# Algorithms for Decision Making: Exam 1

Jaime Davila

3/24/22

**Exam 1 Guidelines**

- You have the entire class time (80 minutes) to complete this exam.
- Please make sure to save **immediately** your work in your submit folder. By the end of class time save the last version of your work and **don't** modify it afterwards.
- You are to work completely independently on the exam.
- You are allowed to use your class notes, moodle, worksheets, homeworks, textbooks, plus the "Help" feature from Rstudio.
- You **are not** permitted to do web searches.
- Please silence your cell phone. Place it and any other connected devices in your bag and do not access them for any reason.

For questions that ask for interpretations and explanations, usually no more than a sentence or two is needed for justification. Be thorough, but do not "brain dump". Notice that three sections of this exam are independent and that you can complete later sections successfully whether or not earlier sections are correct.

Do not spend too long on any one question. If you are not sure about an answer, write a note detailing your concern.

**PLEDGE:** I pledge on my honor that I have neither received nor given assistance during this exam nor have I witnessed others receiving assistance, and I have followed the guidelines as described above.

**SIGNATURE:**

◯ I have intentionally not signed the pledge.

## The Weather in Minnesota (40 points)

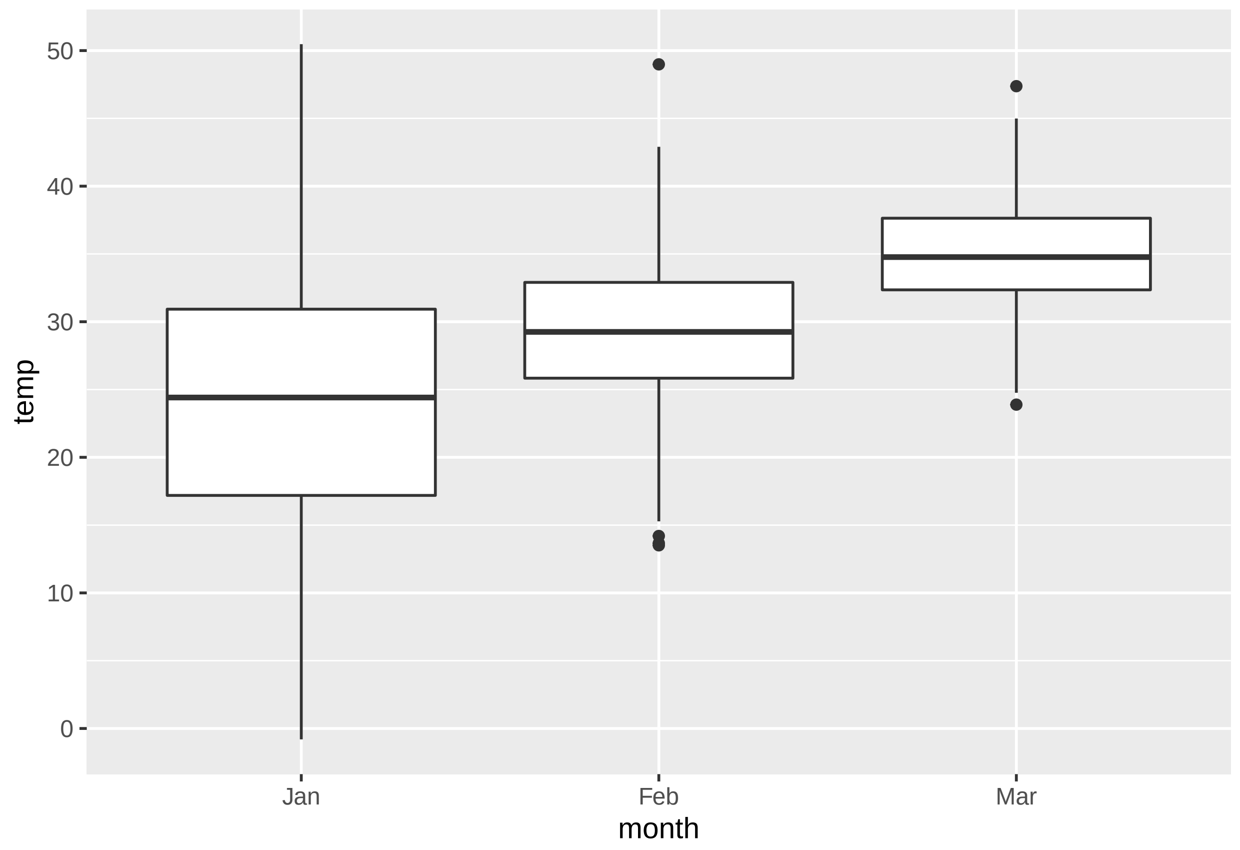
It is spring in Minnesota and we are interested on predicting what month of the year we are living based on the outside temperature. To do that let's load our testing and training datasets

```r
month.levels <-  c("Jan","Feb","Mar")
temp.train.tbl <- read_csv("~/Mscs 341 S22/Class/Data/mn_weather.train.csv") %>%
  mutate(month=factor(month, month.levels))
temp.test.tbl <- read_csv("~/Mscs 341 S22/Class/Data/mn_weather.test.csv") %>%
  mutate(month=factor(month, month.levels))
```

