

ST558 HomeWork 5

Keshav Ramesh

Task 1 Conceptual questions

1. What is the purpose of using cross-validation when fitting a random forest model?
 - cross validation is to check the performance and generalization of the random forest model. Although they already reduce over fitting, cross validation make a more accurate guess of how the model will perform on unseen data
2. Describe the bagged tree algorithm.
 - Bagging is an ensemble technique that builds decision trees on different bootstrapped samples of an original dataset. each tree is created independently and the results are averaged creating a more stable and accurate prediction.
3. What is meant by a general linear model?
 - A general linear model is a statistical model where the outcome is a linear combination of predictors. This includes, simple linear regression, multiple linear regression.
$$Y = X_B + e$$
4. 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?
 - The interaction term allow for the model to check situation where one predictor depends on the value of another., It enables the model to check for more complex relationships between variables.
5. Why do we split our data into a training and test set?
 - Data is split into training and test sets to evaluate the model's ability to generalize. The model is trained on a portion of a set then used to see if that training holds up on unseen data. This prevents over fitting, and increases accuracy.

Task 2: Data Prep

Packages and Data

```
library(tidyverse)
library(tidymodels)
library(caret)
library(yardstick)
library(glmnet)

heart = as_tibble(read_csv("heart.csv"))
```

Question 1

```
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

- heart disease is being treated as a quantitative variable, it is storing values as either 1 or 0.
- This does not make sense as we should be treating the variables as categorical, as a yes or no response. While this does work in theory as a binary 1/0 response, it would be better to treat this as a factor with levels of yes or no, or T and F.

Question 2

```
new_heart = heart %>%
  mutate(HeartDisease_FACTORED = factor(HeartDisease, levels = c(0,1), labels = c("N", "Y"))) %>%
  select(-ST_Slope, -HeartDisease)

head(new_heart)
```

```

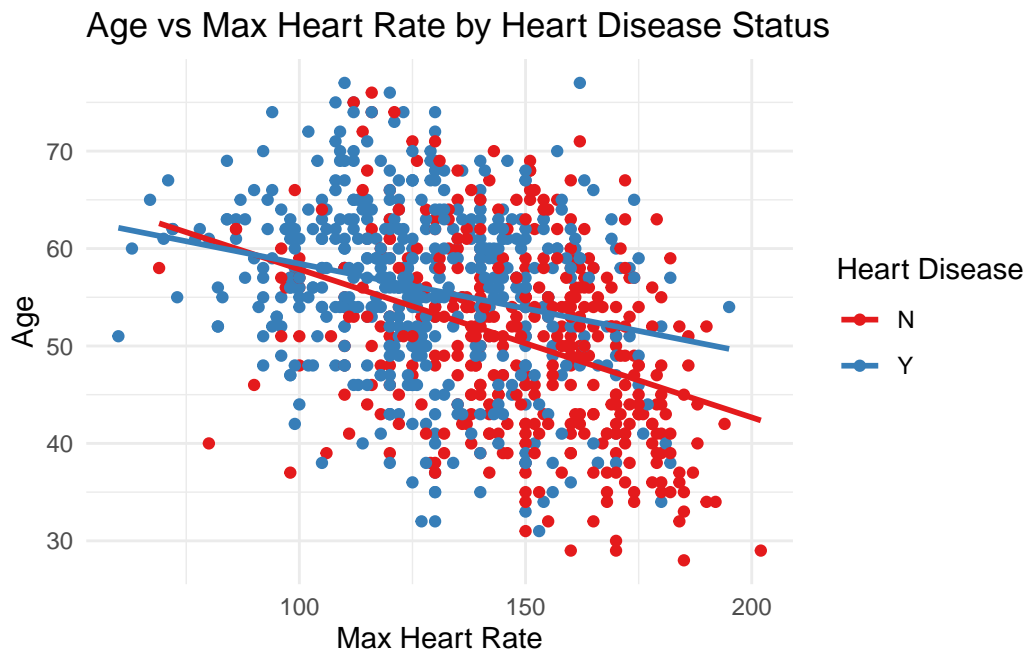
# A tibble: 6 x 11
  Age Sex ChestPainType RestingBP Cholesterol FastingBS RestingECG MaxHR
  <dbl> <chr> <chr>          <dbl>      <dbl>      <dbl> <chr>      <dbl>
1    40 M    ATA             140        289        0 Normal      172
2    49 F    NAP             160        180        0 Normal      156
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>,
#   HeartDisease_FACTORED <fct>

```

Task 3: EDA

