Homework Assignment #4

Math 437 - Modern Data Analysis

Due Sometime After We Get Back From Spring Break

Instructions

You should submit either two or three files:

1. You should write your solutions to the Applied Problems and Conceptual Problem 3 in this R Markdown file and submit the (.Rmd) file.
2. You should knit the final solution file to pdf and submit the pdf. If you are having trouble getting code chunks to run, add `eval = FALSE` to the chunks that do not run. If you are having trouble getting R Studio to play nice with your LaTeX distribution, I will begrudgingly accept an HTML file instead.
3. Solutions to the Key Terms and the other Conceptual Problems can be submitted in a separate Word or pdf file or included in the same files as your solutions to Conceptual Problem 3 and the Applied Problems.

This homework assignment is worth a total of **50 points**.

```
library(ISLR2)
library(ggplot2)
library(dplyr)
library(tidymodels)
library(workflows)
library(parsnip)
library(car)
library(discrim)
library(klaR)

# You will need other packages to fit models in the Applied Problem
# List them here or as you encounter the need for them

misinformation1 <- readr::read_csv("misinformation1.csv")
```

Key Terms (5 pts)

Read Chapter 4 of Introduction to Statistical Learning, Second Edition. Based on your reading, answer the following questions.

1. Explain how to convert probabilities to *odds* and to *logits*.

To convert probabilities to odds, we manipulate the logistic function $p(X) = e^{\beta_0+\beta_1 X}/(1 + e^{\beta_0+\beta_1 X})$ to solve for $e^{\beta_0+\beta_1 X}$. To obtain logits, we take the logarithm of the manipulated equation to obtain $log(\frac{p(X)}{1-p(X)})$.

2. Write a sentence to interpret each Coefficient in Table 4.3.

Holding all other predictors constant, for every \$1 increase in balance, we expect the log odds of default to increase by 0.0057%. Holding all other predictors constant, for every \$1,000 increase in income, we expect

the log odds of default to increase by 0.003%. Holding all other predictors constant, when compared to non-students, we expect the log odds of default to decrease by 0.468% for students.

3. Why is the choice of a *baseline* (or *reference level*) for the response variable critical when interpreting slopes in multinomial logistic regression, but not so much in (standard) logistic regression?

In multinomial regression, the choice of baseline affects how we interpret the log odds, since for each predictors coefficient, we are interpreting them relative to the chosen baseline. In standard logistic regression we only have two classes in our response so when we change our baseline, only the sign of the coefficients change rather than the value and our interpretation remains the same except for the direction of change.

4. In *Bayes' Theorem* (Equation 4.15), what does π_k represent about class k? What does $f_k(x)$ represent?

π_k represents the prior probability that a randomly chosen observation comes from class k. $f_k(x)$ is the density function of X for an observation that comes from the kth class. In other words, for qualitative random variables, it is the probability that an observation $X \approx x$ given that the observation comes from the kth class; for quantitative random variables it is the probability that X falls into a small region dx around x.

5. Compare and contrast the assumptions about the predictor(s) in *linear discriminant analysis* vs. *quadratic discriminant analysis*. Which approach leads to more complex/flexible models?

In linear discriminant analysis (LDA) we are assuming that within each class, our observations come from multivariate normal distributions and that there's a common variance and covariance across classes. In quadratic discriminant analysis (QDA), we are making the same assumptions about normality, but we are not assuming that there's a common covariance matrix across classes. Quadratic discriminant analysis leads to more complex/flexible models since it is estimating a separate covariance matrix for each class.

6. Use the *confusion matrix* in Table 4.5 to find the *sensitivity*, *specificity*, *positive predictive value* and *negative predictive value* for this algorithm. It's okay to keep your answers as fractions.

Sensitivity = 195/333 Specificity = 9432/9667 positive predictive value = 195/430 negative predictive value = 9432/9570

7. What do the x-axis and y-axis on a *receiver operating characteristic* (ROC) curve represent? What is the curve actually a function of?

The x-axis represents the false positive rate, and the y-axis represents the true positive rate. The curve is a function of our threshold value and how it affects sensitivity and the false positive rate.

8. Compare and contrast the assumptions about the predictors in *linear discriminant analysis* vs. *Naive Bayes*. When are they equivalent?

In Naive Bayes, we assume the predictors are independent. They are equivalent when in Naive Bayes, we assume our observations for each predictor come from normal/Gaussian distributions similar to LDA.

9. Consider the five methods compared in Section 4.5.2: LDA, QDA, Naive Bayes, Logistic Regression, K-Nearest Neighbors. Which methods would you guess to perform best when you suspect the decision boundary to be (a) linear, (b) moderately nonlinear, (c) highly complex?

(a) LDA and logistic regression would likely perform best for linear decision boundaries with logisitic regression outperforming LDA when assumptions of normality are not met.

(b) Naive Bayes and QDA would perform best for moderately non-linear decision boundaries with Naive Bayes outperforming QDA when there are only a small number of observations.

(c) K-Nearest Neighbors would perform best for highly complex decision boundaries given that the appropriate value for K is chosen.

10. How does "regression" work in a *generalized linear model*?

Regression works by modeling the response, which is assumed to come from a distribution from the exponential family, and then transforming the expected value of the response so that this transformed mean is a linear function of predictors.

Conceptual Problems

Conceptual Problem 1 (2 pts)

Textbook Exercise 4.8.1.

Conceptual Problem 2 (4 pts)

Part a (1 pt)

Textbook Exercise 4.8.2.

Part b (3 pts)

Suppose that within Group 1, $X \sim Pois(\lambda_1)$ and within Group 2, $X \sim Pois(\lambda_2)$. Using techniques employed in the proof in part (a), find the estimated Bayes decision boundary for classifying an observation to Group 1 vs. Group 2. You may assume $\hat{\lambda_k}$ is computed as the sample mean of the observations in group k in the training set and that $\hat{\pi}_k$ is computed as the proportion of observations in the training set that are in group k.

Conceptual Problem 3 (6 pts total)

Linear discriminant analysis is actually not a Bayesian concept! It was introduced by Fisher in his analysis of iris data as a dimensionality reduction method! In this problem, you will follow Fisher's logic and replicate his discrimination function.

