Lab Assignment #6

Math 437 - Modern Data Analysis

Due March 15, 2023

Instructions

The purpose of this lab is to introduce a few different classification strategies. In this lab we will work with k-nearest neighbors, logistic regression, and their generalizations to more than 2 response categories.

```
library(ISLR2)
library(ggplot2)
library(dplyr)
library(class) # knn
library(nycflights13) # Problem 3
library(nnet) # multinomial logistic regression
library(workflows)
library(parsnip)
library(recipes)
library(rsample)
library(broom)
library(yardstick)
```

This lab assignment is worth a total of **20 points**.

Problem 1: Example Code

Part a (Code: 1.5 pts)

Run the code in ISLR Labs 4.7.1, 4.7.2, and 4.7.6. Put each chunk from the textbook in its own chunk.

4.7.1

```
names(Smarket)
```

```
## [1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"
## [7] "Volume"    "Today"     "Direction"
```

```
dim(Smarket)
```

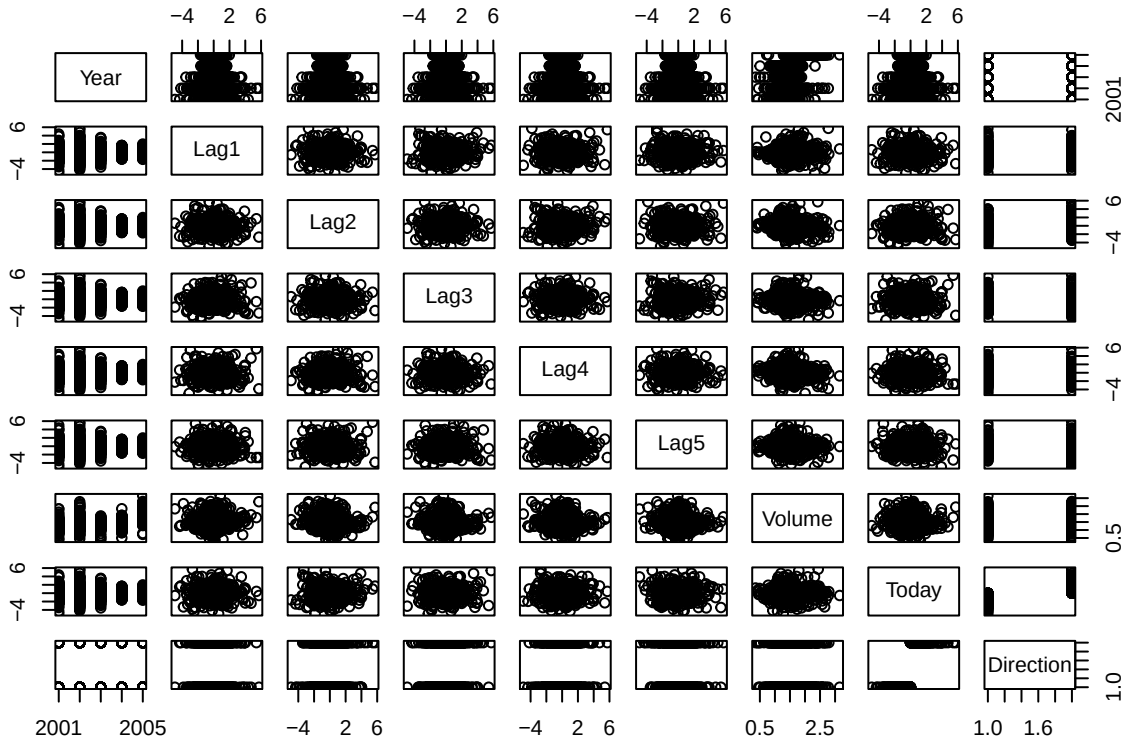
```
## [1] 1250    9
```

```
summary(Smarket)
```

```
##       Year           Lag1                Lag2                Lag3
##  Min.   :2001   Min.   :-4.922000   Min.   :-4.922000   Min.   :-4.922000
##  1st Qu.:2002   1st Qu.:-0.639500   1st Qu.:-0.639500   1st Qu.:-0.640000
##  Median :2003   Median : 0.039000   Median : 0.039000   Median : 0.038500
##  Mean   :2003   Mean   : 0.003834   Mean   : 0.003919   Mean   : 0.001716
##  3rd Qu.:2004   3rd Qu.: 0.596750   3rd Qu.: 0.596750   3rd Qu.: 0.596750
```

```
##  Max.   :2005   Max.   : 5.733000   Max.   : 5.733000   Max.   : 5.733000
##       Lag4                Lag5              Volume           Today
##  Min.   :-4.922000   Min.   :-4.92200   Min.   :0.3561   Min.   :-4.922000
##  1st Qu.:-0.640000   1st Qu.:-0.64000   1st Qu.:1.2574   1st Qu.:-0.639500
##  Median : 0.038500   Median : 0.03850   Median :1.4229   Median : 0.038500
##  Mean   : 0.001636   Mean   : 0.00561   Mean   :1.4783   Mean   : 0.003138
##  3rd Qu.: 0.596750   3rd Qu.: 0.59700   3rd Qu.:1.6417   3rd Qu.: 0.596750
##  Max.   : 5.733000   Max.   : 5.73300   Max.   :3.1525   Max.   : 5.733000
##  Direction
##  Down:602
##  Up  :648
##
##
##
##
```

```
pairs(Smarket)
```