## Question 1 (15 points)

Do a boxplot using `temp.train.tbl` with the distribution of the temperature across the first 3 months of the year. Calculate the mean and standard deviation of the temperature for each of the 3 months. Based on this information, which method would you choose between `lda` and `qda` to predict the month based on the temperature? Justify your answer.

```
temp.train.tbl%>%
  ggplot(aes(x = month, y = temp))+
    geom_boxplot()
```



```
temp.train.tbl%>%
  group_by(month)%>%
  summarize(mean_temp = mean(temp),
            sd_temp = sd(temp))
```

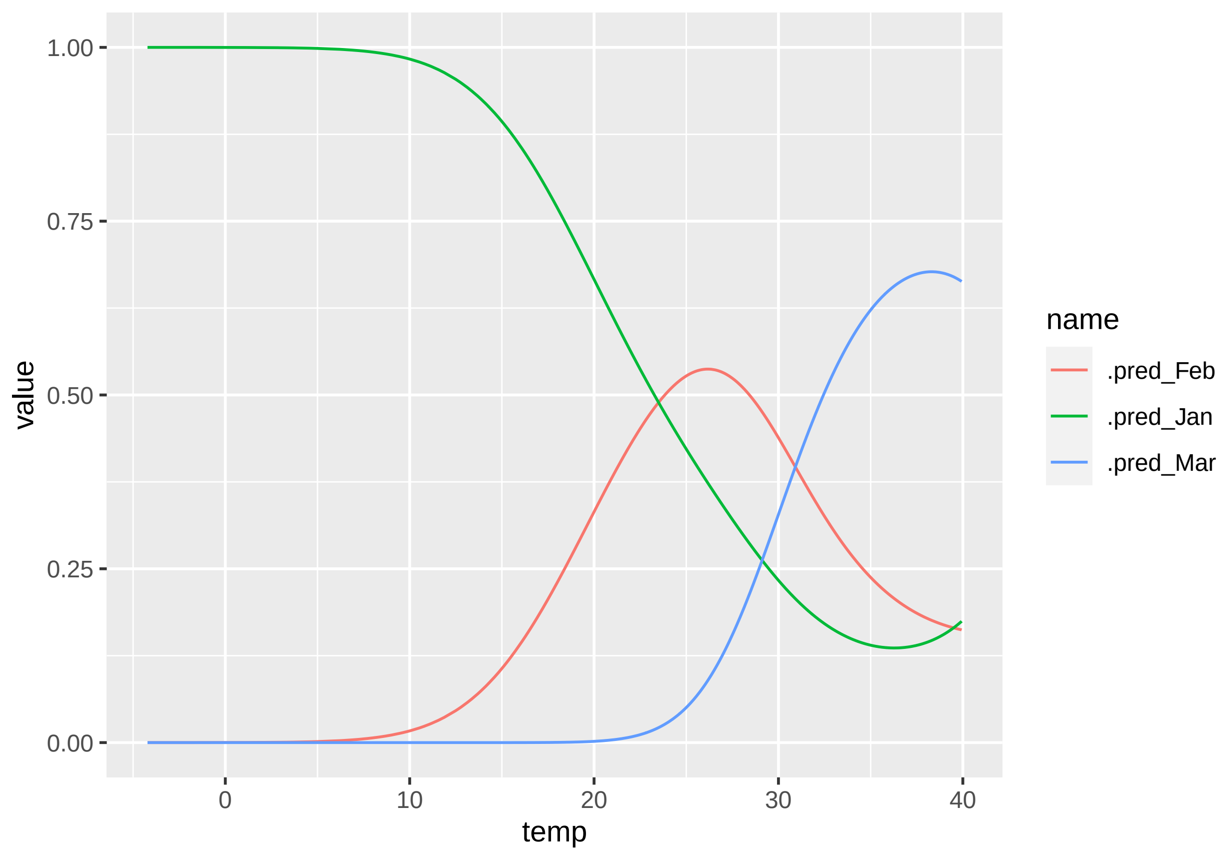
```
## # A tibble: 3 x 3
##   month mean_temp sd_temp
##   <fct>     <dbl>   <dbl>
## 1 Jan        24.2   10.3
## 2 Feb        29.2    5.78
## 3 Mar        34.9    4.01
```

