Model Comparison

Mike Keating

Model Comparison

Task 1: Conceptual Questions

- What is the purpose of using cross-validation when fitting a random forest model?

  Cross-validation is an important method to reduce risk of over fitting a model and to prevent data leakage. Since CV is performed on multiple data splits, it helps form a better understanding of a random forest models performance on unseen (i.e. real world) data.

- Describe the bagged tree algorithm.

  Bagging stands for "bootstrap aggregation" in which multiple tree models are trained on bootstrapped data sets and the results aggregated to form the model's prediction. Bootstrapping is a sub sampling method that utilizes replacement, so each tree is trained on sub samples with varying compositions based on the original training set.

- What is meant by a general linear model?

  A general linear model is a generic framework for predicting the response of a dependent variable on one or more independent variables (regressors, predictors). The response is a general linear combination of the predictors.

- When fitting a multiple linear regression model, what does adding an interaction term do? That is, what does it allow the model to do differently as compared to when it is not included in the model?

  Adding an interaction term accounts for the situation where the value of one regressor can depend on the value of another regressor. This is represented in the formula as a product of these terms. This helps us model effects that are non-additive (say, they effect one type of person more than another, etc)

- Why do we split our data into a training and test set?

  In order to test the effectiveness of our model in real world situations, it needs to be tested on data that it was not trained on. Otherwise, the model may be over fit to the training data and the model would have poor generalization.

Task 2: Data Prep

Packages and Data

```
# Load Dependencies
library(tidyverse)
library(tidymodels)
library(caret)
library(yardstick)
library(ggplot2)
```

Q1: Summary

```
heart <- read_csv("data/heart.csv", show_col_types = FALSE)
summary(heart)
```

```
      Age            Sex            ChestPainType        RestingBP
 Min.   :28.00   Length:918         Length:918         Min.   :  0.0
 1st Qu.:47.00   Class :character   Class :character   1st Qu.:120.0
 Median :54.00   Mode  :character   Mode  :character   Median :130.0
 Mean   :53.51                                         Mean   :132.4
 3rd Qu.:60.00                                         3rd Qu.:140.0
 Max.   :77.00                                         Max.   :200.0
  Cholesterol      FastingBS       RestingECG            MaxHR
 Min.   :  0.0   Min.   :0.0000   Length:918         Min.   : 60.0
 1st Qu.:173.2   1st Qu.:0.0000   Class :character   1st Qu.:120.0
 Median :223.0   Median :0.0000   Mode  :character   Median :138.0
 Mean   :198.8   Mean   :0.2331                      Mean   :136.8
 3rd Qu.:267.0   3rd Qu.:0.0000                      3rd Qu.:156.0
 Max.   :603.0   Max.   :1.0000                      Max.   :202.0
 ExerciseAngina        Oldpeak          ST_Slope          HeartDisease
 Length:918         Min.   :-2.6000   Length:918         Min.   :0.0000
 Class :character   1st Qu.: 0.0000   Class :character   1st Qu.:0.0000
 Mode  :character   Median : 0.6000   Mode  :character   Median :1.0000
                    Mean   : 0.8874                      Mean   :0.5534
```

```
 3rd Qu.: 1.5000                      3rd Qu.:1.0000
 Max.   : 6.2000                      Max.   :1.0000
```

What type of variable (in R) is Heart Disease? Categorical or Quantitative?

Heart disease is a quantitative variable (double)

Does this make sense? Why or why not?

This does not make sense - as this variable should be treated as a factor, since it is a binary classification.

Q2: Alter HeartDisease Variable

```
# Change heart disease to the correct type
# Perform some dataset cleanup

heart <- heart |>
  mutate(HasHeartDisease = as_factor(HeartDisease)) |>
  select(!c(ST_Slope, HeartDisease))

head(heart)
```