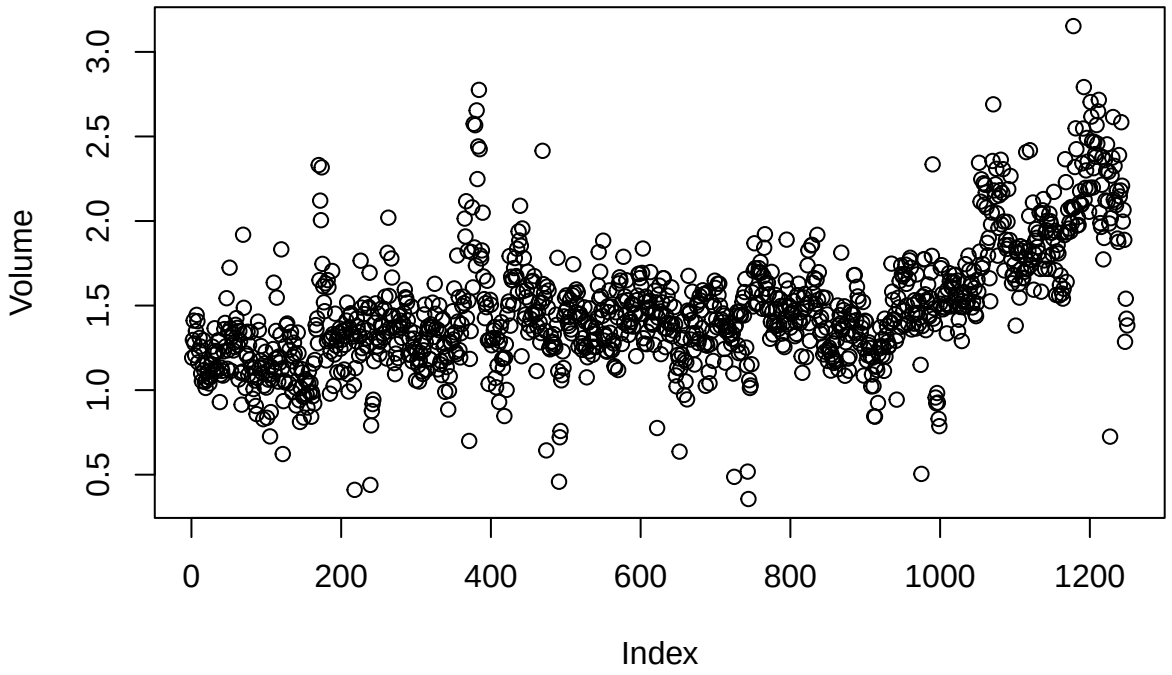

```
cor(Smarket)
```

```
cor(Smarket[, -9])
```

```
##              Year         Lag1         Lag2         Lag3         Lag4
## Year   1.00000000  0.029699649  0.030596422  0.033194581  0.035688718
## Lag1   0.02969965  1.000000000 -0.026294328 -0.010803402 -0.002985911
## Lag2   0.03059642 -0.026294328  1.000000000 -0.025896670 -0.010853533
## Lag3   0.03319458 -0.010803402 -0.025896670  1.000000000 -0.024051036
## Lag4   0.03568872 -0.002985911 -0.010853533 -0.024051036  1.000000000
## Lag5   0.02978799 -0.005674606 -0.003557949 -0.018808338 -0.027083641
## Volume 0.53900647  0.040909908 -0.043383215 -0.041823686 -0.048414246
## Today  0.03009523 -0.026155045 -0.010250033 -0.002447647 -0.006899527
##                Lag5      Volume        Today
```

```
## Year    0.029787995  0.53900647  0.030095229
## Lag1   -0.005674606  0.04090991 -0.026155045
## Lag2   -0.003557949 -0.04338321 -0.010250033
## Lag3   -0.018808338 -0.04182369 -0.002447647
## Lag4   -0.027083641 -0.04841425 -0.006899527
## Lag5    1.000000000 -0.02200231 -0.034860083
## Volume -0.022002315  1.00000000  0.014591823
## Today  -0.034860083  0.01459182  1.000000000
```

```
attach(Smarket)
plot(Volume)
```



