Homework 2

BIOS 635

...

1/28/2021

```
knitr::opts_chunk$set(echo = FALSE, message = FALSE, warning = FALSE, include=TRUE,
                      fig.width = 10, fig.height = 5)
```

```
library(tidyverse)
library(broom)
library(gtsummary)
library(flextable)
library(gt)
library(caret)
library(GGally)
library(e1071)
```

Introduction

In this assignment you will practice using some basic machine learning methods and concepts discussed so far in lecture to develop prediction algorithms from real-world public health datasets. You will predict a continuous outcome first, followed by a binary outcome ("Yes" or "No), using K-nearest neighbor, linear regression, and logistic regression

1

Setup

In the first part, you will work with cancer mortality data in the United States at the county level from 2010-2016, with demographic information on the counties from 2013 US Census estimates.

The outcome of interest in the data is mean yearly per capita (100,000 people) cancer mortalities from 2010-2016, denoted `TARGET_deathRate` in the dataset (`cancer_reg.csv`). So more info on the dataset in the docs folder.

A

First, let's look at summary statistics of the variables of interest in the data using the function `tbl_summary` in the `gtsummary` package. Be sure to print the table as a `flextable` using the function `as_flex_table`. Specifically:

- First, create variable `deathrate_vs_median` in dataset after reading in CSV
  - `deathrate_vs_median`="No" if `TARGET_deathRate`< `median(TARGET_deathRate)`
  - ="Yes" otherwise
- Provide stats for the following variables: `-TARGET_deathRate`, `medIncome`, `povertyPercent`, `MedianAge`, `PctPrivateCoverage`, `PctPublicCoverage`, `PctWhite`, `PctBlack`, `PctAsian`, `PctOtherRace`
  - **NOTE**: Don't remove variables from dataset to only those marked above. Only use functions in `gtsummary` to remove variables from table (see `include` argument)
  - Group the summary statistics by `deathrate_vs_median`
  - Include sample size N using `add_n`
  - Add p-values from one-way ANOVA test for differences in variables between "No" and "Yes" groups of `TARGET_deathRate`
  - For all variables, provide mean and standard deviation (SD) as statistics
  - Add a gray background to the cells in the row corresponding to `TARGET_deathRate`
    - **Hint**: Look at changing row/column background color in `flextable` package after using `as_flex_table` function
  - Also, bold text in header row after using `as_flex_table`

Characteristic	**N**	**No, N = 1,520[1]**	**Yes, N = 1,527[1]**	**p-value[2]**
TARGET_deathRate	3,047	157 (16)	200 (19)	<0.001
medIncome	3,047	51,686 (13,097)	42,461 (8,725)	<0.001
povertyPercent	3,047	15 (6)	19 (6)	<0.001
MedianAge	3,047	45.3 (45.7)	45.3 (45.0)	>0.9
PctPrivateCoverage	3,047	68 (11)	61 (9)	<0.001
PctPublicCoverage	3,047	34 (8)	39 (7)	<0.001
PctWhite	3,047	86 (14)	81 (18)	<0.001
PctBlack	3,047	6 (11)	12 (17)	<0.001
PctAsian	3,047	1.72 (3.34)	0.79 (1.43)	<0.001
PctOtherRace	3,047	2.58 (4.31)	1.39 (2.35)	<0.001

Mean (SD)[1]

One-way ANOVA[2]

