DeFi Survival Data Pipeline Example

Hanzhen Qin(qinh2)

20 February 2025

Survival Data Pipeline

```r
source("~/KDD_DeFi_Survival_Dataset_And_Benchmark/Classification/data_preprocessing.R")
source("~/KDD_DeFi_Survival_Dataset_And_Benchmark/Classification/model_evaluation_visual.R")
source("~/KDD_DeFi_Survival_Dataset_And_Benchmark/Classification/classification_models.R")
source("~/KDD_DeFi_Survival_Dataset_And_Benchmark/Classification/get_classification_cutoff.R")
```

```r
# set the indexEvent and outcomeEvent
indexEvent = "borrow"
outcomeEvent = "deposit"

# load the corresponding train and test data
get_train_test_data(indexEvent, outcomeEvent)
```

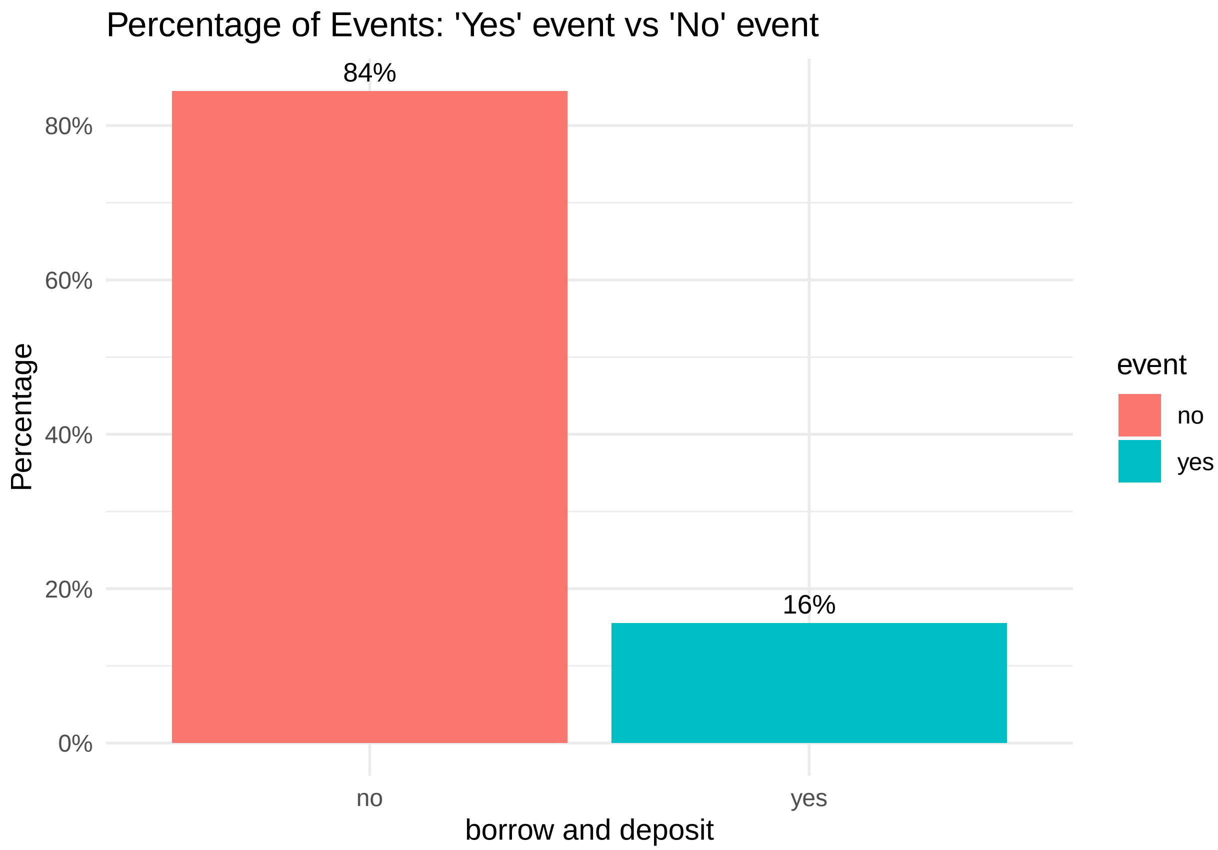
```
## Warning in inner_join(y, X, by = "id"): Detected an unexpected many-to-many relationship between `x`
## i Row 405 of `x` matches multiple rows in `y`.
## i Row 639 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.

## Warning in inner_join(y, X, by = "id"): Detected an unexpected many-to-many relationship between `x`
## i Row 275 of `x` matches multiple rows in `y`.
## i Row 338 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```r
# If you want to check the train and test data, you can run the following codes.
# cat("Train data:\n")
# summary(train)
# cat("Test data:\n")
# summary(test)
```

Using the `get_classification_cutoff` funtion to get the optimal timeDiff, then we will call the `data_processing` function above to get all the training data and test data.

```r
classification_cutoff = get_classification_cutoff(indexEvent, outcomeEvent)
train_data = data_processing(train, classification_cutoff)
test_data = data_processing(test, classification_cutoff)
```

```r
# If you want to watch the percentages between "Yes" and "No" label, run this code.
get_percentage(train_data, indexEvent, outcomeEvent)
```

