Assignment: Predicting Employee Behaviors and Outcomes with Machine Learning

Emma Kruis

2020-01-25

Instructions

This assignment reviews the *Machine Learning* content. You will use the *machine_learning.Rmd* file I reviewed as part of the lectures for this week to complete this assignment. You will *copy and paste* relevant code from that file and update it to answer the questions in this assignment. You will respond to questions in each section after executing relevant code to answer a question. You will submit this assignment to its *Submissions* folder on *D2L*. 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*.

To start:

First, create a folder on your computer to save all relevant files for this course. If you did not do so already, you will want to create a folder named *mgt_592* that contains all of the materials for this course.

Second, inside of *mgt_592*, you will create a folder to host assignments. You can name that folder *assignments*.

Third, inside of *assignments*, you will create folders for each assignment. You can name the folder for this first assignment: *machine_learning*.

Fourth, create three additional folders in *machine_learning* named *scripts*, *data*, and *plots*. Store this script in the *scripts* folder and the data for this assignment in the *data* folder.

Fifth, go to the *File* menu in *RStudio*, select *New Project…*, choose *Existing Directory*, go to your *~/mgt_592/assignments/machine_learning* folder to select it as the top-level directory for this **R Project**.

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.

Load Packages

In this code chunk, we 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)

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

## ggthemes for plot themes
library(ggthemes)

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

## skimr to summarize data
library(skimr)

## corrr for correlation matrices
library(corrr)

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

Task 1: Import Data

We will use the same data as in the analytical lecture: **staffing.tsv**. After you load the data, then you will execute other commands on the data.

Task 1.1

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*.

Questions 1.1: Answer these questions: (1) How many *variables* are there in the data table? (2) How many *observations* are there in the data table? (3) How many *character variables* are there in the data table?

Responses 1.1: *(1) 14 variables; (2) 17,807 observations; (3) 4 character variables.*

```r
#importing data
staff_raw <- read_tsv(
  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()
## )
```

```r
#copy of data
staff_work <- staff_raw

#preview
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"~
```

```r
#removing raw copy from enviroment
rm(staff_raw)
```

Task 2: Clean Data

For this task, you will clean the data.

Task 2.1

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.

Questions 2.1: Answer these questions: (1) How many *nominal* factors are there in the data? (2) How many *ordered* factors are there in the data? (3) How many *numeric* variables are there in the data?

Responses 2.1: *(1) 3 nominal factors (2) 2 ordered factors (3) 10 numeric variables .*

```r
staff_work <-staff_work %>%
  #mutate variable types and values
  mutate(
    #characters to nominal factors
    across(
      .cols = where(is_character),
      .fns = as_factor
    ),
    #change factor labels
    degree = fct_relabel(
      degree,
      str_to_title
    ),
    #change factor labels
    degree= fct_recode(
      degree,
      "Master" = "Masters",
      "Doctor" = " Doctorate"
    ),
    #change order
    degree = fct_relevel(
      degree,
      "Associate", "Bachelor", "Master",
      after = 1
    ),
    #change order
    wrok_exp = fct_relevel(
      work_exp,
      "0-1"
    ),
    #convert to ordered factors
    across(
      .cols = c(work_exp, degree),
      .fns = factor,
      ordered = TRUE
    )
  )
```

```
## Warning: Unknown levels in `f`: Doctorate
```

```
#view data
glimpse(staff_work)
```

```
## Rows: 17,807
## Columns: 15
## $ 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,~
## $ wrok_exp       <fct> 2-5, 0-1, 0-1, 0-1, 6-10, 0-1, 0-1, 6-10, 11+, 0-1, 0-1~
```

Task 3: Examine Data

For this task, you will examine the data.

Task 3.1

Summarize **staff_work** by:

1. selecting all variables *except* for **id**, **citizenship**, and **high_potential**,
2. grouping by **promotion**, and
3. applying **skim_without_charts()**.

Questions 3.1: Answer these questions: (1) What is the *median* **sjt** *difference* between those *promoted* and *not promoted*? (2) How many *promoted* employees had **11+** years of *work experience*?

Responses 3.1: *(1)promoted:49.3,not promoted:46.7, median difference is 2.6 (2) 318 employees.*

```
##overall summary
#calling data
staff_work %>%
  select(-id, -citizenship, -high_potential) %>%
  group_by(promotion) %>%
  skim_without_charts()
```

Table 1: Data summary

Name	Piped data
Number of rows	17807
Number of columns	12
Column type frequency:	
factor	3
numeric	8
Group variables	promotion

Variable type: factor

skim_variable	promotion	n_missing	complete_rate	ordered	n_unique	top_counts
work_exp	No	0	1	TRUE	4	0-1: 6718, 2-5: 5597, 6-1: 2857, 11+: 785
work_exp	Yes	0	1	TRUE	4	0-1: 718, 2-5: 606, 6-1: 400, 11+: 126
degree	No	0	1	TRUE	5	Bac: 7363, Non: 4221, Ass: 3643, Mas: 5
degree	Yes	0	1	TRUE	5	Bac: 915, Non: 435, Ass: 392, Mas: 87
wrok_exp	No	0	1	FALSE	4	0-1: 6718, 2-5: 5597, 6-1: 2857, 11+: 785
wrok_exp	Yes	0	1	FALSE	4	0-1: 718, 2-5: 606, 6-1: 400, 11+: 126

Variable type: numeric

skim_variable	promotion	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
proactive	No	0	1	52.42	3.43	39	50	52	55	67
proactive	Yes	0	1	53.88	3.80	40	51	54	57	65
emot_intel	No	0	1	46.21	3.89	31	44	46	49	63
emot_intel	Yes	0	1	47.68	4.27	33	45	48	51	62
sjt	No	0	1	47.13	3.87	32	45	47	50	63
sjt	Yes	0	1	48.63	4.23	35	46	49	52	62
work_samp	No	0	1	45.34	3.55	31	43	45	48	60
work_samp	Yes	0	1	46.53	3.88	30	44	47	49	59
str_int	No	0	1	48.43	3.85	34	46	48	51	66
str_int	Yes	0	1	49.81	4.20	35	47	50	53	63
consc	No	0	1	51.20	4.14	33	48	51	54	67
consc	Yes	0	1	52.44	4.43	37	50	52	56	66
cog_flex	No	0	1	45.62	4.10	30	43	46	48	64
cog_flex	Yes	0	1	46.82	4.37	28	44	47	50	62
job_perf	No	0	1	44.64	7.74	16	39	44	50	80
job_perf	Yes	0	1	48.25	8.90	20	43	48	54	78

Task 3.2

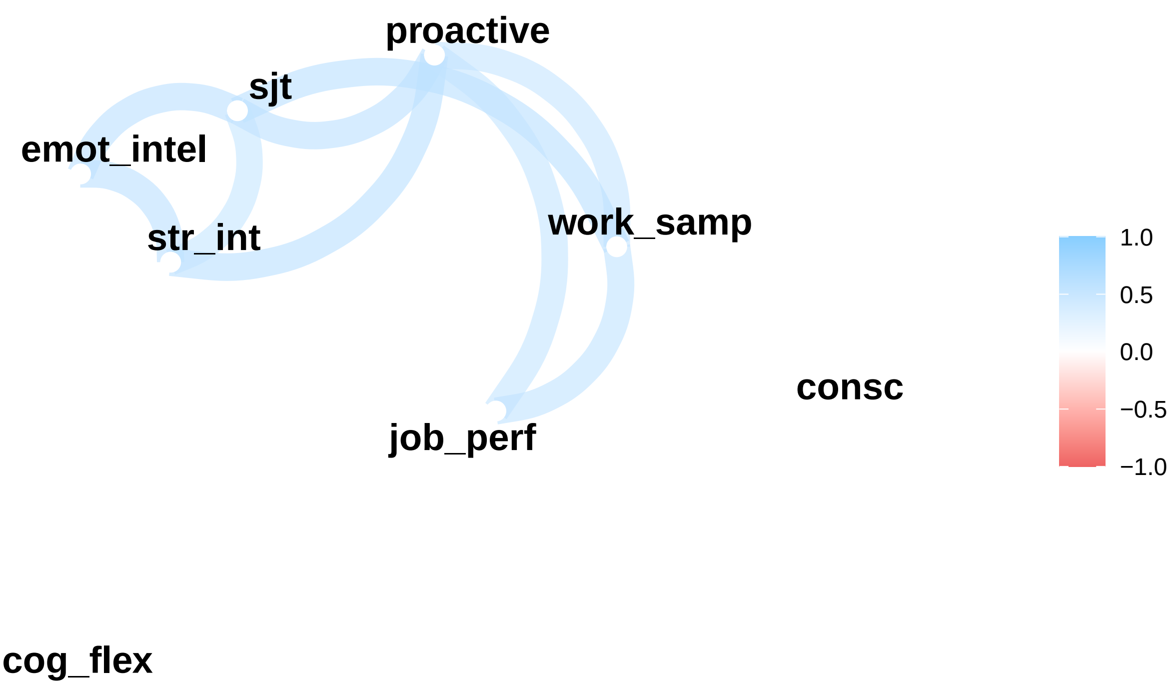
Make a *network correlation plot* from the *numeric* variables of **staff_work** *excluding* **id** and **citizenship**. Use **select()**, **correlate()**, and **network_plot()** appropriately to make the plot. The plot should consist of *8* numeric variables in total. Set the *minimum correlation* to **0.52**.

Questions 3.2: Answer these questions: (1) Which *two* variables are not correlated at least **0.52** with any of the other variables? (2) Which *two* variables are correlated at least **0.52** with *job performance*?

Responses 3.2: *(1) conc and cog_flex (2) work_samp and proactive.*

```
##examine correlations
staff_work %>%
  select(proactive, emot_intel, sjt, work_samp, str_int, consc, cog_flex, job_perf) %>%
  correlate() %>%
  network_plot(
    min_cor = 0.52
  )
```

```
##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'
```