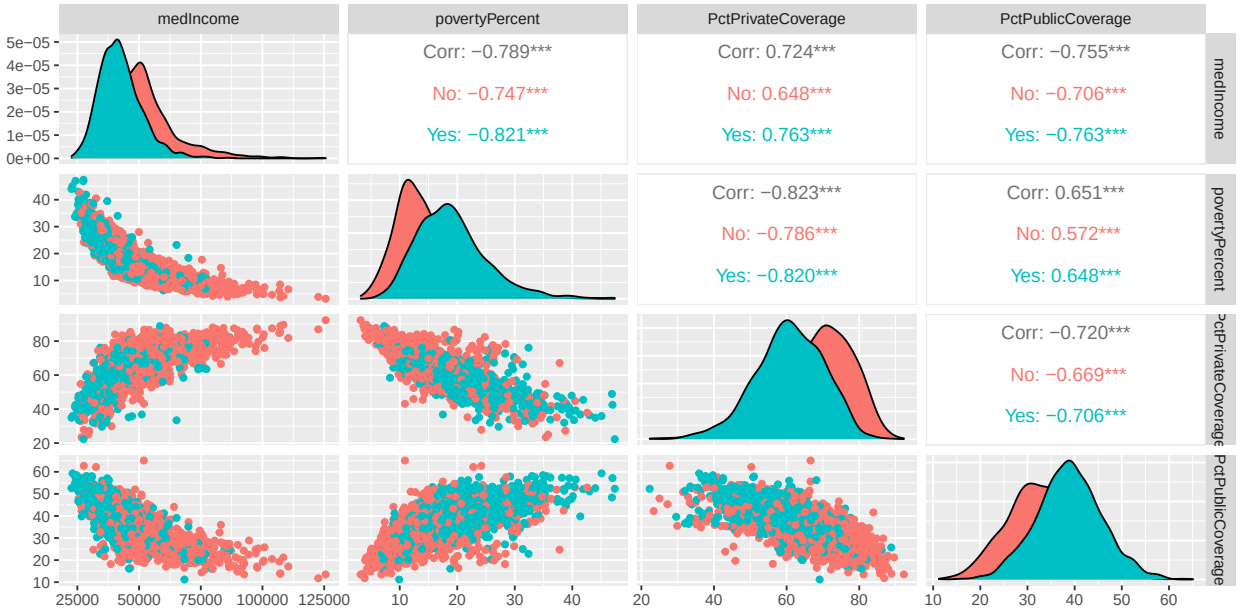
B

Now, let's do some data visualization.

Let's look at some 2-dimensional scatterplots of some of the above variables to assess correlation. Specifically, recreate the following matrix of scatterplots:

- Look at the following variables
  - Use `ggpairs` from the `GGally` package:
    - https://www.r-graph-gallery.com/199-correlation-matrix-with-ggally.html
  - `medIncome`, `povertyPercent`, `PctPrivateCoverage`, `PctPublicCoverage`
  - Color points by `deathrate_vs_median`

- Provide some interpretation of the relationships you see in the figure. Specifically:
  - Are there variables that have high correlations?
    - Do these high correlations make sense conceptually?
  - Compare the distributions of the variables between the two mortality rate groups (see diagonal).



Based on the figure, all variables selected, poverty percentage, private health care coverage, and government-provided health coverage, correlate highly with the median death rate. This does not make conceptual sense to me. Correlations are not appropriate to analyze these data. Based on the diagonal, it seems that the individual variables are normally distributed with some skewness.

C

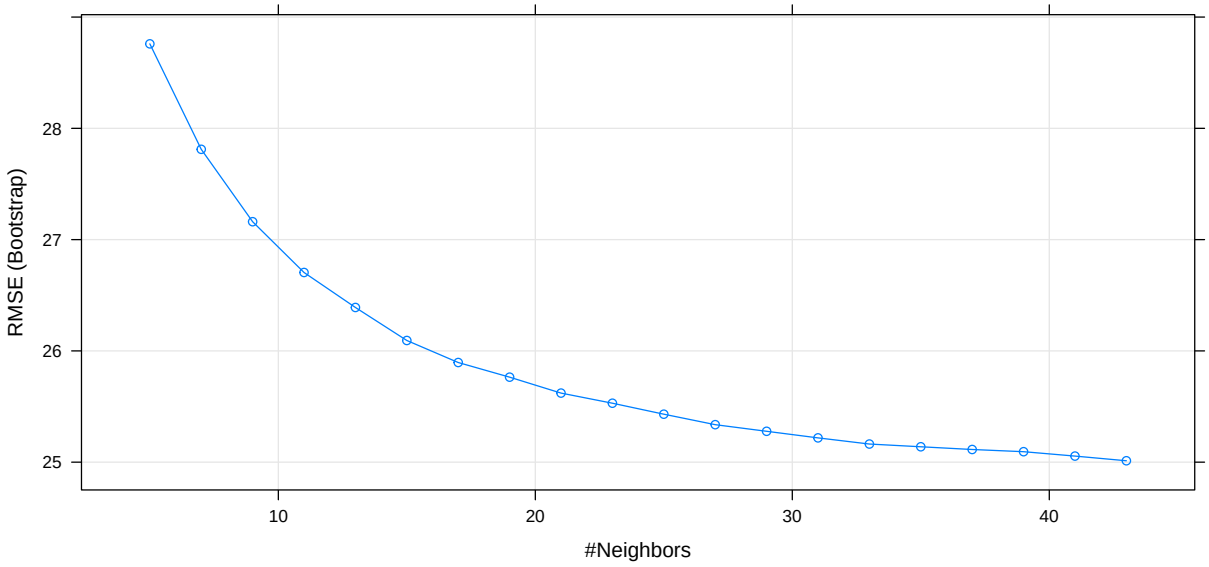
Now, let's begin to create our prediction algorithms for `TARGET_deathRate`. First, we will start with using K-nearest neighbor (KNN).