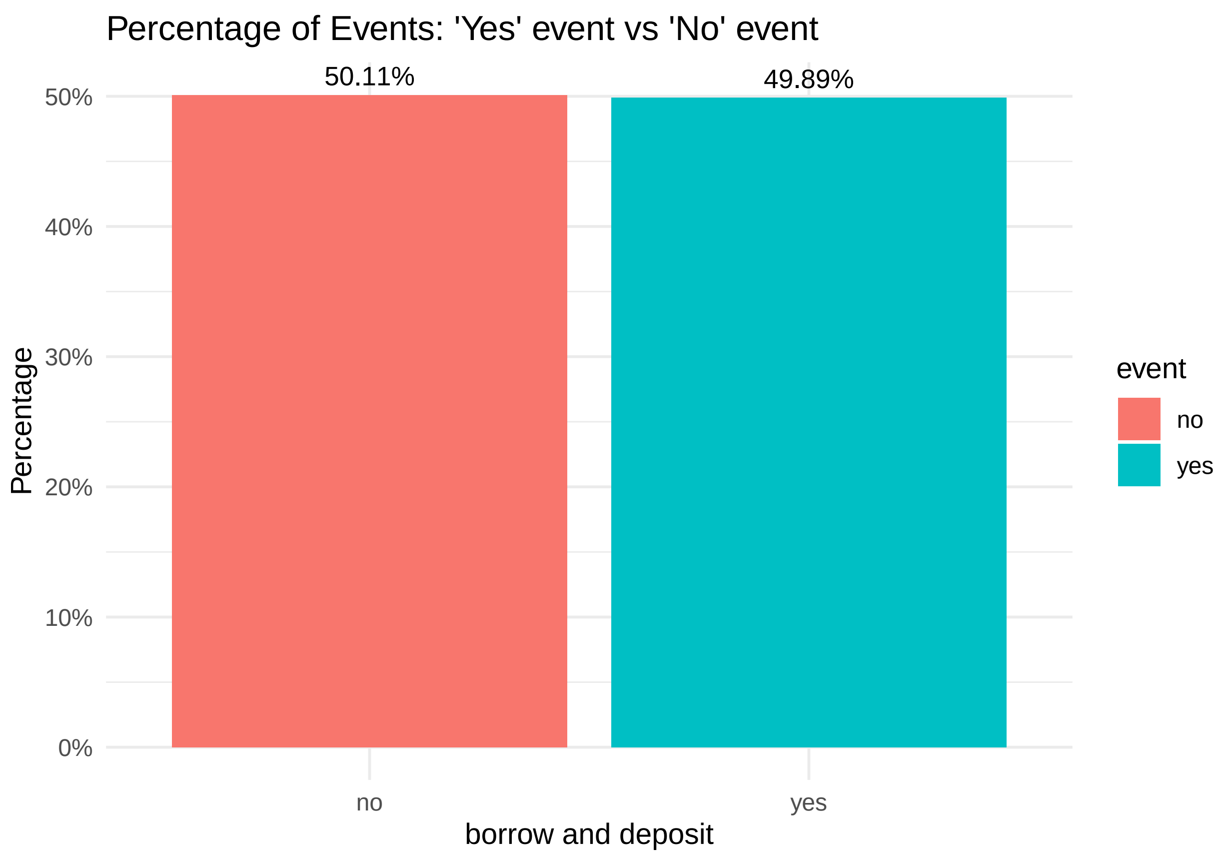


If the ratio of “No” labels to “Yes” labels in the dataset is significantly imbalanced, we can utilize the `smote_data` function to generate a new, more balanced dataset. This balanced dataset ensures that both classes are better represented, helping to mitigate the bias introduced by class imbalance and ultimately improving the accuracy and reliability of our classification model.

```r
train_data <- smote_data(train_data)
```

Then you can check the updated balanced version of train data.

```r
# If you want to watch the percentages between "Yes" and "No" label, run this code.
get_percentage(train_data, indexEvent, outcomeEvent)
```



After obtaining the train and test data, we will apply all the classification models to evaluate the relationship between these events.

```
lr_return = logistic_regression(train_data, test_data)
```

```
## [1] "Logistic Regression (Validation) model prediction accuracy:"
## Balanced accuracy: 67.13%
## F1 score: 66.79%
## [1] "Logistic Regression model prediction accuracy:"
## Balanced accuracy: 62.93%
## F1 score: 46.12%
```

```
accuracy_lr_dataframe = lr_return$metrics_lr_dataframe
accuracy_lr = lr_return$metrics_lr
```

```
dt_return = decision_tree(train_data, test_data)
```

```
## [1] "Decision Tree (Validation) model prediction accuracy:"
## Balanced accuracy: 89.57%
## F1 score: 89.44%
## [1] "Decision Tree model prediction accuracy:"
## Balanced accuracy: 63.74%
## F1 score: 11.70%
```

```r
accuracy_dt_dataframe = dt_return$metrics_dt_dataframe
accuracy_dt = dt_return$metrics_dt
```

```r
nb_return = naive_bayes(train_data, test_data)
```

```
## [1] "Naive Bayes (Validation) model prediction accuracy:"
## Balanced accuracy: 85.89%
## F1 score: 83.51%
## [1] "Naive Bayes model prediction accuracy:"
## Balanced accuracy: 66.04%
## F1 score: 45.49%
```

```r
accuracy_nb_dataframe = nb_return$metrics_nb_dataframe
accuracy_nb = nb_return$metrics_nb
```

```r
xgb_return = XG_Boost(train_data, test_data)
```

```
## [1] "XGBoost (Validation) model prediction accuracy:"
## Balanced accuracy: 98.10%
## F1 score: 98.11%
## [1] "XGBoost model prediction accuracy:"
## Balanced accuracy: 64.53%
## F1 score: 89.90%
```

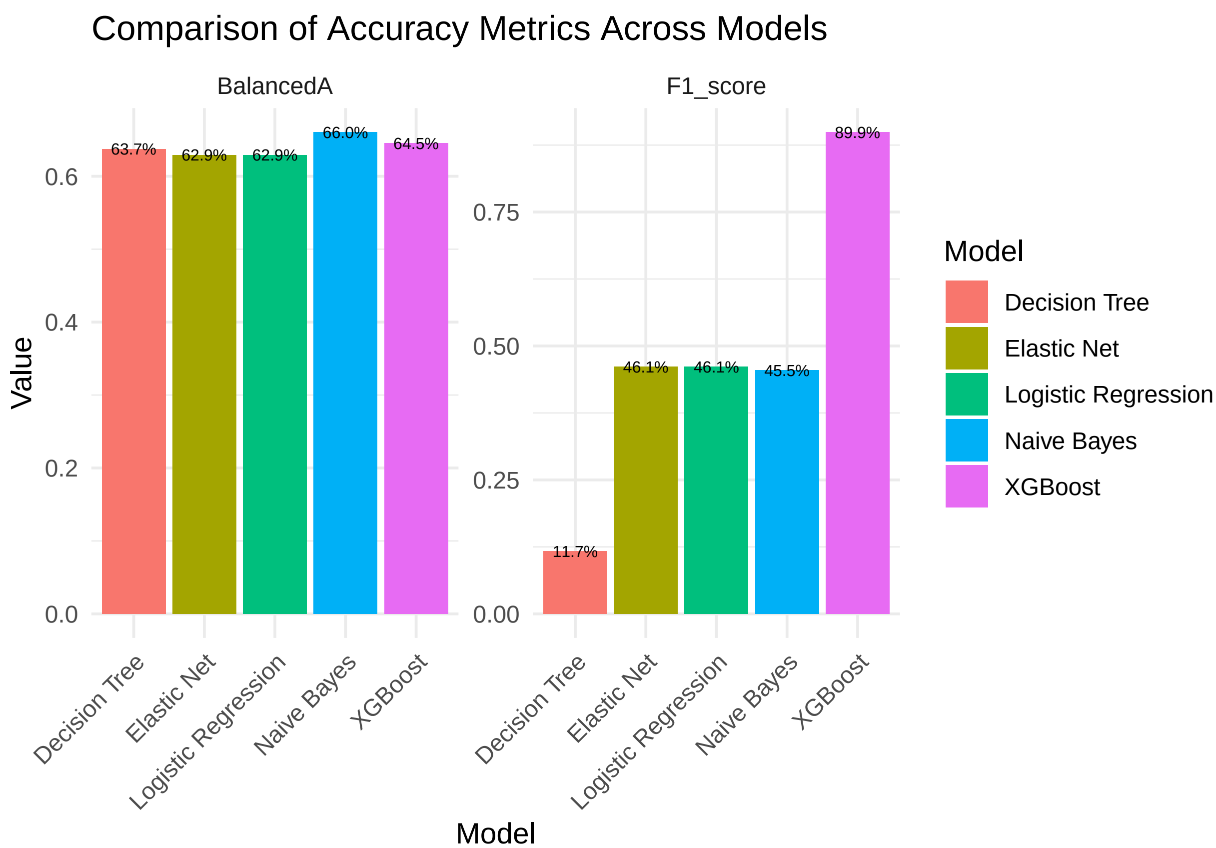
```r
accuracy_xgb_dataframe = xgb_return$metrics_xgb_dataframe
accuracy_xgb = xgb_return$metrics_xgb
```

```r
en_return = elastic_net(train_data, test_data)
```

```
## [1] "Elastic Net (Validation) model prediction accuracy:"
## Balanced accuracy: 67.14%
## F1 score: 66.80%
## [1] "Elastic Net model prediction accuracy:"
## Balanced accuracy: 62.93%
## F1 score: 46.11%
```

```r
accuracy_en_dataframe = en_return$metrics_en_dataframe
accuracy_en = en_return$metrics_en
```

```r
# compare all the classification models
metrics_list_BD <- list(
  list(accuracy_lr, "Logistic Regression"),
  list(accuracy_dt, "Decision Tree"),
  list(accuracy_nb, "Naive Bayes"),
  list(accuracy_xgb, "XGBoost"),
  list(accuracy_en, "Elastic Net")
)
accuracy_comparison_plot(metrics_list_BD)
```




```r
# Show the final dataframe for all classification models,
# including the classification model name, accuracy, data combination name.
data_name_BD <- paste(indexEvent, "+", outcomeEvent)
accuracy_dataframe_list_BD <- list(accuracy_lr_dataframe, accuracy_dt_dataframe,
                                   accuracy_nb_dataframe, accuracy_xgb_dataframe,
                                   accuracy_en_dataframe)
combined_results_BD <- combine_classification_results(accuracy_dataframe_list_BD, data_name_BD)

# display the combined dataframe
pander(combined_results_BD, caption = "Classification Model Performance")
```

