Midterm: Predicting Employee Behaviors and Outcomes with Machine Learning

Emma Kruis

2020-02-01

Instructions

This script reviews *Machine Learning* as part of the *Midterm Review*. You will use content from the lecture and assignment materials on *Machine Learning* to complete this script. You will *copy and paste* relevant code from those files into this script and answer the associated questions for each task. You will respond to questions in each section after executing relevant code to answer a question. You will submit this script to its *Submissions* folder on *D2L* as part of the *Midterm Review*. For this script, you will submit *two* files:

1. this completed *R Markdown* script, and
2. as a first preference, a *PDF* (if you already installed `TinyTeX` properly), as a second preference, a *Microsfot Word* (if your computer has *Microsoft Word*) document, or, as a third preference, an *HTML* (if you did *not* install `TinyTeX` properly and your computer does *not* have *Microsoft Word*) file to *D2L*.

For the *Midterm Review*, create the project directory: *~/mgt_592/assignments/midterm_review*. Convert your project directory into a formal *R Project* directory by going to the *File* menu in *RStudio*, selecting *New Project...*, choosing *Existing Directory*, and going to your *~/mgt_592/assignments/midterm_review* folder to select it as the top-level directory for this **R Project**.
The project directory should contain the following folders: *scripts*, *data*, and *plots*. Store this script in the *scripts* folder and the relevant data in the *data* folder.

Global Settings

The first code chunk sets the global settings for the remaining code chunks in the document. Do *not* change anything in this code chunk.

Task 1: Load Libraries

For this task, you will load the libraries you need for this script.

Task 1.1

In this code chunk, load the following packages:

1. **here**,
2. **tidyverse**,
3. **ggthemes**,
4. **tidymodels**,

5. **skimr**,
6. **corrr**, and
7. **vip**.

Make sure you installed these packages when you reviewed the analytical lecture.

We will use functions from these packages to examine the data. Do *not* change anything in this code chunk.

```
### load libraries for use in current working session
## here for project work flow
library(here)
```

```
## here() starts at /Users/emmakruis/Library/Mobile Documents/com~apple~CloudDocs/year_2/WQ21/mgt_592/a
```

```
## tidyverse for data manipulation and plotting
# loads eight different libraries simultaneously
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.0     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
## ggthemes for plot themes
library(ggthemes)

## tidymodels for modeling
# loads ten different libraries simultaneously
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 0.1.2 --

## v broom     0.7.5      v recipes   0.1.15
## v dials     0.0.9      v rsample   0.0.9
## v infer     0.5.4      v tune      0.1.3
## v modeldata 0.1.0      v workflows 0.2.2
## v parsnip   0.1.5      v yardstick 0.0.7

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
```

```r
## skimr to summarize data
library(skimr)

## corrr for correlation matrices
library(corrr)
```

```
##
## Attaching package: 'corrr'

## The following object is masked from 'package:skimr':
##
##     focus
```

```r
## vip for variable importance
library(vip)
```

```
##
## Attaching package: 'vip'

## The following object is masked from 'package:utils':
##
##     vi
```

Task 2: Import Data

For this task, you will import the data file: **staffing.tsv**.

Task 2.1

Use the **read_tsv()** and **here()** functions to load the data file for this working session. Save the data as the object **staff_raw**.

Make a copy of the data and name the copy: **staff_work**. Use the **glimpse()** function to view a preview of values for each variable in **staff_work**. Remove **staff_raw** from your *global environment*.

```r
### import data objects
## use read_tsv() and here() to import the data file
staff_raw <- read_tsv(
  ## use here() to locate file in our project directory
  here("data", "staffing.tsv")
)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   id = col_double(),
##   proactive = col_double(),
##   emot_intel = col_double(),
##   sjt = col_double(),
##   work_samp = col_double(),
```

```
##   str_int = col_double(),
##   consc = col_double(),
##   cog_flex = col_double(),
##   work_exp = col_character(),
##   degree = col_character(),
##   job_perf = col_double(),
##   citizenship = col_double(),
##   promotion = col_character(),
##   high_potential = col_character()
## )
```

```
### make working copy of data
## save as object
staff_work <- staff_raw

## preview data
glimpse(staff_work)
```

```
## Rows: 17,807
## Columns: 14
## $ id             <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, ~
## $ proactive      <dbl> 48, 59, 52, 53, 57, 57, 55, 52, 53, 50, 47, 55, 53, 50,~
## $ emot_intel     <dbl> 41, 51, 49, 50, 50, 44, 46, 47, 45, 49, 44, 40, 45, 43,~
## $ sjt            <dbl> 44, 51, 51, 52, 46, 49, 50, 43, 49, 54, 42, 51, 48, 48,~
## $ work_samp      <dbl> 45, 52, 46, 49, 51, 51, 49, 45, 45, 44, 45, 46, 47, 45,~
## $ str_int        <dbl> 49, 51, 52, 47, 51, 50, 44, 50, 48, 54, 47, 52, 48, 49,~
## $ consc          <dbl> 54, 54, 51, 51, 55, 54, 54, 56, 52, 48, 52, 50, 48, 53,~
## $ cog_flex       <dbl> 47, 48, 48, 51, 50, 44, 39, 51, 40, 50, 42, 42, 49, 44,~
## $ work_exp       <chr> "2-5", "0-1", "0-1", "0-1", "6-10", "0-1", "0-1", "6-10~
## $ degree         <chr> "none", "none", "none", "bachelor", "bachelor", "associ~
## $ job_perf       <dbl> 43, 58, 49, 40, 58, 50, 47, 52, 45, 45, 37, 37, 41, 45,~
## $ citizenship    <dbl> 40, 46, 47, 51, 48, 39, 45, 48, 38, 47, 37, 45, 48, 41,~
## $ promotion      <chr> "No", "No", "No", "No", "No", "No", "Yes", "No", "No", ~
## $ high_potential <chr> "No", "No", "Yes", "No", "No", "No", "Yes", "Yes", "No"~
```

```
## remove raw copy of data
rm(staff_raw)
```

Task 3: Clean and Prepare Data

For this task, you will clean and prepare the data.

Task 3.1

Perform the following cleaning tasks to update **staff_work**:

1. mutate all character variables to factor variables,
2. *relabel* **degree** so that its factor levels use a capital first letter,
3. change the **Masters** and **Doctorate** factor levels of **degree** to **Master** and **Doctor**, respectively,
4. *relevel* the **degree** and **work_exp** factors in a logical order, and
5. change **degree** and **work_exp** to be *ordered* factors.

