DMLR DeFi Survival Data Pipeline Example(qinh2):

Hanzhen Qin(qinh2)

20 January 2025

Comprehensive Data Combination Classification and Prediction - Simple Example

In this section, I will apply all classification models to evaluate all data combinations, aiming to achieve comprehensive results. The outcomes will be visualized using bar graphs, and a detailed final dataframe will be generated to record all the results systematically.

```
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_v
```

```
##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor
```

```
source("~/DMLR_DeFi_Survival_Dataset_And_Benchmark/DeFi_source/survivalData_pipeline/classification_mod
source("~/DMLR_DeFi_Survival_Dataset_And_Benchmark/DeFi_source/survivalData_pipeline/get_classification_
```

```
# 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.
```

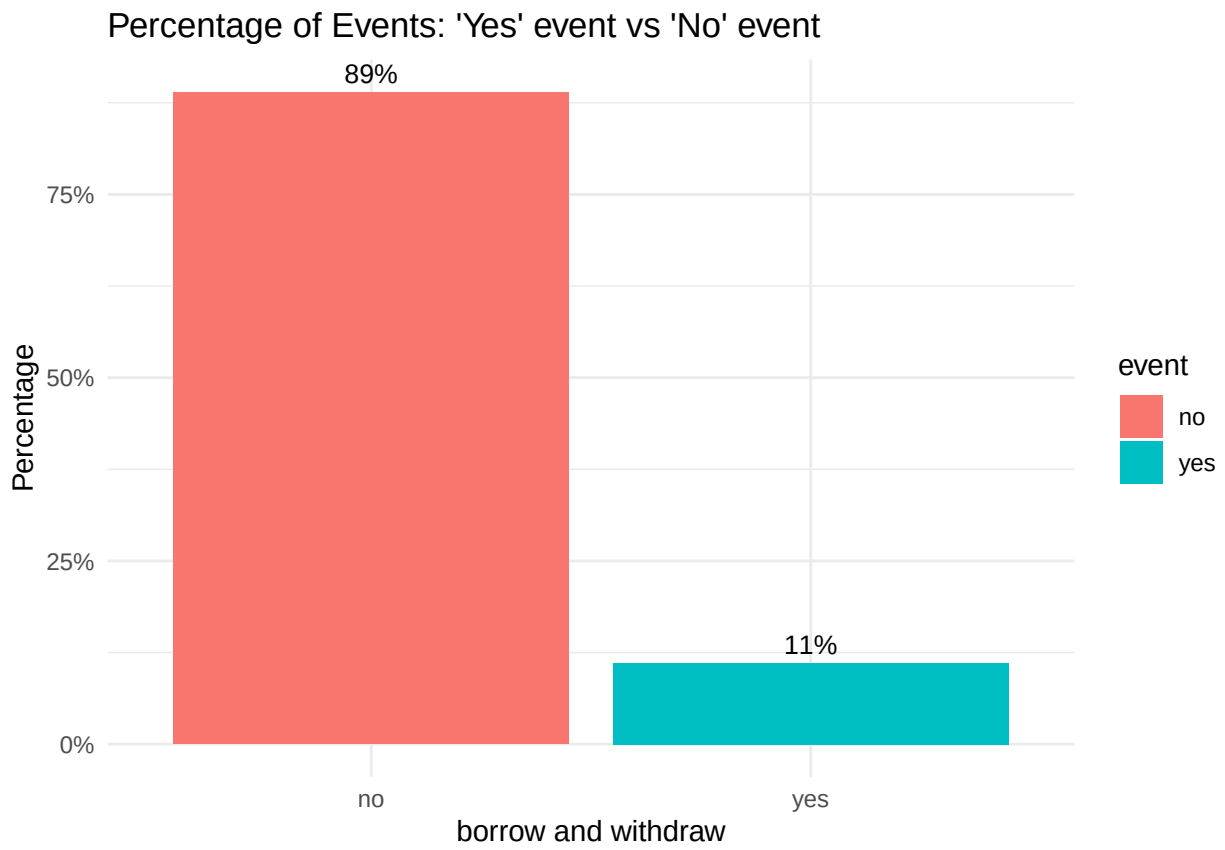
```
## Warning in inner_join(y, X, by = "id"): Detected an unexpected many-to-many relationship between `x`
## i Row 285 of `x` matches multiple rows in `y`.
## i Row 338 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to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

