Module 8 HW

Alanna Hazlett

2024-07-11

Module 8

> Module 8 introduced ensemble models. In this homework, we will build various regression model to predict fish toxicity of chemical compounds. You can download the R Markdown file and use it to answer the following questions. If not otherwise stated, use Tidyverse and Tidymodels for the assignments.
>
> The knitting times for the assignment can be very long without caching and parallel processing. For example, the calculations for problem 1 will take more than 16 minutes without parallel excution. Caching reduces it to 5 minutes. With all results being cached, knitting a document will take only a few seconds. You can find more information about caching here and about parallel processing here.

1. Predicting fish toxicity of chemicals (regression)

The dataset `qsar_fish_toxicity.csv` contains information about toxicity of 908 chemicals to fish. The data was downloaded from the UCI Machine Learning Repository. The dataset contains 7 features and the toxicity of the chemicals. The features are a variety of molecular structure descriptors. The toxicity is measured as the negative logarithm of the concentration that kills 50% of the fish after 96 hours of exposure.

- CIC0: Information indices
- SM1_Dz(Z): 2D matrix-based descriptors
- GATS1i: 2D autocorrelations
- NdsCH: Atom-type counts
- NdssC: Atom-type counts
- MLOGP: Molecular properties
- LC50: Toxicity towards fish (log value)

(a.) Load the data from https://gedeck.github.io/DS-6030/datasets/homework/qsar_fish_toxicity.csv. *Hint*: the dataset has **no** column headers and is fields are separated by a semicolon; read the documentation for `read_delim`.

```
fish<-read_delim(file = "https://gedeck.github.io/DS-6030/datasets/homework/qsar_fish_toxicity.csv",
                 delim = ";", col_names = FALSE, show_col_types = FALSE)
colnames(fish) <- c("CIC0", "SM1_DzZ","GATS1i","NdsCH","NdssC","MLOGP","LC50")
```

(b.) Split the data into training (80%) and test (20%) sets using stratified sampling on `LC50`. Prepare the folds for 10-fold cross validation of all models.

```
set.seed(123)
fish_split <- initial_split(fish, prop=0.8, strata=LC50)
fish_train <- training(fish_split)
fish_holdout <- testing(fish_split)

fish_formula <- LC50 ~ CIC0 + SM1_DzZ + GATS1i + NdsCH + NdssC + MLOGP
```

```
fish_recipe<-recipe(fish_formula, data=fish_train)
fish_resamples <- vfold_cv(fish_train, strata=LC50)
fish_cv_control <- control_resamples(save_pred=TRUE)
fish_metrics <- metric_set(rmse, mae, rsq)
```

(c.) Build the following models using the training set and evaluate them using 10-fold cross validation. For tuned models, use the `autoplot` function to inspect the tuning results and carry out cross-validation with the optimal parameters.

- Linear regression

```
lm_wf <- workflow() %>%
         add_model(linear_reg()) %>%
         add_recipe(fish_recipe)

lm_fit_cv <- lm_wf %>%
    fit_resamples(fish_resamples, control=fish_cv_control,metrics=fish_metrics)

lm_cv_metrics <- collect_metrics(lm_fit_cv)
lm_cv_metrics%>%
    select(.metric,mean,std_err,n) %>%
    knitr::kable(caption="Linear Regression CV Metrics", digits=3) %>%
    kableExtra::kable_styling(full_width=FALSE)
```

Table 1: Linear Regression CV Metrics

.metric	mean	std_err	n
mae	0.705	0.022	10
rmse	0.968	0.040	10
rsq	0.567	0.028	10

- Random forest (tune `min_n` and `mtry`)

```
rf_spec<-rand_forest(mode="regression", mtry=tune(), min_n=tune()) %>%
    set_engine("ranger", importance="impurity")

rf_wf <- workflow() %>%
         add_recipe(fish_recipe) %>%
         add_model(rf_spec)

rf_parameters <- extract_parameter_set_dials(rf_wf) %>%
    update(mtry = mtry(c(1, 6)))
rf_tune <- tune_bayes(rf_wf,resamples=fish_resamples,
                      metrics=fish_metrics,
                      param_info=rf_parameters,
                      iter=25)
```

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

```
best_rf_wf <- rf_wf %>%
              finalize_workflow(select_best(rf_tune, metric="rmse"))

#Returns rmse,mae,rsq  across different values of mtry and min_n
autoplot(rf_tune)
```

```
rf_fit_cv<-best_rf_wf %>%
            fit_resamples(fish_resamples, metrics=fish_metrics, control=fish_cv_control)

rf_cv_metrics <- collect_metrics(rf_fit_cv)
rf_cv_metrics%>%
    select(.metric,mean,std_err,n) %>%
    knitr::kable(caption="Random Forest CV Metrics", digits=3) %>%
    kableExtra::kable_styling(full_width=FALSE)
```

Table 2: Random Forest CV Metrics

.metric	mean	std_err	n
mae	0.636	0.025	10
rmse	0.883	0.045	10
rsq	0.630	0.036	10

```
#show_best(rf_tune, metric='rmse', n=1)
```

