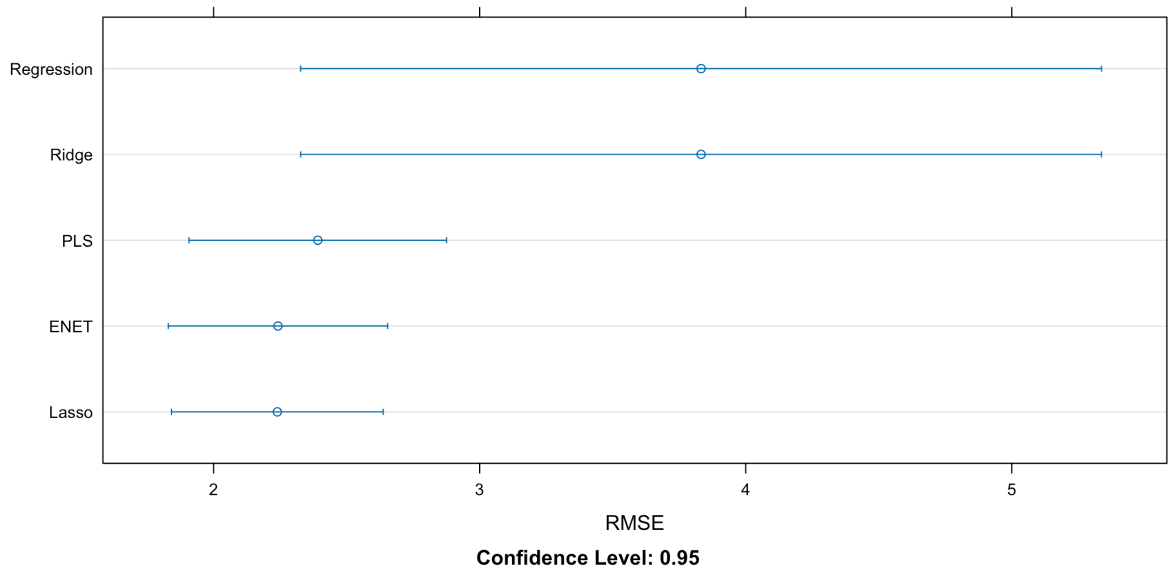
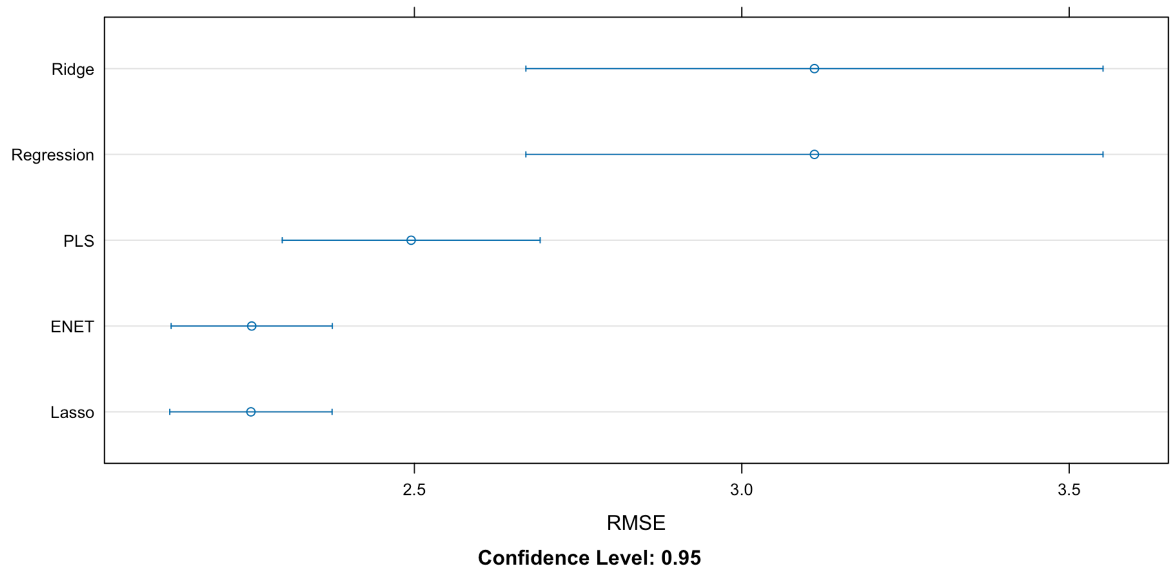
CV

```
dotplot(train_metrics_cv, metric = 'I
```



