Module 5

> In module 5, we learned about tuning hyperparameters of models. In this homework, we are using L1/L2 penalties and dimensionality reduction using Principal Component Analysis (PCA) and Partial Least Squares (PLS) to control the flexibility of our models and reduce the risk of overfitting. We will apply these techniques to build classification and regression models. Both problems 1 and 2 use datasets from previous assignments. You can reuse the preprocessing steps from these assignments.
>
> You can download the R Markdown file (https://gedeck.github.io/DS-6030/homework/Module-5.Rmd) and use it to answer the following questions.
>
> If not otherwise stated, use Tidyverse and Tidymodels for the assignments.
>
> As you will find out, the knitting times for the assignment will get longer as you add more code. To speed up the knitting process, use caching and parallel processing. You can find more information about caching here and about parallel processing here.

Load packages

```
library(tidymodels)
library(tidyverse)
library(patchwork)
library(glmnet)
```

Parallel processing for faster calculations.

```
library(doParallel)
cl <- makePSOCKcluster(4)#parallel::detectCores(logical = FALSE))
registerDoParallel(cl)
```

1. Build elasticnet model for predicting airfare prices (L1/L2 regularization)

Elasticnet just means you are tuning both penalty and mixture

The `Airfares.csv.gz` dataset was already used in problem 1 of module 2. In that assignment, we built a model to predict the price of an airline ticket `FARE` using a linear regression model. In this assignment, we will build a model to predict the price of an airline ticket `FARE` using a linear regression model with both L1 and L2 regularization (tuning parameters `mixture` and `penalty`).

Load the data from `https://gedeck.github.io/DS-6030/datasets/homework/Airfares.csv.gz`

(a.) Load and preprocess the data. Reuse the preprocessing steps you developed in module 2.

```
airfare_raw<-read.csv("Airfares.csv",na = c("*"))
airfare<-airfare_raw %>%
  mutate(S_CODE=as.factor(S_CODE),
         S_CITY=as.factor(S_CITY),
         E_CODE=as.factor(E_CODE),
         E_CITY=as.factor(E_CITY),
         NEW=as.factor(NEW),
         VACATION=as.factor(VACATION),
         SW=as.factor(SW),
         SLOT=as.factor(SLOT),
         GATE=as.factor(GATE))
#airfare_raw %>% glimpse()
```

(b.) Split the data into a training (75%) and test set (25%). Prepare the resamples for 10-fold cross-validation using the training set.

```
set.seed(1)  # for reproducibility
airfare_split <- initial_split(airfare, prop=0.75, strata=FARE)
airfare_train <- training(airfare_split)
airfare_test <- testing(airfare_split)

resamples<-vfold_cv(airfare_train,v=10,strata=FARE)
cv_control <- control_resamples(save_pred=TRUE)
custom_metrics<-metric_set(rmse,mae,rsq)
```

(c.) Define workflow and tuneable parameters

```
formula <- FARE ~ COUPON + NEW + VACATION + SW + HI + S_INCOME +E_INCOME + S_POP + E_POP +
                  SLOT + GATE + DISTANCE + PAX

airfare_recipe <- recipe(formula, data=airfare_train) %>%
    step_normalize(all_numeric_predictors()) %>%
    step_dummy(all_factor_predictors())

#Engine choice: glmnet allows tuning with penalty and mixture
tune_lm_spec <- linear_reg(engine="glmnet", mode="regression",
                      penalty=tune(),mixture=tune())

tune_lm_wf<-workflow() %>%
  add_recipe(airfare_recipe) %>%
  add_model(tune_lm_spec)
```

(d.) Tune the model with 10-fold cross-validation using Bayesian hyperparameter optimization. Make sure that your search space covers a suitable range of values. (see DS-6030: Bayesian Hyperparameter optimization)

```
lm_params <- extract_parameter_set_dials(tune_lm_wf)

lm_bayes<-tune_bayes(tune_lm_wf,resamples=resamples, param_info=lm_params,iter=25)
```

```
## ! No improvement for 10 iterations; returning current results.
```

```
autoplot(lm_bayes)
```

