DMLR DeFi Survival Data Pipeline Example

Hanzhen Qin(qinh2)

24 January 2025

Survival Data Pipeline

- Project name: DMLR DeFi-LTM

```r
source("~/DMLR_DeFi_Survival_Dataset_And_Benchmark/DeFi_source/survivalData_pipeline/data_preprocessing
source("~/DMLR_DeFi_Survival_Dataset_And_Benchmark/DeFi_source/survivalData_pipeline/model_evaluation_vi
source("~/DMLR_DeFi_Survival_Dataset_And_Benchmark/DeFi_source/survivalData_pipeline/classification_mode
source("~/DMLR_DeFi_Survival_Dataset_And_Benchmark/DeFi_source/survivalData_pipeline/get_classification_
```

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

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

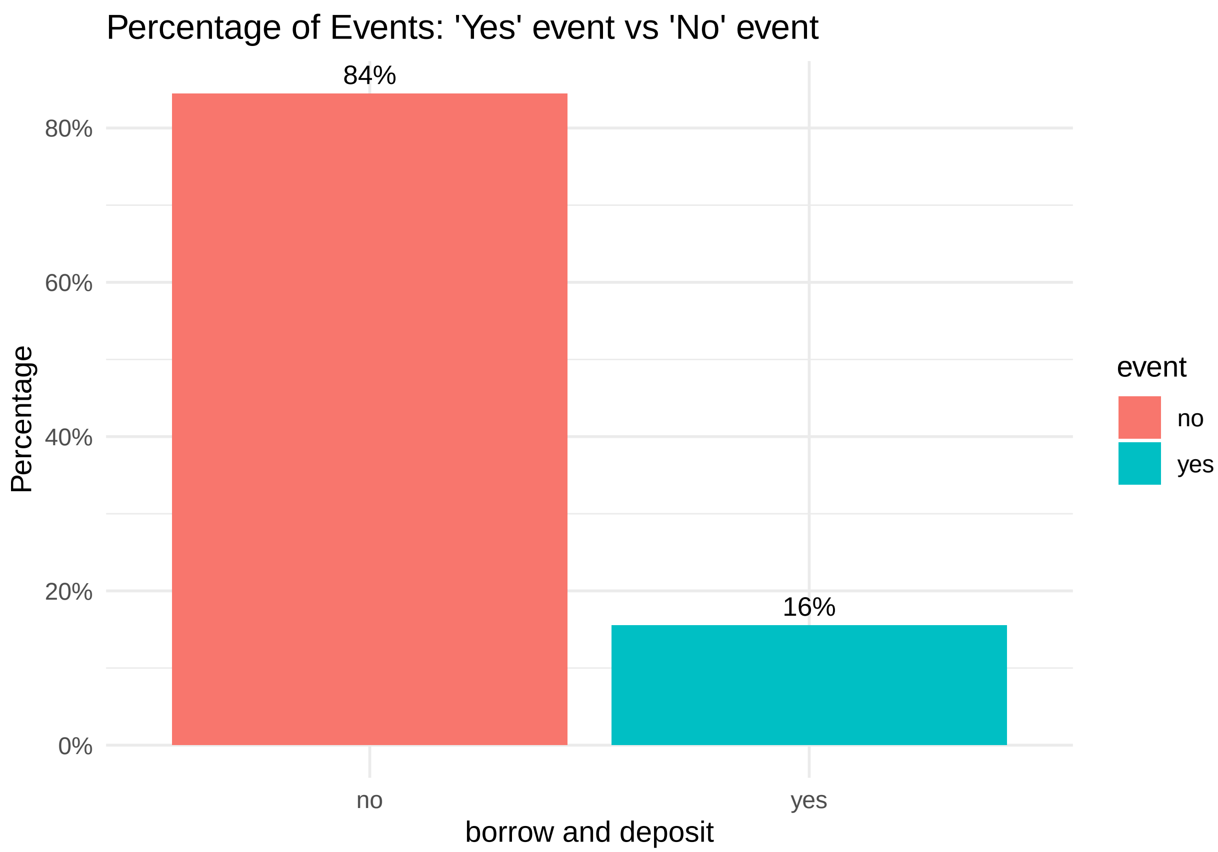
```
## Warning in inner_join(y, X, by = "id"): Detected an unexpected many-to-many relationship between `x`
## i Row 405 of `x` matches multiple rows in `y`.
## i Row 639 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to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

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

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

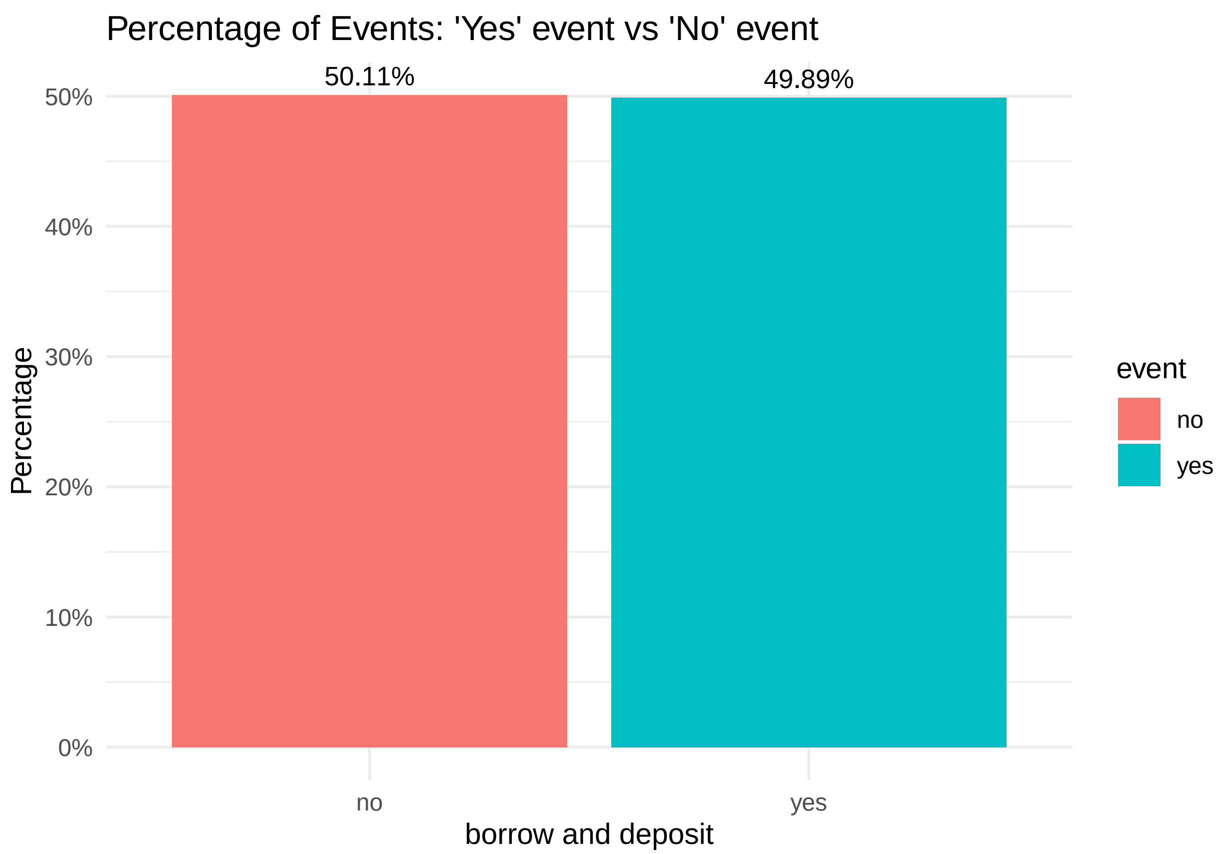


If the ratio of "No" labels to "Yes" labels in the dataset is significantly imbalanced, we can utilize the `smote_data` function to generate a new, more balanced dataset. This balanced dataset ensures that both classes are better represented, helping to mitigate the bias introduced by class imbalance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

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

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

