Week_8_Problem_Set

Dan Crownover

#Setup

We are interested in modeling state-level residential carbon production per capita, so we need to calculate a per capita value. Our units are now metric tons carbon per capita.

[1] 0.4393548 2.2527882 0.3285348 0.6858903 0.6281212 1.4714082

We want to whether the following variables help predict residential carbon per capita. We will use the same explanatory variables from week 5:

temp: Average annual temperature (degrees F) trumpvote: Percent of state that voted for President Trump climatechange: standardized climate change google search share bachelorsdegree: % of state population with Bachelor's Degree Incomepercapitaus: Median per capita income (USD) rps: Renewable portfolio standard (0/1) = Renewable Portfolio Standards mandates a certain percentage of energy production from renewable sources. States that have an RPS have been coded a 1 and states without a zero (0). urban_percent: Percent of state population that is urban West: the state is located in the west

Create a dataframe with only the variables you are interested in modeling. This is so that we can directly compare it to the linear models you all ran a few weeks ago. In practice, when doing machine learning, you would let R pick out the most important variables.

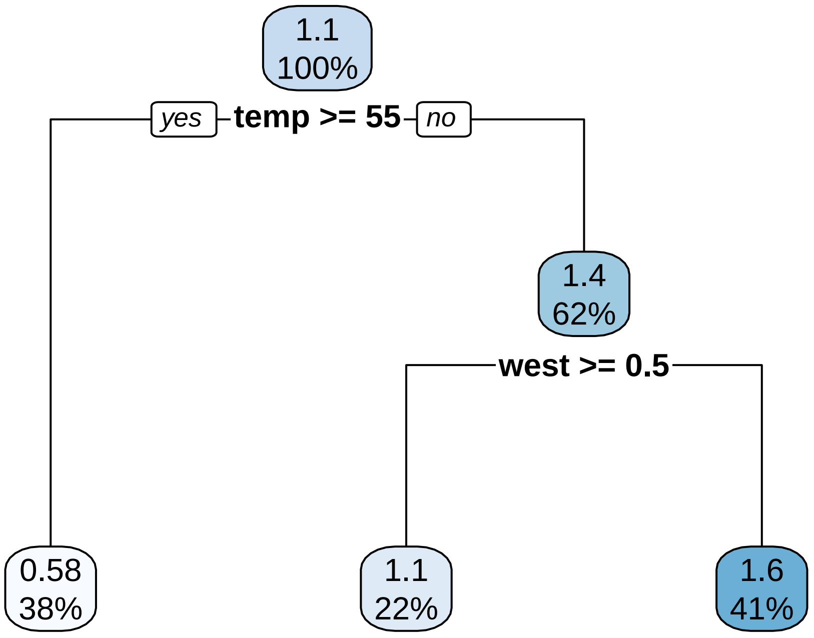
```
carbon <- carbon %>% dplyr::select(temp, trumpvote, climatechange, bachelorsdegree, incomepercapitaus, r
carbon <- carbon %>% drop_na
```

#Problem 1 Build a decision tree using tidymodels with res.carbon.pc.mt as the response variable, and the rest as the predictor variables. Train the tree on 3/4 of the data. Print the tree (7 points)

Code:

```
# Define the decision tree and tell it the the dependent
# variable is continuous ('mode' = 'regression')


help(package = "tidyflow")

library(devtools)

library(tidyr)
library(tidymodels)
library(tidyflow)
library(rpart.plot)
library(vip)
library(baguette)

mod1 <- set_engine(decision_tree(mode = "regression"), "rpart")

tflow <-
```

```r
  # Plug the data
  carbon %>%
  # Begin the tidyflow
  tidyflow(seed = 23151) %>%
  # Separate the data into training/testing (we are keeping 3/4 of the data for training)
  plug_split(initial_split, prop = 3/4) %>%
  # Plug the formula
  plug_formula(res.carbon.pc.mt ~ temp + trumpvote + climatechange + bachelorsdegree + incomepercapitau
  # Plug the model
  plug_model(mod1)

vanilla_fit <- fit(tflow)
tree <- pull_tflow_fit(vanilla_fit)$fit
rpart.plot(tree)
```



Question: Interpret the tree in words. What is the most important variable for predicting residential carbon? (3 points) Answer: the top-most box which says temp >= 55 is the root node. This is the most important variable that predicts residental carbon. Inside the blue box you can see two numbers: 100% which means that the entire sample is present in this node and the number 1.1, the average residental carbon concentration for the entire sample. if temp is greater than 55, then 38% of the entire sample is present in this node and the average concentration of residental carbon for 38% of the sample is .58 units. If residental carbon is less that 55 then 62% of the data is represented in this node and the average residental carbon is 1.4 units. Out of 62% of the data, if tempurature is greater or equal to 0.5, then 22% of the data will have a average concentration of residental carbon of 1.1 units. Out of 62% of the data, if temp is less than .5 units, then 41% of the data will have a average residental carbon cocentration of 1.6 units.

