MATH 4323 Project Report

Shaheer Khan (1855168), Daniel Onitilo (2098047), Tobi Oladunjoye (2001728), Jared Gallardo (1822320)

Professor Wang

11/29/22

INTRODUCTION (Daniel Onitilo)

Inspiration

We spent some time sifting through various data analysis-related topics such as shopping habits, social media consumption, medicine and diseases, as well as sports. Since our group is comprised of seniors graduating within the next year, the job market proved to be a topic that caught our attention. In particular, we were most interested in wage statistics. The topic of wages tends to be a somewhat controversial one, especially within the context of inflation. As inflation rises, wage value declines and this encouraged us to study wage statistics across the U.S. population.

Dataset and Goal

The data selected for this project is the Census Income Dataset, obtained from the UCI Machine Learning Repository. It is a multivariate set with 48,842 observations, and 14 variables. These 14 variables can be further broken down to 13 independent variables and 1 dependent variables, as well as 9 categorical and 5 integer variables. This dataset was selected because it helps us to determine the factors that are highly correlated with a person's wage. It is also best suited for classification tasks, which is our approach to data analysis. In doing so, we can learn the most important factors to focus on, in order to earn ourselves a "high" wage. Our overall goal is to answer this question: **does the average U.S. resident make above or below the median wage value?**

Some variables from the selected dataset and their descriptions are:

- *Age* (Continuous): The age of a person.
- *Workclass* (Classification): General working class a person belongs to.
- *Fnlwgt* (Continuous): The number of people represented by this entry in the population.
- *Education* (Classification): Highest level of education attained by a person
- *Marital-status* (Classification): Marital status of a person.
- *Occupation* (Classification): Nature and type of a person's job.
- *Race* (Classification): The racial identity of a person.
- *NativeCountry* (Classification): The nation a person was born in.

METHODOLOGY

(Daniel Onitilo)

Since we are using a supervised learning approach, the methods we employed for our data analysis were the K-Nearest Neighbors Classification and Support Vector Machines.

KNN Algorithm

KNN is a method to determine the classification of an observation using the K-nearest observations around that point. The KNN approach takes

- a positive integer K,
- and a test observation $x_0 = (x_1, \dots, x_p)$ as inputs.

From this, it identifies the K points in a training dataset closest to the test observation. The metric for distance which is most often used, is the Euclidean distance metric. Then, the KNN algorithm estimates the conditional probability for a given class using the formula below:

$$P(Y = j | X = x_0) = \frac{1}{K} \sum_{i \in N_0} \mathbb{I}(y_i = j)$$

Where j is the given class. And lastly, it then classifies the observation to the class with the highest estimated probability.

An instance of the algorithm: if $K = 3$ is selected then we will find the 3 nearest neighbors, using Euclidean Distance, and find the classification with the most number of neighbors.

Some advantages of the KNN method include its simplicity, relatively high accuracy, and multiple options for distance criteria. There are some drawbacks to the KNN approach, such as its computationally demanding nature, lack of model, and sensitivity to irrelevant features of the data. Before performing this model’s computations, we will scale the data to avoid predictors with large values dominating lower valued predictors, and to normalize the data.

Support Vector Machines

The SVM approach is classification-based approach for working with observations. It employs a maximal margin hyperplane to classify and separate observations. In the generalized n-dimensional vector space $\mathbb{R}^n = \{(x_1, x_2, \dots x_n)^T | x_1, x_2, \dots x_n \in \mathbb{R}\}$, the hyperplane is a subspace of dimension $p - 1$ and can be parametrically defined as

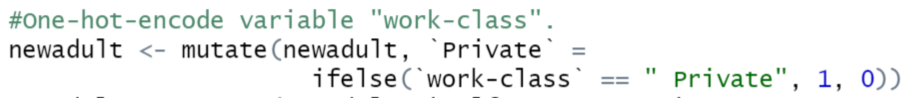
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + B_n X_n = 0$$

This hyperplane separates the vector space into two halves: whether the hyperplane is greater than zero, or less than zero.