Repeated CV

```
dotplot(train_metrics, metric = 'RMSI
```



ADS 503 - Applied Predictive Modeling (M4)

Summer 2024 - Week 4

Dave Hurst

University of San Diego
SHILEY-MARCOS
SCHOOL OF ENGINEERING

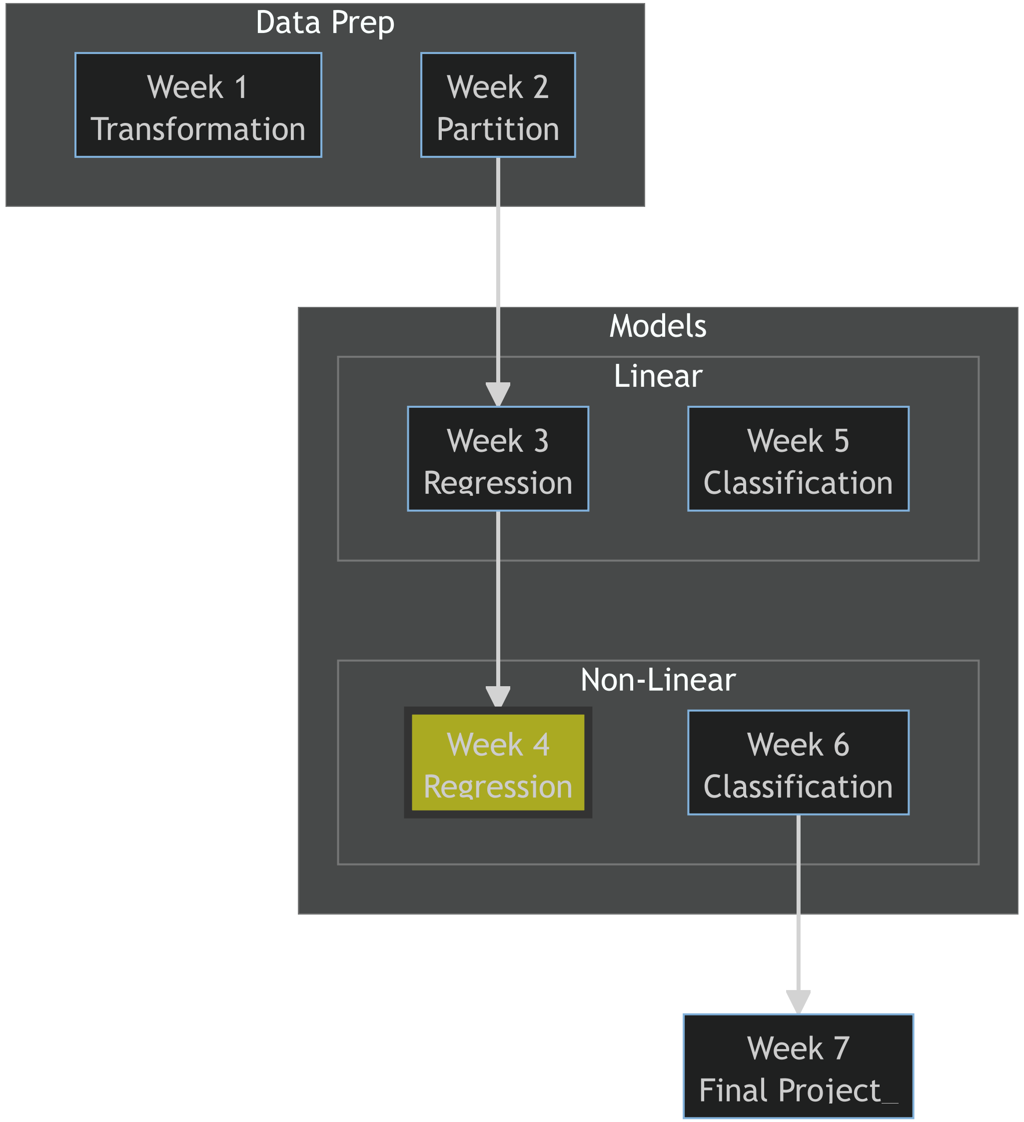
Start Recording!

