HW 4: Clustering and Regression*

due 7/11/2023

Many college courses conclude by giving students the opportunity to evaluate the course and the instructor anonymously.

However, the use of these student evaluations as an indicator of course quality and teaching effectiveness is often criticized because these measures may reflect the influence of non-teaching related characteristics, such as the physical appearance of the instructor.

The article titled, "Beauty in the classroom: instructors' pulchritude and putative pedagogical productivity" (Hamermesh and Parker, 2005) found that instructors who are viewed to be better looking receive higher instructional ratings. (Daniel S. Hamermesh, Amy Parker, Beauty in the classroom: instructors pulchritude and putative pedagogical productivity, Economics of Education Review, Volume 24, Issue 4, August 2005, Pages 369-376, ISSN 0272-7757, 10.1016/j.econedurev.2004.07.013. http://www.sciencedirect.com/science/article/pii/S0272775704001165.)

In this homework, you will analyze the data from this study in order to see how to best predict a professor's rating. We'll take another look at this dataset later in the semester.

Introduction

Click the Github Assignment link on Canvas, then open it in Posit Cloud as a new project from a GitHub repository.

Warm up

Let's warm up with some simple exercises. Open the `hw4.qmd` Quarto file. Update the YAML of your Quarto file with your information, render, commit, and push your changes. Make sure to commit with a meaningful commit message. Then, go to your repo on GitHub and

*Loosely adapted from a Data Science in a Box lab.

confirm that your changes are visible in your `.qmd` file. If anything is missing, commit and push again.

Data

The data were gathered from end of semester student evaluations for a large sample of professors from the University of Texas at Austin. In addition, six students rated the professors' physical appearance. (This is a slightly modified version of the original data set that was released as part of the replication data for Data Analysis Using Regression and Multilevel/Hierarchical Models (Gelman and Hill, 2007).) The result is a data frame where each row contains a different course and columns represent variables about the courses and professors.

The data can be found in the **openintro** package, and it's called `evals`. To import it, we just have to load the package (which is already installed for you).

Once you do so, you can also find out more about the dataset by inspecting its documentation, which you can access by running `?evals` in the console.

You can also find this information here.

```
library(openintro)
data("evals")
```

Exercises

> **!** Remember to Commit Regularly!
>
> *Render, commit, and push your changes to GitHub with an appropriate commit message at regular intervals.*

Data Prep

Q1

Create a new variable called `prof_num_cls` that captures how many courses each professor who teaches the course in the data teaches in the data.

Hint:

- `prof_id` identifies which professor taught a each class

- You can count up how many times a `prof_id` appears in the data set to find out how many courses each professor taught. You can then merge/join this back with `evals`.

> **i** Why `prof_num_cls`?
>
> Why are we doing this? It is important to consider differences between professors that have nothing to do with beauty. For example, professors who teach more classes may be different from those who teach fewer classes. Ideally, we would actually treat `prof_id` *itself* as a categorical variable, which would allows us to capture such differences. However, this is complicated by the fact some of the professors in the sample taught only one course. Inevitably, when we create a train-test split or do cross-validation, these individuals will not be represented in the training data, thus making it impossible to predict an outcome for them. There are ways around this - in a Bayesian or hierarchical modeling framework, for example, we could include what are called random intercepts for each professor. Under this framework it is possible to make predictions for categories not originally seen by the model. However, that is beyond the scope for this course, and so we will attempt to capture at least this one additional potential difference between professors.

Q2

Drop `course_id`, `prof_id`, `bty_f1lower` through `bty_m2upper`, and `cls_did_eval`.

Then, split the data into a test set and a training set, with 70% of the data used for the training set and 30% for the test set. Use `280` as the seed.

Clustering and Modeling

In Lab 4, you explored how to do clustering without a target in mind. As the textbook points out, it can be tricky to decide on the optimal number of clusters in such case. It is different, however, when incorporating clustering into a prediction pipeline. There it is clear how to decide on the optimal number of clusters: however many result in the best out-of-sample fit. In this section, we will explore this option.

Q3: K-Means

Fit a model where you regress `score` on all categorical variables *and* cluster membership based on a K-means clustering of all *numeric* variable (but not the factor variables) with `nstart = 50`. Set the seed to `281` when creating the folds for cross-validation.

