ADM Sample Exam 1

Jaime Davila

3/21/22

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 pack 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". For problems with multiple parts, be aware that you can complete later parts successfully whether or not earlier parts 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.

Introduction

The dataset from The Pudding "Baking the Most Average Chocolate Chip Cookie" collects over 200 recipes of chocolate chip cookies and their popularity. Let's start by loading the dataset using:

```
url <- "https://raw.githubusercontent.com/the-pudding/data/master/cookies/choc_chip_cookie_ingredients.
recipe.raw.tbl <- read_csv(url)
```

Here is some brief information about the columns:

- Ingredient: The name of the ingredient used for a chocolate chip cookie.
- Text: The text obtained from the recipe that makes reference to the ingredient.
- Recipe_Index: A unique identifier associated with each recipe.
- Rating: The online rating of the recipe.

- Quantity: How much of a particular ingredient is required.
- Unit: The units in which the ingredient is measured.

Wrangling (20 points)

Before we start our analysis we need to clean and make sure our dataset is in the right shape. We will divide that into the following three steps:

Step One (5 points)

Create a tibble `recipe.wide.tbl` with columns representing each ingredient and rows for each recipe. Make sure that in case of duplicate entries take the mean of the multiple entries.

Hint: Make sure to use the parameters and `values_fn` from your pivoting function.

```
recipe.wide.tbl <- recipe.raw.tbl %>%
  select(Ingredient, Recipe_Index, Quantity, Rating) %>%
  pivot_wider(names_from=Ingredient,
              values_from = Quantity,
              values_fn=mean)
```

Step two (5 points)

Describe in your own words what the following code does. *Do not* describe what each line of code does, but what is the overall idea that the code is implementing.

```
recipe.filter.tbl <- recipe.wide.tbl %>%
  filter(!is.na(Rating)) %>%
  select(Recipe_Index, Rating, `all purpose flour`, `semisweet chocolate chip`,
         "butter", "sugar") %>%
  rename("flour"=`all purpose flour`,
         "chocolate"=`semisweet chocolate chip`) %>%
  filter(!(is.na(flour) | is.na(chocolate) | is.na(butter) | is.na(sugar)))
```

Step Three (5 points)

Create a new tibble `recipe.final.tbl` that has new variables representing the proportions of flour, chocolate, butter and sugar in the recipe (name your variables `prop.flour`, `prop.chocolate`, `prop.butter` and `prop.sugar`)

```
recipe.final.tbl <- recipe.filter.tbl %>%
  mutate(total = flour+chocolate+butter+sugar) %>%
  mutate(prop.flour = flour/total,
         prop.chocolate = chocolate/total,
         prop.butter = butter/total,
         prop.sugar = sugar/total)
```

Step Four (5 points)

Create a boxplot representing the proportions of the four ingredients in your dataset

```
recipe.final.tbl %>%
  select(Recipe_Index, Rating,
         prop.flour, prop.chocolate, prop.butter, prop.sugar) %>%
  pivot_longer(3:6) %>%
  ggplot(aes(name, value)) +
  geom_boxplot()
```

```
write.table(recipe.final.tbl,"~/Mscs 341 S22/Prof/exams/recipe.Rdata")
```

Modelling (40 points)

Make sure that the tibble you obtained from your wrangling has 55 observations and 11 variables. In case you ran into problems you can use the following code for the next points:

```
recipe.tbl <-  tibble(read.table("~/Mscs 341 S22/Prof/exams/recipe.Rdata"))
dim(recipe.tbl)
```

```
## [1] 55 11
```

Step One (10 points)

We are interested in forecasting how good a recipe will be based on the proportion of the four key ingredients (flour,chocolate, butter and sugar). What are your input and response variables? Is this a classification or prediction problem?

Step Two (20 points)

We start by dividing our dataset into testing and training as follows

```
#Training/Testing dataset
set.seed(54321)
recipe.split <- initial_split(recipe.tbl)
recipe.train.tbl <- training(recipe.split)
recipe.test.tbl <- testing(recipe.split)
```

We would like to use `tidymodels()` to create a KNN model to forecast the rating based on the proportion of flour, chocolate and butter (the proportion of sugar is 1 minus the other ones). Using 10-fold cross validation, plot the `rmse` across `neighbors`=$5, 10, \ldots, 30$ and using the plot find the optimal number of neighbors. Using your own words in a paragraph or two, describe how the 10-fold cross validation process works on this instance.