Some advantages of the SVM approach include its effectiveness for higher dimensional spaces and instances of good margin of dissociation between variables, and reduced memory usage. Drawbacks include difficulty working with large datasets, and potential for overfitting. To ensure that we do not run into the issue of overfitting, we intend to introduce the Support Vector Classifier approach for better classification of the majority of training observations. We also intend to run the algorithm a set number of times, each time with a different value for the tuning parameter. The tuning parameter in this case is a nonnegative value which functions as a budget for margin violations committed by some observations as a result of the SVC implementation.

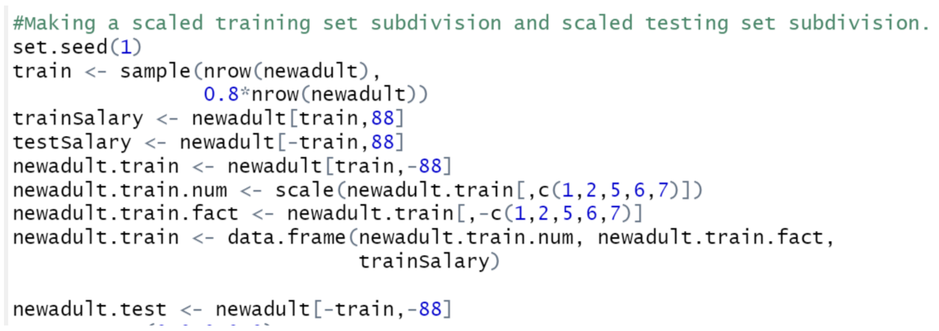
DATA ANALYSIS (The whole group)

Before fitting any models on our dataset, we had to clean up our data and make it ready for analysis. Since the large size of the data made it difficult to run any algorithms with it, the first thing we did was to reduce the size of the data, from 48,842 observations to around 5000 observations. We then proceeded to "one-hot-encode" the dataset. This is essentially a process that converts categorical variables (such as color) to numerical values (in most cases 1s and 0s), so our algorithms can work with the data directly. Some variables that went through this process include *marital-status*, *occupation*, *work-class*, *relationship*, *race*, etc. Then, we removed the redundant variables that we converted via one-hot-encoding, and this process changed the number of variables to 88. A sample one-hot-encode is shown below



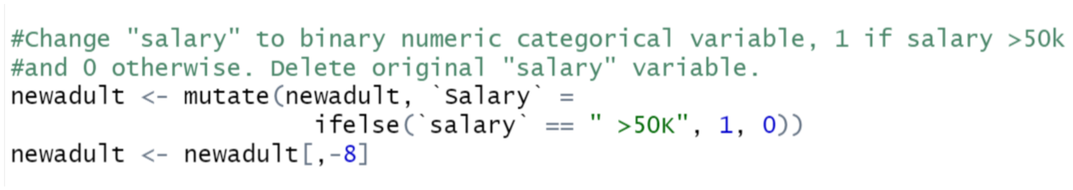
```
#One-hot-encode variable "work-class".
newadult <- mutate(newadult, `Private` =
                    ifelse(`work-class` == " Private", 1, 0))
```

Not all our variables are of the same range. For example, *final-weight*, *capital-gain*, and *capital-loss*, all have values in the thousands. This issue made it necessary for us to scale our data. We split our data into two sections: 80% for training data, 20% for testing, and then scaled these two parts.



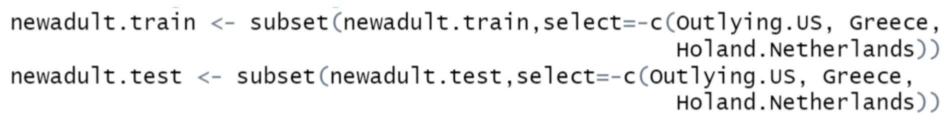
```
#Making a scaled training set subdivision and scaled testing set subdivision.
set.seed(1)
train <- sample(nrow(newadult),
                0.8*nrow(newadult))
trainSalary <- newadult[train,88]
testSalary <- newadult[-train,88]
newadult.train <- newadult[train,-88]
newadult.train.num <- scale(newadult.train[,c(1,2,5,6,7)])
newadult.train.fact <- newadult.train[,-c(1,2,5,6,7)]
newadult.train <- data.frame(newadult.train.num, newadult.train.fact,
                             trainSalary)

newadult.test <- newadult[-train,-88]
```

We modified the salary variable using the code above into a binary variable, which we will use to determine if the average U.S. resident makes above or below the median wage.



```
#Change "salary" to binary numeric categorical variable, 1 if salary >50k
#and 0 otherwise. Delete original "salary" variable.
newadult <- mutate(newadult, `Salary` =
                    ifelse(`salary` == " >50K", 1, 0))
newadult <- newadult[,-8]
```

For the SVM implementation, we removed variables that were factors with only one unique value.

