PLTR Robustness Check

ZAINAB BELGADA* DAMIEN ZINSOU†

15/01/2021

Introduction

In this notebook, we compute the average value of each evaluation metric across the 5 x 2 cross-validation test samples. We compare the performance of PLTR to that of:

- Random Forest
- SVM with both radial and polynomial kernel
- Linear Logistic regression without and with regularization (RIDGE, LASSO, and Adaptive LASSO)
- Penalized Non-Linear Logistic Regression which include as additional variables quadratic and interaction terms (RIDGE, LASSO, and Adaptive LASSO)

Load and Prepare the Housing Dataset

We will begin by loading some helper functions that are implemented in the `scripts` folder before reading the raw housing dataset:

```
source("./../../scripts/prepare_housing_dataset.R")
source("./../../scripts/add_interaction_quadratic_terms.R")

# Load The Raw Dataset
raw_housing_dataset <- readxl::read_excel("./../../data/hmeq.xls", sheet = 'hmeq')
```

Next, we will prepare 3 versions of the dataset:

- `clean_housing_dataset`: Missing value are imputed. Categorical predictors are one-hot encoded.
- `clean_housing_dataset_factor` : Will be used only for in Non-Penalized Linear Logistic Regression. Instead of one-hot encoding, categorical predictors are represented as R's factor datatype in order to avoid multicollinearity.
- `clean_with_interaction_quadratic`: Will be used for Non-Linear Logistic Regression. As the name suggests, this dataset include as additional variables quadratic and interaction terms.

```
clean_housing_dataset <- prepare_housing_dataset(raw_housing_dataset)

clean_housing_dataset_factor <- prepare_housing_dataset(raw_housing_dataset,
                                                        to_dummy = FALSE)

clean_with_interaction_quadratic <- add_interaction_quadratic_terms(clean_housing_dataset)
```

*Institut d'Economie d'Orleans, zainab.belgada@etu.univ-orleans.fr
†Institut d'Economie d'Orleans, zinsou.mezonlin@etu.univ-orleans.fr

Data Partitioning for Cross Validation

Following Dumitrescu et al. (2020), we use the so called N x 2-fold cross-validation of Dietterich (1998), which involves randomly dividing the dataset into two sub-samples of equal size:

- The first (second) part is used to build the model;
- The second (first) part is used for evaluation.

This procedure is repeated N times, and the evaluation metrics are averaged. We set N = 5 for computational reasons.

```
source("./../../scripts/partition_data.R")

# For:
# PLTR, RANDOM FOREST, SVM, PENALIZED LINEAR LOGISTIC REGRESSION
data_partitions_ml <- partition_data(clean_housing_dataset, N = 5, random_seed = 8080)

# For LINEAR LOGISTIC REGRESSION
data_partitions_llr <- partition_data(clean_housing_dataset_factor, N = 5,
                                      random_seed = 8080)

# For PENALIZED NON-LINEAR LOGISTIC REGRESSION
data_partitions_nllr <- partition_data(clean_with_interaction_quadratic, N = 5,
                                       random_seed = 8080)
```

Cross Validation Results

Loading some helper functions:

```
source("./../../scripts/compute_evaluation_criteria.R")
source("./../../scripts/cross_validate.R")
source("./../../scripts/rules_utilities.R")
source("./../../scripts/pltr_learner.R")
source("./../../scripts/random_forest_learner.R")
source("./../../scripts/svm_learner.R")
source("./../../scripts/logistic_learner.R")
source("./../../scripts/penalized_learner.R")
```

1. PLTR - Adaptive LASSO

In order to save some computation time, we will first generate predictors pairs upfront:

```
predictors_set <- clean_housing_dataset %>%
   names() %>%
   tail(n = -1)

predictors_pairs <- predictors_set %>%
   combn(m = 2) %>%
   purrr::array_branch(margin = 2)
```