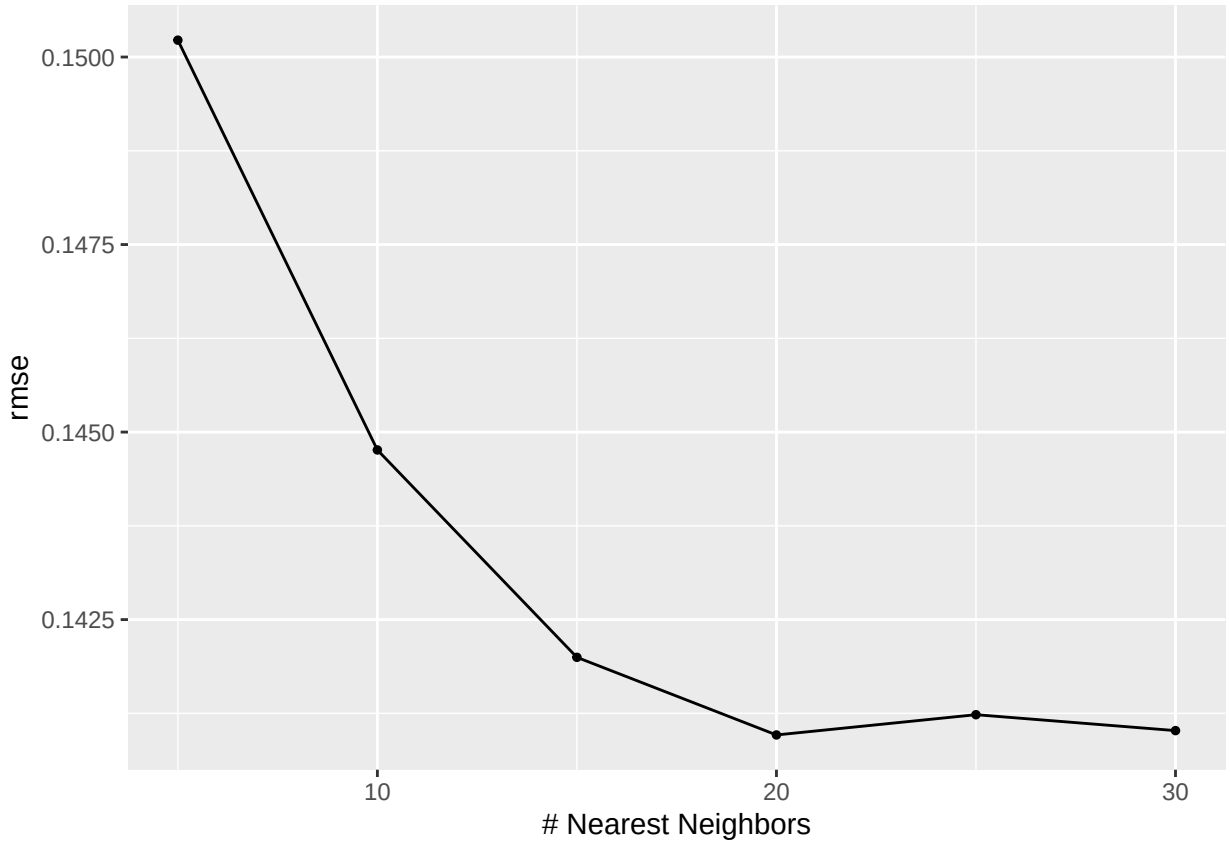
```
set.seed(54321)
knn.model <- nearest_neighbor(neighbors = tune()) %>%
    set_engine("kknn") %>%
    set_mode("regression")

recipe <- recipe(Rating ~ prop.flour + prop.chocolate + prop.butter, data=recipe.train.tbl)
knn.wf <- workflow() %>%
    add_recipe(recipe) %>%
    add_model(knn.model)

# Neighbors optimization using CV
```

```
recipe.folds <- vfold_cv(recipe.train.tbl, v = 10)
neighbors.grid.tbl <- tibble(neighbors = seq(5,30, by=5))

tune.results <- tune_grid(object = knn.wf,
                          resamples = recipe.folds,
                          grid = neighbors.grid.tbl)
autoplot(tune.results, metric="rmse")
```



Step Three (10 points)

The R^2 is the squared correlation between the response variable and the model's prediction and can be calculated using `rsq()` function. Using your results from `Step Two` select the optimal number of neighbors based on the R^2 and evaluate the R^2 on the testing dataset. How does it compare to the R^2 that you calculate using cross-validation?