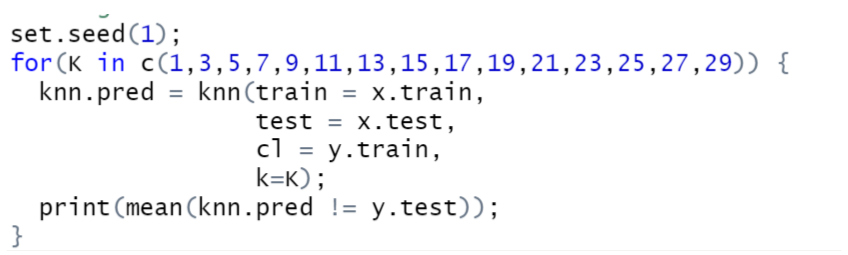


```
newadult.train <- subset(newadult.train,select=-c(Outlying.US, Greece,
                                                  Holand.Netherlands))
newadult.test <- subset(newadult.test,select=-c(Outlying.US, Greece,
                                                  Holand.Netherlands))
```

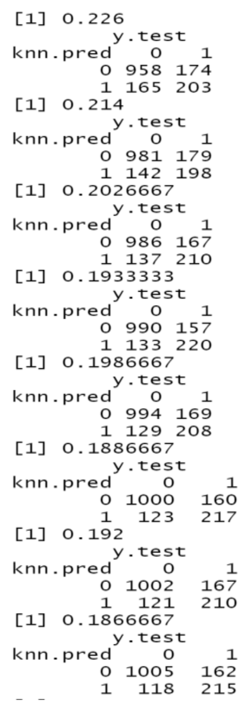
With the cleaning and normalization completed, we were able to then fit our models on our datasets.

KNN Modeling (Shaheer Khan)

We tested the KNN-model first using a validation set approach. We started by training our model on the training data, and calculated the predictor error on the testing data.

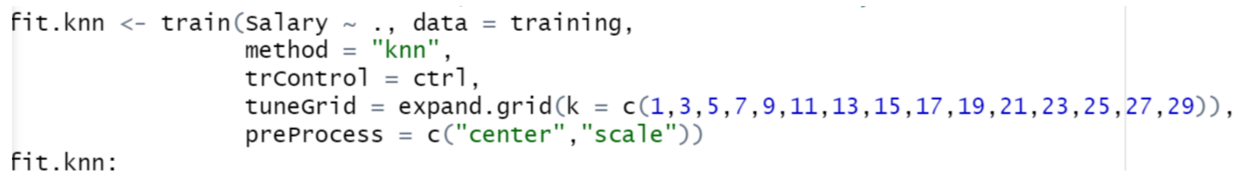


```
set.seed(1);
for(K in c(1,3,5,7,9,11,13,15,17,19,21,23,25,27,29)) {
  knn.pred = knn(train = x.train,
                 test = x.test,
                 cl = y.train,
                 k=K);
  print(mean(knn.pred != y.test));
}
```



```
[1] 0.226
        y.test
knn.pred   0   1
       0 958 174
       1 165 203
[1] 0.214
        y.test
knn.pred   0   1
       0 981 179
       1 142 198
[1] 0.2026667
        y.test
knn.pred   0   1
       0 986 167
       1 137 210
[1] 0.1933333
        y.test
knn.pred   0   1
       0 990 157
       1 133 220
[1] 0.1986667
        y.test
knn.pred   0   1
       0 994 169
       1 129 208
[1] 0.1886667
        y.test
knn.pred    0    1
       0 1000  160
       1  123  217
[1] 0.192
        y.test
knn.pred    0    1
       0 1002  167
       1  121  210
[1] 0.1866667
        y.test
knn.pred    0    1
       0 1005  162
       1  118  215
```

We also implemented KNN with K-Fold cross-validation to find the optimal K value and improve the test error rate.