Fisher's original LDA example concerned only the setosa and versicolor flowers, so we filter the iris dataset to include only those species.

```
iris_sv <- iris %>% filter(Species %in% c("setosa", "versicolor"))
```

Part a (Code: 1.5 pts)

Let x_1 be Sepal Length, x_2 be Sepal Width, x_3 be Petal Length, and x_4 be Petal Width. Fisher wants to find the linear combination of the four variables $X = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 + \lambda_4 x_4$ that maximizes the overall "distance" in sample means between *versicolor* and *setosa*, accounting for variation and covariation within each group.

Specifically, he wants to project this four-dimensional predictor space into a single dimension along which the ratio D^2/S is maximized, where D is the difference in sample means and S is the total within-class sum of squares (i.e., SSE in an ANOVA table) after transformation.

Fisher starts by computing a "sum of squares and products" matrix S in each group. The S matrices can be found more easily in R by obtaining the variance-covariance matrix for the numerical predictors (e.g., using the `cov` function) and multiplying the entries by (n-1). Finally, Fisher adds the S matrices for each group to get the overall within-class variation.

Using R, create the `S_setosa`, `S_versicolor`, and `S_overall` matrices.

```
head(iris_sv)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
S_setosa <-  (cov(iris_sv %>%
                        filter(Species == "setosa") %>%
                        dplyr::select(-Species))*(50-1))

S_versicolor <-  (cov(iris_sv %>%
                        filter(Species == "versicolor") %>%
                        dplyr::select(-Species))*(50-1))

S_overall <-  S_setosa + S_versicolor

print(S_setosa)
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length       6.0882      4.8616       0.8014      0.5062
## Sepal.Width        4.8616      7.0408       0.5732      0.4556
## Petal.Length       0.8014      0.5732       1.4778      0.2974
## Petal.Width        0.5062      0.4556       0.2974      0.5442
```

```
print(S_versicolor)
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length      13.0552       4.174        8.962      2.7332
## Sepal.Width        4.1740       4.825        4.050      2.0190
## Petal.Length       8.9620       4.050       10.820      3.5820
## Petal.Width        2.7332       2.019        3.582      1.9162
```

```
print(S_overall)
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length      19.1434      9.0356       9.7634      3.2394
## Sepal.Width        9.0356     11.8658       4.6232      2.4746
## Petal.Length       9.7634      4.6232      12.2978      3.8794
## Petal.Width        3.2394      2.4746       3.8794      2.4604
```

Part b (Code: 1 pt; Explanation: 0.5 pts)

Fisher then solves a system of four linear equations in four unknowns (λ_1, λ_2, λ_3, λ_4) that relates S to the vector of differences in sample means, i.e., $S\lambda = D$ where D is the vector of differences in sample means.

The discriminant function is then the inner product of λ and x. This solution is unique up to a scaling factor. Fisher suggests to scale λ such that $\lambda_1 = 1$.

Using R, solve the matrix equation (`%*%` does matrix multiplication and `solve` does matrix inversion) and perform Fisher's suggested scaling, then write out the discriminant function as a linear function of x_1, x_2, x_3, x_4.

x_1 be Sepal Length, x_2 be Sepal Width, x_3 be Petal Length, and x_4 be Petal Width.

```
iris_s <- iris_sv %>%
  filter(Species == "setosa")

iris_v <- iris_sv %>%
  filter(Species == "versicolor")

D = matrix(data = c(mean(iris_s$Sepal.Length) - mean(iris_v$Sepal.Length),
                    mean(iris_s$Sepal.Width) - mean(iris_v$Sepal.Width),
                    mean(iris_s$Petal.Length) - mean(iris_v$Petal.Length),
```

```
                    mean(iris_s$Petal.Width) - mean(iris_v$Petal.Width)))

lambda = solve(S_overall, D)


#Use Sepal Length as our reference level
fisher_scaling = lambda * 1/lambda[1,1]
```

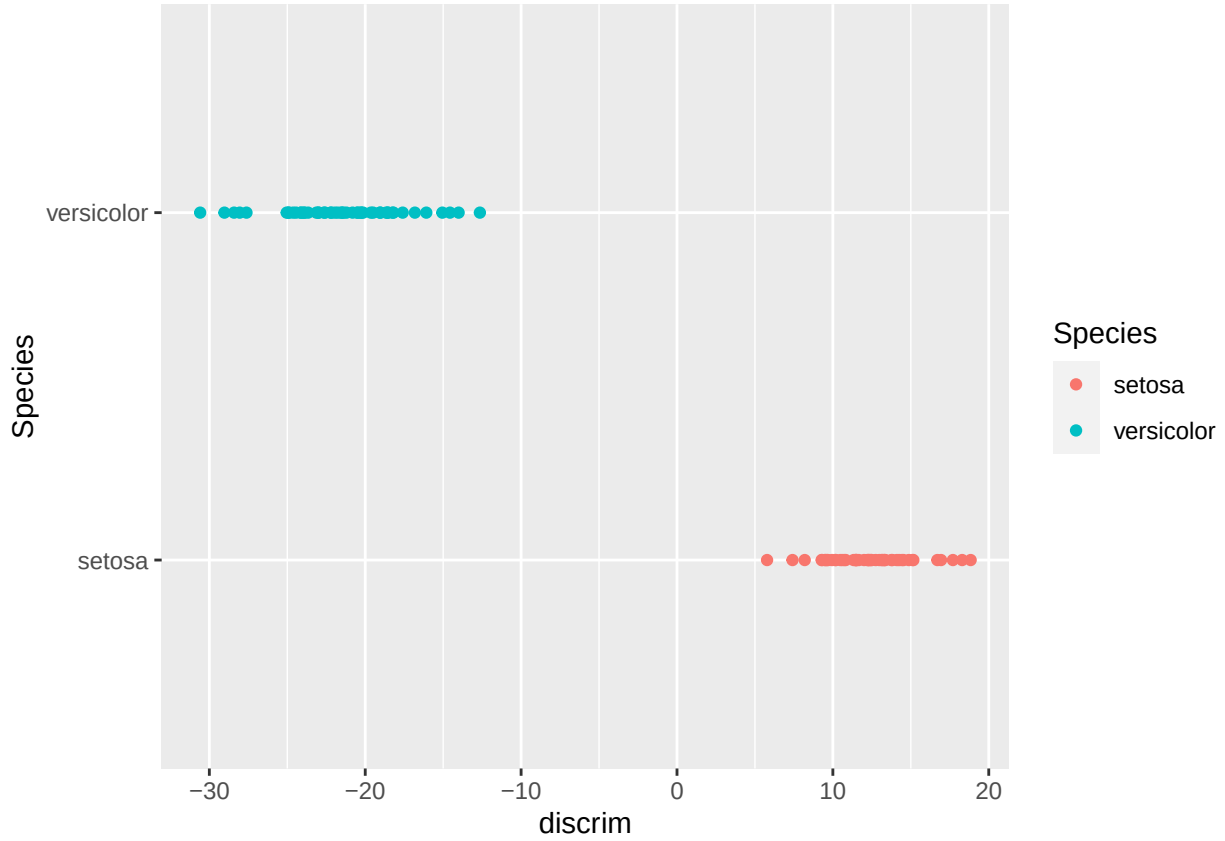
$ Discriminant Function = x_1*1 + x_2* 5.9038 - x_3*7.1299 -x_4*10.1037 $