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

```
## # A tibble: 5 x 7
##   neighbors .metric .estimator  mean     n std_err .config
##       <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <fct>
## 1        20 rsq     standard   0.463    10  0.0901 Preprocessor1_Model4
## 2        25 rsq     standard   0.448    10  0.110  Preprocessor1_Model5
## 3        10 rsq     standard   0.438    10  0.0992 Preprocessor1_Model2
## 4        15 rsq     standard   0.427    10  0.100  Preprocessor1_Model3
## 5        30 rsq     standard   0.424    10  0.122  Preprocessor1_Model6
```

```
best.neighbor <- select_best(tune.results,
                             neighbors, metric = "rsq")

knn.final.wf <- finalize_workflow(knn.wf, best.neighbor)
knn.final.fit <- fit(knn.final.wf, recipe.tbl)

# Evaluation of model on the testing dataset
augment(knn.final.fit, recipe.test.tbl) %>%
  rsq(truth = Rating, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.248
```

Aiming for the Stars (40 points)

We will be using the `stars` dataset from the `dslabs` package. More information on this dataset can be found using `?stars`

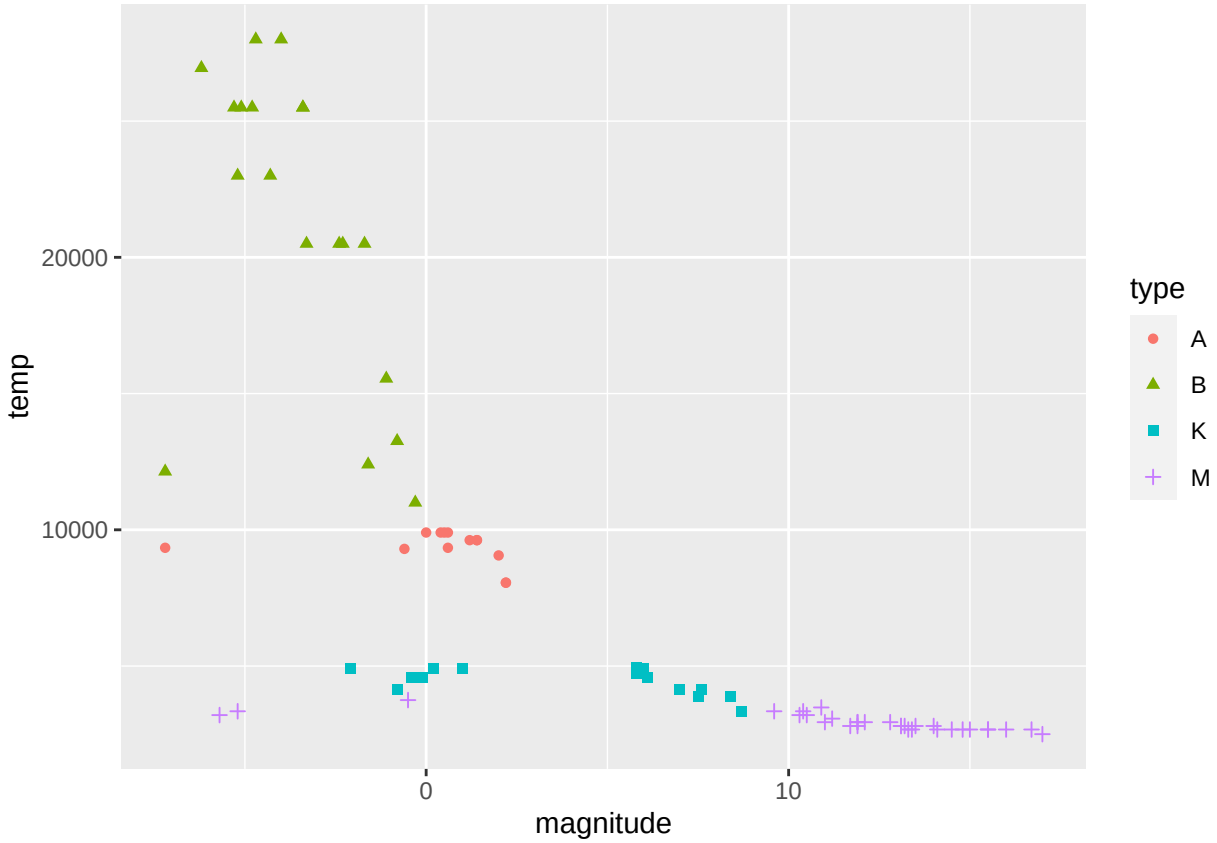
```
data(stars)
stars.tbl <- tibble(stars)
```

Part one (15 points)

Subset your dataset to only stars of type A,B,K and M. Plot the magnitude and temperature and the type of star using color. Based on this information, would and LDA or QDA provide a better model?

```
star.levels = c("A","B","K","M")
stars.filter.tbl <- stars.tbl %>%
  filter(type %in% star.levels) %>%
  mutate(type = factor(type, levels=star.levels))

ggplot(stars.filter.tbl, aes(magnitude, temp,
                             color=type, shape=type))+
  geom_point()
```



