Predictive Analytics Homework 4: Model Tuning, Non-Tabular Data

In this homework we'll practice tuning and evaluating some models, which should help reinforce the validation concepts from this week. You'll also try out some pre-trained deep neural net models to get a better sense of how you might use any of them in your industry.

Part 1: Tuning Models

Let's work with a dataset of concrete mixture strength. Concrete is a mixture of various ingredients; the proportions of them can affect the strength of the resulting concrete. If we could predict the strength without testing it, we could identify promising new formulations, or be better positioned to do "what if" scenarios. So let's try it.

We'll be following the analysis in Tidy Modeling With R chapter 15. We will make slightly different choices here, though, so avoid directly copying code.

```
library(tidyverse)
library(tidymodels)
library(glue)

# Load the full dataset.
data("concrete", package = "modeldata")
glue("Before filtering: {nrow(concrete)} observations")
```

Before filtering: 1030 observations

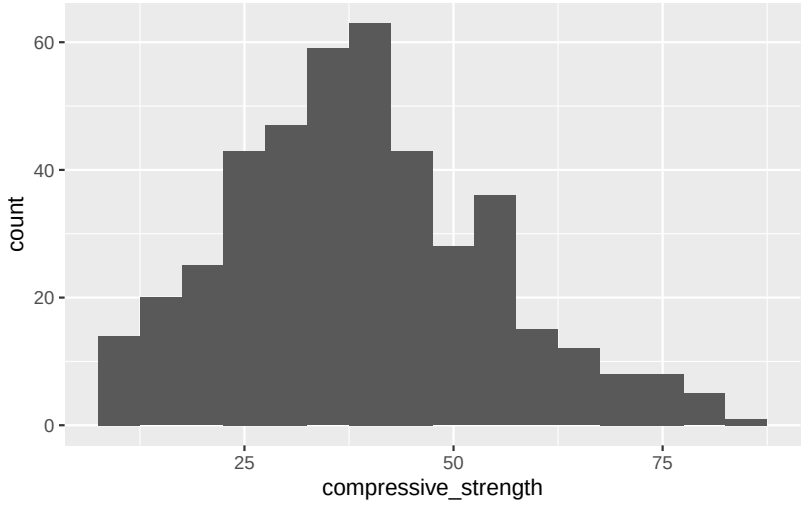
```
# Take the mean of all measurements of the same mixture.
concrete <- concrete %>%
   group_by(across(-compressive_strength)) %>%
   summarize(compressive_strength = mean(compressive_strength),
             .groups = "drop")
glue("After aggregating all trials of the same mixture: {nrow(concrete)} observations remain.")
```

After aggregating all trials of the same mixture: 992 observations remain.

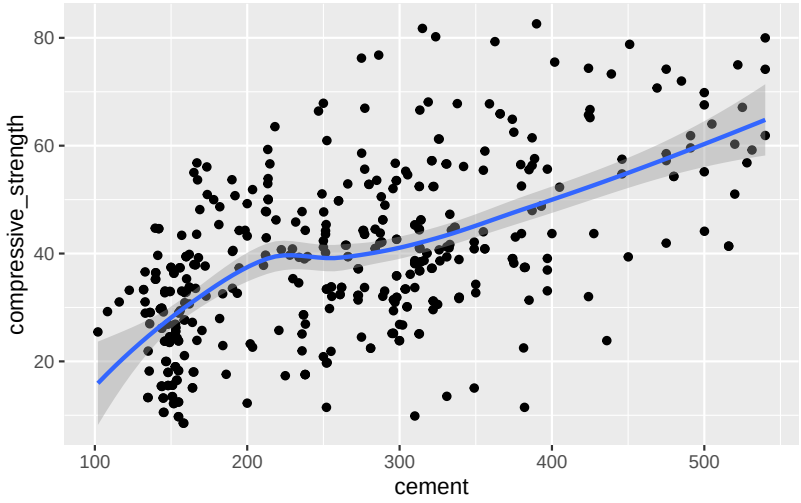
```
# Take only the final measurement (max age).
# Otherwise, arguably, we're cheating.
concrete <- concrete %>%
  group_by(across(-c(compressive_strength, age))) %>%
  slice_tail(n=1) %>%
  ungroup()
glue("After taking only the final measurement: {nrow(concrete)} observations remain.")
```

After taking only the final measurement: 427 observations remain.

Exercise 1: (5pt) Do some exploratory analysis of the data. Make at least two plots of your own choice. Suggestions (1): It's almost always useful to look at the distribution of the target variable, using a plot like this:



And (2) sometimes it's helpful to plot one or more features (predictors) against the target:



Exercise 2: (3pt) Hold out a test set of 20% of the data. Report the number of mixtures in the training set and in the test set; make sure that these numbers make sense.

Exercise 3: (5pt) Compute the MAE of a linear regression model with no preprocessing or regularization, averaged over 5 folds of cross-validation. Use the `workflow` style with a blank recipe that we used in the Tutorial. Use the formula `compressive_strength ~ .` so that the model uses all available features.