In figure one we can see the affect of different values of mixture and penalty on the model performance and it's metrics. The left half of this figure shows the values of metrics across different values of the penalty. This shows the how different amounts penalty we apply affect the model. Most values are relatively the same with the best rmse (the lowest value) around -2.4 to -1 and the best R squared (the highest value) also at about -2.4 to -1. The right half shows the values of the metrics across different values of the mixture. We see a general pattern of rmse getting smaller from 0 to 1 and rsq getting larger from 0 to 1, with the exception of about 0.75 we see that both metrics hit extreme values. The best rmse value occurs close to 0.77 and the best rsq value occurs also at about 0.77. This indicates that of the lasso and ridge penalties that lasso is playing a larger role. Lasso penalizes the sum of squared coefficients.

(e.) Train a final model using the best parameter set.

```
airfare_final<-finalize_workflow(tune_lm_wf, select_best(lm_bayes,metric='rmse')) %>%
                                 fit(airfare_train)
```

(f.) Predict the `FARE` for the test set and calculate the performance metrics on the test set.

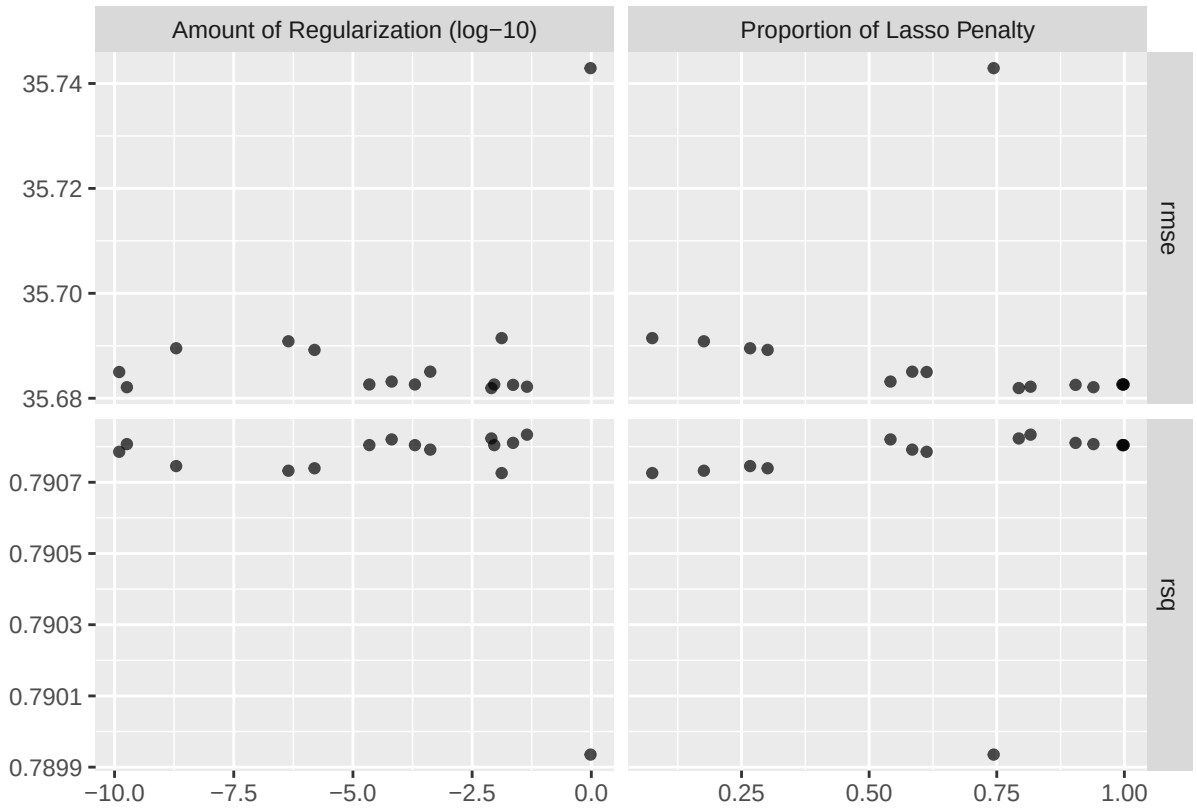

Figure 1: Tuning parameters affect on model metrics

```
#airfare_predictions<-airfare_final %>%
#                            predict(airfare_test)
```

```
metrics(augment(airfare_final, airfare_test), truth=FARE, estimate=.pred) %>%
  knitr::kable(digits=3) %>%
  kableExtra::kable_styling(full_width=FALSE)
```

