# DeFi Survival Data Pipeline Example

Hanzhen Qin(qinh2)

22 February 2025

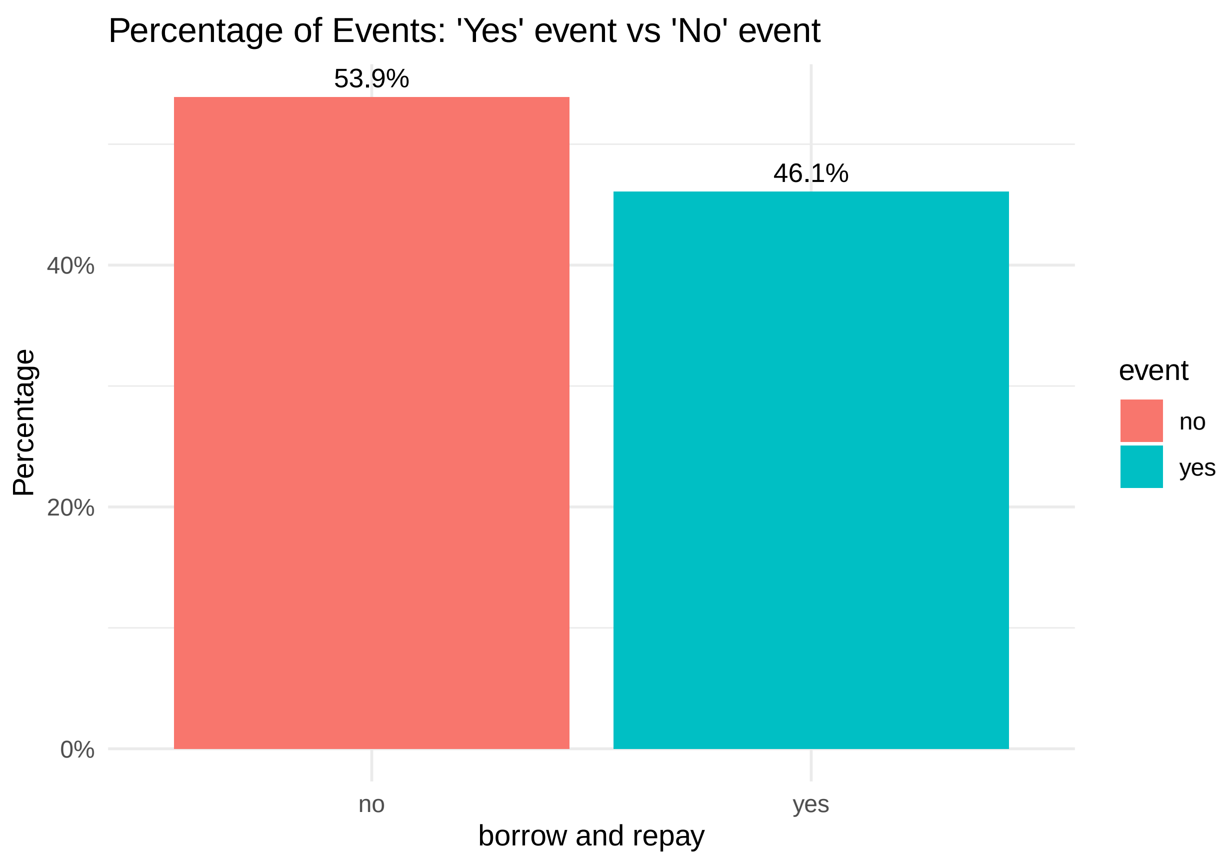
## Survival Data Pipeline

```
# 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 (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%
```