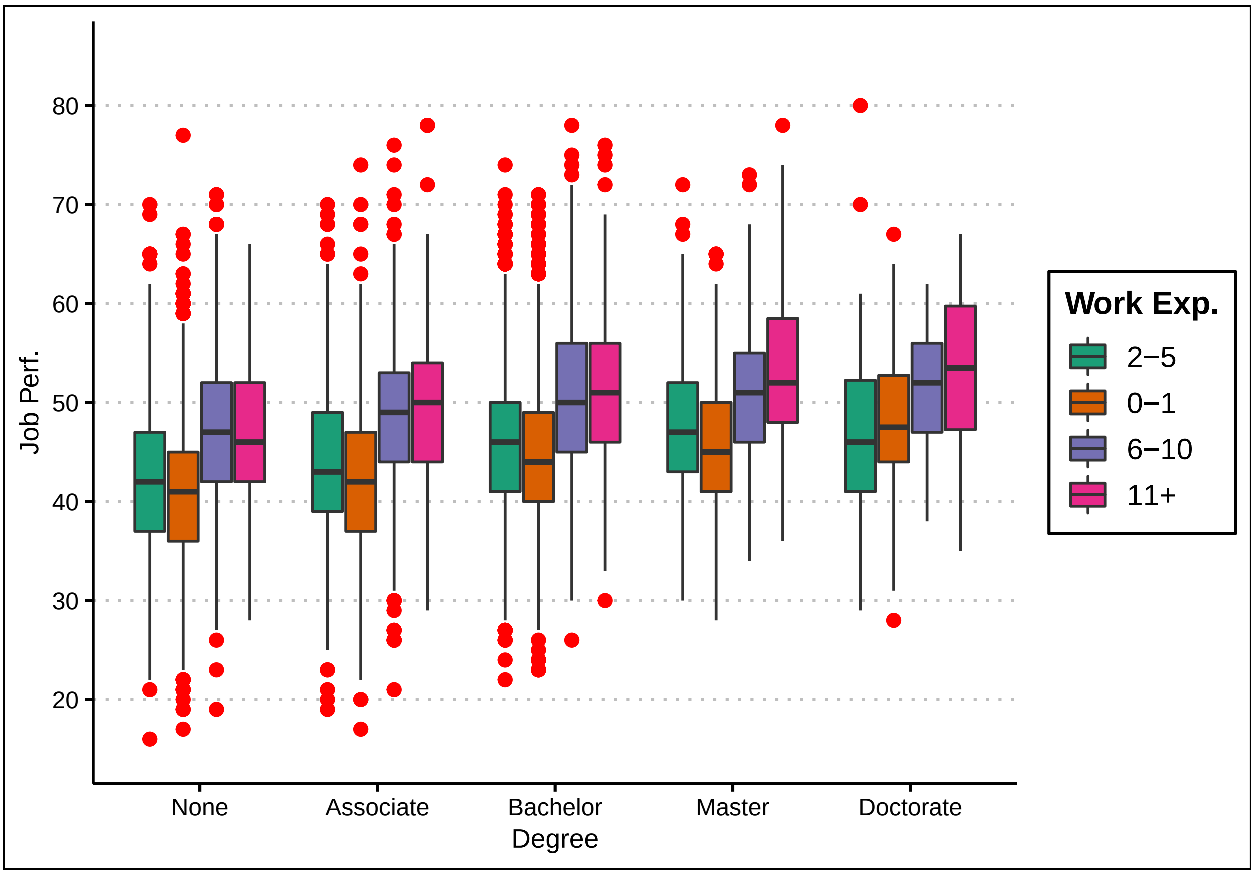
Task 3.3

Use **staff_work** and **ggplot()** to make a *boxplot* of *job performance* scores for different levels of *educational degree* and *work experience*. Place **degree** on the *x-axis* and **job_perf** on the *y-axis* and *fill* by **work_exp**. Color outliers in *red*. Scale the *y-axis* and *fill* appropriately. Use appropriate labels for the axes and legend. Use **theme_clean()**.

Questions 3.3: Answer these questions: (1) Which combination of *educational degree* and *work experience* has the highest *median*? (2) Which level of *work experience* has the *least number of outliers*? (3) Which combination of *educational degree* and *work experience* has the *highest job performance* score?

Responses 3.3: *(1) Doctorate and 11+ work experience (2) 11+ years (3) Doctorate and 11+ years has the highest job performance score .*

```
##examining categorical features
ggplot(
  staff_work,
  aes(
    x = degree,
    y = job_perf,
    fill = work_exp
  )
)+
  geom_boxplot(outlier.colour = "red", outlier.size = 2)+
  scale_y_continuous(limits = c(15, 85), n.breaks = 8)+
  scale_fill_brewer(palette = "Dark2")+
  labs(x= "Degree", y= "Job Perf.", fill= "Work Exp.")+
  theme_clean()
```



Task 3.4

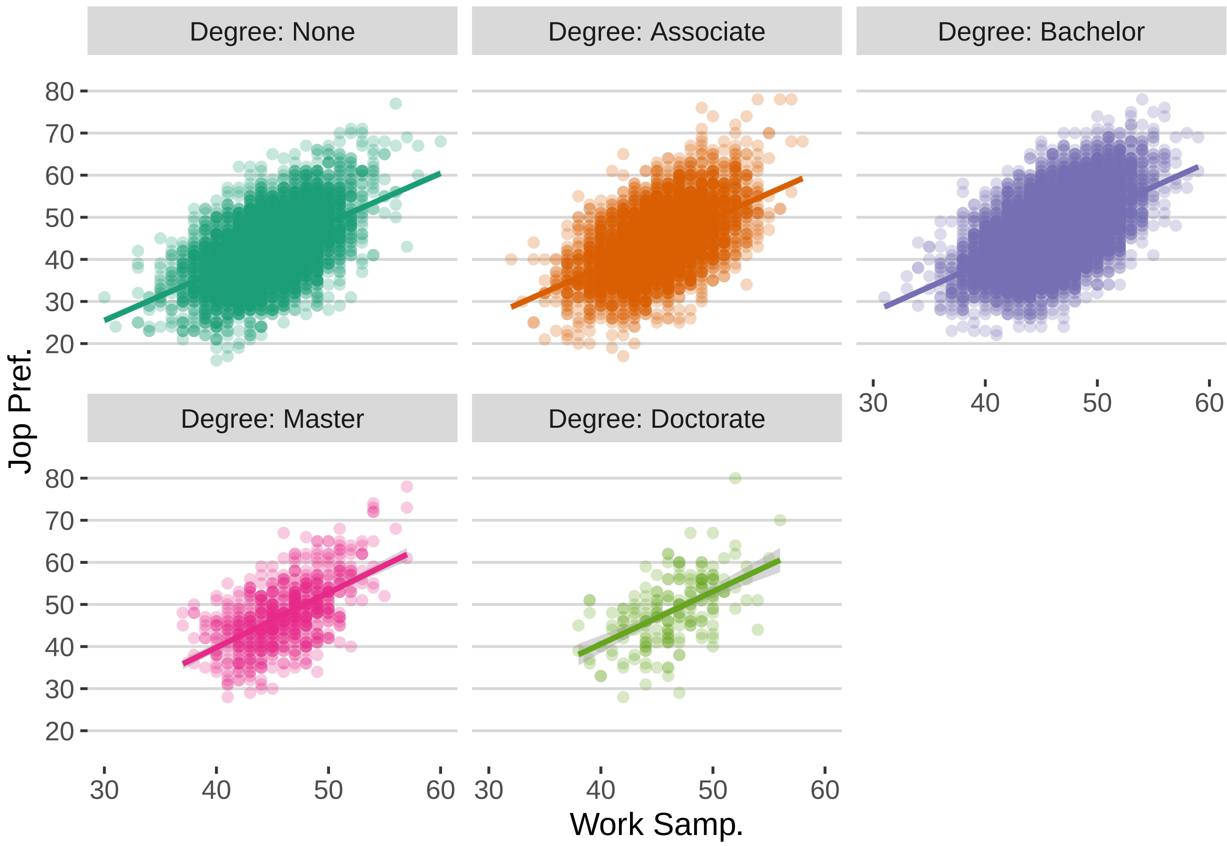
Use **staff_work** and **ggplot()** to make *faceted scatterplots* of *job performance* against *work sample* scores for different levels of *educational degree*. Place **work_samp** on the *x-axis* and **job_perf** on the *y-axis* and *fill* by **degree**. Call the *point* and *smooth* geometries with appropriate settings. Use **facet_wrap** on **degree**. Appropriately combine **as_labeller()**, **setNames()**, **paste()**, and **levels()** to correctly label the facets. For the labels, paste the word **Degree** (note the capital first letter) with the *levels* of **degree** 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 3.4: Do you see much of a *difference in the relationship* between *work sample* and *job performance* scores across the levels of *educational degree*?

Response 3.4: *No there does not appear much difference in terms of general direction however, there is more data points for the educational degrees of none, associates and bachelors suggesting that jop preformace can be predicted but not perfectly for these education degrees .*

```
##examine relations by categories
ggplot(
  staff_work,
  aes(
    x = work_samp,
    y= job_perf,
    color= degree
  )
)+
  geom_point(alpha = 0.25)+
  geom_smooth(method = "lm")+
  facet_wrap(
    vars(degree),
    nrow = 2,
    labeller = as_labeller(
      setNames(
        paste("Degree", levels(staff_work$degree), sep = ": "),
              levels(staff_work$degree)
          )
        )
      )+
  scale_y_continuous(limits = c(15, 85), n.breaks = 8) +
  scale_color_brewer(palette = "Dark2") +
  labs(x = "Work Samp.", y = "Jop Pref.", color= "Degree")+
  theme_hc() +
  theme(legend.position = "none")
```

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



Task 4: 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 4.1

Split **staff_work** into a *training* and *testing* set. Use random seed **1959**. Call **initial_split()** and create an *80%* split using **promotion** 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**.

Calculate the proportions of **promotion** in **staff_train** and **staff_test**.

Questions 4.1: Answer these questions: (1) How many observations are in the *training* set? (2) What proportion of *promoted* individuals are there in the *testing* set? (3) Are the **promotion** proportions essentially equivalent for the *training* and *testing* sets?

Responses 4.1: *(1) 13356 (2) 10.4% (3) yes.*

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

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

```r
  prob = 0.8,
  strata = promotion
)
##examining the initial split
staff_split
```

```
## <Analysis/Assess/Total>
## <13356/4451/17807>
```

```r
##extracting stratification
staff_train <- training(staff_split)

#testing
staff_test <-testing(staff_split)

##confirming stratification
#training
staff_train %>%
  count(promotion) %>%
  mutate(prop = n / sum(n))
```

```
## # A tibble: 2 x 3
##   promotion     n  prop
##   <fct>     <int> <dbl>
## 1 No        11968 0.896
## 2 Yes        1388 0.104
```

```r
#testing
staff_test %>%
  count(promotion) %>%
  mutate(prop = n / sum(n))
```

```
## # A tibble: 2 x 3
##   promotion     n  prop
##   <fct>     <int> <dbl>
## 1 No         3989 0.896
## 2 Yes         462 0.104
```

Task 4.2

Split **staff_train** into *4* total folds. Use random seed **1959**. Call **vfold_cv()** and set the *number of folds* and *number of repeats* to **2** each. Use **promotion** as the *stratification* variable. Save the split as **staff_train_folds**.

Calculate the **promotion** proportions in each *analysis* and *assessment* set from each of the folds in **staff_train_folds** using a single chained command. Start by calling **staff_train_folds** and using the correct combination of **mutate()**, **map()**, **analysis()**, and **assessment()** to create two list columns named **analysis** and **assessment**. Then, calculate the **promotion** proportions for each *analysis* and *assessment* set and saving the calculations in appropriately named columns. Then, use **unnest()** to extract the **promotion** proportions calculations and print wide.

Questions 4.2: Answer these questions: (1) How many observations are in each of the *analysis* and *assessment* sets? (2) What proportion of *promoted* individuals are there in the *first analysis* set? (3) Are the **promotion** proportions essentially equivalent for the *analysis* and *assessment* sets?