4.7.2 Logistic Regression

```
glm.fits <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
                data = Smarket, family = binomial)
summary(glm.fits)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Smarket)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.446  -1.203   1.065   1.145   1.326
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.126000   0.240736  -0.523    0.601
## Lag1        -0.073074   0.050167  -1.457    0.145
## Lag2        -0.042301   0.050086  -0.845    0.398
## Lag3         0.011085   0.049939   0.222    0.824
## Lag4         0.009359   0.049974   0.187    0.851
```

```
## Lag5         0.010313   0.049511   0.208    0.835
## Volume       0.135441   0.158360   0.855    0.392
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1731.2  on 1249  degrees of freedom
## Residual deviance: 1727.6  on 1243  degrees of freedom
## AIC: 1741.6
##
## Number of Fisher Scoring iterations: 3
```

```
coef(glm.fits)
```

```
##  (Intercept)         Lag1         Lag2         Lag3         Lag4         Lag5
## -0.126000257 -0.073073746 -0.042301344  0.011085108  0.009358938  0.010313068
##       Volume
##  0.135440659
```

```
summary(glm.fits)$coef
```

```
##                 Estimate Std. Error    z value  Pr(>|z|)
## (Intercept) -0.126000257 0.24073574 -0.5233966 0.6006983
## Lag1        -0.073073746 0.05016739 -1.4565986 0.1452272
## Lag2        -0.042301344 0.05008605 -0.8445733 0.3983491
## Lag3         0.011085108 0.04993854  0.2219750 0.8243333
## Lag4         0.009358938 0.04997413  0.1872757 0.8514445
## Lag5         0.010313068 0.04951146  0.2082966 0.8349974
## Volume       0.135440659 0.15835970  0.8552723 0.3924004
```

```
summary(glm.fits)$coef[, 4]
```

```
## (Intercept)        Lag1        Lag2        Lag3        Lag4        Lag5
##   0.6006983   0.1452272   0.3983491   0.8243333   0.8514445   0.8349974
##      Volume
##   0.3924004
```

```
glm.probs <- predict(glm.fits, type = "response")
glm.probs[1:10]
```

```
##         1         2         3         4         5         6         7         8
## 0.5070841 0.4814679 0.4811388 0.5152224 0.5107812 0.5069565 0.4926509 0.5092292
##         9        10
## 0.5176135 0.4888378
```

```
contrasts(Direction)
```

```
##      Up
## Down  0
## Up    1
```

```
glm.pred <- rep("Down", 1250)
glm.pred[glm.probs > .5] = "Up"
```

```
table(glm.pred, Direction)
```

```
##         Direction
## glm.pred Down  Up
##     Down  145 141
##     Up    457 507
```

```
mean(glm.pred == Direction)
```

```
## [1] 0.5216
```

```
train <- (Year < 2005)

Smarket.2005 <- Smarket[!train, ]
dim(Smarket.2005)
```

```
## [1] 252   9
```

```
Direction.2005 <- Direction[!train]
```

```
glm.fits <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
                data = Smarket, family = binomial, subset = train)

glm.probs <- predict(glm.fits, Smarket.2005, type = "response")
```

```
glm.pred <- rep("Down", 252)

glm.pred[glm.probs > .5] <- "Up"

table(glm.pred, Direction.2005)
```

```
##         Direction.2005
## glm.pred Down Up
##     Down   77 97
##     Up     34 44
```

```
mean(glm.pred == Direction.2005)
```

```
## [1] 0.4801587
```

```
mean(glm.pred != Direction.2005)
```

```
## [1] 0.5198413
```

```
glm.fits <- glm(Direction ~ Lag1 + Lag2, data = Smarket, family = binomial,
                subset = train)

glm.probs <- predict(glm.fits, Smarket.2005, type = "response")

glm.pred <- rep("Down", 252)

glm.pred[glm.probs > .5] <- "Up"

table(glm.pred, Direction.2005)
```

```
##         Direction.2005
## glm.pred Down  Up
##     Down   35  35
##     Up     76 106
```

```
mean(glm.pred == Direction.2005)
```

```
## [1] 0.5595238
```

```
106/ (106 + 76)
```

```
## [1] 0.5824176
```

```
predict(glm.fits, newdata = data.frame(Lag1 = c(1.2, 1.5), Lag2 = c(1.1, -0.8)),
        type = "response")
```

```
##         1         2
## 0.4791462 0.4960939
```

4.7.6 K-Nearest Neighbors

```
train.X <- cbind(Lag1, Lag2)[train, ]
test.X <- cbind(Lag1, Lag2)[!train, ]
train.Direction <- Direction[train]
```

```
set.seed(1)
knn.pred <- knn(train.X, test.X, train.Direction, k = 1)

table(knn.pred, Direction.2005)
```

```
##         Direction.2005
## knn.pred Down Up
##     Down   43 58
##     Up     68 83
```

```
(83 + 43) / 252
```

```
## [1] 0.5
```

```
knn.pred <- knn(train.X, test.X, train.Direction, k = 3)
table(knn.pred, Direction.2005)
```

```
##         Direction.2005
## knn.pred Down Up
##     Down   48 54
##     Up     63 87
```

```
mean(knn.pred == Direction.2005)
```

```
## [1] 0.5357143
```

```
dim(Caravan)
```

```
## [1] 5822   86
```

```
attach(Caravan)

summary(Purchase)
```

```
##   No  Yes
## 5474  348
```

```
348 / 5822
```

```
## [1] 0.05977327
```

```
standardized.X <- scale(Caravan[, -86])

var(Caravan[, 1])
```

```
## [1] 165.0378
```

```
var(Caravan[, 2])
```

```
## [1] 0.1647078
```

```
var(standardized.X[, 1])
```

