were examined to ensure data integrity but no duplicate values were verified within the dataset.

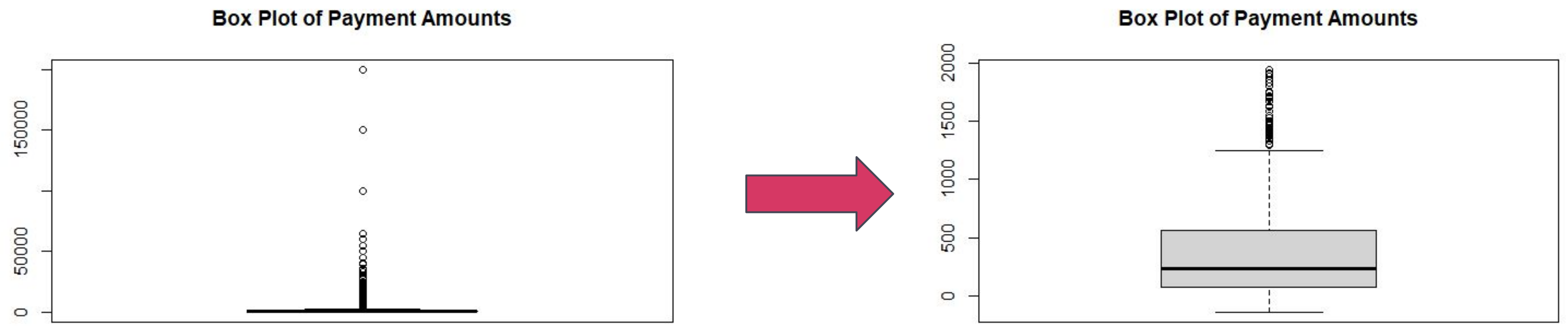
Data Type Conversion

To facilitate analysis, the 'transaction_date' column in the 'Payments' dataset is converted from EPOCH time to a readable date format using the 'as.POSIXct' function in R.



```
> payments <- payments %>%
+     mutate(transaction_date = as.POSIXct(transaction_date, origin = "1970-01-01", tz = "UTC")) %
>% mutate(transaction_date = format(transaction_date, format = "%Y-%m-%d"))
> head (payments)
# A tibble: 6 × 6
  transaction_id contract_id client_id transaction_date payment_amt payment_code
           <dbl>       <dbl>     <dbl> <chr>                  <dbl> <chr>
1          20175         927         1 2018-05-22              66.7 PAYMENT
2           8485         927         1 2017-11-26              66.7 PAYMENT
3          13778         927         1 2018-02-22              66.7 PAYMENT
4          22768         927         1 2018-06-24              66.7 PAYMENT
5          15698         927         1 2018-03-22              66.7 PAYMENT
6          25167         927         1 2018-07-22             417.  PAYMENT
```

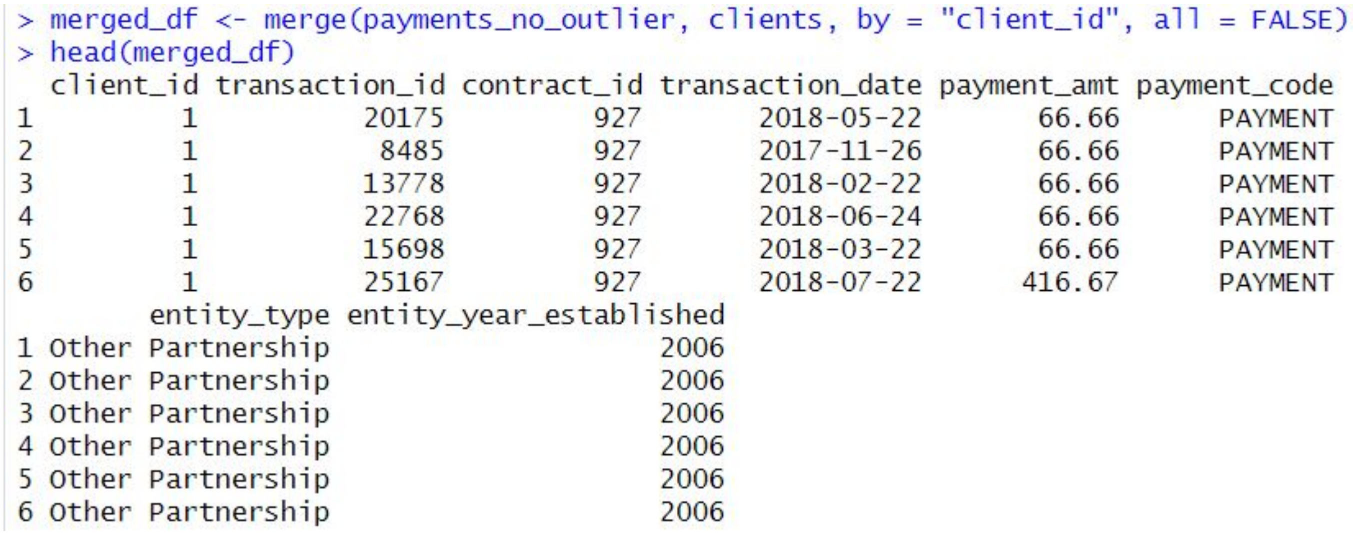
Handling Outliers



Outliers in the 'payment_amt' column is identified and handled to prevent them from skewing our analysis. Inter-Quartile Range (IQR) proximity rule is used to remove outliers.