Agenda

- Course Map
- Assignment 3 Review
  - Hyperparameter Tuning
- Assignment 4 Tips
- QA

University of San Diego
SHILEY-MARCOS
SCHOOL OF ENGINEERING

Course Map


Data Prep
Week 1
Transformation
Week 2
Partition
Models
Linear
Week 3
Regression
Week 5
Classification
Non-Linear
Week 4
Regression
Week 6
Classification
Week 7
Final Project_

Assignment 3 Review

University of San Diego®
SHILEY-MARCOS
SCHOOL OF ENGINEERING

3.1.c with `method = "cv"`

```r
seed <- 503
data(tecator)

# Extract the predictors (absorbance) and response (fat content)
absorbance <- as.data.frame(absorp)
fat_content <- endpoints[,2]  #  fat is the second column

# Split the data into training and validation sets
set.seed(seed)  # for reproducibility
train_index <- createDataPartition(fat_content, p = 0.8, list = FALSE)
train_data <- absorbance[train_index, ]
train_fat <- fat_content[train_index]
test_data <- absorbance[-train_index, ]
test_fat <- fat_content[-train_index]

#CV
train_cv <- trainControl(method = "cv")

tic('31c')
elapsed <- numeric()
# Linear Regression
set.seed(seed); tic('lm')  # for reproducibility/timing
lm_model_cv <- train(train_data, train_fat,
                  method = "lm",
                  preProcess = c("center", "scale"),
                  trControl = train_cv
```

Models/CV	Times
LinReg	2.110
PLS	2.260
Ridge	35.777
Lasso	6.734
ENET	18.011

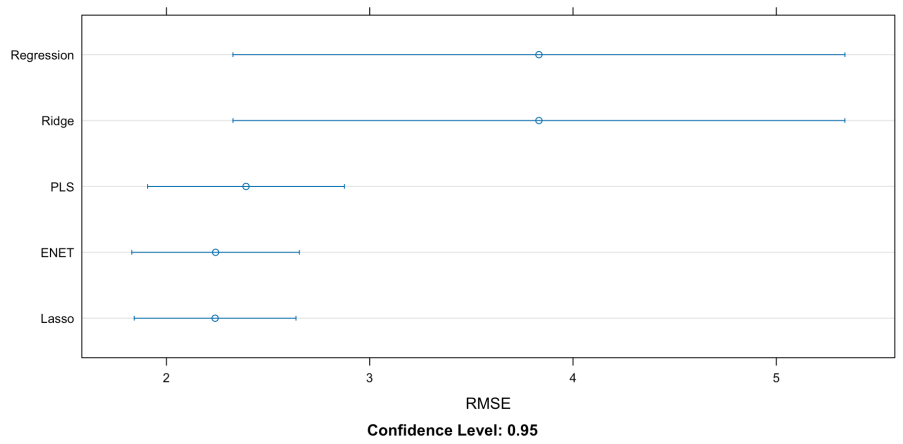
University of San Diego
SHILEY-MARCOS SCHOOL OF ENGINEERING

