Homework 2: Classification

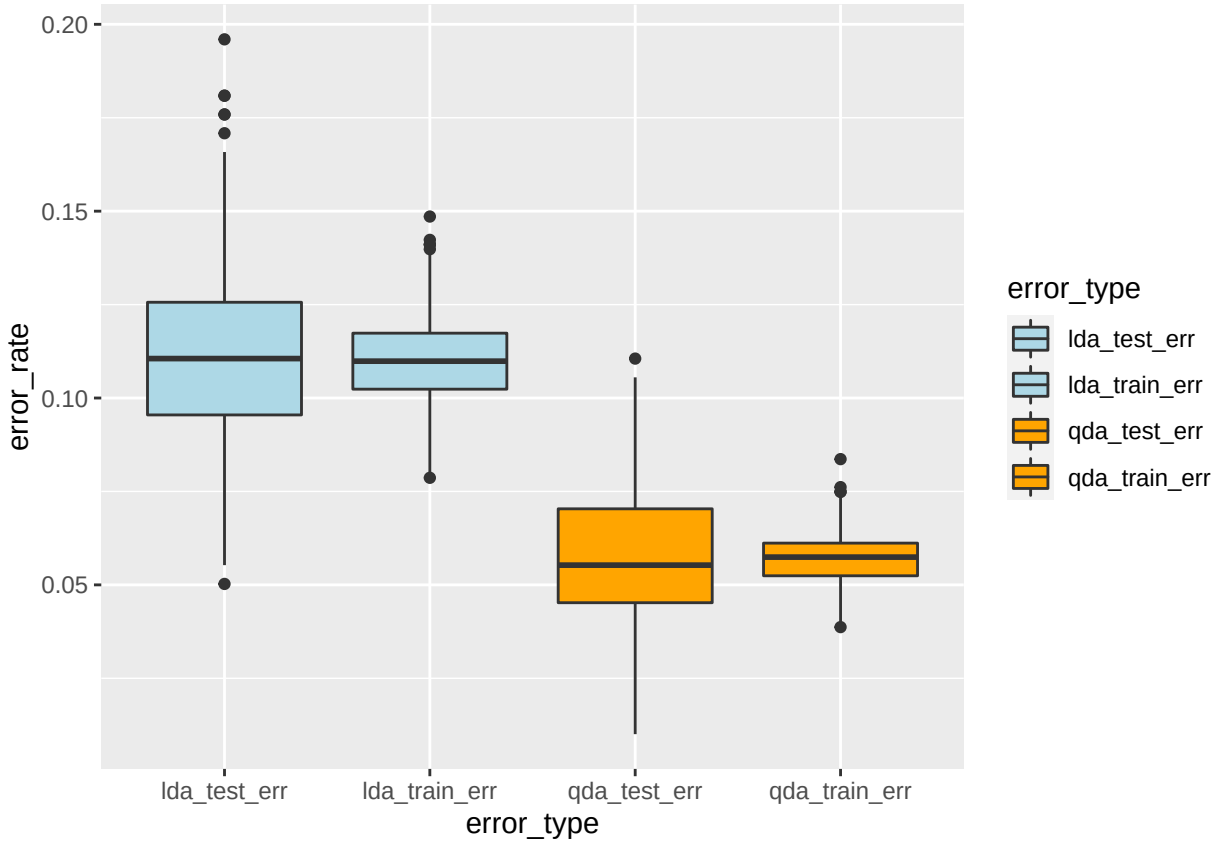
MACS 30100: Perspectives on Computational Modeling
University of Chicago

Wanxi Zhou

A Theoretical Problem

```
library(tidyverse)
library(tidymodels)
library(MASS)

SEED = 1234
K = 1000
N = 1000

# Helper function for classification
bayes_decision <- function(x1, x2){
  y = x1 + x1 ^2 + x2 + x2 ^ 2
  return(ifelse(y >= 0, TRUE, FALSE))
}

# Helper function for data set generation
create_dataset <- function(n, seed){
  set.seed(seed)
  return(tibble(x1 = runif(n, min = -1, max = 1),
                x2 = runif(n, min = -1, max = 1),
                y = bayes_decision(x1, x2)))
}

# Helper function for error rate calculation
err <- function(mod, err_type){
  return(sum(err_type$y != predict(mod, err_type)$class) / nrow(err_type))
}

lda_train_err = as.numeric(N)
qda_train_err = as.numeric(N)
lda_test_err = as.numeric(N)
qda_test_err = as.numeric(N)

# Compute training and test errors of the 2 methods for 1000 simulations
for(i in 1:K){
  dataset <- create_dataset(N, SEED)
  split <- initial_split(dataset,
                         prop = 0.8)
  train <- training(split)
  test <- testing(split)
```

```
  lda_mod <- lda(y ~ ., data = train)
  qda_mod <- qda(y ~ ., data = train)
  lda_train_err[i] <- err(lda_mod, train)
  qda_train_err[i] <- err(qda_mod, train)
  lda_test_err[i] <- err(lda_mod, test)
  qda_test_err[i] <- err(qda_mod, test)
  SEED <- SEED + 1
}

err_tb <- tibble(error_rate = c(lda_train_err, qda_train_err,
                        lda_test_err, qda_test_err),
                 error_type = c(rep("lda_train_err", N),
                        rep("qda_train_err", N),
                        rep("lda_test_err", N),
                        rep("qda_test_err", N)))

# Draw box plot for comparing the training and test errors
# of the 2 methods
myColors <- ifelse(levels(err_tb$error_type) == "lda_train_err" |
                   levels(err_tb$error_type) == "lda_test_err",