Data Merging

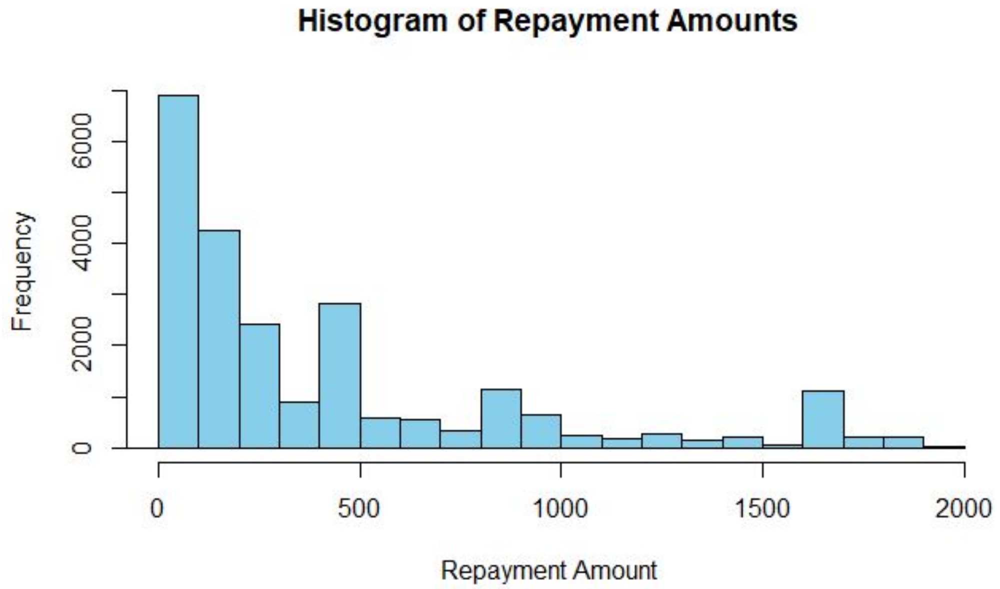
Payments and Clients datasets is merged using the common 'client_id' column to gain a comprehensive view of client information alongside transaction data.



```
> merged_df <- merge(payments_no_outlier, clients, by = "client_id", all = FALSE)
> head(merged_df)
  client_id transaction_id contract_id transaction_date payment_amt payment_code
1         1          20175         927       2018-05-22       66.66      PAYMENT
2         1           8485         927       2017-11-26       66.66      PAYMENT
3         1          13778         927       2018-02-22       66.66      PAYMENT
4         1          22768         927       2018-06-24       66.66      PAYMENT
5         1          15698         927       2018-03-22       66.66      PAYMENT
6         1          25167         927       2018-07-22      416.67      PAYMENT
        entity_type entity_year_established
1 Other Partnership                    2006
2 Other Partnership                    2006
3 Other Partnership                    2006
4 Other Partnership                    2006
5 Other Partnership                    2006
6 Other Partnership                    2006
```

Exploratory Data Analysis (EDA)

Distribution of Payment Amounts



According to the Histogram shown on the left, we can see that the distribution is right skewed.

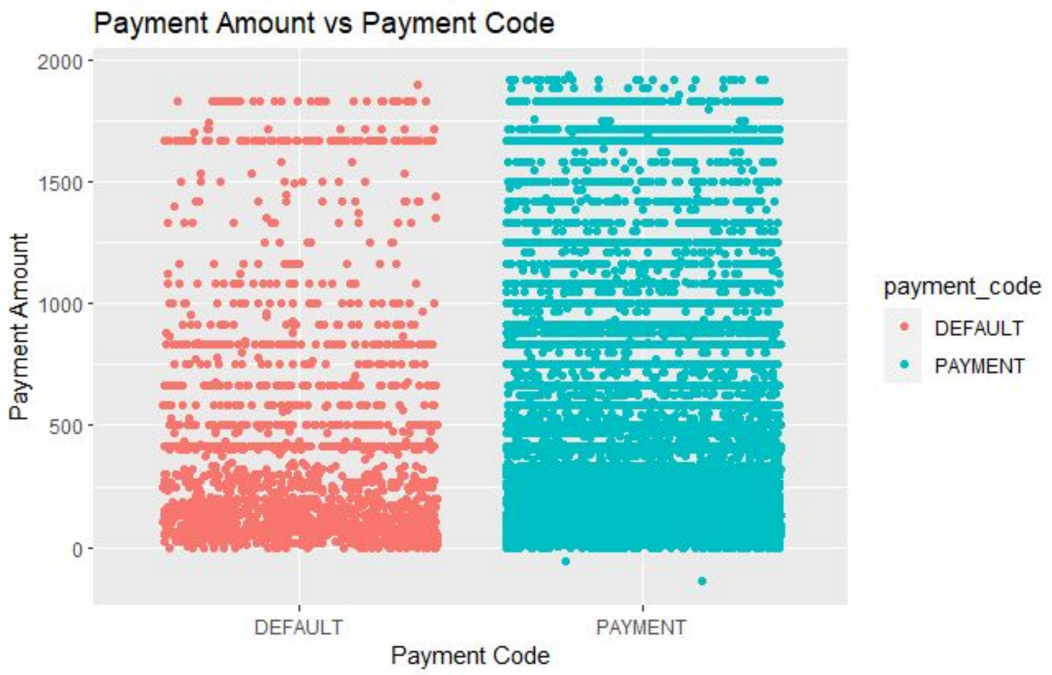
This suggests that the majority of clients make relatively modest payments.