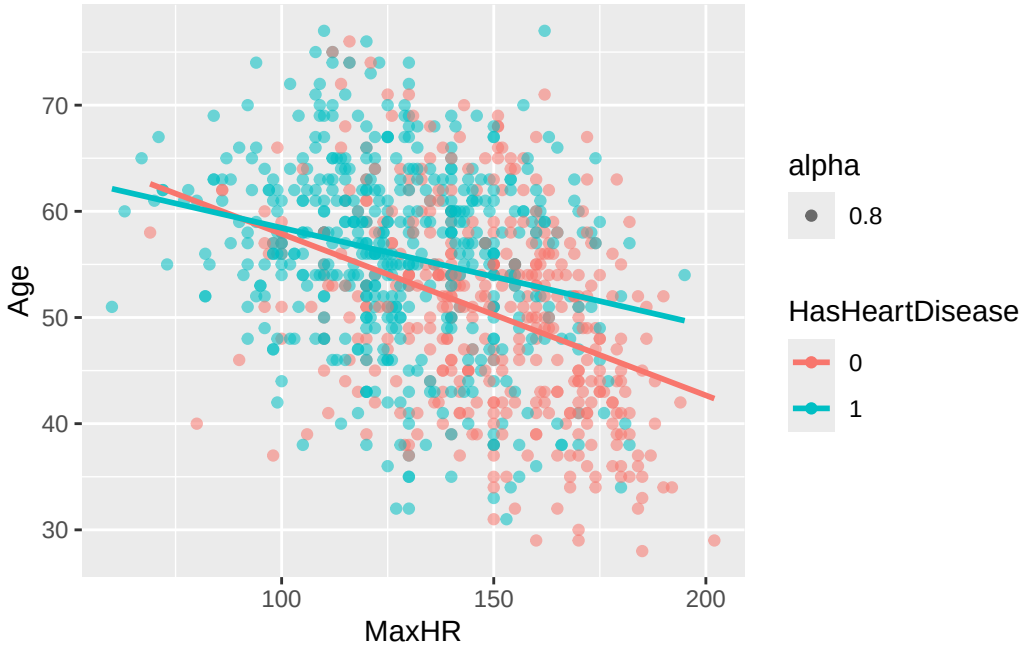
```
# A tibble: 6 x 11
    Age Sex   ChestPainType RestingBP Cholesterol FastingBS RestingECG MaxHR
  <dbl> <chr> <chr>             <dbl>       <dbl>     <dbl> <chr>      <dbl>
1    40 M     ATA                 140         289         0 Normal       172
2    49 F     NAP                 160         180         0 Normal       156
3    37 M     ATA                 130         283         0 ST            98
4    48 F     ASY                 138         214         0 Normal       108
5    54 M     NAP                 150         195         0 Normal       122
6    39 M     NAP                 120         339         0 Normal       170
# i 3 more variables: ExerciseAngina <chr>, Oldpeak <dbl>,
#   HasHeartDisease <fct>
```

Task 3: EDA

Q1: Plot

```
library(ggplot2)
g <- ggplot(heart, aes(x = MaxHR, y = Age, color=HasHeartDisease)) +
  geom_point(aes(alpha = 0.8)) +
  geom_smooth(method = "lm", se=FALSE) +
  labs() +
  theme_gray()
g
```

```
`geom_smooth()` using formula = 'y ~ x'
```



Q2: Additive vs Interaction

Since the above plot shows differing slopes for each factor of HasHeartDisease, an interaction model will be more suitable. However, it doesn't seem like the interaction is incredibly strong.

Task 4: Testing and Training

Split into test and train sets

```
# Set random seed
set.seed(101)

# Split into test and train sets
heart_split <- heart |> initial_split(prop=0.8)
train <- training(heart_split)
test <- testing(heart_split)
```

Task 5: OLS and LASSO

Q1: Fit Interaction Model

ols_mlr_rec <- recipe(Age ~ HasHeartDisease + MaxHR, data = train) |> step_normalize(all_numeric(), -all_outcomes()) |> step_interact(terms = ~ HasHeartDisease)

```
# Fit model
ols_mlr <- lm(Age ~ HasHeartDisease + MaxHR + HasHeartDisease:MaxHR, data = train)
#summary(ols_mlr)

# Recipe
ols_mlr_rec <- recipe(Age ~ HasHeartDisease + MaxHR, data = train) |>
  step_dummy(all_nominal_predictors()) |>
  step_interact(terms = ~ starts_with("HasHeart"):starts_with("MaxHR")) |>
  step_normalize(all_numeric_predictors())
```

```
# Model
ols_mlr <- linear_reg() |> set_engine("lm")

# Workflow
ols_mlr_wfl <- workflow() |> add_recipe(ols_mlr_rec) |> add_model(ols_mlr)
ols_mlr_fit <- ols_mlr_wfl |> fit(train)

ols_mlr_fit |> tidy()
```

```
# A tibble: 4 x 5
  term                       estimate std.error statistic  p.value
  <chr>                         <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)                   53.6      0.313    171.   0
2 MaxHR                         -4.30     0.522     -8.23 8.43e-16
3 HasHeartDisease_X1            -4.25     1.90      -2.24 2.55e- 2
4 HasHeartDisease_X1_x_MaxHR     5.48     1.78       3.07 2.21e- 3
```

Q2: Find RMSE

```
# Collect metrics/performance on the test set
ols_mlr_test_rmse <- ols_mlr_wfl |>
  last_fit(heart_split) |>
  collect_metrics() |>
  filter(.metric == "rmse")
ols_mlr_test_rmse
```

```
# A tibble: 1 x 4
  .metric .estimator .estimate .config
  <chr>   <chr>          <dbl> <chr>
1 rmse    standard        9.10 Preprocessor1_Model1
```