```
new_heart %>%
  ggplot(
    aes(
      x = MaxHR,
      y = Age,
      colour = HeartDisease_FACTORED
    )
  ) +
  geom_point() +
  geom_smooth(method = "lm", se = F) +
  scale_color_brewer(palette = "Set1") +
  labs(
    title = "Age vs Max Heart Rate by Heart Disease Status",
    x = "Max Heart Rate",
    y = "Age",
    color = "Heart Disease"
  ) +
  theme_minimal()
```

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



Question 2

An interaction model would be more accurate. The relationship between max heart rate and age differed depending on whether or not the person has heart disease. They have different slopes indicating that max hr depends on heart disease status. This is aligned with an interactive model not an additive model.

Task 4: Testing and Training

```
set.seed(101)

heart_split = initial_split(new_heart, prop = 0.8)

train = training(heart_split)
test = testing(heart_split)
```

Task 5: OLS and LASSO

Question 1

```
ols_mlr = lm(Age ~ MaxHR + HeartDisease_FACTORED + MaxHR*HeartDisease_FACTORED, data = train)

summary(ols_mlr)
```

Call:

```
lm(formula = Age ~ MaxHR + HeartDisease_FACTORED + MaxHR * HeartDisease_FACTORED,
    data = train)
```

Residuals:

Min	1Q	Median	3Q	Max
-22.7703	-5.7966	0.4516	5.7772	20.6378

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	75.58896	3.07510	24.581	< 2e-16 ***
MaxHR	-0.16992	0.02064	-8.233	8.43e-16 ***
HeartDisease_FACTORED	-8.58502	3.83433	-2.239	0.02546 *
MaxHR:HeartDisease_FACTORED	0.08343	0.02716	3.072	0.00221 **

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.478 on 730 degrees of freedom

Multiple R-squared: 0.1839, Adjusted R-squared: 0.1806

F-statistic: 54.84 on 3 and 730 DF, p-value: < 2.2e-16

Question 2

```
yardstick::rmse_vec(test$Age, predict(ols_mlr, test))
```

```
[1] 9.100206
```

Question 3

```
LASSO_recipe = recipe(Age ~ MaxHR + HeartDisease_FACTORED, data = train) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_normalize(all_predictors()) %>%
  step_interact(~ MaxHR:starts_with("HeartDisease_FACTORED"))

LASSO_recipe
```

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

```
-- Inputs
```

```
Number of variables by role
```

```
outcome: 1
predictor: 2
```

```
-- Operations
```

```
* Dummy variables from: all_nominal_predictors()

* Centering and scaling for: all_predictors()

* Interactions with: MaxHR:starts_with("HeartDisease_FACTORED")
```

Question 4

```
set.seed(101)
cv_folds = vfold_cv(train, v = 10)

lasso_model = linear_reg(penalty = tune(), mixture = 1) %>%
  set_engine("glmnet")

lasso_workflow = workflow() %>%
  add_model(lasso_model) %>%
  add_recipe(LASSO_recipe)
lasso_workflow
```

```
== Workflow =====
Preprocessor: Recipe
Model: linear_reg()
```

```
-- Preprocessor -----
```

```
3 Recipe Steps
```

```
* step_dummy()
```

```

* step_normalize()
* step_interact()

-- Model -----
Linear Regression Model Specification (regression)

Main Arguments:
  penalty = tune()
  mixture = 1

Computational engine: glmnet

```

```

LASSO_grid = lasso_workflow %>%
  tune_grid(resamples = cv_folds, grid = grid_regular(penalty(), levels = 200))

```

```
LASSO_grid
```

```

# Tuning results
# 10-fold cross-validation
# A tibble: 10 x 4
  splits          id      .metrics      .notes
  <list>         <chr>   <list>      <list>
1 <split [660/74]> Fold01 <tibble [400 x 5]> <tibble [0 x 3]>
2 <split [660/74]> Fold02 <tibble [400 x 5]> <tibble [0 x 3]>
3 <split [660/74]> Fold03 <tibble [400 x 5]> <tibble [0 x 3]>
4 <split [660/74]> Fold04 <tibble [400 x 5]> <tibble [0 x 3]>
5 <split [661/73]> Fold05 <tibble [400 x 5]> <tibble [0 x 3]>
6 <split [661/73]> Fold06 <tibble [400 x 5]> <tibble [0 x 3]>
7 <split [661/73]> Fold07 <tibble [400 x 5]> <tibble [0 x 3]>
8 <split [661/73]> Fold08 <tibble [400 x 5]> <tibble [0 x 3]>
9 <split [661/73]> Fold09 <tibble [400 x 5]> <tibble [0 x 3]>
10 <split [661/73]> Fold10 <tibble [400 x 5]> <tibble [0 x 3]>

```