#Problem 2 Build a bagged tree using tidymodels on the same data. Train the tree on 3/4 of the data. Bootstrap 100 times. (7 points)

Code:

```
btree <- bag_tree(mode = "regression") %>% set_engine("rpart", times = 100)

tflow <-
  carbon %>%
  tidyflow(seed = 566521) %>%
  plug_split(initial_split, prop = 3/4) %>%
  plug_formula(res.carbon.pc.mt ~ temp + trumpvote + climatechange + bachelorsdegree + incomepercapitau
  plug_model(btree)

tflow
```

== Tidyflow ==
Data: 50 rows x 9 columns Split: initial_split w/ prop = ~3/4 Formula: res.carbon.pc.mt ~ temp + trumpvote + climatechange + bachelorsdegree + incomepercapitaus + rps + urban_percent + west Resample: None Grid: None Model: Bagged Decision Tree Model Specification (regression)

Main Arguments: cost_complexity = 0 min_n = 2

Engine-Specific Arguments: times = 100

Computational engine: rpart

```
res_problem2 <- tflow %>% fit()
```

#Problem 3 Build a random forest using tidymodels on the same data. Train the forest on 3/4 of the data. Set mtry to 1/3 of the predictor variables. Don't worry about tuning any of the other parameters. Print the variable importance (7 points)

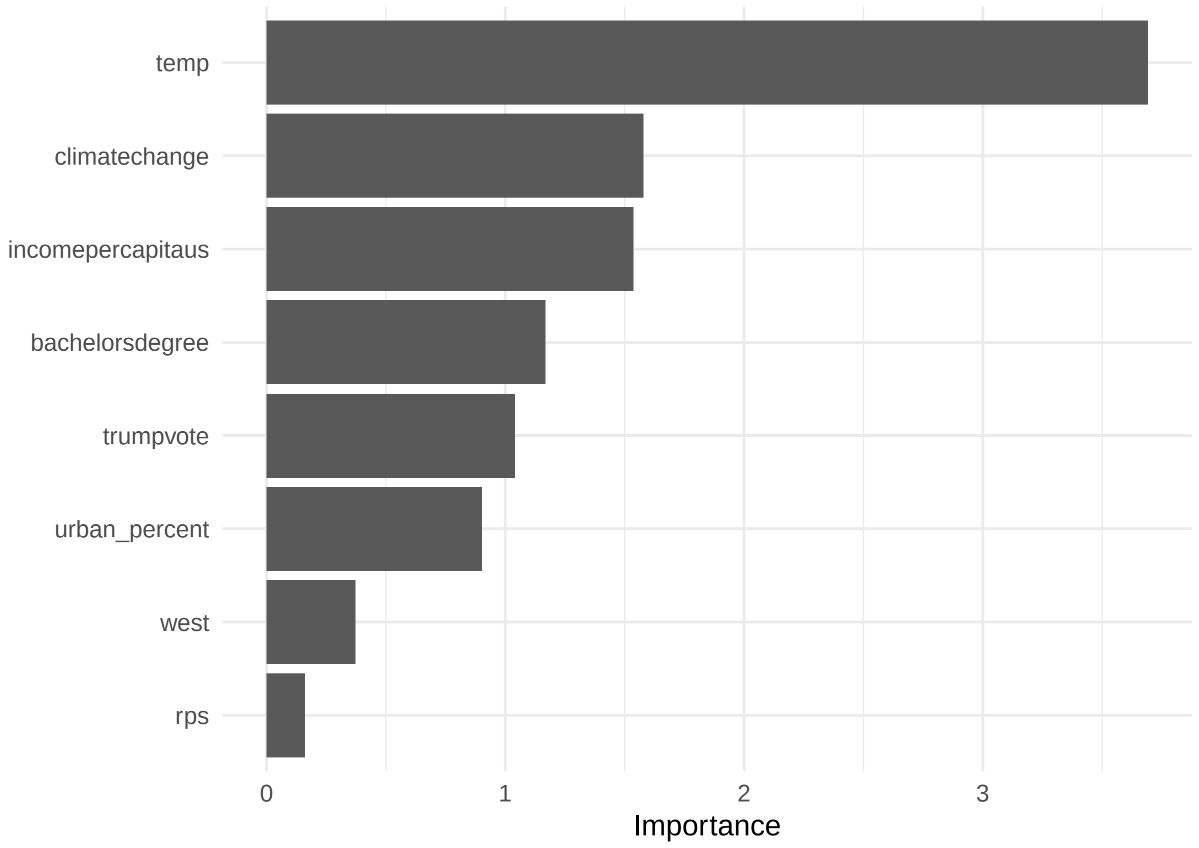
Code:

```
# Define the random forest
rf_mod <-
  rand_forest(mode = "regression",  mtry = 3) %>%
  set_engine("ranger", importance = "impurity")

# Define the `tidyflow` with the random forest model
# and include all variables (including scie_score and read_score)
tflow <-
  carbon %>%
  tidyflow(seed = 23151) %>%
  plug_formula(res.carbon.pc.mt ~ temp + trumpvote + climatechange + bachelorsdegree + incomepercapitau
  plug_split(initial_split, prop = 3/4) %>%
  plug_model(rf_mod)

res_problem3 <- tflow %>% fit()

res_problem3 %>%
  predict_training() %>%
  rmse(res.carbon.pc.mt, .pred)
```