Part c (Code: 1 pt)

Add a new variable to the `iris_sv` data frame, `discrim`, containing the values of the discriminant function for each flower. Create a dot plot showing the Species (response) vs. the discriminant function value (predictor). You may also want to color-code by Species.

```
iris_sv <- iris_sv %>%
  mutate(discrim = Sepal.Length*1 + Sepal.Width * 5.9038 - Petal.Length*7.1299 -Petal.Width*10.1037)
```

```
iris_sv %>%
  ggplot(aes(x = discrim, y = Species, color = Species)) +
  geom_point()
```



Part d (Code and/or Explanation: 2 pts)

Explain how to use your results to classify a *new* iris flower to either *setosa* or *versicolor*. (Hint: think about where the decision boundary is. . .)

In order to classify a new iris flower we would calculate its' discriminant and if that value is less than 0 we would predict it to be veriscolor and if that value were greater than 0 it would be classified as setosa. (I am assuming that the decision boundary is zero but I am not 100% positive)

Applied Problems

Applied Problem 1 (33 pts total)

This problem involves the `misinformation1` dataset. In 2020, while biomedical researchers were attempting to develop a vaccine for COVID-19, public health researchers were attempting to predict whether a person would get a vaccine once one was available.

The `misinformation1` dataset contains responses of a subset of 673 Americans to a survey about COVID-19. The `Vaccine` variable indicates whether the person said they would get the vaccine ("Yes") or said they would not ("No"). Here I create a `misinformation2` dataset to convert the `Vaccine` variable to a factor variable.

```
misinformation2 <- misinformation1 %>% mutate(
  Vaccine = as.factor(Vaccine)
)
```

The researchers who analyzed this data believed that people who thought that COVID-19 was a higher public risk (`COVID_Risk`) and people who had higher trust in scientists (`Trust_in_Scientists`) would be more likely to say they would get the vaccine, while those who were more susceptible to believing misinformation (`Misinformation`) about the vaccine would be less likely to say they would get the vaccine. So we will use those three predictors in our models.

Part a (Code: 1 pt)

Randomly divide the `misinformation2` dataset into a holdout set with 20-25% of the data (anywhere from 130 to 170 observations is fine) and a training set with the remaining observations. I used seed 222 in my split, but you do not need to replicate my results. Either the "Base R" or the tidymodels (using `rsample`) way of doing the split is acceptable.

```
set.seed(222)
misinfo_split <- initial_split(misinformation2, prop = 0.75)
misinfo_train <- training(misinfo_split)
misinfo_test <- testing(misinfo_split)
```

Part b (Code: 4 pts; Explanation: 1 pt)

Use K-nearest neighbors with a Euclidean distance (`dist_power = 2`) metric to predict whether someone will get the COVID-19 vaccine. Use repeated 5-fold cross-validation to find the optimal value of k, and explain why you chose that value of k. Remember to do all the necessary preparation work and fit the model on the entire training set afterwards. It is probably easiest to use a tidymodels workflow to do this part.

```
set.seed(1002)
fv_kfold_tidy <- vfold_cv(misinfo_train, v = 5, repeats = 3)
fv_kfold_tidy
```

```
## #  5-fold cross-validation repeated 3 times
## # A tibble: 15 x 3
##    splits            id      id2
##    <list>            <chr>   <chr>
##  1 <split [403/101]> Repeat1 Fold1
##  2 <split [403/101]> Repeat1 Fold2
##  3 <split [403/101]> Repeat1 Fold3
##  4 <split [403/101]> Repeat1 Fold4
```

```
##  5 <split [404/100]> Repeat1 Fold5
##  6 <split [403/101]> Repeat2 Fold1
##  7 <split [403/101]> Repeat2 Fold2
##  8 <split [403/101]> Repeat2 Fold3
##  9 <split [403/101]> Repeat2 Fold4
## 10 <split [404/100]> Repeat2 Fold5
## 11 <split [403/101]> Repeat3 Fold1
## 12 <split [403/101]> Repeat3 Fold2
## 13 <split [403/101]> Repeat3 Fold3
## 14 <split [403/101]> Repeat3 Fold4
## 15 <split [404/100]> Repeat3 Fold5
```

