Module 4 HW

Alanna Hazlett

2024-06-18

Module 4

> This module introduced cross-validation and bootstrapping as methods to estimate model performance. The homework assignment will give you the opportunity to apply these methods to two datasets. In assignment (1), you will use cross-validation to compare logistic regression, LDA, and QDA classification models. Assignment (2) uses bootstrap to estimate confidence intervals for the mean absolute error and root mean squared error of a linear regression model.
>
> Use *Tidyverse* and *Tidymodels* packages for the assignments.
>
> You can download the R Markdown file (https://gedeck.github.io/DS-6030/homework/Module-4.Rmd) and use it as a basis for your solution.
>
> As you will find out, the knitting times for this assignment will be longer than in previous homework. To speed up the knitting process, use caching and parallel processing. You can find more information about caching and about parallel processing in the course material.

```
library(tidymodels)
library(tidyverse)
library(discrim)
library(patchwork)
library(doParallel)
```

1. Diabetes dataset

The Diabetes Health Indicators Dataset contains healthcare statistics and lifestyle survey information about people in general along with their diagnosis of diabetes. The 35 features consist of some demographics, lab test results, and answers to survey questions for each patient. The target variable for classification is whether a patient has diabetes, is pre-diabetic, or healthy. For this study, the target variable was converted to a binary variable with 1 for diabetes or pre-diabetic and 0 for healthy. Information about the dataset can be found here.

For this exercise use caching and parallel processing to speed up your computations.

Set up parallel processing

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

(a.) Load the diabetes dataset from https://gedeck.github.io/DS-6030/datasets/diabetes/diabetes_binary_5050split_health_indicators_BRFSS2015.csv.gz. Convert the `Diabetes_binary` to a factor with labels *Healthy* and *Diabetes*. Convert all other variables that only contain values of 0 and 1 to factors.

```
# | warnings: FALSE
# | message: FALSE
diabetes<-read_csv("https://gedeck.github.io/DS-6030/datasets/diabetes/diabetes_binary_5050split_health
```

```
                    show_col_types = FALSE)
diabetes<-diabetes %>%
  mutate(Diabetes_binary = factor(ifelse(Diabetes_binary == 0,"Healthy","Diabetes"))) %>%
  mutate(HighBP=factor(HighBP),HighChol=factor(HighChol),CholCheck=factor(CholCheck),
         Smoker=factor(Smoker),Stroke= factor(Stroke),
         HeartDiseaseorAttack= factor(HeartDiseaseorAttack),
         PhysActivity= factor(PhysActivity),
         Fruits= factor(Fruits),Veggies= factor(Veggies),HvyAlcoholConsump= factor(HvyAlcoholConsump),
         AnyHealthcare= factor(AnyHealthcare),NoDocbcCost= factor(NoDocbcCost),
         DiffWalk= factor(DiffWalk),Sex= factor(Sex))
```

(b.) Split the data into a training and test set using a 50/50 split. Use the `set.seed()` function to ensure reproducibility.

```
set.seed(12345)
diabetes_split<-initial_split(diabetes,prop=0.5,strata=Diabetes_binary)
diabetes_train<-training(diabetes_split)
diabetes_test<-testing(diabetes_split)
```

(c.) Build a logistic regression model to predict `Diabetes_binary` using all other variables as predictors. Use the training set to fit the model using 10-fold cross-validation. Report the cross-validation accuracy and ROC-AUC of the model. (see DS-6030: Model validation using cross-validation)

I included the logistic regression accuracy and ROC-AUC in a table with QDA and LDA in part d.

```
formula <- Diabetes_binary ~ HighBP+HighChol+CholCheck+BMI+Smoker+Stroke+HeartDiseaseorAttack+
                             PhysActivity+Fruits+Veggies+HvyAlcoholConsump+AnyHealthcare+NoDocbcCost+
                             GenHlth+MentHlth+PhysHlth+DiffWalk+Sex+Age+Education+Income
diabetes_recipe<-recipe(formula,data=diabetes_train) %>%
  step_normalize(all_numeric_predictors())
logreg_spec<-logistic_reg(mode="classification") %>%
  set_engine('glm')
logreg_wf <-workflow() %>%
  add_recipe(diabetes_recipe) %>%
  add_model(logreg_spec)
resamples <- vfold_cv(diabetes_train, v=10, strata=Diabetes_binary)
model_metrics <- metric_set(roc_auc, accuracy)
cv_control <- control_resamples(save_pred=TRUE)
logreg_cv <- fit_resamples(logreg_wf, resamples, metrics=model_metrics, control=cv_control)
```