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

```
var(standardized.X[, 2])
```

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

```
test <- 1:1000

train.X <- standardized.X[-test, ]
test.X <- standardized.X[test, ]

train.Y <- Purchase[-test]
test.Y <- Purchase[test]

set.seed(1)
knn.pred <- knn(train.X, test.X, train.Y, k = 1)

mean(test.Y != knn.pred)
```

```
## [1] 0.118
```

```
mean(test.Y != "No")
```

```
## [1] 0.059
```

```
table(knn.pred, test.Y)
```

```
##         test.Y
## knn.pred  No Yes
##      No  873  50
##      Yes  68   9
```

```
9 / (68 + 9)
```

```
## [1] 0.1168831
```

```
knn.pred <- knn(train.X, test.X, train.Y, k = 3)
table(knn.pred, test.Y)
```

```
##         test.Y
## knn.pred  No Yes
##      No  920  54
##      Yes  21   5
```

```
5 / 26
```

```
## [1] 0.1923077
```

```
knn.pred <- knn(train.X, test.X, train.Y, k = 5)

table(knn.pred, test.Y)
```

```
##         test.Y
## knn.pred  No Yes
##      No  930  55
##      Yes  11   4
```

```
4 / 15
```

```
## [1] 0.2666667
```

```
glm.fits <- glm(Purchase ~., data = Caravan, family = binomial, subset = -test)
```

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

```
glm.probs <-  predict(glm.fits, Caravan[test, ], type = "response")

glm.pred <- rep("No", 1000)

glm.pred[glm.probs > .5] <- "Yes"

table(glm.pred, test.Y)
```

```
##         test.Y
## glm.pred  No Yes
##      No  934  59
##      Yes   7   0
```

```
glm.pred <- rep("No", 1000)

glm.pred[glm.probs > .25] <- "Yes"

table(glm.pred, test.Y)
```

```
##         test.Y
## glm.pred  No Yes
##      No  919  48
##      Yes  22  11
```

```
11 / (22 + 11)
```

```
## [1] 0.3333333
```

Part b (Explanation: 1 pt)

In Lab 4.7.6, both the k-nn model with `k = 5` and the logistic regression model using `probs > 0.5` as the classification threshold produced a test error rate (misclassification rate) of 6.6% using the `Caravan` dataset. Explain why we would prefer to use the k-nn model.

We may prefer to use a nonparametric method such as KNN so we are not making assumptions about the structure/distribution of our response or predictors.

Part c (Explanation: 1 pt)

If you forget to include the `family = binomial` argument when calling `glm`, what does R try to do instead of running a logistic regression? It may be useful to look up the documentation for the `glm` function, look up the part of the Logistic Regression Class Activity where we actually did this, and/or read Section 4.6.3 of the textbook.

When a family is not specified, R will default to running a linear regression instead.

Problem 2: Analysis of iris Dataset: virginica vs versicolor

The `iris` dataset is a well-known benchmark dataset for evaluating the performance of new classification algorithms on small sets of data. It is found in the `datasets` package (which should automatically be loaded when you start R).

The dataset consists of five variables: the response variable (`Species`) and four explanatory variables (`Sepal.Length`, `Sepal.Width`, `Petal.Length`, and `Petal.Width`, each measured in cm). Because the species are fairly distinct in terms of the four explanatory variables, we should expect near-100% accuracy with any decent classification model.