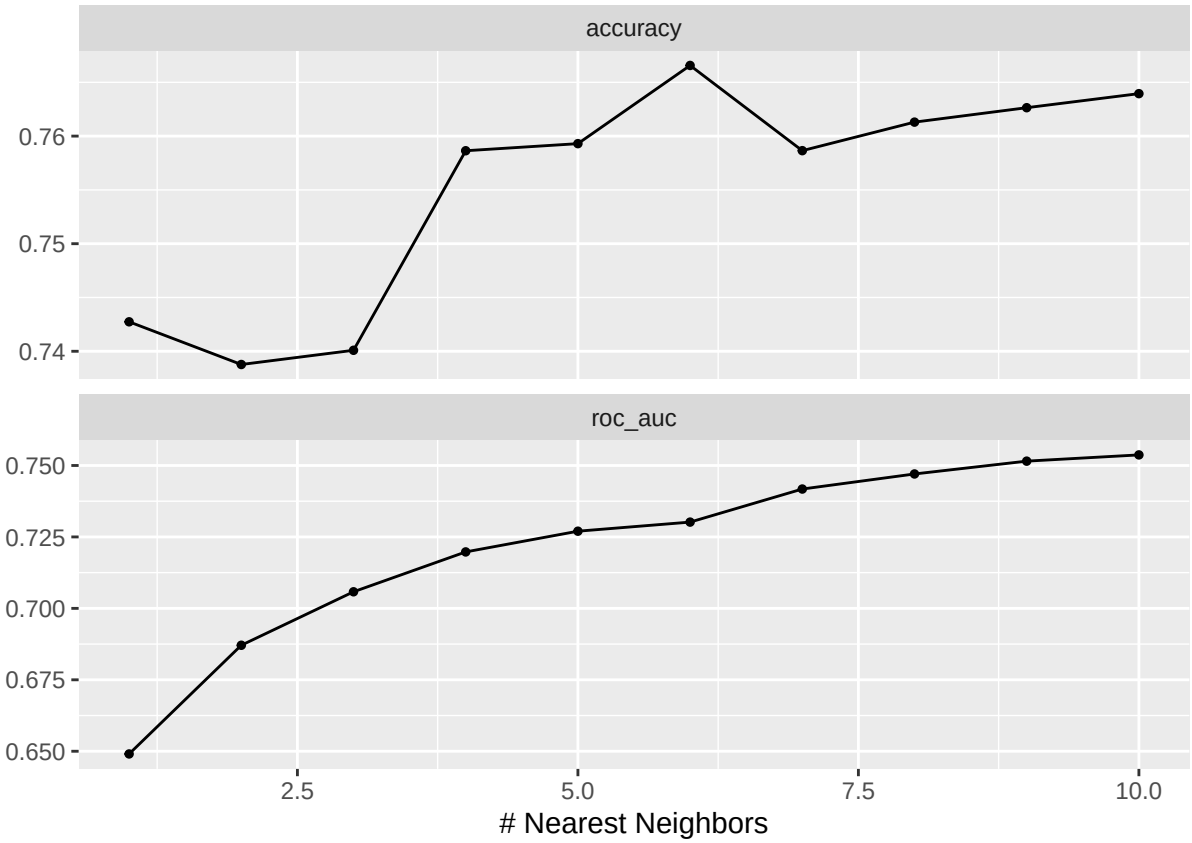
```
knn_model <- nearest_neighbor(mode = "classification", neighbors = tune(), dist_power = 2)

knn_wflow <- workflow() %>%
  add_model(knn_model)
```

```
knn_param_grid <- expand.grid(neighbors = seq(1, 10))
```

```
knn_recipe <- recipe(
  Vaccine ~ COVID_Risk + Trust_in_Scientists + Misinformation, # response ~ predictors
  data = misinfo_train
) %>%
  step_normalize(all_numeric_predictors()) # center and scale numeric predictors
```

```
knn_tune <- tune_grid(knn_model,
                      knn_recipe,
                      resamples = fv_kfold_tidy,
                      grid = knn_param_grid)
```

```
autoplot(knn_tune)
```



The optimal value I chose is k = 6 because it has the highest accuracy.

Part c (Code: 1.5 pts; Computation: 1 pt)

Make your predictions on the holdout set. Then, obtain the confusion matrix for this model on the holdout set. Using the confusion matrix, estimate the accuracy, sensitivity (recall), specificity, positive predictive value (precision), and negative predictive value for the K-nearest neighbors model with your optimal value of K.

Confirm your estimates by getting the `summary` of the confusion matrix. Remember to use the argument `event_level = "second"` in the summary function because we want to predict whether someone *will* get a vaccine.

```
knn_model_final <- nearest_neighbor(mode = "classification", neighbors = 6, dist_power = 2)

knn_wflow <- workflow() %>%
  add_model(knn_model_final) %>%
  add_recipe(knn_recipe)


knn_fit <- fit(knn_wflow, data = misinfo_train)

predictions_knn_df <- broom::augment(knn_fit, new_data = misinfo_test)

knn_conf_mat <- conf_mat(predictions_knn_df, truth = Vaccine, estimate = .pred_class)
knn_conf_mat
```

```
##           Truth
```

```
## Prediction  No Yes
##        No   27  18
##        Yes  18 106
```

accuracy: $133/169 = 0.79$

sensitivity: TP/TP+FN $= 106/124 = 0.85$

specificity: TN/TN+FP $= 27/45 = 0.6$

ppv: TP/TP+FP $= 106/124 = 0.85$

npv: TN/TN+FN $= 27/46 = 0.6$

```
summary(knn_conf_mat, event_level = "second")
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.787
##  2 kap                  binary         0.455
##  3 sens                 binary         0.855
##  4 spec                 binary         0.6
##  5 ppv                  binary         0.855
##  6 npv                  binary         0.6
##  7 mcc                  binary         0.455
##  8 j_index              binary         0.455
##  9 bal_accuracy         binary         0.727
## 10 detection_prevalence binary         0.734
## 11 precision            binary         0.855
## 12 recall               binary         0.855
## 13 f_meas               binary         0.855
```

Part d (Code: 1 pt; Explanation: 1 pt)

Logistic regression fixes a lot of issues that are present in k-nearest neighbors. For one thing, we don't have to do any of the pre-processing and can use the variables on their original scale, which makes interpretation much easier.

Fit a logistic regression model on the training set predicting `Vaccine` from `COVID_Risk`, `Trust_in_Scientists`, and `Misinformation`. Use the `glm` function (don't use tidymodels here, because tidymodels will remove the information you need to do the inference in Part f) with an appropriate `family` argument.

Use the `summary` or `coef` function to obtain the coefficient estimates, and write out the equation of the fitted logistic regression model. You can use either the log-odds formulation or the probability formulation, but please make sure you are describing what you are finding the log-odds or probability of.

```
logr_misinfo <- glm(Vaccine ~ COVID_Risk + Trust_in_Scientists + Misinformation,
                    data = misinfo_train, family = "binomial")
summary(logr_misinfo)
```

```
##
## Call:
## glm(formula = Vaccine ~ COVID_Risk + Trust_in_Scientists + Misinformation,
##     family = "binomial", data = misinfo_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6574   0.1978   0.4349   0.6856   1.7946
```

```
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -3.12175    0.71521  -4.365 1.27e-05 ***
## COVID_Risk           0.91610    0.13759   6.658 2.77e-11 ***
## Trust_in_Scientists  0.42697    0.13820   3.089  0.00201 **
## Misinformation      -0.39788    0.08936  -4.453 8.49e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 562.40  on 503  degrees of freedom
## Residual deviance: 447.72  on 500  degrees of freedom
## AIC: 455.72
##
## Number of Fisher Scoring iterations: 5
```

logit(Someone will get a Vaccine) = -3.12 + 0.92(`COVID_Risk`) + 0.43(`Trust_in_Scientists`) - 0.398(`Misinformation`)

Part e (Explanation: 2 pts)

Write a sentence interpreting the coefficient corresponding to `Trust_in_Scientists` in the logistic regression model. It is easiest to exponentiate the coefficient and discuss a *multiplicative* increase in odds.