(d.) Use the approach from (c) to build LDA and QDA models. Report the cross-validation accuracy and ROC-AUC of each model.

```
lda_spec <- discrim_linear(mode="classification") %>%
    set_engine('MASS')
qda_spec <- discrim_quad(mode="classification") %>%
    set_engine('MASS')
lda_wf <- workflow() %>%
    add_recipe(diabetes_recipe) %>%
    add_model(lda_spec)
qda_wf <- workflow() %>%
    add_recipe(diabetes_recipe) %>%
    add_model(qda_spec)
lda_cv <- fit_resamples(lda_wf, resamples, metrics=model_metrics, control=cv_control)
qda_cv <- fit_resamples(qda_wf, resamples, metrics=model_metrics, control=cv_control)
```

```
#collect_metrics() retrieves the .metrics from the cv,
#which contains splits,id,.metrics,.notes, and .predictions.
#collect_predictions() retrieves the .predictions from the cv.
cv_metrics <- bind_rows(
collect_metrics(logreg_cv) %>% mutate(model="Logistic regression"),
collect_metrics(lda_cv) %>% mutate(model="LDA"),
collect_metrics(qda_cv) %>% mutate(model="QDA")
)
cv_metrics %>%
select(model, .metric, mean) %>%
pivot_wider(names_from=".metric", values_from="mean") %>%
knitr::kable(caption="Cross-validation performance metrics", digits=3)
```

Table 1: Cross-validation performance metrics

model	accuracy	roc_auc
Logistic regression	0.748	0.825
LDA	0.747	0.824
QDA	0.728	0.783

```
ggplot(cv_metrics, aes(x=mean, y=model, xmin=mean-std_err, xmax=mean+std_err)) +
    geom_point() +
    geom_linerange() +
    facet_wrap(~ .metric)
```

In Figure 1 and Table 1, we can see that for the training data the logistic regression model preformed the best in regards to accuracy and AUC. It is quickly followed by LDA and QDA did a bit worse than both.

(e.) Create a plot that compares the ROC curves of the three models from (c) and (d). The ROC curve should be based on the predictions from cross-validation. How do the models compare? Which model would you choose for prediction?

```
roc_cv_plot <- function(model_cv, model_name) {
    cv_predictions <- collect_predictions(model_cv)
    cv_ROC <- cv_predictions %>%
        roc_curve(truth=Diabetes_binary, .pred_Diabetes, event_level="first")
    autoplot(cv_ROC) +
        labs(title=model_name)
}
g1 <- roc_cv_plot(logreg_cv, "Logistic regression")
g2 <- roc_cv_plot(lda_cv, "LDA")
g3 <- roc_cv_plot(qda_cv, "QDA")
g1 + g2 + g3
```

Overlay:

```
roc_cv_data <- function(model_cv) {
    cv_predictions <- collect_predictions(model_cv)
    cv_predictions %>%
        roc_curve(truth=Diabetes_binary, .pred_Diabetes, event_level="first")
}
bind_rows(
    roc_cv_data(logreg_cv) %>% mutate(model="Logistic regression"),
```