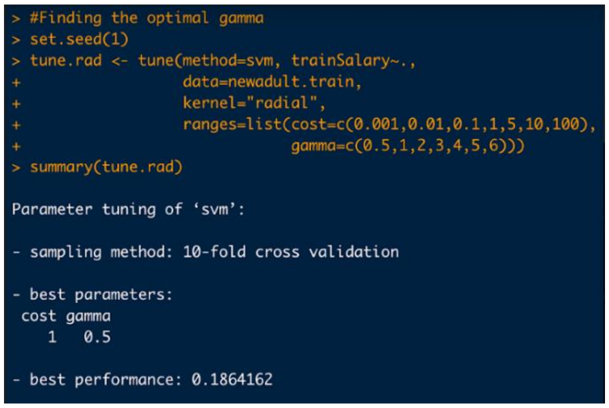
```
fit.knn <- train(Salary ~ ., data = training,
                 method = "knn",
                 trControl = ctrl,
                 tuneGrid = expand.grid(k = c(1,3,5,7,9,11,13,15,17,19,21,23,25,27,29)),
                 preProcess = c("center","scale"))
fit.knn:
```

After both approaches, we selected the optimal K value of 13 with an accuracy of 91.97% and a κ value of 0.7772.

SVM Modeling (Jared Gallardo, Tobi Oladunjoye)

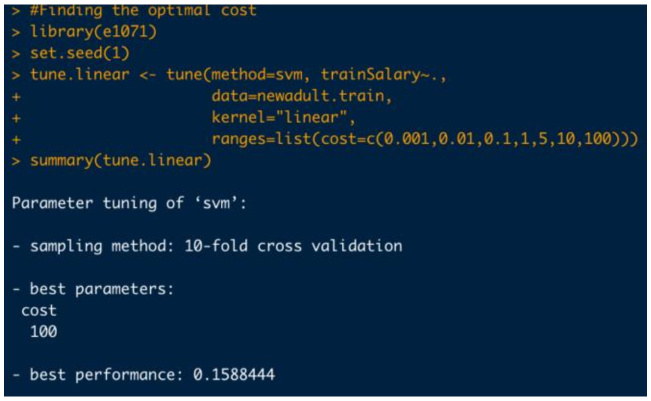
For the SVM approach, we first sought to determine the optimal kernel. We tested all three kernes: radial, polynomial and linear, and the results obtained are shown below.

For radial, we got a best cost $c = 100, \gamma = 0.5$ and test error rate of 0.1864



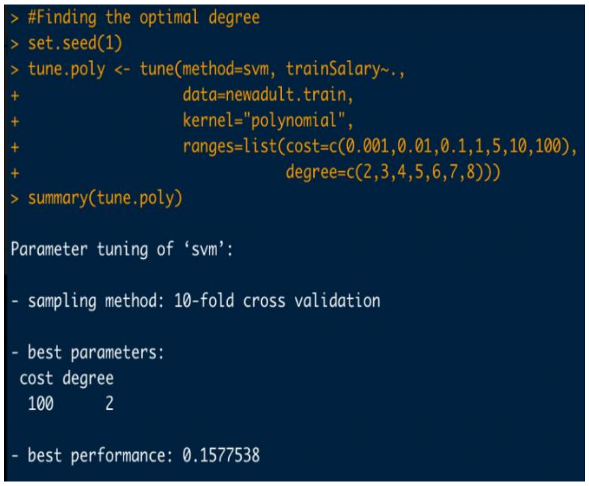
```
> #Finding the optimal gamma
> set.seed(1)
> tune.rad <- tune(method=svm, trainSalary~.,
+                  data=newadult.train,
+                  kernel="radial",
+                  ranges=list(cost=c(0.001,0.01,0.1,1,5,10,100),
+                              gamma=c(0.5,1,2,3,4,5,6)))
> summary(tune.rad)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
    1   0.5

- best performance: 0.1864162
```

For linear, we obtained a best cost $c = 100$ with a test error rate of 0.1588



```
> #Finding the optimal cost
> library(e1071)
> set.seed(1)
> tune.linear <- tune(method=svm, trainSalary~.,
+                     data=newadult.train,
+                     kernel="linear",
+                     ranges=list(cost=c(0.001,0.01,0.1,1,5,10,100)))
> summary(tune.linear)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost
  100

- best performance: 0.1588444
```

For the polynomial kernel, we obtained a best cost $c = 100$, with degree of 2 and a test error of 0.1578.