```
# 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 model prediction accuracy:"
## Class accuracy (Specificity): 93%
## Negative 1 accuracy (Sensitivity/Recall): 33%
## Balanced accuracy: 63%
## Overall accuracy: 67%
## Precision: 80%
## F1 score: 47%
```

```
accuracy_lr_dataframe = lr_return$metrics_lr_dataframe
accuracy_lr = lr_return$metrics_lr
pander(accuracy_lr_dataframe)
```

Table 1: Table continues below

Model	Class_Accuracy	Negative_1_Accuracy	Balanced_Accuracy
Logistic Regression	93%	33%	63%

Overall_Accuracy	Precision	F1_Score
67%	80%	47%

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

```
## [1] "Decision tree model prediction accuracy:"
## Class accuracy (Specificity): 83%
## Negative 1 accuracy (Sensitivity/Recall): 44%
## Balanced accuracy: 64%
## Overall accuracy: 81%
## Precision: 7%
## F1 score: 13%
```

```r
accuracy_dt_dataframe = dt_return$metrics_dt_dataframe
accuracy_dt = dt_return$metrics_dt
pander(accuracy_dt_dataframe)
```

Table 3: Table continues below

Model	Class_Accuracy	Negative_1_Accuracy	Balanced_Accuracy
Decision Tree	83%	44%	64%

Overall_Accuracy	Precision	F1_Score
81%	7%	13%

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

```
## [1] "Naive Bayes model prediction accuracy:"
## Class accuracy (Specificity): 88%
## Negative 1 accuracy (Sensitivity/Recall): 44%
## Balanced accuracy: 66%
## Overall accuracy: 79%
## Precision: 47%
## F1 score: 45%
```

```r
accuracy_nb_dataframe = nb_return$metrics_dataframe
accuracy_nb = nb_return$metrics
pander(accuracy_nb_dataframe)
```

Table 5: Table continues below

Model	Class_Accuracy	Negative_1_Accuracy	Balanced_Accuracy
Naive Bayes	88%	44%	66%

Overall_Accuracy	Precision	F1_Score
79%	47%	45%

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

```
## [1] "XGBoost model prediction accuracy:"
## Class accuracy (Specificity): 82%
## Negative 1 accuracy (Sensitivity/Recall): 41%
## Balanced accuracy: 62%
## Overall accuracy: 82%
## Precision: 1%
## F1 score: 2%
```

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

```
gbm_return = GBM(train_data, test_data)
```

```
## [1] "GBM model prediction accuracy:"
## Class accuracy (Specificity): 83%
## Negative 1 accuracy (Sensitivity/Recall): 41%
## Balanced accuracy: 62%
## Overall accuracy: 80%
## Precision: 16%
## F1 score: 23%
```

```
accuracy_gbm_dataframe = gbm_return$metrics_gbm_dataframe
accuracy_gbm = gbm_return$metrics_gbm
```

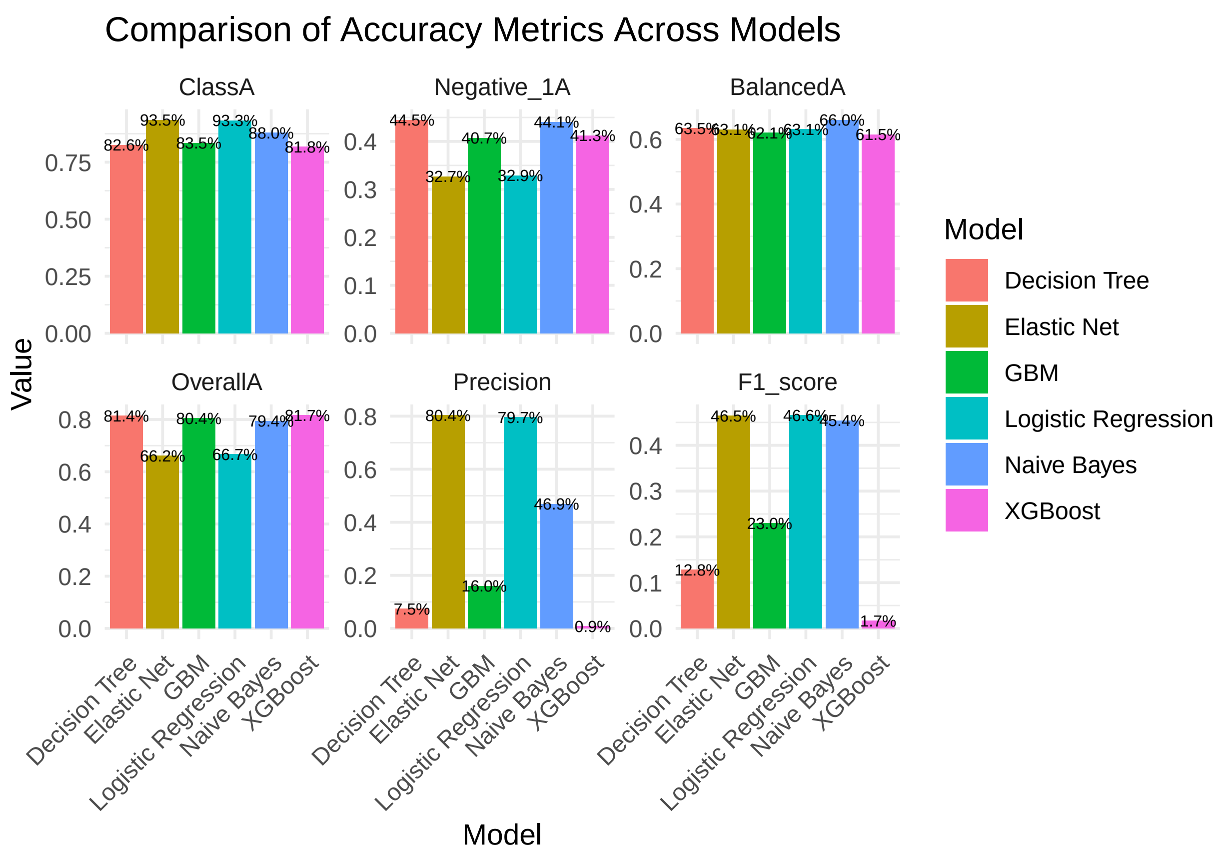
```
en_return = elastic_net(train_data, test_data)
```