Use **glimpse()** to preview the updated **staff_work** data object.

```
### convert variables
## overwrite working data
staff_work <- staff_work %>%
  ## mutate variable types and values
  mutate(
    ## characters to nominal factors
    across(
      # find character variables
      .cols = where(is_character),
      # convert to factors
      .fns = as_factor
    ),
    ## change factor labels
    degree = fct_relabel(
      # factor
      degree,
      # function
      str_to_title
    ),
    ## change factor labels
    degree = fct_recode(
      # factor
      degree,
      # change level
      "Master" = "Masters",
      # change level
      "Doctor" = "Doctorate"
    ),
    ## change order
    degree = fct_relevel(
      # factor
      degree,
      # order of levels
      "Associate", "Bachelor", "Master",
      # after
      after = 1
    ),
    ## change order
    work_exp = fct_relevel(
      # factor
      work_exp,
      # order of levels
      "0-1"
    ),
    ## convert to ordered factors
    across(
      # columns
      .cols = c(work_exp, degree),
      # function
      .fns = factor,
      # argument to function
      ordered = TRUE
```

```
  )
 )

## glimpse data to confirm changes
glimpse(staff_work)
```

```
## Rows: 17,807
## Columns: 14
## $ id             <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, ~
## $ proactive      <dbl> 48, 59, 52, 53, 57, 57, 55, 52, 53, 50, 47, 55, 53, 50,~
## $ emot_intel     <dbl> 41, 51, 49, 50, 50, 44, 46, 47, 45, 49, 44, 40, 45, 43,~
## $ sjt            <dbl> 44, 51, 51, 52, 46, 49, 50, 43, 49, 54, 42, 51, 48, 48,~
## $ work_samp      <dbl> 45, 52, 46, 49, 51, 51, 49, 45, 45, 44, 45, 46, 47, 45,~
## $ str_int        <dbl> 49, 51, 52, 47, 51, 50, 44, 50, 48, 54, 47, 52, 48, 49,~
## $ consc          <dbl> 54, 54, 51, 51, 55, 54, 54, 56, 52, 48, 52, 50, 48, 53,~
## $ cog_flex       <dbl> 47, 48, 48, 51, 50, 44, 39, 51, 40, 50, 42, 42, 49, 44,~
## $ work_exp       <ord> 2-5, 0-1, 0-1, 0-1, 6-10, 0-1, 0-1, 6-10, 11+, 0-1, 0-1~
## $ degree         <ord> None, None, None, Bachelor, Bachelor, Associate, Associ~
## $ job_perf       <dbl> 43, 58, 49, 40, 58, 50, 47, 52, 45, 45, 37, 37, 41, 45,~
## $ citizenship    <dbl> 40, 46, 47, 51, 48, 39, 45, 48, 38, 47, 37, 45, 48, 41,~
## $ promotion      <fct> No, No, No, No, No, No, Yes, No, No, Yes, No, No, No, N~
## $ high_potential <fct> No, No, Yes, No, No, No, Yes, Yes, No, No, Yes, No, No,~
```

Task 4: Examine Data

For this task, you will examine the data.

Task 4.1

Use **staff_work** and **ggplot()** to make *faceted scatterplots* of *citizenship* against *cognitive flexibility* scores for different levels of *work experience*. Place **cog_flex** on the *x-axis* and **citizenship** on the *y-axis* and *fill* by **degree**. Call the *point* and *smooth* geometries with appropriate settings. Use **facet_wrap** on **work_exp**. Appropriately combine **as_labeller()**, **setNames()**, **paste()**, and **levels()** to correctly label the facets. For the labels, paste **Work Exp.** (note the capital first letter) with the *levels* of **work_exp** with a *colon* separator. Scale the *y-axis* and *color* appropriately. Use appropriate labels for the axes and legend. Use **theme_hc()** and remove the legend.

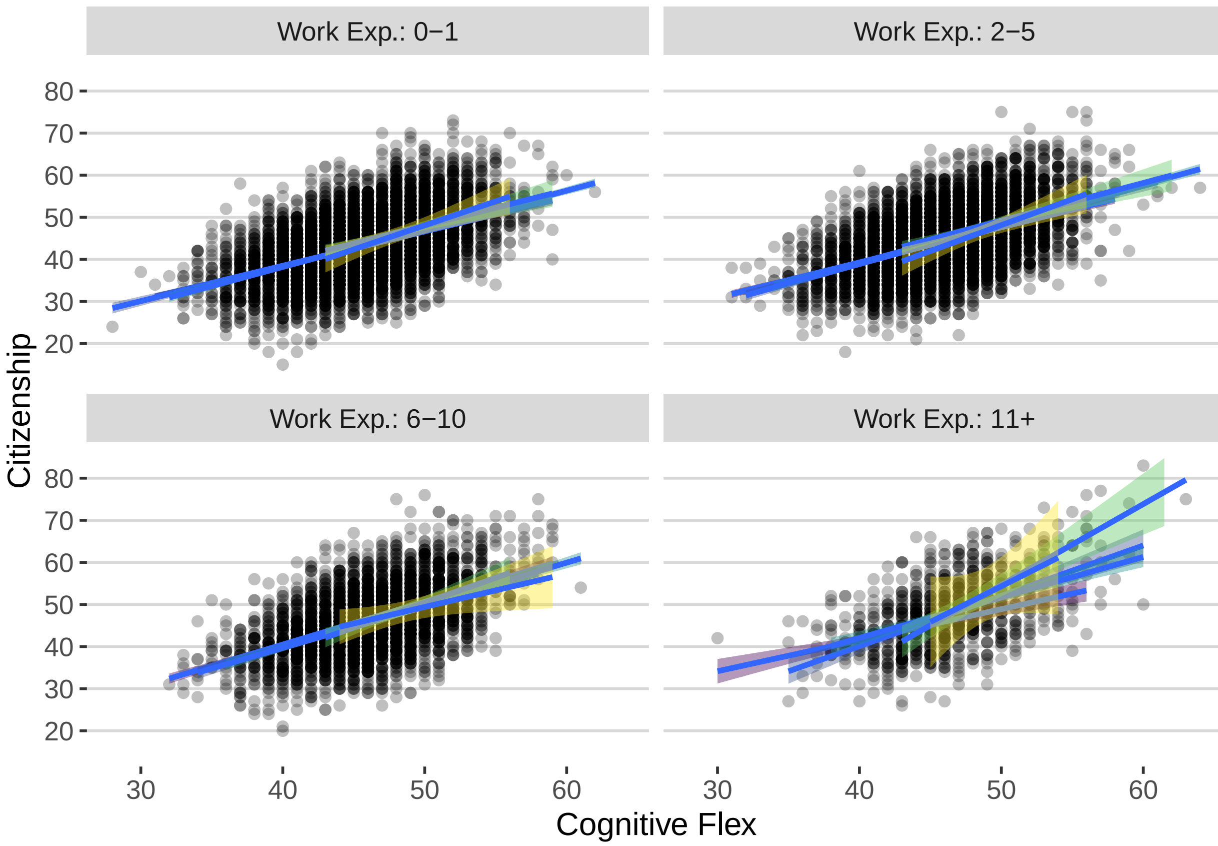
Question 4.1: Do you see much of a *difference in the relationship* between *cognitive flexibility* and *citizenship* scores across the levels of *work experience*?