.metric	.estimator	.estimate
rmse	standard	35.836
rsq	standard	0.760
mae	standard	27.721

2. NASA: Asteroid classification - classification with dimensionality reduction

The dataset `nasa.csv` contains information about asteroids and if they are considered to be hazardous or not.

(a.) Load the data from https://gedeck.github.io/DS-6030/datasets/nasa.csv and preprocess the data. You can find the necessary preprocessing steps in module 3.

```
remove_columns <- c('Name', 'Est Dia in M(min)',
    'Semi Major Axis', 'Jupiter Tisserand Invariant',
    'Epoch Osculation', 'Mean Motion', 'Aphelion Dist',
    'Equinox', 'Orbiting Body', 'Orbit Determination Date',
    'Close Approach Date', 'Epoch Date Close Approach',
    'Miss Dist.(Astronomical)', 'Miles per hour')
```

```
asteroids <- read_csv("https://gedeck.github.io/DS-6030/datasets/nasa.csv",
                      show_col_types = FALSE) %>%
    select(-all_of(remove_columns)) %>%
    select(-contains("Relative Velocity")) %>%
    select(-contains("Est Dia in KM")) %>%
    select(-contains("Est Dia in Feet")) %>%
    select(-contains("Est Dia in Miles")) %>%
    select(-contains("Miss Dist.(lunar)")) %>%
    select(-contains("Miss Dist.(kilometers)")) %>%
    select(-contains("Miss Dist.(miles)")) %>%
    distinct() %>%
    mutate(Hazardous = base::as.factor(Hazardous))
dim(asteroids)
```

```
## [1] 3692   15
```

(b.) Split the dataset into a training and test set. Use 80% of the data for training and 20% for testing. Use stratified sampling to ensure that the training and test set have the same proportion of hazardous asteroids.

```
set.seed(11)  # for reproducibility
asteroids_split <- initial_split(asteroids, prop=0.80, strata=Hazardous)
asteroids_train <- training(asteroids_split)
asteroids_test <- testing(asteroids_split)

asteroids_resamples<-vfold_cv(asteroids_train,v=10,strata=Hazardous)
asteroids_cv_control <- control_resamples(save_pred=TRUE)
```

(c.) Build a logistic regression classification model using principal components (`step_pca`) as predictors. Use cross-validation to determine the optimal number of components. (see DS-6030: Specifying tunable parameters)

(i.) Use `step_normalize` and `step_pca` to preprocess the data in a recipe.

```
asteroids_formula <- Hazardous ~ `Neo Reference ID` + `Absolute Magnitude` + `Est Dia in M(max)` +
                                 `Orbit ID` + `Orbit Uncertainity` + `Minimum Orbit Intersection` +
                                 Eccentricity + Inclination + `Asc Node Longitude` +
                                 `Orbital Period` + `Perihelion Distance` + `Perihelion Arg` +
                                 `Perihelion Time` + `Mean Anomaly`

#Can use glm, as we are not using tuning parameters, such as penalty and mixture
asteroids_spec<-logistic_reg(engine="glm", mode="classification")

asteroids_recipe <- recipe(asteroids_formula, data=asteroids_train) %>%
    step_normalize(all_numeric_predictors()) %>%
    step_pca(all_numeric_predictors(),num_comp=tune())

asteroids_wf<-workflow() %>%
  add_recipe(asteroids_recipe) %>%
  add_model(asteroids_spec)
```

(ii.) Use the `tune` function to find the best number of components (`num_comp`) in the range 1 to 14 using AUC as the selection criterium.

```
asteroids_params<-extract_parameter_set_dials(asteroids_wf) %>%
  update(num_comp=num_comp(range=c(1,14)))
asteroids_tune_results<-tune_grid(asteroids_wf, resamples=asteroids_resamples,grid=grid_regular(asteroi
```