3.1.c with `method = "cv"`

```r
train_metrics_cv <- resamples(list(
    Regression = lm_model_cv,
    PLS = pls_model_cv,
    Ridge = ridge_model_cv,
    Lasso = lasso_model_cv,
    ENET = enet_model_cv))
summary(train_metrics_cv)$statistics$RMSE
```

```
               Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
Regression 2.154094 2.416813 3.067239 3.832277 4.194495 8.925659    0
PLS        1.077971 2.070936 2.547476 2.391443 2.874441 3.182084    0
Ridge      2.154062 2.416863 3.067253 3.832244 4.194643 8.924921    0
Lasso      1.217658 1.971209 2.236171 2.239400 2.560048 3.040155    0
ENET       1.116217 1.971931 2.241003 2.241807 2.638687 3.000962    0
```

```r
dotplot(train_metrics_cv, metric = 'RMSE
```




```r
diff(train_metrics_cv, metric = "RMSE") |> summary()
```

```
Call:
summary.diff.resamples(object = diff(train_metrics_cv, metric = "RMSE"))

p-value adjustment: bonferroni
Upper diagonal: estimates of the difference
Lower diagonal: p-value for H0: difference = 0

RMSE
           Regression PLS        Ridge      Lasso      ENET      
Regression             1.441e+00  3.316e-05  1.593e+00  1.590e+00
PLS        0.6874                -1.441e+00  1.520e-01  1.496e-01
Ridge      1.0000     0.6872                 1.593e+00  1.590e+00
Lasso      0.4395     1.0000     0.4394                -2.407e-03
ENET       0.4471     1.0000     0.4470     1.0000               
```

University of San Diego
SHILEY-MARCOS SCHOOL OF ENGINEERING

3.1.c with `method = "repeatedcv"`

```r
#Repeated CV
train_repeated_cv <- trainControl(method = "repeatedcv", repeats = 5)

tic('31c')
elapsed <- numeric()
# Linear Regression
set.seed(seed); tic('lm')  # for reproducibility/timing
lm_model <- train(train_data, train_fat,
                  method = "lm",
                  preProcess = c("center", "scale"),
                  trControl = train_repeated_cv
                  )
tics_lm <- toc(quiet = TRUE); elapsed <- c(elapsed, tics_lm$tic - tics_lm$toc)


# Partial Least Squares
set.seed(seed); tic('pls')  # for reproducibility
pls_model <- train(train_data, train_fat,
                   method = "pls",
                   tuneLength = 40,
                   preProcess = c("center", "scale"),
                   trControl = train_repeated_cv
                   )
tics_pls <- toc(quiet = TRUE); elapsed <- c(elapsed, tics_pls$tic - tics_pls$toc)

# Ridge Regression
```

Models/RCV	Times
LinReg	8.212
PLS	8.598
Ridge	182.964
Lasso	30.580
ENET	77.334

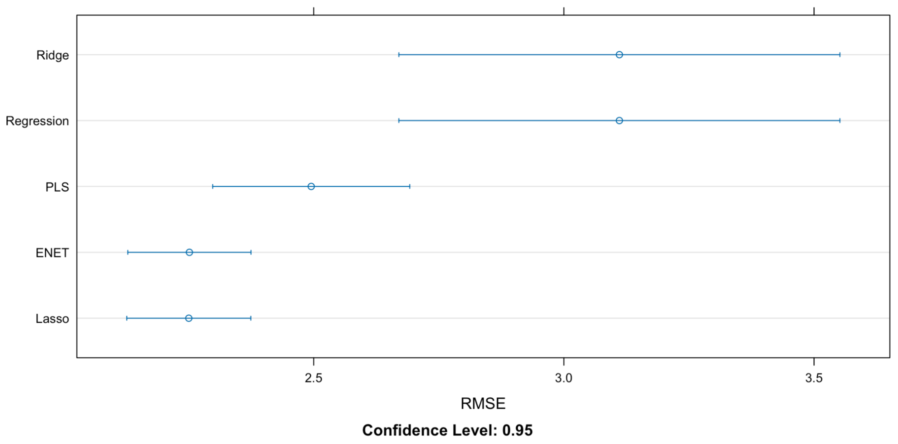
University of San Diego
SHILEY-MARCOS SCHOOL OF ENGINEERING