Response 4.1: *Relatively all the same except for the work experience of the 11+ group.*

```
### examine relations by categories
## call plot
ggplot(
  # data
  staff_work,
  # mapping
  aes(
    # x-axis
    x = cog_flex,
    # y-axis
```

```r
    y = citizenship,
    # factor
    fill = degree
  )
) +
  ## points
  geom_point(alpha = 0.25) +
  ## fit linear model
  geom_smooth(method = "lm") +
  ## facets
  facet_wrap(
    # variable
    vars(work_exp),
    # number of rows
    nrow = 2,
    # labels
    labeller = as_labeller(
      # look-up table
      setNames(
        # vector elements
        paste("Work Exp.", levels(staff_work$work_exp), sep = ": "),
        # names of elements
        levels(staff_work$work_exp)
      )
    )
  ) +
  ## scale y-axis
  scale_y_continuous(limits = c(15, 85), n.breaks = 8) +
  ## scale fill
  scale_color_brewer(palette = "Dark2") +
  ## labels
  labs(x = "Cognitive Flex", y = "Citizenship", fill = "Work Exp.") +
  ## define theme
  theme_hc() +
  ## remove legend
  theme(legend.position = "none")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Task 5: Split Data

For this task, you will split the data into a training and testing set. Then, you will create cross-validation folds for the training set.

Task 5.1

Split **staff_work** into a *training* and *testing* set. Use random seed **1959**. Call **initial_split()** and create an *80%* split using **citizenship** as the *stratification* variable. Save the split as **staff_split**.

Extract the *training* set with **training()** and save it as **staff_train**. Extract the *testing* set with **testing()** and save it as **staff_test**.

```
### make an initial split of the data
## set seed
## initial application of neural networks
set.seed(1959)



## split data
staff_split <- initial_split(
  # data
  staff_work,
  # split proportion
```

```r
  prob = 0.8,
  # stratify by an outcome
  strata = citizenship
)



## examine initial split
staff_split
```

```
## <Analysis/Assess/Total>
## <13357/4450/17807>
```

```r
### extract training and testing sets
## training
staff_train <- training(staff_split)

## testing
staff_test <- testing(staff_split)
```

Task 5.2

Split **staff_train** into repeated folds. Use random seed **1959**. Call **vfold_cv()** and set the *number of folds* to **3** and *number of repeats* to **2**. Use **citizenship** as the *stratification* variable. Save the split as **staff_train_folds**.

```r
## set seed
set.seed(1959)


## split training
staff_train_folds <- vfold_cv(
  # training data
  staff_train,
  # number of folds
  v = 3,
  # repeats
  repeats = 2,
  # stratify by an outcome
  strata = citizenship
)



## examine folds
staff_train_folds
```

```
## #  3-fold cross-validation repeated 2 times using stratification
## # A tibble: 6 x 3
##   splits              id      id2
##   <list>              <chr>   <chr>
```

```
## 1 <split [8904/4453]> Repeat1 Fold1
## 2 <split [8905/4452]> Repeat1 Fold2
## 3 <split [8905/4452]> Repeat1 Fold3
## 4 <split [8904/4453]> Repeat2 Fold1
## 5 <split [8905/4452]> Repeat2 Fold2
## 6 <split [8905/4452]> Repeat2 Fold3
```

Task 6: Data Preparation

For this task, you will create a modeling recipe using the training data.

Task 6.1

Create a recipe named **staff_rec**. Use **recipe()** on **staff_train** and specify **citizenship** as the only *outcome* variable and the remaining variables as *predictor* variables. Add a removal step to the recipe using **step_rm()** and remove **id**, **job_perf**, **high_potential**, and **promotion**. Add a normalization step to the recipe using **step_normalize()** and normalize *all predictors* except for the *nominal predictors*. Add a dummystep to the recipe using **step_dummy()** and create dummy variables for *all nominal predictors*.

Use **prep()** and **bake()** on **staff_rec** to view the result of the recipe transformations. Print wide.

Questions 6.1: Answer these questions: (1) How many *variables* are there in the *baked* recipe? (2) Is the *first* employee in the *training* set *below* or *above* the mean on *situational judgment test* (**sjt**) score?

Responses 6.1: *(1) 14 variables. 1 outcome vairable and 13 predictors. (2) above the mean.*

```
### create modeling recipe
## save as object
staff_rec <- recipe(
  # identify outcomes
  citizenship ~
    # all other variables
    .,
  # data
  data = staff_train
) %>%
  ## remove variables
  step_rm(
    # list variables
    id, job_perf, promotion, high_potential
  ) %>%
  ## normalize predictors
  step_normalize(
    # perform for all predictors
    all_predictors(),
    # except for factors
    -all_nominal()
  ) %>%
  ## dummy variables
  step_dummy(
    # all factor variables
    all_nominal(),
    # except for outcomes
    -has_role("outcome")
```

```
  )



### prep and bake recipe
## call recipe
staff_rec %>%
  ## estimate parameters
  prep() %>%
  ## apply computations to data
  bake(
    # training data
    new_data = NULL
  ) %>%
  ## print wide
  print(width = Inf)
```