(iii.) Use the `autoplot` function on the cross-validation results to visualize the results. Describe your observations.

```
autoplot(asteroids_tune_results)
```

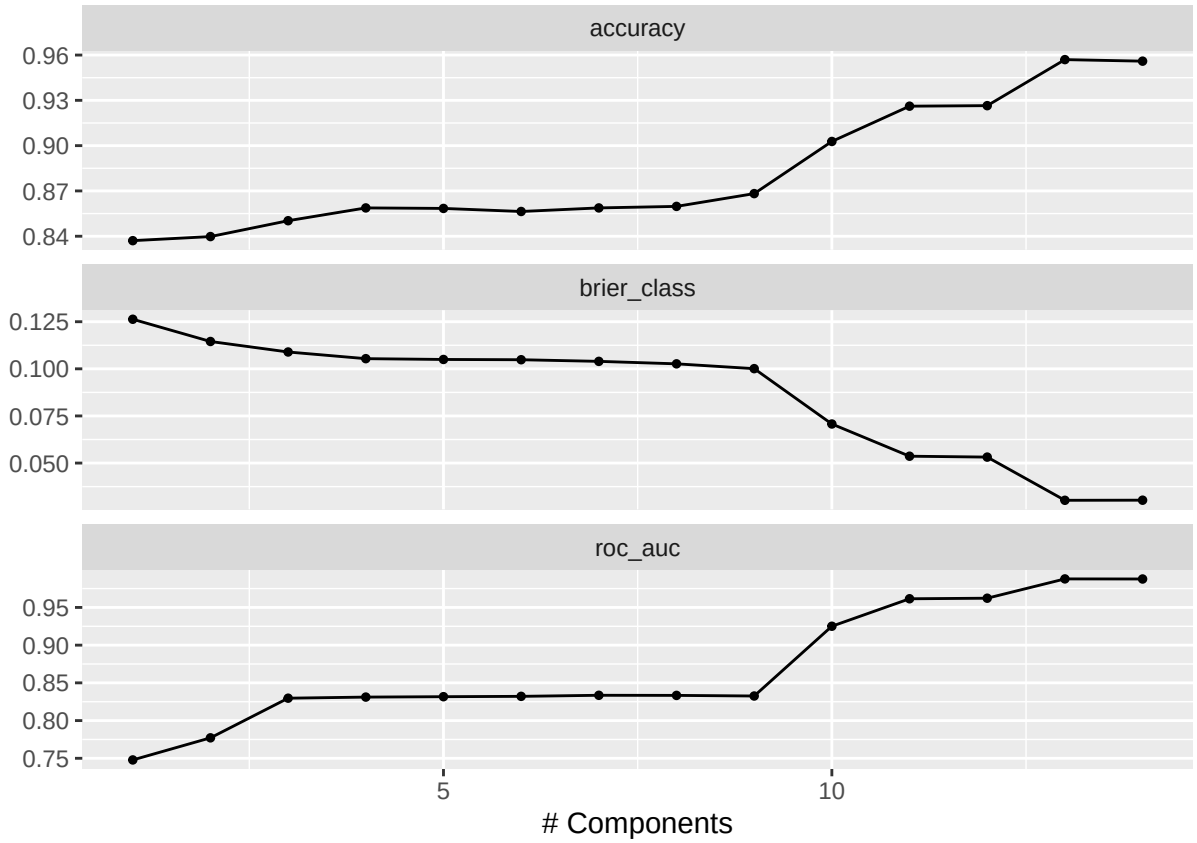

Figure 2: Principal Component Cross Validation Results on Asteroids Dataset

In figure two we see that generally as the number of principal components increases the model performs better. We see an increase from zero to 3 principal components, then performance levels out, maintaining about the same performance from 3 principal components until 9 principal components. There is another increase from 9 to 13 principal components, and then it levels off again from 13 to 14 principal components. The best performance is at 13 principal components, as it results in the highest accuracy and roc auc and the lowest brier score.