- Boosting model (tune `min_n` and `mtry`)

```
#lightgbm engine does not work
boost_spec<-boost_tree(mode="regression", engine="xgboost",
                       min_n=tune(), mtry=tune())

boost_wf <- workflow() %>%
  add_recipe(fish_recipe) %>%
  add_model(boost_spec)

boost_parameters <- extract_parameter_set_dials(boost_wf)%>%
    update(mtry = mtry(c(1, 6)))

boost_tune <- tune_bayes(boost_wf,resamples=fish_resamples,
                         metrics=fish_metrics,
                         param_info=boost_parameters,
                         iter=25)

#rmse,mae,rsq across different values of min_n and mtry
autoplot(boost_tune)
```

```
best_boost_wf <- boost_wf %>%
    finalize_workflow(select_best(boost_tune, metric="rmse"))
boost_fit_cv <- fit_resamples(best_boost_wf, fish_resamples,
                              metrics=fish_metrics, control=fish_cv_control)

boost_cv_metrics<-collect_metrics(boost_fit_cv)

boost_cv_metrics%>%
    select(.metric,mean,std_err,n) %>%
    knitr::kable(caption="Boost CV Metrics", digits=3) %>%
    kableExtra::kable_styling(full_width=FALSE)
```

Table 3: Boost CV Metrics

.metric	mean	std_err	n
mae	0.665	0.020	10
rmse	0.910	0.037	10
rsq	0.612	0.026	10

```
#show_best(boost_tune, metric='rmse', n=1)
```

- k-nearest neighbors (tune neighbors)

```
knn_spec <- nearest_neighbor(engine="kknn", mode="regression",neighbors=tune())
knn_wf <- workflow() %>%
          add_recipe(fish_recipe) %>%
          add_model(knn_spec)
knn_parameters <- extract_parameter_set_dials(knn_wf) %>%
      update(neighbors=neighbors(c(2, 20)))
knn_tune <- tune_bayes(knn_wf,resamples=fish_resamples,
                       metrics=fish_metrics,
                       param_info=knn_parameters,
                       iter=25)

#mae, rmse, rsq across different numbers of nearest neighbors
autoplot(knn_tune)
```

```
best_knn_wf <- knn_wf %>%
    finalize_workflow(select_best(knn_tune, metric="rmse"))

knn_fit_cv <- fit_resamples(best_knn_wf, fish_resamples, metrics=fish_metrics, control=fish_cv_control)

knn_cv_metrics<-collect_metrics(knn_fit_cv)

#n is number of top models we want
#show_best(knn_tune, metric='rmse', n=1)
knn_cv_metrics%>%
    select(.metric,mean,std_err,n) %>%
    knitr::kable(caption="KNN CV Metrics", digits=3) %>%
    kableExtra::kable_styling(full_width=FALSE)
```

Table 4: KNN CV Metrics

.metric	mean	std_err	n
mae	0.652	0.019	10
rmse	0.912	0.036	10
rsq	0.611	0.029	10

(d.) Report the cross-validation metrics (RMSE and r^2). What do you observe? Which model would you pick based on these results

```
#Make table of CV performance metrics
cv_metrics<-bind_rows(
    collect_metrics(lm_fit_cv) %>% mutate(model="Linear Regression"),
    collect_metrics(rf_fit_cv) %>% mutate(model="Random Forest"),
    collect_metrics(boost_fit_cv) %>% mutate(model="Boosting"),
    collect_metrics(knn_fit_cv) %>% mutate(model="KNN"),
```

```
)

cv_metrics%>%
    select(model, .metric, mean) %>%
    pivot_wider(names_from = .metric, values_from = mean) %>%
    knitr::kable(caption="Cross-validation performance metrics", digits=3)
```

Table 5: Cross-validation performance metrics

model	mae	rmse	rsq
Linear Regression	0.705	0.968	0.567
Random Forest	0.636	0.883	0.630
Boosting	0.665	0.910	0.612
KNN	0.652	0.912	0.611

```
ggplot(cv_metrics, aes(x=mean, y=model, xmin=mean-std_err, xmax=mean+std_err)) +
    geom_point() +
    geom_linerange() +
    facet_wrap(~ .metric)+
    labs(title="Cross Validation Metrics Across Models")
```

In Figure 4 we see Linear Regression performed the worst of the models, with highest values of mae and rmse and the lowest value of R squared. Boosting and KNN followed and performed better than Linear Regression. The best performing model according to the cross validation metrics is Random Forest, as it has the lowest mae and rmse and the highest R squared.