When `Trust_in_Scientists` increases by one unit, holding all other predictors in the model constant, the multiplicative increase in odds of someone saying they will get a vaccine increases by 1.5.

Part f (Code: 0.5 pt; Explanation: 2 pts)

Using the `confint` function, obtain 95% confidence intervals for the parameters of the population logistic regression model. Based on the confidence intervals, which of the researchers' suspicions about the relationship between `Vaccine` and the three predictors can you conclude are true?

```
confint(logr_misinfo, level = 0.95)
```

```
## Waiting for profiling to be done...
```

```
##                          2.5 %     97.5 %
## (Intercept)         -4.5603106 -1.7487722
## COVID_Risk           0.6535791  1.1944511
## Trust_in_Scientists  0.1573488  0.7006974
## Misinformation      -0.5741017 -0.2228080
```

Based on the confidence intervals, we can conclude both of the scientists suspicions are true. People we thought that COVID-19 was a higher public risk and people who had higher trust in scientists would be more likely to say they would get a vaccine. Those who were more susceptible to believing misinformation about the vaccine would be less likely to say they would get the vaccine.

Part g (Code: 2 pts; Computation: 1 pt)

Let's make sure we understand how to make class predictions without using tidymodels.

Use the `predict` function to obtain predictions for the validation set. Remember to include the argument `type = "response"` to output predictions as probabilities! Note that `augment` is a bit finicky when passing in a `glm` object, so you should not use that function here.

Using similar code to Lab 4.7.2 or an `if_else` statement, classify each respondent in the validation set as either getting the vaccine ("Yes") or not ("No") using the estimated Bayes decision boundary.

Use the `table` function to obtain the confusion matrix for this model on the holdout set. Based on the confusion matrix, estimate the accuracy, sensitivity (recall), specificity, positive predictive value (precision), and negative predictive value for the logistic regression model.

Confirm your estimates using the `conf_mat` and `summary` functions in the `yardstick` package. Remember to use the argument `event_level = "second"` in the summary function because we want to predict whether someone *will* get a vaccine.

```
predicted_probs <- predict(logr_misinfo, newdata = misinfo_test, type = "response")
logr_predictions <- tibble(Misinformation = misinfo_test$Misinformation, Trust_in_Scientists = misinfo_

p.threshold <- 0.5

predicted_category <- if_else(predicted_probs > p.threshold, "Yes", "No")

logr_prediction <- tibble(Misinformation = misinfo_test$Misinformation, Trust_in_Scientists = misinfo_t
                          pred_prob = predicted_probs, pred_class = predicted_category)

table(logr_prediction$pred_class, logr_prediction$Vaccine,
      dnn = c("Predicted","Actual"))
```

```
##           Actual
## Predicted  No Yes
##       No   21  10
##       Yes  24 114
```

accuracy: $135/169 = 0.799$

sensitivity: TP/TP+FN $= 114/124 = 0.92$

specificity: TN/TN+FP $= 21/46 = 0.46$

ppv: TP/TP+FP $= 114/138 = 0.83$

npv: TN/TN+FN $= 21/31 = 0.68$

```
logr_prediction$pred_class<-as.factor(logr_prediction$pred_class)
logr_predictions <- logr_predictions %>% mutate(pred_No = 1-Prediction)
logr_conf_mat <- conf_mat(logr_prediction, truth = Vaccine, estimate = pred_class)
logr_conf_mat
```

```
##            Truth
## Prediction  No Yes
##         No   21  10
##         Yes  24 114
```

```
summary(logr_conf_mat, event_level = "second")
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.799
##  2 kap                  binary         0.428
##  3 sens                 binary         0.919
##  4 spec                 binary         0.467
##  5 ppv                  binary         0.826
##  6 npv                  binary         0.677
```

```
##  7 mcc                  binary         0.441
##  8 j_index              binary         0.386
##  9 bal_accuracy         binary         0.693
## 10 detection_prevalence binary         0.817
## 11 precision            binary         0.826
## 12 recall               binary         0.919
## 13 f_meas               binary         0.870
```

Part h (Code: 2.5 pts; Computation: 1 pt)

Fit a naive Bayes model on the training set predicting `Vaccine` from `COVID_Risk`, `Trust_in_Scientists`, and `Misinformation` and obtain predictions for the holdout set. You can use either version from lab (using the `naiveBayes` function in the e1071 package) or the tidymodels version (using the klaR and discrim packages).

Obtain the confusion matrix for this model on the holdout set. Using the confusion matrix, estimate the accuracy, sensitivity (recall), specificity, positive predictive value (precision), and negative predictive value for the naive Bayes model.

Confirm your estimates by summarizing the confusion matrix again. Remember to use the argument `event_level = "second"` in the summary function because we want to predict whether someone *will* get a vaccine.

```
nb_model <- naive_Bayes(mode = "classification", engine = "klaR")

nb_wflow <- workflow() %>%
  add_model(nb_model)
```

```
nb_recipe <- recipe(
  Vaccine ~ Misinformation + Trust_in_Scientists + COVID_Risk,
  data = misinfo_train
) %>% step_dummy(all_nominal_predictors())

nb_wflow <- nb_wflow %>%
  add_recipe(nb_recipe)
```

```
nb_fit <- fit(nb_wflow, data = misinfo_train)
```

```
predictions_nb_df <- broom::augment(nb_fit, new_data = misinfo_test)
predictions_nb_df %>% dplyr::select(
  Vaccine,
  .pred_class,
  .pred_Yes,
  .pred_No)
```

```
## # A tibble: 169 x 4
##    Vaccine .pred_class .pred_Yes .pred_No
##    <fct>   <fct>           <dbl>    <dbl>
##  1 Yes     Yes             0.982   0.0176
##  2 Yes     Yes             0.823   0.177
##  3 Yes     Yes             0.864   0.136
##  4 Yes     Yes             0.858   0.142
##  5 No      Yes             0.683   0.317
##  6 No      No              0.387   0.613
##  7 No      No              0.217   0.783
##  8 Yes     Yes             0.873   0.127
##  9 No      Yes             0.745   0.255
```

```
## 10 No      No              0.395   0.605
## # ... with 159 more rows
```

```
nb_conf_mat <- conf_mat(predictions_nb_df, truth = Vaccine, estimate = .pred_class)
nb_conf_mat
```

```
##           Truth
## Prediction  No Yes
##        No   27  15
##        Yes  18 109
```