3.1.c with `method = "repeatedcv"`

```r
train_metrics <- resamples(list(
    Regression = lm_model,
    PLS = pls_model,
    Ridge = ridge_model,
    Lasso = lasso_model,
    ENET = enet_model))
summary(train_metrics)$statistics$RMSE
```

```
               Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
Regression 1.274747 2.063995 2.590646 3.110938 3.568557 8.925659    0
PLS        1.077971 2.053263 2.496077 2.495050 2.903803 4.966348    0
Ridge      1.274757 2.063859 2.590538 3.110968 3.568723 8.924921    0
Lasso      1.217658 1.960208 2.240318 2.250353 2.535707 3.229732    0
ENET       1.240802 1.974018 2.232149 2.251569 2.522635 3.257020    0
```

```r
dotplot(train_metrics, metric = 'RMSE')
```




```r
diff(train_metrics, metric = "RMSE") |> summary()
```

```
Call:
summary.diff.resamples(object = diff(train_metrics, metric = "RMSE"))

p-value adjustment: bonferroni
Upper diagonal: estimates of the difference
Lower diagonal: p-value for H0: difference = 0

RMSE
           Regression PLS        Ridge      Lasso      ENET      
Regression             6.159e-01 -3.037e-05  8.606e-01  8.594e-01
PLS        0.092884              -6.159e-01  2.447e-01  2.435e-01
Ridge      1.000000   0.092848               8.606e-01  8.594e-01
Lasso      0.003122   0.023851   0.003120              -1.216e-03
ENET       0.003152   0.030100   0.003151   1.000000             
```

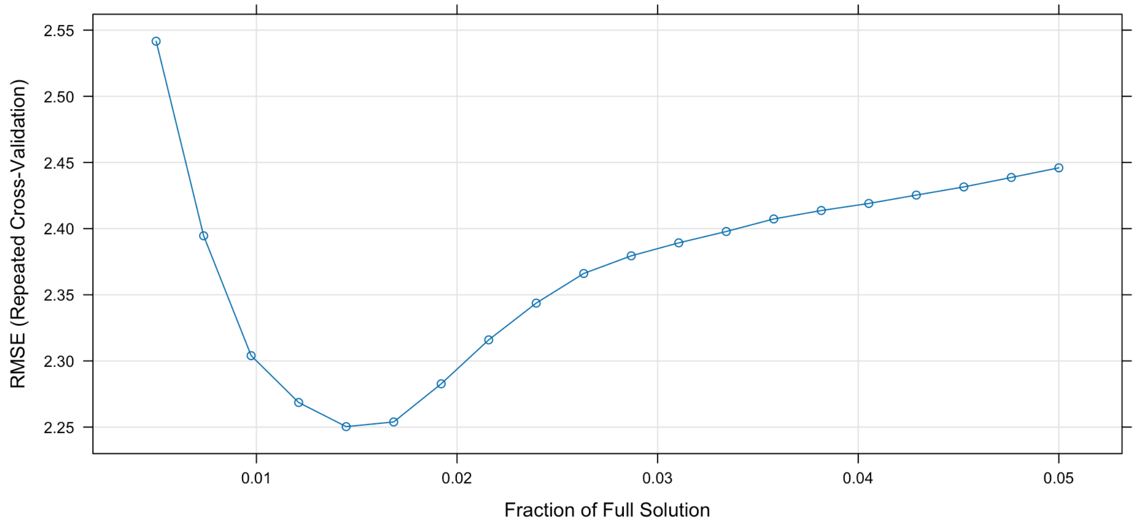
University of San Diego®
SHILEY-MARCOS
SCHOOL OF ENGINEERING