Summary Statistics

After handling outliers, we can see the statistics shown below:

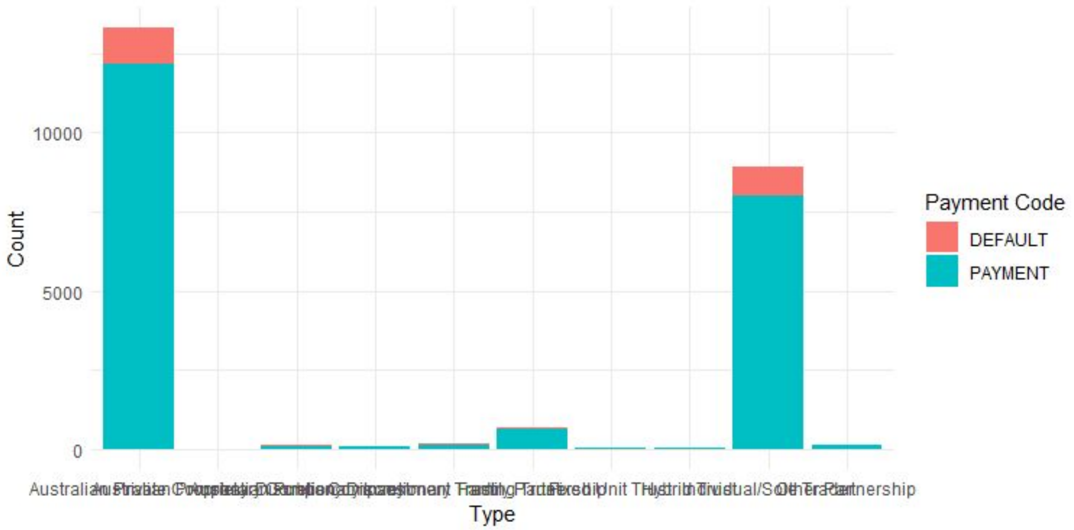
- **Mean = $423.66** - On average, clients pay around this value.
- **Median = $249.99** - Half of the payments fall below this amount, indicating a significant portion of smaller payments.
- **Standard Deviation = $472.84** - Suggests a wide range of payment amounts, with some clients making significantly larger payments than others.

Default vs. Non-Default Payments



From the scatterplot on the left, both DEFAULT and non-DEFAULT payments have dense clusters in lower payment amounts. It suggest that low payments are common among all clients. However, it's essential to pay close attention to the presence of DEFAULT payments within this range, indicating a higher risk of defaults.

Entity Type Distribution

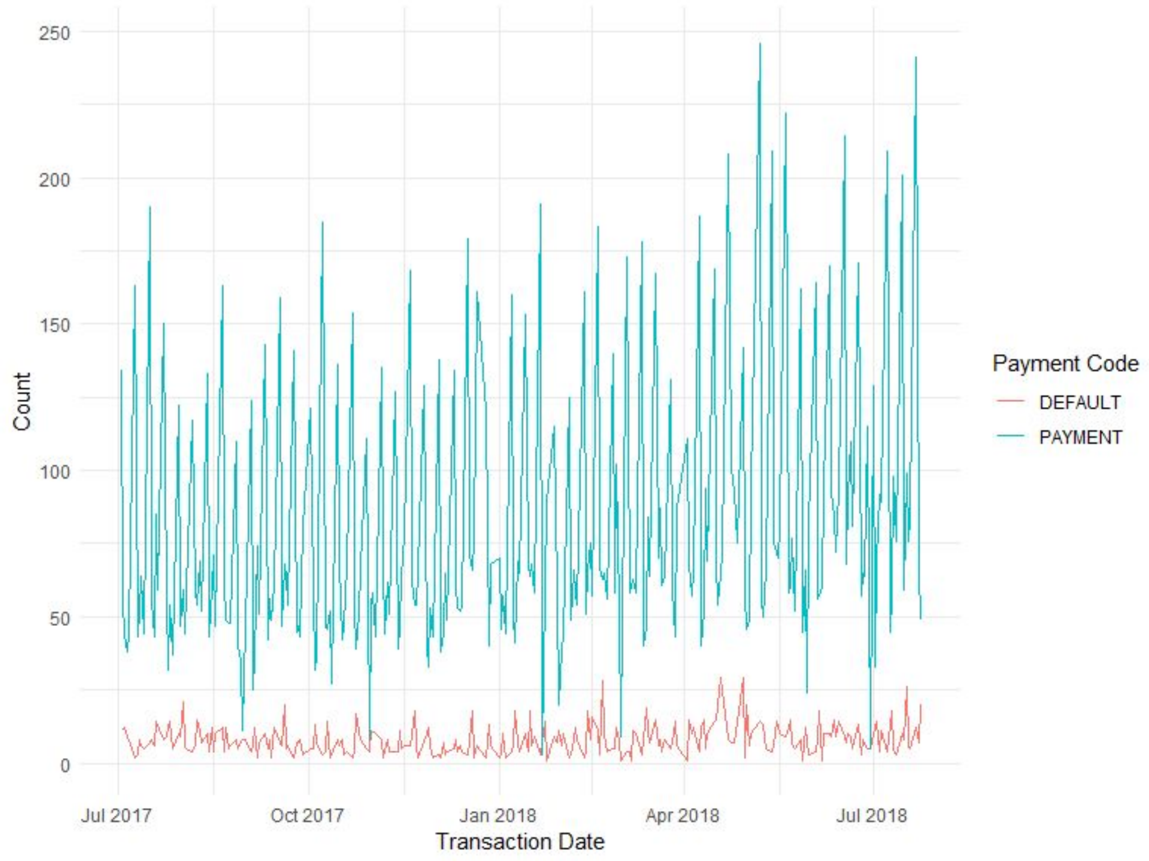


We find a diverse mix of entity types among our clients, with 'Australian Private Company' and 'Individual/Sole Trader' being the most common.