```
## # A tibble: 13,357 x 15
##    proactive emot_intel    sjt work_samp str_int   consc cog_flex citizenship
##        <dbl>      <dbl>  <dbl>     <dbl>   <dbl>   <dbl>    <dbl>       <dbl>
##  1     1.84       1.18   0.940     1.81    0.619  0.637     0.547          46
##  2    -0.167      0.670  0.940     0.149   0.875 -0.0813    0.547          47
##  3     0.120      0.923  1.19      0.979  -0.404 -0.0813    1.27           51
##  4     1.27       0.923 -0.327     1.53    0.619  0.876     1.03           48
##  5     1.27      -0.595  0.433     1.53    0.363  0.637    -0.418          39
##  6     0.694     -0.0888 0.686     0.979  -1.17   0.637    -1.62           45
##  7    -0.167      0.164 -1.09     -0.127   0.363  1.12      1.27           48
##  8     0.120     -0.342  0.433    -0.127  -0.148  0.158    -1.38           38
##  9    -1.60      -0.595 -1.34     -0.127  -0.404  0.158    -0.900          37
## 10     0.120     -0.342  0.180     0.426  -0.148 -0.800     0.788          48
##    work_exp_1 work_exp_2 work_exp_3 degree_1 degree_2  degree_3 degree_4
##         <dbl>      <dbl>      <dbl>    <dbl>    <dbl>     <dbl>    <dbl>
##  1     -0.671        0.5     -0.224   -0.632    0.535 -3.16e- 1    0.120
##  2     -0.671        0.5     -0.224   -0.632    0.535 -3.16e- 1    0.120
##  3     -0.671        0.5     -0.224    0       -0.535 -4.10e-16    0.717
##  4      0.224       -0.5     -0.671    0       -0.535 -4.10e-16    0.717
##  5     -0.671        0.5     -0.224   -0.316   -0.267  6.32e- 1   -0.478
##  6     -0.671        0.5     -0.224   -0.316   -0.267  6.32e- 1   -0.478
##  7      0.224       -0.5     -0.671   -0.316   -0.267  6.32e- 1   -0.478
##  8      0.671        0.5      0.224   -0.632    0.535 -3.16e- 1    0.120
##  9     -0.671        0.5     -0.224   -0.632    0.535 -3.16e- 1    0.120
## 10     -0.224       -0.5      0.671   -0.632    0.535 -3.16e- 1    0.120
## # ... with 13,347 more rows
```

Task 7: Fit Continuous Outcome Models

For this task, you will fit models to predict *citizenship*.

Task 7.1

Create a *metric set* of *mean absolute error*, *root mean squared error*, and *r-squared* named **reg_met**.

Create an *elastic net* model specification named **glmnet_reg_spec**. Use the **linear_reg()** specification and set the **penalty** and **mixture** parameters to **tune()**. Use the **glmnet** engine.

Create a *tuning grid* named **glmnnet_reg_grid**. Specify the *tuning grid* using **glmnet_reg_spec**, **parameters()**, and **regular_grid()** with **levels** set to **2**.

Create an *elastic net* model workflow named **glmnet_reg_wflow** using **workflow()**. Use **add_model()** to add a model using **glmnet_reg_spec**. Use **add_recipe()** to add **staff_rec**.

Create an object named **glmnet_reg_tune** to save fitted models to folds using **glmnet_reg_wflow** and the *tuning grid*. In **tune_grid()**, set the *folds* to **staff_train_folds**, **grid** to **glmnet_reg_grid**, and **metrics** to **reg_met**.

Apply **autoplot()** to **glmnet_reg_tune**. Move the legend to the *top*.

Apply **collect_metrics()** to **glmnet_reg_tune** and print *long* and *wide*.

Apply **show_best()** to **glmnet_reg)tune** and set the **metric** to **rmse**.

Create a *final workflow* named **glmnet_reg_wflow_final** using **glmnet_reg_wflow** and **finalize_workflow()**. Inside of **finalize_workflow()**, create a **tibble()** and set **penalty** to **1e-10** and **mixture** to **0.05**.

Create an object named **glmnet_reg_fit** to save a fitted model to the complete *training* data using **glmnet_reg_wflow_final**. In **fit()**, specify **staff_train**.

Use **pull_workflow_fit()** and **tidy()** on **glmnet_reg_fit** to view the estimated regression coefficients.

Questions 7.1: Answer these questions: (1) What is the *average mean absolute error* across the repeated folds for the *first tuning set*? (2) What is the *value* of the *best average root mean squared error* across the folds? (3) What is the *regression coefficient* for the *linear contrast* of *educational degree* (**degree_1**)? (4) Interpret the *regression coefficient* for *conscientiousness* (**consc**).

Responses 7.1: *(1) 5.22 (2) 5.22 (3) 0.487 (4) For every one unit change in conscientiousness we expect promotion to increase by 1.51 holding the other predictors constant..*

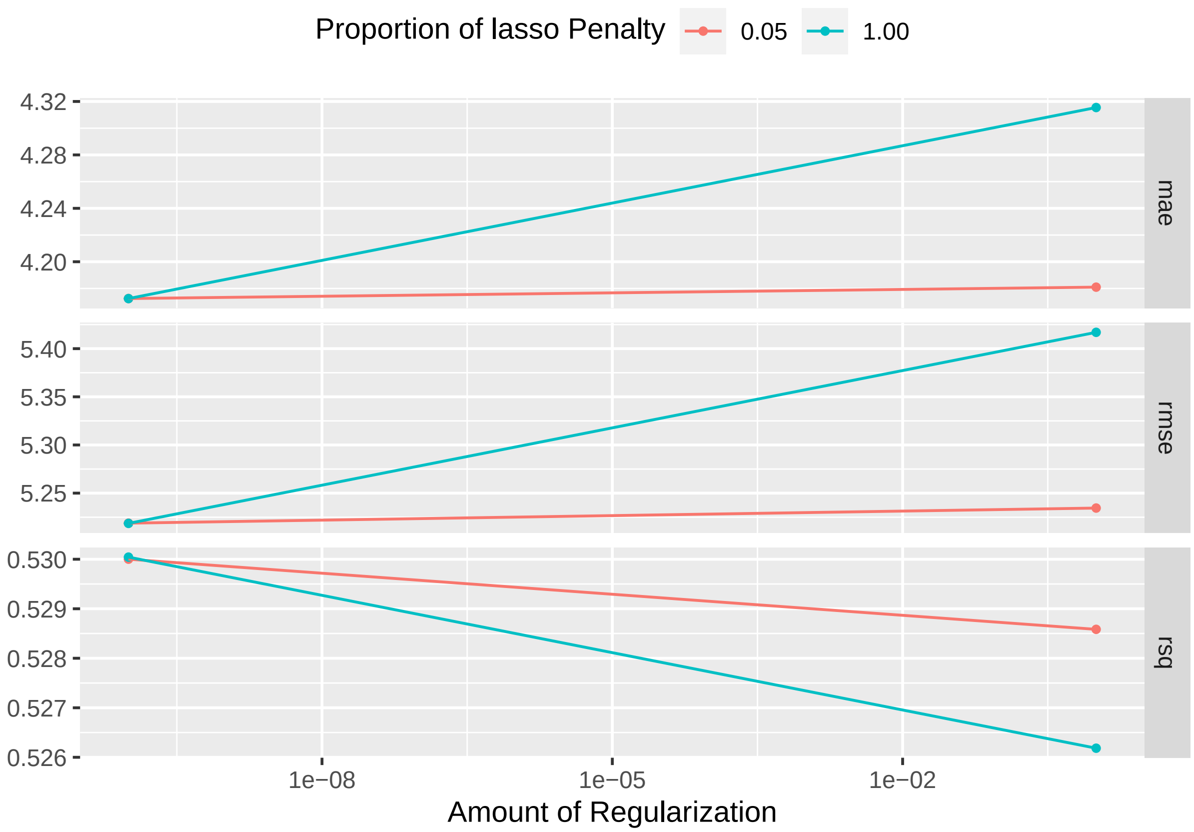
```r
### specify model metric to optimize
## save as object
reg_met <- metric_set(mae, rmse, rsq)