(iv.) Report the best number of components and the associated regression metrics.

```
asteroids_best_params<-show_best(asteroids_tune_results, metric='roc_auc', n=1)
asteroids_best_params %>%
    knitr::kable(digits=3) %>%
    kableExtra::kable_styling(full_width=FALSE)
```

num_comp	.metric	.estimator	mean	n	std_err	.config
13	roc_auc	binary	0.988	10	0.002	Preprocessor13_Model1

(v.) Using the best parameter set, train a final model using the full training set and determine the performance metrics on the test set.

```
tuned_asteroids_model <- asteroids_wf %>%
    finalize_workflow(select_best(asteroids_tune_results, metric="roc_auc")) %>%
    fit(asteroids_train)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
metrics(augment(tuned_asteroids_model, asteroids_test)
        , truth=Hazardous, estimate=.pred_class) %>%
        knitr::kable(digits=3) %>%
        kableExtra::kable_styling(full_width=FALSE)
```

.metric	.estimator	.estimate
accuracy	binary	0.958
kap	binary	0.845

(d.) Repeat (c) using the PLS method (`step_pls`) as predictors. (see DS-6030: Partial least squares regression)

(i.) Use `step_normalize` and `step_pls` to preprocess the data in a recipe.

```
library(recipes)
library(embed)
library(BiocManager)
#BiocManager::mixOmics
#Message for this code block: Show in New Window
#1 package (mixOmics) is needed for this step but is not installed.
#To install run: install.packages("mixOmics") - couldn't install on this version of R - it has been rem

#Utilizing asteroids_formula and asteroids_spec previously defined
asteroids_recipe_pls <- recipe(asteroids_formula, data=asteroids_train) %>%
    step_normalize(all_numeric_predictors()) %>%
    step_pls(all_numeric_predictors(),outcome="Hazardous",num_comp=tune())


asteroids_wf_pls<-workflow() %>%
  add_recipe(asteroids_recipe_pls) %>%
  add_model(asteroids_spec)
```

(ii) Use the `tune` function to find the best number of components (`num_comp`) in the range 1 to 14 using AUC as the selection criterium.

```
#Utilizing asteroids_resamples previously defined

asteroids_params_pls<-extract_parameter_set_dials(asteroids_wf_pls) %>%
  update(num_comp=num_comp(range=c(1,14)))
asteroids_tune_results_pls<-tune_grid(asteroids_wf_pls,
                                      resamples = asteroids_resamples,
                                      grid=grid_regular(asteroids_params_pls,levels=14))
```

(iii.) Use the `autoplot` function on the cross-validation results to visualize the results. Describe your observations.

```
autoplot(asteroids_tune_results_pls)
```

In figure three we see there is quick improvement in the performance from zero to five components and then a significant slowing in improvement, but still gradual from five to twelve and then some