(d.) Following cross-validation and tuning, fit a final model with the optimal parameters to the training set.

```
rf_model<-best_rf_wf %>%
          fit(fish_train)
```

(e.) Evaluate it on the test set and report the performance metrics RMSE and MAE.

```
all_metrics<-bind_rows(
        bind_cols(
            model="Random Forest",
            dataset="Train",
            metrics(rf_model %>% augment(fish_train), truth=LC50, estimate=.pred),
        ),
        bind_cols(
            model="Random Forest",
            dataset="Test",
            metrics(rf_model %>% augment(fish_holdout), truth=LC50, estimate=.pred),
        ),
    )

metrics_table <- function(metrics, caption) {
  metrics %>%
      pivot_wider(names_from=.metric, values_from=.estimate) %>%
      select(-.estimator) %>%
      knitr::kable(caption=caption, digits=3) %>%
      kableExtra::kable_styling(full_width=FALSE)
}
```

```
metrics_table(all_metrics,"Random Forest Performance Metrics")
```

Table 6: Random Forest Performance Metrics

model	dataset	rmse	rsq	mae
Random Forest	Train	0.381	0.944	0.274
Random Forest	Test	0.824	0.680	0.587

(f.) Create a visualization that compares the RMSE values for training and test sets of the four models. Do you see a difference between the models? Is there an indication of overfitting for any of the models?

```
lm_model<-lm_wf %>%
              fit(fish_train)

#RF model already defined

boost_model<-best_boost_wf %>%
              fit(fish_train)

knn_model<-best_knn_wf %>%
              fit(fish_train)
```

```
tnt_metrics<-bind_rows(
    fish_metrics(augment(lm_model,fish_train),truth=LC50,estimate=.pred) %>%
                            mutate(model="Linear Regression",dataset="Train"),
    fish_metrics(augment(lm_model,fish_holdout),truth=LC50,estimate=.pred) %>%
                            mutate(model="Linear Regression",dataset="Test"),
    fish_metrics(augment(rf_model,fish_train),truth=LC50,estimate=.pred) %>%
                            mutate(model="Random Forest",dataset="Train"),
    fish_metrics(augment(rf_model,fish_holdout),truth=LC50,estimate=.pred) %>%
                            mutate(model="Random Forest",dataset="Test"),
    fish_metrics(augment(boost_model,fish_train),truth=LC50,estimate=.pred) %>%
                            mutate(model="Boosting",dataset="Train"),
    fish_metrics(augment(boost_model,fish_holdout),truth=LC50,estimate=.pred) %>%
                            mutate(model="Boosting",dataset="Test"),
    fish_metrics(augment(knn_model,fish_train),truth=LC50,estimate=.pred) %>%
                            mutate(model="KNN",dataset="Train"),
    fish_metrics(augment(knn_model,fish_holdout),truth=LC50,estimate=.pred) %>%
                            mutate(model="KNN",dataset="Test"),
)

tnt_metrics%>%
    select(model, dataset, .metric, .estimate) %>%
    pivot_wider(names_from = .metric, values_from = .estimate) %>%
    knitr::kable(caption="Performance metrics", digits=3)
```

Table 7: Performance metrics

model	dataset	rmse	mae	rsq
Linear Regression	Train	0.962	0.698	0.565
Linear Regression	Test	0.887	0.672	0.635
Random Forest	Train	0.381	0.274	0.944

model	dataset	rmse	mae	rsq
Random Forest	Test	0.824	0.587	0.680
Boosting	Train	0.709	0.516	0.767
Boosting	Test	0.884	0.639	0.633
KNN	Train	0.766	0.546	0.731
KNN	Test	0.787	0.556	0.705

```
ggplot(tnt_metrics,aes(x=.estimate, y=model, color=dataset)) +
    geom_point() +
    facet_wrap(~ .metric)+
    labs(title="Train and Test Metrics Across Models")
```

In Table 6 and Figure 5 we can see many differences between the models. Interestingly enough for linear regression the test set performed better than the training set, as the test set has lower values for mae and rmse and a higher value for R squared. All of the other models show the training set performing better than the test set as we would naturally expect. We can see that Random Forest is very much overfitted, as the training set has far better performance than the test set. In the Boosting model we also see somewhat of a difference between the training and test sets, this could potentially also be overfitted. The KNN model almost performs exactly the same for the training and test set (in comparison to the other models) and for the test set it has the best performance of the models with the lowest mae and rmse and the highest R squared.

(g.) Create residual plots from the cross-validated predictions for the four models (add `geom_smooth` line). Combine the four plots in a single figure using the `patchwork` package. What do you observe? Are there differences in the residuals of the four models.

```
residual_plot<-function(cv_fit,model){
  cv_predictions<-collect_predictions(cv_fit)
  residual<-cv_predictions$LC50 - cv_predictions$.pred
  g<-ggplot(cv_predictions,aes(x=cv_predictions$LC50,y=residual))+
             geom_point()+
             geom_smooth(method = 'loess', formula = 'y ~ x')+
             labs(x="LC50",y="Residual",title=model)+
             ylim(-5,6)
  return(g)
}
lm_plot<-residual_plot(lm_fit_cv,"Linear Regression")
rf_plot<-residual_plot(rf_fit_cv,"Random Forest")
boost_plot<-residual_plot(boost_fit_cv,"Boost")
knn_plot<-residual_plot(knn_fit_cv,"KNN")

(lm_plot + rf_plot) / (boost_plot + knn_plot)
```

In Figure 6 we have the residual plots for all of the models. Immediately noticeable is that the plots all share a somewhat similar curve. The areas that you can see differ between the models are in the 0 to -3 residual values and 1.25 to 3.75 for the response (lower left). Linear Regression has multiple values with larger residual values than the other models, with Boost and KNN following, and Random Forest having residuals closest to zero in this region. The other notable area is 1.5 to 6 for residuals and 7.5 to 10 for the response (right side middle to upper portion of graph). The variance of the residuals in this region varies among the models with random forest seeming to have the most variance, with the exception of the one point that Linear Regression has that is near a value of 6.