accuracy: $136/169 = 0.80$

sensitivity: $TP/TP+FN = 109/124 = 0.88$

specificity: $TN/TN+FP = 27/45 = 0.6$

ppv: $TP/TP+FP = 109/127 = 0.86$

npv: $TN/TN+FN = 27/42 = 0.64$

```
summary(nb_conf_mat, event_level = "second")
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.805
##  2 kap                  binary         0.489
##  3 sens                 binary         0.879
##  4 spec                 binary         0.6
##  5 ppv                  binary         0.858
##  6 npv                  binary         0.643
##  7 mcc                  binary         0.490
##  8 j_index              binary         0.479
##  9 bal_accuracy         binary         0.740
## 10 detection_prevalence binary         0.751
## 11 precision            binary         0.858
## 12 recall               binary         0.879
## 13 f_meas               binary         0.869
```

Part i (Code: 0.5 pts; Explanation: 1 pt)

Obtain the correlation matrix and vif for the predictors in the logistic regression model. Using your results, argue that the major assumption of naive Bayes is reasonably justified with these predictors.

```
vif(logr_misinfo)
```

```
##          COVID_Risk Trust_in_Scientists      Misinformation
##            1.057062            1.080632            1.061278
```

```
misinformation2 %>%
  dplyr::select(Misinformation, Trust_in_Scientists, COVID_Risk) %>%
  cor()
```

```
##                     Misinformation Trust_in_Scientists COVID_Risk
## Misinformation           1.0000000          -0.3118322 -0.1557215
## Trust_in_Scientists     -0.3118322           1.0000000  0.3706794
## COVID_Risk              -0.1557215           0.3706794  1.0000000
```

Since our VIF is below 5 and there are not high amounts of correlation between each group we can reasonably assume `Trust_in_Scientists`, `COVID_Risk`, and `Misinformation` are independent.

Part j (Code: 2.5 pts; Computation: 1 pt)

Fit a linear discriminant analysis (LDA) model on the training set predicting `Vaccine` from `COVID_Risk`, `Trust_in_Scientists`, and `Misinformation` and obtain predictions for the holdout set. You should use the `lda` function in the MASS package but can use it either by itself (as shown in the lab) or as the "engine" in the tidymodels workflow.

Obtain the confusion matrix for this model on the holdout set. Using the confusion matrix, estimate the accuracy, sensitivity (recall), specificity, positive predictive value (precision), and negative predictive value for the LDA model.

Confirm your estimates by summarizing the confusion matrix again. Remember to use the argument `event_level = "second"` in the summary function because we want to predict whether someone *will* get a vaccine.

```
lda_model <- discrim_linear(mode = "classification", engine = "MASS")

lda_wflow <- workflow() %>%
  add_model(lda_model)
```

```
lda_wflow <- lda_wflow %>%
  add_recipe(nb_recipe)
```

```
lda_fit <- fit(lda_wflow, data = misinfo_train)
```

```
predictions_lda_df <- broom::augment(lda_fit, new_data = misinfo_test)
predictions_lda_df %>% dplyr::select(
  Vaccine,
  .pred_class,
  .pred_Yes,
  .pred_No)
```

```
## # A tibble: 169 x 4
##    Vaccine .pred_class .pred_Yes .pred_No
##    <fct>   <fct>           <dbl>    <dbl>
##  1 Yes     Yes             0.961   0.0388
##  2 Yes     Yes             0.843   0.157
##  3 Yes     Yes             0.886   0.114
##  4 Yes     Yes             0.757   0.243
##  5 No      Yes             0.748   0.252
##  6 No      Yes             0.551   0.449
##  7 No      No              0.339   0.661
##  8 Yes     Yes             0.862   0.138
##  9 No      Yes             0.633   0.367
## 10 No      Yes             0.569   0.431
## # ... with 159 more rows
```

```
lda_conf_mat <- conf_mat(predictions_lda_df, truth = Vaccine, estimate = .pred_class)
lda_conf_mat
```

```
##           Truth
## Prediction  No Yes
##        No   21  11
##        Yes  24 113
```

accuracy: 134/169 = 0.79

sensitivity: TP/TP+FN = 113/124 = 0.91

specificity: TN/TN+FP = 21/45 = 0.47

ppv: TP/TP+FP = 113/137 = 0.82

npv: TN/TN+FN = 21/32 = 0.66

```
summary(lda_conf_mat, event_level = "second")
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.793
##  2 kap                  binary         0.416
##  3 sens                 binary         0.911
##  4 spec                 binary         0.467
##  5 ppv                  binary         0.825
##  6 npv                  binary         0.656
##  7 mcc                  binary         0.426
##  8 j_index              binary         0.378
##  9 bal_accuracy         binary         0.689
## 10 detection_prevalence binary         0.811
## 11 precision            binary         0.825
## 12 recall               binary         0.911
## 13 f_meas               binary         0.866
```

Part k (Code: 3 pts)