```
## [1] "Elastic Net model prediction accuracy:"
## Class accuracy (Specificity): 94%
## Negative 1 accuracy (Sensitivity/Recall): 33%
## Balanced accuracy: 63%
## Overall accuracy: 66%
## Precision: 80%
## F1 score: 46%
```

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

```
# 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_gbm, "GBM"),
```

```r
  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_gbm_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 7: Classification Model Performance (continued below)

Model	Class_Accuracy	Negative_1_Accuracy	Balanced_Accuracy
Logistic Regression	93%	33%	63%
Decision Tree	83%	44%	64%
Naive Bayes	88%	44%	66%
XGBoost	82%	41%	62%
GBM	83%	41%	62%

Model	Class_Accuracy	Negative_1_Accuracy	Balanced_Accuracy
Elastic Net	94%	33%	63%

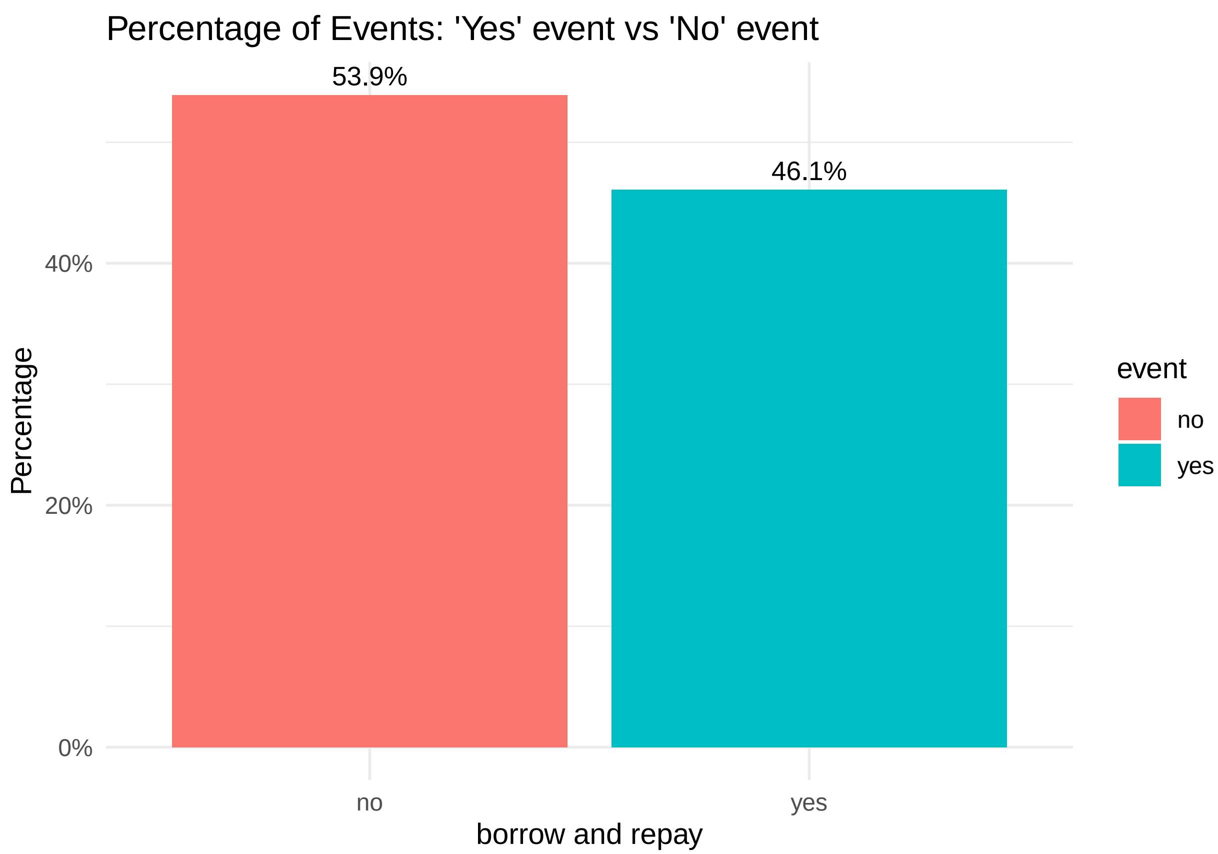
Overall_Accuracy	Precision	F1_Score	Data_Combination
67%	80%	47%	borrow + deposit
81%	7%	13%	borrow + deposit
79%	47%	45%	borrow + deposit
82%	1%	2%	borrow + deposit
80%	16%	23%	borrow + deposit
66%	80%	46%	borrow + deposit

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

# 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.
```

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

```
# 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 model prediction accuracy:"
## Class accuracy (Specificity): 74%
## Negative 1 accuracy (Sensitivity/Recall): 73%
## Balanced accuracy: 74%
## Overall accuracy: 74%
## Precision: 62%
## F1 score: 67%
```

```
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 model prediction accuracy:"
## Class accuracy (Specificity): 76%
## Negative 1 accuracy (Sensitivity/Recall): 66%
## Balanced accuracy: 71%
## Overall accuracy: 71%
## Precision: 69%
## F1 score: 68%
```

```
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 model prediction accuracy:"
## Class accuracy (Specificity): 64%
## Negative 1 accuracy (Sensitivity/Recall): 79%
## Balanced accuracy: 72%
## Overall accuracy: 67%
## Precision: 33%
## F1 score: 47%
```

```
accuracy_nb_dataframe = nb_return$metrics_dataframe
accuracy_nb = nb_return$metrics
```

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

```
## [1] "XGBoost model prediction accuracy:"
## Class accuracy (Specificity): 69%
## Negative 1 accuracy (Sensitivity/Recall): 59%
## Balanced accuracy: 64%
## Overall accuracy: 65%
## Precision: 60%
## F1 score: 59%
```

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