                   rgb(0.1,0.1,0.7,0.5), rgb(0.8,0.1,0.3,0.6))
err_tb %>%
  ggplot(aes(x = error_type, y = error_rate, fill = error_type)) +
  geom_boxplot() +
  scale_fill_manual(values = c("lightblue", "lightblue",
                               "orange", "orange"))
```




```
# Generate summary table
err_tb %>%
  group_by(error_type) %>%
  summarize(mean = mean(error_rate),
            median = median(error_rate),
            var = var(error_rate),
            sd = sd(error_rate),
            max = max(error_rate),
            min = min(error_rate),
            .groups = "drop")
```

```
## # A tibble: 4 x 7
##   error_type      mean median       var      sd    max    min
##   <chr>          <dbl>  <dbl>     <dbl>   <dbl>  <dbl>  <dbl>
## 1 lda_test_err  0.111  0.111  0.000487  0.0221  0.196  0.0503
## 2 lda_train_err 0.110  0.110  0.000118  0.0109  0.149  0.0787
## 3 qda_test_err  0.0580 0.0553 0.000269  0.0164  0.111  0.0101
## 4 qda_train_err 0.0570 0.0574 0.0000454 0.00674 0.0836 0.0387
```

From both the box plot and the table, we could conclude that QDA outperforms LDA when the Bayes decision boundary is non-linear (especially of the quadratic form). Both the training and testing errors of the QDA models are less than the LDA classifier and the former also has lower variance, which means that QDA could produce more accurate model for prediction in this case.

For both approaches,their training error and testing error share similar value, with the testing error slightly lower than the training error. As expected, it could be that the model is more familiar with the training set since it is trained on that set.

In non-linear contexts like this, since QDA generates a quadratic decision boundary, it could more accurately approximate the real Bayes classifier, resulting in lower bias in estimates. Therefore, even though higher flexibility raises the variance of the QDA model, lower biases guarantee better performance of QDA than LDA under this circumstance.