Time series plot

———

Payments appear to exhibit a seasonal pattern, with a noticeable increase in defaults around March 2016 and May 2016.



Feature Engineering

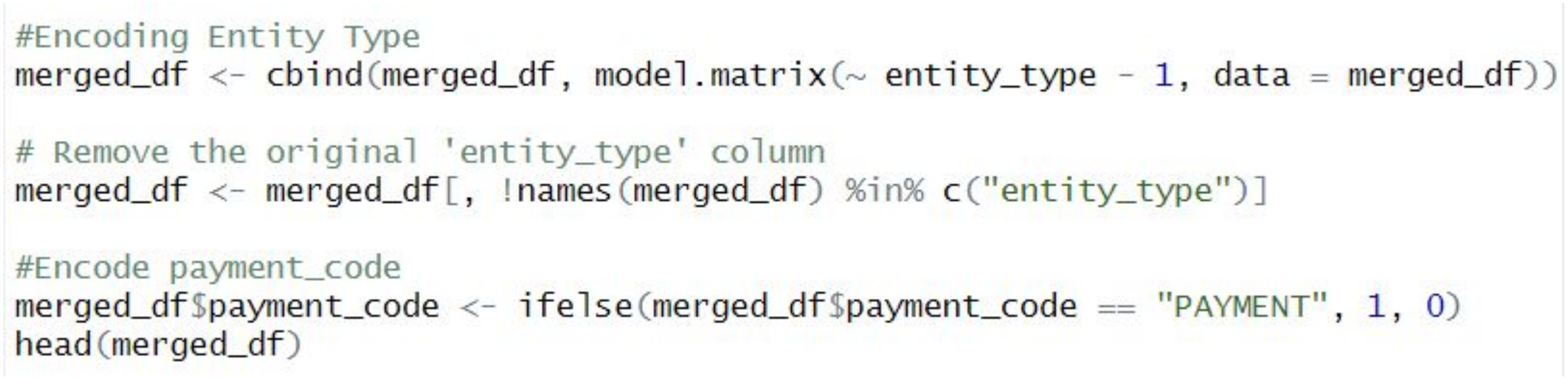
Enhancing Predictive Insights

A crucial step in data analysis that involves creating new variables (features) or transforming existing ones to improve the performance of machine learning models or gain deeper insights from the data

Encoding Categorical Variables

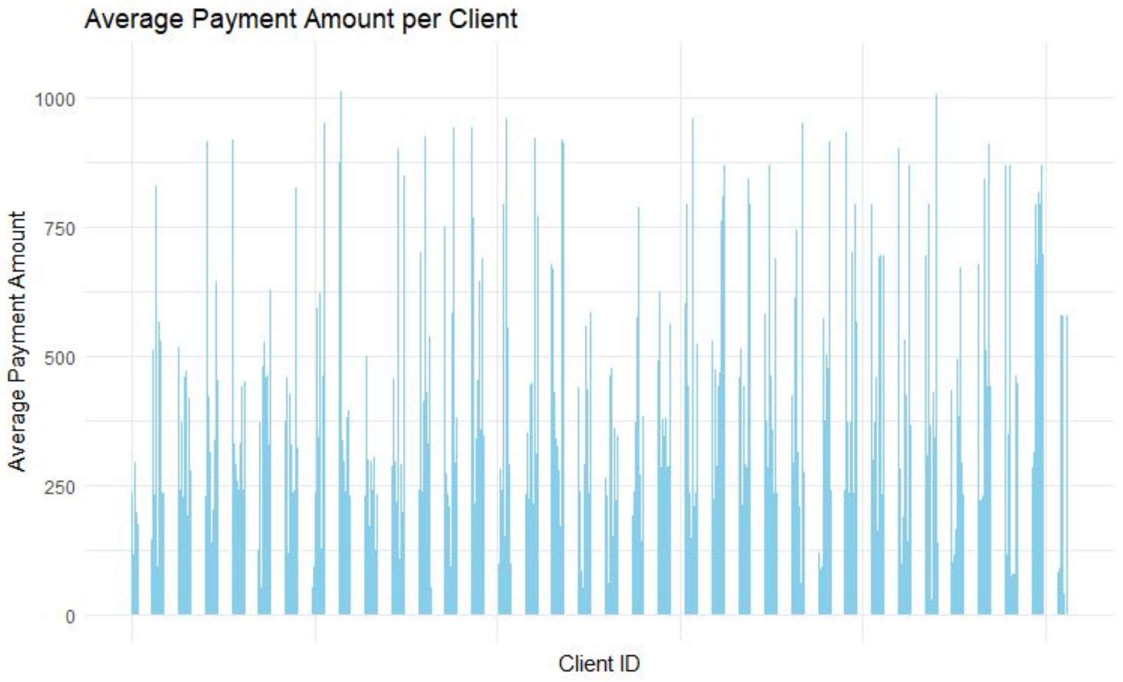
For categorical variables like 'entity_type', one-hot encoding is used to convert them into numerical format for analysis.