Running 5 x 2-fold cross validation:

```
pltr_alasso_results <- cross_validate(
   cv_partitions = data_partitions_ml,
```

```
   learner = pltr_learner,
   evaluator = compute_evaluation_criteria,
   shinyProgress = FALSE,
   predictors_pairs = predictors_pairs,
   penalty = 2)
```

```
## Processing N = 1: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 1, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 2: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 2, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 3: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 3, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 4: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 4, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 5: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 5, But Now Training on Fold 2 and Testing on Fold 1
```

```
pltr_alasso_avg_results <- pltr_alasso_results %>%
   dplyr::select(AUC, GINI, PCC, BS, KS) %>%
   colMeans() %>%
   dplyr::bind_rows() %>%
   dplyr::mutate(LEARNING_ALGORITHM = "PLTR (ADAPTIVE LASSO)") %>%
   dplyr::relocate(LEARNING_ALGORITHM, .before = AUC)

pltr_alasso_avg_results
```

```
## # A tibble: 1 x 6
##   LEARNING_ALGORITHM      AUC  GINI   PCC     BS    KS
##   <chr>                 <dbl> <dbl> <dbl>  <dbl> <dbl>
## 1 PLTR (ADAPTIVE LASSO) 0.907 0.815 0.889 0.0826 0.687
```

2. PLTR - LASSO

```
pltr_lasso_results <- cross_validate(
   cv_partitions = data_partitions_ml,
   learner = pltr_learner,
   evaluator = compute_evaluation_criteria,
   shinyProgress = FALSE,
   predictors_pairs = predictors_pairs,
   penalty = 1)
```

```
## Processing N = 1: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 1, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 2: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 2, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 3: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 3, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 4: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 4, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 5: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 5, But Now Training on Fold 2 and Testing on Fold 1
```

```
pltr_lasso_avg_results <- pltr_lasso_results %>%
   dplyr::select(AUC, GINI, PCC, BS, KS) %>%
```

```
   colMeans() %>%
   dplyr::bind_rows() %>%
   dplyr::mutate(LEARNING_ALGORITHM = "PLTR (LASSO)") %>%
   dplyr::relocate(LEARNING_ALGORITHM, .before = AUC)

pltr_lasso_avg_results
```

```
## # A tibble: 1 x 6
##   LEARNING_ALGORITHM   AUC  GINI   PCC     BS    KS
##   <chr>              <dbl> <dbl> <dbl>  <dbl> <dbl>
## 1 PLTR (LASSO)       0.911 0.821 0.891 0.0810 0.689
```

3. PLTR - RIDGE

```
pltr_ridge_results <- cross_validate(
   cv_partitions = data_partitions_ml,
   learner = pltr_learner,
   evaluator = compute_evaluation_criteria,
   shinyProgress = FALSE,
   predictors_pairs = predictors_pairs,
   penalty = 0)
```

```
## Processing N = 1: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 1, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 2: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 2, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 3: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 3, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 4: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 4, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 5: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 5, But Now Training on Fold 2 and Testing on Fold 1
```

```
pltr_ridge_avg_results <- pltr_ridge_results %>%
   dplyr::select(AUC, GINI, PCC, BS, KS) %>%
   colMeans() %>%
   dplyr::bind_rows() %>%
   dplyr::mutate(LEARNING_ALGORITHM = "PLTR (RIDGE)") %>%
   dplyr::relocate(LEARNING_ALGORITHM, .before = AUC)

pltr_ridge_avg_results
```

```
## # A tibble: 1 x 6
##   LEARNING_ALGORITHM   AUC  GINI   PCC     BS    KS
##   <chr>              <dbl> <dbl> <dbl>  <dbl> <dbl>
## 1 PLTR (RIDGE)       0.911 0.823 0.890 0.0817 0.689
```

4. Random Forest

```
rf_results <- cross_validate(
   cv_partitions = data_partitions_ml,
   learner = random_forest_learner,
```

```
   evaluator = compute_evaluation_criteria,
   random_seed = 8081,
   shinyProgress = FALSE)
```

```
## Processing N = 1: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 1, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 2: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 2, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 3: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 3, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 4: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 4, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 5: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 5, But Now Training on Fold 2 and Testing on Fold 1
```

```
rf_avg_results <- rf_results %>%
   dplyr::select(AUC, GINI, PCC, BS, KS) %>%
   colMeans() %>%
   dplyr::bind_rows() %>%
   dplyr::mutate(LEARNING_ALGORITHM = "RANDOM FOREST") %>%
   dplyr::relocate(LEARNING_ALGORITHM, .before = AUC)

rf_avg_results
```

```
## # A tibble: 1 x 6
##   LEARNING_ALGORITHM   AUC  GINI   PCC     BS    KS
##   <chr>              <dbl> <dbl> <dbl>  <dbl> <dbl>
## 1 RANDOM FOREST      0.955 0.910 0.912 0.0668 0.785
```