An Applied Problem

```
library(here)
library(tidyverse)
library(tidymodels)
library(MASS)
library(class)
library(klaR)
library(plotROC)
library(gridExtra)

 # Load and preprocess the data set
anes <- read_csv(here("data", "anes_pilot_2016.csv"))
anes_sample <- anes %>%
  dplyr::select(fttrump, ftobama, fthrc, ftrubio, ideo5, pid3) %>%
  mutate(fttrump = replace(fttrump, fttrump > 100, NA),
         ftobama = replace(ftobama, ftobama > 100, NA),
         fthrc = replace(fthrc, fthrc > 100, NA),
         ftrubio = replace(ftrubio, ftrubio > 100, NA),
         pid3 = replace(pid3, pid3 > 3, NA),
         ideo5 = replace(ideo5, ideo5 > 5, NA),
         democrat = as.factor(ifelse(pid3 == 1, 1, 0))) %>%
  drop_na() %>%
  dplyr::select(-pid3) %>%
  relocate(democrat)

# Split the data set and use the training set for cross validation approach
set.seed(222)

split <- initial_split(anes_sample,
                       prop = 0.8)
train <- training(split)
test <- testing(split)
cv_train <- vfold_cv(data = train,
                     v = 10)

# Create recipe and models (except KNN)
recipe <- recipe(democrat ~ .,
                 data = anes_sample)

mod_logistic <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

mod_lda <- discrim::discrim_linear() %>%
  set_engine("MASS")

mod_qda <- discrim::discrim_regularized(frac_common_cov = 0) %>%
```

```
  set_engine("klaR") %>%
  translate()


########################## LOGISTIC REGRESSION ##############################
# Start work flow
workflow <- workflow() %>%
  add_recipe(recipe) %>%
  add_model(mod_logistic)

# Collect test error and AUC
res_logistic <- workflow %>%
  fit_resamples(resamples = cv_train) %>%
  collect_metrics() %>%
  as_tibble() %>%
  dplyr::select(.metric, mean)

res_logistic[1, ] = list("error",
                         as.numeric(1 - res_logistic[1, 2]))
res_logistic <- res_logistic %>%
  add_column(model_type = "logistic")

# Plot ROC curve
predictions = NULL
truths = NULL
id = NULL
for(i in 1:10){
  fold <- cv_train$splits[[i]]