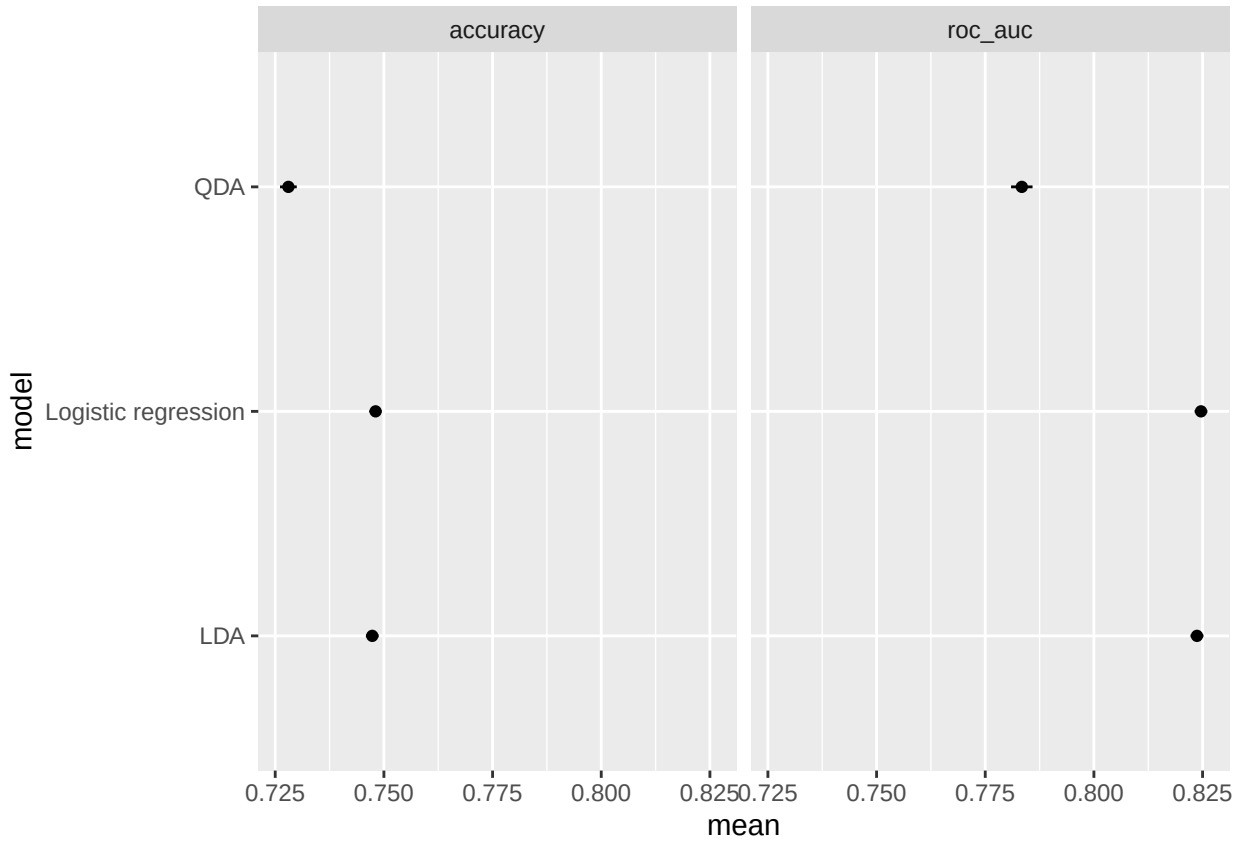


Figure 1: Training Data Metrics

```
    roc_cv_data(lda_cv) %>% mutate(model="LDA"),
    roc_cv_data(qda_cv) %>% mutate(model="QDA")
) %>%
ggplot(aes(x=1-specificity, y=sensitivity, color=model)) +
    geom_line()
```

In Figure 3 with the overlayed ROC curves we can see it is almost indistinguishable what model performed the best, logistic regression or LDA. Both have higher values of sensitivity and specificity than QDA. We know from Table 1 that logistic regression just barely beats out LDA, which means that I would choose logistic regression for predictive purposes. QDA is the clearly the worse out of the three models for the training data.

(f.) Use the three models to calculate metrics on the test set. Report the accuracy and ROC-AUC of each model. How do the models compare?

```
logreg_fit<-logreg_wf %>% fit(diabetes_train)
LDA_fit<-lda_wf %>% fit(diabetes_train)
QDA_fit<-qda_wf %>% fit(diabetes_train)


calculate_metrics <- function(model, train, test, model_name) {
    roc_auc(model %>% augment(train), Diabetes_binary, .pred_Diabetes, event_level="first")
    bind_rows(
        bind_cols(
            model=model_name,
```