5. SVM: Radial Kernel

```
svm_radial_results <- cross_validate(
   cv_partitions = data_partitions_ml,
   learner = svm_learner,
   evaluator = compute_evaluation_criteria,
   kernel = "radial",
   shinyProgress = FALSE)
```

```
## Processing N = 1: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 1, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 2: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 2, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 3: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 3, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 4: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 4, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 5: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 5, But Now Training on Fold 2 and Testing on Fold 1
```

```
svm_radial_avg_results <- svm_radial_results %>%
   dplyr::select(AUC, GINI, PCC, BS, KS) %>%
   colMeans() %>%
   dplyr::bind_rows() %>%
   dplyr::mutate(LEARNING_ALGORITHM = "SVM (RADIAL)") %>%
```

```
   dplyr::relocate(LEARNING_ALGORITHM, .before = AUC)

svm_radial_avg_results
```

```
## # A tibble: 1 x 6
##   LEARNING_ALGORITHM   AUC  GINI   PCC     BS    KS
##   <chr>              <dbl> <dbl> <dbl>  <dbl> <dbl>
## 1 SVM (RADIAL)       0.823 0.646 0.812 0.0918 0.597
```

6. SVM: Polynomial Kernel

```
svm_polynomial_results <- cross_validate(
   cv_partitions = data_partitions_ml,
   learner = svm_learner,
   evaluator = compute_evaluation_criteria,
   kernel = "polynomial",
   shinyProgress = FALSE)
```

```
## Processing N = 1: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 1, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 2: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 2, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 3: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 3, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 4: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 4, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 5: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 5, But Now Training on Fold 2 and Testing on Fold 1
```

```
svm_polynomial_avg_results <- svm_polynomial_results %>%
   dplyr::select(AUC, GINI, PCC, BS, KS) %>%
   colMeans() %>%
   dplyr::bind_rows() %>%
   dplyr::mutate(LEARNING_ALGORITHM = "SVM (POLYNOMIAL)") %>%
   dplyr::relocate(LEARNING_ALGORITHM, .before = AUC)

svm_polynomial_avg_results
```

```
## # A tibble: 1 x 6
##   LEARNING_ALGORITHM   AUC  GINI   PCC    BS    KS
##   <chr>              <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 SVM (POLYNOMIAL)   0.783 0.566 0.839 0.111 0.499
```

7. Linear Logistic Regression

```
llr_results <- cross_validate(
   cv_partitions = data_partitions_llr,
   learner = logistic_learner,
   evaluator = compute_evaluation_criteria,
   shinyProgress = FALSE)
```

```
## Processing N = 1: Training on Fold 1 and Testing on Fold 2
```

```
## Still Processing N = 1, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 2: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 2, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 3: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 3, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 4: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 4, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 5: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 5, But Now Training on Fold 2 and Testing on Fold 1
```

```
llr_avg_results <- llr_results %>%
   dplyr::select(AUC, GINI, PCC, BS, KS) %>%
   colMeans() %>%
   dplyr::bind_rows() %>%
   dplyr::mutate(LEARNING_ALGORITHM = "LINEAR LR") %>%
   dplyr::relocate(LEARNING_ALGORITHM, .before = AUC)

llr_avg_results
```

```
## # A tibble: 1 x 6
##   LEARNING_ALGORITHM   AUC  GINI   PCC    BS    KS
##   <chr>              <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 LINEAR LR          0.792 0.583 0.836 0.123 0.447
```

8. Penalized Linear Logistic Regression: RIDGE

```
llr_ridge_results <- cross_validate(
   cv_partitions = data_partitions_ml,
   learner = penalized_learner,
   evaluator = compute_evaluation_criteria,
   penalty = 0,
   shinyProgress = FALSE)
```

```
## Processing N = 1: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 1, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 2: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 2, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 3: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 3, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 4: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 4, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 5: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 5, But Now Training on Fold 2 and Testing on Fold 1
```

```
llr_ridge_avg_results <- llr_ridge_results %>%
   dplyr::select(AUC, GINI, PCC, BS, KS) %>%
   colMeans() %>%
   dplyr::bind_rows() %>%
   dplyr::mutate(LEARNING_ALGORITHM = "LINEAR LR (RIDGE)") %>%
   dplyr::relocate(LEARNING_ALGORITHM, .before = AUC)

llr_ridge_avg_results
```

```
## # A tibble: 1 x 6
```

```
##   LEARNING_ALGORITHM   AUC  GINI   PCC    BS    KS
##   <chr>              <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 LINEAR LR (RIDGE)  0.792 0.583 0.834 0.123 0.450
```