Table 1: Classification Model Performance

Model	Balanced_Accuracy	F1_Score	Data_Combination
Logistic Regression	62.93%	46.12%	borrow + deposit
Decision Tree	63.74%	11.70%	borrow + deposit
Naive Bayes	66.04%	45.49%	borrow + deposit
XGBoost	64.53%	89.90%	borrow + deposit
Elastic Net	62.93%	46.11%	borrow + deposit

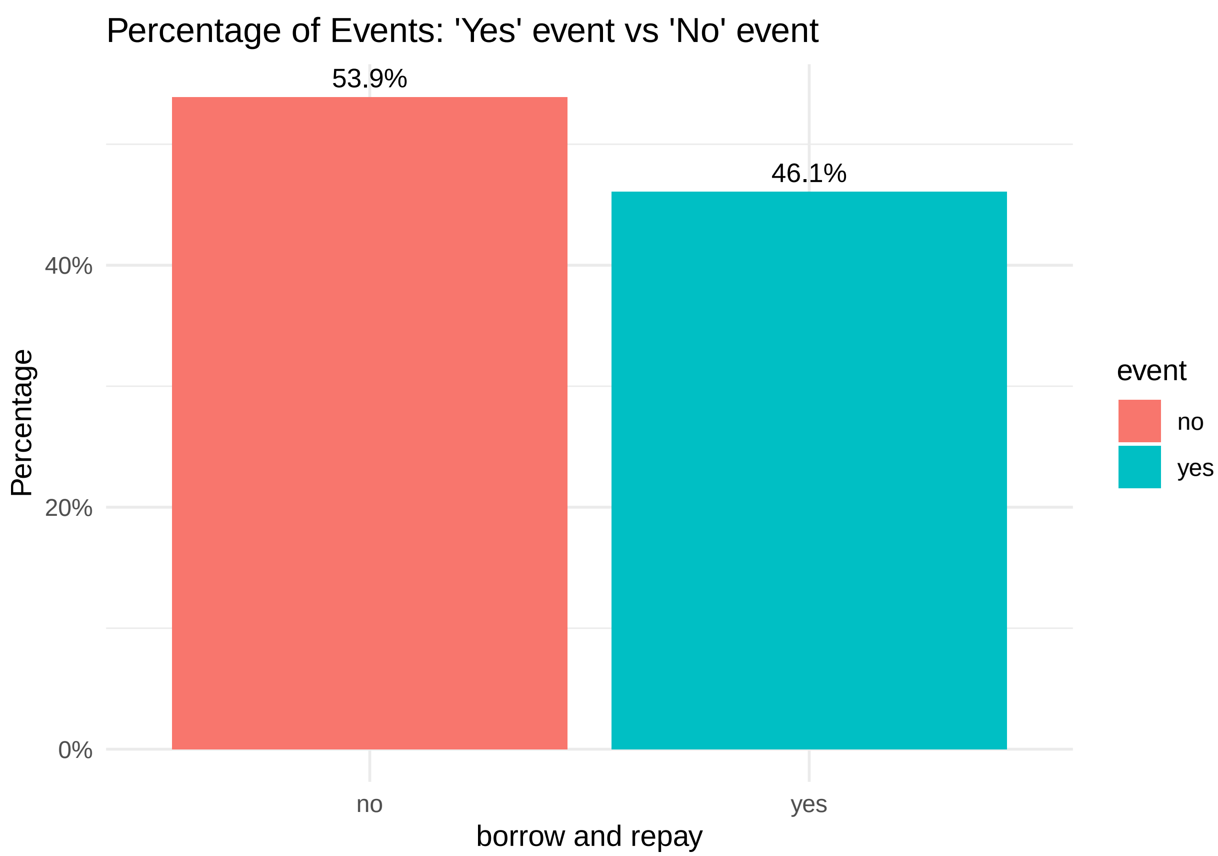
```r
# set the indexEvent and outcomeEvent
indexEvent = "borrow"
outcomeEvent = "repay"
```

```r
# load the corresponding train and test data
get_train_test_data(indexEvent, outcomeEvent)
```

```
## Warning in inner_join(y, X, by = "id"): Detected an unexpected many-to-many relationship between `x`
## i Row 56 of `x` matches multiple rows in `y`.
## i Row 10453 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.

## Warning in inner_join(y, X, by = "id"): Detected an unexpected many-to-many relationship between `x`
## i Row 1858 of `x` matches multiple rows in `y`.
## i Row 6521 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```r
classification_cutoff = get_classification_cutoff(indexEvent, outcomeEvent)
train_data = data_processing(train, classification_cutoff)
test_data = data_processing(test, classification_cutoff)
```

```r
# If you want to watch the percentages between "Yes" and "No" label, run this code.
get_percentage(train_data, indexEvent, outcomeEvent)
```



```
lr_return = logistic_regression(train_data, test_data)
```

```
## [1] "Logistic Regression (Validation) model prediction accuracy:"
## Balanced accuracy: 71.04%
## F1 score: 65.86%
## [1] "Logistic Regression model prediction accuracy:"
## Balanced accuracy: 68.71%
## F1 score: 65.50%
```

```
accuracy_lr_dataframe = lr_return$metrics_lr_dataframe
accuracy_lr = lr_return$metrics_lr
```

```
dt_return = decision_tree(train_data, test_data)
```

```
## [1] "Decision Tree (Validation) model prediction accuracy:"
## Balanced accuracy: 68.54%
## F1 score: 63.51%
## [1] "Decision Tree model prediction accuracy:"
## Balanced accuracy: 70.90%
## F1 score: 67.73%
```

```
accuracy_dt_dataframe = dt_return$metrics_dt_dataframe
accuracy_dt = dt_return$metrics_dt
```

```
nb_return = naive_bayes(train_data, test_data)
```

```
## [1] "Naive Bayes (Validation) model prediction accuracy:"
## Balanced accuracy: 69.93%
## F1 score: 33.92%
## [1] "Naive Bayes model prediction accuracy:"
## Balanced accuracy: 71.85%
## F1 score: 46.62%
```

```
accuracy_nb_dataframe = nb_return$metrics_nb_dataframe
accuracy_nb = nb_return$metrics_nb
```

```
xgb_return = XG_Boost(train_data, test_data)
```

```
## [1] "XGBoost (Validation) model prediction accuracy:"
## Balanced accuracy: 73.86%
## F1 score: 77.19%
## [1] "XGBoost model prediction accuracy:"
## Balanced accuracy: 67.48%
## F1 score: 72.48%
```