Hyperparameter Optimizaton

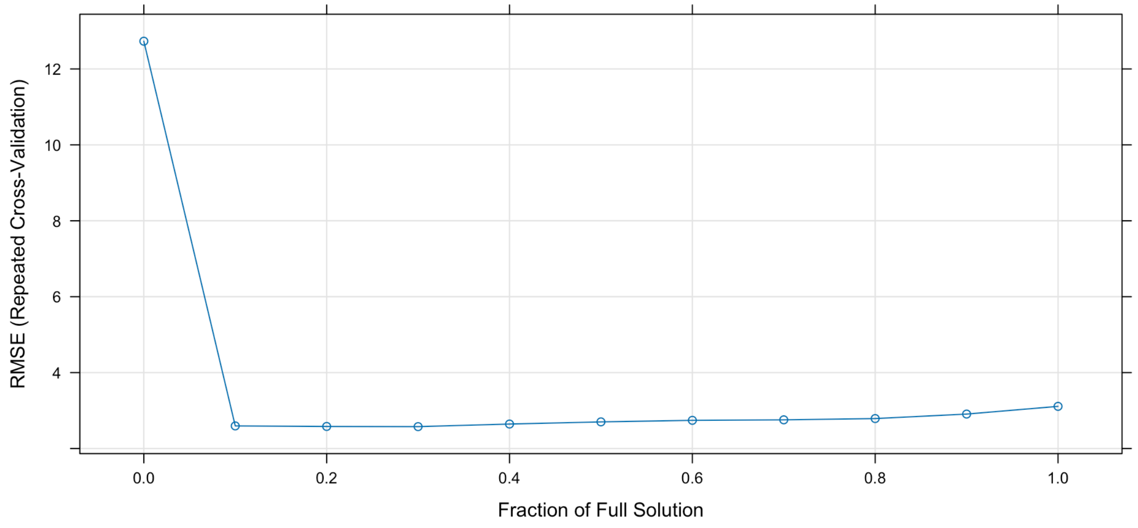
```r
lambda_grid <- data.frame(.fraction = seq(0.005, 0.05, length = 20))
lambda_grid2 <- data.frame(.fraction = seq(0, 1, length = 11))

# Lasso Regression
set.seed(seed)
lasso_model2 <- train(train_data, train_fat,
                     method = "lasso",
                     tuneGrid =  lambda_grid2,
                     preProcess = c("center", "scale"),
                     trControl = train_repeated_cv
)
```

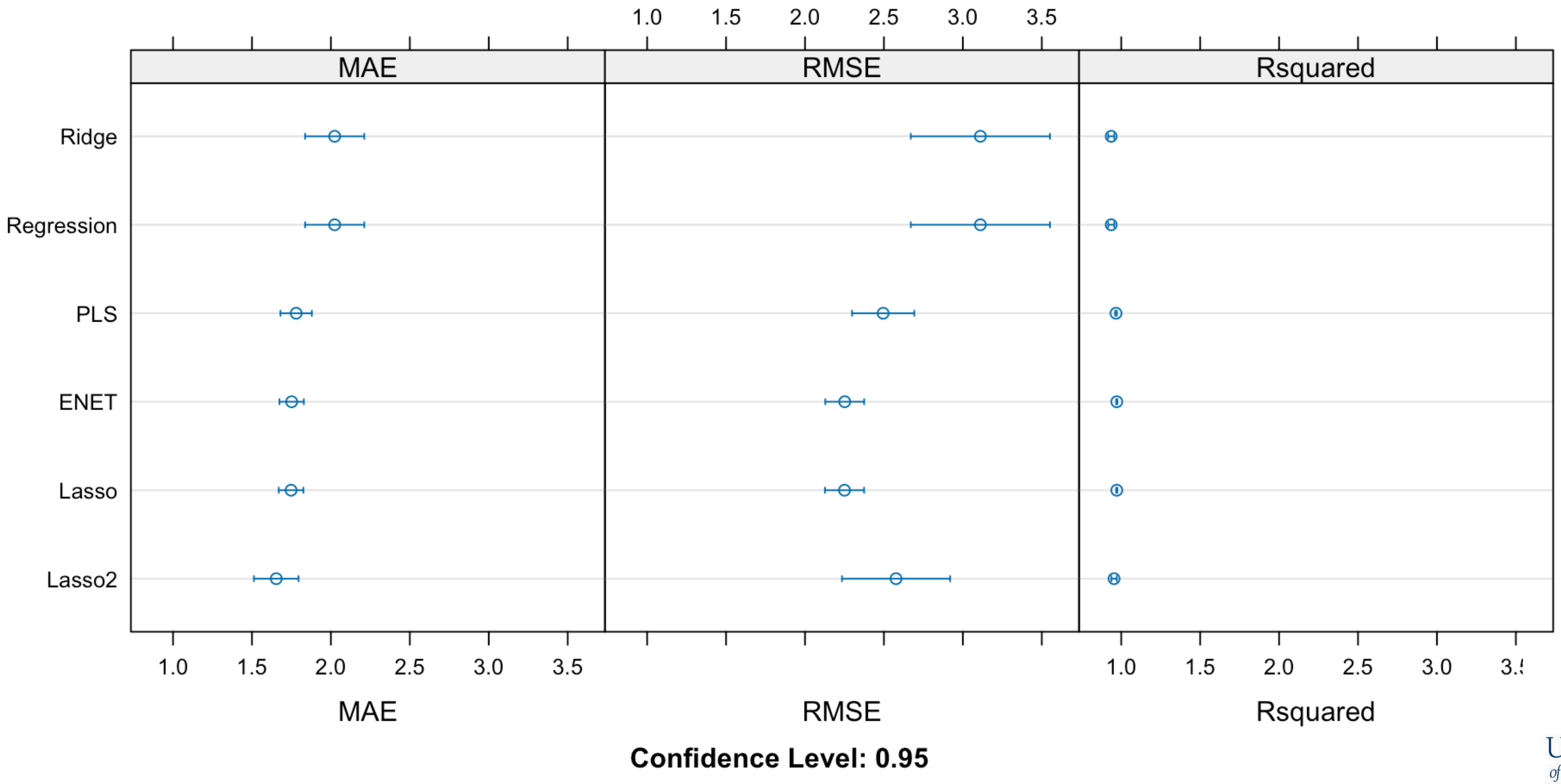
```r
plot(lasso_model)
```




```r
plot(lasso_model2)
```