```
gbm_return = GBM(train_data, test_data)
```

```
## [1] "GBM model prediction accuracy:"
## Class accuracy (Specificity): 64%
## Negative 1 accuracy (Sensitivity/Recall): 90%
## Balanced accuracy: 77%
## Overall accuracy: 68%
## Precision: 29%
## F1 score: 44%
```

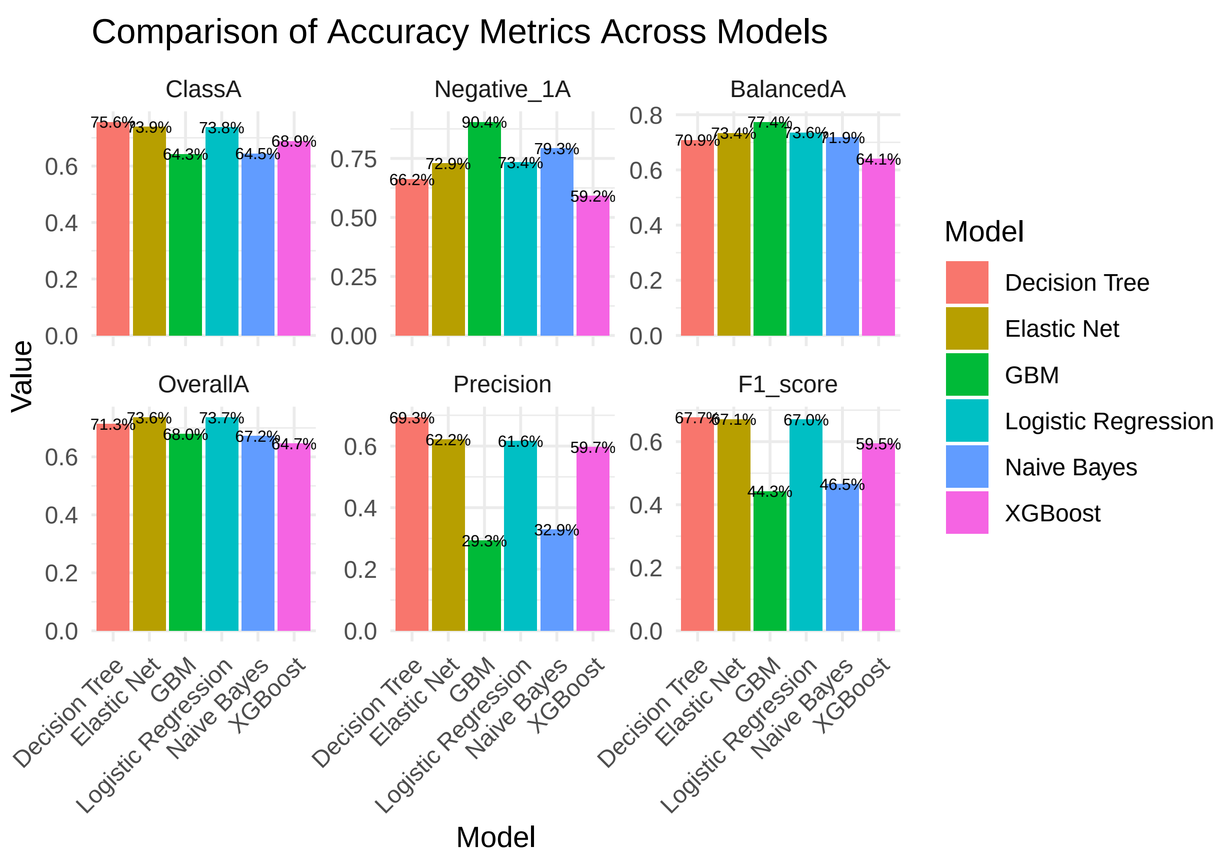
```
accuracy_gbm_dataframe = gbm_return$metrics_gbm_dataframe
accuracy_gbm = gbm_return$metrics_gbm
```

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

```
## [1] "Elastic Net model prediction accuracy:"
## Class accuracy (Specificity): 74%
## Negative 1 accuracy (Sensitivity/Recall): 73%
## Balanced accuracy: 73%
## Overall accuracy: 74%
## Precision: 62%
## F1 score: 67%
```

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

```
# 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_gbm, "GBM"),
  list(accuracy_en, "Elastic Net")
)
accuracy_comparison_plot(metrics_list_BR)
```




```
# 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_gbm_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 9: Classification Model Performance (continued below)