A tibble: 1 x 3

.metric .estimator .estimate 1 rmse standard 0.189

```r
res_problem3 %>%
  pull_tflow_fit() %>%
  .[['fit']] %>%
  vip() +
  theme_minimal()
```



Question: How does random forest rank variable importance? (3 points) Answer: Random Forest ranks variable importance based on the decrease in node impurity that each variable causes when it is used to split the data at a node in the trees of the forest. This decrease in impurity is measured using the metric of Gini impurity. Specifically, during the construction of each tree in the forest, at each split, the algorithm considers a subset of variables and selects the one that maximally reduces the impurity in the resulting child nodes. The importance of a variable is then calculated as the total decrease in node impurity averaged over all trees in the forest where that variable was used for splitting. In essence, variables that lead to greater reductions in impurity across the forest are considered more important, as they contribute more to the overall predictive power of the Random Forest model.

#Problem 4 Build a boosted regression tree using tidymodels on the same data. Train on 3/4 of the data. Set ntree to 500. Don't worry about tuning any of the other parameters (7 points)

Code:

```r
library(xgboost)
boost_mod <-
  boost_tree(mode = "regression", trees = 500) %>%
  set_engine("xgboost")

tflow <-
  carbon %>%
  tidyflow(seed = 51231) %>%
  plug_formula(res.carbon.pc.mt ~ temp + trumpvote + climatechange + bachelorsdegree + incomepercapitaus
  plug_split(initial_split, prop = 3/4) %>%
  plug_model(boost_mod)

res_boostproblem4 <- fit(tflow)
```

#Problem 5 Build a multiple linear regression using tidymodels and the same data and formula as above (hint - google parsnip linear models). Train on 3/4 of the data. (7 points)

Code:

```r
 plug_lin <- linear_reg(mode = "regression", engine = "lm")



tflow <-
  carbon %>%
  tidyflow(seed = 51231) %>%
  plug_formula(res.carbon.pc.mt ~ temp + trumpvote + climatechange + bachelorsdegree + incomepercapitaus
  plug_split(initial_split, prop = 3/4) %>%
  plug_model(plug_lin)

res_boost_linearmodelProb5 <- fit(tflow)
```

#Problem 6 Calculate the RMSE for each of the 5 modeling techniques above on the in-sample (training) data and the out-of-sample (testing data). Provide a table of all 10 values. Which modelling technique performed best in and out of sample? (9 points)

Code:

```r
##### problem 1 rmse
vanilla_fit_training_rmse <-
  vanilla_fit %>%
  predict_training() %>%
  rmse(res.carbon.pc.mt, .pred)

vanilla_fit_training_rmse
```