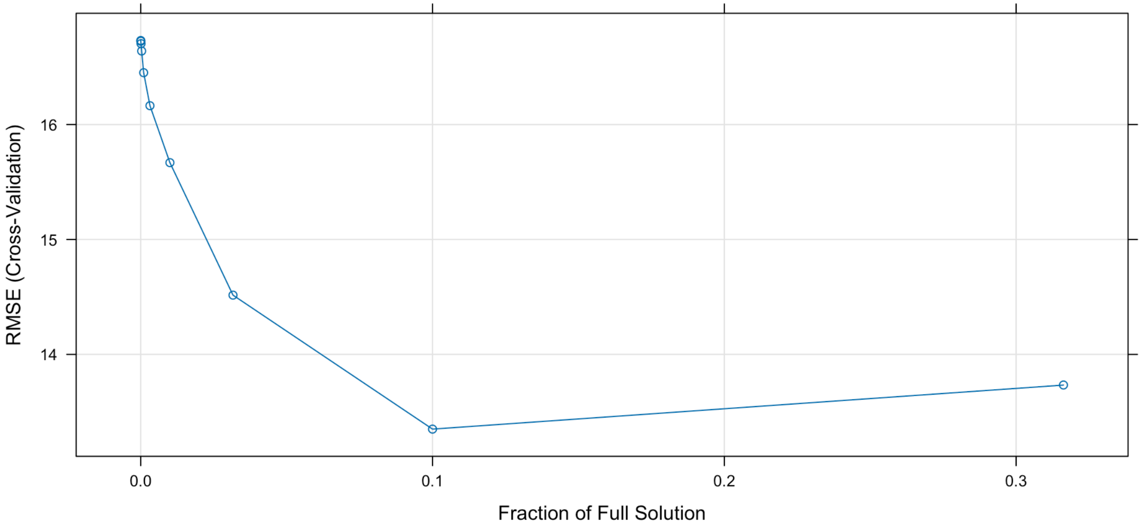


```r
dotplot(train_metrics2)
```

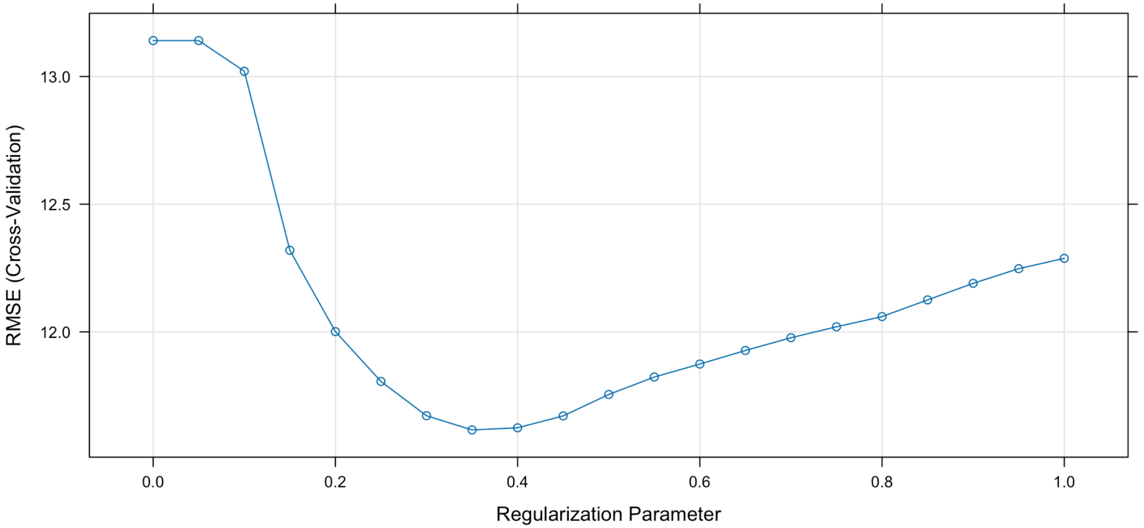


3.2.e - Lasso Issue

```r
# Lasso Regression
set.seed(seed)
lambda_grid <- data.frame(.fraction = 10^seq(-5, -.5, 
fp_lasso_model <- train(train_fingerprints_trans, trai
                       method = "lasso",
                       tuneGrid =  lambda_grid,
                       trControl = trainControl(method =
)
plot(fp_lasso_model)
```




```r
# alternate Lasso
lasso_grid <-  expand.grid(alpha = 1, lambda = seq(0,
fp_lasso_model2 <- train(train_fingerprints_trans, tra
                       method = "glmnet",
                       tuneGrid =  lasso_grid,
                       trControl = trainControl(method =
)
plot(fp_lasso_model2)
```



Assignment 4 Tips

- Use the A4-M1.qmd template (whether you use posit or not)
- Plot your model (hyperparameters) (page limit extended to 21)
- posit.cloud solutions should solve quickly (< 90s for entire notebook)

Q&A

Is there a difference between using `preProcess` within `train()` versus manually transforming the data?

```r
# Apply pre-processing with `caret::train()` (repeated
# Split the data into training and validation sets
set.seed(seed)  # for reproducibility
train_index <- createDataPartition(fat_content, p = 0.
train_data <- absorbance[train_index, ]
train_fat <- fat_content[train_index]