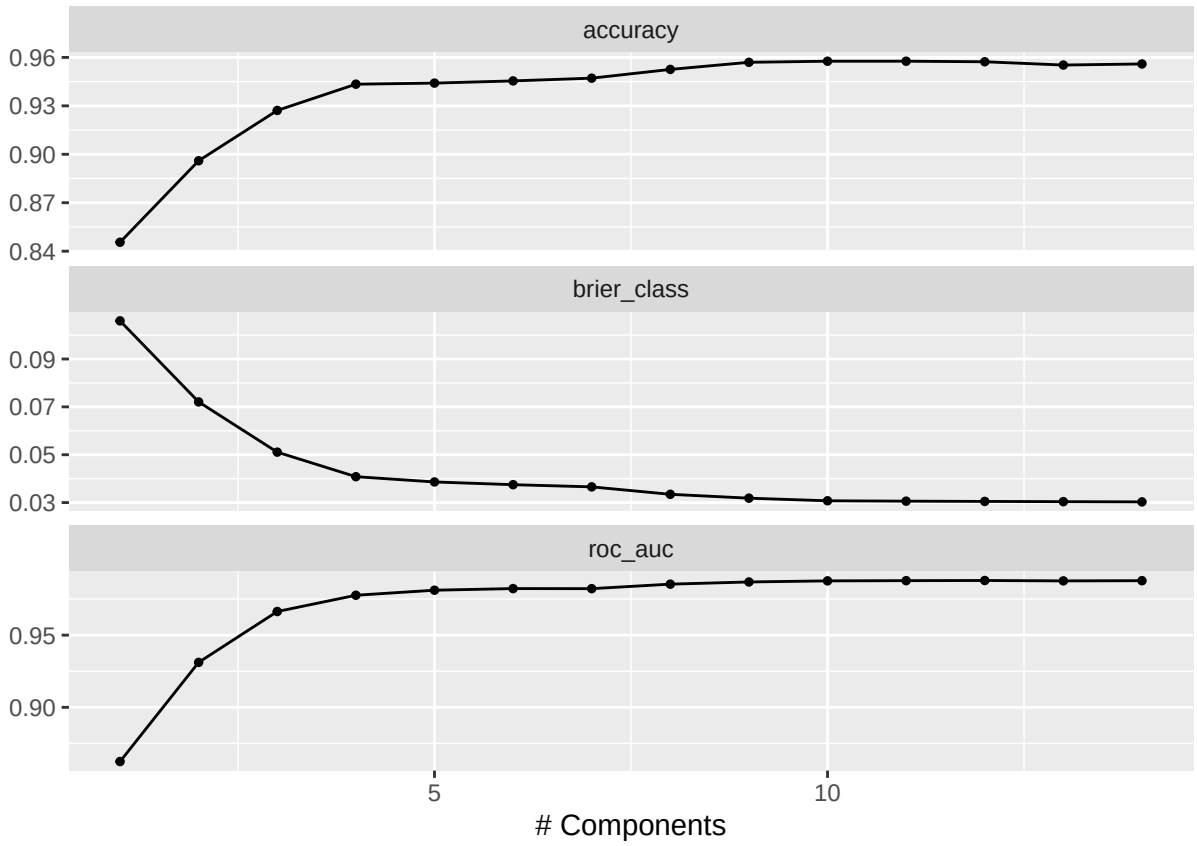

Figure 3: Partial Least Squares Cross Validation Results on Asteroids Dataset

leveling off and a bit of a decrease in accuracy at 13 components. The best performance occurs at 12 components.

(iv.) Report the best number of components and the associated regression metrics.

```
asteroids_best_params_pls<-show_best(asteroids_tune_results_pls, metric='roc_auc', n=1)
asteroids_best_params_pls %>%
  knitr::kable(digits=3) %>%
  kableExtra::kable_styling(full_width=FALSE)
```

num_comp	.metric	.estimator	mean	n	std_err	.config
12	roc_auc	binary	0.988	10	0.002	Preprocessor12_Model1

(v.) Using the best parameter set, train a final model using the full training set and determine the performance metrics on the test set.

```
tuned_asteroids_model_pls <- asteroids_wf_pls %>%
    finalize_workflow(select_best(asteroids_tune_results_pls, metric="roc_auc")) %>%
    fit(asteroids_train)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
metrics(augment(tuned_asteroids_model_pls, asteroids_test),
        truth=Hazardous, estimate=.pred_class) %>%
```

```
        knitr::kable (digits=3) %>%
        kableExtra::kable_styling(full_width=FALSE)
```

.metric	.estimator	.estimate
accuracy	binary	0.955
kap	binary	0.834

(e.) Compare the tuning results in (c) and (d) and comment on the differences. What do you think is going on? Could you reduce the number of components further from what is suggested by CV?

```
bind_rows(
    metrics(augment(tuned_asteroids_model, asteroids_test), truth=Hazardous, estimate=.pred_class) %>%
      mutate(model="PCA"),
    metrics(augment(tuned_asteroids_model_pls, asteroids_test), truth=Hazardous, estimate=.pred_class) %
      mutate(model="PLS"),
) %>%
ggplot(aes(x=.estimate, y=model)) +
    geom_point() +
    facet_wrap(~ .metric)
```