Note: The text chapter passes `strata = "compressive_strength"` to `initial_split`. Optionally, look up what that does, consider whether it's appropriate, and if so, do it too.

```
# A tibble: 1 x 6
  .metric .estimator  mean     n std_err .config
  <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1 mae     standard    6.88     5   0.425 Preprocessor1_Model1
```

Exercise 4 (4pt): Repeat the previous exercise, but this time, use the following additional pre-processing steps. **Write a sentence comparing the results quantitatively.** Note: use the same resamples; don't repeat the `vfold_cv`.

1. Normalize all numeric predictors. (`step_normalize`, `all_numeric_predictors`)

2. Add all interaction terms (`step_interact(~ all_predictors() : all_predictors())`).

```
# A tibble: 1 x 6
  .metric .estimator  mean     n std_err .config
  <chr>   <chr>      <dbl> <int>   <dbl> <chr>
1 mae     standard    5.68     5   0.393 Preprocessor1_Model1
```

Exercise 5 (4pt): Repeat the previous exercise, but with a model type of your choice. Summarize your findings: which model would you choose, and what performance would you expect it to have on new concrete mixtures?

Exercise 6 (4pt): Evaluate all of your models on the test set. (Fit them on the complete training set, ignoring the resamples.) Write a sentence or two comparing the performance you estimated using cross-validation with the performance you observe on the test set.

The code shouldn't be the blocker here, so I'm including two ways: first a simple way using just what you've already used:

```
all_predictions <- bind_rows(
  linreg1 = fit(linreg1, train) %>% augment(test),
  linreg2 = fit(linreg2, train) %>% augment(test),
  rf      = fit(rf_model, train) %>% augment(test),
  .id = "model"
) %>% mutate(model = as_factor(model))
all_predictions %>%
  group_by(model) %>%
  yardstick::mae(truth = compressive_strength, estimate = .pred)
```

```
# A tibble: 3 x 4
  model   .metric .estimator .estimate
  <fct>   <chr>   <chr>          <dbl>
1 linreg1 mae     standard        6.66
2 linreg2 mae     standard        5.61
3 rf      mae     standard        4.72
```

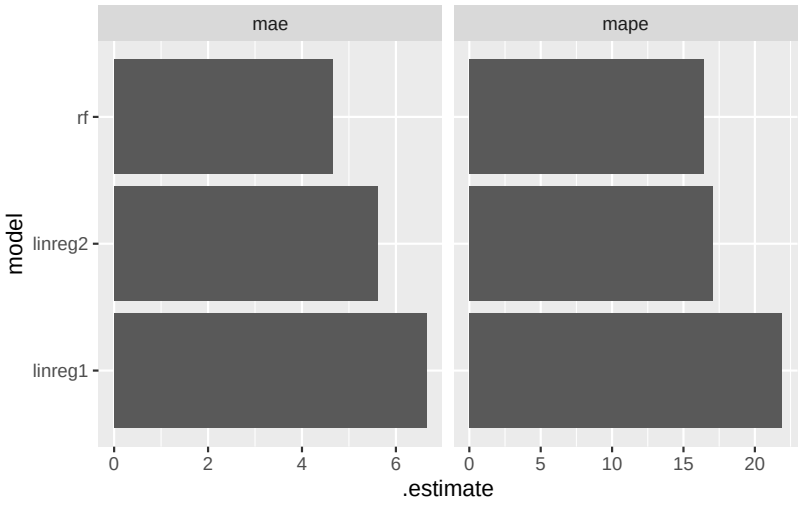
And now here's a more complex approach that (1) shows off multiple metrics, (2) is less repetitive, and (3) works (with some adaptation) for cross-validation also:

```
# Train all of the models on the training set.
all_models <- as_workflow_set(
  linreg1 = linreg1,
  linreg2 = linreg2,
  rf = rf_model
) %>%
  mutate(fit = map(info, ~ fit(.x$workflow[[1]], train)))

# Compute all of their predictions on the test set
all_predictions <-
  all_models %>%
  mutate(preds = map(fit, ~ augment(.x, data_with_split_morked))) %>%
  select(model=wflow_id, preds) %>%
  unnest(preds)

# Compute metrics
all_metrics <- all_predictions %>%
  group_by(model, split) %>%
  yardstick::metric_set(mae, mape)(truth = compressive_strength, estimate = .pred)

# Plot metrics on test set
all_metrics %>%
  filter(split == "test") %>%
  ggplot(aes(x = .estimate, y = model)) + geom_col() + facet_wrap(vars(.metric), scales = 'free
```