```

LASSO_grid %>%
  collect_metrics() %>%
  filter(.metric == "rmse")

```

```

# A tibble: 200 x 7
  penalty .metric .estimator mean      n std_err .config
  <dbl> <chr>   <chr>      <dbl> <int>  <dbl> <chr>
1 1e-10 rmse    standard    8.47    10  0.124 Preprocessor1_Model001
2 1.12e-10 rmse    standard    8.47    10  0.124 Preprocessor1_Model002
3 1.26e-10 rmse    standard    8.47    10  0.124 Preprocessor1_Model003
4 1.41e-10 rmse    standard    8.47    10  0.124 Preprocessor1_Model004
5 1.59e-10 rmse    standard    8.47    10  0.124 Preprocessor1_Model005
6 1.78e-10 rmse    standard    8.47    10  0.124 Preprocessor1_Model006
7 2.00e-10 rmse    standard    8.47    10  0.124 Preprocessor1_Model007
8 2.25e-10 rmse    standard    8.47    10  0.124 Preprocessor1_Model008
9 2.52e-10 rmse    standard    8.47    10  0.124 Preprocessor1_Model009
10 2.83e-10 rmse    standard    8.47    10  0.124 Preprocessor1_Model010
# i 190 more rows

```

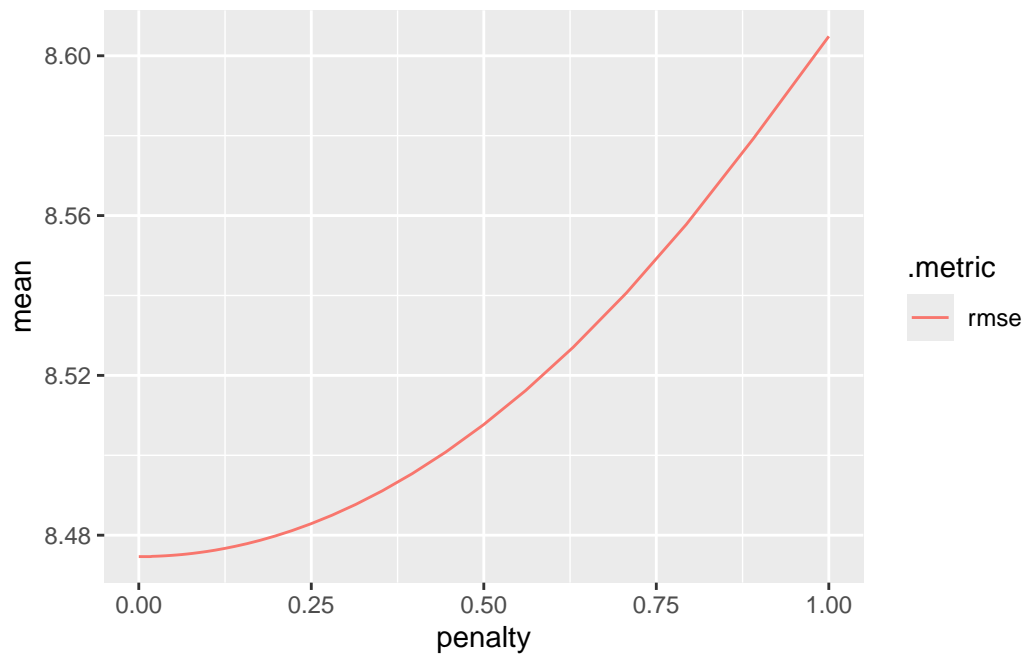
```

LASSO_grid %>%
  collect_metrics() %>%
  filter(.metric == "rmse") %>%

```



```
ggplot(aes(penalty, mean, color = .metric)) +
  geom_line()
```



```
lowest_rmse <- LASSO_grid %>%
  select_best(metric = "rmse")
lowest_rmse
```

```
# A tibble: 1 x 2
  penalty .config
  <dbl> <chr>
1 0.0000000001 Preprocessor1_Model001
```

```
lasso_workflow %>%
  finalize_workflow(lowest_rmse)
```

```
== Workflow =====
Preprocessor: Recipe
Model: linear_reg()

-- Preprocessor -----
3 Recipe Steps

* step_dummy()
* step_normalize()
* step_interact()

-- Model -----
Linear Regression Model Specification (regression)

