Data Science for Public Policy

Aaron R. Williams - Georgetown University

PPOL 670 | Assignment 07
Machine Learning

Due Date: Wednesday, April 6th at 6:29 PM.

Deliverable:

1. A .Rmd file with your R code.
2. The resulting project .html file.
3. The URL to your private GitHub repository from assignment 06.

Grading Rubric

- [3 points] Exercise 01
- [7 points] Exercise 02

Points: 10 points

Plagiarism on homework or projects will be dealt with to the full extent allowed by Georgetown policy (see http://honorcouncil.georgetown.edu).

Setup

Add a second `.Rmd` to the GitHub repository from assignment06.Rmd and update your `README.md`. If you switched partners, then create a new repository.

Exercise 01 (3 points)

This exercise will use restaurant inspections data from the NYC OpenData Portal by way of tidytuesday. The grades can be A, B, or C but this exercise will work with "A" or "Not A" so that this is a binary classification problem. Use the following code to create the data set for this application. (also included as a .R script)

```
library(tidyverse)
library(lubridate)

# use this url to download the data directly into R
df <- read_csv("https://data.cityofnewyork.us/api/views/43nn-pn8j/rows.csv")

# clean names with janitor
sampled_df <- df %>%
  janitor::clean_names()

# create an inspection year variable
sampled_df <- sampled_df %>%
  mutate(inspection_date = mdy(inspection_date)) %>%
  mutate(inspection_year = year(inspection_date))

# get most-recent inspection
sampled_df <- sampled_df %>%
  group_by(camis) %>%
  filter(inspection_date == max(inspection_date)) %>%
  ungroup()

# subset the data
sampled_df <- sampled_df %>%
  select(camis, boro, zipcode, cuisine_description, inspection_date,
         action, violation_code, violation_description, grade,
         inspection_type, latitude, longitude, council_district,
         census_tract, inspection_year, critical_flag) %>%
  filter(complete.cases(.)) %>%
  filter(inspection_year >= 2017) %>%
  filter(grade %in% c("A", "B", "C"))

# create the binary target variable
sampled_df <- sampled_df %>%
  mutate(grade = if_else(grade == "A", "A", "Not A")) %>%
  mutate(grade = as.factor(grade))

# create extra predictors
sampled_df <- sampled_df %>%
  group_by(boro, zipcode, cuisine_description, inspection_date,
         action, violation_code, violation_description, grade,
         inspection_type, latitude, longitude, council_district,
         census_tract, inspection_year)  %>%
  mutate(vermin = str_detect(violation_description, pattern = "mice|rats|vermin|roaches")) %>%
  summarize(violations = n(),
            vermin_types = sum(vermin),
            critical_flags = sum(critical_flag == "Y")) %>%
  ungroup()

# write the data
```

```
write_csv(sampled_df, "restaurant_grades.csv")
```

1. Estimate a Model

a. Split the data into training and testing sets with the seed `20201020`.
b. Create a recipe with (at least) `themis::step_downsample(grade)`. Downsampling is one technique to deal with class imbalances. Only one step is required but you can use more.
c. Estimate a decision tree for classification with the "rpart" engine. There is no need to use resampling methods like v-fold cross-validation because you are only estimating one model.

2. Evaluate the Model

a. Create a confusion matrix using your model estimated on the training data and the testing data.
b. Calculate the precision and recall/sensitivity using `library(tidymodels)`.
c. Describe the quality of the model.

3. Improvement

a. Describe and justify ideas for improving the model with 2-3 sentences. Improvements can include different feature engineering, different algorithms, different hyperparameters, and adding auxiliary information.

4. Variable Importance

Measures of variable importance can be calculated with most types of predictive models. Max Kuhn offers a brief outline here. The authors of `library(rpart)` briefly explain the method for the rpart model on page 11 here.

`library(vip)` (info here) and `library(tidymodels)` can quickly create a plot of the variable importance. The following code plots the ten most important predictions from a fit workflow.

```
library(vip)

rpart_fit %>%
  extract_fit_parsnip() %>%
  vip(num_features = 10)
```

a. Briefly explain how this measure is calculated and interpret the results. If the above descriptions are confusing, just Google "decision tree feature importance" and find an explanation that you find intuitive.

5. Application

a. Write 2-3 sentences about how an ombudsman working for the NYC health department could use a model like the model you just created.