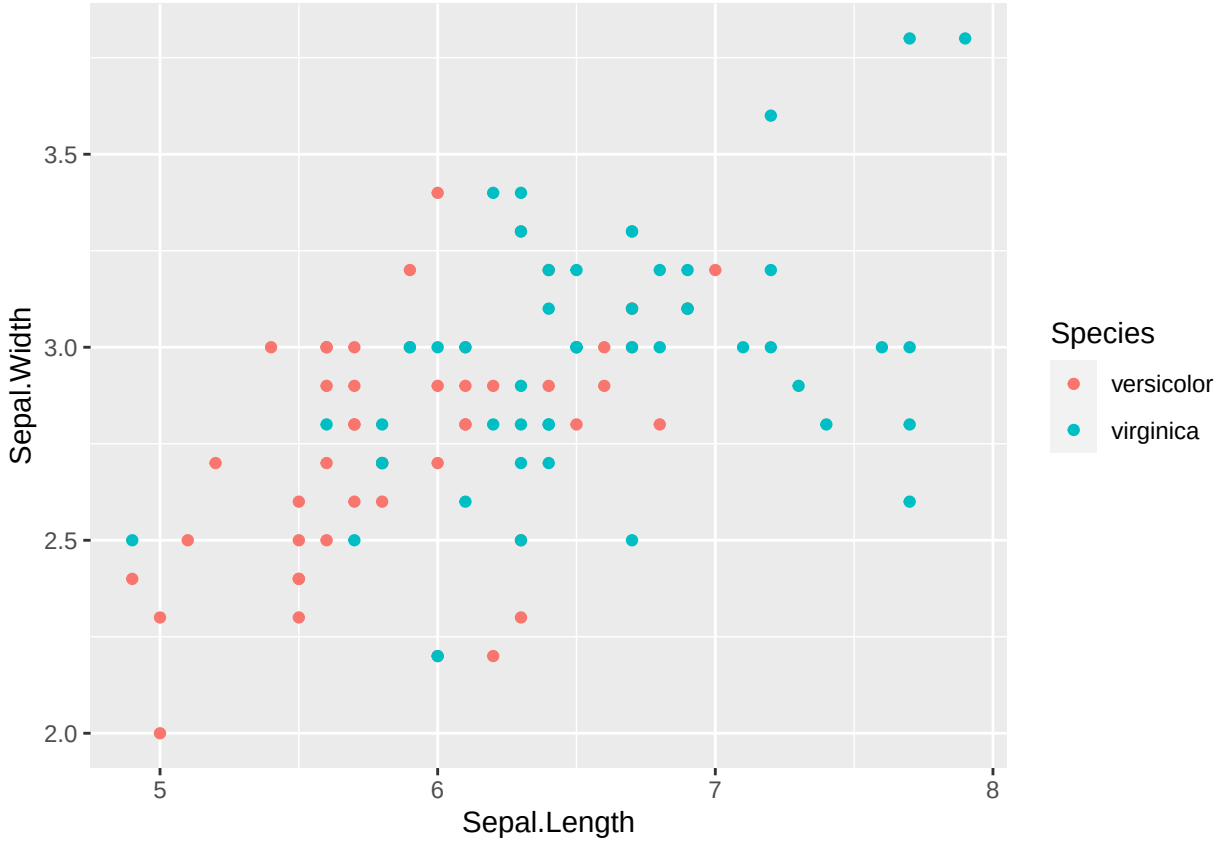
In this problem, we will do all of our modeling without using the `tidymodels` packages.

Part a (Code: 2 pts)

We will start out by working with only the species `virginica` and `versicolor`. Create a new dataset, `iris_vv`, by filtering to only include those two species. Then, pick two of the explanatory variables and, using the `ggplot2` package, create a scatterplot showing the relationship between those two variables, color-coded based on the `Species`.

```
iris_vv <- iris %>%
  filter(Species == "virginica" | Species == "versicolor")

ggplot(iris_vv, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) +
  geom_point()
```



Part b (Code: 2 pts)

Prepare your dataset for the k-nearest neighbors algorithm by doing the following:

Step 1: Create the training and holdout sets

Randomly select 20 observations to be in the holdout set. The remaining observations are in the training set. Call these new datasets `iris.train` and `iris.test`.

HINT: The easiest way to do this is to `sample` from the indices and subset accordingly.

```
set.seed(1003)
n <- nrow(iris_vv)
test.rows <- sample(n, floor(20), replace = FALSE)
iris_vv$Species <- relevel(iris_vv$Species, ref = "virginica")
iris.train <- iris_vv[-test.rows,]
iris.test <- iris_vv[test.rows,]
```

Step 2: Scale the predictor variables

Create two new datasets, `iris.train.X` and `iris.test.X`, containing the explanatory variables in each set. Remember that you need to scale the variables in both the training and holdout sets based on the training set!

```
iris.train.X <- iris.train %>%
  dplyr::select(-Species) %>%
  scale()

iris.test.X <- iris.test %>%
  dplyr::select(-Species) %>%
  mutate(Sepal.Length = (Sepal.Length-mean(iris.train$Sepal.Length))/sd(iris.train$Sepal.Length),
         Sepal.Width = (Sepal.Width-mean(iris.train$Sepal.Width))/sd(iris.train$Sepal.Width),
         Petal.Length = (Petal.Length-mean(iris.train$Petal.Length))/sd(iris.train$Petal.Length),
         Petal.Width = (Petal.Width-mean(iris.train$Petal.Width))/sd(iris.train$Petal.Width)
         )
```

Part c (Code: 2 pts; Explanation: 0.5 pts)

With the `knn` function, use 5-nearest neighbors to predict the species of each observation in the `iris.test` holdout set. Create a `table` showing the predicted and actual species in the holdout set. What is the prediction accuracy of this model?

```
train_response <- iris.train$Species


k1 <- knn(train = iris.train.X, test = iris.test.X, k=5,
          cl = train_response)


table(k1, iris.test$Species)
```

```
##
## k1           virginica versicolor
##   virginica          7          1
##   versicolor         0         12
```

```
mean(k1 == iris.test$Species)
```

```
## [1] 0.95
```

The prediction accuracy of this model is 95%.