(g.) For the boosting models, report the variable importance. You can use the `vip` package to create a variable importance plot.

```r
best_boost_wf %>%
  fit(fish_train) %>%
  extract_fit_parsnip() %>%
  vip(num_features = 10)
```

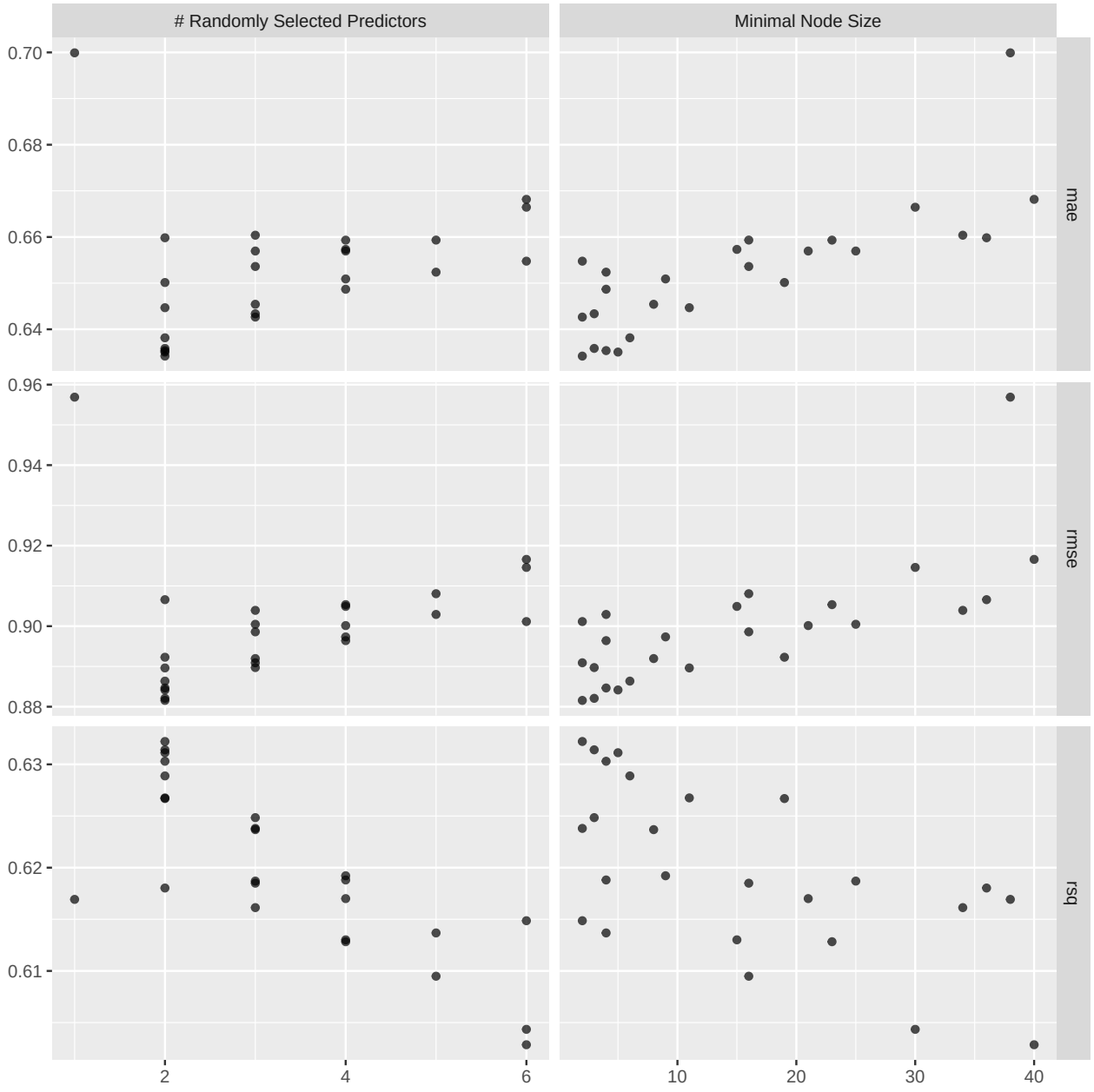

Figure 1: Random Forest Tuning Plot