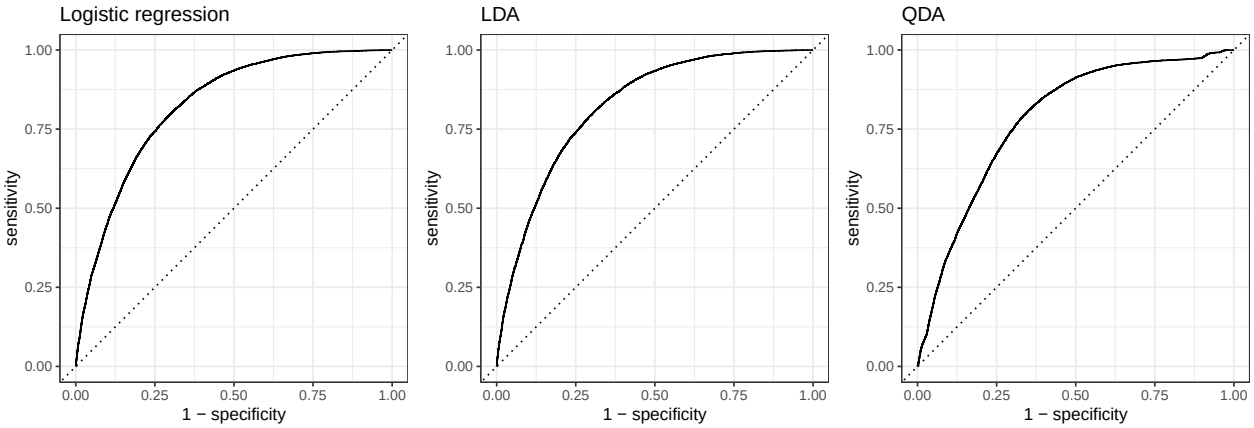


Figure 2: ROC curves based on cross-validation predictions

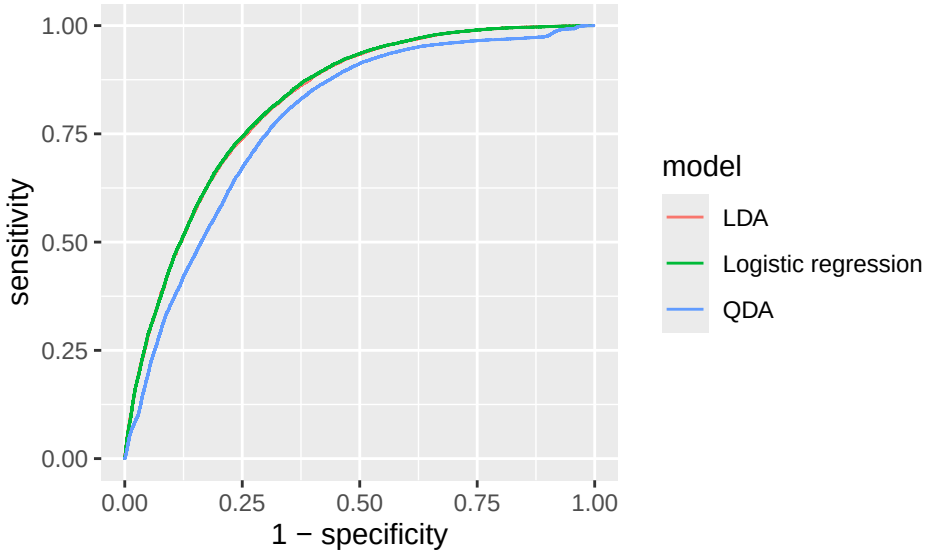


Figure 3: Overlay of cross-validation ROC curves

```
        dataset="train",
        metrics(model %>% augment(train), truth=Diabetes_binary, estimate=.pred_class),
    ),
    bind_cols(
        model=model_name,
        dataset="train",
        roc_auc(model %>% augment(train), Diabetes_binary, .pred_Diabetes, event_level="first"),
    ),
    bind_cols(
        model=model_name,
        dataset="test",
        metrics(model %>% augment(test), truth=Diabetes_binary, estimate=.pred_class),
    ),
    bind_cols(
        model=model_name,
        dataset="test",
```

```
            roc_auc(model %>% augment(test), Diabetes_binary, .pred_Diabetes, event_level="first"),
        ),
    )
}

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(bind_rows(calculate_metrics(logreg_fit,diabetes_train,diabetes_test,"Logistic Regression"),
                        calculate_metrics(LDA_fit,diabetes_train,diabetes_test,"LDA"),
                        calculate_metrics(QDA_fit,diabetes_train,diabetes_test,"QDA")),
              "Metrics for Classification Models")
```