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

Fit a logistic regression model on the training set predicting `Species` from the four explanatory variables. Unlike with k-nearest neighbors, you do not need to scale the variables. Note that there may be some

weirdness with figuring out the reference level for `Species`, so you may want to use the `relevel` function to set that before fitting the model.

Use the `summary` or `coef` function to obtain the coefficient estimates, and write out the equation of the fitted logistic regression model.

```
lr1 <- glm(Species ~ ., data = iris.train, family = "binomial")
```

```
## Warning: glm.fit: algorithm did not converge
```

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

```
summary(lr1)
```

```
##
## Call:
## glm(formula = Species ~ ., family = "binomial", data = iris.train)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -5.700e-05  -2.100e-08  -2.100e-08   2.100e-08   5.206e-05
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)     498.12  289051.00   0.002    0.999
## Sepal.Length    131.53   65605.04   0.002    0.998
## Sepal.Width      10.01   50871.28   0.000    1.000
## Petal.Length   -218.30   91154.63  -0.002    0.998
## Petal.Width    -174.46   79647.95  -0.002    0.998
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1.1045e+02  on 79  degrees of freedom
## Residual deviance: 1.2418e-08  on 75  degrees of freedom
## AIC: 10
##
## Number of Fisher Scoring iterations: 25
```

logit(p) = 498.12 + 131.53(Sepal.Length) + 10.01(Sepal.Width) - 218.3(Petal.Length) - 174.46(Petal.Width)

where p = probability of being a versicolor iris

Part e (Code: 2 pts; Explanation: 0.5 pts)

Use the `predict` function to obtain predictions for the `iris.test` holdout set. Remember to include the argument `type = "response"` to output predictions as probabilities!

Predict the class of each flower in the holdout set using a decision boundary of 0.5 (i.e., if the predicted probability of being in *versicolor* vs. *virginica* is above 0.5, predict the flower to be in *versicolor*). Create a `table` showing the predicted and actual species in the holdout set. What is the prediction accuracy of this model on the holdout set?

```
lr1_pred <- predict(lr1, newdata = iris.test, type = "response")

logtbl <- rep("Virginica", 20)
logtbl[lr1_pred > .5] = "Versicolor"

table(logtbl, iris.test$Species)
```

```
##
## logtbl       virginica versicolor
##   Versicolor         0         10
##   Virginica          7          3
```

The prediction accuracy of this model is 0.85.

Problem 3: Analysis of flights Dataset

There are three major airports that serve the New York City metro area: John F. Kennedy International Airport (JFK), Newark Liberty International Airport (EWR), and LaGuardia Airport (LGA). Newark is a United Airlines hub, JFK is an American Airlines hub, while Delta has hubs at both JFK and LGA.

The `flights` dataset contains flights from 2013 that departed one of those three airports. Here we filter the dataset to include only flights that departed on one of those three carriers.

```
flights2 <- flights %>% filter(
  carrier %in% c("UA", "AA", "DL"),
  !is.na(dep_delay)
) %>%
  mutate(origin = as.factor(origin))
```

Let's try to predict which airport a flight departed from into based on the carrier as well as the distance the flight has covered and the departure delay. In this problem we'll do everything the `tidymodels` way.

Part a (Code: 1 pt)

First, we'll use 3-nearest neighbors to predict the departure airport of each observation in the `flights_test` holdout set.

```
knn_model <- nearest_neighbor(mode = "classification", neighbors = 3,
                              dist_power = 2)

set.seed(2222)
flights_split <- initial_split(flights2, prop = 0.8)
flights_train <- training(flights_split)
flights_test <- testing(flights_split)

knn_prep <- recipe(
  origin ~ carrier + distance + dep_delay, # response ~ predictors
  data = flights_train) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_normalize(all_numeric_predictors()) # center and scale numeric predictors

flights_knn <- workflow() %>%
  add_model(knn_model) %>%
  add_recipe(knn_prep)
```

In the chunk below, write code that will `fit` the model on the training set. (If your model runs for ~3 seconds without an error message, start on part (b) while you wait for the chunk to churn.)

```
logr_fit <- fit(flights_knn, data = flights_train)
```