Based on the SD of the temperatures, I would use a QDA model to predict the month based on temperature. One of the assumptions we make when using LDA is that the standard deviations across groups are roughly equal, but here we see that the standard deviation of January is more than double that of March and roughly double that of February. QDA provides with more flexibility to predict the month and does not assume that standard deviations are equal across groups, so it provides a better fit than LDA here.

## Question 2 (15 points)

Using `tidymodels()` create the model you chose in `Question 1` (either `lda` or `qda`) for predicting the month based on the temperature. Using your testing dataset, plot the predicted probability of a temperature corresponding to January, February or March in a single graph. How do you interpret this plot and what are the classification boundaries for each month? Using the information from this plot on which month will your model be making more mistakes?

```
qda.model <- discrim_quad()%>%
  set_engine("MASS")%>%
  set_mode("classification")

recipe <- recipe(month ~ temp, data = temp.train.tbl)

qda.wflow <- workflow()%>%
  add_recipe(recipe)%>%
  add_model(qda.model)

qda.fit <- fit(qda.wflow, temp.train.tbl)

pred.qda.tbl <- qda.fit%>%
  augment(new_data = temp.test.tbl)

pred.qda.tbl%>%
  pivot_longer(4:6)%>%
  ggplot(aes(x = temp, y = value, color = name))+
           geom_line()
```



From the line plot above, we can identiy which month will be predicted at a given temp value. This can be done by looking at a certain temp value and examining which color line is highest. This is the month that will be predicted for that temp value. We see that from 0 to ~23 degrees we will predict January, from ~23 to 31 we will predict February, and from 31 to 40 we will predict March. These are our classification boundaries. Based on this it seems like our model will make the most mistakes on February, because when it predicts February the value (predicted probability) is only ~50%, so there are plenty of occasions where it will guess February because it's in that 23-31 range but in reality March and January have plenty of those days as well.

## Question 3 (10 points)

Calculate the sensitivity and specificity of your model and interpret them in the context of your problem. Based on the confusion matrix, on which month does your model make more mistakes?

```
#Sensitivity = (Number of Yes predicted as Yes) / (Total number of Yes)
#Specificity = (Number of No predicted as No) / (Total number of No)
conf_mat(pred.qda.tbl, month, .pred_class)
```

```
##           Truth
## Prediction Jan Feb Mar
##        Jan 177  57   0
##        Feb  90 134  43
##        Mar  61 110 263
```

```
(177+134+263) / (177+90+61+57+134+110+43+263) #Sensitivity
```

```
## [1] 0.6139037
```

```
sens(pred.qda.tbl, month, .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    macro          0.615
```

```
yardstick::spec(pred.qda.tbl, month, .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 spec    macro          0.808
```

Using the confusion matrix, we see that what we predicted in question 2 was right - the most mistakes are made on February. There we see only 44% of February days are correctly predicted, in comparison to 86% correct in March and 54% in January.

We see the sensitivity is right around 61.4%, while the specificity is at around 80.8%. This means that we're predicting around 61.4% of months correctly, and that we're getting about 80.8% of 'no's correct.