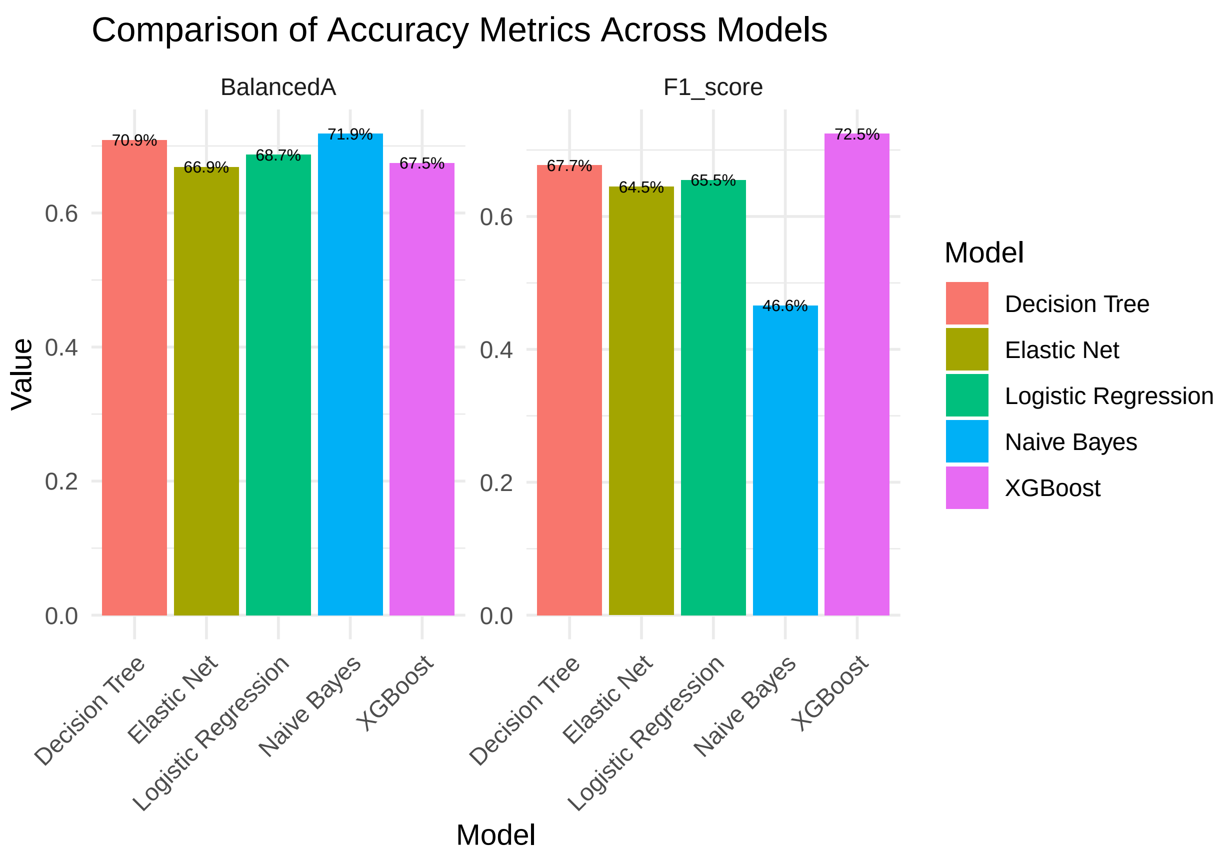
```
accuracy_xgb_dataframe = xgb_return$metrics_xgb_dataframe
accuracy_xgb = xgb_return$metrics_xgb
```

```r
en_return = elastic_net(train_data, test_data)
```

```
## [1] "Elastic Net (Validation) model prediction accuracy:"
## Balanced accuracy: 71.19%
## F1 score: 66.03%
## [1] "Elastic Net model prediction accuracy:"
## Balanced accuracy: 66.87%
## F1 score: 64.48%
```

```r
accuracy_en_dataframe = en_return$metrics_en_dataframe
accuracy_en = en_return$metrics_en
```

```r
# compare all the classification models
metrics_list_BR <- list(
  list(accuracy_lr, "Logistic Regression"),
  list(accuracy_dt, "Decision Tree"),
  list(accuracy_nb, "Naive Bayes"),
  list(accuracy_xgb, "XGBoost"),
  list(accuracy_en, "Elastic Net")
)
accuracy_comparison_plot(metrics_list_BR)
```



```r
# Show the final dataframe for all classification models,
# including the classification model name, accuracy, data combination name.
data_name_BR <- paste(indexEvent, "+", outcomeEvent)
accuracy_dataframe_list_BR <- list(accuracy_lr_dataframe, accuracy_dt_dataframe,
                                   accuracy_nb_dataframe, accuracy_xgb_dataframe,
                                   accuracy_en_dataframe)
combined_results_BR <- combine_classification_results(accuracy_dataframe_list_BR, data_name_BR)

# display the combined dataframe
pander(combined_results_BR, caption = "Classification Model Performance")
```

Table 2: Classification Model Performance

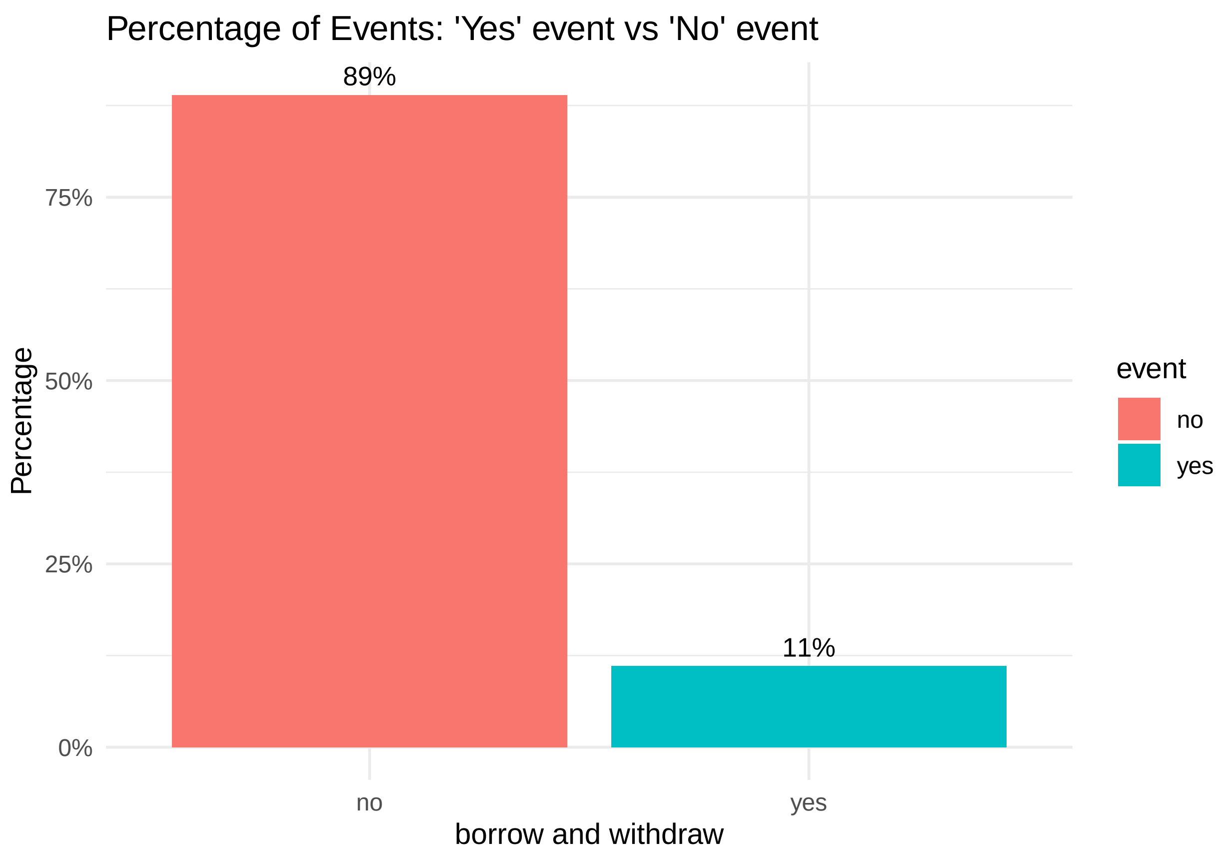
Model	Balanced_Accuracy	F1_Score	Data_Combination
Logistic Regression	68.71%	65.50%	borrow + repay
Decision Tree	70.90%	67.73%	borrow + repay
Naive Bayes	71.85%	46.62%	borrow + repay
XGBoost	67.48%	72.48%	borrow + repay
Elastic Net	66.87%	64.48%	borrow + repay

```r
# set the indexEvent and outcomeEvent
indexEvent = "borrow"
outcomeEvent = "withdraw"

# load the corresponding train and test data
get_train_test_data(indexEvent, outcomeEvent)
```