```
# 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)
```

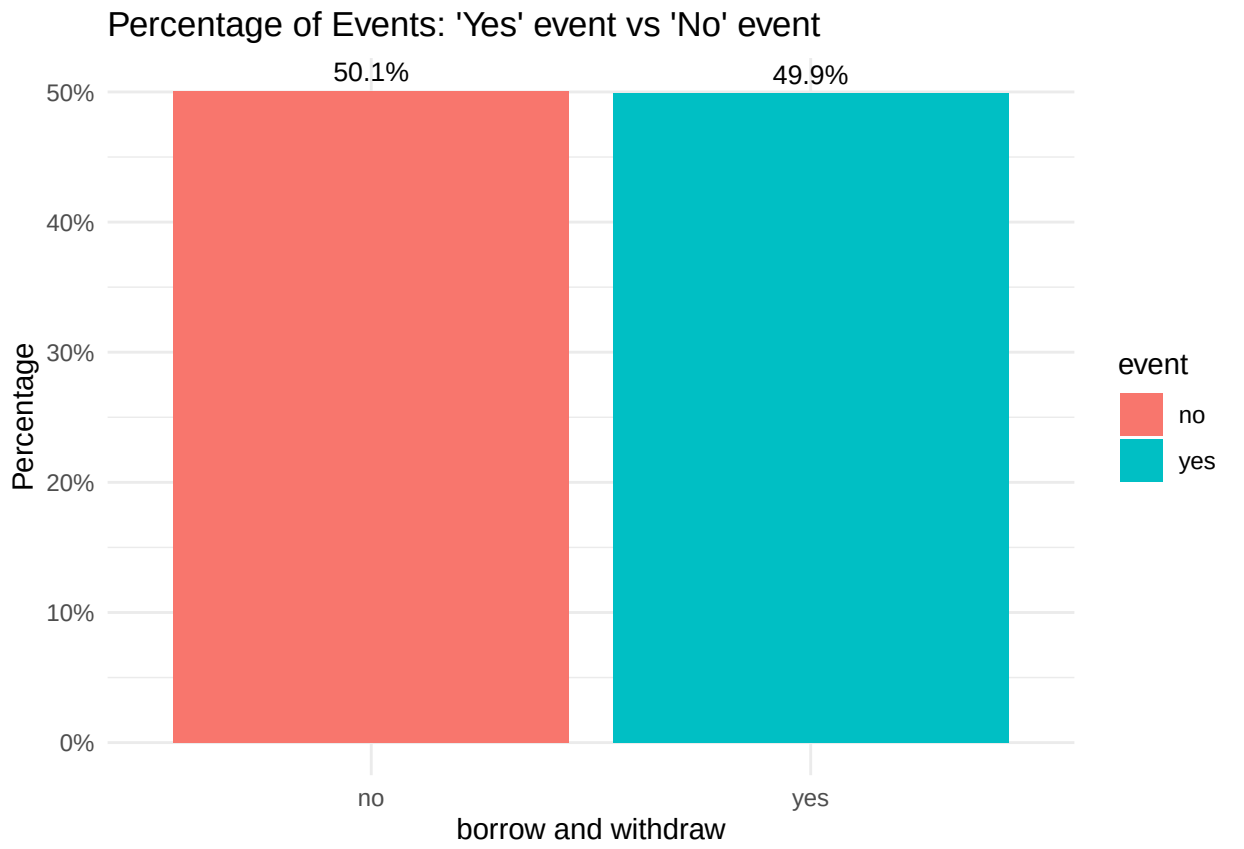


If the ratio of "No" labels to "Yes" labels in the dataset is significantly imbalanced, we can utilize the `smote_data` function to generate a new, more balanced dataset. This balanced dataset ensures that both classes are better represented, helping to mitigate the bias introduced by class imbalance 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 start to analyze the data combination with `indexEvent = "Deposit"` and `outcomeEvent = "Withdraw"`. 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): 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
pander(accuracy_lr_dataframe, caption = "Logistic Regression Performance")
```