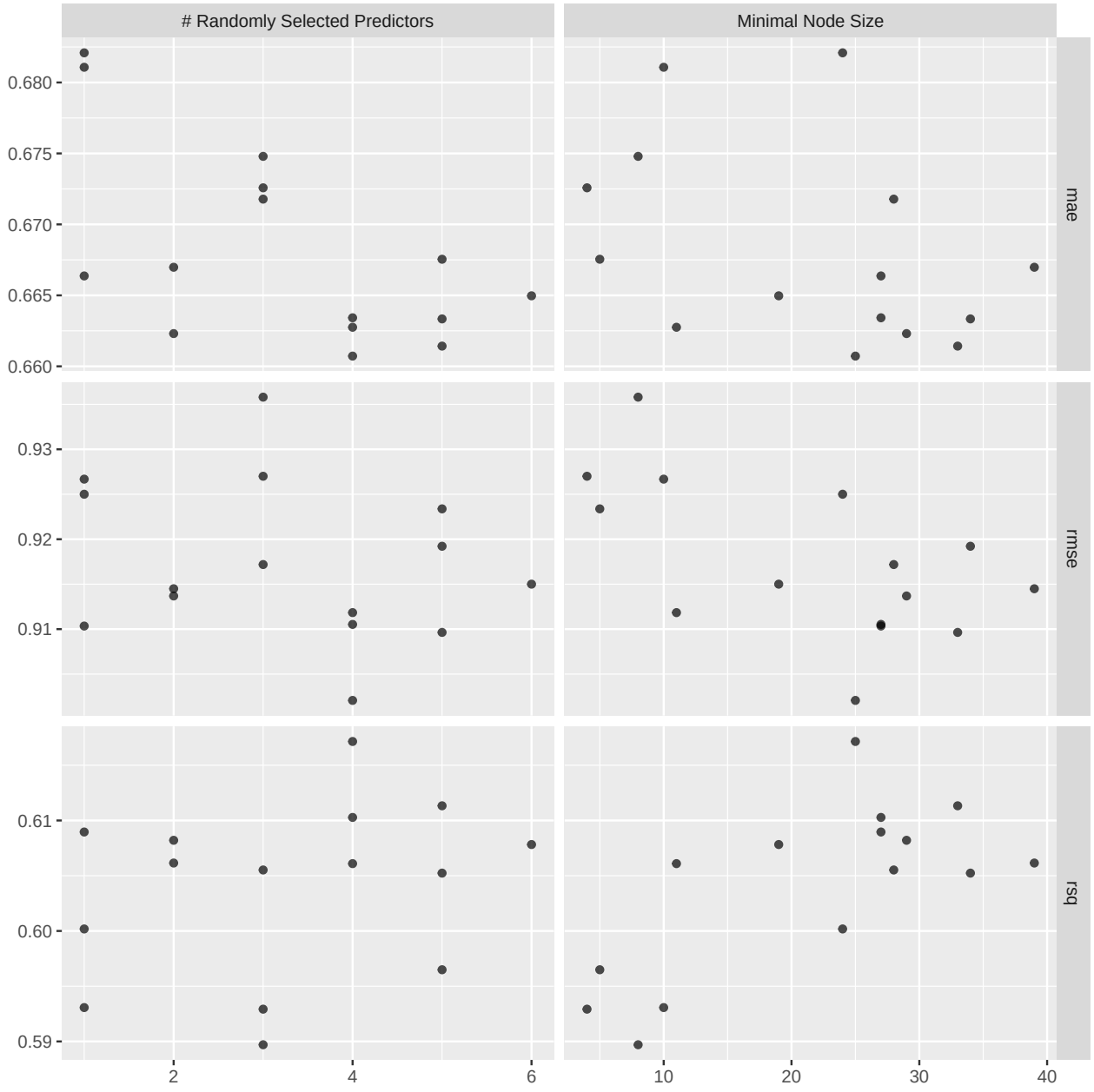

Figure 2: Boosting Tuning Plot

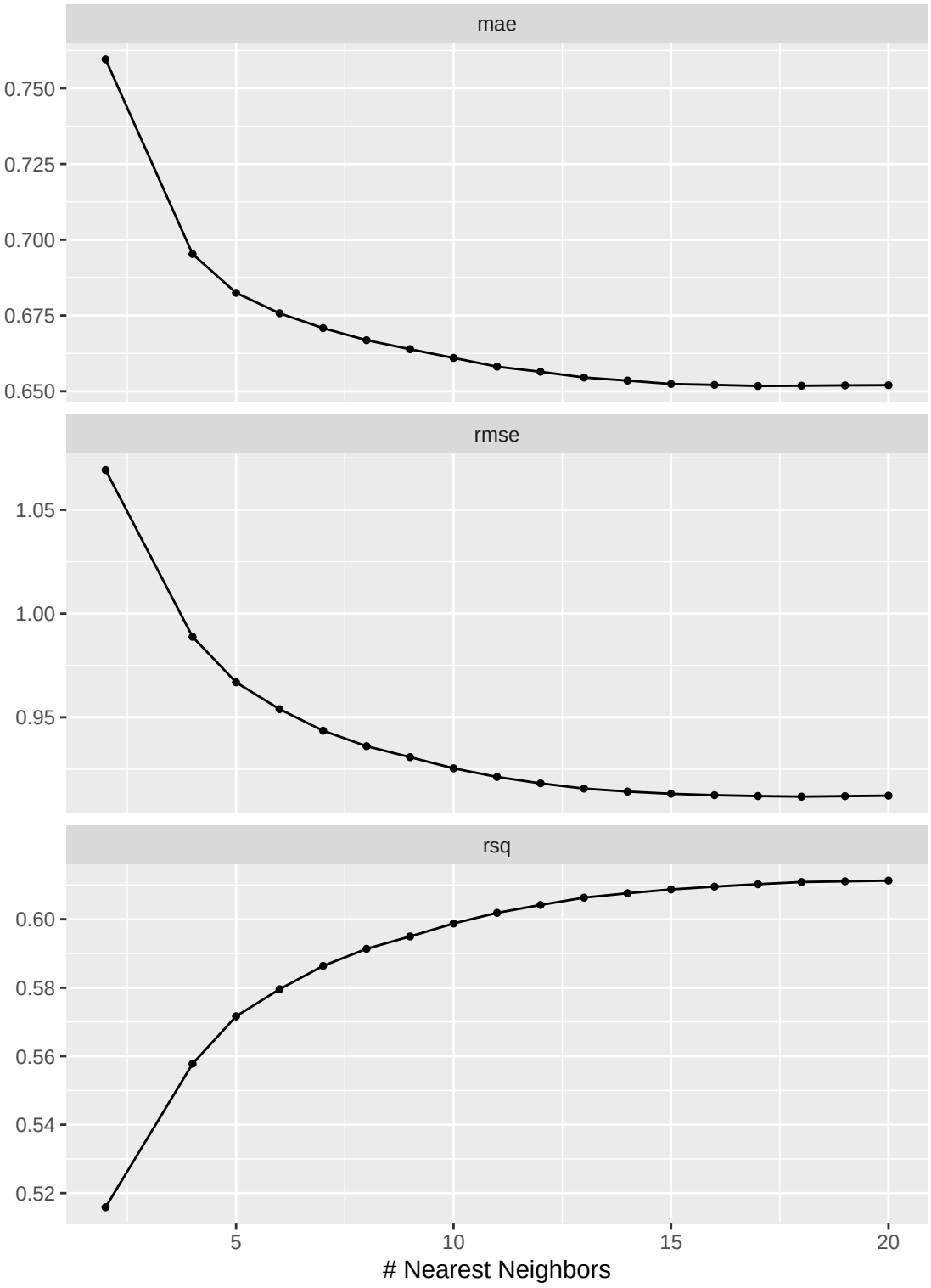

Figure 3: KNN Tuning Plot