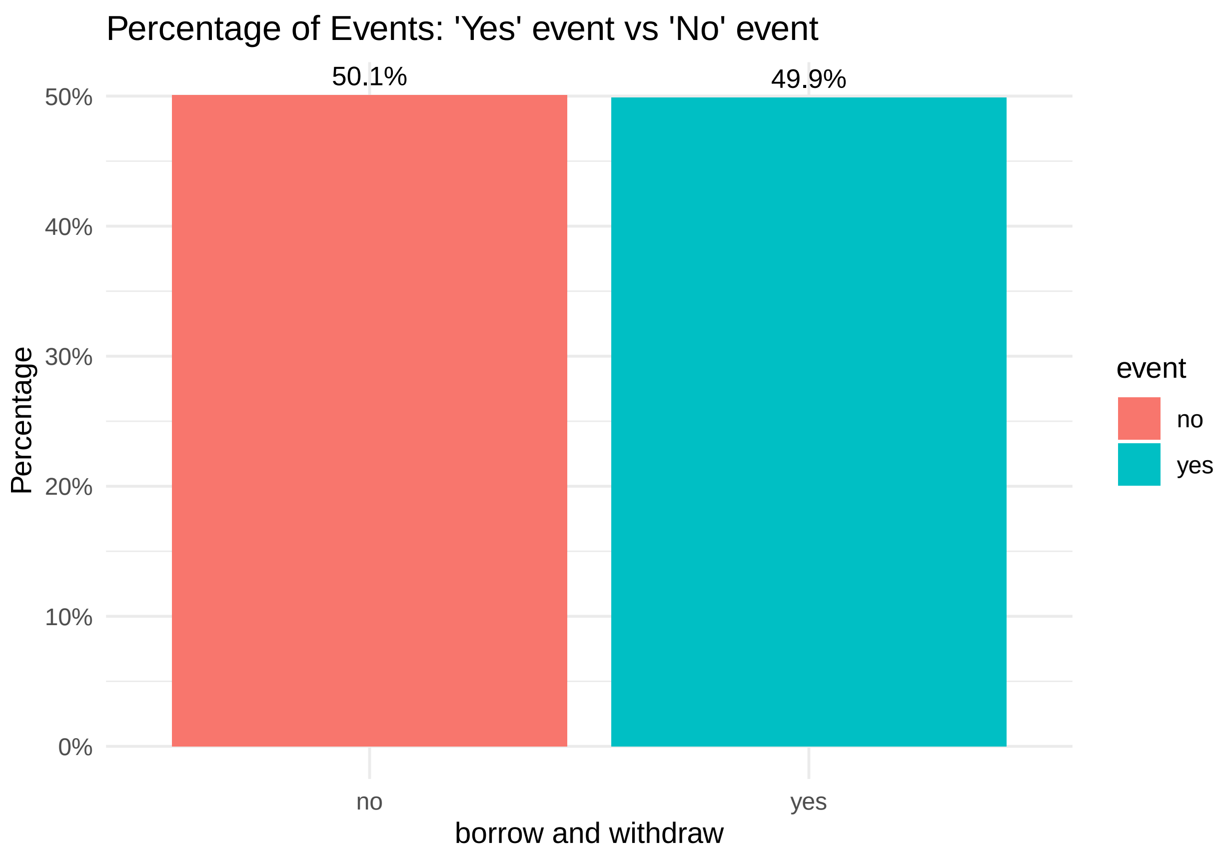
```
## Warning in inner_join(y, X, by = "id"): Detected an unexpected many-to-many relationship between `x`
## i Row 465 of `x` matches multiple rows in `y`.
## i Row 639 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.

## Warning in inner_join(y, X, by = "id"): Detected an unexpected many-to-many relationship between `x`
## i Row 285 of `x` matches multiple rows in `y`.
## i Row 338 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```r
classification_cutoff = get_classification_cutoff(indexEvent, outcomeEvent)
train_data = data_processing(train, classification_cutoff)
test_data = data_processing(test, classification_cutoff)
```

```r
# If you want to watch the percentages between "Yes" and "No" label, run this code.
get_percentage(train_data, indexEvent, outcomeEvent)
```




```
train_data <- smote_data(train_data)
get_percentage(train_data, indexEvent, outcomeEvent)
```




```r
lr_return = logistic_regression(train_data, test_data)
```

```
## [1] "Logistic Regression (Validation) model prediction accuracy:"
## Balanced accuracy: 69.57%
## F1 score: 69.27%
## [1] "Logistic Regression model prediction accuracy:"
## Balanced accuracy: 61.39%
## F1 score: 40.81%
```

```r
accuracy_lr_dataframe = lr_return$metrics_lr_dataframe
accuracy_lr = lr_return$metrics_lr
```

```r
dt_return = decision_tree(train_data, test_data)
```

```
## [1] "Decision Tree (Validation) model prediction accuracy:"
## Balanced accuracy: 91.89%
## F1 score: 91.82%
## [1] "Decision Tree model prediction accuracy:"
## Balanced accuracy: 63.60%
## F1 score: 10.97%
```

```r
accuracy_dt_dataframe = dt_return$metrics_dt_dataframe
accuracy_dt = dt_return$metrics_dt
```

```r
nb_return = naive_bayes(train_data, test_data)
```

```
## [1] "Naive Bayes (Validation) model prediction accuracy:"
## Balanced accuracy: 88.23%
## F1 score: 86.36%
## [1] "Naive Bayes model prediction accuracy:"
## Balanced accuracy: 66.13%
## F1 score: 45.23%
```

```r
accuracy_nb_dataframe = nb_return$metrics_nb_dataframe
accuracy_nb = nb_return$metrics_nb
```

```r
xgb_return = XG_Boost(train_data, test_data)
```

```
## [1] "XGBoost (Validation) model prediction accuracy:"
## Balanced accuracy: 99.29%
## F1 score: 99.29%
## [1] "XGBoost model prediction accuracy:"
## Balanced accuracy: 54.73%
## F1 score: 92.41%
```