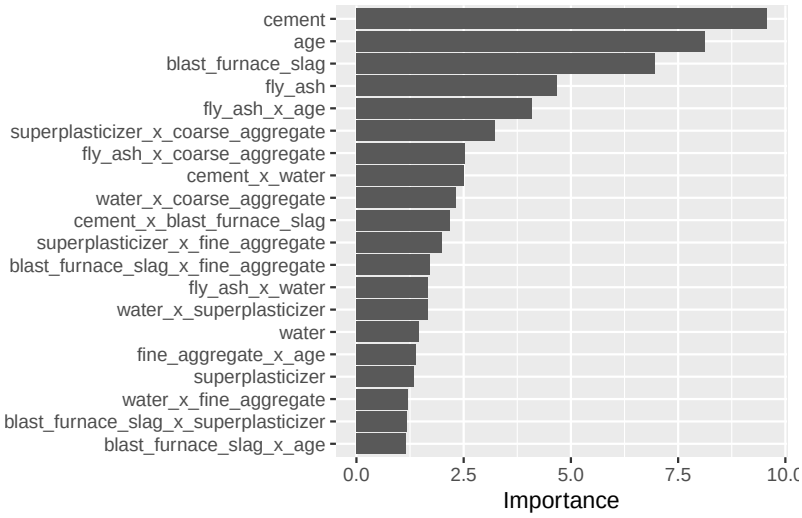


Aside: variable importance plots

A Variable Importance Plot gives us a quick-and-dirty measure of how "important" each variable (feature/predictor) is to the model *overall* (which doesn't necessarily mean that it's the deciding factor in any *particular* decision; see chapter 18 (Explaining Models and Predictions) from Tidy Modeling with R for some ways to estimate the effect for individual predictions) . No method has yet been developed that reliably indicates variable importance (or explains decisions in any way) in a way that fully matches human intuitions about explanations, so don't expect too much here. But it can be helpful in the "quick and dirty" sense to see what your model is doing, even when it's too complex to directly visualize (like a random forest).

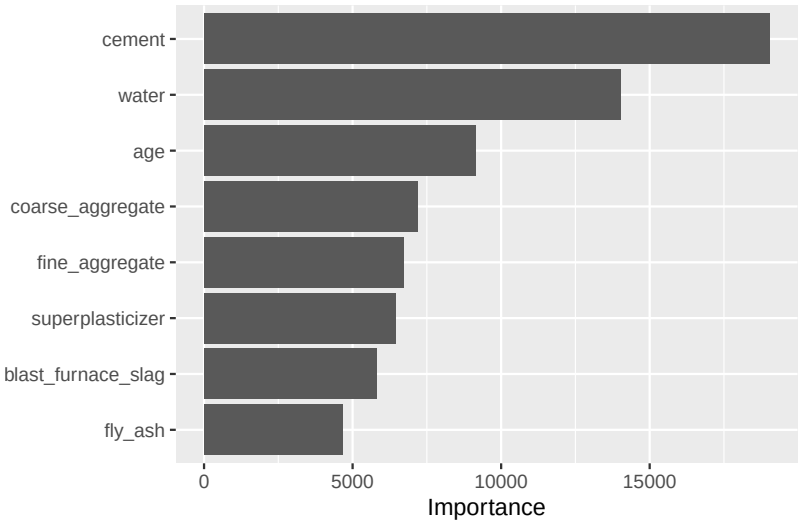
For a linear regression, nothing special is needed. Notice that the interaction features (the ones with `_x_` in the middle) show up as well, as more important than some of the main features.

```
library(vip)
fit(linreg2, train) %>%
  extract_fit_parsnip() %>%
  vip(num_features = 20)
```



For a random forest, you need to specify what sort of "importance" you want to plot. Here we'll follow the approach of the tidymodels official tutorial. Notice that the VIP does not show interaction features. Think: does that mean that the model does not use interactions between multiple features?

```
rf_model <- workflow(
  preprocessor = recipe(compressive_strength ~ ., data = train),
  spec = rand_forest(mode = "regression") %>%
    set_engine("ranger", importance = "impurity")
)
fit(rf_model, train) %>%
  extract_fit_parsnip() %>%
  vip(num_features = 20)
```



(Answer to thought question: *no*; the model must use interactions because of how it's structured: each tree can only look at one variable at a time per split.)

Part 2: Pretrained Perceptual Models (10pt)

Try out two different pre-built neural network models that might be useful in your industry or area of interest. (See below for tips on finding ones.)

For each model, write a brief summary (bullet point is fine) including:

- What model you chose. Include the full URL or enough detail that someone else could find it.
- A specific example where it works well (copy and paste the input and output, if you can). You may use public data as examples (e.g., Wikipedia, review sites, news articles, etc.).
- A specific example where it breaks (returns incorrect or inappropriate results).
- An initial reaction of whether you think this model would be useful for your industry:
  - How easy was it to break it?

  - How did its behavior compare with your expectations?
  - Is it solving a useful problem already, without fine-tuning?
  - Could it be fine-tuned to solve a useful problem? (Might the model's existing behavior give it a head start on learning the task you might want it to do?)

Where to find a model?:

- Hugging Face
  - Tasks
  - Spaces (quality can vary widely!)
  - Models
- Papers With Code
- NVIDIA AI Demos
- TensorFlow demos