#### elastic net
### model specification
## save as object
glmnet_reg_spec <-
  ## regression specification
  linear_reg(
    # tune penalty
    penalty = tune(),
    # tune mixture
    mixture = tune()
  ) %>%
  ## specify engine
  set_engine("glmnet")




### view a tuning grid
## call model specification
```

```r
glmnet_reg_grid <- glmnet_reg_spec %>%
  ## parameters
  parameters() %>%
  ## grid
  grid_regular(levels = 2)
```

```r
### create initial workflow
## save as object
glmnet_reg_wflow <- workflow() %>%
  ## add model
  add_model(glmnet_reg_spec) %>%
  ## add recipe
  add_recipe(
    # previous recipe
    staff_rec
  )
```

```r
### estimate models
## save as object
glmnet_reg_tune <-
  ## workflow
  glmnet_reg_wflow %>%
  ## tune
  tune_grid(
    # folds
    staff_train_folds,
    # grid
    grid = glmnet_reg_grid,
    # metrics
    metrics = reg_met
  )
```

```r
### plot metrics
## produce plot
autoplot(glmnet_reg_tune) +
  ## move legend
  theme(legend.position = "top")
```




```r
### show metrics
## call function
collect_metrics(glmnet_reg_tune) %>%
  ## print long
  print(n = Inf, width = Inf)
```

```
## # A tibble: 12 x 8
##          penalty mixture .metric .estimator  mean     n std_err
##            <dbl>   <dbl> <chr>   <chr>      <dbl> <int>   <dbl>
##   1 0.0000000001    0.05 mae     standard   4.17      6 0.0268
##   2 0.0000000001    0.05 rmse    standard   5.22      6 0.0243
##   3 0.0000000001    0.05 rsq     standard   0.530     6 0.00190
##   4 1               0.05 mae     standard   4.18      6 0.0281
##   5 1               0.05 rmse    standard   5.23      6 0.0251
##   6 1               0.05 rsq     standard   0.529     6 0.00208
##   7 0.0000000001    1    mae     standard   4.17      6 0.0268
##   8 0.0000000001    1    rmse    standard   5.22      6 0.0243
##   9 0.0000000001    1    rsq     standard   0.530     6 0.00193
##  10 1               1    mae     standard   4.32      6 0.0286
##  11 1               1    rmse    standard   5.42      6 0.0289
##  12 1               1    rsq     standard   0.526     6 0.00280
##     .config
##     <chr>
##   1 Preprocessor1_Model1
##   2 Preprocessor1_Model1
##   3 Preprocessor1_Model1
##   4 Preprocessor1_Model2
```

```
##   5 Preprocessor1_Model2
##   6 Preprocessor1_Model2
##   7 Preprocessor1_Model3
##   8 Preprocessor1_Model3
##   9 Preprocessor1_Model3
##  10 Preprocessor1_Model4
##  11 Preprocessor1_Model4
##  12 Preprocessor1_Model4
```

```
### show best
## call function
show_best(
  # results
  glmnet_reg_tune,
  # metric
  metric = "rmse"
)
```

```
## # A tibble: 4 x 8
##        penalty mixture .metric .estimator  mean     n std_err .config
##          <dbl>   <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 0.0000000001    1    rmse    standard    5.22     6  0.0243 Preprocessor1_Mod~
## 2 0.0000000001    0.05 rmse    standard    5.22     6  0.0243 Preprocessor1_Mod~
## 3 1               0.05 rmse    standard    5.23     6  0.0251 Preprocessor1_Mod~
## 4 1               1    rmse    standard    5.42     6  0.0289 Preprocessor1_Mod~
```

```
### create final workflow
## save as object
glmnet_reg_wflow_final <-
  ## initial workflow
  glmnet_reg_wflow %>%
  ## finalize workflow
  finalize_workflow(
    # tibble
    tibble(
      # penalty
      penalty = 1e-10,
      # mixture
      mixture = 0.05
    )
  )



### fit to complete training data
## save as object
glmnet_reg_fit <-
  ## workflow
  glmnet_reg_wflow_final %>%
  ## fit
  fit(staff_train)
```

```r
### view coefficients
## call model object
glmnet_reg_fit %>%
  ## pull fit
  pull_workflow_fit() %>%
  ## coefficients
  tidy()
```