9. Penalized Linear Logistic Regression: LASSO

```
llr_lasso_results <- cross_validate(
   cv_partitions = data_partitions_ml,
   learner = penalized_learner,
   evaluator = compute_evaluation_criteria,
   penalty = 1,
   shinyProgress = FALSE)
```

```
## Processing N = 1: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 1, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 2: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 2, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 3: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 3, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 4: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 4, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 5: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 5, But Now Training on Fold 2 and Testing on Fold 1
```

```
llr_lasso_avg_results <- llr_lasso_results %>%
   dplyr::select(AUC, GINI, PCC, BS, KS) %>%
   colMeans() %>%
   dplyr::bind_rows() %>%
   dplyr::mutate(LEARNING_ALGORITHM = "LINEAR LR (LASSO)") %>%
   dplyr::relocate(LEARNING_ALGORITHM, .before = AUC)

llr_lasso_avg_results
```

```
## # A tibble: 1 x 6
##   LEARNING_ALGORITHM   AUC  GINI   PCC    BS    KS
##   <chr>              <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 LINEAR LR (LASSO)  0.791 0.583 0.836 0.123 0.443
```

10. Penalized Linear Logistic Regression: Adaptive LASSO

```
llr_alasso_results <- cross_validate(
   cv_partitions = data_partitions_ml,
   learner = penalized_learner,
   evaluator = compute_evaluation_criteria,
   penalty = 2,
   shinyProgress = FALSE)
```

```
## Processing N = 1: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 1, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 2: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 2, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 3: Training on Fold 1 and Testing on Fold 2
```

```
## Still Processing N = 3, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 4: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 4, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 5: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 5, But Now Training on Fold 2 and Testing on Fold 1
```

```
llr_alasso_avg_results <- llr_alasso_results %>%
   dplyr::select(AUC, GINI, PCC, BS, KS) %>%
   colMeans() %>%
   dplyr::bind_rows() %>%
   dplyr::mutate(LEARNING_ALGORITHM = "LINEAR LR (ADAPTIVE LASSO)") %>%
   dplyr::relocate(LEARNING_ALGORITHM, .before = AUC)

llr_alasso_avg_results
```

```
## # A tibble: 1 x 6
##   LEARNING_ALGORITHM           AUC  GINI   PCC    BS    KS
##   <chr>                      <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 LINEAR LR (ADAPTIVE LASSO) 0.786 0.573 0.834 0.124 0.425
```

11. Penalized Non-Linear Logistic Regression: RIDGE

```
nllr_ridge_results <- cross_validate(
   cv_partitions = data_partitions_nllr,
   learner = penalized_learner,
   evaluator = compute_evaluation_criteria,
   penalty = 0,
   shinyProgress = FALSE)
```

```
## Processing N = 1: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 1, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 2: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 2, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 3: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 3, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 4: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 4, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 5: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 5, But Now Training on Fold 2 and Testing on Fold 1
```

```
nllr_ridge_avg_results <- nllr_ridge_results %>%
   dplyr::select(AUC, GINI, PCC, BS, KS) %>%
   colMeans() %>%
   dplyr::bind_rows() %>%
   dplyr::mutate(LEARNING_ALGORITHM = "NON-LINEAR LR (RIDGE)") %>%
   dplyr::relocate(LEARNING_ALGORITHM, .before = AUC)

nllr_ridge_avg_results
```

```
## # A tibble: 1 x 6
##   LEARNING_ALGORITHM      AUC  GINI   PCC    BS    KS
##   <chr>                 <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 NON-LINEAR LR (RIDGE) 0.819 0.638 0.852 0.112 0.494
```