Let's consider the features included in our summary statistics table (`TARGET_deathRate`, `medIncome`, `povertyPercent`, `MedianAge`, `PctPrivateCoverage`, `PctPublicCoverage`, `PctWhite`, `PctBlack`, `PctAsian`, `PctOtherRace`).

- First, we will split our data into separate training and testing sets (60% in training and 40% in testing) randomly.

- Next, train a KNN algorithm on the training dataset.
  - Use `train` function in `caret` function (see lecture slides). Use `tuneLength`=20 and center and scale the features (see `preProcess` argument).
  - Leave everything else at default. What is the "best" tuning parameter value chosen for parameter k? What criteria is used by R to select this "best" parameter?
  - Plot the RMSE for each considered value of k during the tuning process. What does k represent based on the plot (**Hint**: see lecture slides and x-axis of plot)
- Lastly, test your algorthim at this "best" tuning parameter value on the test set. Print out the test set performance based on RMSE, R^2, and MAE using `flextable`

```
## k-Nearest Neighbors
##
## 1830 samples
##    9 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 1830, 1830, 1830, 1830, 1830, 1830, ...
## Resampling results across tuning parameters:
##
##   k   RMSE      Rsquared    MAE
##    5  28.75924  0.07915240  21.87622
##    7  27.81163  0.09203467  21.13764
##    9  27.16083  0.10540807  20.66236
##   11  26.70468  0.11535739  20.30006
##   13  26.39013  0.12370580  20.05417
##   15  26.09354  0.13374493  19.80187
##   17  25.89560  0.14074446  19.65314
##   19  25.76389  0.14554280  19.54177
##   21  25.62079  0.15146141  19.43995
##   23  25.52976  0.15488438  19.38134
##   25  25.43128  0.15917878  19.31174
##   27  25.33701  0.16387628  19.24222
##   29  25.27743  0.16665114  19.18275
##   31  25.21832  0.16916290  19.13966
##   33  25.16298  0.17181051  19.10422
##   35  25.13842  0.17311999  19.07886
##   37  25.11383  0.17410562  19.06013
##   39  25.09413  0.17505617  19.04945
##   41  25.05475  0.17718553  19.01985
##   43  25.01264  0.17938479  18.98506
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 43.
```



k	RMSE	Rsquared	MAE
5	28.7	0.1	21.8
7	27.8	0.1	21.1
9	27.2	0.1	20.6
11	26.8	0.1	20.3
13	26.5	0.1	20.1
15	26.4	0.1	19.9
17	26.2	0.1	19.8
19	26.0	0.1	19.6
21	25.9	0.1	19.5
23	25.9	0.2	19.5
25	25.7	0.2	19.4
27	25.7	0.2	19.3
29	25.6	0.2	19.2
31	25.6	0.2	19.2
33	25.5	0.2	19.1
35	25.5	0.2	19.1
37	25.4	0.2	19.1
39	25.4	0.2	19.0
41	25.3	0.2	19.0
43	25.3	0.2	18.9

- The "best" tuning parameter value chosen for parameter k is 43.

- R takes a determines the "best" tuning parameter based on a combination of the lowest RMSE, highest R^2, and lowest MAE. In other words, R looks for the parameter that explains the most variance and produces low error.
- Based on the plot, k represents the number of nearest neighbors.

D

I

Let's next move to a linear regression model for prediction. We consider the same features listed in 1c with the same outcome variable.