Table 1: Logistic Regression Performance (continued below)

Model	Class_Accuracy	Negative_1_Accuracy	Balanced_Accuracy
Logistic Regression	95%	28%	62%

Overall_Accuracy	Precision	F1_Score
69%	80%	41%

```
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%
```

```
accuracy_dt_dataframe = dt_return$metrics_dt_dataframe
accuracy_dt = dt_return$metrics_dt
pander(accuracy_dt_dataframe, caption = "Decision Tree Performance")
```

Table 3: Decision Tree Performance (continued below)

Model	Class_Accuracy	Negative_1_Accuracy	Balanced_Accuracy
Decision Tree	87%	40%	64%

Overall_Accuracy	Precision	F1_Score
86%	6%	11%

```
# compare all the classification models
metrics_list_BW <- list(
  list(accuracy_lr, "Logistic Regression"),
  list(accuracy_dt, "Decision Tree")
)
accuracy_comparison_plot(metrics_list_BW)
```


```
# Show the final dataframe for all four types of 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)
combined_results_BW <- combine_classification_results(accuracy_dataframe_list_BW, data_name_BW)

# display the combined dataframe
# print(combined_results)
pander(combined_results_BW, caption = "Classification Model Performance for borrow + withdraw")
```

Table 5: Classification Model Performance for borrow + withdraw (continued below)

Model	Class_Accuracy	Negative_1_Accuracy	Balanced_Accuracy
Logistic Regression	95%	28%	62%
Decision Tree	87%	40%	64%

Overall_Accuracy	Precision	F1_Score	Data_Combination
69%	80%	41%	borrow + withdraw
86%	6%	11%	borrow + withdraw

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

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

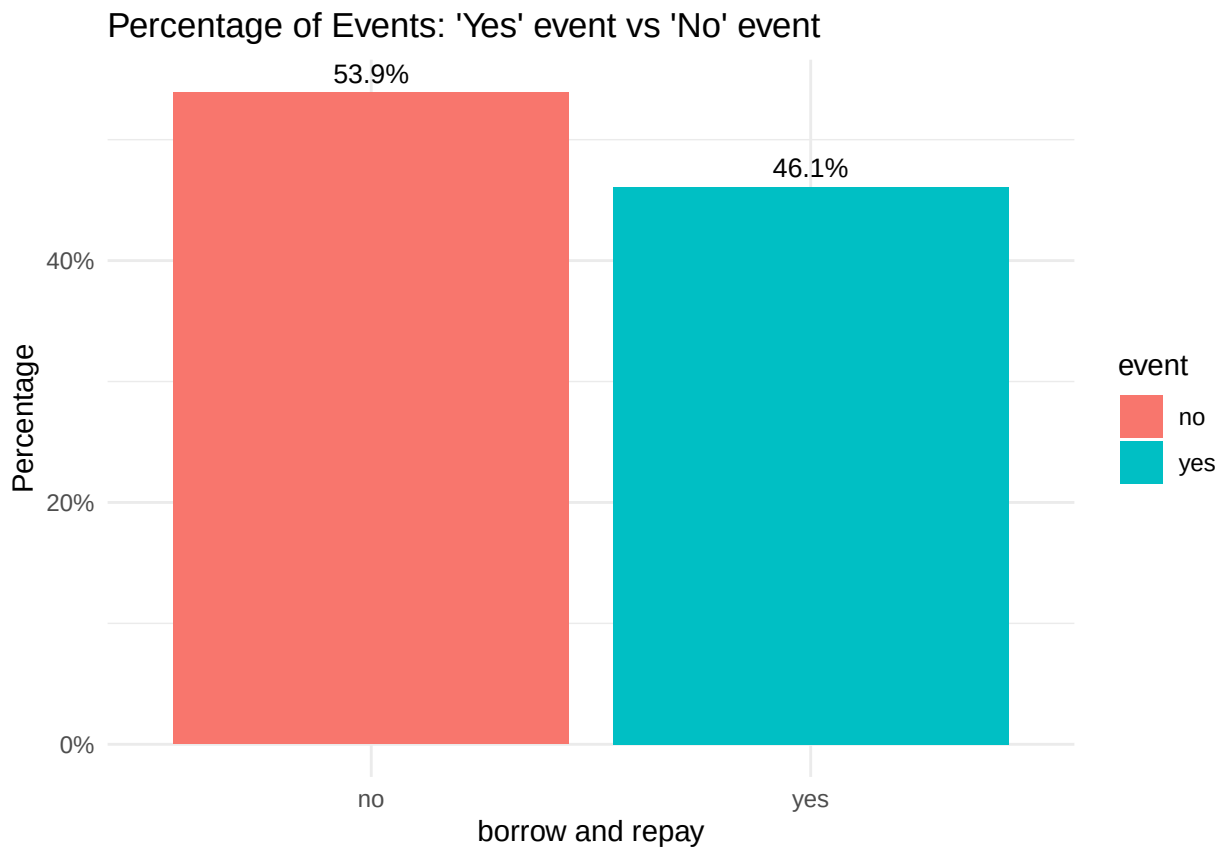
```
## Warning in inner_join(y, X, by = "id"): Detected an unexpected many-to-many relationship between `x`
## i Row 56 of `x` matches multiple rows in `y`.
## i Row 10453 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to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