  tr <- analysis(fold)
  te <- assessment(fold)
  train_mod <- glm(democrat ~ .,
                   data = tr,
                   family = "binomial")
  pred <- predict(train_mod,
                  newdata = te,
                  type = "response")
  truths = c(truths,
             as.numeric(as.character(te$democrat)))
  predictions = c(predictions, pred)
  id = c(id, rep(as.character(fold$id), nrow(te)))
}

roc_logistic <- tibble(id = id,
                       truth = truths,
                       pred = predictions) %>%
  ggplot(aes(m = pred, d = truth, color = id)) +
  geom_roc(labels = F, size = 0.5) +
  ggtitle("Logistic ROC Curve") +
  scale_color_manual(values = c(rep("grey", 10))) +
  theme_minimal() +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.3))
```

```
################################### LDA ######################################
# Change model in work flow
workflow <- update_model(workflow, mod_lda)

# Collect test error and AUC
res_lda <- workflow %>%
  fit_resamples(resamples = cv_train) %>%
  collect_metrics() %>%
  as_tibble() %>%
  dplyr::select(.metric, mean)

res_lda[1, ] = list("error",
                    as.numeric(1 - res_lda[1, 2]))
res_lda <- res_lda %>%
  add_column(model_type = "lda")

# Plot ROC curve
predictions = NULL
truths = NULL
id = NULL
for(i in 1:10){
  fold <- cv_train$splits[[i]]
  tr <- analysis(fold)
  te <- assessment(fold)
  train_mod <- lda(democrat ~ .,
                   data = tr)
  pred <- predict(train_mod,
                  newdata = te)$posterior[,2]
  truths = c(truths,
             as.numeric(as.character(te$democrat)))
  predictions = c(predictions, pred)
  id = c(id, rep(as.character(fold$id), nrow(te)))
}

roc_lda <- tibble(id = id,
                  truth = truths,
                  pred = predictions) %>%
  ggplot(aes(m = pred, d = truth, color = id)) +
  geom_roc(labels = F, size = 0.3) +
  ggtitle("LDA ROC Curve") +
  scale_color_manual(values = c(rep("grey", 10))) +
  theme_minimal() +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.3))