A tibble: 1 x 3

.metric .estimator .estimate 1 rmse standard 0.347

```
vanilla_fit_testing_rmse <-
  vanilla_fit %>%
  predict_testing() %>%
  rmse(res.carbon.pc.mt, .pred)

vanilla_fit_testing_rmse
```

A tibble: 1 x 3

.metric .estimator .estimate 1 rmse standard 0.516

```
##### problem 2 rmse

res_problem2_training_rmse <-
  res_problem2 %>%
  predict_training() %>%
  rmse(res.carbon.pc.mt, .pred)

res_problem2_training_rmse
```

A tibble: 1 x 3

.metric .estimator .estimate 1 rmse standard 0.154

```
res_problem2_testing_rmse <-
  res_problem2 %>%
  predict_testing() %>%
  rmse(res.carbon.pc.mt, .pred)

res_problem2_testing_rmse
```

A tibble: 1 x 3

.metric .estimator .estimate 1 rmse standard 0.311

```
##### problem 3 rmse


res_problem3_training_rmse <-
  res_problem3 %>%
  predict_training() %>%
  rmse(res.carbon.pc.mt, .pred)

res_problem3_training_rmse
```

A tibble: 1 x 3

.metric .estimator .estimate 1 rmse standard 0.189

```r
res_problem3_testing_rmse <-
  res_problem3 %>%
  predict_testing() %>%
  rmse(res.carbon.pc.mt, .pred)

res_problem3_testing_rmse
```

A tibble: 1 x 3

.metric .estimator .estimate 1 rmse standard 0.441

```r
#### problem 4 rmse

res_boostproblem4_training_rmse <-
  res_boostproblem4 %>%
  predict_training() %>%
  rmse(res.carbon.pc.mt, .pred)

res_boostproblem4_training_rmse
```

A tibble: 1 x 3

.metric .estimator .estimate 1 rmse standard 0.000688

```r
res_boostproblem4_testing_rmse <-
  res_boostproblem4 %>%
  predict_testing() %>%
  rmse(res.carbon.pc.mt, .pred)

res_boostproblem4_testing_rmse
```

A tibble: 1 x 3

.metric .estimator .estimate 1 rmse standard 0.492

```r
#### problem 5 rmse


res_boost_linearmodelProb5_training_rmse <-
  res_boost_linearmodelProb5 %>%
  predict_training() %>%
  rmse(res.carbon.pc.mt, .pred)

res_boost_linearmodelProb5_training_rmse
```

A tibble: 1 x 3

.metric .estimator .estimate 1 rmse standard 0.217

```
res_boost_linearmodelProb5_testing_rmse <-
  res_boost_linearmodelProb5 %>%
  predict_testing() %>%
  rmse(res.carbon.pc.mt, .pred)

res_boost_linearmodelProb5_testing_rmse
```

A tibble: 1 x 3

.metric .estimator .estimate 1 rmse standard 0.396

```
library(knitr)

# Data
data <- data.frame(
  Model = c("vanilla_fit", "vanilla_fit", "res_problem2", "res_problem2", "res_problem3", "res_problem3
  Set = c("training_rmse", "testing_rmse", "training_rmse", "testing_rmse", "training_rmse", "testing_r
  Value = c(0.3471506, 0.5155675, 0.1538571, 0.311066, 0.1889467, 0.4408348, 0.0006881222, 0.4920126, 0
)

# Reshape data to wide format
data_wide <- reshape(data, idvar = "Model", timevar = "Set", direction = "wide")

# Rename columns
colnames(data_wide) <- c("Model", "Training RMSE", "Testing RMSE")

# Print the table
kable(data_wide, caption = "RMSE values for different models and datasets")
```

Table 1: RMSE values for different models and datasets

	Model	Training RMSE	Testing RMSE
1	vanilla_fit	0.3471506	0.5155675
3	res_problem2	0.1538571	0.3110660
5	res_problem3	0.1889467	0.4408348
7	res_boostproblem4	0.0006881	0.4920126
9	res_boost_linearmodelProb5	0.2174126	0.3957971

Answer:

Based on the values:

For the Training RMSE: Res Boost Problem 4 has the lowest value (0.0006881), indicating better performance in the training dataset.

For the Testing RMSE: Res Problem 2 has the lowest value (0.3110660), indicating better performance in the testing dataset. In summary, Res Boost Problem 4 performed best in the training dataset, while Res Problem 2 performed best in the testing dataset.