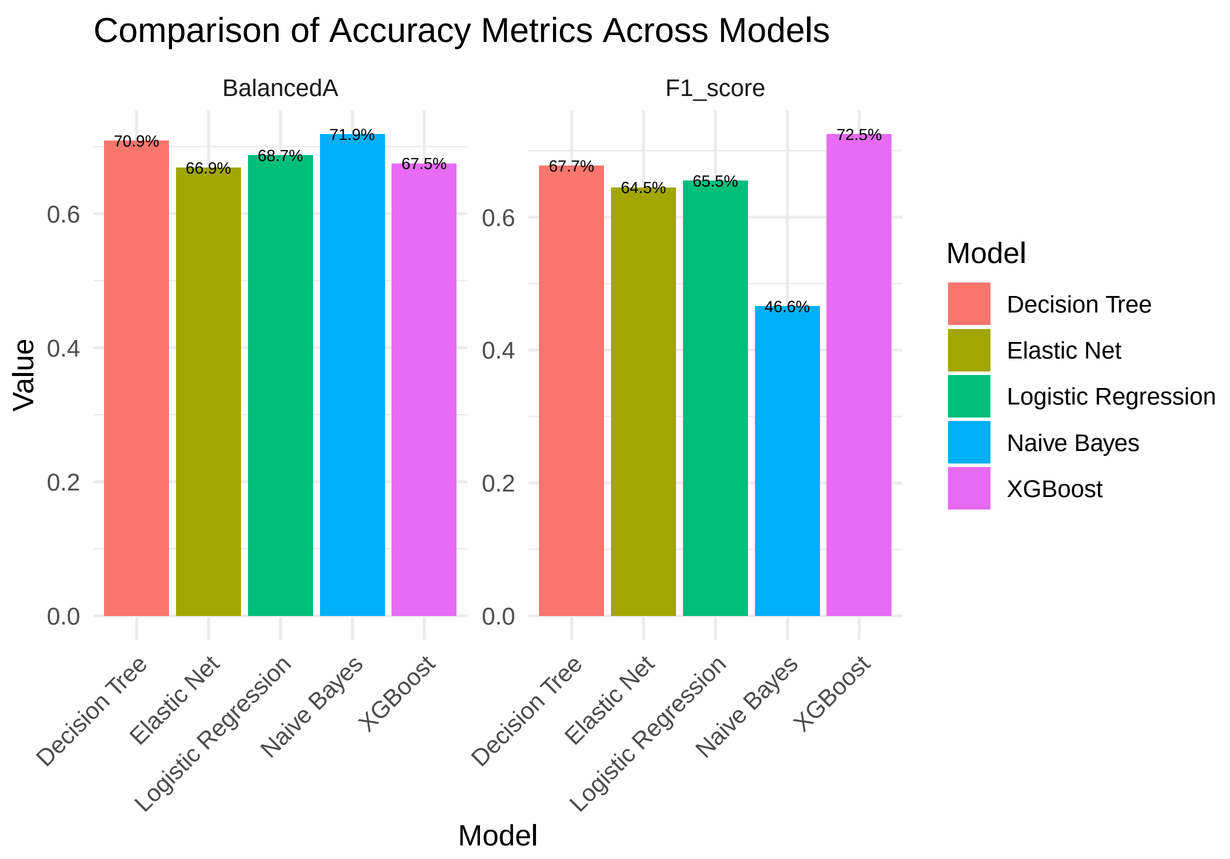
```
accuracy_xgb_dataframe = xgb_return$metrics_xgb_dataframe
accuracy_xgb = xgb_return$metrics_xgb
```

```
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%
```

```
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_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 1: 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 |

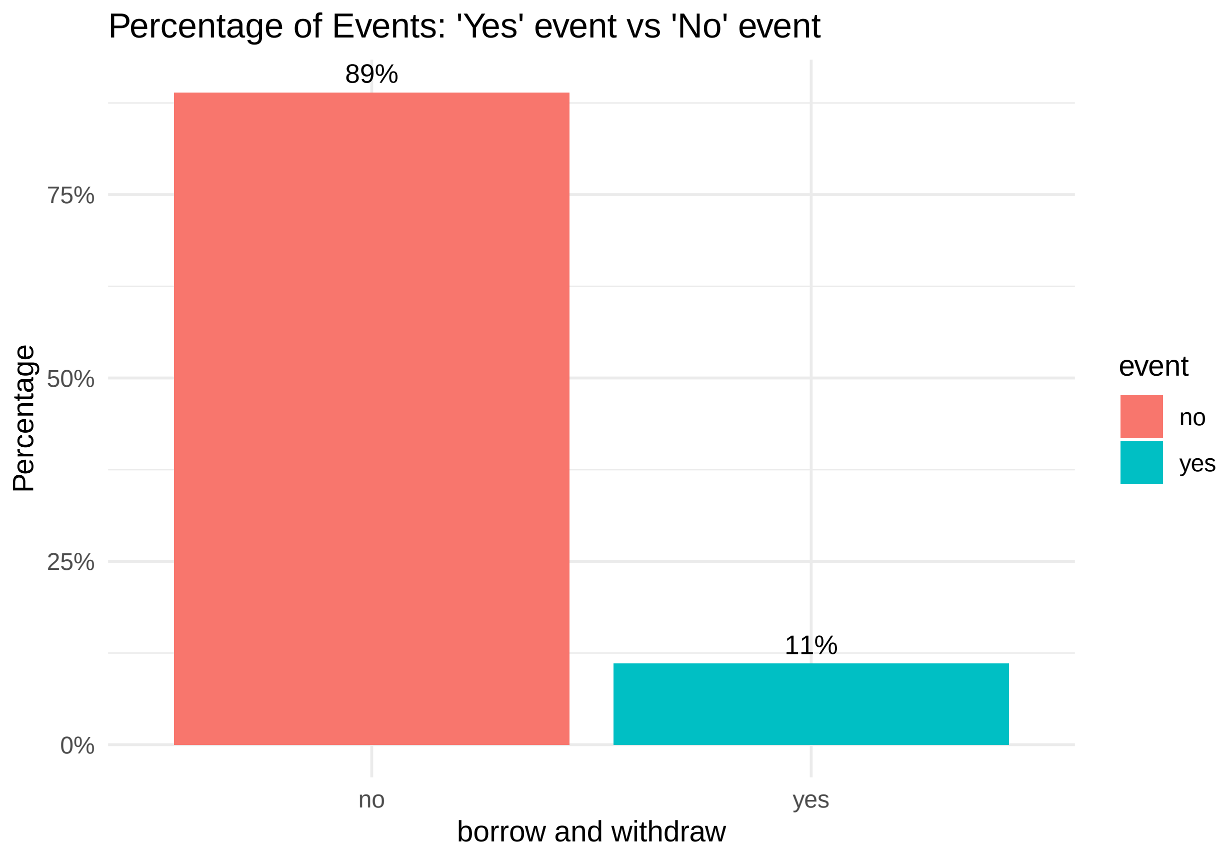
```r
# set the indexEvent and outcomeEvent
indexEvent = "borrow"
outcomeEvent = "withdraw"