################################### QDA ######################################
# Change model in work flow
workflow <- update_model(workflow, mod_qda)

# Collect test error and AUC
res_qda <- workflow %>%
  fit_resamples(resamples = cv_train) %>%
  collect_metrics() %>%
```

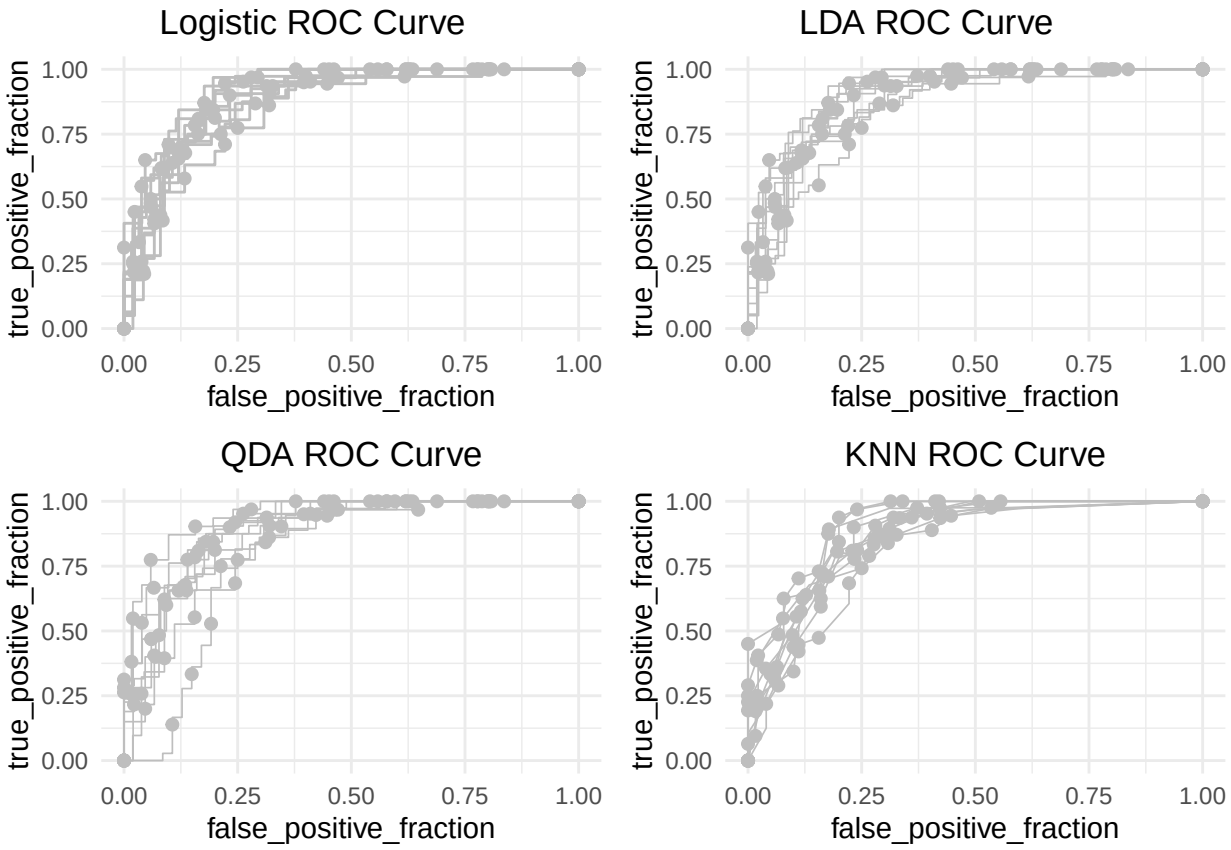
```
  as_tibble() %>%
  dplyr::select(.metric, mean)

res_qda[1, ] = list("error",
                    as.numeric(1 - res_qda[1, 2]))
res_qda <- res_qda %>%
  add_column(model_type = "qda")

#Plot ROC curve
predictions = NULL
truths = NULL
id = NULL
for(i in 1:10){
  fold <- cv_train$splits[[i]]
  tr <- analysis(fold)
  te <- assessment(fold)
  train_mod <- qda(democrat ~ .,
                   data = tr)
  pred <- predict(train_mod,
                  newdata = te)$posterior[,2]
  truths = c(truths,
             as.numeric(as.character(te$democrat)))
  predictions = c(predictions, pred)
  id = c(id, rep(as.character(fold$id), nrow(te)))
}

roc_qda <- tibble(id = id,
                  truth = truths,
                  pred = predictions) %>%
  ggplot(aes(m = pred, d = truth, color = id)) +
  geom_roc(labels = F, size = 0.3) +
  ggtitle("QDA ROC Curve") +
  scale_color_manual(values = c(rep("grey", 10))) +
  theme_minimal() +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5))

################################### KNN ######################################
# Select best k value for KNN
tb_knn <- tibble(k = 1:10,
                 err_rate = NA,
                 auc = NA)

for(i in 1:10){
  mod_knn <- nearest_neighbor(neighbors = i,
                              dist_power = 2) %>%
    set_engine("kknn") %>%
    set_mode("classification")

  workflow <- update_model(workflow, mod_knn)

  res_knn <- workflow %>%
    fit_resamples(resamples = cv_train) %>%
```

```
    collect_metrics()

  tb_knn$err_rate[i] <- res_knn$mean[1]
  tb_knn$auc[i] <- res_knn$mean[2]
}

# Select k = 10
print(tb_knn)
```

```