```
## # A tibble: 15 x 3
##    term        estimate      penalty
##    <chr>          <dbl>        <dbl>
##  1 (Intercept)  44.8    0.0000000001
##  2 proactive     1.14   0.0000000001
##  3 emot_intel    1.65   0.0000000001
##  4 sjt           1.58   0.0000000001
##  5 work_samp     0.0140 0.0000000001
##  6 str_int       0.0313 0.0000000001
##  7 consc         1.51   0.0000000001
##  8 cog_flex      1.64   0.0000000001
##  9 work_exp_1    0.990  0.0000000001
## 10 work_exp_2    0.308  0.0000000001
## 11 work_exp_3   -0.0413 0.0000000001
## 12 degree_1      0.487  0.0000000001
## 13 degree_2     -0.0649 0.0000000001
## 14 degree_3     -0.424  0.0000000001
## 15 degree_4     -0.221  0.0000000001
```

Task 7.2

Create a *random forest* model specification named **rf_reg_spec**. Use the **rand_forest()** specification and set the **mode** to **regression**. Use the **ranger** engine.

Create a *random forest* model workflow named **rf_reg_wflow** using **workflow()**. Use **add_model()** to add a model using **rf_reg_spec**. Use **add_recipe()** to add **staff_rec**.

Create an object named **rf_reg_folds** to save fitted models to folds using **rf_reg_wflow**. In **fit_resamples()**, set **resamples** to **staff_train_folds** and **metrics** to **reg_met**.

Apply **collect_metrics()** to **rf_reg_folds**.

Create an object named **rf_reg_fit** to save a fitted model to the complete *training* data using **rf_reg_wflow**. Use **update_model()** to update the model specification to the set **importance** parameter to **impurity**. In **fit()**, specify **staff_train**.

Use **pull_workflow_fit()** and **vip()** on **rf_reg_fit** to view the importance values of predictors.

Questions 7.2: Answer these questions: (1) What is the *average mean absolute error* across the folds? (2) Which *predictor* is *most important*?

Responses 7.2: *(1) 4.19 (2) sjt.*

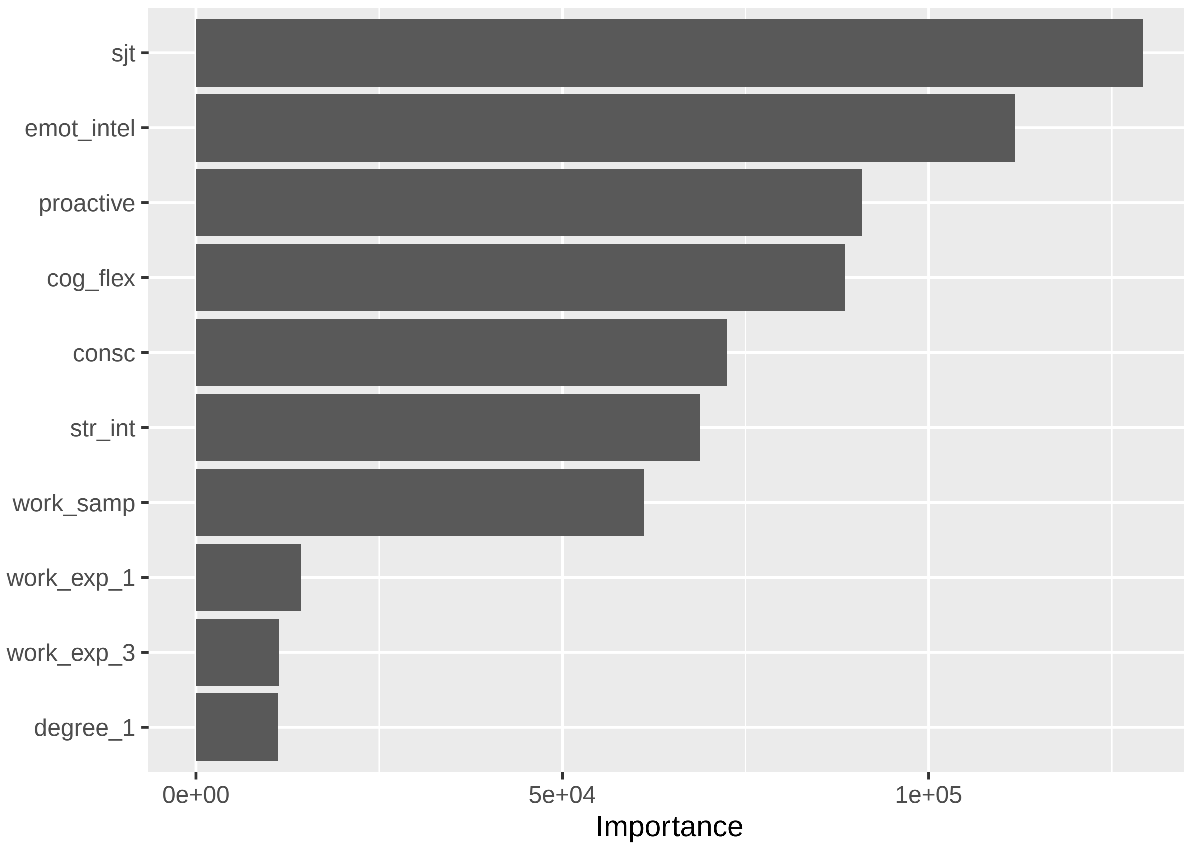
```r
#### random forest
### model specification
## save as object
rf_reg_spec <-
```

```
  ## rf specification
  rand_forest(
    # regression
    mode = "regression"
  ) %>%
  ## specify engine
  set_engine("ranger")



### create initial workflow
## save as object
rf_reg_wflow <- workflow() %>%
  ## add model
  add_model(rf_reg_spec) %>%
  ## add recipe
  add_recipe(
    # previous recipe
    staff_rec
  )




### estimate models
## save as object
rf_reg_folds <-
  ## workflow
  rf_reg_wflow %>%
  ## fit
  fit_resamples(
    # folds
    resamples = staff_train_folds,
    # metrics
    metrics = reg_met
  )



### show metrics
## call function
collect_metrics(rf_reg_folds)
```

```
## # A tibble: 3 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 mae     standard   4.18      6 0.0251  Preprocessor1_Model1
## 2 rmse    standard   5.24      6 0.0205  Preprocessor1_Model1
## 3 rsq     standard   0.527     6 0.00250 Preprocessor1_Model1
```

```
##fit to complete training data
rf_reg_fit <-
  rf_reg_wflow %>%
```

```
  update_model(
    rand_forest(
      mode = "regression"
    ) %>%
      set_engine(
        "ranger",
        importance = "impurity"
      )
  ) %>%
  fit(staff_train)