```

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

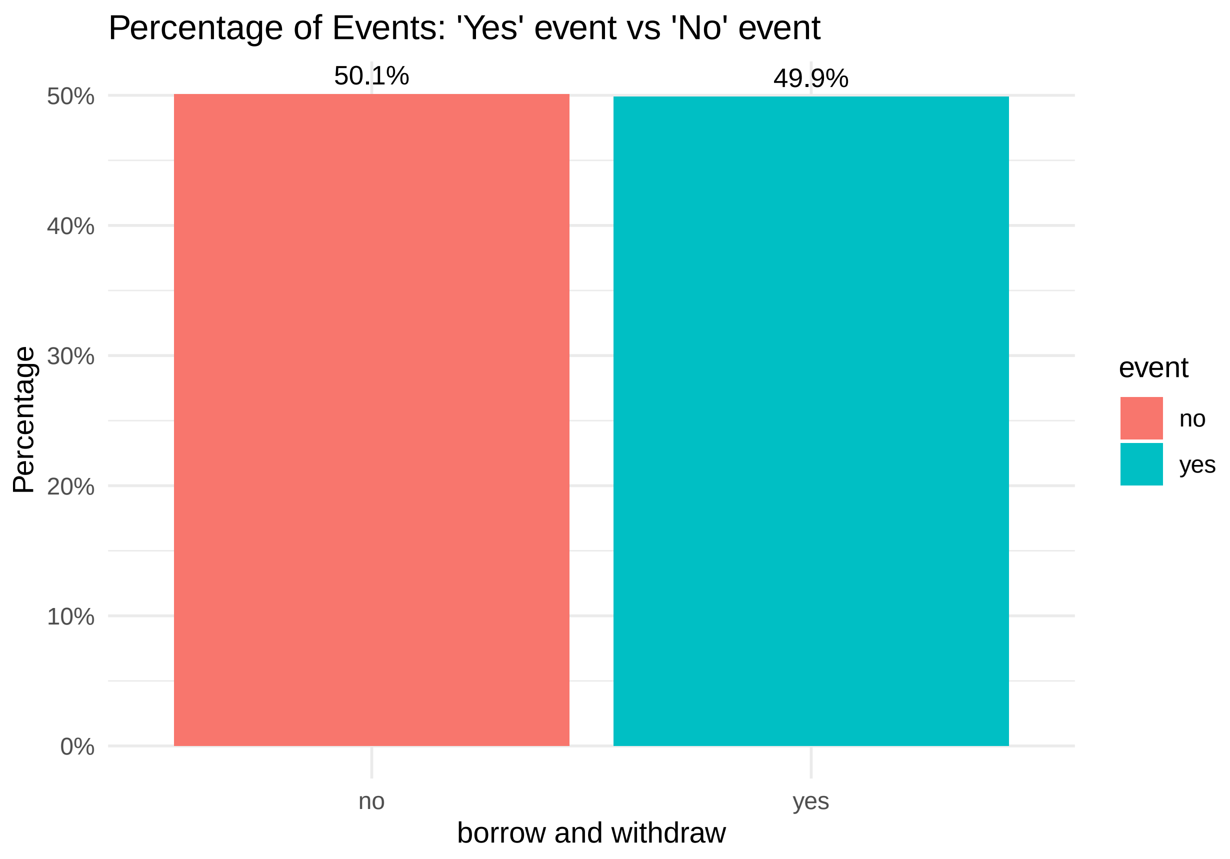
```
## Warning in inner_join(y, X, by = "id"): Detected an unexpected many-to-many relationship between `x`
## i Row 465 of `x` matches multiple rows in `y`.
## i Row 639 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to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

```
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: 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%
```