Main Arguments:
  penalty = 1e-10
  mixture = 1

Computational engine: glmnet
```

```
#fit it to the entire training set to see the model fit
LASSO_final <- lasso_workflow %>%
  finalize_workflow(lowest_rmse) %>%
  fit(train)

tidy(LASSO_final)
```

```
# A tibble: 4 x 3
  term                estimate    penalty
  <chr>                <dbl>    <dbl>
1 (Intercept)         54.0  0.0000000001
2 MaxHR               -3.08  0.0000000001
3 HeartDisease_FACTORED_Y    1.36  0.0000000001
4 MaxHR_x_HeartDisease_FACTORED_Y    1.03  0.0000000001
```

Question 5

I would expect the outputs to be generally the same between the OLS and LASSO models.

based on the tidy output of LASSO_final we can see that the LASSO model has the same predictors that the OLS model has. This would indicate that RMSE would also be very similar.

Question 6

```
ols_mlr %>%
  predict(test) %>%
  rmse_vec(truth = test$Age)
```

```
[1] 9.100206
```

```
LASSO_final %>%
  predict(test) %>%
  pull() %>%
  rmse_vec(truth = test$Age)
```

```
[1] 9.095981
```

Question 7

They are both using the same predictors, and their interactions are also the same and simple.

Task 6: Logistic regression

Question 1

```
logrec1 <- recipe(HeartDisease_FACTORED ~ MaxHR, data = train) %>%
  step_normalize(MaxHR)

logrec2 <- recipe(HeartDisease_FACTORED ~ Age + Sex + ChestPainType + RestingBP, data = train) %>%
  step_normalize(all_numeric(), -HeartDisease_FACTORED) %>%
  step_dummy(all_nominal_predictors())
```

```
set.seed(101)
cv_repeats <- vfold_cv(train, 10, 5)
```

```
log_spec <- logistic_reg() %>%
  set_engine("glm")
```

```
LR1_wkf <- workflow() |>
add_recipe(logrec1) |>
add_model(log_spec)
```

```
LR2_wkf <- workflow() |>
add_recipe(logrec2) |>
add_model(log_spec)
```

```
LR1_fit <- LR1_wkf |>
fit_resamples(cv_repeats, metrics = metric_set(accuracy, mn_log_loss))
```

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

```
rbind(LR1_fit |> collect_metrics(),
LR2_fit |> collect_metrics())|>
mutate(Model = c("Model1", "Model1", "Model2", "Model2")) |>
select(Model, everything())
```

```
# A tibble: 4 x 7
  Model .metric .estimator mean     n std_err .config
  <chr> <chr>    <chr>    <dbl> <int>   <dbl> <chr>
1 Model1 accuracy  binary    0.677    50 0.00670 Preprocessor1_Model1
2 Model1 mn_log_loss binary    0.607    50 0.00550 Preprocessor1_Model1
3 Model2 accuracy  binary    0.776    50 0.00676 Preprocessor1_Model1
4 Model2 mn_log_loss binary    0.490    50 0.00913 Preprocessor1_Model1
```

```
mean(train$HeartDisease_FACTORED == "Y")
```

```
[1] 0.5694823
```

Model 2 was the better performing model it had the highest accuracy and the lowest loss.

Question 2

```
final_model <- LR2_wkf %>%
  fit(data = train)

predictions <- predict(final_model, test, type = "class") %>%
  bind_cols(test)

confusionMatrix(
  data = predictions$.pred_class,
  reference = predictions$HeartDisease_FACTORED
)
```

Confusion Matrix and Statistics

```

      Reference
Prediction N  Y
      N  71 17
      Y  23 73

      Accuracy : 0.7826
      95% CI : (0.716, 0.8399)
      No Information Rate : 0.5109
      P-Value [Acc > NIR] : 2.577e-14

      Kappa : 0.5656

      McNemar's Test P-Value : 0.4292

      Sensitivity : 0.7553
      Specificity : 0.8111
      Pos Pred Value : 0.8068
      Neg Pred Value : 0.7604
      Prevalence : 0.5109
      Detection Rate : 0.3859
      Detection Prevalence : 0.4783
      Balanced Accuracy : 0.7832

      'Positive' Class : N
```

Question 3

The model's sensitivity was 0.7553: The model will miss about 24.5% of healthy patients. It will false flag almost 1/4 healthy people as diseased.

The model's specificity was 0.8111: The model correctly identified 81% of actual heart disease cases but will miss 19% of diseased patients.

The model is better at confirming positive cases than it is with ruling out disease.