Exercise 02 (7 points)

The objective of this exercise is to predict Chicago "L" train ridership at the Clarke/Lake station. The data come from Feature and Target Engineering by Max Kuhn and Kjell Johnson. I recommend reading the description. After loading `library(tidymodels)`, run the following to set up the data.

```
Chicago_modeling <- Chicago %>%
  slice(1:5678)

Chicago_implementation <- Chicago %>%
  slice(5679:5698) %>%
  select(-ridership)
```

1. Convert date into a useable variable

In this application, unaltered date won't be useful for prediction because all new observations will have unseen dates by definition. Use `library(lubridate)` to generate the following before splitting your data:

- `weekday` is the day of the week. Use `label = TRUE` so the value is a factor
- `month` is the month of the year. Use `label = TRUE` so the value is a factor.
- `yearday` is the numeric day number of the year.

2. Set up a testing environment

a. Split `Chicago_modeling` into a training set and a testing set with seed `20211101`. Use the default `prop`.
b. Demonstrate some exploratory data analysis with the training data.
c. Set up v-fold cross-validation with 10 folds.

3. Test different approaches

Use cross-validation to compare at least three different models that meet the following conditions:

1. Create a recipe object called `chicago_rec` for feature engineering that uses at least three `step_*()` functions from `library(recipes)`. You must use `step_holiday()`.
2. Define three different model specifications using `library(parsnip)`. You should use one parametric regression model (linear regression, LASSO, etc.), a random forest model, and a model of your choice (e.g. KNN, decision/regression tree, MARS). One of these model specification objects must use hyperparameter tuning. You will need to create an appropriate parameter grid for the hyperparameter you choose to tune.
3. Create a workflow for each model specification you created in part two. You will use `chicago_rec` as the recipe for all workflows.
4. Use the v-fold cross-validation you created in question two to fit each of your three workflows. Hint: you'll use `fit_resamples()` for the models without hyperparameter tuning and tune_grid() for the model with hyperparameter tuning.
5. Calculate the MAE and RMSE for each resample. Plot the RMSE across the 10 resamples for the three different models. Find the average RMSE of each model across the 10 resamples (this should result in three numbers). Hint: check out `collect_metrics()` and its `summarize` argument.

4. Estimate the out-of-sample error rate

a. Typically, we would estimate the best model on all of the training data. For simplicity, we will skip this step.
b. Using your "best" model estimated during cross validation, make predictions with the testing data and calculate the RMSE. This is your out-of-sample error rate estimate. You only want to touch the testing data once!

5. Implement the final model

Using your "best" model from the cross validation, estimate the ridership for the 20 observations in `Chicago_implementation`. Be sure to print your 20 predictions so they are visible in your submission. You only want to touch the implementation data once!

6. Briefly describe your final model

Write a few sentences describing your final model. Is it a good model? Is it globally interpretable? Is it locally interpretable? What are the most important predictors?

Stretch (5 points)

This stretch exercise is a great opportunity to get started on your project. Please reach out early and often. Please reach out if you have an alternative idea.

It is acceptable to end up with a model that is not useful! Your grade depends on your evaluation of the usefulness–not the creation of a perfect model.

1. Set up

a. Find a government/policy-relevant data set with a practical regression or classification application. You can use the data you have already found/used for your project of interest. Briefly describe the data set. Clearly state the prediction application and describe the target variable.
b. Split the data into training and testing data.
c. Using data visualization and other skills from this course, perform an exploratory data analysis (on the training data!). During this process, perform necessary data cleaning (please reach out if the data cleaning is significant work but you wish to use this data for your project).
d. Explicitly pick an error metric. Explicitly state the costs of an error. How much error is too much if you use a regression model? Is a false positive or a false negative more costly if you use a classification problem?

2. Come up with Models

a. Describe three specifications for candidate models. In the three specifications:

- State the predictor variables included in the model and necessary preprocessing for each variable
- Use at least least two different types of preprocessing
- Use at least two different algorithms (i.e. linear regression, KNN, CART, random forest).
- Use at least one algorithm with hyperparameters.

3. Estimation

a. Implement one candidate model from step two on the training data with resampling methods and hyperparameter tuning (if applicable) using `library(tidymodels)`.

4. Interpretation

a. Interpret the results in the context of the application. Explain why your model is useful or not considering implementation, the error rate, and context.
b. Write at least three sentences explaining and justifying two different feature engineering choices and two different modeling options you would try next to improve upon your first result.