## # A tibble: 10 x 3
##        k err_rate   auc
##    <int>    <dbl> <dbl>
##  1     1    0.748 0.740
##  2     2    0.748 0.806
##  3     3    0.748 0.827
##  4     4    0.748 0.840
##  5     5    0.759 0.847
##  6     6    0.761 0.851
##  7     7    0.761 0.857
##  8     8    0.769 0.860
##  9     9    0.775 0.865
## 10    10    0.777 0.868
```

```
# Collect test error and AUC
res_knn <- workflow %>%
  fit_resamples(resamples = cv_train) %>%
  collect_metrics() %>%
  as_tibble() %>%
  dplyr::select(.metric, mean)

res_knn[1, ] = list("error",
                    as.numeric(1 - res_knn[1, 2]))
res_knn <- res_knn %>%
  add_column(model_type = "knn")

# Plot ROC curve
predictions = NULL
truths = NULL
id = NULL
for(i in 1:10){
  fold <- cv_train$splits[[i]]
  tr <- analysis(fold)
  te <- assessment(fold)
  knn_mod <- knn(tr, te, tr$democrat,
                 k = 10, prob = T)
  pred <- knn(tr, te, tr$democrat, k = 10, prob = T)
  prop <- attributes(pred)$prob
  prop <- ifelse(pred == 0, 1 - prop, prop)
  truths = c(truths,
             as.numeric(as.character(te$democrat)))
  predictions = c(predictions, prop)
  id = c(id, rep(as.character(fold$id), nrow(te)))
}
```

```
roc_knn <- tibble(id = id,
                  truth = truths,
                  pred = predictions) %>%
  ggplot(aes(m = pred, d = truth, color = id)) +
  geom_roc(labels = F, size = 0.3) +
  ggtitle("KNN ROC Curve") +
  scale_color_manual(values = c(rep("grey", 10))) +
  theme_minimal() +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5))


## Aggregate test errors, ROC curves and AUC for model selection
grid.arrange(roc_logistic, roc_lda, roc_qda, roc_knn,
             ncol = 2, nrow = 2)
```




```
bind_rows(res_logistic, res_lda, res_qda, res_knn) %>%
  group_by(.metric) %>%
  arrange(desc(mean))
```

```
## # A tibble: 8 x 3
## # Groups:   .metric [2]
##   .metric  mean model_type
##   <chr>   <dbl> <chr>
## 1 roc_auc 0.893 lda
## 2 roc_auc 0.893 logistic
```

```
## 3 roc_auc 0.887 qda
## 4 roc_auc 0.868 knn
## 5 error   0.223 knn
## 6 error   0.202 qda
## 7 error   0.196 lda
## 8 error   0.191 logistic
```

```