12. Penalized Non-Linear Logistic Regression: LASSO

```
nllr_lasso_results <- cross_validate(
   cv_partitions = data_partitions_nllr,
   learner = penalized_learner,
   evaluator = compute_evaluation_criteria,
   penalty = 1,
   shinyProgress = FALSE)
```

```
## Processing N = 1: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 1, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 2: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 2, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 3: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 3, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 4: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 4, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 5: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 5, But Now Training on Fold 2 and Testing on Fold 1
```

```
nllr_lasso_avg_results <- nllr_lasso_results %>%
   dplyr::select(AUC, GINI, PCC, BS, KS) %>%
   colMeans() %>%
   dplyr::bind_rows() %>%
   dplyr::mutate(LEARNING_ALGORITHM = "NON-LINEAR LR (LASSO)") %>%
   dplyr::relocate(LEARNING_ALGORITHM, .before = AUC)

nllr_lasso_avg_results
```

```
## # A tibble: 1 x 6
##   LEARNING_ALGORITHM      AUC  GINI   PCC    BS    KS
##   <chr>                 <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 NON-LINEAR LR (LASSO) 0.821 0.642 0.857 0.110 0.498
```

13. Penalized Non-Linear Logistic Regression: Adaptive LASSO

```
nllr_alasso_results <- cross_validate(
   cv_partitions = data_partitions_nllr,
   learner = penalized_learner,
   evaluator = compute_evaluation_criteria,
   penalty = 2,
   shinyProgress = FALSE)
```

```
## Processing N = 1: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 1, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 2: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 2, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 3: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 3, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 4: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 4, But Now Training on Fold 2 and Testing on Fold 1
## Processing N = 5: Training on Fold 1 and Testing on Fold 2
## Still Processing N = 5, But Now Training on Fold 2 and Testing on Fold 1
```

```r
nllr_alasso_avg_results <- nllr_alasso_results %>%
   dplyr::select(AUC, GINI, PCC, BS, KS) %>%
   colMeans() %>%
   dplyr::bind_rows() %>%
   dplyr::mutate(LEARNING_ALGORITHM = "NON-LINEAR LR (ADAPTIVE LASSO)") %>%
   dplyr::relocate(LEARNING_ALGORITHM, .before = AUC)

nllr_alasso_avg_results
```

```
## # A tibble: 1 x 6
##   LEARNING_ALGORITHM               AUC  GINI   PCC    BS    KS
##   <chr>                          <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 NON-LINEAR LR (ADAPTIVE LASSO) 0.789 0.579 0.844 0.119 0.447
```

Comparing Performance

Summary results:

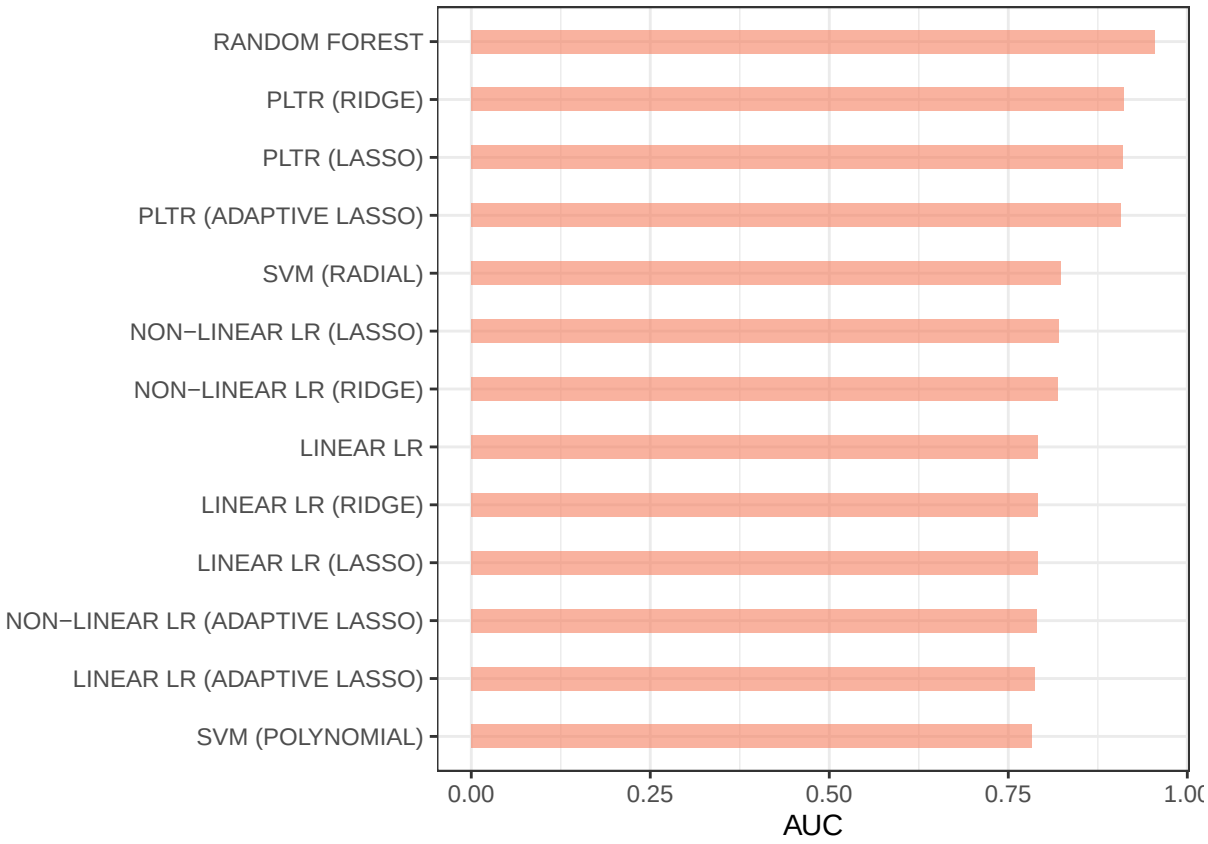
```r
summary_results <- dplyr::bind_rows(
   pltr_alasso_avg_results,
   pltr_lasso_avg_results,
   pltr_ridge_avg_results,
   rf_avg_results,
   svm_radial_avg_results,
   svm_polynomial_avg_results,
   llr_avg_results,
   llr_ridge_avg_results,
   llr_lasso_avg_results,
   llr_alasso_avg_results,
   nllr_ridge_avg_results,
   nllr_lasso_avg_results,
   nllr_alasso_avg_results
)

summary_results %>%
   dplyr::arrange(dplyr::desc(AUC))
```

```
## # A tibble: 13 x 6
##    LEARNING_ALGORITHM               AUC  GINI   PCC     BS    KS
##    <chr>                          <dbl> <dbl> <dbl>  <dbl> <dbl>
##  1 RANDOM FOREST                  0.955 0.910 0.912 0.0668 0.785
##  2 PLTR (RIDGE)                   0.911 0.823 0.890 0.0817 0.689
##  3 PLTR (LASSO)                   0.911 0.821 0.891 0.0810 0.689
##  4 PLTR (ADAPTIVE LASSO)          0.907 0.815 0.889 0.0826 0.687
##  5 SVM (RADIAL)                   0.823 0.646 0.812 0.0918 0.597
##  6 NON-LINEAR LR (LASSO)          0.821 0.642 0.857 0.110  0.498
##  7 NON-LINEAR LR (RIDGE)          0.819 0.638 0.852 0.112  0.494
##  8 LINEAR LR                      0.792 0.583 0.836 0.123  0.447
##  9 LINEAR LR (RIDGE)              0.792 0.583 0.834 0.123  0.450
## 10 LINEAR LR (LASSO)              0.791 0.583 0.836 0.123  0.443
## 11 NON-LINEAR LR (ADAPTIVE LASSO) 0.789 0.579 0.844 0.119  0.447
## 12 LINEAR LR (ADAPTIVE LASSO)     0.786 0.573 0.834 0.124  0.425
## 13 SVM (POLYNOMIAL)               0.783 0.566 0.839 0.111  0.499
```