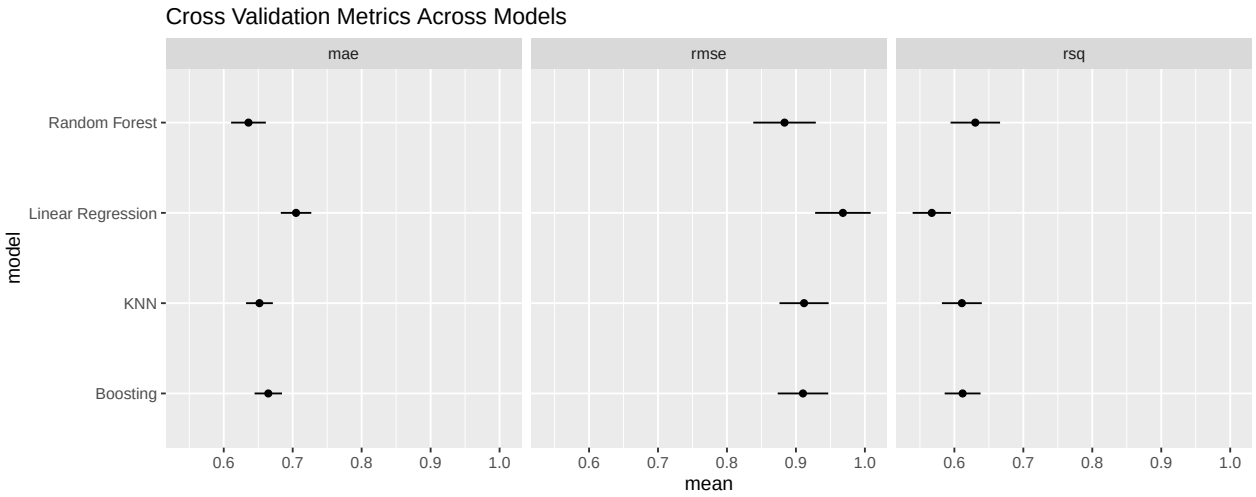

Figure 4: CV Metrics

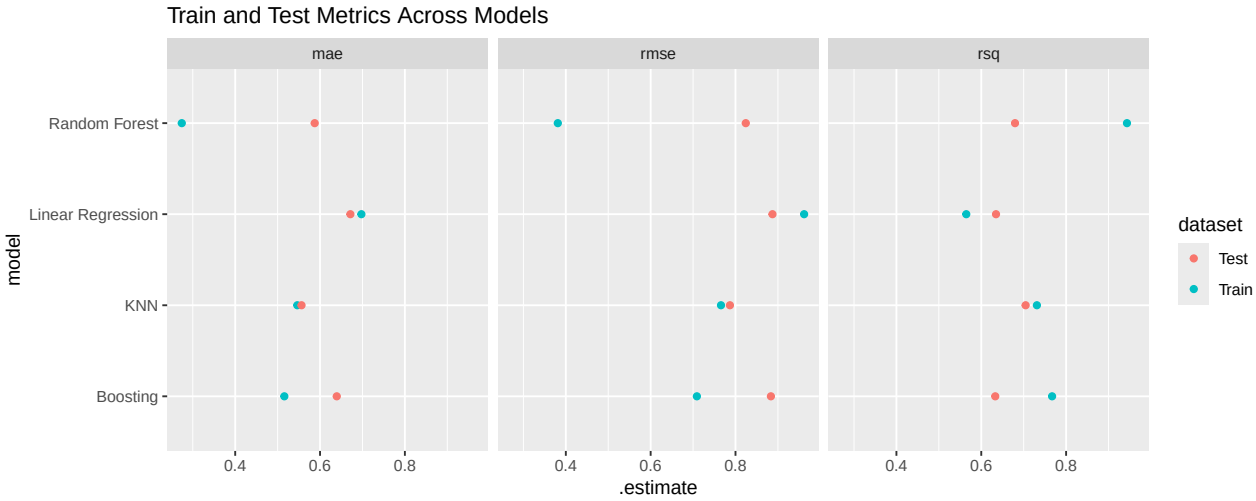

Figure 5: Train and Test Metrics