#################### SELECT BETWEEN LOGISTIC AND LDA ########################
cov(anes_sample[anes_sample$democrat == 0,2:6])
```

```
##            fttrump    ftobama      fthrc     ftrubio      ideo5
## fttrump 1277.00825 -574.53829 -384.24397  285.783238  14.286343
## ftobama -574.53829 1062.63940  792.34768 -250.175932 -18.519943
## fthrc   -384.24397  792.34768  944.36428 -180.202563 -16.415382
## ftrubio  285.78324 -250.17593 -180.20256  754.857134   8.908719
## ideo5     14.28634  -18.51994  -16.41538    8.908719   1.048521
```

```
cov(anes_sample[anes_sample$democrat == 1,2:6])
```

```
##             fttrump     ftobama      fthrc     ftrubio      ideo5
## fttrump  651.268249 -176.644827 -64.964756  219.644766  7.2498216
## ftobama -176.644827  547.494352 253.714455 -180.963870 -4.7523418
## fthrc    -64.964756  253.714455 692.026534  -76.103528 -1.6595187
## ftrubio  219.644766 -180.963870 -76.103528  574.473616  8.5196485
## ideo5      7.249822   -4.752342  -1.659519    8.519649  0.9909229
```

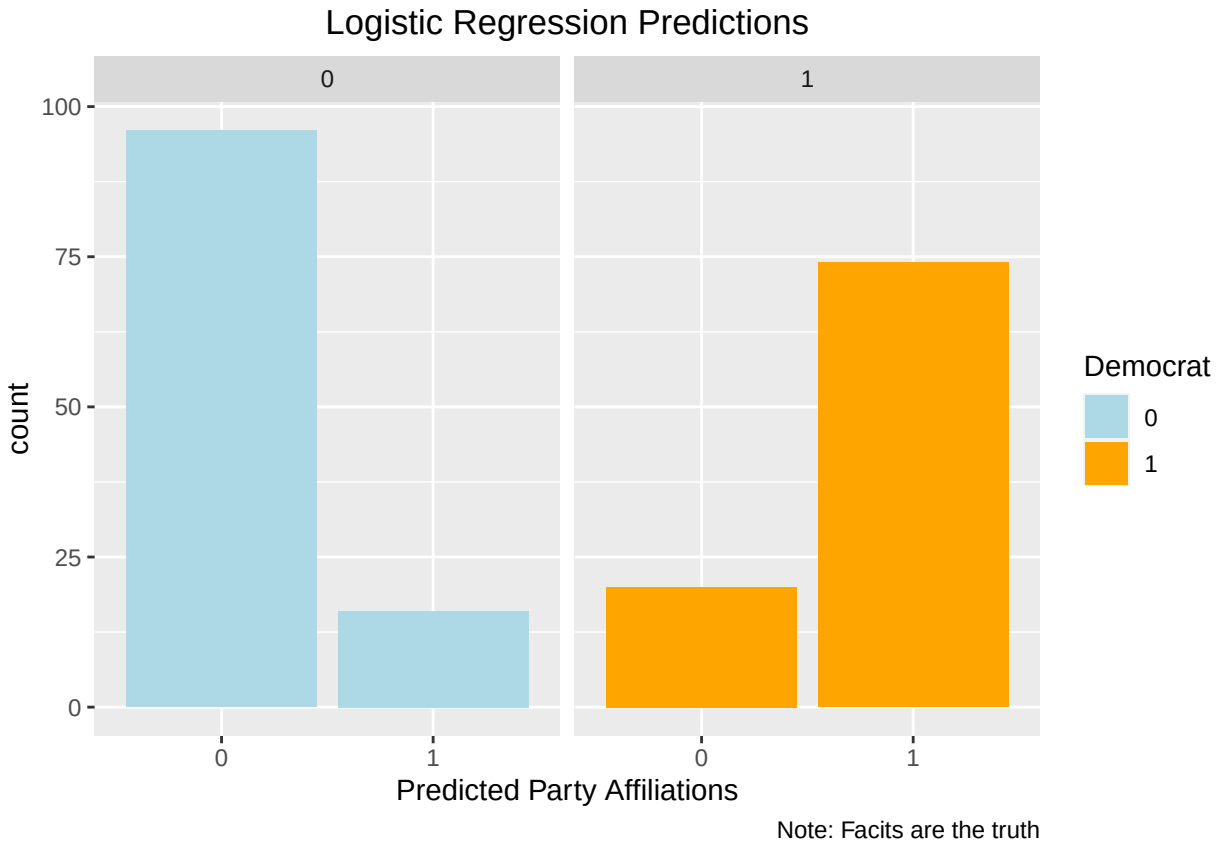
```
############################## CHOOSE LOGISTIC ##################################
# Evaluate logistic model using test set
final_mod <- workflow %>%
  update_model(mod_logistic) %>%
  last_fit(split)

# Collect test error and AUC
m <- final_mod %>%
  collect_metrics() %>%
  as_tibble() %>%
  dplyr::select(.metric, .estimate)

m %>%
  add_row(.metric = "err_rate",
          .estimate = 1 - m$.estimate[1])
```

```
## # A tibble: 3 x 2
##   .metric  .estimate
##   <chr>        <dbl>
## 1 accuracy     0.825
## 2 roc_auc      0.902
## 3 err_rate     0.175
```

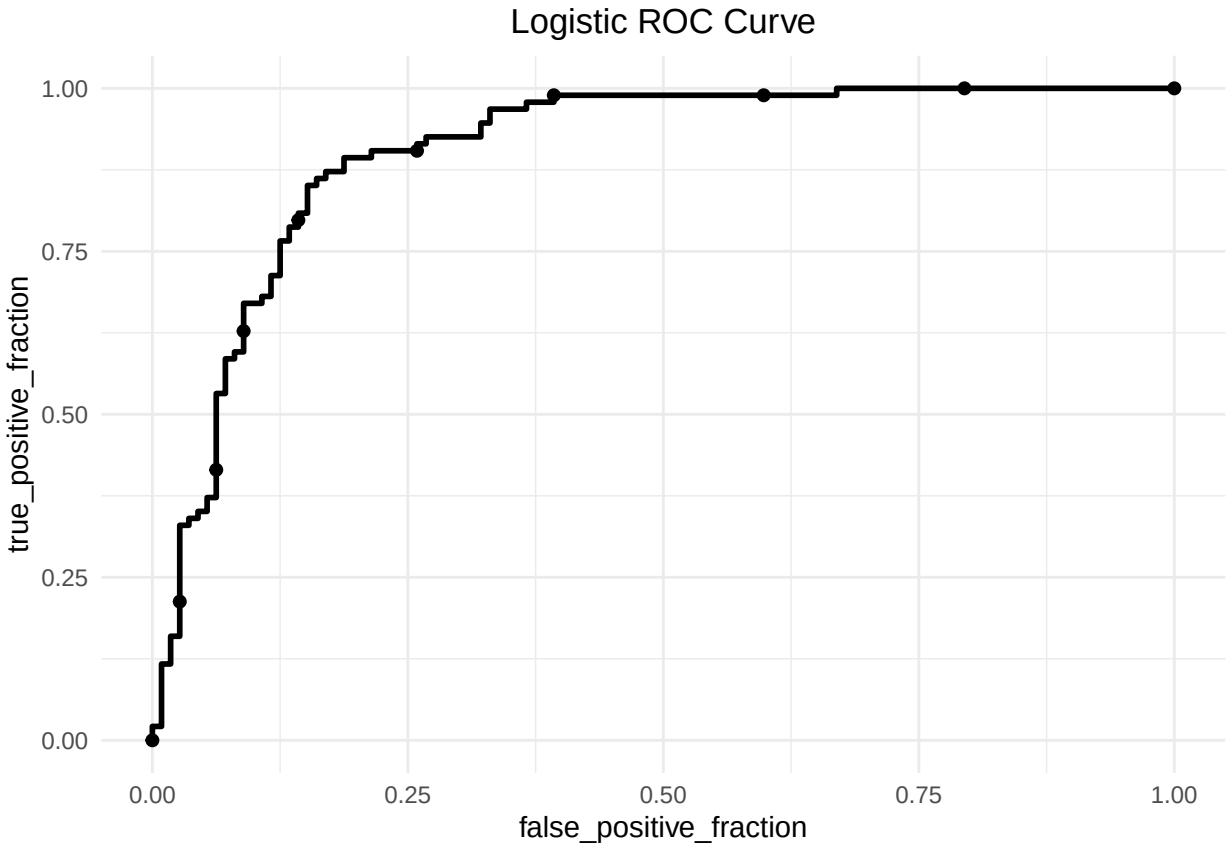
```
# Generate confusion metric
res <- final_mod %>%
  collect_predictions()

res %>%
  conf_mat(truth = democrat,
           estimate = .pred_class,
           dnn = c("Pred", "Truth"))
```

```
##      Truth
## Pred  0  1
##    0 96 20
##    1 16 74
```

```
# Plot logistic regression predictions
res %>%
  ggplot() +
  geom_bar(aes(x = .pred_class,
               fill = democrat)) +