Q3: LASSO

```
# NOTE: The recipe/preprocessing steps are identical in this step to the ols model
LASSO_recipe <- recipe(Age ~ HasHeartDisease + MaxHR, data = train) |>
  step_dummy(all_nominal_predictors()) |>
  step_interact(terms = ~ starts_with("HasHeart"):MaxHR) |>
  step_normalize(all_numeric_predictors())

LASSO_recipe
```

```
-- Recipe ----------------------------------------------------------------------



-- Inputs

Number of variables by role

outcome:   1
predictor: 2
```

```
-- Operations

* Dummy variables from: all_nominal_predictors()

* Interactions with: starts_with("HasHeart"):MaxHR

* Centering and scaling for: all_numeric_predictors()
```

Q4: LASSO Model Selection

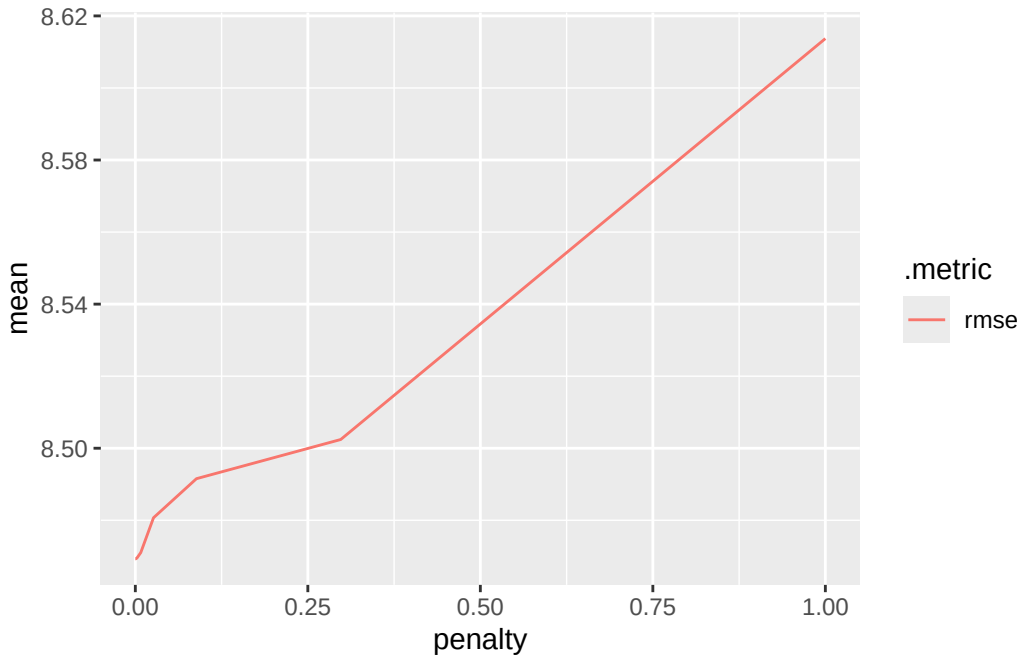
```r
# recall mixture = 1 sets to LASSO
LASSO_spec <- linear_reg(penalty = tune(), mixture = 1) |>
  set_engine("glmnet")

# Create workflow
LASSO_wkf <- workflow() |> add_recipe(LASSO_recipe) |> add_model(LASSO_spec)

# Cross-validation folds

heart_cv_folds <- vfold_cv(train, 5)

# Create grid object
LASSO_grid <- LASSO_wkf |>
  tune_grid(resamples = heart_cv_folds,
            grid = grid_regular(penalty(),
                                levels = 20))
```

```
Warning: package 'glmnet' was built under R version 4.5.1
```

```r
# Display grid
LASSO_grid
```

```
# Tuning results
# 5-fold cross-validation
# A tibble: 5 x 4
  splits            id    .metrics          .notes
  <list>            <chr> <list>            <list>
1 <split [587/147]> Fold1 <tibble [40 x 5]> <tibble [0 x 3]>
2 <split [587/147]> Fold2 <tibble [40 x 5]> <tibble [0 x 3]>
3 <split [587/147]> Fold3 <tibble [40 x 5]> <tibble [0 x 3]>
4 <split [587/147]> Fold4 <tibble [40 x 5]> <tibble [0 x 3]>
5 <split [588/146]> Fold5 <tibble [40 x 5]> <tibble [0 x 3]>
```

```
LASSO_grid |> collect_metrics() |>
  filter(.metric == "rmse") |>
  ggplot(aes(penalty, mean, color = .metric)) + geom_line()
```




```
# Select best model based on RMSE
lowest_rmse <- LASSO_grid |> select_best(metric = "rmse")

LASSO_final <- LASSO_wkf |> finalize_workflow(lowest_rmse) |>
  fit(train)
tidy(LASSO_final)
```

```
# A tibble: 4 x 3
  term                        estimate      penalty
  <chr>                          <dbl>        <dbl>
1 (Intercept)                    53.6  0.0000000001
2 MaxHR                          -4.24 0.0000000001
3 HasHeartDisease_X1             -4.01 0.0000000001
4 HasHeartDisease_X1_x_MaxHR      5.26 0.0000000001
```