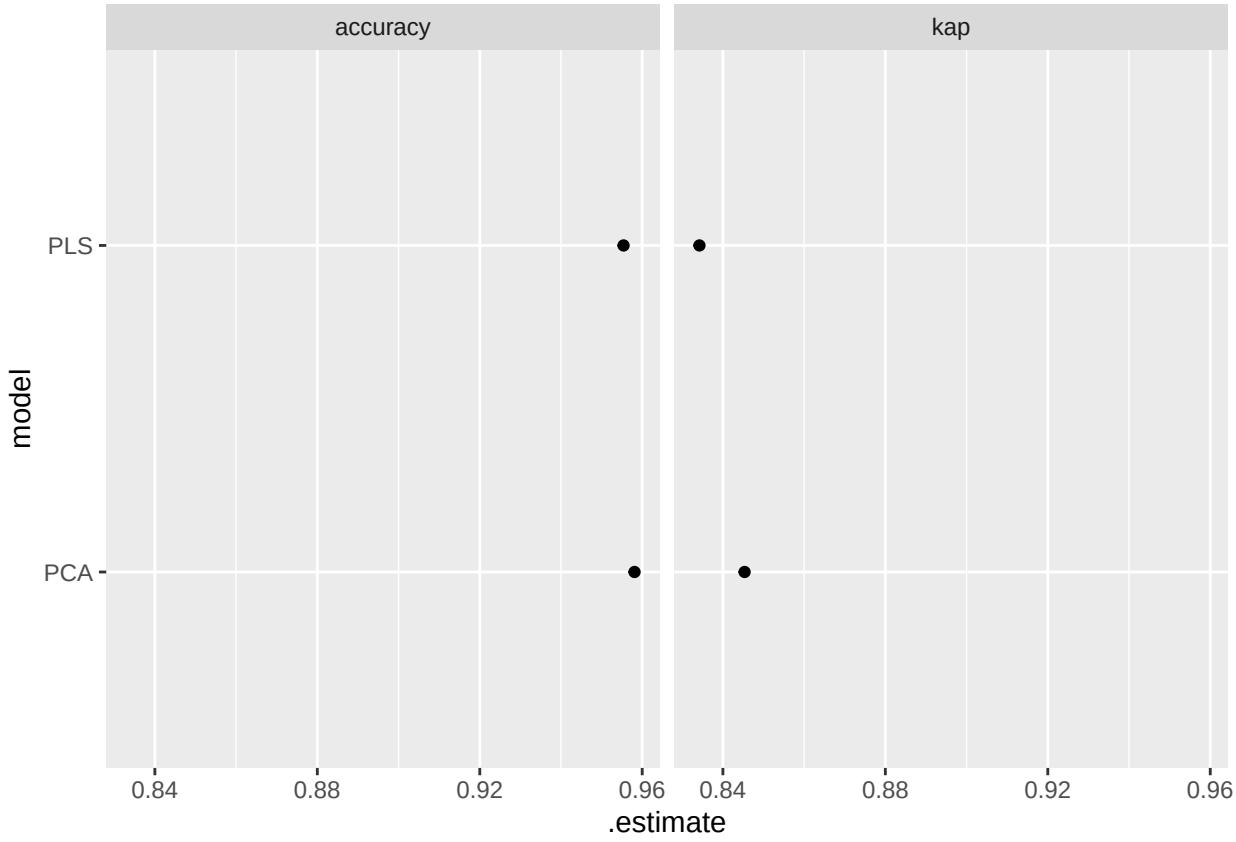

Figure 4: Metric Comparison Between PCA and PLS

In figure four we can see that PCA has a slightly higher accuracy than PLS and that PCA has a higher kappa than PLS, this indicates that with the best tuning parameters on this model that PCA has better predictive capabilities.

When looking at figure two, the number of components for principal component analysis, I think that the improvement in performance justifies the increase in the number of components. However when looking at figure three, the number of components for partial least squares, I do not think that there is enough of an increase in performance to justify the increase in components. At about 5 components the improvement slows drastically. And if someone wanted to find a middle ground at about 8 components the increase in performance essentially stops. The improvement in model simplicity very well could justify utilizing 5 or 8 components over the 13 that the cross validation has chosen as the best parameter. In order to know fully what would be preferred, model simplicity or very good predictive performance, we would need to consult with a subject matter expert.

```
stopCluster(cl)
registerDoSEQ()
```