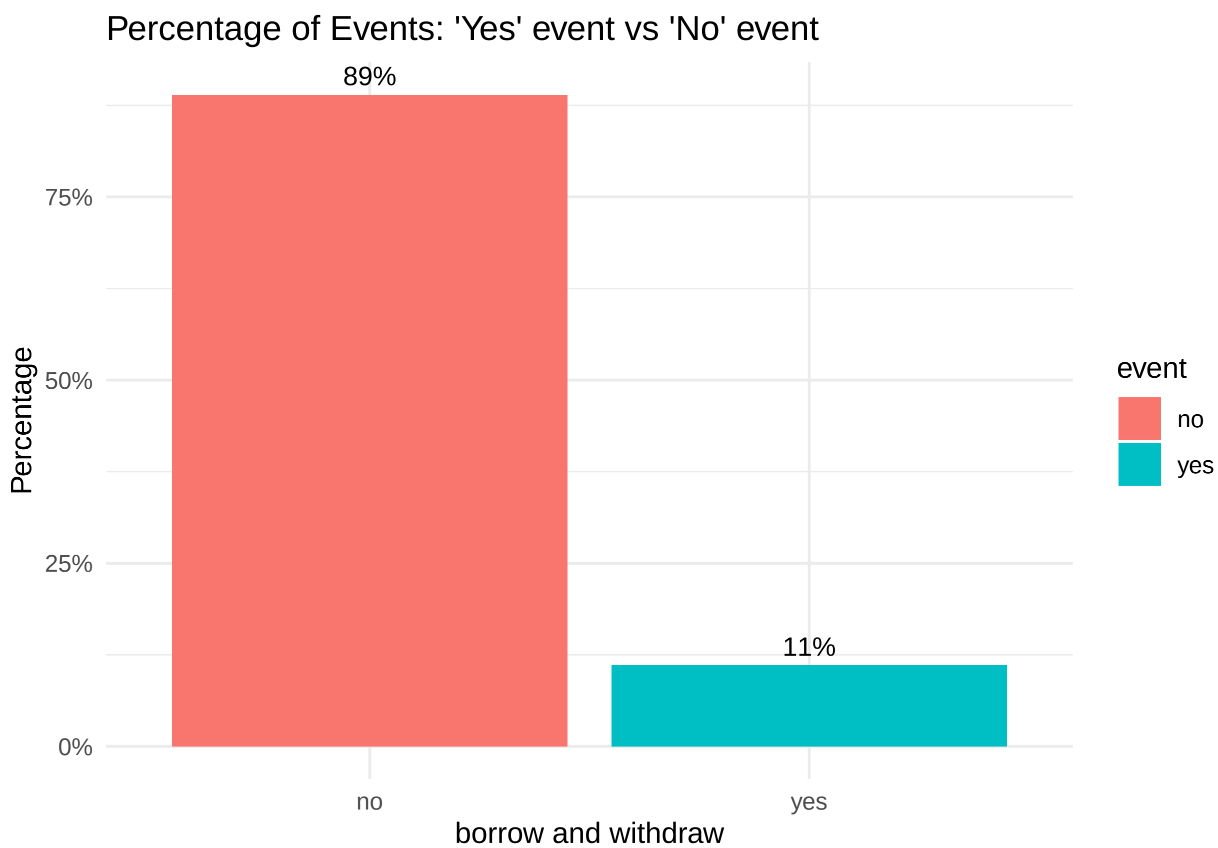
Model	Class_Accuracy	Negative_1_Accuracy	Balanced_Accuracy	Overall_Accuracy	Precision	F1_Score	Data_Combination
Logistic Regression	74%	73%	74%	74%	62%	67%	borrow + repay
Decision Tree	76%	66%	71%	71%	69%	68%	borrow + repay
Naive Bayes	64%	79%	72%	67%	33%	47%	borrow + repay
XGBoost	69%	59%	64%	65%	60%	59%	borrow + repay
GBM	64%	90%	77%	68%	29%	44%	borrow + repay
Elastic Net	74%	73%	73%	74%	62%	67%	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)
```

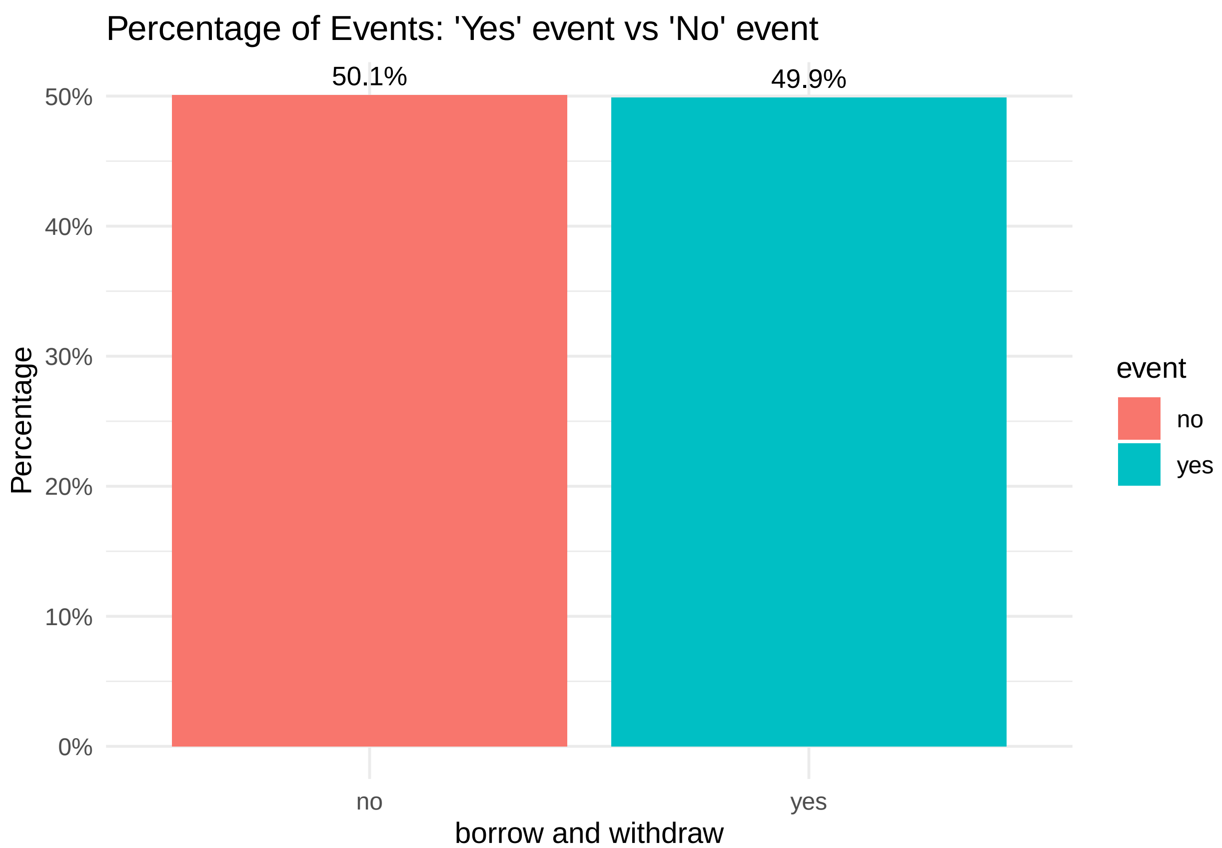
```
## Warning in inner_join(y, X, by = "id"): Detected an unexpected many-to-many relationship between `x`
## i Row 465 of `x` matches multiple rows in `y`.
## i Row 639 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to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




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

```
## [1] "Logistic regression model prediction accuracy:"
## Class accuracy (Specificity): 95%
## Negative 1 accuracy (Sensitivity/Recall): 28%
## Balanced accuracy: 62%
## Overall accuracy: 69%
## Precision: 80%
## F1 score: 41%
```

```
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 model prediction accuracy:"
## Class accuracy (Specificity): 87%
## Negative 1 accuracy (Sensitivity/Recall): 40%
## Balanced accuracy: 64%
## Overall accuracy: 86%
## Precision: 6%
## F1 score: 11%
```

```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 model prediction accuracy:"
## Class accuracy (Specificity): 92%
## Negative 1 accuracy (Sensitivity/Recall): 41%
## Balanced accuracy: 66%
## Overall accuracy: 83%
## Precision: 51%
## F1 score: 45%
```

```r
accuracy_nb_dataframe = nb_return$metrics_dataframe
accuracy_nb = nb_return$metrics
```

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

```
## [1] "XGBoost model prediction accuracy:"
## Class accuracy (Specificity): 86%
## Negative 1 accuracy (Sensitivity/Recall): 12%
## Balanced accuracy: 49%
## Overall accuracy: 86%
## Precision: 0%
## F1 score: 1%
```

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

```r
gbm_return = GBM(train_data, test_data)
```

```
## [1] "GBM model prediction accuracy:"
## Class accuracy (Specificity): 89%
## Negative 1 accuracy (Sensitivity/Recall): 39%
## Balanced accuracy: 64%
## Overall accuracy: 84%
## Precision: 27%
## F1 score: 32%
```