Q5: Model Expectations

Without looking at the RMSE calculations, I would expect the RMSE calculations to be roughly the same. This is because we already pre-selected only HasHeartDisease and MaxHR as predictors. The LASSO method would likely have a more dramatic effect on models with larger amount of initial predictors. I also imagine both models might reach a prediction plateau based on only these features.

Q6: Compare OLS and LASSO

```
# Get rmse from our LASSO model
LASSO_test_rmse <- LASSO_wkf |>
  finalize_workflow(lowest_rmse) |>
  last_fit(heart_split) |>
  collect_metrics() |>
  filter(.metric == "rmse")

# Combine our metrics
rbind(ols_mlr_test_rmse, LASSO_test_rmse) |>
  mutate(Model = c("OLS", "LASSO")) |>
  select(Model, everything())
```

```
# A tibble: 2 x 5
  Model .metric .estimator .estimate .config
  <chr> <chr>   <chr>          <dbl> <chr>
1 OLS   rmse    standard        9.10 Preprocessor1_Model1
2 LASSO rmse    standard        9.09 Preprocessor1_Model1
```

Q7: Explain RMSE Similarity

At the end of the day, the coefficients are still fairly similar and the LASSO method did not enact a large amount of shrinkage. The number of predictors was largely unchanged (none shrunk to exactly zero) and the models likely arrive at similar predictions.

Task 6: Logistic Regression

```
# First Model
# All predictors

# Use the same split as before
LR1_recipe <- recipe(HasHeartDisease ~ . , data = train) |>
```

```
  step_normalize(all_numeric_predictors())

# Second Model
#
LR2_recipe <- recipe(HasHeartDisease ~ Age + Sex + Cholesterol, data = train) |>
  step_normalize(all_numeric_predictors())

# Spec shared between both models
LR_spec <- logistic_reg() |>
  set_engine("glm")

# Create workflows
LR1_wkf <- workflow() |>
  add_recipe(LR1_recipe) |>
  add_model(LR_spec)

LR2_wkf <- workflow() |>
  add_recipe(LR2_recipe) |>
  add_model(LR_spec)

LR_cv_folds = vfold_cv(train, 5, 3) # 3 repeats

# Fit models
LR1_fit <- LR1_wkf |>
  fit_resamples(LR_cv_folds, metrics = metric_set(accuracy, mn_log_loss))

LR2_fit <- LR2_wkf |>
  fit_resamples(LR_cv_folds, metrics = metric_set(accuracy, mn_log_loss))

# Metrics
rbind(LR1_fit |> collect_metrics(),
      LR2_fit |> collect_metrics()) |> mutate(Model = c("Model 1", "Model 1", "Model 2", "Mo
```

```
# A tibble: 4 x 7
  Model   .metric     .estimator  mean     n std_err .config
  <chr>   <chr>       <chr>      <dbl> <int>   <dbl> <chr>
1 Model 1 accuracy    binary     0.814    15 0.00673 Preprocessor1_Model1
2 Model 1 mn_log_loss binary     0.414    15 0.0107  Preprocessor1_Model1
3 Model 2 accuracy    binary     0.687    15 0.00801 Preprocessor1_Model1
4 Model 2 mn_log_loss binary     0.589    15 0.00853 Preprocessor1_Model1
```

Model 1 (all predictors) performs better on both accuracy (0.811) and log loss (0419, lower is

better).

```
# Fit model 1
LR_final_fit <- LR1_wkf |> fit(data = train)
LR_preds <- predict(LR_final_fit, test) |> pull(.pred_class)

confusionMatrix(data = LR_preds, reference = test$HasHeartDisease)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 83 17
         1 11 73

               Accuracy : 0.8478
                 95% CI : (0.7876, 0.8964)
    No Information Rate : 0.5109
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6951

 Mcnemar's Test P-Value : 0.3447

            Sensitivity : 0.8830
            Specificity : 0.8111
         Pos Pred Value : 0.8300
         Neg Pred Value : 0.8690
             Prevalence : 0.5109
         Detection Rate : 0.4511
   Detection Prevalence : 0.5435
      Balanced Accuracy : 0.8470

       'Positive' Class : 0
```

Sensitivity is also known as recall and represents the true positive rate. This is how well we identify true positives (i.e. 1, or the patient has heart disease)

Specificity represents the true negative rate, or how well our model identifies true negatives.