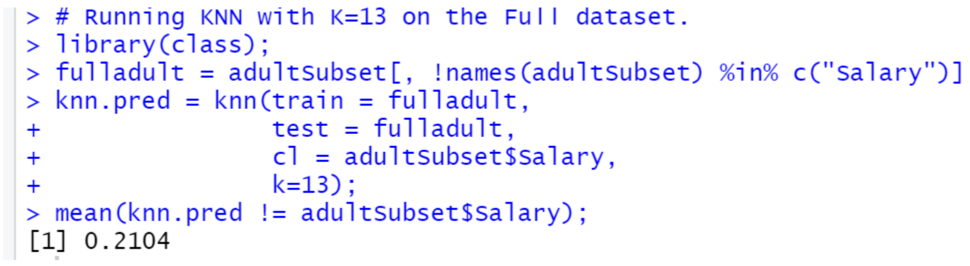


```
> #Finding the optimal degree
> set.seed(1)
> tune.poly <- tune(method=svm, trainSalary~.,
+                   data=newadult.train,
+                   kernel="polynomial",
+                   ranges=list(cost=c(0.001,0.01,0.1,1,5,10,100),
+                               degree=c(2,3,4,5,6,7,8)))
> summary(tune.poly)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost degree
  100      2

- best performance: 0.1577538
```

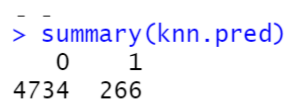
The polynomial kernel narrowly outperformed the linear kernel by 0.1% and thus was our best performing SVM model.

<u>Final Implementation with best model (Shaheer Khan)</u>

Our best SVM model had a test error of 15.78% whereas our best KNN model with cross had a test error of 8.03%, so our overall best model is KNN with 5-fold cross validation and optimal K=13. Since we selected KNN from our final comparison, we proceeded to fit our best model on the full data set.



```
> # Running KNN with K=13 on the Full dataset.
> library(class);
> fulladult = adultSubset[, !names(adultSubset) %in% c("Salary")]
> knn.pred = knn(train = fulladult,
+                test = fulladult,
+                cl = adultSubset$Salary,
+                k=13);
> mean(knn.pred != adultSubset$Salary);
[1] 0.2104
```



```
> summary(knn.pred)
   0    1
4734  266
```

With our data analysis, we were able to predict with a 79% accuracy, if a person makes above or below the median wage. We found that 266 people out of our sample of 5000 people make at least $50,000 a year.

<u>Conclusion (Daniel Onitilo)</u>

We were able to provide an accurate response to our main research question, using an optimal *K* value of 13. One difficulty we had with our dataset was choosing a subset on which to run our models. Our initial dataset was too large, so we experimented by trying to find an optimal size. We encountered warning messages such as the R program reaching the max iteration limit, when trying to figure out the best model for our analysis.

To improve our data analysis, we cut our dataset down to 5000 observations, which enabled us to implement our models. We also used the one-hot-encoding approach to declutter our data and convert categorical variables from text to binary, and we also scaled our quantitative data to enable our analysis work smoothly and as accurately as possible.

Since our group intends is heading into the workforce starting next year, and hope to work in the field of Data Science, this project was be a good experience for us all. We were able to apply data analysis skills we learned over the semester into a dataset that was relevant to our situation and will accompany us all in the hunt for a good-paying job.

REFERENCES

Fawcett, Amanda. "Data Science in 5 Minutes: What is One Hot Encoding?", *Educative.io*, 11 February 2021. Referenced from: Data Science in 5 Minutes: What is One Hot Encoding? (educative.io)

Wang, Wendy. "Classification, k-Nearest Neighbors." MATH 4323, Department of Mathematics, University of Houston.

Wang, Wendy. "Maximal Margin & Support Vector Classifiers." MATH 4323, Department of Mathematics, University of Houston.