As for 'payment_code', binary encoding is implemented by using 0 and 1 to represent ‘Default’ and ‘Payment’.



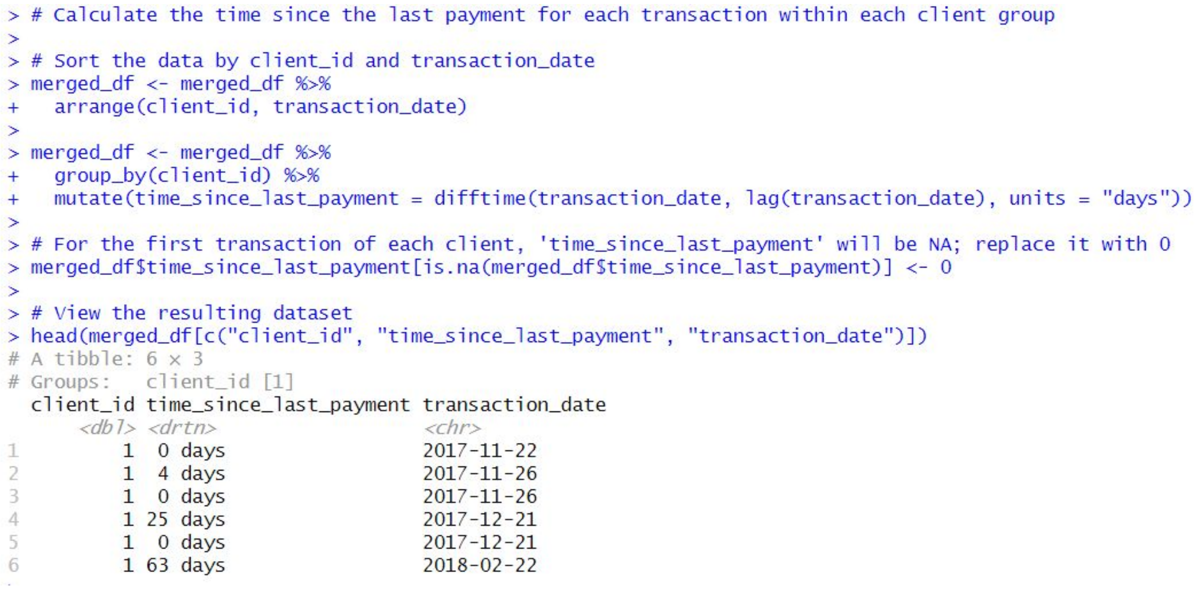
```
#Encoding Entity Type
merged_df <- cbind(merged_df, model.matrix(~ entity_type - 1, data = merged_df))

# Remove the original 'entity_type' column
merged_df <- merged_df[, !names(merged_df) %in% c("entity_type")]

#Encode payment_code
merged_df$payment_code <- ifelse(merged_df$payment_code == "PAYMENT", 1, 0)
head(merged_df)
```

Average Payment Amount per Client



The average payment amount of each client is calculated and plotted with a graph shown on the left.

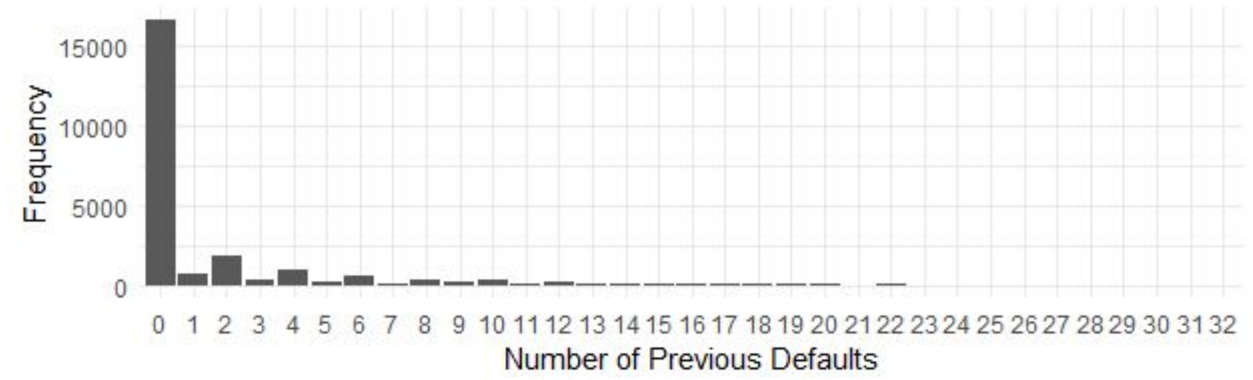
Time since last payment



```
> # Calculate the time since the last payment for each transaction within each client group
>
> # Sort the data by client_id and transaction_date
> merged_df <- merged_df %>%
+   arrange(client_id, transaction_date)
>
> merged_df <- merged_df %>%
+   group_by(client_id) %>%
+   mutate(time_since_last_payment = difftime(transaction_date, lag(transaction_date), units = "days"))
>
> # For the first transaction of each client, 'time_since_last_payment' will be NA; replace it with 0
> merged_df$time_since_last_payment[is.na(merged_df$time_since_last_payment)] <- 0
>
> # View the resulting dataset
> head(merged_df[c("client_id", "time_since_last_payment", "transaction_date")])
# A tibble: 6 × 3
# Groups:   client_id [1]
  client_id time_since_last_payment transaction_date
      <dbl> <drtn>                  <chr>
1         1  0 days                 2017-11-22
2         1  4 days                 2017-11-26
3         1  0 days                 2017-11-26
4         1 25 days                 2017-12-21
5         1  0 days                 2017-12-21
6         1 63 days                 2018-02-22
```

A 'time_since_last_payment' feature was engineered to capture how frequently clients make payments.