##view coefficients
rf_reg_fit %>%
  pull_workflow_fit() %>%
  vip()
```



Task 7.3

Create a *neural network* model specification named **nn_reg_spec**. Use the **mlp()** specification and set the **mode** to **regression**, **hidden_units** to **30**, and **epochs** to **100**. Use the **nnet** engine.

Create a *neural network* model workflow named **nn_reg_wflow** using **workflow()**. Use **add_model()** to add a model using **nn_reg_spec**. Use **add_recipe()** to add **staff_rec**.

Create an object named **nn_reg_folds** to save fitted models to folds using **nn_reg_wflow**. In **fit_resamples()**, set **resamples** to **staff_train_folds** and **metrics** to **reg_met**.

Apply **collect_metrics()** to **nn_reg_folds**.

Create an object named **nn_reg_fit** to save a fitted model to the complete *training* data using **nn_reg_wflow**. In **fit()**, specify **staff_train**.

Use **pull_workflow_fit()** on **nn_reg_fit** to view the importance values of predictors.

Questions 7.3: Answer these questions: (1) What is the *average root mean squared error* across the folds? (2) How many *nodes* are in the *input* layer of the neural network? (3) How many *weights* are in the neural network?

Responses 7.3: *(1) 4.22 (2) 14 nodes (3) 481 weights.*

```
#### neural network
### model specification
## save as object
nn_reg_spec <-
  ## nn specification
  mlp(
    # regression
    mode = "regression",
    # number of hidden units
    hidden_units = 30,
    # epochs
    epochs = 100
  ) %>%
  ## specify engine
  set_engine("nnet")




### create initial workflow
## save as object
nn_reg_wflow <- workflow() %>%
  ## add model
  add_model(nn_reg_spec) %>%
  ## add recipe
  add_recipe(
    # previous recipe
    staff_rec)



### estimate models
## save as object
nn_reg_folds <-
  ## workflow
  nn_reg_wflow %>%
  ## fit
  fit_resamples(
    # folds
    resamples = staff_train_folds,
    # metrics
```

```
    metrics = reg_met
  )


### show metrics
## call function
collect_metrics(nn_reg_folds)
```

```
## # A tibble: 3 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 mae     standard   4.21      6 0.0271  Preprocessor1_Model1
## 2 rmse    standard   5.29      6 0.0227  Preprocessor1_Model1
## 3 rsq     standard   0.518     6 0.00215 Preprocessor1_Model1
```

```
### fit to complete training data
## save as object
nn_reg_fit <-
  ## workflow
  nn_reg_wflow %>%
  ## fit
  fit(staff_train)



### view coefficients
## call model object
nn_reg_fit %>%
  ## pull fit
  pull_workflow_fit()
```

```
## parsnip model object
##
## Fit time:  7.9s
## a 14-30-1 network with 481 weights
## inputs: proactive emot_intel sjt work_samp str_int consc cog_flex work_exp_1 work_exp_2 work_exp_3 de
## output(s): ..y
## options were - linear output units
```

Task 8: Evaluate Continuous Outcome Models

For this task, you will evaluate the *citizenship* models on the testing data.

Task 8.1

Create an object named **glmnet_reg_pred**. Apply **predict()** to **glmnet_reg_fit** and **staff_test**. Rename the **.pred** column to **glmnet_reg_pred**.

Create an object named **rf_reg_pred**. Apply **predict()** to **rf_reg_fit** and **staff_test**. Rename the **.pred** column to **rf_reg_pred**.

Create an object named **nn_reg_pred**. Apply **predict()** to **nn_reg_fit** and **staff_test**. Rename the **.pred** column to **nn_reg_pred**.

Create an object named **staff_test_reg**. Use **select()** on **staff_test** to choose **citizenship**. Then, bind columns with **glmnet_reg_pred**, **rf_reg_pred**, and **nn_reg_pred**.

Print a table of metrics on the models. Use **map_dfr()** and set the *data* input by *removing* **citizenship** from **staff_test_reg**. Then, call **reg_met()** as the function input to **map_dfr()**. Inside of **reg_met()**, set the **data** to **staff_reg_test**, **truth** to **citizenship**, and **estimate** to **.x**. Set the **.id** to **model**. Use **pivot_wider()** to pivot the data table wide by setting **id_cols** to **model**, **names_from** to **.metric**, and **values_from** to **.estimate**.

Questions 8.1: Answer these questions: (1) What is *mean absolute error* of the *random forest* model? (2) Which model has the *lowest root mean squared error*?

Responses 8.1: *(1) 4.22 (2) glmnet_reg_pred had the lowest rmse.*

```
### elastic net
## save as object
glmnet_reg_pred <- predict(
  # fitted model
  glmnet_reg_fit,
  # test data
  new_data = staff_test
) %>%
  ## rename
  rename(glmnet_reg_pred = .pred)



### random forest
## save as object
rf_reg_pred <- predict(
  # fitted model
  rf_reg_fit,
  # test data
  new_data = staff_test
) %>%
  ## rename
  rename(rf_reg_pred = .pred)




### neural network
## save as object
nn_reg_pred <- predict(
  # fitted model
  nn_reg_fit,
  # test data
  new_data = staff_test
) %>%
  ## rename
  rename(nn_reg_pred = .pred)
```

```r
#### combine tibbles
### observed and predicted values
## save as object
staff_test_reg <- staff_test %>%
  ## select observed values
  select(citizenship) %>%
  ## bind columns
  bind_cols(
    # elastic net
    glmnet_reg_pred,
    # random forest
    rf_reg_pred,
    # neural network
    nn_reg_pred
  )
```

```r
#### compute metrics
### performance on testing data
## map
map_dfr(
  # data
  staff_test_reg %>%
    # remove observed values
    select(-citizenship),
  # function
  ~ reg_met(
      # data
      data = staff_test_reg,
      # observed
      truth = citizenship,
      # predicted
      estimate = .x
  ),
  # model
  .id = "model"
) %>%
  ## pivot wide
  pivot_wider(
    # model
    id_cols = model,
    # names
    names_from = .metric,
    # values
    values_from = .estimate
  )
```

```
## # A tibble: 3 x 4
##   model             mae  rmse   rsq
##   <chr>           <dbl> <dbl> <dbl>
## 1 glmnet_reg_pred  4.20  5.25 0.534
```