For each of the four models, obtain the Matthews Correlation Coefficient, (mean) log-loss, and Brier score.

```
summary(knn_conf_mat, event_level = "second") %>% filter(.metric == "mcc")
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 mcc     binary         0.455
```

```
brier_knn <- predictions_knn_df %>% mutate(
  squared_error = case_when(
    .pred_class == "Yes" ~ (1 - .pred_Yes)^2,
    .pred_class == "No" ~ (1 - .pred_No)^2
  )
)
mean(brier_knn$squared_error)
```

```
## [1] 0.05404819
```

```
mn_log_loss(predictions_knn_df,
            truth = Vaccine,
            .pred_Yes,
            event_level = "second"
)
```

```
## # A tibble: 1 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
```

```
## 1 mn_log_loss binary          2.12
```

```
summary(logr_conf_mat, event_level = "second") %>% filter(.metric == "mcc")
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 mcc     binary         0.441
```

```
logr_predictions <- logr_predictions %>% mutate(pred_Class = predicted_category)
brier_nb <- logr_predictions %>% mutate(
  squared_error = case_when(
    pred_Class == "Yes" ~ (1 - Prediction)^2,
    pred_Class == "No" ~ (1 - pred_No)^2
  )
)
mean(brier_nb$squared_error)
```

```
## [1] 0.06186946
```

```
mn_log_loss(logr_predictions,
            truth = Vaccine,
            Prediction,
            event_level = "second"
)
```

```
## # A tibble: 1 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 mn_log_loss binary         0.472
```

```
summary(nb_conf_mat, event_level = "second") %>% filter(.metric == "mcc")
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 mcc     binary         0.490
```

```
brier_nb <- predictions_nb_df %>% mutate(
  squared_error = case_when(
    .pred_class == "Yes" ~ (1 - .pred_Yes)^2,
    .pred_class == "No" ~ (1 - .pred_No)^2
  )
)
mean(brier_nb$squared_error)
```

```
## [1] 0.04853135
```

```
mn_log_loss(predictions_nb_df,
            truth = Vaccine,
            .pred_Yes,
            event_level = "second"
)
```

```
## # A tibble: 1 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 mn_log_loss binary         0.481
```

```r
summary(lda_conf_mat, event_level = "second") %>% filter(.metric == "mcc")
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 mcc     binary         0.426
```

```r
brier_lda <- predictions_lda_df %>% mutate(
  squared_error = case_when(
    .pred_class == "Yes" ~ (1 - .pred_Yes)^2,
    .pred_class == "No" ~ (1 - .pred_No)^2
  )
)
mean(brier_lda$squared_error)
```

```
## [1] 0.05847291
```

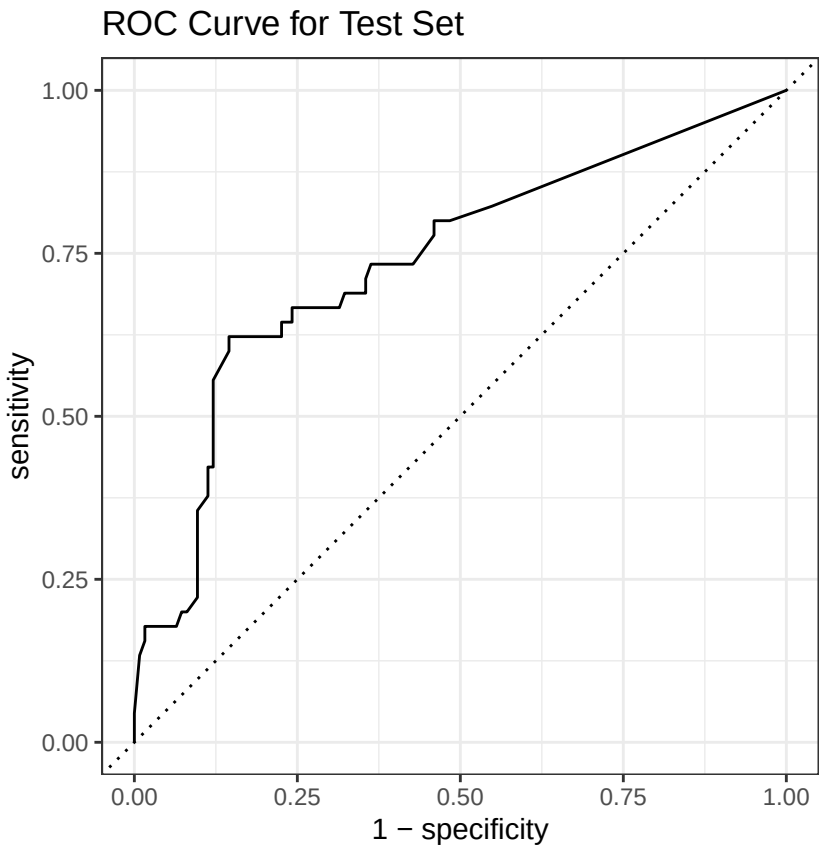
```r
mn_log_loss(predictions_lda_df,
            truth = Vaccine,
            .pred_Yes,
            event_level = "second"
)
```

```
## # A tibble: 1 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 mn_log_loss binary         0.475
```

Part l (Code: 2 pts)

For each of the four models, produce a plot of the receiver operating characteristic (ROC) curve, and obtain the area under the curve (AUC).

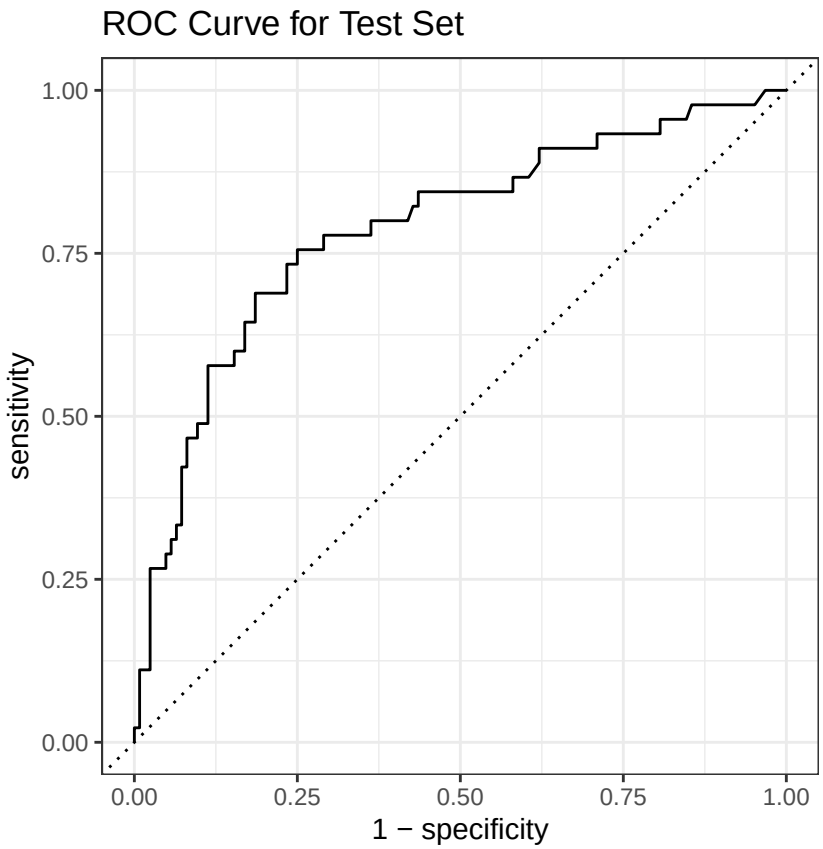
```r
# Construct the ROC curve
roc_tibble_knn <- roc_curve(predictions_knn_df, truth = Vaccine, .pred_No)

# Plot the ROC curve
autoplot(roc_tibble_knn) + labs(title = "ROC Curve for Test Set")
```




```
roc_auc(predictions_knn_df, truth = Vaccine, .pred_No)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.738
```