```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)
### Get The Specified Accuracy dataframe

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 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
pander(accuracy_lr_dataframe, caption = "Logistic Regression Performance")
```

Table 7: Logistic Regression Performance (continued below)

Model	Class_Accuracy	Negative_1_Accuracy	Balanced_Accuracy
Logistic Regression	74%	73%	74%

Overall_Accuracy	Precision	F1_Score
74%	62%	67%

```
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
pander(accuracy_dt_dataframe, caption = "Decision Tree Performance")
```

Table 9: Decision Tree Performance (continued below)

Model	Class_Accuracy	Negative_1_Accuracy	Balanced_Accuracy
Decision Tree	76%	66%	71%

Overall_Accuracy	Precision	F1_Score
71%	69%	68%

```
# compare all the classification models
metrics_list_BR <- list(
  list(accuracy_lr, "Logistic Regression"),
  list(accuracy_dt, "Decision Tree")
)
accuracy_comparison_plot(metrics_list_BR)
```


```
# Show the final dataframe for all four types of 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)
combined_results_BR <- combine_classification_results(accuracy_dataframe_list_BR, data_name_BR)

# display the combined dataframe
# print(combined_results)
pander(combined_results_BR, caption = "Classification Model Performance for borrow + repay")
```

Table 11: Classification Model Performance for borrow + repay (continued below)

Model	Class_Accuracy	Negative_1_Accuracy	Balanced_Accuracy
Logistic Regression	74%	73%	74%
Decision Tree	76%	66%	71%

Overall_Accuracy	Precision	F1_Score	Data_Combination
74%	62%	67%	borrow + repay
71%	69%	68%	borrow + repay

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_BW, combined_results_BR))
pander(combined_classification_results, caption = "Classification Model Performance for all data")
```

Table 13: Classification Model Performance for all data (continued below)

Model	Class_Accuracy	Negative_1_Accuracy	Balanced_Accuracy
Logistic Regression	95%	28%	62%
Decision Tree	87%	40%	64%
Logistic Regression	74%	73%	74%
Decision Tree	76%	66%	71%

Overall_Accuracy	Precision	F1_Score	Data_Combination
69%	80%	41%	borrow + withdraw
86%	6%	11%	borrow + withdraw
74%	62%	67%	borrow + repay
71%	69%	68%	borrow + repay

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_BW <- specific_accuracy_statistics(data_name_BW, "balanced_accuracy",
                                                         metrics_list_BW)
ba_accuracy_dataframe_BR <- specific_accuracy_statistics(data_name_BR, "balanced_accuracy",
                                                         metrics_list_BR)
combined_accuracy_dataframe <- combine_accuracy_dataframes(
  list(ba_accuracy_dataframe_BW, ba_accuracy_dataframe_BR))
pander(combined_accuracy_dataframe, caption = "Combined accuracy dataframe")
```

Table 15: Combined accuracy dataframe

balanced_accuracy	Logistic.Regression	Decision.Tree
borrow + withdraw	61.7	63.6
borrow + repay	73.6	70.9