Part two (15 points)

As usual let's create our training/testing dataset

```
#Training/Testing dataset
set.seed(12345)
stars.split <- initial_split(stars.filter.tbl)
stars.train.tbl <- training(stars.split)
stars.test.tbl <- testing(stars.split)
```

Implement an LDA and QDA model that would predict the type of star. Based on the confusion matrix what model does a better job? Explain briefly why.

```
lda.model <- discrim_linear() %>%
  set_engine("MASS") %>%
  set_mode("classification")

recipe <- recipe(type ~ magnitude+temp, data=stars.train.tbl)
lda.wflow <- workflow() %>%
  add_recipe(recipe) %>%
  add_model(lda.model)

lda.fit <- fit(lda.wflow, stars.train.tbl)

augment(lda.fit,stars.test.tbl) %>%
  conf_mat(type, .pred_class)
```

```
##            Truth
## Prediction A B K M
##          A 4 1 0 0
##          B 0 3 0 0
##          K 0 0 2 0
##          M 0 0 2 8
```

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

recipe <- recipe(type ~ magnitude+temp, data=stars.train.tbl)
qda.wflow <- workflow() %>%
  add_recipe(recipe) %>%
  add_model(qda.model)

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

augment(qda.fit,stars.test.tbl) %>%
  conf_mat(type, .pred_class)
```

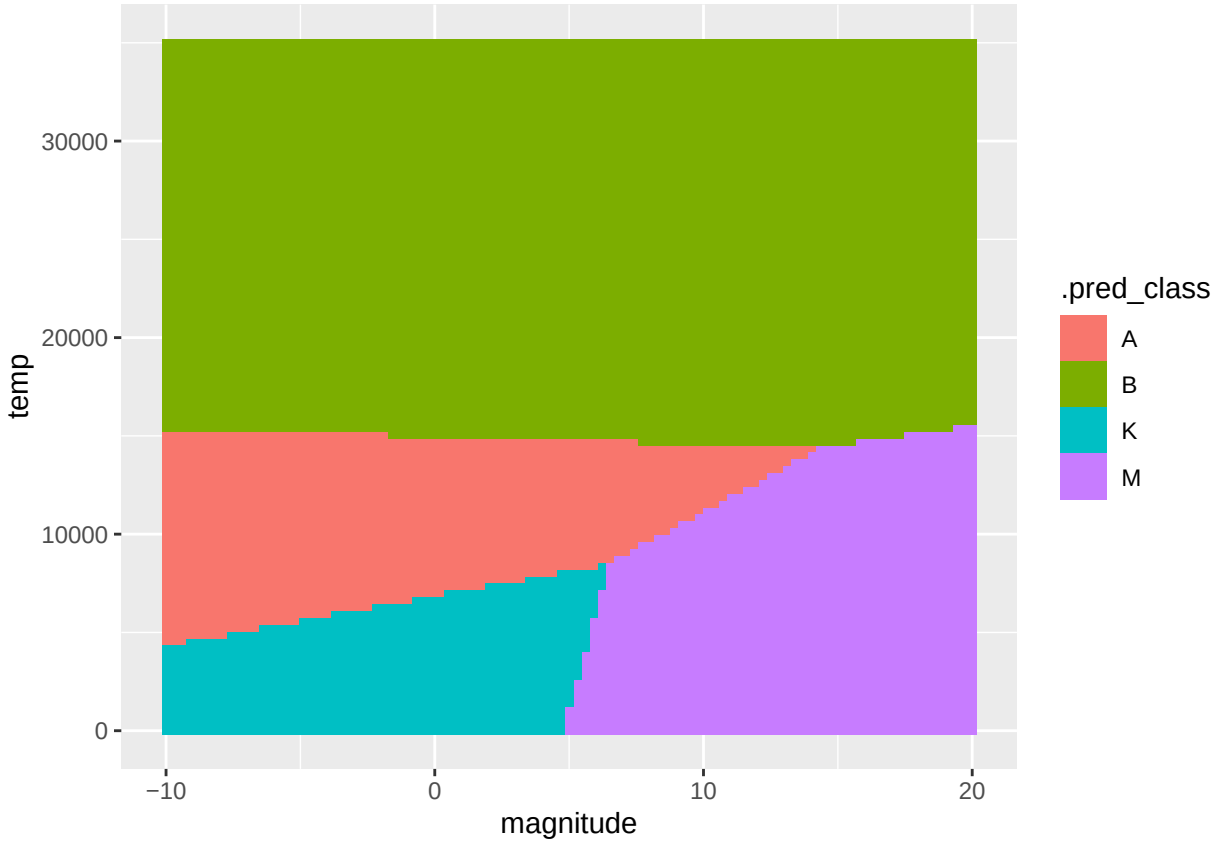
```
##            Truth
## Prediction A B K M
##          A 4 1 0 0
##          B 0 3 0 0
##          K 0 0 4 0
##          M 0 0 0 8
```

Part three (10 points)

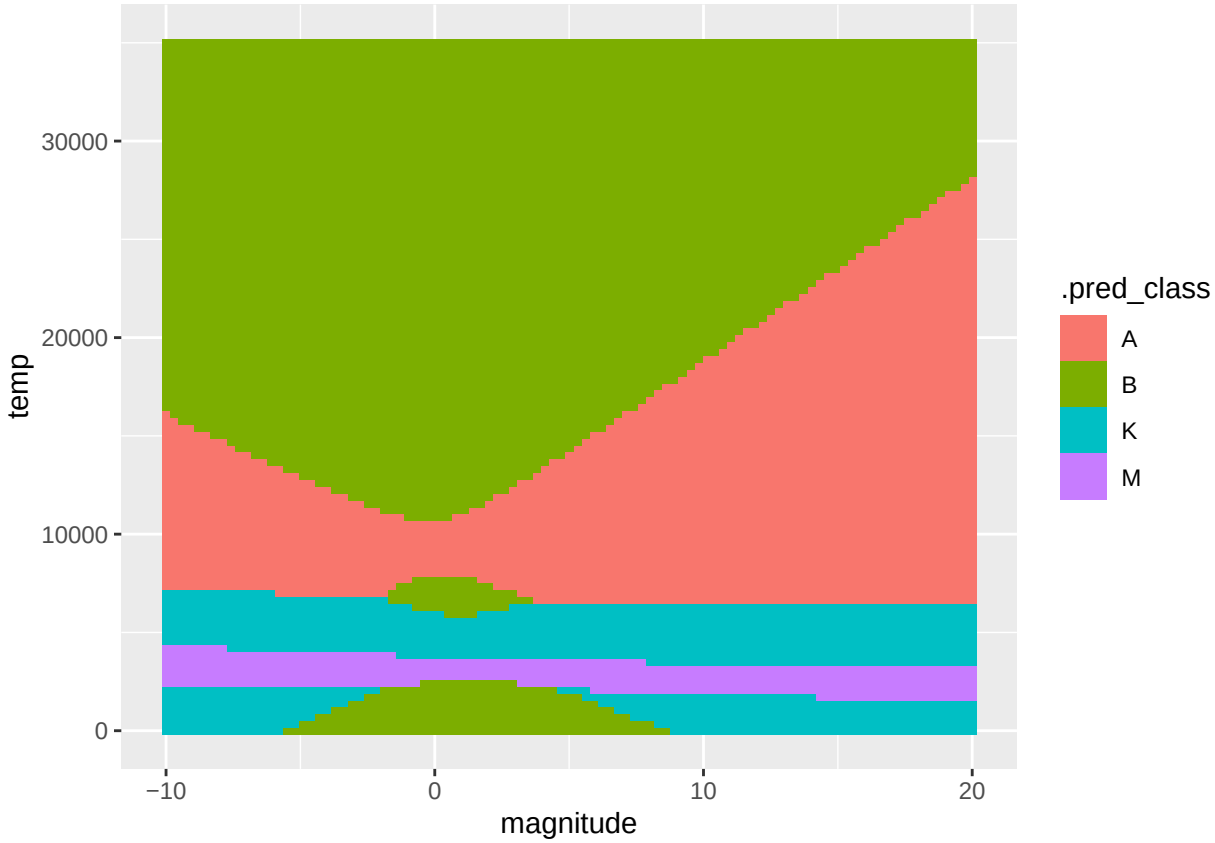
Create a two dimensional grid with around 10,000 points and plot the predicted class for each model (lda and qda). How do the boundary regions differ?

```
grid.tbl <-  expand_grid(magnitude=seq(-10,20, by=0.3),
                         temp = seq(0,35000, by=350))

augment(lda.fit, grid.tbl) %>%
  ggplot(aes(magnitude,temp, fill=.pred_class)) +
  geom_raster()
```




```
augment(qda.fit, grid.tbl) %>%
  ggplot(aes(magnitude,temp, fill=.pred_class)) +
  geom_raster()
```


30000
20000
10000
0
temp
−10
0
10
20
magnitude
.pred_class
A
B
K
M