Table 2: Metrics for Classification Models

model	dataset	accuracy	kap	roc_auc
Logistic Regression	train	0.749	0.498	0.825
Logistic Regression	test	0.749	0.497	0.825
LDA	train	0.748	0.495	0.824
LDA	test	0.747	0.495	0.824
QDA	train	0.729	0.458	0.785
QDA	test	0.730	0.460	0.784

For training the order of best to worst based on accuracy and roc auc was logistic, LDA, QDA. For test it is also logistic, then LDA, and then QDA. For all three models we see that the training metrics are relatively similar to the test metrics, indicating that there is not a presence of overfitting. This indicates that logistic regression is the better model to use.

2. Estimate model performance using bootstrap

(a.) Use the `mtcars` dataset from DS-6030: The mtcars dataset to estimate the mean absolute error and root mean squared error of the linear regression model for the prediction of `mpg` using bootstrap. Use 1000 bootstrap samples. Report the mean and standard deviation of the two metrics. (see DS-6030: Model validation using bootstrapping)

```
cars<-mtcars %>%
  mutate(vs = factor(vs),
         am = factor(am))

# Use bootstrap to assess model performance
set.seed(1353)
resamples <- rsample::bootstraps(cars,times=1000)

# define the model
linreg_formula <- mpg ~ cyl + disp + hp + drat + wt + qsec + vs + am + gear + carb
cars_recipe <- recipe(linreg_formula, data=cars) %>%
    step_normalize(all_numeric_predictors())
linreg_spec <- linear_reg(mode="regression") %>%
```

```
      set_engine('lm')
linreg_wf <- workflow() %>%
    add_recipe(cars_recipe) %>%
    add_model(linreg_spec)
custom_metrics<-metric_set(rmse,mae)
linreg_boot <- linreg_wf %>%
    fit_resamples(resamples,control=control_resamples(save_pred=TRUE),metrics=custom_metrics)
```

```
## > A | warning: prediction from rank-deficient fit; consider predict(., rankdeficient="NA")
```

```
## There were issues with some computations   A: x1There were issues with some computations   A: x1
```

```
boot_metrics <- collect_metrics(linreg_boot)
pivot_wider(boot_metrics,names_from=.metric,values_from = mean,std_err)
```