Next, use the `augment` function to create the tibble `flights_knn_predictions` containing the predictions for each flight in the `flights_test` dataset.

```r
predictions_df <- broom::augment(logr_fit, new_data = flights_test,
                                 type = "prob")
summary(predictions_df)
```

```
##       year          month             day           dep_time    sched_dep_time
##  Min.   :2013   Min.   : 1.000   Min.   : 1.00   Min.   :   4   Min.   : 500
##  1st Qu.:2013   1st Qu.: 4.000   1st Qu.: 8.00   1st Qu.: 857   1st Qu.: 900
##  Median :2013   Median : 7.000   Median :16.00   Median :1358   Median :1350
##  Mean   :2013   Mean   : 6.537   Mean   :15.75   Mean   :1328   Mean   :1318
##  3rd Qu.:2013   3rd Qu.:10.000   3rd Qu.:23.00   3rd Qu.:1730   3rd Qu.:1719
##  Max.   :2013   Max.   :12.000   Max.   :31.00   Max.   :2400   Max.   :2355
##
##    dep_delay          arr_time    sched_arr_time   arr_delay
##  Min.   :-20.000   Min.   :   1   Min.   :   1   Min.   :-75.000
##  1st Qu.: -5.000   1st Qu.:1124   1st Qu.:1140   1st Qu.:-20.000
##  Median : -1.000   Median :1601   Median :1618   Median : -8.000
##  Mean   :  9.948   Mean   :1537   Mean   :1562   Mean   :  1.627
##  3rd Qu.:  7.000   3rd Qu.:1956   3rd Qu.:1959   3rd Qu.:  9.000
##  Max.   :911.000   Max.   :2400   Max.   :2359   Max.   :915.000
##                    NA's   :21                    NA's   :85
##    carrier             flight         tailnum          origin
##  Length:27567       Min.   :   1.0   Length:27567       EWR:10777
##  Class :character   1st Qu.: 463.5   Class :character   JFK: 7707
##  Mode  :character   Median :1115.0   Mode  :character   LGA: 9083
##                     Mean   :1089.9
##                     3rd Qu.:1635.0
##                     Max.   :2599.0
##
##      dest              air_time        distance         hour
##  Length:27567       Min.   : 25.0   Min.   : 116   Min.   : 5.00
##  Class :character   1st Qu.:126.0   1st Qu.: 762   1st Qu.: 9.00
##  Mode  :character   Median :163.0   Median :1096   Median :13.00
##                     Mean   :194.2   Mean   :1394   Mean   :12.93
##                     3rd Qu.:283.0   3rd Qu.:2133   3rd Qu.:17.00
##                     Max.   :695.0   Max.   :4963   Max.   :23.00
##                     NA's   :85
##      minute        time_hour                      .pred_class   .pred_EWR
##  Min.   : 0.00   Min.   :2013-01-01 05:00:00.0   EWR:10739   Min.   :0.0000
##  1st Qu.: 5.00   1st Qu.:2013-04-04 07:00:00.0   JFK: 7685   1st Qu.:0.0000
##  Median :25.00   Median :2013-07-02 14:00:00.0   LGA: 9143   Median :0.0000
##  Mean   :25.37   Mean   :2013-07-02 21:42:56.2               Mean   :0.3894
##  3rd Qu.:44.00   3rd Qu.:2013-10-01 09:00:00.0               3rd Qu.:1.0000
##  Max.   :59.00   Max.   :2013-12-31 20:00:00.0               Max.   :1.0000
##
##    .pred_JFK        .pred_LGA
##  Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :0.0000
##  Mean   :0.2779   Mean   :0.3326
##  3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000
##
```

```
predictions_df %>%
  dplyr::select(origin, .pred_class)
```

```
## # A tibble: 27,567 x 2
##    origin .pred_class
##    <fct>  <fct>
##  1 EWR    EWR
##  2 LGA    LGA
##  3 JFK    JFK
##  4 EWR    EWR
##  5 JFK    JFK
##  6 JFK    JFK
##  7 LGA    LGA
##  8 LGA    LGA
##  9 JFK    JFK
## 10 JFK    JFK
## # ... with 27,557 more rows
```

Part b (Explanation: 1 pt)

While you wait for your code to run, as best you can, answer the following questions:

- How did we ensure that roughly 20% of the data was in the holdout set, even though we didn't know the exact size of my dataset?

We ensured roughly 20% of the data was in the holdout set by specifying a proportion of 0.8 for the training set and putting the remaining data in the holdout set. The tidymodels function `initial_split` retains a random selection of the data of the proportion specified and the functions `training` and `testing` subset by this selection.

- What does the `step_dummy` function do, and why did we need to do it?

The `step_dummy` function creates a set of binary dummy variables for a factor or character variable. So in our case it is creating additional columns of 0/1 data for ALL of our nominal predictors, which is only the carrier for now. We have `r length(unique(flights$carrier))` different carriers. We have to convert this to binary because tidymodels does not like working with factors (or charachters). ## Part c (Code: 0.5 pts; Explanation: 0.5 pts)

Now let's use multinomial logistic regression to do the same predictions. Multinomial logistic regression requires much less fuss than k-nearest neighbors when it comes to data prep. Remember, since this uses *regression* frameworks, rescaling the variables just messes with intercept and slopes!