Responses 4.2: *(1) Analysis sets have 10017 observations and the Assessment sets have 3339 observations (2) 10.4% (3) yes.*

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

##split training
staff_train_folds <- vfold_cv(
  staff_train,
  v = 4,
  repeats = 2,
  strata = promotion
)

#examine folds
staff_train_folds
```

```
## #  4-fold cross-validation repeated 2 times using stratification
## # A tibble: 8 x 3
##   splits               id      id2
##   <list>               <chr>   <chr>
## 1 <split [10017/3339]> Repeat1 Fold1
## 2 <split [10017/3339]> Repeat1 Fold2
## 3 <split [10017/3339]> Repeat1 Fold3
## 4 <split [10017/3339]> Repeat1 Fold4
## 5 <split [10017/3339]> Repeat2 Fold1
## 6 <split [10017/3339]> Repeat2 Fold2
## 7 <split [10017/3339]> Repeat2 Fold3
## 8 <split [10017/3339]> Repeat2 Fold4
```

```
##extracting sets
staff_train_folds %>%
  mutate(
    #analysis set
    analysis = map(
      splits,
      analysis
    ),
    #assessment set
    assessment = map(
      splits,
      assessment
    ),
    ##computing proportions for analysis
    ana_strat = map(
      analysis,
      ~ .x %>%
        count(promotion) %>%
        mutate(prop = n /sum(n))
    ),
```

```
    ass_strat =map(
      analysis,
      ~ .x %>%
        count(promotion) %>%
        mutate(prop = n /sum(n))
    )
  )%>%
  ##unnest columns
  unnest(
    cols = c(ana_strat, ass_strat),
    names_sep = "_"
  )%>%
  print(width = Inf)
```

```
## # A tibble: 16 x 11
##    splits               id      id2   analysis
##    <list>               <chr>   <chr> <list>
##  1 <split [10017/3339]> Repeat1 Fold1 <tibble [10,017 x 15]>
##  2 <split [10017/3339]> Repeat1 Fold1 <tibble [10,017 x 15]>
##  3 <split [10017/3339]> Repeat1 Fold2 <tibble [10,017 x 15]>
##  4 <split [10017/3339]> Repeat1 Fold2 <tibble [10,017 x 15]>
##  5 <split [10017/3339]> Repeat1 Fold3 <tibble [10,017 x 15]>
##  6 <split [10017/3339]> Repeat1 Fold3 <tibble [10,017 x 15]>
##  7 <split [10017/3339]> Repeat1 Fold4 <tibble [10,017 x 15]>
##  8 <split [10017/3339]> Repeat1 Fold4 <tibble [10,017 x 15]>
##  9 <split [10017/3339]> Repeat2 Fold1 <tibble [10,017 x 15]>
## 10 <split [10017/3339]> Repeat2 Fold1 <tibble [10,017 x 15]>
## 11 <split [10017/3339]> Repeat2 Fold2 <tibble [10,017 x 15]>
## 12 <split [10017/3339]> Repeat2 Fold2 <tibble [10,017 x 15]>
## 13 <split [10017/3339]> Repeat2 Fold3 <tibble [10,017 x 15]>
## 14 <split [10017/3339]> Repeat2 Fold3 <tibble [10,017 x 15]>
## 15 <split [10017/3339]> Repeat2 Fold4 <tibble [10,017 x 15]>
## 16 <split [10017/3339]> Repeat2 Fold4 <tibble [10,017 x 15]>
##    assessment            ana_strat_promotion ana_strat_n ana_strat_prop
##    <list>                <fct>                     <int>          <dbl>
##  1 <tibble [3,339 x 15]> No                         8976          0.896
##  2 <tibble [3,339 x 15]> Yes                        1041          0.104
##  3 <tibble [3,339 x 15]> No                         8976          0.896
##  4 <tibble [3,339 x 15]> Yes                        1041          0.104
##  5 <tibble [3,339 x 15]> No                         8976          0.896
##  6 <tibble [3,339 x 15]> Yes                        1041          0.104
##  7 <tibble [3,339 x 15]> No                         8976          0.896
##  8 <tibble [3,339 x 15]> Yes                        1041          0.104
##  9 <tibble [3,339 x 15]> No                         8976          0.896
## 10 <tibble [3,339 x 15]> Yes                        1041          0.104
## 11 <tibble [3,339 x 15]> No                         8976          0.896
## 12 <tibble [3,339 x 15]> Yes                        1041          0.104
## 13 <tibble [3,339 x 15]> No                         8976          0.896
## 14 <tibble [3,339 x 15]> Yes                        1041          0.104
## 15 <tibble [3,339 x 15]> No                         8976          0.896
## 16 <tibble [3,339 x 15]> Yes                        1041          0.104
##    ass_strat_promotion ass_strat_n ass_strat_prop
##    <fct>                     <int>          <dbl>
```

```
##  1 No                           8976          0.896
##  2 Yes                          1041          0.104
##  3 No                           8976          0.896
##  4 Yes                          1041          0.104
##  5 No                           8976          0.896
##  6 Yes                          1041          0.104
##  7 No                           8976          0.896
##  8 Yes                          1041          0.104
##  9 No                           8976          0.896
## 10 Yes                          1041          0.104
## 11 No                           8976          0.896
## 12 Yes                          1041          0.104
## 13 No                           8976          0.896
## 14 Yes                          1041          0.104
## 15 No                           8976          0.896
## 16 Yes                          1041          0.104
```

Task 5: Data Preparation

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

Task 5.1

Create a recipe named **staff_rec**. Use **recipe()** on **staff_train** and specify **job_perf** and **promotion** as *outcome* variables and the remaining variables as *predictor* variables. Add a removal step to the recipe using **step_rm()** and remove **id**, **citizenship**, and **high_potential**. 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* except for nominal variable that has an *outcome* role (i.e., **promotion**).

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

Questions 5.1: Answer these questions: (1) How many *variables* are there in the *baked* recipe? (2) How many *factor* variables are there in the *baked* recipe? (3) Is the *first* employee in the *training* set *below* or *above* the mean on *cognitive flexibility* (**cog_flex**)?

Responses 5.1: *(1) There at 15 variables; 2 outcomes variables and 13 predictor variables (2) there is one factor variable which is the promotion variable (3) above.*

```
staff_rec <- recipe(
  job_perf + promotion ~
    .,
  data = staff_train
) %>%
  step_rm(
    id, citizenship, high_potential
  )%>%
  step_normalize(
    all_predictors(),
    -all_nominal()
  )%>%
  step_dummy(
    all_nominal(),
    -has_role("outcome")
```

```r
  )

##prep and bake recipe
staff_rec %>%
  prep() %>%
  bake(
    new_data = NULL
  ) %>%
  print(width =Inf)
```

```
## # A tibble: 13,356 x 19
##    proactive emot_intel    sjt work_samp str_int   consc cog_flex job_perf
##        <dbl>      <dbl>  <dbl>     <dbl>   <dbl>   <dbl>    <dbl>    <dbl>
##  1     1.84       1.17   0.943     1.82    0.621  0.638    0.548        58
##  2     0.113      0.917  1.20      0.982  -0.403 -0.0787   1.27         40
##  3     1.26      -0.598  0.434     1.54    0.365  0.638   -0.416        50
##  4     0.687     -0.0930 0.688     0.982  -1.17   0.638   -1.62         47
##  5    -0.175      0.160 -1.09     -0.131   0.365  1.12     1.27         52
##  6    -0.749      0.665  1.71     -0.409   1.39  -0.795    1.03         45
##  7     0.687     -1.61   0.943     0.147   0.877 -0.318   -0.898        37
##  8     0.113     -0.346  0.180     0.425  -0.147 -0.795    0.789        41
##  9    -0.749     -0.851  0.180    -0.131   0.109  0.399   -0.416        45
## 10     0.975      0.160  0.180     0.703   0.109  1.12     0.0658       48
##    promotion work_exp_1 work_exp_2 work_exp_3 degree_1 degree_2  degree_3
##    <fct>          <dbl>      <dbl>      <dbl>    <dbl>    <dbl>     <dbl>
##  1 No            -0.224       -0.5      0.671   -0.632    0.535 -3.16e- 1
##  2 No            -0.224       -0.5      0.671    0       -0.535 -4.10e-16
##  3 No            -0.224       -0.5      0.671   -0.316   -0.267  6.32e- 1
##  4 Yes           -0.224       -0.5      0.671   -0.316   -0.267  6.32e- 1
##  5 No             0.224       -0.5     -0.671   -0.316   -0.267  6.32e- 1
##  6 Yes           -0.224       -0.5      0.671    0.632    0.535  3.16e- 1
##  7 No            -0.224       -0.5      0.671   -0.316   -0.267  6.32e- 1
##  8 No            -0.671        0.5     -0.224   -0.632    0.535 -3.16e- 1
##  9 No             0.224       -0.5     -0.671    0       -0.535 -4.10e-16
## 10 No             0.224       -0.5     -0.671    0       -0.535 -4.10e-16
##    degree_4 wrok_exp_X2.5 wrok_exp_X6.10 wrok_exp_X11.
##       <dbl>         <dbl>          <dbl>         <dbl>
##  1    0.120             0              0             0
##  2    0.717             0              0             0
##  3   -0.478             0              0             0
##  4   -0.478             0              0             0
##  5   -0.478             0              1             0
##  6    0.120             0              0             0
##  7   -0.478             0              0             0
##  8    0.120             1              0             0
##  9    0.717             0              1             0
## 10    0.717             0              1             0
## # ... with 13,346 more rows
```

Task 6: Fit Continuous Outcome Models

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

Task 6.1

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

Create a *linear model* workflow named **lm_wflow** using **workflow()**. Use **add_model()** to add a model to the workflow with **linear_reg()** specification and **lm** engine. Use **add_recipe()** to add **staff_rec** and removing **promotion** with **step_rm()**.

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

Apply **collect_metrics()** to **lm_fit_folds**.

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

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

Questions 6.1: Answer these questions: (1) What is the *average mean absolute error* across the four folds? (2) What is the *regression coefficient* for the *cubic contrast* of *work experience* (**work_exp_3**)? (3) Interpret the *regression coefficient* for *emotional intelligence* (**emot_intel**).

Responses 6.1: *(1) 4.13 (2) -1.57 (3) for every one unit change in emotional intelligence we expect a promotion to decrease by 1.57 holding the other predictors constant. .*

```
reg_met <- metric_set(mae, rmse, rsq)

##linear model
lm_wflow <- workflow() %>%
  add_model(
    linear_reg() %>%
      set_engine("lm")
  ) %>%
  add_recipe(
    staff_rec %>%
      step_rm(promotion)
  )

##show metrics
lm_fit_folds <-
  lm_wflow %>%
  fit_resamples(
    resamples = staff_train_folds,
    metrics = reg_met
  )
```

```
## ! Fold1, Repeat1: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold2, Repeat1: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold3, Repeat1: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold4, Repeat1: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold1, Repeat2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
```

```
## ! Fold2, Repeat2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold3, Repeat2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold4, Repeat2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
```

```r
##show metrics
collect_metrics(lm_fit_folds)
```

```
## # A tibble: 3 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 mae     standard   4.14      8 0.0204  Preprocessor1_Model1
## 2 rmse    standard   5.21      8 0.0243  Preprocessor1_Model1
## 3 rsq     standard   0.570     8 0.00403 Preprocessor1_Model1
```

```r
reg_met <- metric_set(mae, rmse, rsq)
#### linear model
lm_wflow <- workflow() %>%
  add_model(
    linear_reg() %>%
      set_engine("lm")
    ) %>%
  add_recipe(
    staff_rec %>%
      step_rm(promotion)
    )

 lm_fit_folds <-
   lm_wflow %>%
   fit_resamples(
     resamples = staff_train_folds,
     metrics = reg_met
     )
```

```
## ! Fold1, Repeat1: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold2, Repeat1: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold3, Repeat1: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold4, Repeat1: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold1, Repeat2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold2, Repeat2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold3, Repeat2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold4, Repeat2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
```

```r
### show metrics
collect_metrics(lm_fit_folds)
```

```
## # A tibble: 3 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 mae     standard   4.14      8 0.0204  Preprocessor1_Model1
## 2 rmse    standard   5.21      8 0.0243  Preprocessor1_Model1
## 3 rsq     standard   0.570     8 0.00403 Preprocessor1_Model1
```

```r
## fit to complete training data
lm_fit <-
  lm_wflow %>%
  fit(staff_train)