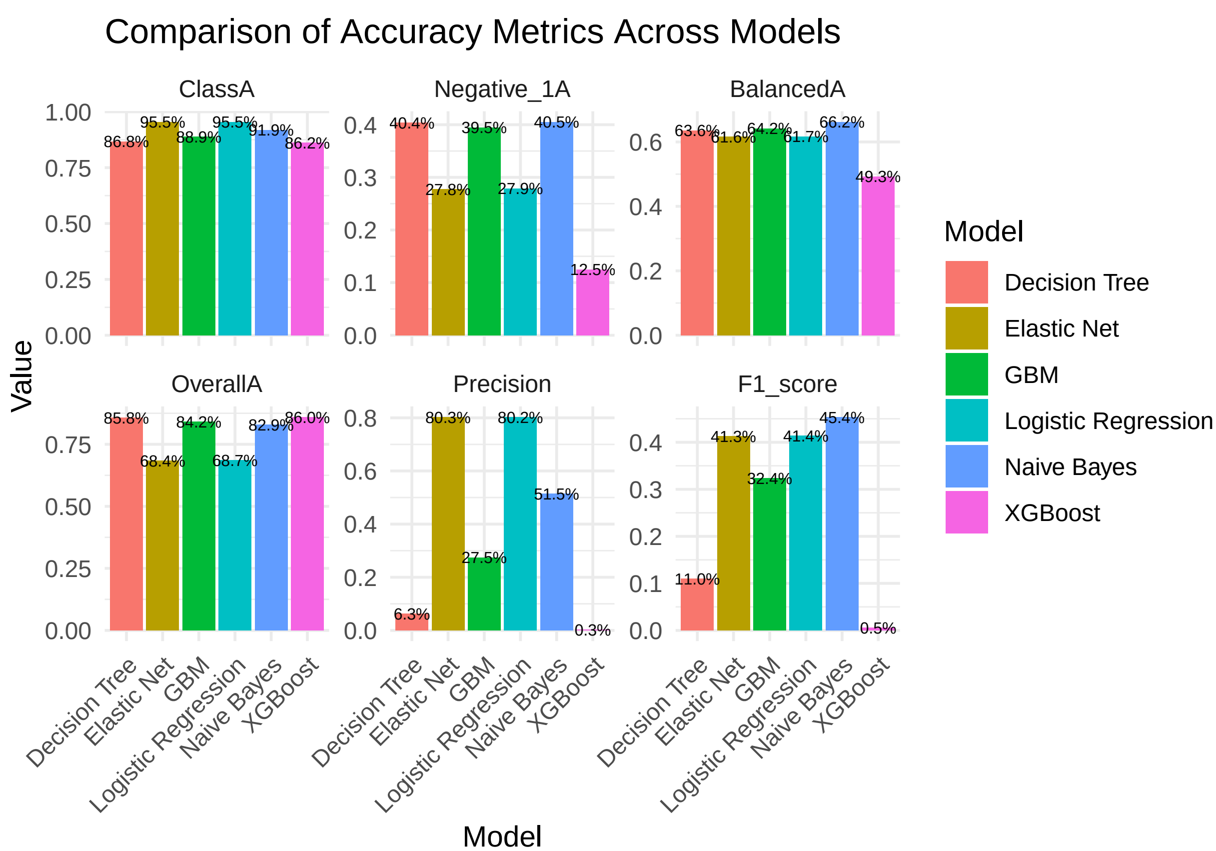
```r
accuracy_gbm_dataframe = gbm_return$metrics_gbm_dataframe
accuracy_gbm = gbm_return$metrics_gbm
```

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

```
## [1] "Elastic Net model prediction accuracy:"
## Class accuracy (Specificity): 95%
## Negative 1 accuracy (Sensitivity/Recall): 28%
## Balanced accuracy: 62%
## Overall accuracy: 68%
## Precision: 80%
## F1 score: 41%
```

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

```
# 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_gbm, "GBM"),
  list(accuracy_en, "Elastic Net")
)
accuracy_comparison_plot(metrics_list_BW)
```




```
# 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_gbm_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 11: Classification Model Performance (continued below)

Model	Class_Accuracy	Negative_1_Accuracy	Balanced_Accuracy
Logistic Regression	95%	28%	62%
Decision Tree	87%	40%	64%
Naive Bayes	92%	41%	66%
XGBoost	86%	12%	49%
GBM	89%	39%	64%
Elastic Net	95%	28%	62%

Overall_Accuracy	Precision	F1_Score	Data_Combination
69%	80%	41%	borrow + withdraw
86%	6%	11%	borrow + withdraw
83%	51%	45%	borrow + withdraw
86%	0%	1%	borrow + withdraw
84%	27%	32%	borrow + withdraw
68%	80%	41%	borrow + withdraw

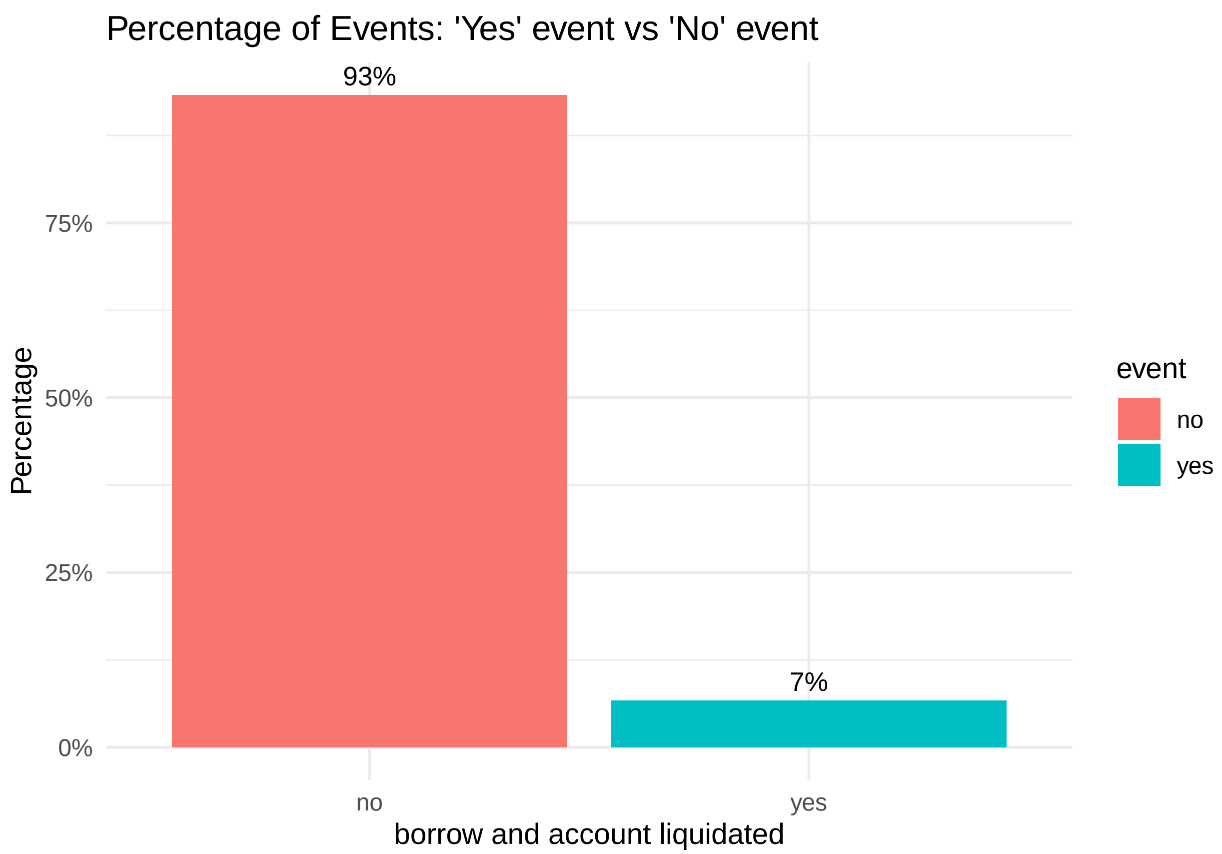
```
# set the indexEvent and outcomeEvent
indexEvent = "borrow"
outcomeEvent = "account liquidated"