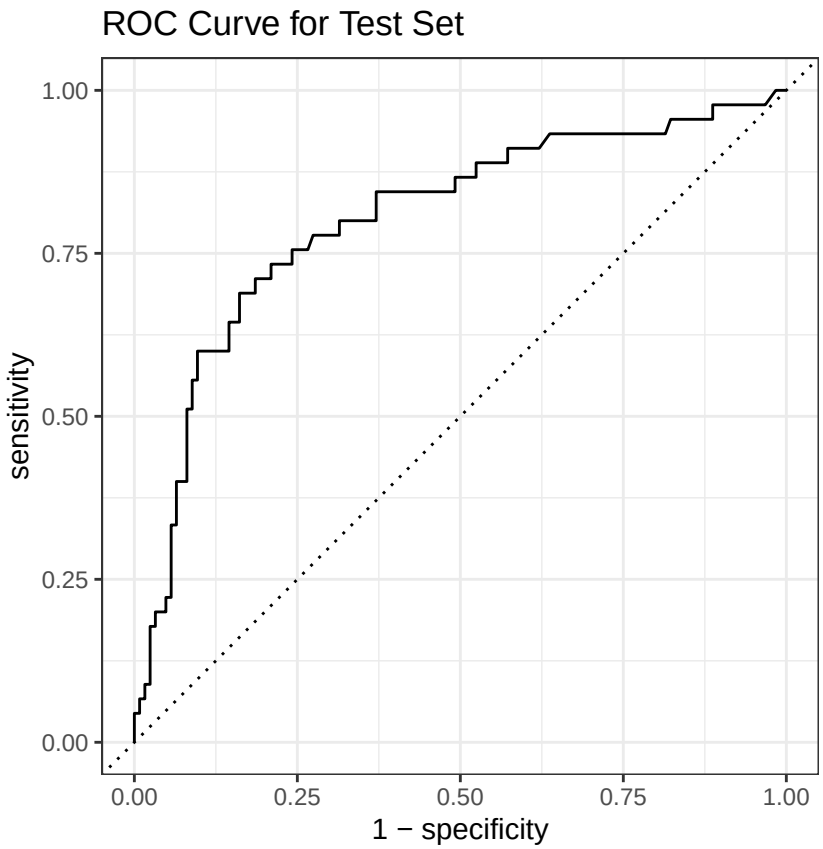
```
# Construct the ROC curve
roc_tibble_logr <- roc_curve(logr_predictions, truth = Vaccine, pred_No)

# Plot the ROC curve
autoplot(roc_tibble_logr) + labs(title = "ROC Curve for Test Set")
```




```
roc_auc(logr_predictions, truth = Vaccine, pred_No)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.787
```

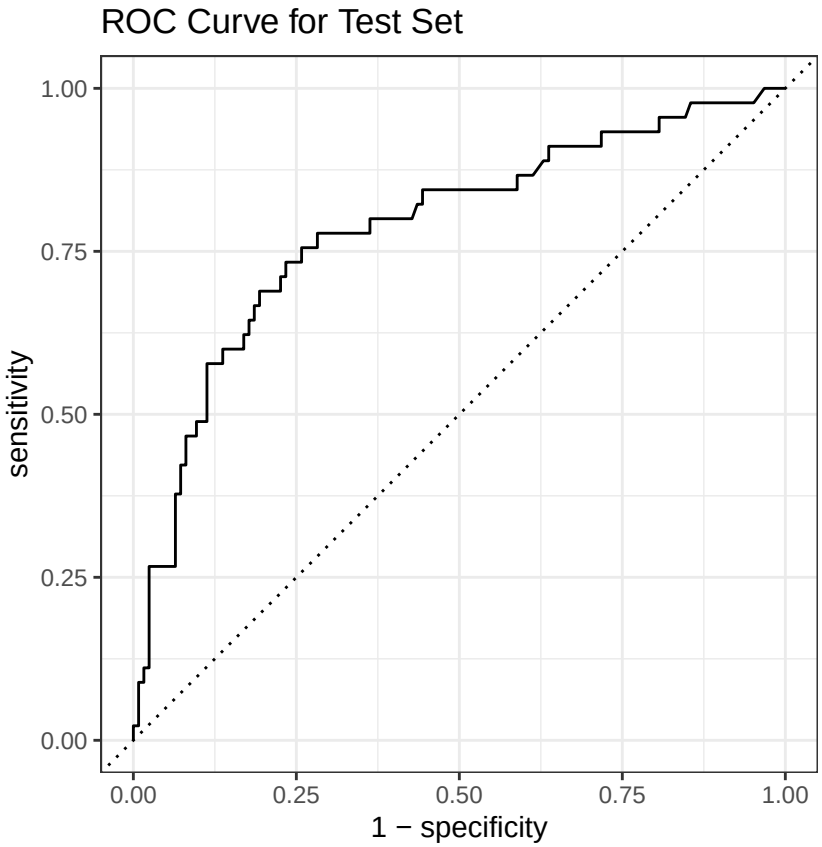
```
# Construct the ROC curve
roc_tibble_nb <- roc_curve(predictions_nb_df, truth = Vaccine, .pred_No)

# Plot the ROC curve
autoplot(roc_tibble_nb) + labs(title = "ROC Curve for Test Set")
```




```
roc_auc(predictions_nb_df, truth = Vaccine, .pred_No)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.802
```

```
# Construct the ROC curve
roc_tibble_lda <- roc_curve(predictions_lda_df, truth = Vaccine, .pred_No)

# Plot the ROC curve
autoplot(roc_tibble_lda) + labs(title = "ROC Curve for Test Set")
```




```
roc_auc(predictions_lda_df, truth = Vaccine, .pred_No)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.785
```

Part m (Explanation: 1.5 pts)

Which of the four models you fit (k-nn, logistic regression, naive Bayes, or LDA) would you argue is the "best" model for predicting whether or not someone would get a COVID-19 vaccine? Justify your answer based on a metric *other* than accuracy.

Based on the Brier score for each model, the naive Bayes model is the "best" model. This model has the lowest Brier score compared to the others at 0.049.