##view coefficients
lm_fit %>%
  pull_workflow_fit() %>%
  tidy()
```

```
## # A tibble: 18 x 5
##    term           estimate std.error statistic    p.value
##    <chr>             <dbl>     <dbl>     <dbl>      <dbl>
##  1 (Intercept)     46.2       0.112    414.     0.
##  2 proactive        1.31      0.0646    20.3    5.10e- 90
##  3 emot_intel       1.57      0.0610    25.7    5.26e-142
##  4 sjt             -0.0761    0.0683    -1.11   2.65e-  1
##  5 work_samp        1.52      0.0642    23.7    6.04e-122
##  6 str_int          0.0562    0.0659     0.854  3.93e-  1
##  7 consc            1.76      0.0560    31.4    7.35e-209
##  8 cog_flex         1.71      0.0601    28.5    2.51e-173
##  9 work_exp_1       3.18      0.149     21.4    6.81e-100
## 10 work_exp_2      -0.211     0.125     -1.68   9.31e-  2
## 11 work_exp_3      -1.52      0.0993   -15.3    1.10e- 52
## 12 degree_1         0.851     0.293      2.90   3.71e-  3
## 13 degree_2         0.316     0.247      1.28   2.02e-  1
## 14 degree_3        -0.193     0.212     -0.908  3.64e-  1
## 15 degree_4        -0.0968    0.141     -0.688  4.92e-  1
## 16 wrok_exp_X2.5   NA        NA         NA     NA
## 17 wrok_exp_X6.10  NA        NA         NA     NA
## 18 wrok_exp_X11.   NA        NA         NA     NA
```

Task 6.2

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** and removing **promotion** with **step_rm()**.

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 **mae**.

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**.

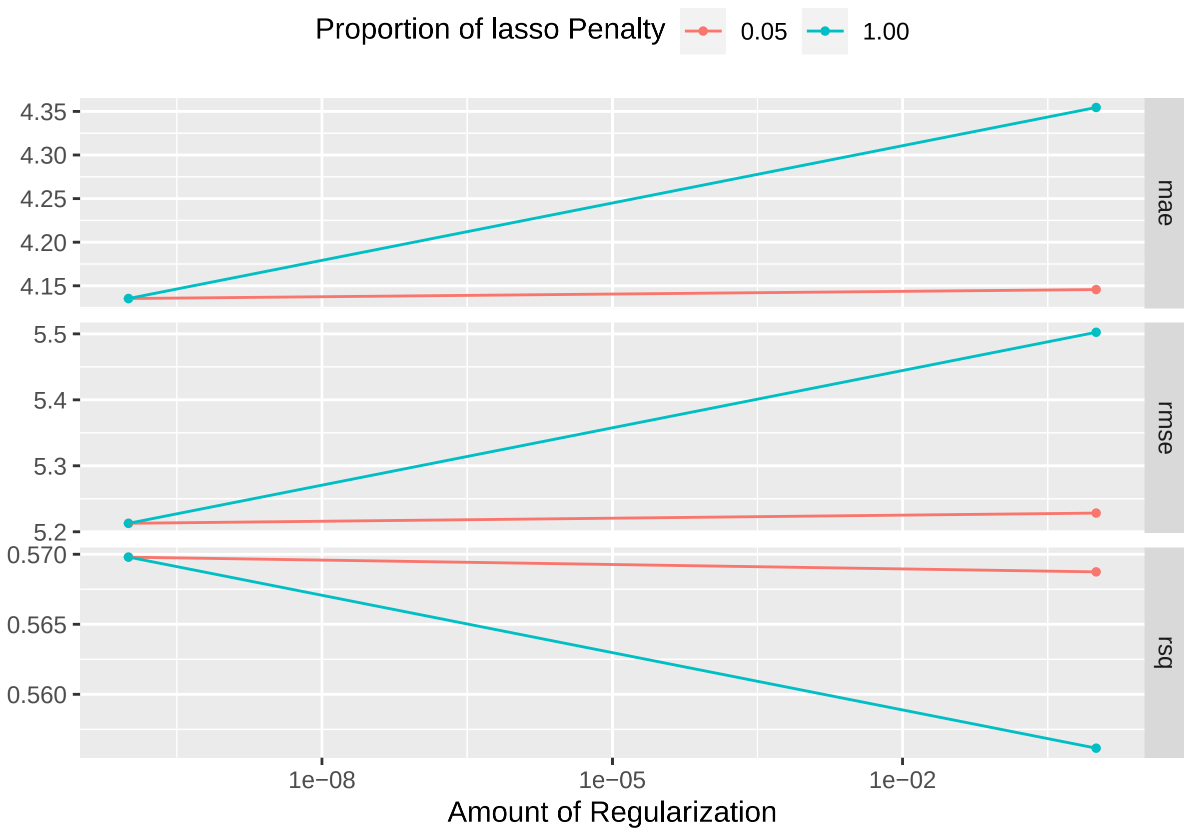
Use **pull_workflow_fit()** and **tidy()** on **glmnet_reg_fit** to view the estimated regression coefficients.

Questions 6.2: Answer these questions: (1) What is the *average root mean squared error* across the folds for the *first tuning set*? (2) What is the *value* of the *best average mean absolute error* across the folds? (3) What is the *regression coefficient* for the *quartic contrast* of *educational degree* (**degree_4**)? (4) Interpret the *regression coefficient* for *proactiveness* (**proactive**).

Responses 6.2: *(1) 4.13 (2) 4.13 (3) -0.235 (4)for every one unit change in proactiveness we expect promotion to increase by 1.29 holding the other predictors constant .*

```
##elastic net
glmnet_reg_spec <-
  linear_reg(
    penalty = tune(),
    mixture = tune()
  ) %>%
  set_engine("glmnet")

##view a tuning grid
glmnet_reg_grid <- glmnet_reg_spec %>%
  parameters() %>%
  grid_regular(levels =2)


##create initial workflow
glmnet_reg_wflow <- workflow() %>%
  add_model(glmnet_reg_spec) %>%
  add_recipe(
    staff_rec %>%
      step_rm(promotion)
  )

##esitmate models
glmnet_reg_tune <-
  glmnet_reg_wflow %>%
  tune_grid(
    staff_train_folds,
    grid = glmnet_reg_grid,
    metrics = reg_met
  )
```

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




```r
##show metrics
collect_metrics(glmnet_reg_tune) %>%
  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   4.14      8 0.0202
##  2 0.0000000001    0.05 rmse    standard   5.21      8 0.0242
##  3 0.0000000001    0.05 rsq     standard   0.570     8 0.00403
##  4 1               0.05 mae     standard   4.15      8 0.0181
##  5 1               0.05 rmse    standard   5.23      8 0.0232
##  6 1               0.05 rsq     standard   0.569     8 0.00402
##  7 0.0000000001    1    mae     standard   4.14      8 0.0203
##  8 0.0000000001    1    rmse    standard   5.21      8 0.0242
##  9 0.0000000001    1    rsq     standard   0.570     8 0.00403
## 10 1               1    mae     standard   4.35      8 0.0153
## 11 1               1    rmse    standard   5.50      8 0.0208
## 12 1               1    rsq     standard   0.556     8 0.00406
##    .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
show_best(
  glmnet_reg_tune,
  metric = "mae"
)
```

```
## # 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    mae     standard    4.14     8  0.0203 Preprocessor1_Mod~
## 2 0.0000000001    0.05 mae     standard    4.14     8  0.0202 Preprocessor1_Mod~
## 3 1               0.05 mae     standard    4.15     8  0.0181 Preprocessor1_Mod~
## 4 1               1    mae     standard    4.35     8  0.0153 Preprocessor1_Mod~
```

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


##fit to complete training data
glmnet_reg_fit <-
  glmnet_reg_wflow_final %>%
  fit(staff_train)



##view coefficients
glmnet_reg_fit %>%
  pull_workflow_fit() %>%
  tidy()
```

```
## # A tibble: 18 x 3
##    term           estimate      penalty
##    <chr>             <dbl>        <dbl>
```

```
##  1 (Intercept)     45.5    0.0000000001
##  2 proactive        1.30   0.0000000001
##  3 emot_intel       1.54   0.0000000001
##  4 sjt             -0.0398 0.0000000001
##  5 work_samp        1.51   0.0000000001
##  6 str_int          0.0738 0.0000000001
##  7 consc            1.75   0.0000000001
##  8 cog_flex         1.69   0.0000000001
##  9 work_exp_1       1.81   0.0000000001
## 10 work_exp_2       0      0.0000000001
## 11 work_exp_3      -0.707  0.0000000001
## 12 degree_1         0.772  0.0000000001
## 13 degree_2         0.251  0.0000000001
## 14 degree_3        -0.200  0.0000000001
## 15 degree_4        -0.0855 0.0000000001
## 16 wrok_exp_X2.5   -0.0877 0.0000000001
## 17 wrok_exp_X6.10   1.70   0.0000000001
## 18 wrok_exp_X11.    1.37   0.0000000001
```

Task 6.3

Create a *support vector machine* model specification named **svm_reg_spec**. Use the **svm_poly()** specification and set the **mode** to **regression**. Use the **kernlab** engine.

Create a *support vector machine* model workflow named **svm_reg_wflow** using **workflow()**. Use **add_model()** to add a model using **svm_reg_spec**. Use **add_recipe()** to add **staff_rec** and removing **promotion** with **step_rm()**.

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

Apply **collect_metrics()** to **svm_reg_folds**.

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

Use **pull_workflow_fit()** and **tidy()** on **svm_reg_fit** to view summary results.

Questions 6.3: Answer these questions: (1) What is the *average r-squared* across the folds? (2) How many *support vectors* were produced using the complete *training* data?

Responses 6.3: *(1) 0.567 (2) 11713.*

```
##support vector machine
svm_reg_spec <-
  svm_poly(
    mode = "regression"
  )%>%
  set_engine("kernlab")

##create intial workflow
svm_reg_wflow <- workflow() %>%
  add_model(svm_reg_spec) %>%
  add_recipe(
    staff_rec %>%
      step_rm(promotion)
  )
```

```r
##estimate models
svm_reg_folds <-
  svm_reg_wflow %>%
  fit_resamples(
    resamples = staff_train_folds,
    metrics = reg_met
  )

##show metrics
collect_metrics(svm_reg_folds)
```

```
## # A tibble: 3 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 mae     standard   4.14      8 0.0198  Preprocessor1_Model1
## 2 rmse    standard   5.21      8 0.0237  Preprocessor1_Model1
## 3 rsq     standard   0.570     8 0.00402 Preprocessor1_Model1
```

```r
##fit to complete_training data
svm_reg_fit <-
  svm_reg_wflow %>%
  fit(staff_train)
```

```
##  Setting default kernel parameters
```

```r
##view coefficients
svm_reg_fit %>%
  pull_workflow_fit()
```

```
## parsnip model object
##
## Fit time:  24.1s
## Support Vector Machine object of class "ksvm"
##
## SV type: eps-svr  (regression)
##  parameter : epsilon = 0.1  cost C = 1
##
## Polynomial kernel function.
##  Hyperparameters : degree =  1  scale =  1  offset =  1
##
## Number of Support Vectors : 11695
##
## Objective Function Value : -5688.654
## Training error : 0.429498
```

Task 6.4

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** and removing **promotion** with **step_rm()**.