```
## 2 rf_reg_pred      4.23  5.29 0.528
## 3 nn_reg_pred      4.23  5.30 0.525
```

Task 8.2

Create a long table named **staff_test_reg_long** from **staff_test_reg** by applying **pivot_longer()**. Set the **cols** to **glmnet_reg_pred:nn_reg_pred**, **names_to** to **model**, and **values_to** to **pred**. Convert **model** to a *factor* variable.

Create a plot named **reg_plot** using **ggplot()** and **staff_test_reg_long**. Set the *x-axis* to **pred** and *y-axis* to **citizenship**. Call **geom_point()** and set **alpha** to **0.5**. Call **geom_abline()** and create *red diagonal dashed* line with **size** set to **2**. Call **facet_wrap()** and facet by **model** with *two* rows and setting the labels to the full names of the models. Scale the *x-axis* and *y-axis* with *six* breaks. Label the axes to indicate the modeling of *citizenship*.

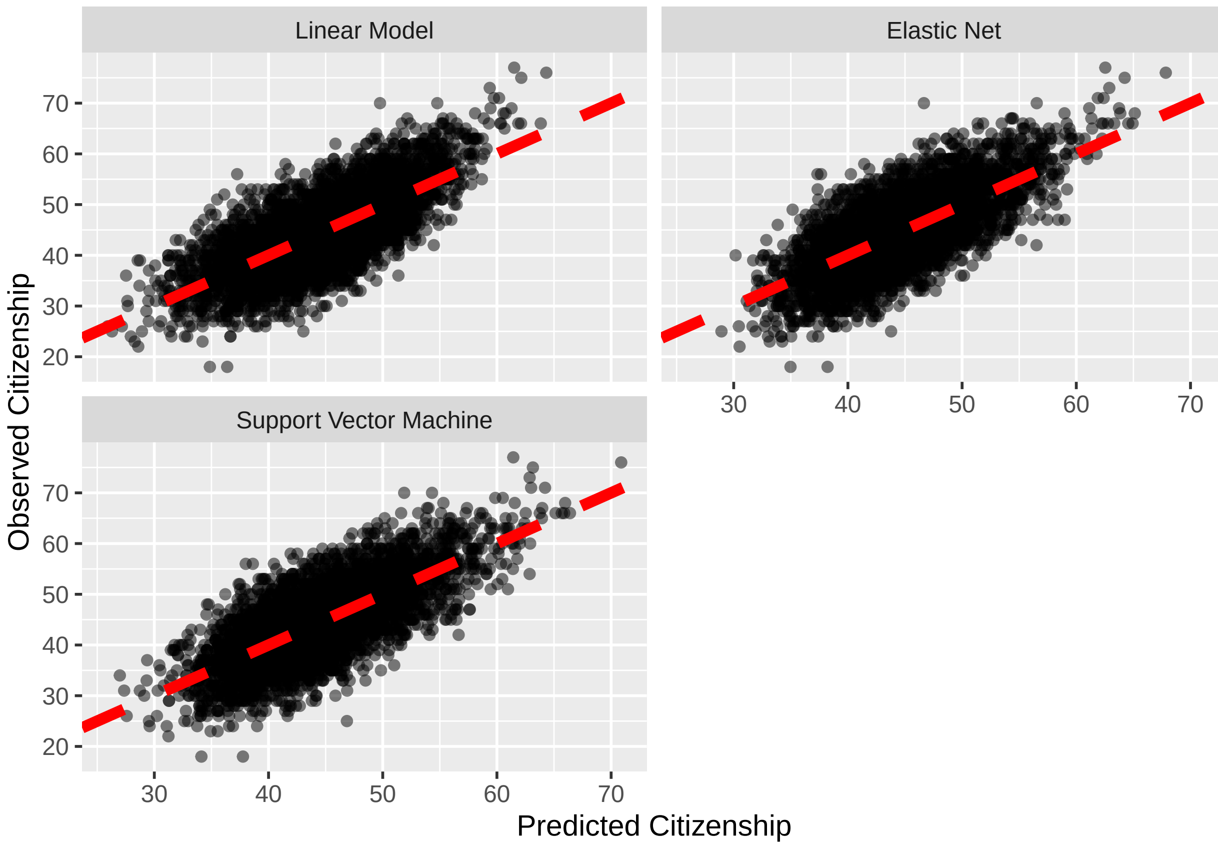
Display the plot.

Question 8.2: Does predicting *citizenship* in this data require advanced machine learning models? Explain.

Response 8.2: *No they all do an equally job of predicting job performance so only a linear model is necessary.* .

```
### make long table for plots
## save as object
staff_test_reg_long <- staff_test_reg %>%
  ## pivot long for plot
  pivot_longer(
    # columns to pivot
    cols = glmnet_reg_pred:nn_reg_pred,
    # names
    names_to = "model",
    # values
    values_to = "pred"
  ) %>%
  ## update variable
  mutate(model = as_factor(model))




#### create plot
### observed versus predicted values
## save as object
reg_plot <- ggplot(
  # data
  staff_test_reg_long,
  # mapping
  aes(
    # predicted values
    x = pred,
    # observed values
    y = citizenship
  )
) +
  ## add points
```

```
  geom_point(alpha = 0.5) +
  ## add one-to-one diagonal line
  geom_abline(lty = 2, color = "red", size = 2) +
  ## add facet
  facet_wrap(
    # variable
    vars(model),
    # number of rows
    nrow = 2,
    # labels
    labeller = as_labeller(
      # look-up table
      setNames(
        # vector elements
        c(
          "Linear Model", "Elastic Net", "Support Vector Machine",
          "Random Forest", "Neural Network"
        ),
        # names of elements
        levels(staff_test_reg_long$model)
      )
    )
  ) +
  ## scale y-axis
  scale_y_continuous(n.breaks = 6) +
  ## scale x-axis
  scale_x_continuous(n.breaks = 6) +
  ## labels
  labs(x = "Predicted Citizenship", y = "Observed Citizenship")



## display plot
reg_plot
```



Task 9: Save Object

For this task, you will save a plot.

Task 9.1

Save **reg_plot** as **ml_citizenship.png** in the **plots** folder of the project directory using **ggsave()**. Use a width of *9 inches* and height of *9 inches* for all plots.

```
##save plots to folder in project directory
ggsave(
  here("plots", "ml_citizenship.png"),
  plot =reg_plot,
  units = "in", width = 9, height = 9
  )
```

Task 10: Conceptual Question

For your last task, you will respond to a conceptual question.

Question 10.1: Describe the differences in the layers of a *neural network* machine learning model.

Response 10.1: *There are three layers of neurons in the neural network. There is the input layer which is where the data enters. Then there is the hidden layer which is where the information is processed. Lastly, there is the output later which is where the system decides what to do based on the data.*