! Adding Clusters to Data

When we do clustering, we often are trying to augment the data. We will want to add the cluster assignments to our data. We can do this using `mutate()`. See below for an example using the built-in `iris` dataset:

```
# load package
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.2     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.4.2     v tibble    3.2.1
v lubridate 1.9.2     v tidyr     1.3.0
v purrr     1.0.1
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to bec
```

```
set.seed(1)
iris_kmeans <- kmeans(iris %>% select(-Species), centers = 3, nstart = 20)
# cluster assignment
iris <- iris %>%
  mutate(cluster = iris_kmeans$cluster)
head(iris)
```

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

`cluster` is an integer vector by default. This is fine in most cases, but it is important to keep in mind that cluster labels are *not* numeric. Cluster 3 is not "larger" than cluster 1 by default. Cluster labels are meaningless without context. As such, when we include clusters in a regression, we have to treat them as categorical variables.

```
iris <- iris %>%
  mutate(cluster = factor(cluster))
head(iris$cluster)
```

```
[1] 3 3 3 3 3 3
Levels: 1 2 3
```

```r
# this regression is meaningless, just an example:
lm(Sepal.Length ~ cluster, data = iris) %>%
  coef()
```

```
(Intercept)    cluster2    cluster3
  5.9016129   0.9483871  -0.8956129
```

Use 10-fold cross-validation on your training set to find the optimal number of clusters, going from 2 to 10 clusters. What is the optimal number of clusters?

Then use the test set to estimate the out-of-sample error for the optimal model. You will have to *refit* K-means on the whole training data to do this (set the seed to 282 before doing so).

! Prediction

Prediction for a different set (i.e. assigning new points to clusters previously found) is not straightforward with the output of the `kmeans()` function, as `kmeans` objects do not have a `predict()` method. The logic *is* rather straightforward, however: we need to identify to which cluster center each point is the closest. That is our prediction for the cluster membership.
We will use the `get.knnx()` function from the `FNN` package to do this for us. The general syntax is the following:

```r
pred_clust <- get.knnx(kmeans_ob$center, newdata, 1)$nn.index[,1]
```

For example, we can split the `swiss` data (a built-in dataset in base R) into two sets, perform clustering on the first set, and then apply the discovered clusters to the second set.

```r
# load package
library(FNN)
# set seed for reproducibe results
set.seed(12)
# create train/test split
train <- sample(1:nrow(swiss), 0.7 * nrow(swiss))
# fit kmeans on swiss training set
swiss_kmeans <- kmeans(swiss[train,], centers = 3, nstart = 20)
# predict cluster assignment for swiss test set
pred_clust <- get.knnx(swiss_kmeans$center, swiss[-train,], 1)$nn.index[,1]
# create a data frame of cluster assignments for the training data
clust_train <- data.frame(canton = names(swiss_kmeans$cluster),
                          cluster = swiss_kmeans$cluster,
                          set = "train")
# create a data frame of cluster assignments for the test data predictions
clust_test <-  data.frame(canton = row.names(swiss[-train,]),
                          cluster = pred_clust,
                          set = "test")
# show assignment to clusters for both sets
bind_rows(clust_train, clust_test) |>
  count(cluster, set)
```

```
  cluster   set  n
1       1  test  5
2       1 train 15
3       2  test  7
4       2 train  9
5       3  test  3
6       3 train  8
```

Q4

Look at the cluster centers from the K-means fit to the whole training data. Can you identify any patterns between the clusters? Use some form of visualization to back up your point.

Lasso

Q5

Using cross-validation on the training set to find the optimal λ (set the seed to `283`), fit a Lasso model. What is the test set MSE? Which variables are selected by the Lasso?

Comparing Approaches

Q6

Which approach did better in prediction, when considering the test MSE calculations?

Can you think of a way to combine the two methods (you do not have to implement this method)?

What do you think are the advantages and disadvantages for each method? What can Lasso tell us that K-means cannot? What about the opposite?

Final Steps

When you have made your final edits, remember to render, stage the changed files, then commit, and push to the repo.

Don't forget to submit your pdf on Canvas.

Grading

Please put your name on your answer document and add the date. You won't get points for doing this, but you will *lose* points for not doing so.

Remember to comment your code. You don't get points for commenting, but you will *lose* points for not commenting your code.

For clarity's sake, you should label your code chunks (this helps make debugging easier). You will *lose* points for not naming chunks.

If an exercise asks a question, you should answer it in narrative form and not just rely on the code. You will be *penalized* otherwise.

Please suppress warnings and messages. You will *lose* points if you do not suppress warnings and messages.

If you are asked to make a visualization, make sure that it is presentable, professional in appearance, has a title, and has properly labeled axes. You will *lose* points if you do not do this.

Points Breakdown

- Q1: 2 pts
- Q2: 2 pts
- Q3: 10 pts
- Q4: 3 pts
- Q5: 4 pts
- Q6: 3 pts
- Workflow (includes making sufficient commits): 4 pts

Total: 28 pts