Blood Pressure Status Classification

STOR 565 Group 1 Final Paper

Judy Chao and Seanna Chen

2022-11-30

Abstract

With existing health disparities in the United States, certain groups are more likely to have hypertension, or high blood pressure, than others. In an effort to determine what social and environmental factors may lead to one developing hypertension, this project aims to use various classification methods to determine whether someone would be more or less likely to have normal blood pressure, pre-hypertension, or hypertension. The methods include using a multinomial logistic regression model, K-nearest neighbors, support vector machines, and random forests to find the best fitting model using information that we have about the subject. The dataset the project uses for modeling comes from the National Health and Nutrition Examination Survey, which provides both survey question answers relative to general health and lifestyle, as well as physical health examination results. From the four classification methods, we find that they all performed quite similarly, with random forest classification having the highest accuracy in determining the actual classes of each observation, although all the methods have a tendency to classify the observations as normal blood pressure. We also find that the most telling predictors of hypertension include subjects who are older in age and have a lower income. This outcome aligns with well-established understandings in the health field that those who experience health disparities are often older and lower in income.

1 Introduction

Roughly half of the population in the United States is affected by hypertension, or most commonly known as high blood pressure. Although hypertension is regarded as a commonly known health issue, few know that hypertension is commonly linked to higher risks of developing heart disease, heart attacks, and strokes. Because of existing health disparities amongst different groups of people in the United States, hypertension is also a health issue that disproportionately affects some more than others, whether it be through compounding social factors that predisposes one to have the quality of life they have, or through other factors. Our goal in this project is to use demographic and health behavior characteristics to predict blood pressure status and seek confirmatory analysis of previously observed trends regarding hypertension amongst people living in the United States.

2 Data Overview

The dataset used for this project was retrieved from Kaggle, and is from the National Health and Nutrition Examination Survey, or NHANES, during 2013 and 2014. NHANES is conducted by the Center of Disease Control every year, and collects data from physical examinations and survey questions to help answer questions on how national institutions can best serve the emerging needs of the United States population. We chose the survey results from 2013 and 2014 because the data was more readily available in comparison to other years. From Kaggle, there were six datasets containing 10,175 observations with information from each person's survey questions pertaining to demographic, socioeconomic, dietary, and other health-related information, and also examination results containing medical, dental, and physiological measurements, as well as laboratory test results.

From the 1,824 variables in six datasets, we narrowed down the variables used to:

- `gender` : the gender of the subject
- `age` : the age of the subject
- `householdnum` : the number of people living in their household
- `educationlvl` : the highest level of education the subject received
- `citizen` : the United States citizenship status of the subject
- `yrsinUS` : the number of years the subject has lived in the United States
- `householdinc` : the household income of the subject
- `povratio` : the ratio of the family income to the poverty income level in their area
- `foodhp`: the number of meals the subject had that were not home prepared in the past seven days
- `ready2meals` : the number of "ready-to-eat" meals the subject had in the past thirty days
- `frznmeals` : the number of frozen prepared meals the subject had in the past 30 days
- `foodinsecure` : whether the subject had ever worried about running out of food
- `self-ratedhealth` : the subject's self-rating of how well they consider their health condition to be
- `RACE_MexAm`, `RACE_OtherHisp`, `RACE_White`, `RACE_Black`, `RACE_Asian`: if the subject does or does not identify as the racial group Mexican American, Other Hispanic, White, Black, and Asian, respectively
- `MARITAL_Married`, `MARITAL_Widowed`, `MARITAL_Divorced`, `MARITAL_Separated`, `MARTIAL_NeverMarried`: if the subject does or does not identify as the marital status married, widowed, divorced, separated, and never married, respectively.
- `USborn` : whether the subject was born in the United States or not
- `insured` : whether the subject was covered by health insurance
- `hypertension` : the blood pressure status of the subject, containing classes "Normal", "Pre-Hypertensive", and "Hypertensive"

The variable `hypertensive` was produced from the variables in the original dataset `BPXSY2` and `BPXDI2`, which contained the second systolic and diastolic blood pressure readings out four readings taken from each subject. This variable is our response variable for the models in this project.