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

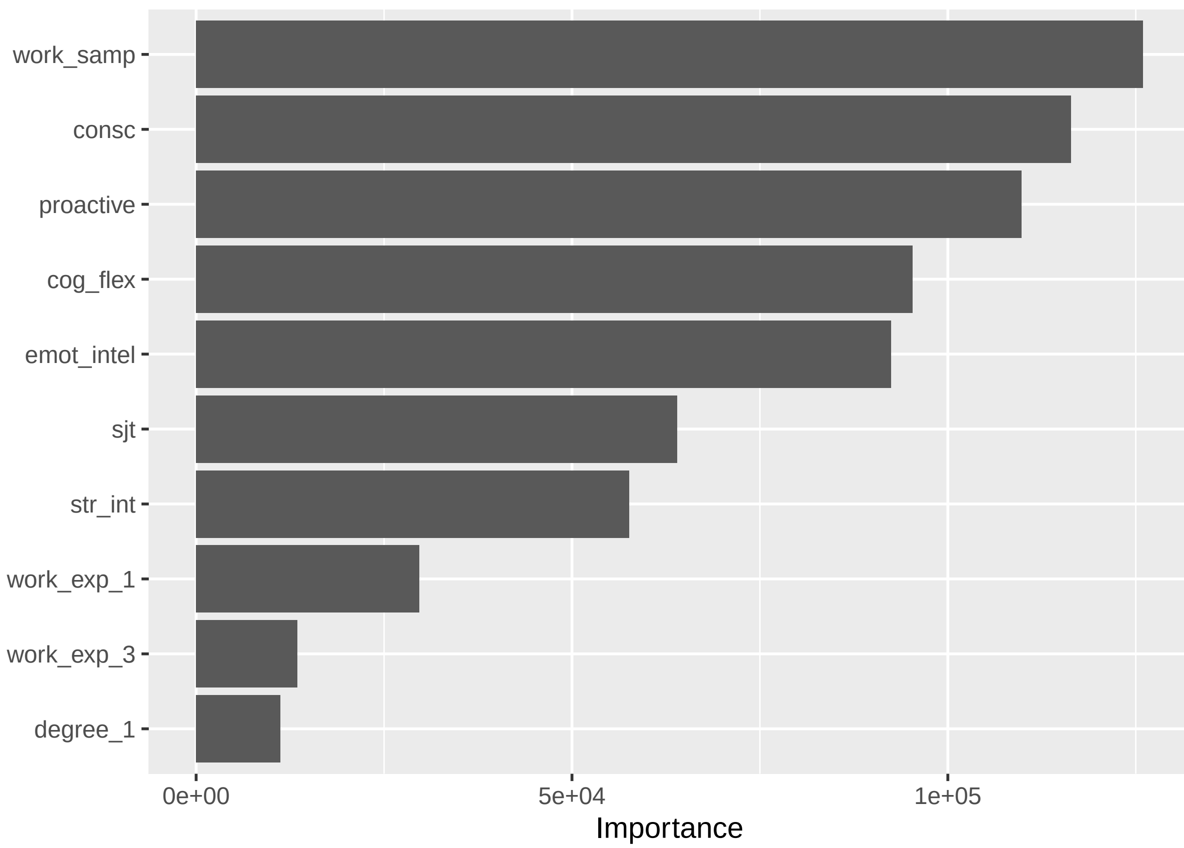
Responses 6.4: *(1) 4.16 (2) work_samp.*

```
##random forest
rf_reg_spec <-
  rand_forest(
    mode = "regression"
  ) %>%
  set_engine("ranger")

##create initial workflow
rf_reg_wflow <-workflow() %>%
  add_model(rf_reg_spec) %>%
  add_recipe(
    staff_rec %>%
      step_rm(promotion)
  )

##estimate models
rf_reg_folds  <-
  rf_reg_wflow%>%
  fit_resamples(
    resamples = staff_train_folds,
    metrics = reg_met
  )

##show metrics
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      8 0.0155  Preprocessor1_Model1
## 2 rmse    standard   5.26      8 0.0233  Preprocessor1_Model1
## 3 rsq     standard   0.562     8 0.00388 Preprocessor1_Model1
```

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

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

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



Task 6.5

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** and removing **promotion** with **step_rm()**.

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 6.5: 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 6.5: *(1) 5.30 (2) 17 (3) 571 weights.*

```
##neural network, model specifications
nn_reg_spec <- mlp(
  mode = "regression",
  hidden_units = 30,
  epochs = 100
) %>%
  set_engine("nnet")

##create initial workflow
nn_reg_wflow <- workflow() %>%
  add_model(nn_reg_spec) %>%
  add_recipe(
    staff_rec %>%
      step_rm(promotion)
  )


##estimate models
nn_reg_folds <-
  nn_reg_wflow %>%
  fit_resamples(
    resamples = staff_train_folds,
    metrics = reg_met
  )

##show metrics
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      8 0.0194  Preprocessor1_Model1
## 2 rmse    standard   5.30      8 0.0259  Preprocessor1_Model1
## 3 rsq     standard   0.556     8 0.00407 Preprocessor1_Model1
```

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


##view coefficients
```

```
nn_reg_fit %>%
  pull_workflow_fit()
```

```
## parsnip model object
##
## Fit time:  9s
## a 17-30-1 network with 571 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 7: Evaluate Continuous Outcome Models

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

Task 7.1

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

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 **svm_reg_pred**. Apply **predict()** to **svm_reg_fit** and **staff_test**. Rename the **.pred** column to **svm_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 **job_perf**. Then, bind columns with **lm_pred**, **glmnet_reg_pred**, **svm_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* **job_perf** 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 **job_perf**, 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 7.1: Answer these questions: (1) What is *mean absolute error* of the *support vector machine* model? (2) Which model has the *lowest root mean squared error*?

Responses 7.1: *(1) 4.14 (2) lm_pred, glmnet_reg_pred, and svg_reg_pred all have a rmse of 5.23 whcih is the lowest.*

```
##predictions
lm_pred <- predict(
  lm_fit,
  new_data = staff_test
) %>%
  rename(lm_pred = .pred)
```

```
## Warning in predict.lm(object = object$fit, newdata = new_data, type =
## "response"): prediction from a rank-deficient fit may be misleading
```

```r
##elastic net
glmnet_reg_pred <- predict(
  glmnet_reg_fit,
  new_data = staff_test
) %>%
  rename(glmnet_reg_pred = .pred)

##support vector machine
svm_reg_pred <- predict(
  svm_reg_fit,
  new_data =staff_test
) %>%
  rename(svm_reg_pred = .pred)

##random forest
rf_reg_pred <- predict(
  rf_reg_fit,
  new_data = staff_test
) %>%
  rename(rf_reg_pred = .pred)

##neural network
nn_reg_pred <- predict(
  nn_reg_fit,
  new_data = staff_test,
) %>%
  rename(nn_reg_pred = .pred)


##combine tibbles
staff_test_reg <- staff_test %>%
  select(job_perf) %>%
  bind_cols(
    lm_pred,
    glmnet_reg_pred,
    svm_reg_pred,
    rf_reg_pred,
    nn_reg_pred
  )

##compute metrics
map_dfr(
  staff_test_reg %>%
    select( -job_perf),
  ~ reg_met(
    data = staff_test_reg,
    truth = job_perf,
    estimate = .x
  ),
  .id = "model"
) %>%
  pivot_wider(
    id_cols = model,
```

```
    names_from = .metric,
    values_from = .estimate
  )
```

```
## # A tibble: 5 x 4
##   model            mae  rmse   rsq
##   <chr>          <dbl> <dbl> <dbl>
## 1 lm_pred         4.13  5.18 0.572
## 2 glmnet_reg_pred 4.13  5.19 0.572
## 3 svm_reg_pred    4.13  5.19 0.572
## 4 rf_reg_pred     4.13  5.20 0.570
## 5 nn_reg_pred     4.12  5.19 0.573
```

Task 7.2

Create a long table named **staff_test_reg_long** from **staff_test_reg** by applying **pivot_longer()**. Set the **cols** to **lm_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 **job_perf**. 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 *job performance*.

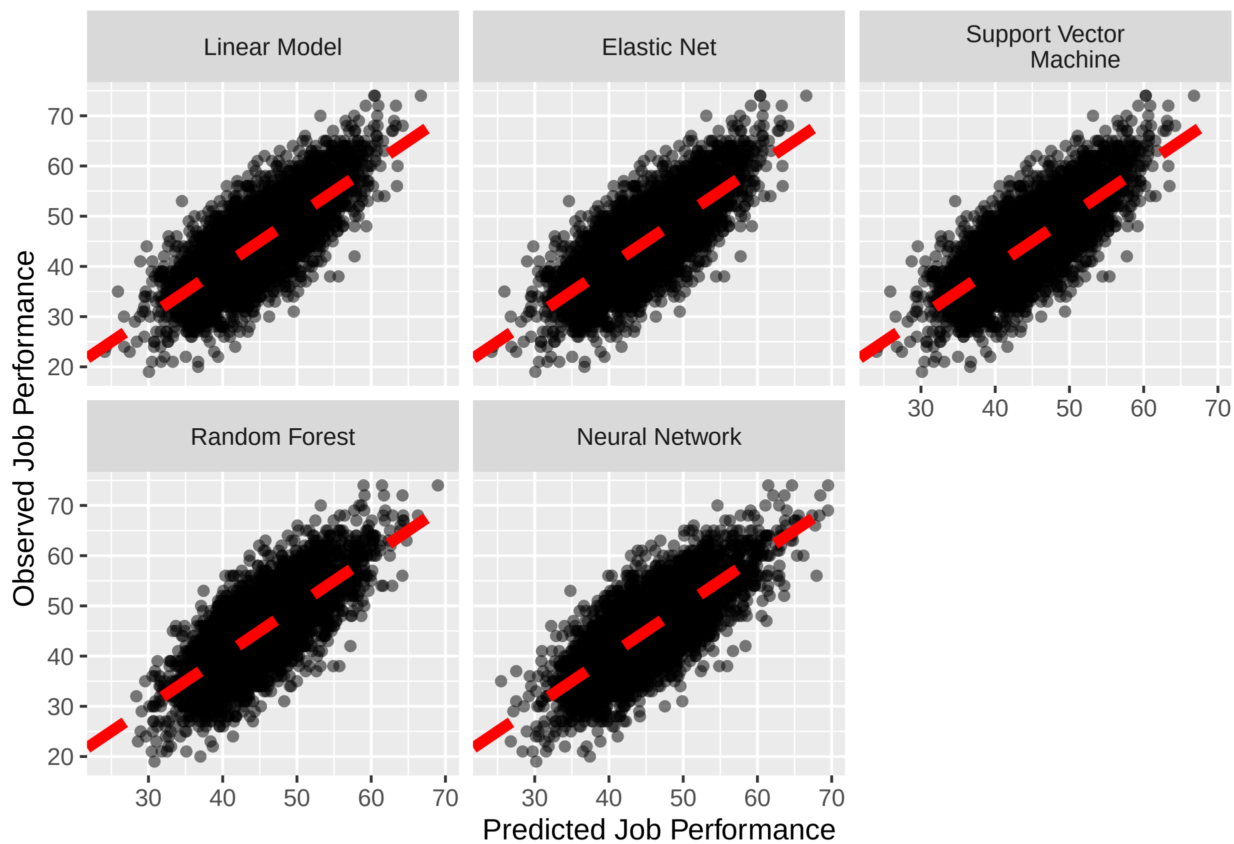
Display the plot.

Question 7.2: Does predicting *job performance* in this data require advanced machine learning models? Explain.

Response 7.2: *No they all do an equally job of predicting job performance so only a linear model is necessary. .*

```
##long table for plots
staff_test_reg_long <- staff_test_reg %>%
  pivot_longer(
    cols = lm_pred:nn_reg_pred,
    names_to = "model",
    values_to = "pred"
  ) %>%
  mutate(model = as_factor(model))

##observed versus predicted values
reg_plot <- ggplot(
  staff_test_reg_long,
  aes(
    x = pred,
    y = job_perf
  )
) +
  geom_point(alpha = 0.5) +
  geom_abline(lty = 2, color = "red", size = 2) +
  facet_wrap(
    vars(model),
```

```r
    nrow = 2,
    labeller = as_labeller(
      setNames(
        c(
        "Linear Model", "Elastic Net", "Support Vector
         Machine", "Random Forest", "Neural Network"
         ),
      levels(staff_test_reg_long$model)
    )
  )
) +
  scale_y_continuous(n.breaks = 6) +
  scale_x_continuous(n.breaks = 6) +
  labs(x = "Predicted Job Performance", y = "Observed Job Performance")

##display plot
reg_plot
```



Task 8: Fit Categorical Outcome Models

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

Task 8.1

Create a *metric set* of *area under the receiver-operator characteristic curve*, *Matthews correlation coefficient*, and *accuracy* named **class_met**.

Create a *linear model* workflow named **glm_wflow** using **workflow()**. Use **add_model()** to add a model to the workflow with **logistic_reg()** specification and **glm** engine. Use **add_recipe()** to add **staff_rec** and removing **job_perf** with **step_rm()**.