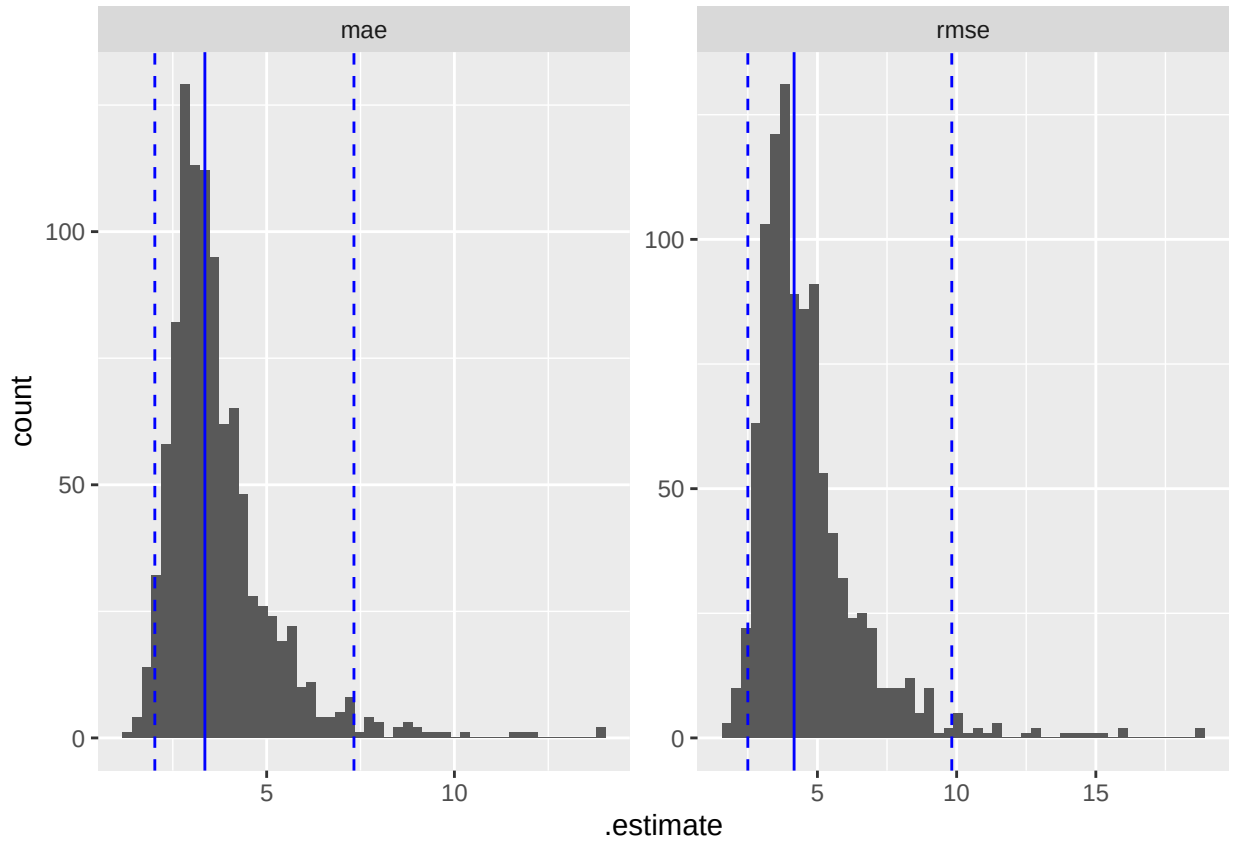
```
## Warning: Specifying the `id_cols` argument by position was deprecated in tidyr 1.3.0.
## i Please explicitly name `id_cols`, like `id_cols = std_err`.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## # A tibble: 2 x 3
##   std_err   mae  rmse
##     <dbl> <dbl> <dbl>
## 1  0.0449  3.69 NA
## 2  0.0621 NA     4.66
```

In Table 3 we see that the mean of the mae is 3.69 with a standard error of 0.04. The mean of rmse was 4.66 with a standard error of 0.06.

(b.) Create a plot of the distribution of the performance metrics. Comment on the shape of the distribution.

```
quantiles <- linreg_boot %>%
    collect_metrics(summarize=FALSE) %>%
    group_by(.metric) %>%
    summarize(
        q0.025 = quantile(.estimate, 0.025),
        median = quantile(.estimate, 0.5),
        q0.975 = quantile(.estimate, 0.975)
    )
linreg_boot %>%
    collect_metrics(summarize=FALSE) %>%
ggplot(aes(x=.estimate)) +
    geom_histogram(bins=50) +
    facet_wrap(~.metric, scales="free") +
    geom_vline(data=quantiles, aes(xintercept=median), color="blue") +
    geom_vline(data=quantiles, aes(xintercept=q0.025), color="blue", linetype="dashed") +
    geom_vline(data=quantiles, aes(xintercept=q0.975), color="blue", linetype="dashed")
```



Both metrics are relatively similar in distribution. They are right skewed (the tail is on the right side). RMSE has a slightly higher mean than MAE. The range is relatively similar for both metrics, although MAE ends at about 14 and RMSE ends at about 17.

(c.) Use the performance metrics calculated for the bootstrap samples to estimate the 95% confidence interval for the mean absolute error and root mean squared error. Report the confidence intervals.

```
quantiles
```

```
## # A tibble: 2 x 4
##   .metric q0.025 median q0.975
##   <chr>    <dbl>  <dbl>  <dbl>
## 1 mae       2.02   3.35   7.32
## 2 rmse      2.50   4.16   9.83
```

We can be 95% confident that the true values of MAE and RMSE lie within these intervals.

Stop cluster

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