  facet_wrap(~ democrat) +
  labs(title = "Logistic Regression Predictions",
       x = "Predicted Party Affiliations",
       fill = "Democrat",
       caption = "Note: Facits are the truth") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual(values = c("lightblue", "orange"))
```




```
# Plot ROC curve
tibble(truth = as.numeric(as.character(res$democrat)),
       pred = res$.pred_1) %>%
  ggplot(aes(m = pred, d = truth)) +
  geom_roc(labels = F) +
  ggtitle("Logistic ROC Curve") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```



According to the test errors, AUC, and ROC curves of different models computed from the training data using cross validation, we can observe that logistic regression and LDA models share almost identical performance results. However, since each class of the sample data demonstrates variant covariance matrices, logistic regression is favored in this context, for the purpose of more accurately predicting future unfamiliar data sets.

According to the final evaluation results, the logistic model achieves an approximately 0.2 test error and 0.9 AUC, exhibiting reliable prediction performance. The ROC curve also hugs the top left corner, indicating a both low Type I error and Type II error. Therefor, the area under the curve approaches 1.

The relatively higher performance of logistic regression, along with LDA, indicates the possibility that the decision boundary of party affiliation in this case might be approximately linear. Compared to all the other classifiers, the selected approach, logistic regression, is less flexible, and consequently has lower variance. Furthermore, if in reality, the response class is indeed separated by a linear boundary, the logistic model could generate less biased estimates and perform relatively accurate classification.

The result supports the theorical assumption that concepts (e.g., feelings towards various political figures) indirectly related to one's party affiliation could actually drive their party affiliation.