Number of previous default



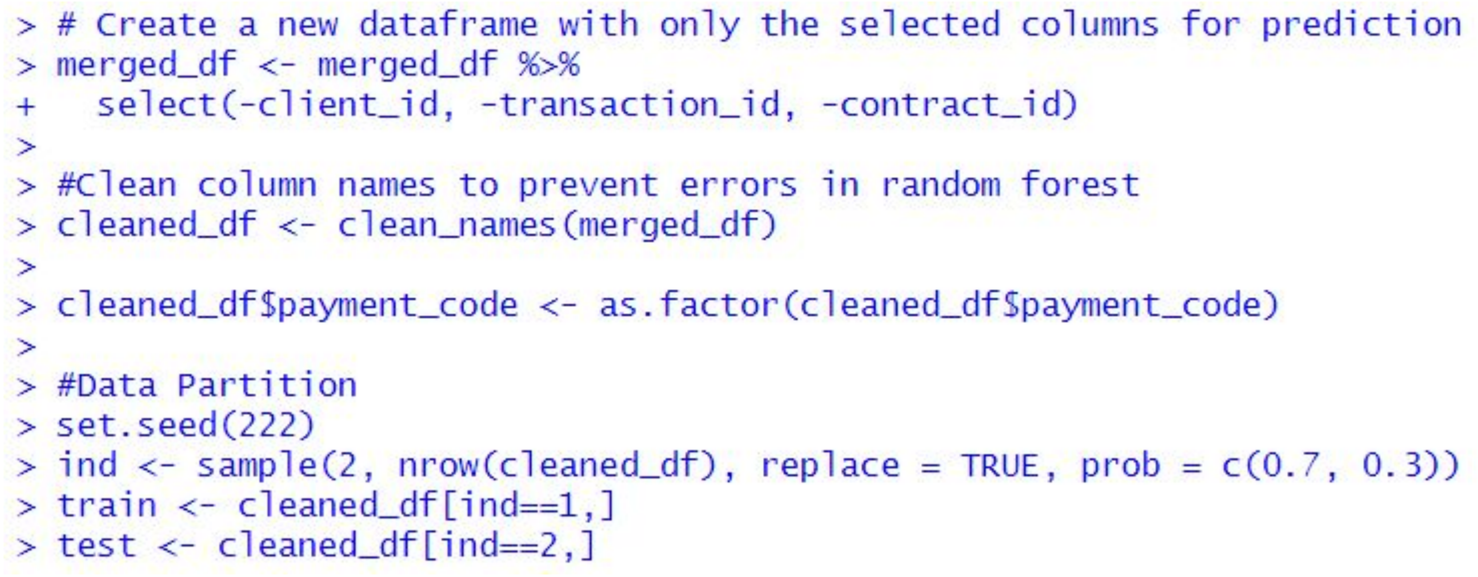
A 'previous_default' feature was engineered to quantify previous defaults for each client. We can see that a number of clients have an alarming number of previous defaults.

Model Building and Prediction

Constructing the Random Forest Model

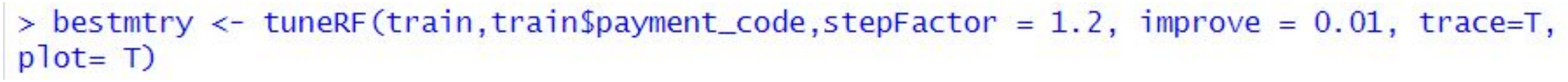
Among all the available classification methods, random forests provide the highest accuracy. The random forest technique can also handle big data with numerous variables running into thousands. It can automatically balance data sets when a class is more infrequent than other classes in the data. It is also less prone to overfitting.

Data Prep and Partition

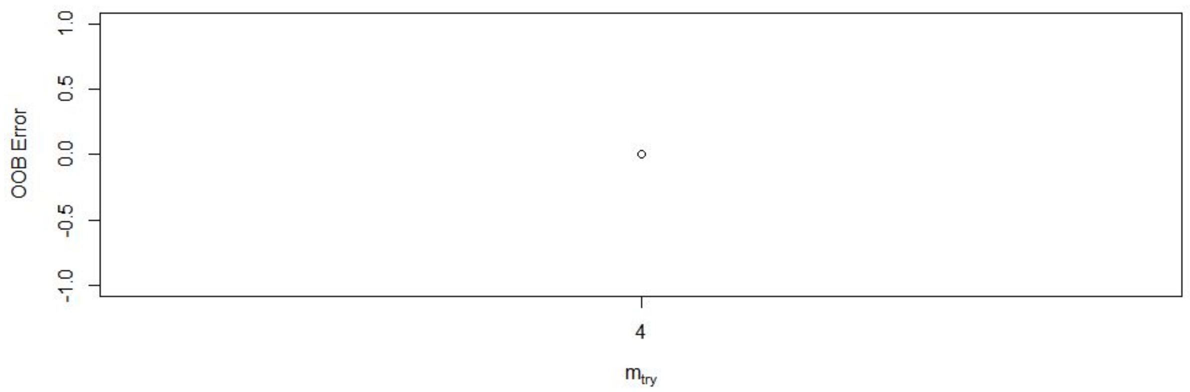


```
> # Create a new dataframe with only the selected columns for prediction
> merged_df <- merged_df %>%
+   select(-client_id, -transaction_id, -contract_id)
>
> #Clean column names to prevent errors in random forest
> cleaned_df <- clean_names(merged_df)
>
> cleaned_df$payment_code <- as.factor(cleaned_df$payment_code)
>
> #Data Partition
> set.seed(222)
> ind <- sample(2, nrow(cleaned_df), replace = TRUE, prob = c(0.7, 0.3))
> train <- cleaned_df[ind==1,]
> test <- cleaned_df[ind==2,]
```