## Wrangling with Terminator 2 (20 points)

`Movielens` is a dataset containing user provided feedback from different movies. You can find more information about this dataset using `?movielens`. We can load this dataset by doing:

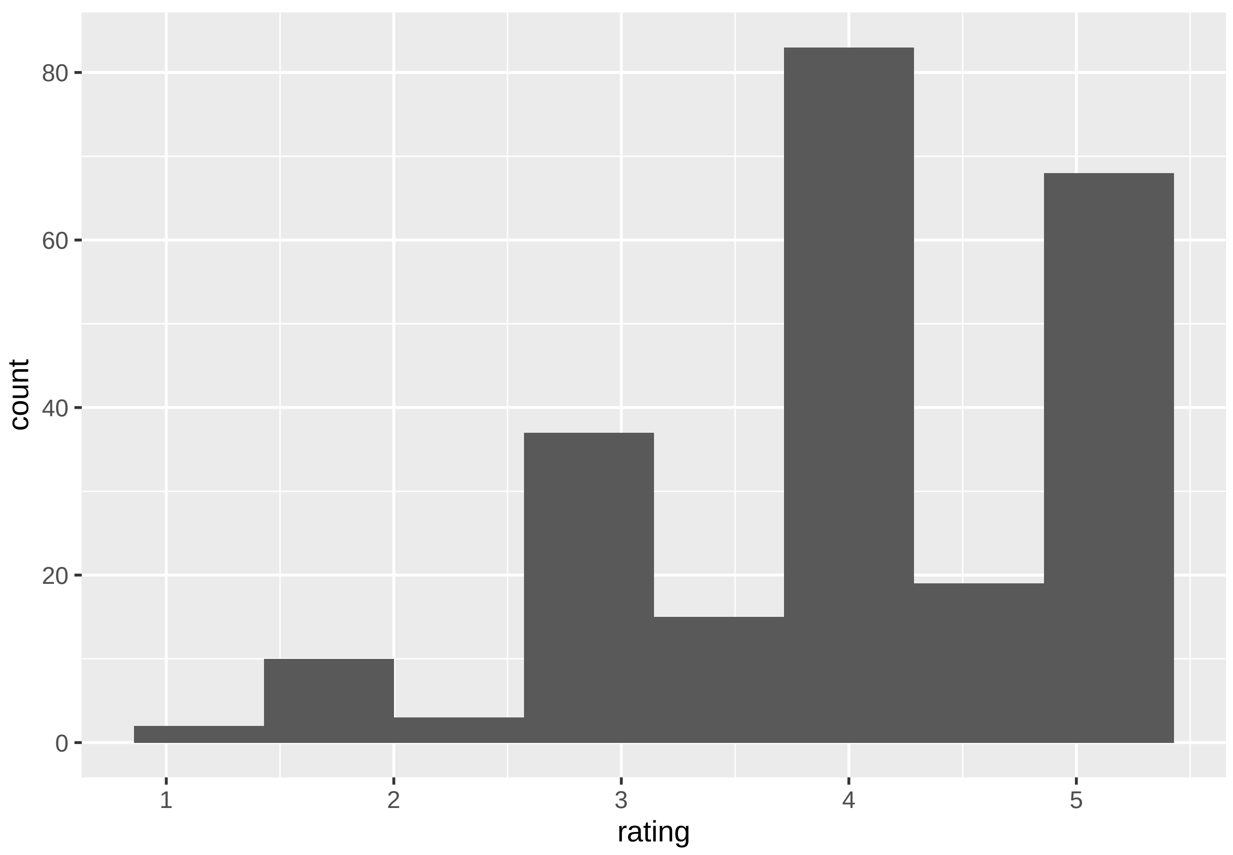
```
data(movielens)
movielens.tbl <- tibble(movielens)
```

We are interested in creating a simple recommendation system for the classic 90s movie and one of Prof. Davila's favorites, "Terminator 2: Judgment Day"

### Step one (5 points)

What are the movieId and genres corresponding to "Terminator 2: Judgment Day"? Do a histogram of the user ratings for this movie.

```
movielens.tbl%>%
  filter(title == "Terminator 2: Judgment Day")%>%
  ggplot(aes(x = rating))+
  geom_histogram(bins= 8)
```



The movieId is 589, and the genres are Action/Sci-Fi, which makes sense for a Terminator movie.