The two main functions I use to do this are `VGAM::vglm` and `nnet::multinom`. I find that `vglm` is more generalizable, but it doesn't play nice with `tidymodels`, so we'll use `multinom` instead.

```
# just run this - it all works
multinom_model <- multinom_reg(mode = "classification") # everything else is default

multinom_prep <- recipe(
  origin ~ carrier + distance + dep_delay, # response ~ predictors
  data = flights_train
) %>%
  step_dummy(all_nominal_predictors())

flights_multinom <- workflow() %>%
  add_model(multinom_model) %>%
  add_recipe(multinom_prep)
```

In the chunk below, write code that will `fit` the model on the training set and store the result in the object `flights_multinom_fit`.

```
flights_multinom_fit <- fit(flights_multinom, data = flights_train)
flights_multinom_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: multinom_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 1 Recipe Step
##
## * step_dummy()
##
## -- Model ---------------------------------------------------------------------
## Call:
## nnet::multinom(formula = ..y ~ ., data = data, trace = FALSE)
##
## Coefficients:
##     (Intercept)      distance    dep_delay carrier_DL carrier_UA
## JFK   -1.013889  0.0017241863 -0.001430744 0.38079333  -4.682678
## LGA    2.168939 -0.0005995516 -0.001377484 0.09162532  -3.114253
##
## Residual Deviance: 153646
## AIC: 153666
```

What reference (baseline) level was chosen? Since we did not explicitly choose a baseline level the code defaults to the first option alphabetically which was EWR.

Part d (Code: 0.5 pts)

Getting the coefficients out as a separate R object is a bit of a pain, but we can do it:

```
multinom_coef <- extract_fit_engine(flights_multinom_fit) %>% coef()
print(multinom_coef)
```

```
##     (Intercept)      distance    dep_delay carrier_DL carrier_UA
## JFK   -1.013889  0.0017241863 -0.001430744 0.38079333  -4.682678
## LGA    2.168939 -0.0005995516 -0.001377484 0.09162532  -3.114253
```

To interpret our coefficients, we can usually hack our way to *softmax* coding by adding a zero row (corresponding to the reference/baseline level) at the very top of the coefficient matrix. Then we can look at the effect of a change in the predictor variable on the log-odds of being in any response group relative to any other response group.

```
coef_softmax <- rbind(EWR = numeric(5),
                      multinom_coef)
print(coef_softmax)
```

```
##     (Intercept)      distance    dep_delay carrier_DL carrier_UA
## EWR    0.000000  0.0000000000  0.000000000 0.00000000   0.000000
## JFK   -1.013889  0.0017241863 -0.001430744 0.38079333  -4.682678
## LGA    2.168939 -0.0005995516 -0.001377484 0.09162532  -3.114253
```

Now, for two flights with the same carrier and distance, a one-mile increase in distance is associated with a 0.0017 increase in the log-odds of being from JFK as compared to EWR, and a 0.0006 decrease in the log-odds of being from LGA as compared to EWR. By subtracting the two nonzero slopes, we find that a

one-mile increase in distance is associated with a $0.0017 - (-0.0006) = 0.0023$ increase in the log-odds of being from JFK as compared to LGA.

Inference with a multinomial logistic regression model is possible by calling the `multinom` function directly (tidymodels appears to drop the standard errors from the output), but instead we're going to instead focus on prediction.

If we use the `multinom` function directly, then we can use the `predict` function just like in linear and logistic regression, but we have to use either `class` or `probs` as our `type` argument - we can't use `response` and then pick an arbitrary boundary threshold, since now we have multiple boundaries between different groups.

The `augment` function does away with this ridiculousness and gives us both class predictions and the predicted probability of being in each class without having the remember what argument to use for `type`. Using the `augment` function, create the tibble `flights_multinom_predictions` containing the predictions for each flight in the `flights_test` dataset.

```
flights_multinom_predictions <- augment(flights_multinom_fit,new_data = flights_test ,type = "probs")
```

```
flights_multinom_predictions %>% dplyr::select(
  origin,
  .pred_class,
  .pred_EWR,
  .pred_JFK,
  .pred_LGA
) %>%
  slice(43:45) # equivalent to data[43:45,] but works with a pipe
```

```
## # A tibble: 3 x 5
##   origin .pred_class .pred_EWR .pred_JFK .pred_LGA
##   <fct>  <fct>           <dbl>     <dbl>     <dbl>
## 1 EWR    JFK            0.0522   0.799       0.149
## 2 LGA    JFK            0.116    0.452       0.432
## 3 EWR    EWR            0.792    0.00912     0.199
```

When we have multi-class classification problems, it is possible that no class reaches a 50% probability (e.g., observation 44). In these situations, observations are still classified to the *most likely* airport, we just note that there is a lot more uncertainty about the prediction.

Part e (Code: 1 pt)

Using the `accuracy` function in the `yardstick` package, find the prediction accuracy of each model.

```
accuracy(predictions_df, truth = origin, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.987
```

```
accuracy(flights_multinom_predictions,truth = origin, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.717
```

Part f (Explanation: 1 pt)

Briefly discuss the advantages of using the 3-nearest neighbors algorithm over the multinomial logistic regression model on the `flights` data, and the advantages of using the multinomial logistic regression model over the 3-nearest neighbors algorithm. Prediction accuracy is important, but not the only thing you should compare.

The advantage of KNN is the fact that it better utilizes the data in the sense that we can use it greater for prediction since its' predictions are either 0 or 1. On the other hand multinomial logistic regression is good for the interpretation of slopes and intercepts but does not do as good of a job at prediction with an accuracy of only 72% compared to KNN at 98%.