Finding optimized value of 'm'(random variables)

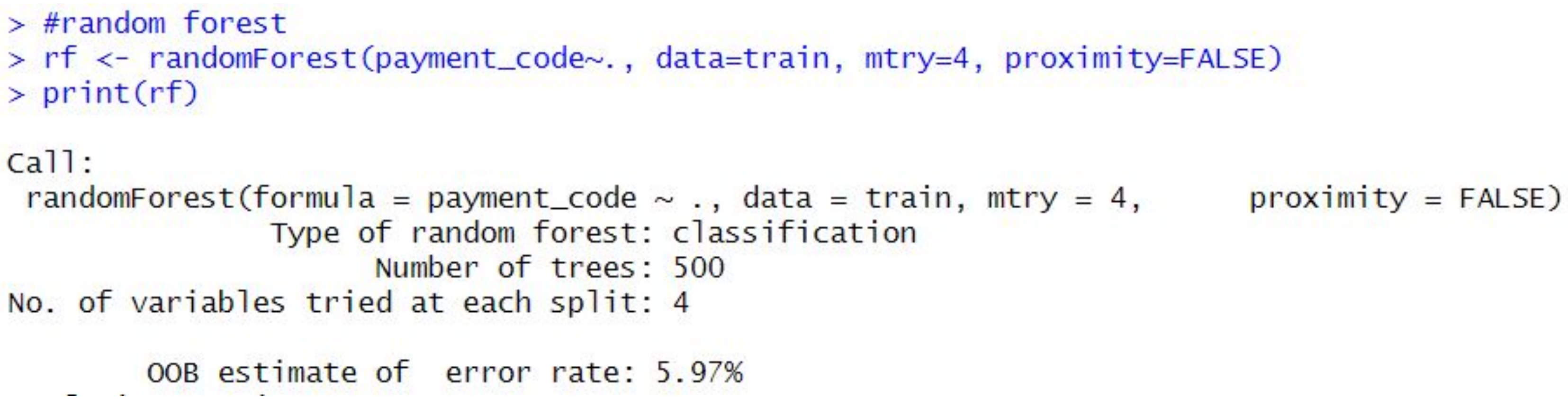


```
> bestmtry <- tuneRF(train,train$payment_code,stepFactor = 1.2, improve = 0.01, trace=T,
plot= T)
```



```
tuneRF returns the best optimized value of random variable
is 4 corresponding to a OOB of 0%
```

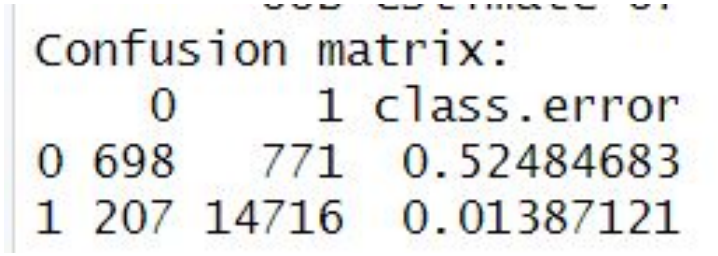
Build model



```
> #random forest
> rf <- randomForest(payment_code~., data=train, mtry=4, proximity=FALSE)
> print(rf)

Call:
 randomForest(formula = payment_code ~ ., data = train, mtry = 4,      proximity = FALSE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 5.97%
```

The model summary suggests that, the type of random forest is classification. 500 random forest trees were created. At every node, the node splits into 4 child nodes .

Confusion matrix

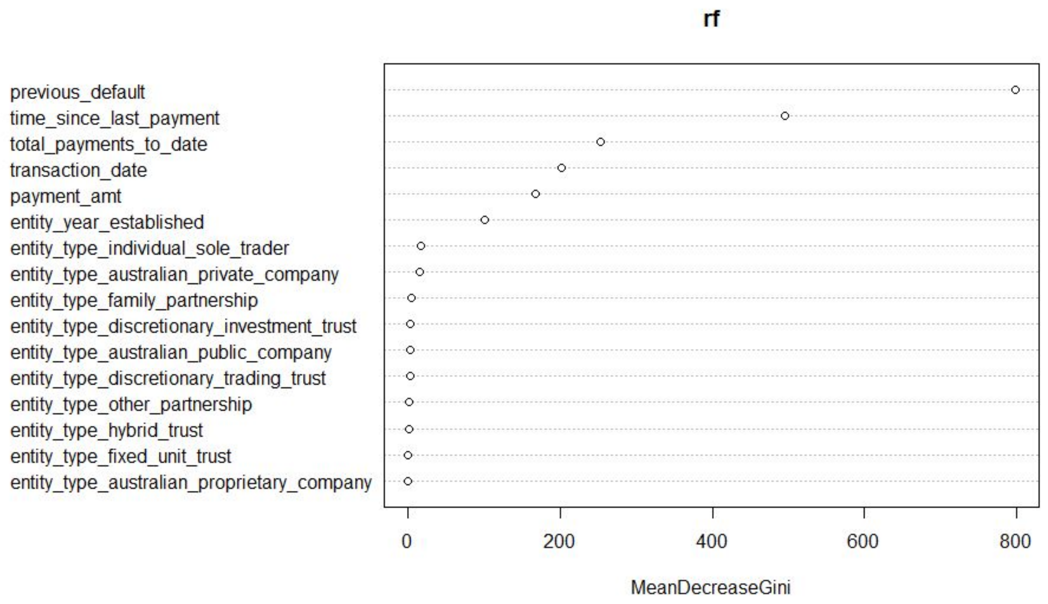


```
Confusion matrix:
    0     1 class.error
0 698   771  0.52484683
1 207 14716  0.01387121
```