Create an object named **glm_fit_folds** to save fitted models to folds using **glm_wflow**. In **fit_resamples()**, set **resamples** to **staff_train_folds** and **metrics** to **class_met**.

Apply **collect_metrics()** to **glm_fit_folds**.

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

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

Questions 8.1: Answer these questions: (1) What is the *average accuracy* across the four folds? (2) What is the *regression coefficient* for the *quadratic contrast* of *work experience* (**work_exp_2**)? (3) Interpret the *regression coefficient* for *emotional intelligence* (**emot_intel**).

Responses 8.1: *(1) 0.896 (2) -0.00421 (3) For every one unit change in emotinal intelligence we expect promotion to increase by 0.161 holding the other predictors constant..*

```
##specify model metric to optimize
class_met <-metric_set(roc_auc, mcc, accuracy)

##logistic regression
glm_wflow <- workflow() %>%
  add_model(
    logistic_reg() %>%
      set_engine("glm")
  ) %>%
  add_recipe(
    staff_rec %>%
      step_rm(job_perf)
  )


##estimate model on folds
glm_fit_folds <-
  glm_wflow %>%
  fit_resamples(
    resamples = staff_train_folds,
    metrics = class_met
  )
```

```
## ! Fold1, Repeat1: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold2, Repeat1: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold3, Repeat1: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold4, Repeat1: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
```

```
## ! Fold1, Repeat2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold2, Repeat2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold3, Repeat2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...

## ! Fold4, Repeat2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
```

```
##show metrics
collect_metrics(glm_fit_folds)
```

```
## # A tibble: 3 x 6
##   .metric  .estimator     mean     n   std_err .config
##   <chr>    <chr>         <dbl> <int>     <dbl> <chr>
## 1 accuracy binary      0.896       8 0.0000566 Preprocessor1_Model1
## 2 mcc      binary     -0.00589     4 0         Preprocessor1_Model1
## 3 roc_auc  binary      0.626       8 0.00298   Preprocessor1_Model1
```

```
##fit to complete training data
glm_fit <-
  glm_wflow %>%
  fit(staff_train)

##view coefficients
glm_fit %>%
  pull_workflow_fit() %>%
  tidy()
```

```
## # A tibble: 18 x 5
##    term           estimate std.error statistic    p.value
##    <chr>             <dbl>     <dbl>     <dbl>      <dbl>
##  1 (Intercept)    -2.21       0.0679  -32.6     2.84e-233
##  2 proactive       0.212      0.0412    5.15    2.66e-  7
##  3 emot_intel      0.138      0.0390    3.55    3.91e-  4
##  4 sjt             0.117      0.0436    2.68    7.32e-  3
##  5 work_samp      -0.00829    0.0410   -0.202   8.40e-  1
##  6 str_int         0.00272    0.0419    0.0649  9.48e-  1
##  7 consc           0.129      0.0356    3.63    2.82e-  4
##  8 cog_flex        0.0921     0.0374    2.46    1.38e-  2
##  9 work_exp_1      0.132      0.0872    1.51    1.30e-  1
## 10 work_exp_2     -0.0487     0.0743   -0.656   5.12e-  1
## 11 work_exp_3     -0.0159     0.0601   -0.264   7.92e-  1
## 12 degree_1       -0.0248     0.176    -0.141   8.88e-  1
## 13 degree_2        0.0231     0.148     0.155   8.76e-  1
## 14 degree_3       -0.0159     0.128    -0.124   9.01e-  1
## 15 degree_4       -0.0419     0.0855   -0.490   6.24e-  1
## 16 wrok_exp_X2.5  NA         NA        NA      NA
## 17 wrok_exp_X6.10 NA         NA        NA      NA
## 18 wrok_exp_X11.  NA         NA        NA      NA
```

Task 8.2

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

Create a *tuning grid* named **glmnnet_class_grid**. Specify the *tuning grid* using **glmnet_reg_spec**, **parameters()**, and **grid_max_entropy()** with **size** set to **5**.

Create an *elastic net* model workflow named **glmnet_class_wflow** using **workflow()**. Use **add_model()** to add a model using **glmnet_class_spec**. Use **add_recipe()** to add **staff_rec** and removing **job_perf** with **step_rm()**.

Create an object named **glmnet_class_tune** to save fitted models to folds using **glmnet_class_wflow** and the *tuning grid*. In **tune_grid()**, set the *folds* to **staff_train_folds**, **grid** to **glmnet_class_grid**, and **metrics** to **class_met**.

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

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

Apply **show_best()** to **glmnet_class_tune** and set the **metric** to **roc_auc**.

Create a *final workflow* named **glmnet_class_wflow_final** using **glmnet_class_wflow** and **finalize_workflow()**. Inside of **finalize_workflow()**, create a **tibble()** and set **penalty** and **mixture** to values based on tuning.

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

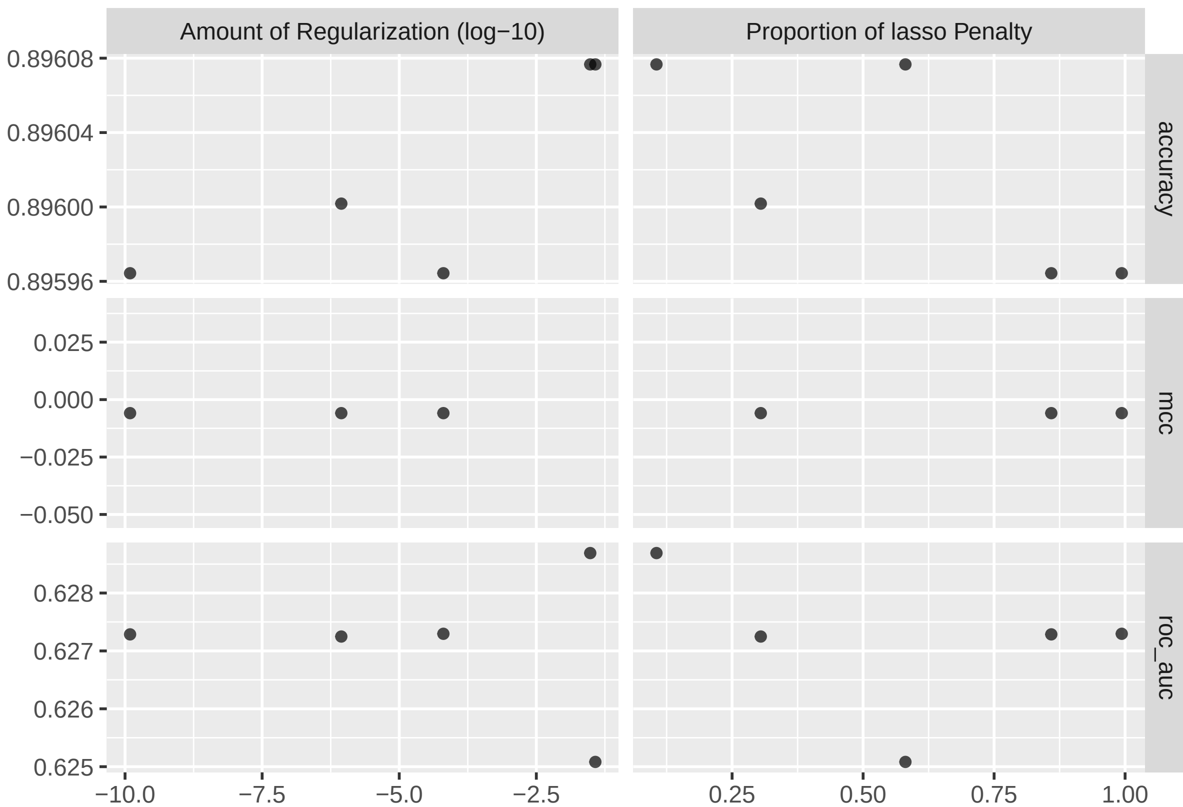
Use **pull_workflow_fit()** and **tidy()** on **glmnet_class_fit** to view the estimated regression coefficients.

Questions 8.2: Answer these questions: (1) What is the *average* **roc_auc** across the folds for the *third tuning set*? (2) What is the *value* of the *best average* **roc_auc** across the folds? (3) What is the *regression coefficient* for the *quadratic contrast* of *educational degree* (**degree_2**)? (4) Interpret the *regression coefficient* for *proactiveness* (**proactive**).

Responses 8.2: *(1) 0.622 (2) 0.622 (3) 0 (4) For every one unit change in proactiveness we expect promotion to increase by 0.0989 holding the other predictors constant .*

```
##elastic net
glmnet_class_spec <-
  logistic_reg(
    penalty = tune(),
    mixture = tune()
  ) %>%
  set_engine("glmnet")

##view a tuning grid
glmnet_class_grid <- glmnet_class_spec %>%
  parameters() %>%
  grid_max_entropy(size = 5)

##create initial work flow
glmnet_class_wflow <- workflow() %>%
  add_model(glmnet_class_spec) %>%
  add_recipe(
    staff_rec %>%
      step_rm(job_perf)
  )
```

```r
##estimate models
glmnet_class_tune <-
  glmnet_class_wflow %>%
  tune_grid(
    staff_train_folds,
    grid = glmnet_class_grid,
    metrics = class_met
  )

##plot metrics
autoplot(glmnet_class_tune)
```




```r
##show metrics
collect_metrics(glmnet_class_tune) %>%
  print(n = Inf, width = Inf)
```

```
## # A tibble: 15 x 8
##     penalty mixture .metric  .estimator     mean     n    std_err
##       <dbl>   <dbl> <chr>    <chr>         <dbl> <int>      <dbl>
##   1 3.04e- 2   0.106 accuracy binary       0.896      8  0
##   2 3.04e- 2   0.106 mcc      binary     NaN          0 NA
##   3 3.04e- 2   0.106 roc_auc  binary       0.629      8  0.00292
##   4 8.78e- 7   0.305 accuracy binary       0.896      8  0.0000490
##   5 8.78e- 7   0.305 mcc      binary      -0.00589    2  0
##   6 8.78e- 7   0.305 roc_auc  binary       0.627      8  0.00282
##   7 3.77e- 2   0.581 accuracy binary       0.896      8  0
##   8 3.77e- 2   0.581 mcc      binary     NaN          0 NA
```

```
##   9 3.77e- 2   0.581 roc_auc  binary       0.625       8  0.00261
## 10 1.24e-10   0.859 accuracy binary       0.896       8  0.0000548
## 11 1.24e-10   0.859 mcc      binary      -0.00589     3  0
## 12 1.24e-10   0.859 roc_auc  binary       0.627       8  0.00281
## 13 6.38e- 5   0.994 accuracy binary       0.896       8  0.0000548
## 14 6.38e- 5   0.994 mcc      binary      -0.00589     3  0
## 15 6.38e- 5   0.994 roc_auc  binary       0.627       8  0.00281
##    .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
## 13 Preprocessor1_Model5
## 14 Preprocessor1_Model5
## 15 Preprocessor1_Model5
```

```
##show best
show_best(
  glmnet_class_tune,
  metric = "roc_auc"
)
```

```
## # A tibble: 5 x 8
##    penalty mixture .metric .estimator  mean     n std_err .config
##      <dbl>   <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 3.04e- 2   0.106 roc_auc binary     0.629     8 0.00292 Preprocessor1_Model1
## 2 6.38e- 5   0.994 roc_auc binary     0.627     8 0.00281 Preprocessor1_Model5
## 3 1.24e-10   0.859 roc_auc binary     0.627     8 0.00281 Preprocessor1_Model4
## 4 8.78e- 7   0.305 roc_auc binary     0.627     8 0.00282 Preprocessor1_Model2
## 5 3.77e- 2   0.581 roc_auc binary     0.625     8 0.00261 Preprocessor1_Model3
```

```
##create final workflow
glmnet_class_wflow_final <-
  glmnet_class_wflow %>%
  finalize_workflow(
    tibble(
      penalty = 5.90e-2,
      mixture = 0.166
    )
  )

##fit to complete training data
glmnet_class_fit <-
  glmnet_class_wflow_final %>%
```

```r
  fit(staff_train)

##view coefficients
glmnet_class_fit %>%
  pull_workflow_fit() %>%
  tidy()
```

```
## # A tibble: 18 x 3
##    term           estimate penalty
##    <chr>             <dbl>   <dbl>
##  1 (Intercept)     -2.19     0.059
##  2 proactive        0.115    0.059
##  3 emot_intel       0.0732   0.059
##  4 sjt              0.0743   0.059
##  5 work_samp        0.0215   0.059
##  6 str_int          0.0356   0.059
##  7 consc            0.0504   0.059
##  8 cog_flex         0.0254   0.059
##  9 work_exp_1       0        0.059
## 10 work_exp_2       0        0.059
## 11 work_exp_3       0        0.059
## 12 degree_1         0        0.059
## 13 degree_2         0        0.059
## 14 degree_3         0        0.059
## 15 degree_4         0        0.059
## 16 wrok_exp_X2.5    0        0.059
## 17 wrok_exp_X6.10   0        0.059
## 18 wrok_exp_X11.    0        0.059
```

Task 8.3

Create a *support vector machine* model specification named **svm_class_spec**. Use the **svm_poly()** specification and set the **mode** to **classification**. Use the **kernlab** engine.