3 Data Pre-Processing and Feature Engineering

Response Variable

To begin, we transformed the systolic and diastolic blood pressure readings to our response variable, `hypertension`, with three classes: `Normal`, `Pre-Hypertensive`, and `Hypertensive`. As per the American Heart Association, normal blood pressure is defined as a systolic BP below 120 mmHg and diastolic BP below 80 mmHg, elevated/pre-hypertensive blood pressure is defined as a systolic BP between 120 and 129 mmHg and diastolic BP between 80 and 89 mmHg, and high/hypertensive blood pressure is defined as a systolic BP at or above 130 mmHg and diastolic BP at or above 90 mmHg.

Observations Missing the Response Variable

With the response variable in order, we dropped any observations that were missing, which indicated that the participant did not have their blood pressure read. This left us with 7,170 remaining observations, out of 10,175 total recorded observations. This is likely a result of nonresponse bias, which occurs when participants do not complete the entirety of a survey. One important consequence to consider is that certain members of the population may be more or less likely prone to nonresponse bias, which could then affect the distribution of the observations we have kept. Nonetheless, we continued with our analysis.

Missing Data Imputation

The next step to complete was to impute any missing data. Variables such as `age`, `gender`, and `race` did not have any missing values, and therefore were left untouched. The two variables with the greatest proportion of missing values were `marital` status and `self-ratedhealth`, each with 27.28% and 15.54% missing values respectively. The remaining features had at most 2.58% of values missing, which were of less concern. We used a mixture of mean, median, and mode imputation techniques for these missing values. We also considered scenarios in which missing data may indicate "bad news", in which we may want to impute with worst-case values, or in some other situations, missing data may indicate "good news", for which we would want to impute with best-case scenario values. Missing marital status was imputed with "not married," missing birth country was imputed with "not born in the US," missing education level was imputed with "completed high school," missing citizenship status was imputed with "noncitizen," missing years of residency in the US was imputed with `age`, missing household income and family-income-to-poverty-line ratio was imputed with the median, missing number of meals not home prepared in the past month was also imputed with the median, missing number of ready-to-eat meals and number of frozen meals consumed in the past week was imputed with the mean, missing food insecurity status was imputed with "never food insecure," missing health insurance coverage status was imputed with "uninsured," and missing self-rated physical health was imputed with "fair." Finally, our data has no missing values.

One-Hot Encoding for Categorical Variables

Since some of the candidate methods we consider later in this report are not able to handle categorical variables in their original state, we transformed the categorical variables into one-hot encoded features. The two variables that required this transformation were marital status and race. Marital status consisted of seven options: married, widowed, divorced, separated, never married, and living with partner. This translated into six one-hot encoded binary vectors that together represent marital status. Likewise, race/ethnicity included six options: Mexican American, Other Hispanic, Non-Hispanic White, Non-Hispanic Black, Non-Hispanic Asian, and Other Race - Including Multi-Racial, which translated into five one-hot encoded features. The binary categorical variables such as gender and citizenship status were left in their original state since they were already encoded as 0 or 1. Lastly, other categorical variables such as education level and self-rated physical health are ordinal variables, with clear ordering, and as such, those were also left in their original states.

Min-Max Scaling

Finally, since the one-hot encoding resulted in several feature vectors of zeros and ones, along with some already binary variables, we believed it was best to perform some form of scaling to avoid any problems that may arise due to unequal scaling for candidate methods that rely on distance metrics. For example, the variable `age` had a maximum value of 80 years old, which would not be treated the same as one of the other binary variables had we not performed scaling. To stay consistent within the 0 to 1 range, we utilized min-max scaling to address this issue.