The confusion matrix suggests that , 698 clients were correctly identified as DEFAULT, 14716 clients were correctly identified as non-DEFAULT.

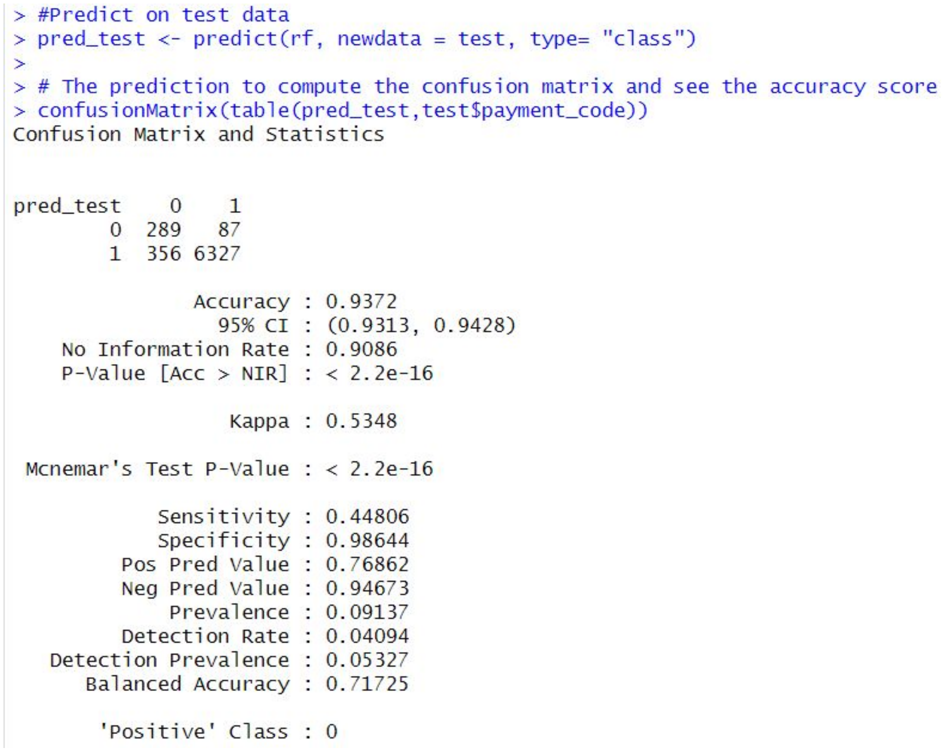
207 clients were incorrectly identified as DEFAULT, and 771 clients were incorrectly identified as non-DEFAULT.

Identify importance of variables



The graph on the left shows the importance of each variables with 'previous_default' as most significant followed by 'time_since_last_payment'

Predicting Test Sets



```
> #Predict on test data
> pred_test <- predict(rf, newdata = test, type= "class")
>
> # The prediction to compute the confusion matrix and see the accuracy score
> confusionMatrix(table(pred_test,test$payment_code))
Confusion Matrix and Statistics


pred_test    0    1
        0  289   87
        1  356 6327

               Accuracy : 0.9372
                 95% CI : (0.9313, 0.9428)
    No Information Rate : 0.9086
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5348

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.44806
            Specificity : 0.98644
         Pos Pred Value : 0.76862
         Neg Pred Value : 0.94673
             Prevalence : 0.09137
         Detection Rate : 0.04094
   Detection Prevalence : 0.05327
      Balanced Accuracy : 0.71725

       'Positive' Class : 0
```

Using the model created before to predict values in the test set, we can see that the confusion matrix shows an accuracy of 93.72% which is considered good.

Recommendations

Strategies to Reduce Payment Defaults

1. **Early Warning System:**

Implement an early warning system that identifies high-risk clients showing signs of potential defaults. Use the predictive model to regularly assess client risk.

2. **Customer Segmentation:**

Segment clients based on their payment behavior and risk profiles. Tailor communication and engagement strategies for each segment.

3. **Proactive Communication:**

Establish proactive communication channels with clients who exhibit payment irregularities. Send reminders, offer assistance, and address their concerns promptly.

Strategies to Reduce Payment Defaults

4. **Risk-Based Pricing:**

Implement risk-based pricing strategies, offering lower interest rates or fees to low-risk clients, while adjusting rates for higher-risk clients.

5. **Regular Model Updates:**

Continuously monitor and update the predictive model to adapt to changing client behavior and market conditions.

6. **Data Quality Improvement:**

Invest in data quality initiatives to ensure accurate and up-to-date client information for better risk assessment.

Thank You