Create a *support vector machine* model workflow named **svm_class_wflow** using **workflow()**. Use **add_model()** to add a model using **svm_class_spec**. Use **add_recipe()** to add **staff_rec** and removing **job_perf** with **step_rm()**.

Create an object named **svm_class_folds** to save fitted models to folds using **svm_class_wflow**. In **fit_resamples()**, set **resamples** to **staff_train_folds** and **metrics** to **class_met**.

Apply **collect_metrics()** to **svm_class_folds**.

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

Use **pull_workflow_fit()** and **tidy()** on **svm_class_fit** to view summary results.

Questions 8.3: Answer these questions: (1) What is the *average accuracy* across the folds? (2) How many *support vectors* were produced using the complete *training* data?

Responses 8.3: *(1) 0.896 (2) 2935.*

```r
##support vector machine
svm_class_spec <-
  svm_poly(
```

```r
    mode = "classification"
  ) %>%
  set_engine("kernlab")

##create initial workflow
svm_class_wflow <- workflow() %>%
  add_model(svm_class_spec) %>%
  add_recipe(
    staff_rec %>%
      step_rm(job_perf)
  )

##estimate models
svm_class_folds <-
  svm_class_wflow %>%
  fit_resamples(
    resamples = staff_train_folds,
    metrics = class_met
  )

##show metrics
collect_metrics(svm_class_folds)
```

```
## # A tibble: 3 x 6
##   .metric  .estimator   mean     n std_err .config
##   <chr>    <chr>       <dbl> <int>   <dbl> <chr>
## 1 accuracy binary      0.896     8  0      Preprocessor1_Model1
## 2 mcc      binary    NaN         0 NA      Preprocessor1_Model1
## 3 roc_auc  binary      0.542     8  0.0273 Preprocessor1_Model1
```

```r
##fit to complete training data
svm_class_fit <-
  svm_class_wflow %>%
  fit(staff_train)
```

```
##  Setting default kernel parameters
## maximum number of iterations reached 0.002611544 0.002529389
```

```r
##view coefficients
svm_class_fit %>%
  pull_workflow_fit()
```

```
## parsnip model object
##
## Fit time:  8.4s
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Polynomial kernel function.
##  Hyperparameters : degree =  1  scale =  1  offset =  1
```

```
##
## Number of Support Vectors : 2940
##
## Objective Function Value : -2776.001
## Training error : 0.103923
## Probability model included.
```

Task 8.4

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

Create a *random forest* model workflow named **rf_class_wflow** using **workflow()**. Use **add_model()** to add a model using **rf_class_spec**. Use **add_recipe()** to add **staff_rec** and removing **job_perf** with **step_rm()**.

Create an object named **rf_class_folds** to save fitted models to folds using **rf_class_wflow**. In **fit_resamples()**, set **resamples** to **staff_train_folds** and **metrics** to **class_met**.

Apply **collect_metrics()** to **rf_class_folds**.

Create an object named **rf_class_fit** to save a fitted model to the complete *training* data using **rf_class_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_class_fit** to view the importance values of predictors.

Questions 8.4: Answer these questions: (1) What is the *average* **roc_auc** across the folds? (2) Which *predictor* is *most important*?

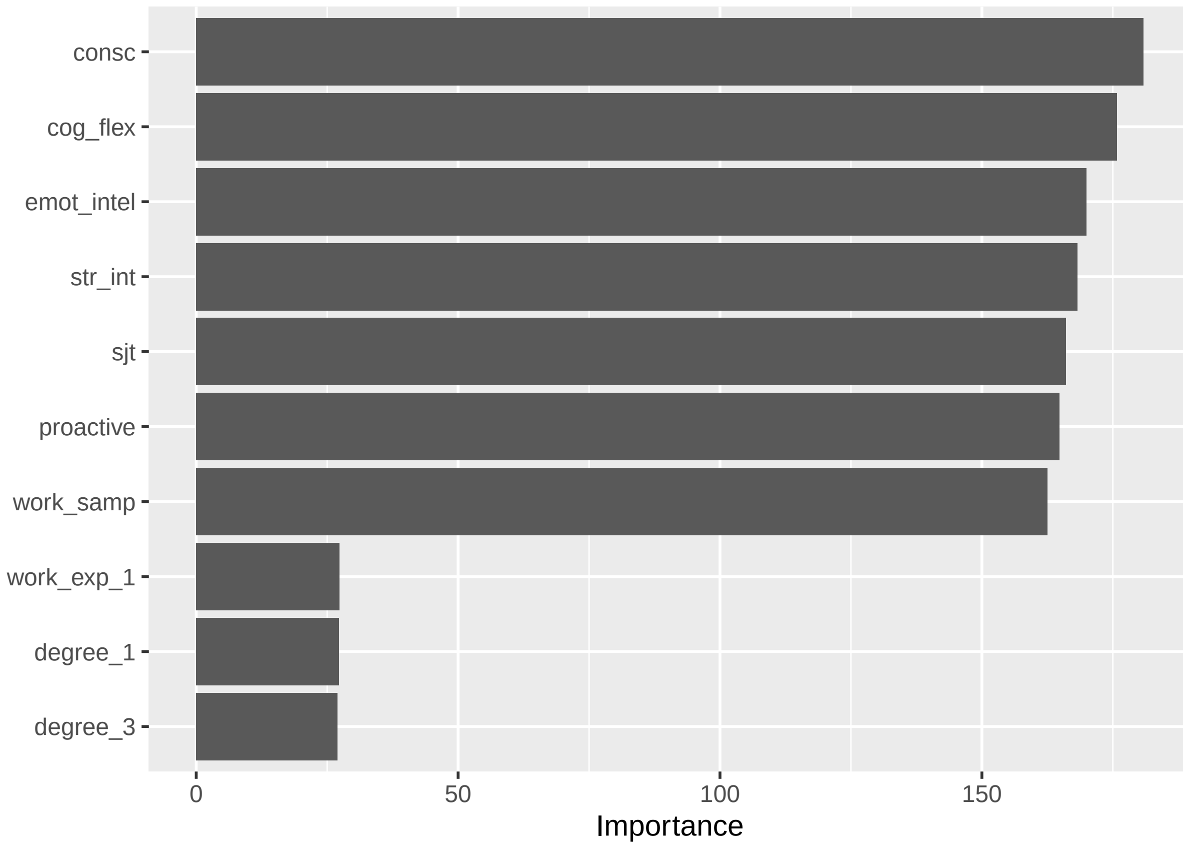
Responses 8.4: *(1) 0.595 (2)consc.*

```
##random forest
rf_class_spec <-
  rand_forest(
    mode = "classification"
  ) %>%
  set_engine("ranger")

##create initial work flow
rf_class_wflow <- workflow() %>%
  add_model(rf_class_spec) %>%
  add_recipe(
    staff_rec %>%
      step_rm(job_perf)
  )

##estimate models
rf_class_folds <-
  rf_class_wflow %>%
  fit_resamples(
    resamples = staff_train_folds,
    metrics = class_met
  )
```

```r
##show metrics
collect_metrics(rf_class_folds)
```

```
## # A tibble: 3 x 6
##   .metric  .estimator   mean     n  std_err .config
##   <chr>    <chr>       <dbl> <int>    <dbl> <chr>
## 1 accuracy binary     0.896      8 0.000470 Preprocessor1_Model1
## 2 mcc      binary     0.0295     8 0.0148   Preprocessor1_Model1
## 3 roc_auc  binary     0.601      8 0.00515  Preprocessor1_Model1
```

```r
##fit to complete training data
rf_class_fit <-
  rf_class_wflow %>%
  update_model(
    rand_forest(
      mode = "classification"
    ) %>%
      set_engine(
        "ranger",
        importance = "impurity"
      )
  ) %>%
  fit(staff_train)

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



Task 8.5

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

Create a *neural network* model workflow named **nn_class_wflow** using **workflow()**. Use **add_model()** to add a model using **nn_class_spec**. Use **add_recipe()** to add **staff_rec** and removing **job_perf** with **step_rm()**.

Create an object named **nn_class_folds** to save fitted models to folds using **nn_class_wflow**. In **fit_resamples()**, set **resamples** to **staff_train_folds** and **metrics** to **class_met**.

Apply **collect_metrics()** to **nn_class_folds**.

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

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

Questions 8.5: Answer these questions: (1) What is the *average accuracy* 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 8.5: *(1)0.878 (2) 17 (3) 571.*

```
##neural network
#model specificaiton
nn_class_spec <-
  mlp(
    mode = "classification",
```

```r
    hidden_units = 30,
    epochs = 100
  ) %>%
  set_engine("nnet")

##create initial workflow
nn_class_wflow <- workflow() %>%
  add_model(nn_class_spec) %>%
  add_recipe(
    staff_rec %>%
      step_rm(job_perf)
  )

##estimates models
nn_class_folds <-
  nn_class_wflow %>%
  fit_resamples(
    resamples = staff_train_folds,
    metrics = class_met
  )


##show metrics
collect_metrics(nn_class_folds)
```

```
## # A tibble: 3 x 6
##   .metric  .estimator   mean     n  std_err .config
##   <chr>    <chr>       <dbl> <int>    <dbl> <chr>
## 1 accuracy binary     0.880      8 0.000916 Preprocessor1_Model1
## 2 mcc      binary     0.0459     8 0.00784  Preprocessor1_Model1
## 3 roc_auc  binary     0.557      8 0.00431  Preprocessor1_Model1
```

```r
##fit to complete training data
nn_class_fit <-
  nn_class_wflow %>%
  fit(staff_train)