### Step two (5 points)

Create a table with the average movie ratings for each user. Also, create a table with the average rating for movies with genre "Action|Sci-Fi" for each user.

```
userId.rating<- movielens.tbl%>%
  group_by(userId)%>%
  summarize(rating.all = mean(rating))

userId.rating
```

```
## # A tibble: 671 x 2
##    userId rating.all
##     <int>      <dbl>
##  1      1       2.55
##  2      2       3.49
##  3      3       3.57
##  4      4       4.35
##  5      5       3.91
##  6      6       3.26
##  7      7       3.47
##  8      8       3.87
##  9      9       3.76
## 10     10       3.70
## # ... with 661 more rows
```

```
userId.rating.scifi <- movielens.tbl%>%
  filter(genres == "Action|Sci-Fi")%>%
  group_by(userId)%>%
  summarize(rating.scifi = mean(rating))

userId.rating.scifi
```

```
## # A tibble: 307 x 2
##    userId rating.scifi
##     <int>        <dbl>
##  1      2         5
##  2      4         5
##  3      7         2
##  4      8         4
##  5     12         3
##  6     15         2.75
##  7     17         0.5
##  8     19         3
##  9     21         4
## 10     22         5
## # ... with 297 more rows
```

### Step three (10 points)

Create a table `terminator2.movie.tbl` with columns:

- `userId`
- The average rating of all movies for that `userId`.
- The average rating of all movies in the “Action|Sci-Fi” for that userId.
- The rating that user gave to Terminator 2

Make sure to remove rows that have “NA” in either of those columns

```
user.terminator <- movielens.tbl%>%
  filter(title == "Terminator 2: Judgment Day")

terminator2.movie.tbl <- userId.rating%>%
  inner_join(userId.rating.scifi, by = "userId")%>%
  inner_join(user.terminator, by = "userId")%>%
  select(1,2,3,8)%>%
  rename(rating.terminator = "rating")
#Using inner joins ensures no NAs
```

## Rating Terminator 2 (40 points)

Make sure that the tibble you obtained from the previous point has 237 observations and 4 variables. In case you ran into problems you can use the following code:

```
terminator2.movie.tbl <- read_csv("~/Mscs 341 S22/Class/Data/movies.csv")
dim(terminator2.movie.tbl)
```

```
## [1] 237   4
```

Let's create our testing and testing dataset

```
set.seed(123456)
terminator2.split <- initial_split(terminator2.movie.tbl)
terminator2.train.tbl <- training(terminator2.split)
terminator2.test.tbl <- testing(terminator2.split)
```

## Linear models (20 points)

Using `tidymodels()` create a linear model to predict the rating of Terminator 2 based on `avg.rating` and `avg.action.scifi`. How do you interpret the coefficients of your linear model? What is your `rmse` and $R^2$ in your testing dataset?

```
lm.model <- linear_reg() %>%
  set_engine("lm")

lm.wflow <- workflow() %>%
  add_model(lm.model) %>%
  add_formula(rating ~ avg.rating+avg.action.scifi)

lm.fit <- fit(lm.wflow, terminator2.train.tbl)

extract_fit_engine(lm.fit)%>%
  tidy()
```

```
## # A tibble: 3 x 5
##   term             estimate std.error statistic  p.value
##   <chr>               <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)         1.81     0.305       5.94 1.48e- 8
## 2 avg.rating         -0.241    0.0949     -2.54 1.19e- 2
## 3 avg.action.scifi    0.823    0.0419     19.6  5.72e-46
```

```
pred.test <- augment(lm.fit, terminator2.test.tbl)

pred.test%>%rsq(truth = rating, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.536
```

```
pred.test%>%rmse(rating, .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.570
```

Starting with the rmse and $R^2$, we see that the when we run our model on the testing set we get an $R^2$ of 0.536 and an rmse of 0.57. Neither of these are terrible, but they aren't great either.

Looking at the output from tidy(), we can examine the coefficients. The intercept says that if a user had an average overall rating of 0 and an overall action/scifi rating of 0, then we'd expect them to give Terminatory a 1.81. The avg.rating coefficient of -0.241 suggests that if all else remained constant and a users overall rating increased by 1, we'd actually expect them to give Terminator 2 a rating that was .241 lower. Similarly, for avg.action.scifi, if all else remained constant and we saw a one point increase in a users action/scifi rating, we'd expect their terminator 2 rating to be 0.823 points higher.