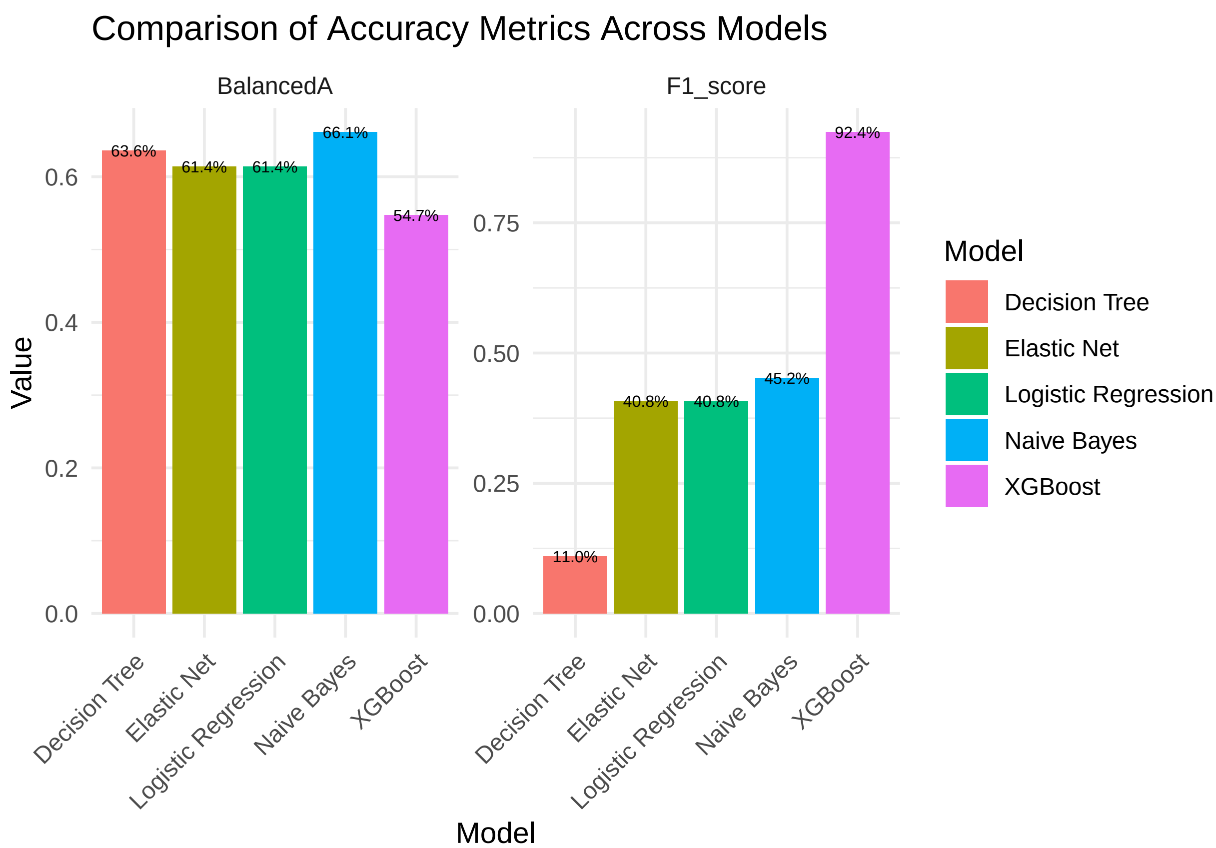
```r
accuracy_xgb_dataframe = xgb_return$metrics_xgb_dataframe
accuracy_xgb = xgb_return$metrics_xgb
```

```r
en_return = elastic_net(train_data, test_data)
```

```
## [1] "Elastic Net (Validation) model prediction accuracy:"
## Balanced accuracy: 69.57%
## F1 score: 69.27%
## [1] "Elastic Net model prediction accuracy:"
## Balanced accuracy: 61.40%
## F1 score: 40.82%
```

```r
accuracy_en_dataframe = en_return$metrics_en_dataframe
accuracy_en = en_return$metrics_en
```

```r
# compare all the classification models
metrics_list_BW <- list(
  list(accuracy_lr, "Logistic Regression"),
  list(accuracy_dt, "Decision Tree"),
  list(accuracy_nb, "Naive Bayes"),
  list(accuracy_xgb, "XGBoost"),
  list(accuracy_en, "Elastic Net")
)
accuracy_comparison_plot(metrics_list_BW)
```




```r
# Show the final dataframe for all classification models,
# including the classification model name, accuracy, data combination name.
data_name_BW <- paste(indexEvent, "+", outcomeEvent)
accuracy_dataframe_list_BW <- list(accuracy_lr_dataframe, accuracy_dt_dataframe,
                                   accuracy_nb_dataframe, accuracy_xgb_dataframe,
                                   accuracy_en_dataframe)
combined_results_BW <- combine_classification_results(accuracy_dataframe_list_BW, data_name_BW)

# display the combined dataframe
pander(combined_results_BW, caption = "Classification Model Performance")
```

Table 3: Classification Model Performance

Model	Balanced_Accuracy	F1_Score	Data_Combination
Logistic Regression	61.39%	40.81%	borrow + withdraw
Decision Tree	63.60%	10.97%	borrow + withdraw
Naive Bayes	66.13%	45.23%	borrow + withdraw
XGBoost	54.73%	92.41%	borrow + withdraw
Elastic Net	61.40%	40.82%	borrow + withdraw

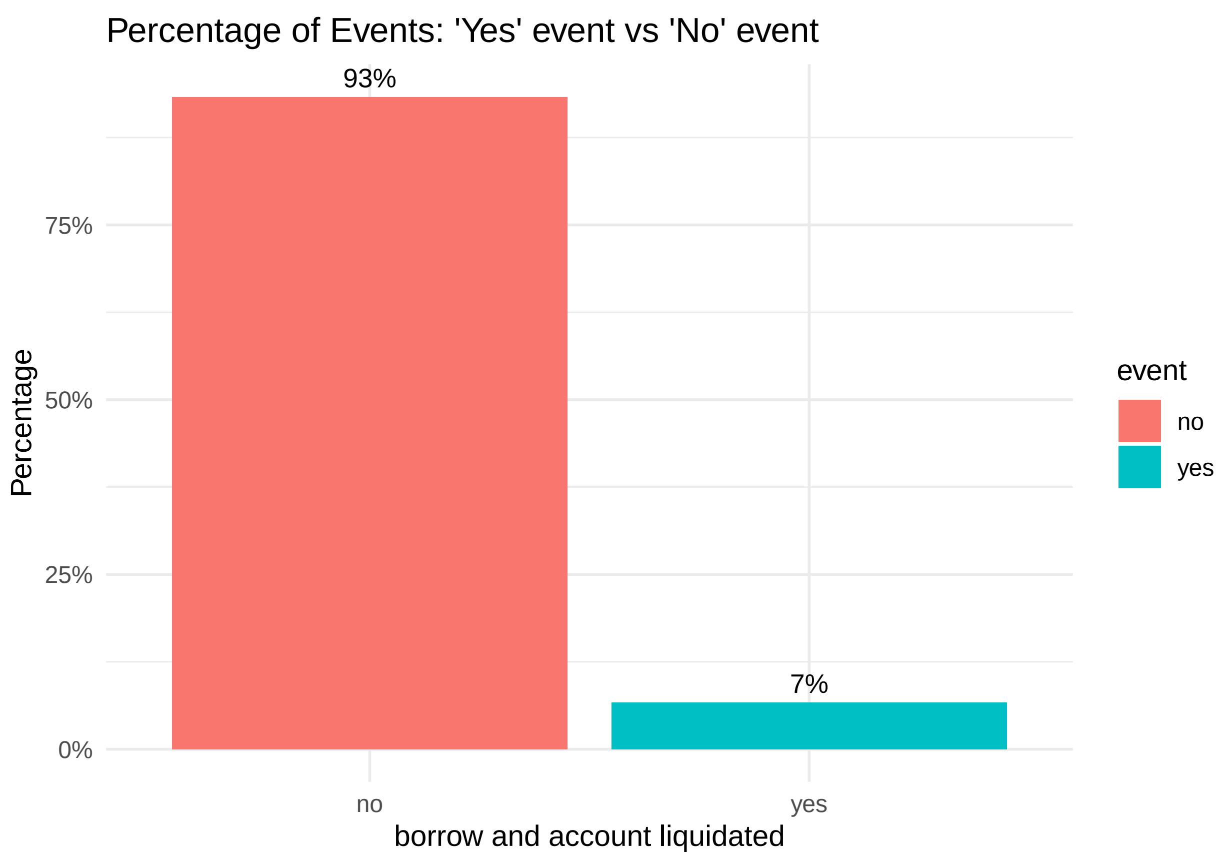
```r
# set the indexEvent and outcomeEvent
indexEvent = "borrow"
outcomeEvent = "account liquidated"
```

```r
# load the corresponding train and test data
get_train_test_data(indexEvent, outcomeEvent)
```