- Use the same training and testing sets created in 1c
- Train a linear regression algorithm with all of the above features. Print out the following results:
  - Coefficient estimate table from `summary` function (estimate, standard error, test statistic, p-value)
    - Create this table using the `tidy` function from `broom` and print out using `flextable`
  - Evaluate the following assumptions using the corresponding visual
    - 1. Homoskedasicity (fitted value by residual plot)
    - 2. Normality (QQ plot of residuals vs theoretical normal distribution)
  - One may argue that normally distributed residuals are not a concern for this dataset. Why?
  - One common belief in regression is that your **outcome** is assumed to be normally distributed. Why is this incorrect?

II

- Test the algorithm developed in the previous step on the test dataset. Print out the following in a `flextable`
  - Test set RMSE, R^2, adjusted R^2, and MAE
- In a separate `flextable`, print out these same metrics based on the performance in the training set
  - Evaluate the differences between the training and testing performance
- Based on your plots in 1b, do you have any concerns about collinearity? If so, how would you change the set of feature variable used to fix this concern? How did you choose this set?
  - **Note**: you don't need to actually re-run the regression analysis with this reduced set of features

term	estimate	std.error	statistic	p.value
(Intercept)	181.0	15.4	11.7	<0.005
medIncome	-0.0	0.0	-1.5	0.13
povertyPercent	0.5	0.2	2.5	0.01
MedianAge	-0.0	0.0	-1.2	0.23
PctPrivateCoverage	-0.4	0.1	-3.5	<0.005
PctPublicCoverage	0.4	0.1	3.5	<0.005
PctWhite	0.1	0.1	1.0	0.31
PctBlack	0.3	0.1	3.2	<0.005
PctAsian	-0.1	0.3	-0.3	0.74
PctOtherRace	-1.5	0.2	-7.5	<0.005

term	estimate	std.error	statistic	p.value
(Intercept)	180.3	17.8	10.1	<0.005
medIncome	-0.0	0.0	-2.1	0.03
povertyPercent	0.9	0.3	3.3	<0.005
MedianAge	0.0	0.0	1.4	0.17
PctPrivateCoverage	-0.1	0.1	-0.7	0.49
PctPublicCoverage	0.4	0.1	2.8	<0.005
PctWhite	-0.1	0.1	-1.2	0.21
PctBlack	0.0	0.1	0.3	0.77
PctAsian	-0.7	0.3	-2.1	0.03
PctOtherRace	-1.5	0.2	-7.5	<0.005

Part I

Based on the residuals vs fitted plot, the data fits the assumption of Homoskedasicity. The values appear to be scattered evenly without any clear trends. There are outliers present that need to be addressed. Based on the QQ plot, the resdiuals appear to be normally distributed (not the best), with the exception of outliers. The normally distributed residuals are not a concern for this dataset, because they follow the assumptions of a linear model. Your **outcome** variable is not always normally distributed is incorrect, because not all **outcome**variables follow a normal distribution. An **outcome** varaible can be binary or logistic.

Part II

The significance of the regression coefficients changes in the test data set. For example, the variable "medIncome" has a p-value of 0.00 in the train data set, but a p-value in the test data set is 0.42. However, the regression coefficients are relatively the same in the train and the test data set. Based on the graph in 1b, multi-collinearity is definitely an issue here. The high correlations between all of the varaibles are inducing linearly dependent results. Using different variables with smaller correlation values or using a step-wise regression procedure would help.

2

Setup

In the second part, you will work with diabetes incidence data in the US, composed of Native American, female hospital patients at 21 years old.

The outcome of interest, `Outcome` in the data is binary indicator if the patient has a diagnosis of diabetes (0 = "No", 1 = "Yes"). You will try to predict this outcome based on patient traits as features. See the docs folder for more information. The dataset is called `diabetes_data.csv`.

A

First, let's look at summary statistics of the variables of interest in the data using the function `tbl_summary` in the `gtsummary` package. Be sure to print the table as a `flextable` using the function `as_flex_table`. Specifically:

- Provide stats for the following variables:
  - `Pregnancies`, `Glucose`, `BloodPressure`, `SkinThickness`, `Insulin`, `BMI`, `Age`
  - **NOTE**: Don't remove variables from dataset to only those marked above. Only use functions in `gtsummary` to remove variables from table (see `include` argument)
  - Group the summary statistics by `Outcome`
  - Include sample size N using `add_n`
  - Add p-values from one-way ANOVA test for differences in variables between groups of `Outcome`
  - For all variables, provide mean and standard deviation (SD) as statistics
  - Also, bold text in header row after using `as_flex_table`