# 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.
```

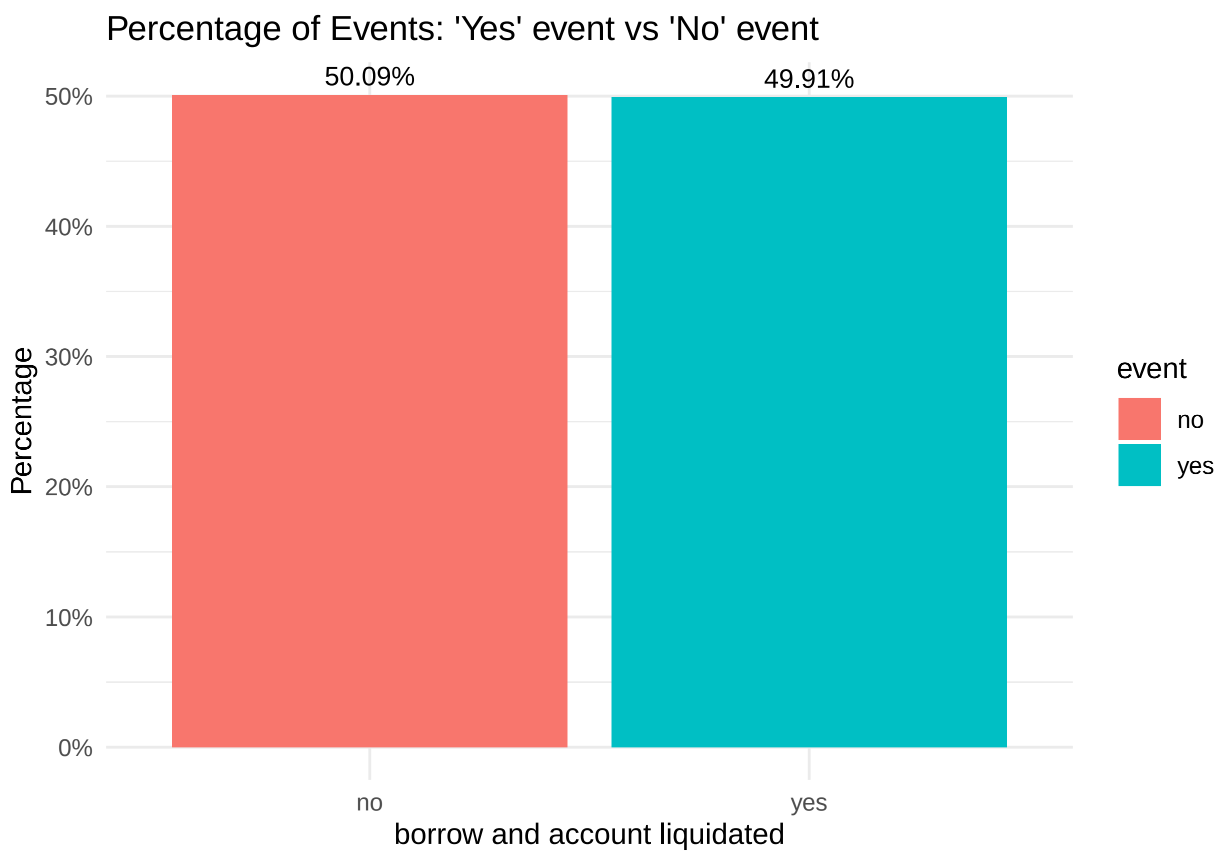
```
## Warning in inner_join(y, X, by = "id"): Detected an unexpected many-to-many relationship between `x`
## i Row 82 of `x` matches multiple rows in `y`.
## i Row 82 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence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

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

```
# 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)
```




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

```
## [1] "Logistic regression model prediction accuracy:"
## Class accuracy (Specificity): 99%
## Negative 1 accuracy (Sensitivity/Recall): 2%
## Balanced accuracy: 50%
## Overall accuracy: 63%
## Precision: 50%
## F1 score: 3%
```

```
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 model prediction accuracy:"
## Class accuracy (Specificity): 100%
## Negative 1 accuracy (Sensitivity/Recall): 2%
## Balanced accuracy: 51%
## Overall accuracy: 19%
## Precision: 99%
## F1 score: 3%
```

```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 model prediction accuracy:"
## Class accuracy (Specificity): 100%
## Negative 1 accuracy (Sensitivity/Recall): 2%
## Balanced accuracy: 51%
## Overall accuracy: 21%
## Precision: 97%
## F1 score: 3%
```

```r
accuracy_nb_dataframe = nb_return$metrics_dataframe
accuracy_nb = nb_return$metrics
```

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

```
## [1] "XGBoost model prediction accuracy:"
## Class accuracy (Specificity): 99%
## Negative 1 accuracy (Sensitivity/Recall): 0%
## Balanced accuracy: 49%
## Overall accuracy: 98%
## Precision: 0%
## F1 score: NaN%
```

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

```r
gbm_return = GBM(train_data, test_data)
```

```
## [1] "GBM model prediction accuracy:"
## Class accuracy (Specificity): 99%
## Negative 1 accuracy (Sensitivity/Recall): 0%
## Balanced accuracy: 49%
## Overall accuracy: 98%
## Precision: 0%
## F1 score: NaN%
```

```r
accuracy_gbm_dataframe = gbm_return$metrics_gbm_dataframe
accuracy_gbm = gbm_return$metrics_gbm
```