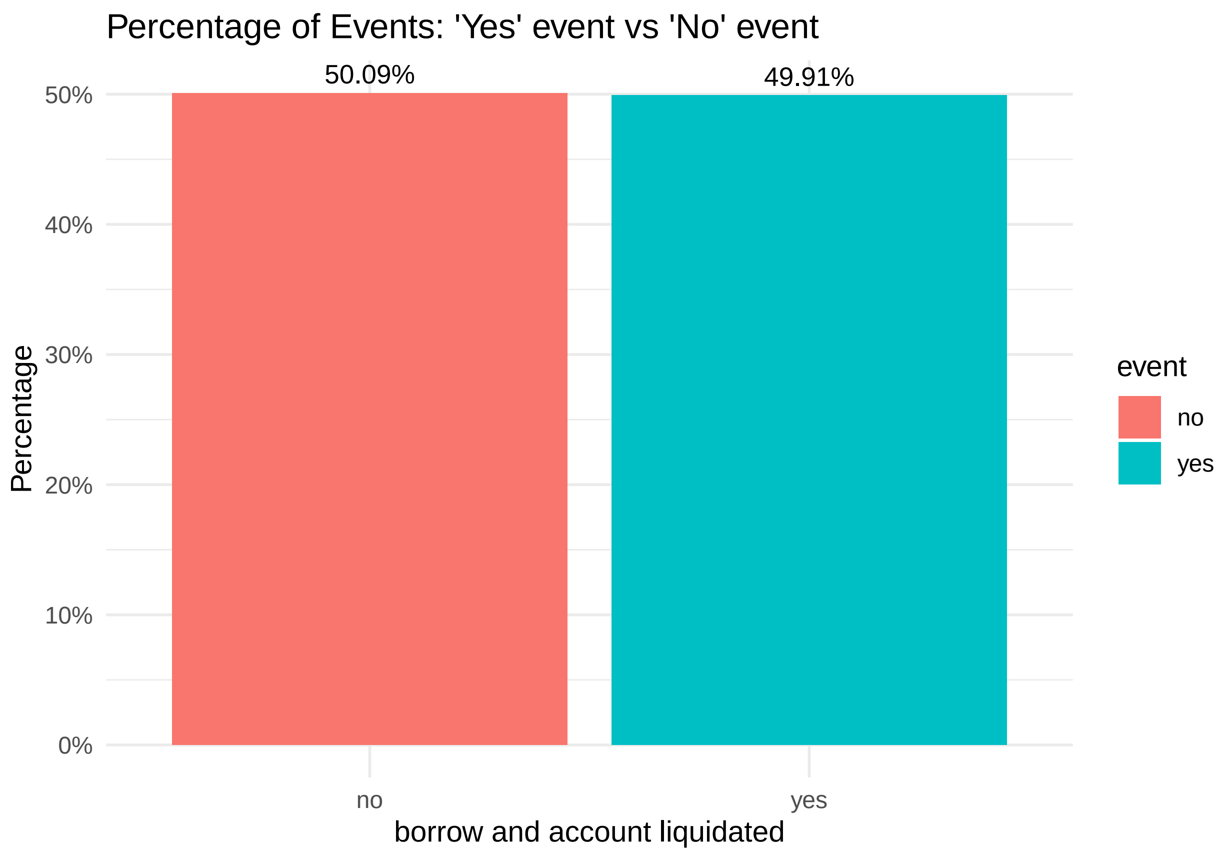
```
## Warning in inner_join(y, X, by = "id"): Detected an unexpected many-to-many relationship between `x`
## i Row 570 of `x` matches multiple rows in `y`.
## i Row 637 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.

## Warning in inner_join(y, X, by = "id"): Detected an unexpected many-to-many relationship between `x`
## i Row 82 of `x` matches multiple rows in `y`.
## i Row 82 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```r
classification_cutoff = get_classification_cutoff(indexEvent, outcomeEvent)
train_data = data_processing(train, classification_cutoff)
test_data = data_processing(test, classification_cutoff)
```

```r
# If you want to watch the percentages between "Yes" and "No" label, run this code.
get_percentage(train_data, indexEvent, outcomeEvent)
```



```r
train_data <- smote_data(train_data)
get_percentage(train_data, indexEvent, outcomeEvent)
```




```r
lr_return = logistic_regression(train_data, test_data)
```

```
## [1] "Logistic Regression (Validation) model prediction accuracy:"
## Balanced accuracy: 71.77%
## F1 score: 70.98%
## [1] "Logistic Regression model prediction accuracy:"
## Balanced accuracy: 50.44%
## F1 score: 3.52%
```

```r
accuracy_lr_dataframe = lr_return$metrics_lr_dataframe
accuracy_lr = lr_return$metrics_lr
```

```r
dt_return = decision_tree(train_data, test_data)
```

```
## [1] "Decision Tree (Validation) model prediction accuracy:"
## Balanced accuracy: 98.99%
## F1 score: 98.99%
## [1] "Decision Tree model prediction accuracy:"
## Balanced accuracy: 50.74%
## F1 score: 3.06%
```

```r
accuracy_dt_dataframe = dt_return$metrics_dt_dataframe
accuracy_dt = dt_return$metrics_dt
```

```r
nb_return = naive_bayes(train_data, test_data)
```

```
## [1] "Naive Bayes (Validation) model prediction accuracy:"
## Balanced accuracy: 99.03%
## F1 score: 99.02%
## [1] "Naive Bayes model prediction accuracy:"
## Balanced accuracy: 50.68%
## F1 score: 3.08%
```

```r
accuracy_nb_dataframe = nb_return$metrics_nb_dataframe
accuracy_nb = nb_return$metrics_nb
```

```r
xgb_return = XG_Boost(train_data, test_data)
```

```
## [1] "XGBoost (Validation) model prediction accuracy:"
## Balanced accuracy: 99.98%
## F1 score: 99.98%
## [1] "XGBoost model prediction accuracy:"
## Balanced accuracy: 49.35%
## F1 score: 99.03%
```