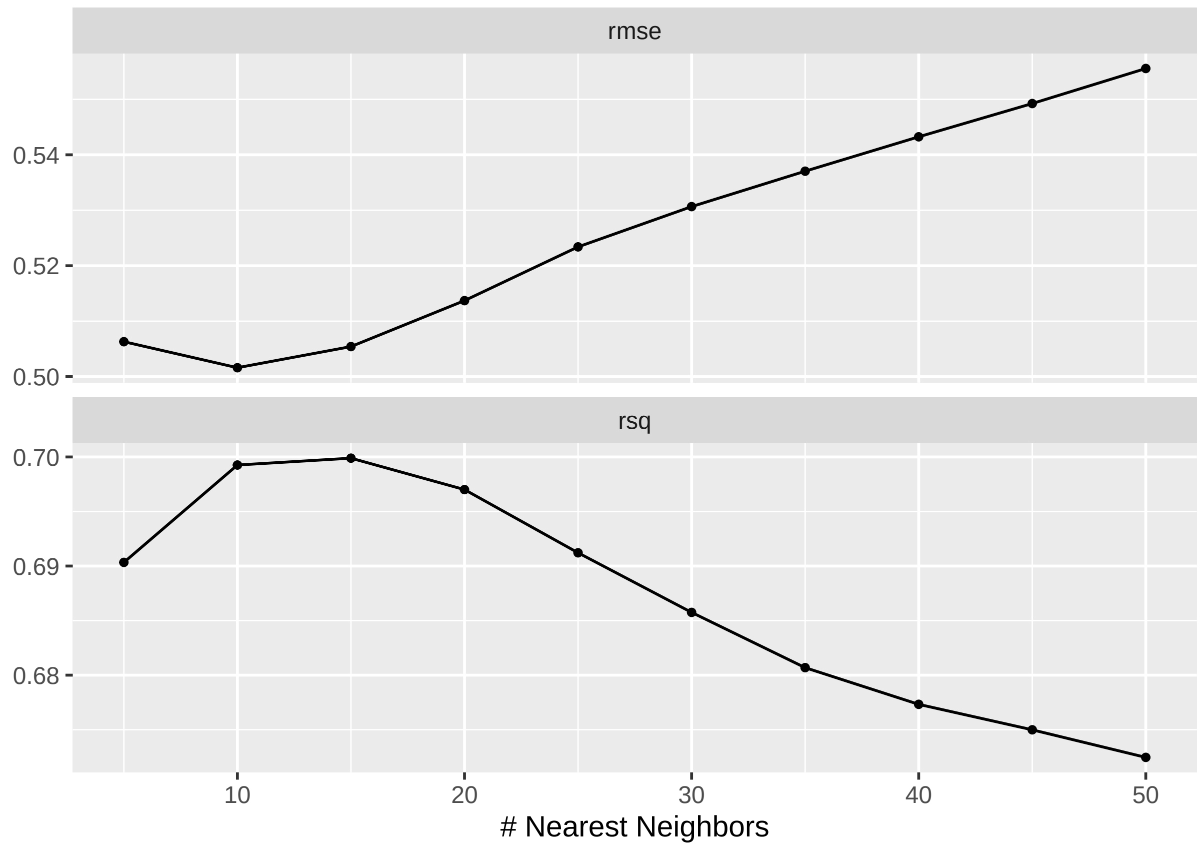
## KNN models (20 points)