4 Exploratory Data Analysis

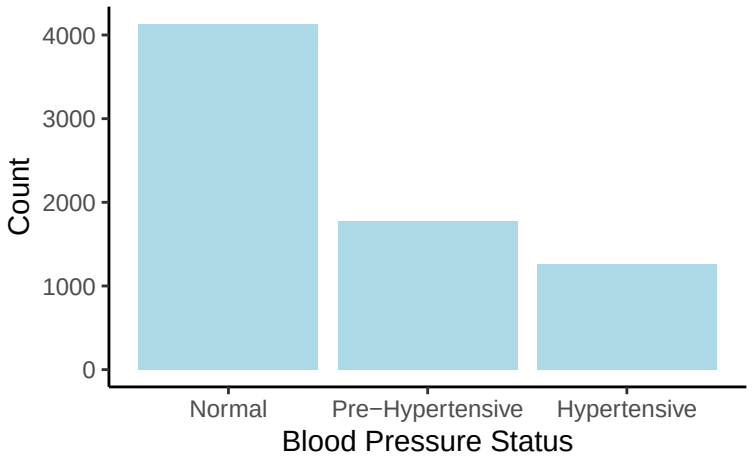
We gained some insights on our data before continuing to the modeling stages through exploratory data analysis.

Question 1: How do systolic and diastolic blood pressure relate?

sSince systolic and diastolic blood pressure were our raw response variables, we wished to look into the relationship between the two readings. As expected, there is an upwards trend. On the plot below, the solid lines represent the normal cutoff values and the dashed lines represent the elevated BP cutoff values. We observed that there are some outliers to note. For one, there is an observation with a systolic blood pressure of 230 mmHg, which is very high and indicates hypertensive crisis. This observation turned out to be an 80+ year old white, married, U.S. citizen male with some college education and health insurance. Other outliers include those with very low diastolic blood pressure. The observations with a diastolic BP less than or equal to 15 mmHg were typically young (between 8 and 16 years old), male, U.S. citizens, had health insurance, and reported their self-rated health as good or poor. The low readings for multiple participants could be attributed to faulty equipment or poorly fitting blood pressure cuffs, which are necessary for accurate readings.

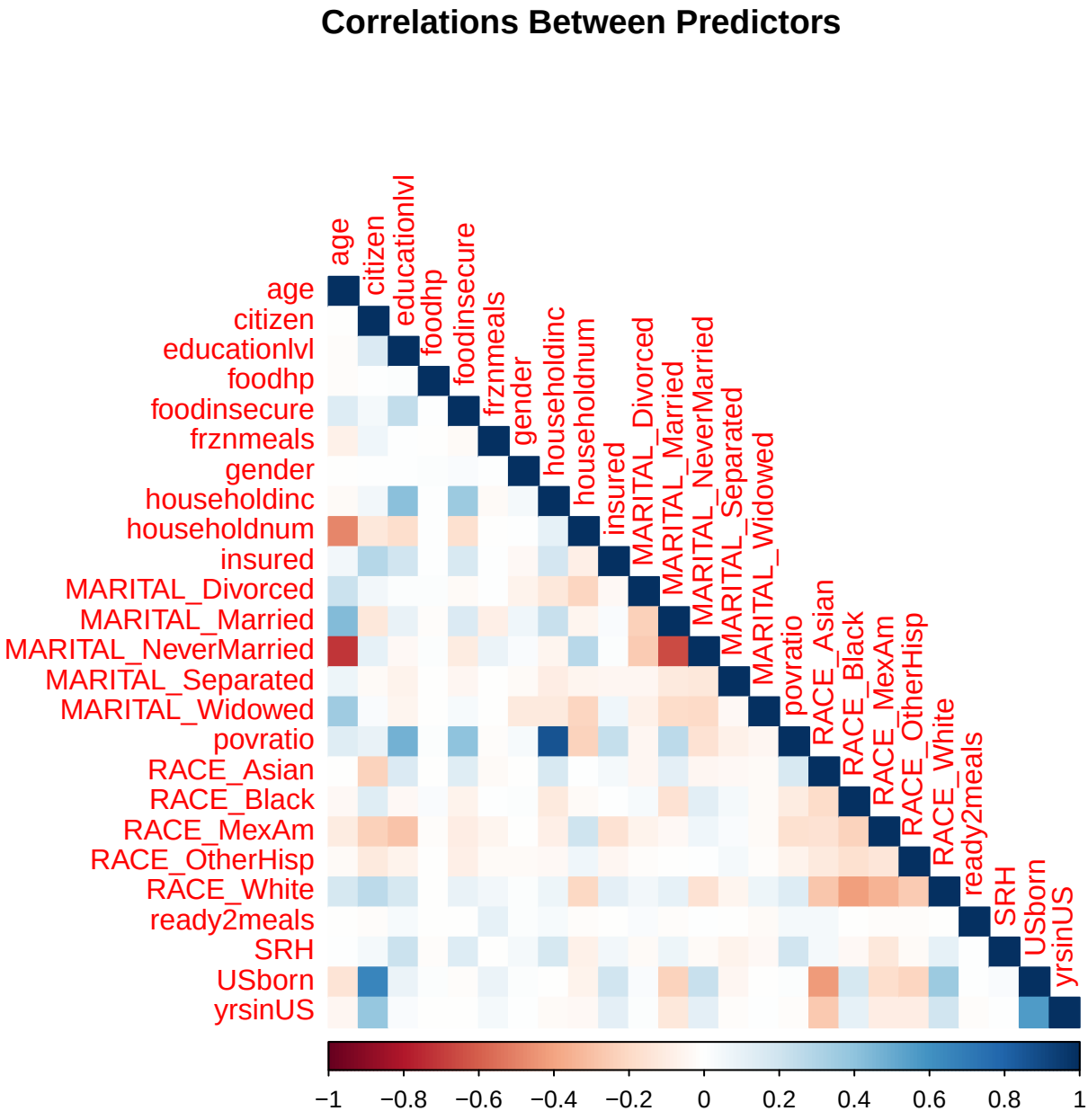
Question 2: Are the classes balanced?