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

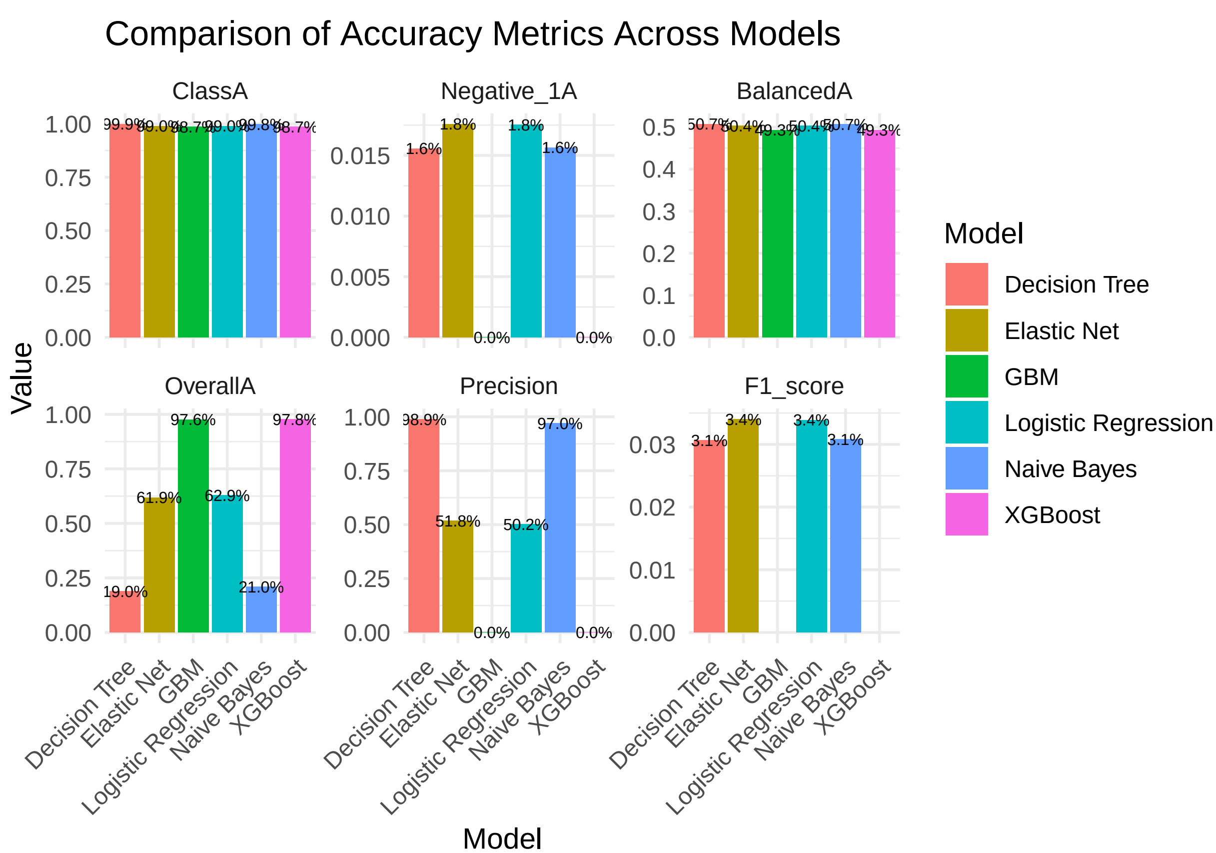
```
## [1] "Elastic Net model prediction accuracy:"
## Class accuracy (Specificity): 99%
## Negative 1 accuracy (Sensitivity/Recall): 2%
## Balanced accuracy: 50%
## Overall accuracy: 62%
## Precision: 52%
## F1 score: 3%
```

```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_gbm, "GBM"),
  list(accuracy_en, "Elastic Net")
)
accuracy_comparison_plot(metrics_list_BAL)
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_text()`).
```




```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_gbm_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 13: Classification Model Performance (continued below)

Model	Class_Accuracy	Negative_1_Accuracy	Balanced_Accuracy
Logistic Regression	99%	2%	50%
Decision Tree	100%	2%	51%
Naive Bayes	100%	2%	51%
XGBoost	99%	0%	49%
GBM	99%	0%	49%
Elastic Net	99%	2%	50%

Overall_Accuracy	Precision	F1_Score	Data_Combination
63%	50%	3%	borrow + account liquidated
19%	99%	3%	borrow + account liquidated
21%	97%	3%	borrow + account liquidated
98%	0%	NaN%	borrow + account liquidated
98%	0%	NaN%	borrow + account liquidated
62%	52%	3%	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.

```
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 15: Classification Model Performance for all data (continued below)

Model	Class_Accuracy	Negative_1_Accuracy	Balanced_Accuracy
Logistic Regression	99%	2%	50%
Decision Tree	100%	2%	51%
Naive Bayes	100%	2%	51%
XGBoost	99%	0%	49%
GBM	99%	0%	49%
Elastic Net	99%	2%	50%
Logistic Regression	93%	33%	63%
Decision Tree	83%	44%	64%
Naive Bayes	88%	44%	66%

Model	Class_Accuracy	Negative_1_Accuracy	Balanced_Accuracy
XGBoost	82%	41%	62%
GBM	83%	41%	62%
Elastic Net	94%	33%	63%
Logistic Regression	74%	73%	74%
Decision Tree	76%	66%	71%
Naive Bayes	64%	79%	72%
XGBoost	69%	59%	64%
GBM	64%	90%	77%
Elastic Net	74%	73%	73%
Logistic Regression	95%	28%	62%
Decision Tree	87%	40%	64%
Naive Bayes	92%	41%	66%
XGBoost	86%	12%	49%
GBM	89%	39%	64%
Elastic Net	95%	28%	62%

Overall_Accuracy	Precision	F1_Score	Data_Combination
63%	50%	3%	borrow + account liquidated
19%	99%	3%	borrow + account liquidated
21%	97%	3%	borrow + account liquidated
98%	0%	NaN%	borrow + account liquidated
98%	0%	NaN%	borrow + account liquidated
62%	52%	3%	borrow + account liquidated
67%	80%	47%	borrow + deposit
81%	7%	13%	borrow + deposit
79%	47%	45%	borrow + deposit
82%	1%	2%	borrow + deposit
80%	16%	23%	borrow + deposit
66%	80%	46%	borrow + deposit
74%	62%	67%	borrow + repay
71%	69%	68%	borrow + repay
67%	33%	47%	borrow + repay
65%	60%	59%	borrow + repay
68%	29%	44%	borrow + repay
74%	62%	67%	borrow + repay
69%	80%	41%	borrow + withdraw
86%	6%	11%	borrow + withdraw
83%	51%	45%	borrow + withdraw
86%	0%	1%	borrow + withdraw
84%	27%	32%	borrow + withdraw
68%	80%	41%	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).

```
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 17: Combined accuracy dataframe (continued below)

balanced_accuracy	Logistic.Regression	Decision.Tree
borrow + account liquidated	50.4	50.7
borrow + deposit	63.1	63.5
borrow + repay	73.6	70.9
borrow + withdraw	61.7	63.6

Naive.Bayes	XGBoost	GBM	Elastic.Net
50.7	49.3	49.3	50.4
66	61.5	62.1	63.1
71.9	64.1	77.4	73.4
66.2	49.3	64.2	61.6