test_data <- absorbance[-train_index, ]
test_fat <- fat_content[-train_index]

#CV
train_cv <- trainControl(method = "cv")

# Partial Least Squares
set.seed(seed);
pls_model_auto <- train(train_data, train_fat,
                  method = "pls",
                  tuneLength = 40,
                  preProcess = c("center", "scale"),
                  trControl = train_cv
                  )
```

```r
# Apply pre-processing manually
absorb_prep <- preProcess(absorbance, method = c("cent
absorb_trans <- predict(absorb_prep, newdata = absorba
train_data_trans <- absorb_trans[train_index, ]
test_data_trans <- absorb_trans[-train_index, ]

# Partial Least Squares
set.seed(seed);
pls_model_manual <- train(train_data_trans, train_fat,
                          method = "pls",
                          tuneLength = 40,
                          preProcess = c("center", "sc
                          trControl = train_cv
)
```

```r
# Compare the outputs.
predictions <- tibble(
    auto = predict(pls_model_auto, newdata = test_data
    manual = predict(pls_model_manual, newdata = test_
) |>
    mutate(abs_error = abs(manual - auto))
sum(predictions$abs_error)
```

```
[1] 6.375721e-08
```

No difference. In summary, `caret` knows to apply the same transformations automatically in the `predict()` step, ensuring that the data is processed consistently throughout the model lifecycle.