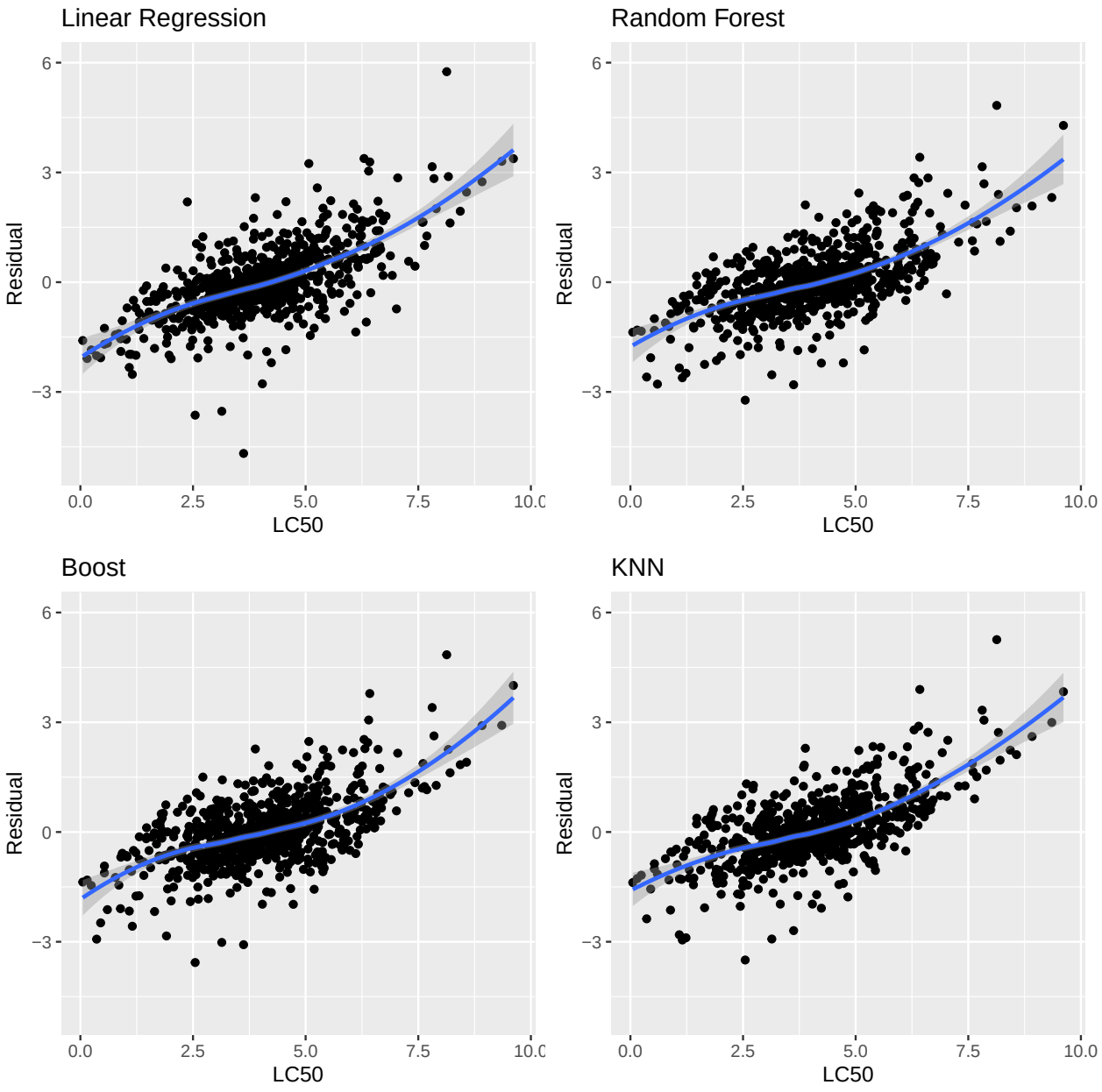

Figure 6: Residual Plots Across Models

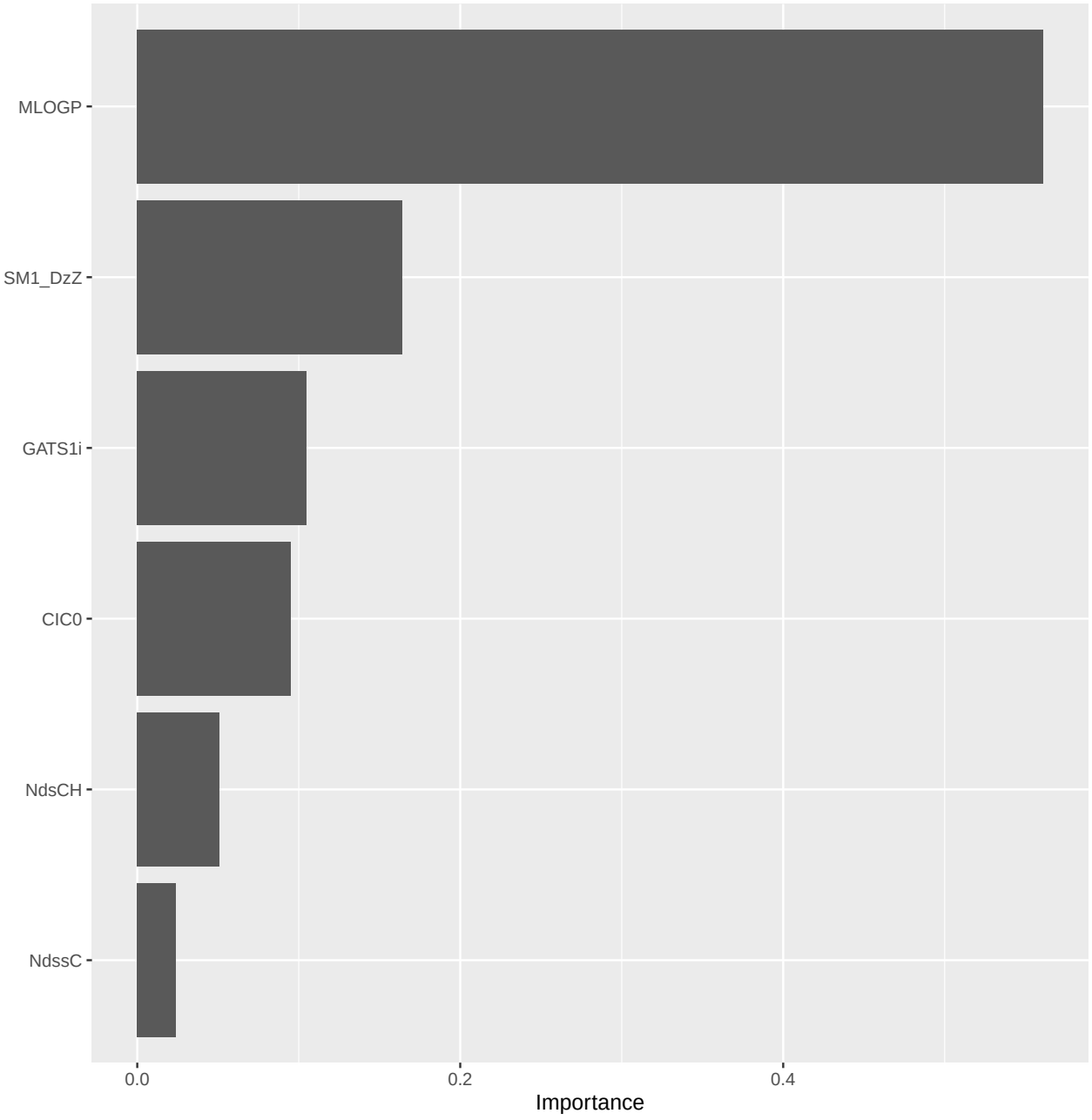

Figure 7: Variable Importance for Boosting Model