Characteristic	**N**	**0, N = 500[1]**	**1, N = 268[1]**	**p-value[2]**
Pregnancies	768	3.3 (3.0)	4.9 (3.7)	<0.001
Glucose	768	110 (26)	141 (32)	<0.001
BloodPressure	768	68 (18)	71 (21)	0.072
SkinThickness	768	20 (15)	22 (18)	0.038
Insulin	768	69 (99)	100 (139)	<0.001
BMI	768	30 (8)	35 (7)	<0.001
Age	768	31 (12)	37 (11)	<0.001

Mean (SD)[1]

One-way ANOVA[2]

B

Now, let's begin to create our prediction algorithms for `Outcome`. First, we will start with using K-nearest neighbor (KNN).

Let's consider the features included in our summary statistics table (`Pregnancies`, `Glucose`, `BloodPressure`, `SkinThickness`, `Insulin`, `BMI`, `Age`).

- First, we will split our data into separate training and testing sets (60% in training and 40% in testing) randomly.
- Next, train a KNN algorithm on the training dataset.
  - Use `train` function in `caret` function (see lecture slides). Use `tuneLength`=20 and center and scale the features (see `preProcess` argument).
  - Leave everything else at default. What is the "best" tuning parameter value chosen for parameter k? What criteria is used by R to select this "best" parameter?
  - Plot the Prediction Accuracy for each considered value of k during the tuning process. What does k represent based on the plot (**Hint**: see lecture slides and x-axis of plot)
- Lastly, test your algorithm at this "best" tuning parameter value on the test set. Print out the test set performance based on Prediction Accuracy, Sensitivity, Specificity, PPV, and NPV using `flextable`.
  - **Hint**: Use `confusionMatrix` function in `caret` package. Then convert to data frame to print as `flextable`

```
## k-Nearest Neighbors
```

```
##
## 461 samples
##   7 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 461, 461, 461, 461, 461, 461, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.6987391  0.3233315
##    7  0.7194479  0.3588123
##    9  0.7269048  0.3740956
##   11  0.7249846  0.3632310
##   13  0.7350650  0.3854438
##   15  0.7366397  0.3808936
##   17  0.7367052  0.3787175
##   19  0.7385311  0.3790794
##   21  0.7378082  0.3726191
##   23  0.7389886  0.3737843
##   25  0.7404617  0.3758774
##   27  0.7328174  0.3562299
##   29  0.7309626  0.3501954
##   31  0.7309367  0.3492696
##   33  0.7275309  0.3417243
##   35  0.7284423  0.3414663
##   37  0.7251696  0.3328981
##   39  0.7300305  0.3454921
##   41  0.7305493  0.3424064
##   43  0.7279164  0.3362257
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 25.
```

rowname	Data
Accuracy	0.7
Sensitivity	0.4
Specificity	0.9
Pos Pred Value	0.7
Neg Pred Value	0.7

Answers to 2b

- The "best" tuning parameter value chosen for parameter k is 25.
- R looks for the parameter that produces the highest accuracy rate and the highest Kappa value, or the classification accuracy, taking random chance into account.

- Based on the plot, k represents the number of nearest neighbors.

C

Finally, we will end with using logistic regression. We consider the same features listed in 2b with the same outcome variable.

- Train a logistic regression algorithm with all of the above features. Print out the following results:
  - Coefficient estimate table from `summary` function (estimate, standard error, test statistic, p-value)
    - Create this table using the `tidy` function from `broom` and print out using `flextable`
  - Print out the test set performance based on Prediction Accuracy, Sensitivity, Specificity, PPV, and NPV using `flextable`.
    - **Hint**: Use `confusionMatrix` function in `caret` package. Then convert to data frame to print as `flextable`

term	estimate	std.error	statistic	p.value
(Intercept)	-9.8	1.0	-9.7	<0.005
Pregnancies	0.1	0.0	2.9	<0.005
Glucose	0.0	0.0	8.1	<0.005
BloodPressure	-0.0	0.0	-2.2	0.03
SkinThickness	0.0	0.0	0.5	0.59
Insulin	-0.0	0.0	-1.8	0.08
BMI	0.1	0.0	5.5	<0.005
Age	0.0	0.0	2.0	0.05

rowname	Data
Accuracy	0.7
Sensitivity	0.5
Specificity	0.9
Pos Pred Value	0.7
Neg Pred Value	0.8