##view coefficients
nn_class_fit %>%
  pull_workflow_fit()
```

```
## parsnip model object
##
## Fit time:  6.4s
## a 17-30-1 network with 571 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 - entropy fitting
```

Task 9: Evaluate Categorical Outcome Models

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

Task 9.1

Create an object named **glm_pred**. Use **select()** on **staff_test** to choose **promotion**. Apply **bind_cols()** and **predict()** to **glm_fit** and **staff_test** with **type** set to **prob**.

Create an object named **glmnet_class_pred**. Use **select()** on **staff_test** to choose **promotion**. Apply **bind_cols()** and **predict()** to **glmnet_class_fit** and **staff_test** with **type** set to **prob**.

Create an object named **svm_class_pred**. Use **select()** on **staff_test** to choose **promotion**. Apply **bind_cols()** and **predict()** to **svm_class_fit** and **staff_test** with **type** set to **prob**.

Create an object named **rf_class_pred**. Use **select()** on **staff_test** to choose **promotion**. Apply **bind_cols()** and **predict()** to **rf_class_fit** and **staff_test** with **type** set to **prob**.

Create an object named **nn_class_pred**. Use **select()** on **staff_test** to choose **promotion**. Apply **bind_cols()** and **predict()** to **nn_class_fit** and **staff_test** with **type** set to **prob**.

Print a table of metrics on the models. Use **map_dfr()** and set the *data* input to a list using **list()** containing **glm_pred**, **glmnet_class_pred**, **svm_class_pred**, **rf_class_pred**, and **nn_class_pred**. Then, call **roc_auc()** as the function input to **map_dfr()**. Inside of **roc_auc()**, set the **data** to **.x**, **truth** to **promotion**, *estimate* to **.pred_Yes**, and **event_level** to **second**. Set the **.id** to **model**.

Questions 9.1: Answer these questions: (1) What is **roc_auc** of the *random forest* model? (2) Which model has the *highest* **roc_auc**?

Responses 9.1: *(1) 0.624 (2) glmnet had the highest roc_auc.*

```
##predictions
glm_pred <- staff_test %>%
  select(promotion) %>%
  bind_cols(
    predict(
      glm_fit,
      new_data = staff_test,
      type = "prob"
    )
  )
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
##elastic net
glmnet_class_pred <- staff_test %>%
  select(promotion) %>%
  bind_cols(
    predict(
      glmnet_class_fit,
      new_data = staff_test,
      type = "prob"))


##support vector machine
svm_class_pred <- staff_test %>%
  select(promotion) %>%
  bind_cols(
    predict(
      svm_class_fit,
```

```r
      new_data = staff_test,
      type = "prob"
    )
  )

##random forest
rf_class_pred <- staff_test %>%
  select(promotion) %>%
  bind_cols(
    predict(
      rf_class_fit,
      new_data = staff_test,
      type = "prob"
    )
  )

##neural network
nn_class_pred <- staff_test %>%
  select(promotion) %>%
  bind_cols(
    predict(
      nn_class_fit,
      new_data = staff_test,
      type = "prob"
    )
  )

##compute area under ROC curve
map_dfr(
  list(
    glm = glm_pred,
    glmnet =glmnet_class_pred,
    svm =svm_class_pred,
    rf = rf_class_pred
  ),
  ~roc_auc(
    data = .x,
    truth = promotion,
    .pred_Yes,
    event_level = "second"
  ),
  .id = "model"
)
```

```
## # A tibble: 4 x 4
##   model  .metric .estimator .estimate
##   <chr>  <chr>   <chr>          <dbl>
## 1 glm    roc_auc binary         0.638
## 2 glmnet roc_auc binary         0.635
## 3 svm    roc_auc binary         0.543
## 4 rf     roc_auc binary         0.615
```

Task 9.2

Create an object named **glm_roc** using **roc_curve()**. Set the *data* to **glm_pred**, **truth** to **promotion**, *prediction* to **.pred_Yes**, and **event_level** to **second**.

Create an object named **glmnet_roc** using **roc_curve()**. Set the *data* to **glmnet_class_pred**, **truth** to **promotion**, *prediction* to **.pred_Yes**, and **event_level** to **second**.

Create an object named **svm_roc** using **roc_curve()**. Set the *data* to **svm_class_pred**, **truth** to **promotion**, *prediction* to **.pred_Yes**, and **event_level** to **second**.

Create an object named **rf_roc** using **roc_curve()**. Set the *data* to **rf_class_pred**, **truth** to **promotion**, *prediction* to **.pred_Yes**, and **event_level** to **second**.

Create an object named **nn_roc** using **roc_curve()**. Set the *data* to **nn_class_pred**, **truth** to **promotion**, *prediction* to **.pred_Yes**, and **event_level** to **second**.

Create a plot named **roc_plot** using **ggplot()**. Call **geom_abline()** and create *gray diagonal dashed* line. Add *five* **geom_path()** layers with *data* set to **glm_roc**, **glmnet_roc**, **svm_roc**, **rf_roc**, and **nn_roc**, respectively. Map **1 - specificity** to the *x-axis*, **sensitivity** to the *y-axis*, and **color** to **glm**, **glmnet**, **svm**, **rf**, and **nn**, respectively. Apply **scale_color_manual()** correctly. Label the axes and legend correclty. Use **theme_bw()** and move the legend to the *bottom*.

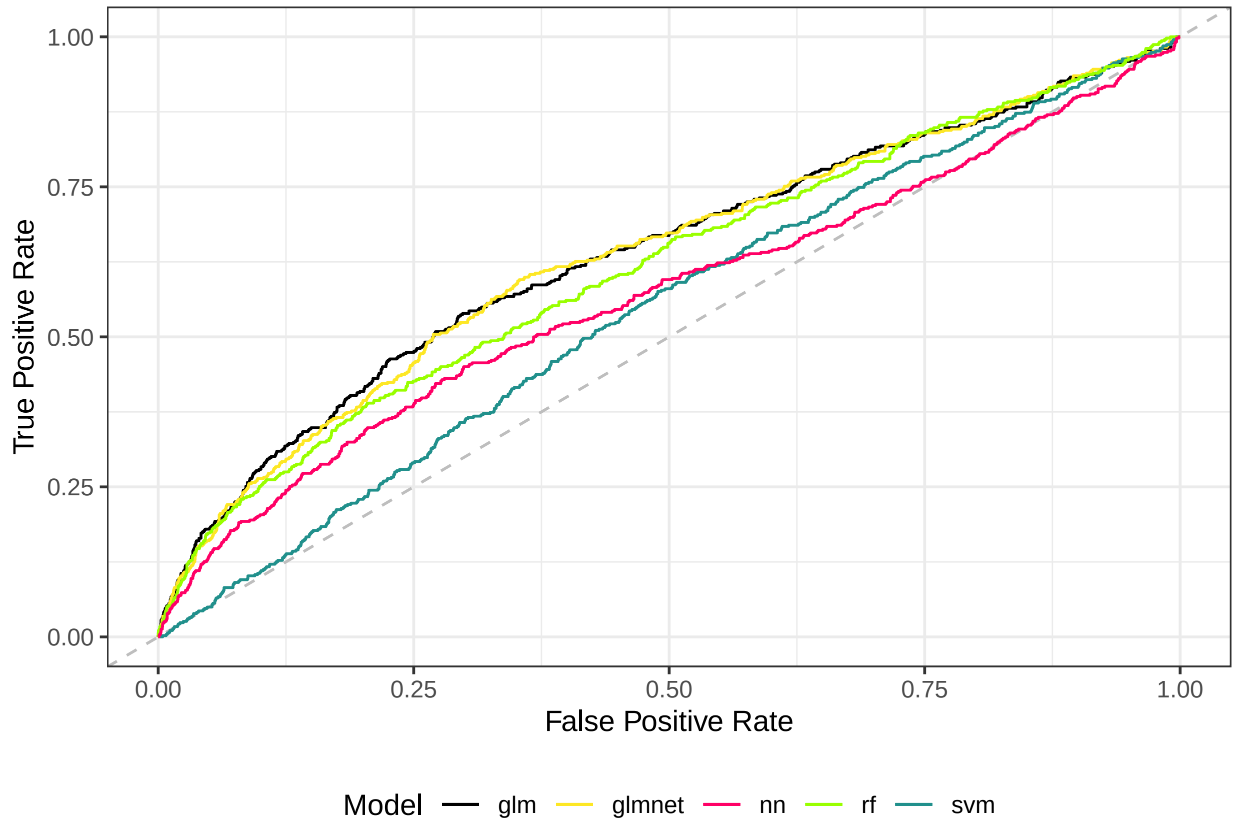
Display the plot.

Question 9.2: Does predicting *promotion* in this data require advanced machine learning models? Explain.

Response 9.2: *Yes because the SVM model is in the false positive rate and does not do a good job at predicting promotion. .*

```r
##compute ROC curve
glm_roc <- roc_curve(
  glm_pred,
  truth = promotion,
  .pred_Yes,
  event_level = "second"
)

##elastic net
glmnet_roc <- roc_curve(
  glmnet_class_pred,
  truth = promotion,
  .pred_Yes,
  event_level = "second"
)

##support vector machine
svm_roc <- roc_curve(
  svm_class_pred,
  truth = promotion,
  .pred_Yes,
  event_level = "second"
)

##random forest
rf_roc <- roc_curve(
  rf_class_pred,
  truth = promotion,
```

```r
  .pred_Yes,
  event_level = "second"
)

##neural network
nn_roc <- roc_curve(
  nn_class_pred,
  truth = promotion,
  .pred_Yes,
  event_level = "second"
)

##ROC curves
roc_plot <- ggplot() +
  geom_abline(linetype = 2, color = "gray") +
  geom_path(
    glm_roc,
    mapping = aes(x = 1 - specificity, y = sensitivity, color = "glm")
    )+
  geom_path(
    data = glmnet_roc,
    mapping = aes(x = 1 - specificity, y = sensitivity, color = "glmnet")
    )+
  geom_path(
    data = svm_roc,
    mapping = aes(x = 1 - specificity, y = sensitivity, color = "svm"),
    )+
  geom_path(
    data = rf_roc,
    mapping = aes(x = 1 - specificity, y = sensitivity, color = "rf"),
    )+
  geom_path(
    data = nn_roc,
    mapping = aes(x = 1 - specificity, y = sensitivity, color = "nn"),
    )+
  scale_color_manual(
    values = c(
      "glm" = "#000000", "glmnet" = "#FDE725FF",
      "svm" = "#21908CFF", "rf" = "#99FF00",
      "nn" = "#FF0066"
      )
    )+
  labs(x = "False Positive Rate", y = "True Positive Rate", color = "Model") +
  theme_bw() +
  theme(legend.position = "bottom")

##display plot
roc_plot
```



Task 10: Save Plots and Data

For this task, you will save the plots and the working data.

Task 10.1

Save **staff_train** and **staff_test** as the data files **staff_train.tsv** and **staff_test.tsv**, respectively, in the **data** folder of the project directory using **write_tsv()**.

Save the two plot objects as **png** files in the **plots** folder of the project directory. Save **reg_plot** as **reg.png** and **roc_plot** as **roc.png**. Use a width of *9 inches* and height of *9 inches* for all plots.

```
##save working data
write_tsv(
  staff_train,
  file = here("data", "staff_train.tsv")
  )

##save working data
write_tsv(
  staff_test,
  file = here ("data", "staff_test.tsv")
  )

##save plots to folder in project directory
ggsave(
```

```
  here("plots", "reg.png"),
  plot =reg_plot,
  units = "in", width = 9, height = 9
  )

##save a single plot to a file
ggsave(
  here("plots", "roc.png"),
  plot = roc_plot,
  units = "in", width = 9, height = 9
  )
```

Task 11: Conceptual Questions

For your last task, you will respond to conceptual questions based on the conceptual lectures for this week.

Question 11.1: What is the difference between *ridge*, *lasso*, and *elastic net* regression?

Response 11.1: *Ridge is penalizing predictors if they are too far away from zero which foreces then to be small in a continuous way. Lasso also adds a penalty for non-zero coeffiences but penalizes their absolute values. The elastic net is the combination of the ridge and lasso errors.*

Question 11.2: What is *repeated v-fold cross-validation*? Provide an example.

Response 11.2: *the repated v- fold cross validation splits data into equal v groups. FOr example, if you have v= 2 and the total of splits is equal to 10 then you have 6 groups of 2 that are generated seperately. .*

Question 11.3: What is a *tuning parameter*?

Response 11.3: *A tuning parameter is also known as the ridge regression penalty. It controls for the strength of the penalty temr in both the ridge regression and lasso regression.*