HW 8 - Basic Modeling Practice

This homework is meant to give you a chance to do some structured practice with fitting linear models in R.

Data

We will use a dataset from the UCI Machine Learning Repository. This data set is about bike sharing rentals and is available at the assignment link. You can learn more about the data here. The data is available at https://www4.stat.ncsu.edu/~online/datasets/SeoulBikeData.csv

The data description describes the following variables:

- Date : **day/month/year**
- Rented Bike count - Count of bikes rented at each hour
- Hour - Hour of the day
- Temperature-Temperature in Celsius
- Humidity - %
- Windspeed - m/s
- Visibility - 10m
- Dew point temperature - Celsius
- Solar radiation - MJ/m2
- Rainfall - mm
- Snowfall - cm
- Seasons - Winter, Spring, Summer, Autumn
- Holiday - Holiday/No holiday
- Functional Day - NoFunc(Non Functional Hours), Fun(Functional hours)

To Do:

Create a document that goes through your process of

- reading in the data
- conducting a basic EDA (checking for missingness, creating some numerical and some graphical summaries, and doing some data manipulations)
- splitting the data into a training and test set using `tidymodels`
- fitting three different multiple linear regression models on the training data using 10 fold cross-validation to pick a best model on the training data
- fitting your chosen model on the entire training data set and predicting on the test set, reporting the RMSE

More details are below!

Reading Data

- First read in the data
- When using `readr::read_csv()` I got an error `Error in nchar(x, "width") : invalid multibyte string, element 1`
- Google this and it is a quick fix!

EDA

Checking the Data

You should have a narrative through what you are doing here! Steps:

1. check for missingness
2. Check the column types and the values within the columns to make sure they make sense (basic summary stats for numeric columns and check the unique values for the categorical variables).
3. Convert the `Date` column into an actual date (if need be). Recall the `lubridate` package.
4. Turn the character variables (`Seasons`, `Holiday`, and `Functioning Day`) into factors.
5. Lastly, rename the all the variables to have easy to use names (I use lower snake case but whatever you'd like is fine)
6. Create summary statistics (especially related to the bike rental count). These should be done across your categorical variables as well. You should notice something about the `Functioning Day` variable. Subset the data appropriately based on that.
7. To simplify our analysis, we'll summarize across the hours so that each day has one observation associated with it.
    - (I'm using my new names here. Your names may not match and that's ok!) Let's `group_by()` the `date`, `seasons`, and `holiday` variables.
    - Find the sum of the `bike_count`, `rainfall`, and `snowfall` variables
    - Find the mean of all the weather related variables.
    - This will be our new data that we'll analyze!
8. Recreate your basic summary stats and then create some plots to explore relationships. Report correlation between your numeric variables as well.

Again, you should have a narrative throughout this!

Split the Data

- Use functions from `tidymodels` to split the data into a training and test set (75/25 split). Use the `strata` argument to stratify the split on the `seasons` variable.
- On the training set, create a 10 fold CV split

Fitting MLR Models

First, let's create some recipes.

For the 1st recipe:

- Let's ignore the `date` variable for modeling (so we'll need to remove that or give it a different ID) but use it to create a weekday/weekend (factor) variable. (See step 2 of the `shinymodels` tutorial! You can use `step_date()` then `step_mutate()` with a `factor(if_else(...))` to create the variable. I then had to remove the intermediate variable created.)
- Let's standardize the numeric variables since their scales are pretty different.
- Let's create dummy variables for the seasons, holiday, and our new day type variable

For the 2nd recipe:

- Do the same steps as above.
- Add in interactions between seasons and holiday, seasons and temp, temp and rainfall. For the seasons interactions, you can use `starts_with()` to create the proper interactions.

For the 3rd recipe:

- Do the same as the 2nd recipe.
- Add in quadratic terms for each numeric predictor

Now set up our linear model fit to use the "lm" engine.

Fit the models using 10 fold CV via `fit_resamples()` and consider the training set CV error to choose a best model.

Using your 'best' model, fit the model to the entire training data set (use the `last_fit()` function).

- Compute the RMSE metric on the test set.
- Obtain the final model (fit on the entire training set) coefficient table using `extract_fit_parsnip()` and `tidy()`.