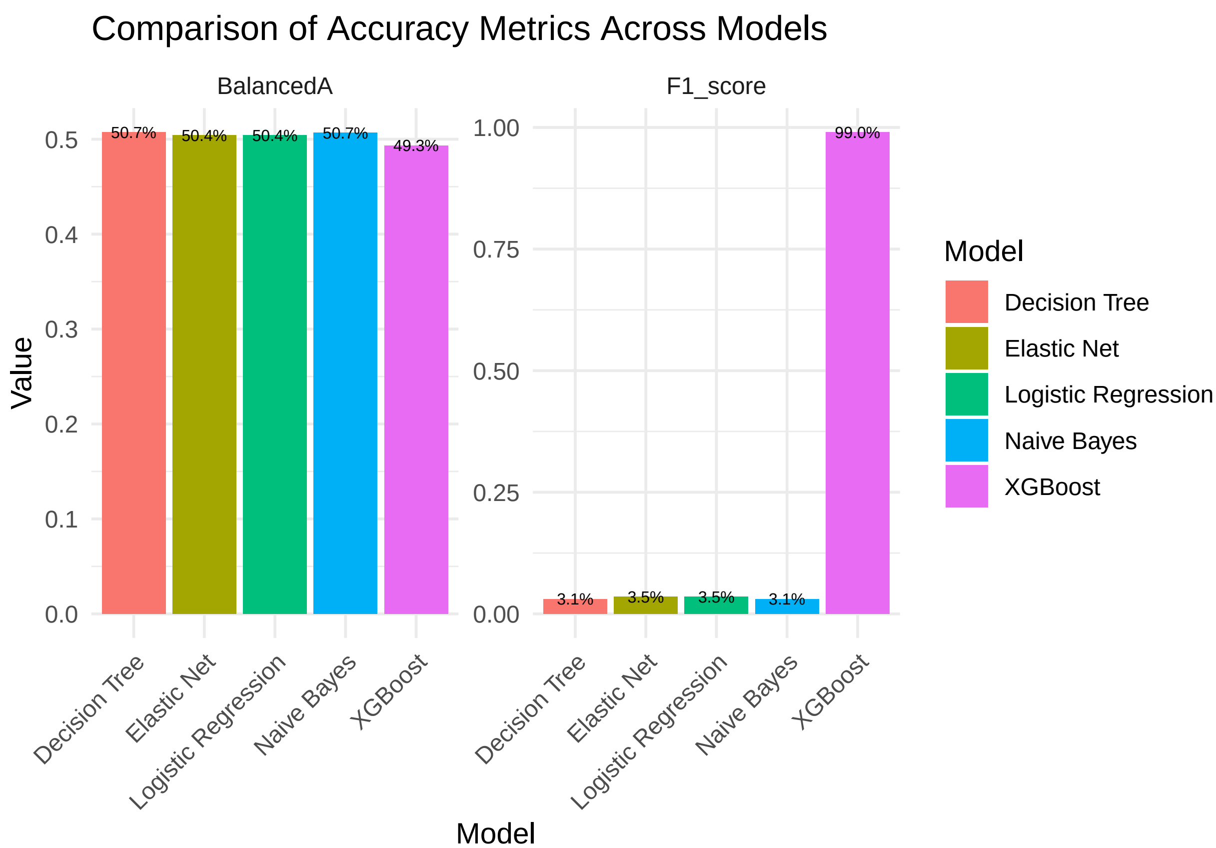
```r
accuracy_xgb_dataframe = xgb_return$metrics_xgb_dataframe
accuracy_xgb = xgb_return$metrics_xgb
```

```r
en_return = elastic_net(train_data, test_data)
```

```
## [1] "Elastic Net (Validation) model prediction accuracy:"
## Balanced accuracy: 71.79%
## F1 score: 71.00%
## [1] "Elastic Net model prediction accuracy:"
## Balanced accuracy: 50.44%
## F1 score: 3.52%
```

```r
accuracy_en_dataframe = en_return$metrics_en_dataframe
accuracy_en = en_return$metrics_en
```

```r
# compare all the classification models
metrics_list_BAL <- list(
  list(accuracy_lr, "Logistic Regression"),
  list(accuracy_dt, "Decision Tree"),
  list(accuracy_nb, "Naive Bayes"),
  list(accuracy_xgb, "XGBoost"),
  list(accuracy_en, "Elastic Net")
)
accuracy_comparison_plot(metrics_list_BAL)
```




```r
# Show the final dataframe for all classification models,
# including the classification model name, accuracy, data combination name.
data_name_BAL <- paste(indexEvent, "+", outcomeEvent)
accuracy_dataframe_list_BAL <- list(accuracy_lr_dataframe, accuracy_dt_dataframe,
                                    accuracy_nb_dataframe, accuracy_xgb_dataframe,
                                    accuracy_en_dataframe)
combined_results_BAL <- combine_classification_results(accuracy_dataframe_list_BAL, data_name_BAL)

# display the combined dataframe
pander(combined_results_BAL, caption = "Classification Model Performance")
```

Table 4: Classification Model Performance

Model	Balanced_Accuracy	F1_Score	Data_Combination
Logistic Regression	50.44%	3.52%	borrow + account liquidated
Decision Tree	50.74%	3.06%	borrow + account liquidated
Naive Bayes	50.68%	3.08%	borrow + account liquidated
XGBoost	49.35%	99.03%	borrow + account liquidated
Elastic Net	50.44%	3.52%	borrow + account liquidated

Classification Model Performance For All Data Combinations

After we run all the data combinations, we can use the `combine_accuracy_dataframes` to combine all the classification models' performance into one dataframe.

```r
combined_classification_results <- combine_accuracy_dataframes(
  list(combined_results_BAL, combined_results_BD, combined_results_BR, combined_results_BW))
pander(combined_classification_results, caption = "Classification Model Performance for all data")
```

Table 5: Classification Model Performance for all data

Model	Balanced_Accuracy	F1_Score	Data_Combination
Logistic Regression	50.44%	3.52%	borrow + account liquidated
Decision Tree	50.74%	3.06%	borrow + account liquidated
Naive Bayes	50.68%	3.08%	borrow + account liquidated
XGBoost	49.35%	99.03%	borrow + account liquidated
Elastic Net	50.44%	3.52%	borrow + account liquidated
Logistic Regression	62.93%	46.12%	borrow + deposit
Decision Tree	63.74%	11.70%	borrow + deposit
Naive Bayes	66.04%	45.49%	borrow + deposit
XGBoost	64.53%	89.90%	borrow + deposit
Elastic Net	62.93%	46.11%	borrow + deposit
Logistic Regression	68.71%	65.50%	borrow + repay
Decision Tree	70.90%	67.73%	borrow + repay
Naive Bayes	71.85%	46.62%	borrow + repay
XGBoost	67.48%	72.48%	borrow + repay
Elastic Net	66.87%	64.48%	borrow + repay
Logistic Regression	61.39%	40.81%	borrow + withdraw
Decision Tree	63.60%	10.97%	borrow + withdraw
Naive Bayes	66.13%	45.23%	borrow + withdraw
XGBoost	54.73%	92.41%	borrow + withdraw
Elastic Net	61.40%	40.82%	borrow + withdraw

Generating Dataframe For Specified Accuracy

This section is only for a special need, not required for the whole pipeline workflow!!!

In this section, the final output is a combined data frame that consolidates performance metrics for multiple classification models across different data scenarios. Each row represents a specific scenario (e.g., "borrow + withdraw" or "borrow + repay"), while the columns display the selected performance metric (e.g., "balanced_accuracy") and the corresponding values for each classification model (e.g., Logistic Regression, Decision Tree).

```r
ba_accuracy_dataframe_BAL <- specific_accuracy_statistics(data_name_BAL, "balanced_accuracy",
                                                          metrics_list_BAL)
ba_accuracy_dataframe_BD <- specific_accuracy_statistics(data_name_BD, "balanced_accuracy",
                                                          metrics_list_BD)
ba_accuracy_dataframe_BR <- specific_accuracy_statistics(data_name_BR, "balanced_accuracy",
                                                          metrics_list_BR)
ba_accuracy_dataframe_BW <- specific_accuracy_statistics(data_name_BW, "balanced_accuracy",
                                                          metrics_list_BW)
combined_accuracy_dataframe <- combine_accuracy_dataframes(
  list(ba_accuracy_dataframe_BAL, ba_accuracy_dataframe_BD, ba_accuracy_dataframe_BR,
       ba_accuracy_dataframe_BW))
pander(combined_accuracy_dataframe, caption = "Combined accuracy dataframe")
```

Table 6: Combined accuracy dataframe (continued below)

balanced_accuracy	Logistic.Regression	Decision.Tree
borrow + account liquidated	50.4	50.7
borrow + deposit	62.9	63.7
borrow + repay	68.7	70.9
borrow + withdraw	61.4	63.6

Naive.Bayes	XGBoost	Elastic.Net
50.7	49.3	50.4
66	64.5	62.9
71.9	67.5	66.9
66.1	54.7	61.4