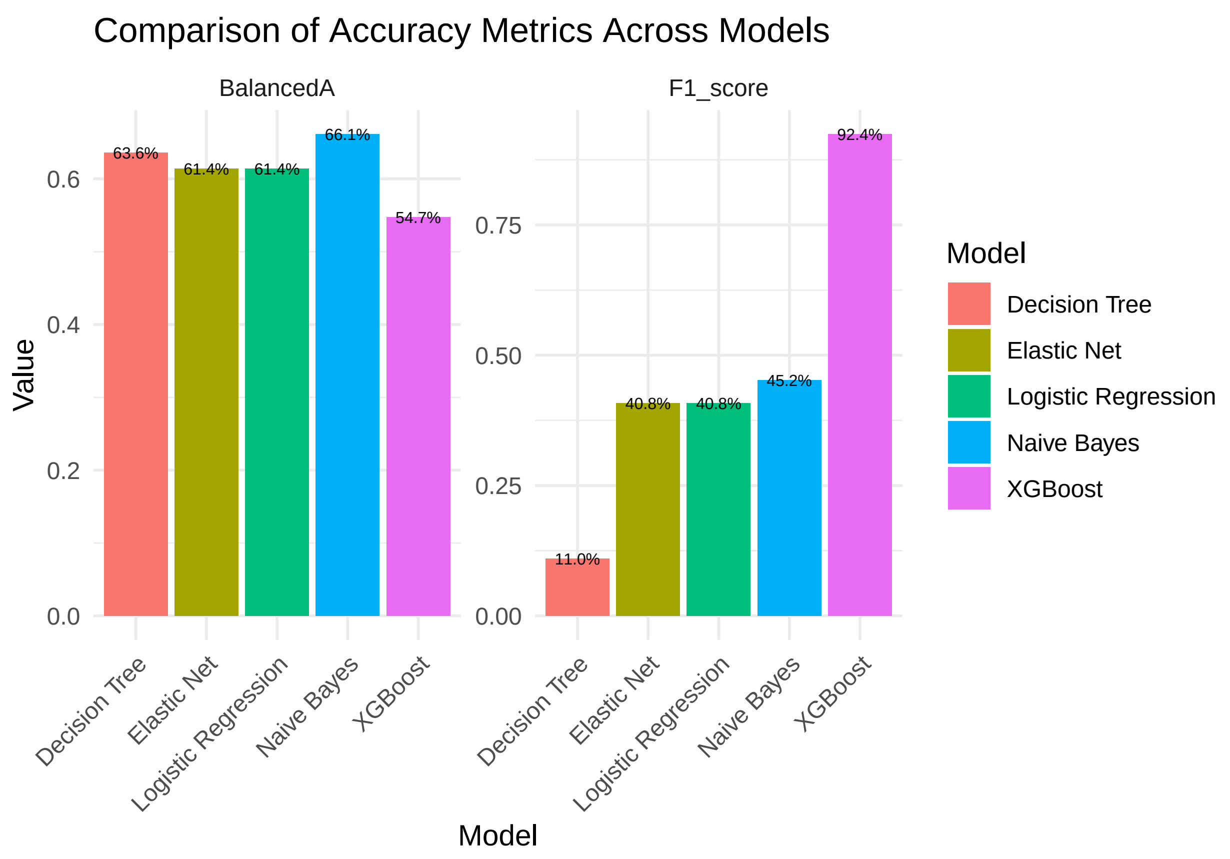
```r
accuracy_xgb_dataframe = xgb_return$metrics_xgb_dataframe
accuracy_xgb = xgb_return$metrics_xgb
```

```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 2: 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 |

## 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_BR, combined_results_BW))
pander(combined_classification_results, caption = "Classification Model Performance for all data")
```

Table 3: Classification Model Performance for all data

| 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 |
| 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).

```
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_BR, ba_accuracy_dataframe_BW))
pander(combined_accuracy_dataframe, caption = "Combined accuracy dataframe")
```

Table 4: Combined accuracy dataframe (continued below)

| balanced_accuracy | Logistic.Regression | Decision.Tree | Naive.Bayes | XGBoost | Elastic.Net |
|---|---|---|---|---|---|
| borrow + repay | 68.7 | 70.9 | 71.9 | 67.5 | 66.9 |
| borrow + withdraw | 61.4 | 63.6 | 66.1 | 54.7 | 61.4 |