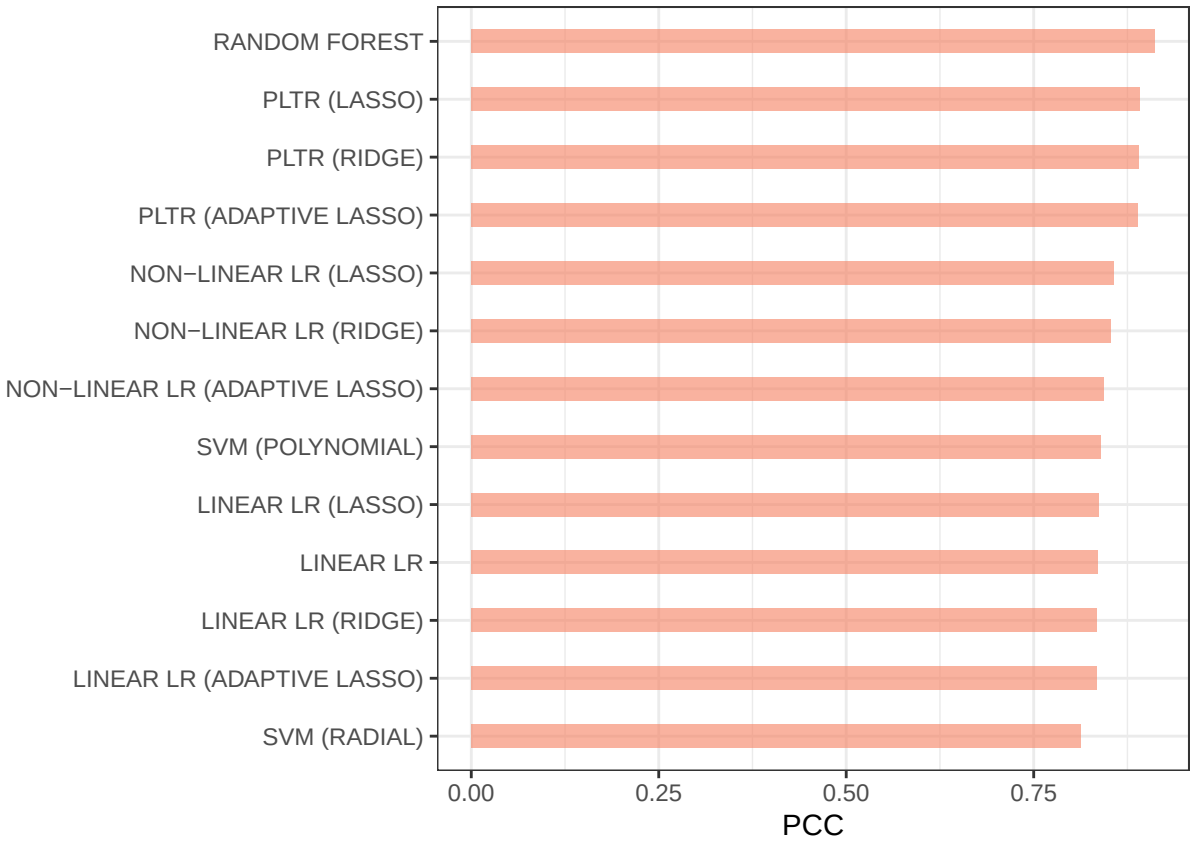
By AUC:

```
summary_results %>%
   dplyr::mutate(
      LEARNING_ALGORITHM = factor(LEARNING_ALGORITHM,
                                  levels = LEARNING_ALGORITHM[order(AUC)])
   ) %>%
   ggplot(aes(x = LEARNING_ALGORITHM, y = AUC)) +
   geom_bar(stat = "identity", fill = "#f68060", alpha = .6, width = .4) +
   coord_flip() +
   xlab("") +
   ylab("AUC") +
   theme_bw()
```



By PCC:

```
summary_results %>%
   dplyr::mutate(
      LEARNING_ALGORITHM = factor(LEARNING_ALGORITHM,
                                  levels = LEARNING_ALGORITHM[order(PCC)])
   ) %>%
   ggplot(aes(x = LEARNING_ALGORITHM, y = PCC)) +
   geom_bar(stat = "identity", fill = "#f68060", alpha = .6, width = .4) +
   coord_flip() +
   xlab("") +
   ylab("PCC") +
   theme_bw()
```


RANDOM FOREST
PLTR (LASSO)
PLTR (RIDGE)
PLTR (ADAPTIVE LASSO)
NON−LINEAR LR (LASSO)
NON−LINEAR LR (RIDGE)
NON−LINEAR LR (ADAPTIVE LASSO)
SVM (POLYNOMIAL)
LINEAR LR (LASSO)
LINEAR LR
LINEAR LR (RIDGE)
LINEAR LR (ADAPTIVE LASSO)
SVM (RADIAL)
0.00
0.25
0.50
0.75
PCC