Using `tidymodels()` create a KNN model to predict the rating of Terminator 2 based on `avg.rating` and `avg.action.scifi`. Create a 10-fold cross-validation set and use it to find the optimal `neigbors` by minimizing the `rmse`. Describe in your own words how to construct the 5th training and testing dataset from your cross-validation set. Calculate the `rmse` and $R^2$ of your model in your testing dataset and compare it with the linear model from the previous point.

```
set.seed(654321)
knn.model <- nearest_neighbor(neighbors = tune()) %>%
    set_engine("kknn")%>%
    set_mode("regression")

recipe <- recipe(rating ~ avg.rating + avg.action.scifi, data=terminator2.train.tbl)

knn.wf <- workflow() %>%
    add_recipe(recipe) %>%
    add_model(knn.model)

# Neighbors optimization using CV
recipe.folds <- vfold_cv(terminator2.train.tbl, v = 10)
recipe.grid<- grid_regular(neighbors(range = c(5, 50)), levels = 10)#UNSURE HOW MANY LEVELS WE SHOULD U
recipe.grid
```

```
## # A tibble: 10 x 1
##    neighbors
##        <int>
##  1         5
##  2        10
##  3        15
##  4        20
##  5        25
##  6        30
##  7        35
##  8        40
##  9        45
## 10        50
```

```
tune.results <- tune_grid(
  object = knn.wf,
  resamples = recipe.folds,
  grid = recipe.grid
)

autoplot(tune.results)
```



```
show_best(tune.results, metric = "rmse")
```

```
## # A tibble: 5 x 7
##   neighbors .metric .estimator  mean     n std_err .config
##       <int> <chr>   <chr>      <dbl> <int>   <dbl> <fct>
## 1        10 rmse    standard   0.502    10  0.0229 Preprocessor1_Model02
## 2        15 rmse    standard   0.505    10  0.0261 Preprocessor1_Model03
## 3         5 rmse    standard   0.506    10  0.0265 Preprocessor1_Model01
## 4        20 rmse    standard   0.514    10  0.0293 Preprocessor1_Model04
## 5        25 rmse    standard   0.523    10  0.0316 Preprocessor1_Model05
```

```
best.neighbor <- select_best(tune.results, neighbors, metric = "rmse")
knn.final.wf <- finalize_workflow(knn.wf, best.neighbor)
knn.final.fit <- fit(knn.final.wf, terminator2.train.tbl)

#RMSE and R^2 on testing dataset
pred.rating <- augment(knn.final.fit, terminator2.test.tbl)

rmse(pred.rating, truth = rating, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.610
```

```
rsq(pred.rating, truth = rating, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.468
```

```
#Getting 5th testing and training
testing(recipe.folds[[1]][[5]])
```

```
## # A tibble: 18 x 4
##    userId avg.rating avg.action.scifi rating
##     <dbl>      <dbl>            <dbl>  <dbl>
##  1    177       3.97             4       4
##  2    124       3.76             5       5
##  3    184       3.73             5       5
##  4    545       3.90             3       3
##  5    213       2.66             2.39    4
##  6    300       4.06             4.5     4.5
##  7    532       3.68             4       4
##  8    157       3.40             3.25    4
##  9    455       3.68             4.5     4.5
## 10    244       3.66             3.5     4
## 11    385       3.49             4       4
## 12    461       2.84             4.25    5
## 13    664       3.80             3.67    4.5
## 14    145       4.5              5       5
## 15    293       3.47             2.75    3.5
## 16    534       3.64             3       3
## 17     77       3.32             3.25    4
## 18    287       4.53             4.62    5
```

```
training(recipe.folds[[1]][[5]])
```

```
## # A tibble: 159 x 4
##    userId avg.rating avg.action.scifi rating
##     <dbl>      <dbl>            <dbl>  <dbl>
##  1    659       3.39             5       5
##  2    313       3.53             3.5     3.5
##  3    358       3.18             5       5
##  4    564       3.55             3.31    5
##  5    519       4.40             4.5     4.5
##  6    632       4.17             5       5
##  7    214       4.02             5       5
##  8      7       3.47             2       3
##  9    555       3.17             4       4
## 10    367       3.35             4       4.5
## # ... with 149 more rows
```

We see that the optimal number of neighbors is k=10, and when we run a model with k=10 we get an rmse of 0.61 and an $R^2$ of 0.468. These values are actually both worse than our linear model, as we have a higher rmse and a lower $R^2$.

While we can see the 5th training and testing datasets, there is more that goes into constructing them. To construct them, we first split our data into 10 different folds (chunks). We then go through 10 times and train our models using a different fold as the testing dataset each time. So, our 5th training dataset is all the folds except for the 5th 'chunk', which our 5th testing dataset is the 5th 'chunk'/fold itself. We can construct this in R using the training and testing functions by applying it to the 'folds' object that we make.

## Extra Credit! (5 points)

Knit your document into a PDF and submit it though the Moodle webpage.