The next common question to ask for classification problems is to see if the classes are balanced. As stated earlier, almost half of the United States population suffers from high blood pressure. Again, the National Health and Nutrition Examination Survey is a nationally representative, so what we see in the plot below roughly reflects the distribution we see in the general population. Out of 7,170 observations 58% had normal blood pressure, 25% were pre-hypertensive, and 17% were hypertensive. There is some unbalancedness to this, so we consider weighting for some of our modeling procedures.

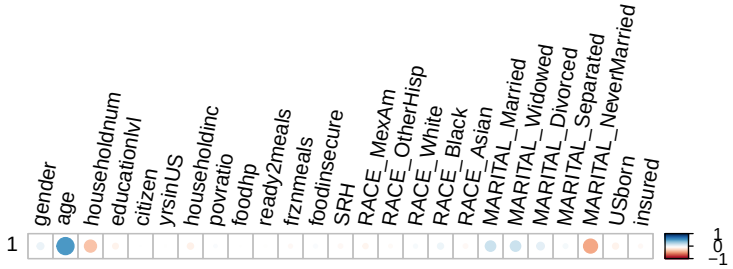


Question 3: What do the correlations between predictors and with systolic and diastolic blood pressure look like?

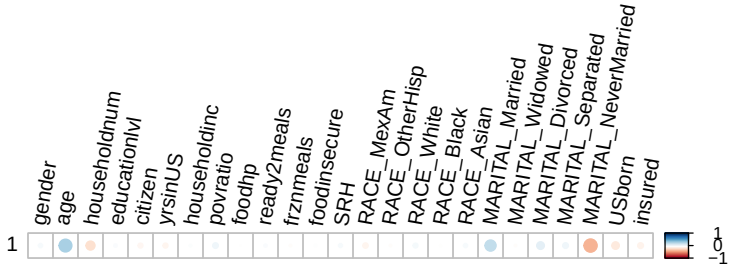
Looking at the correlations between variables, we see that age is negatively correlated with being married. This is expected, since young people are typically not married. Other associations include that asian participants are less likely to be U.S. born, more highly educated individuals have higher income, and younger participants have larger household sizes. When looking at the correlations between the predictors and blood pressure, both systolic and diastolic BP are positively correlated with age and negatively associated with a marital status of never married.



Correlation between Systolic BP and other variables

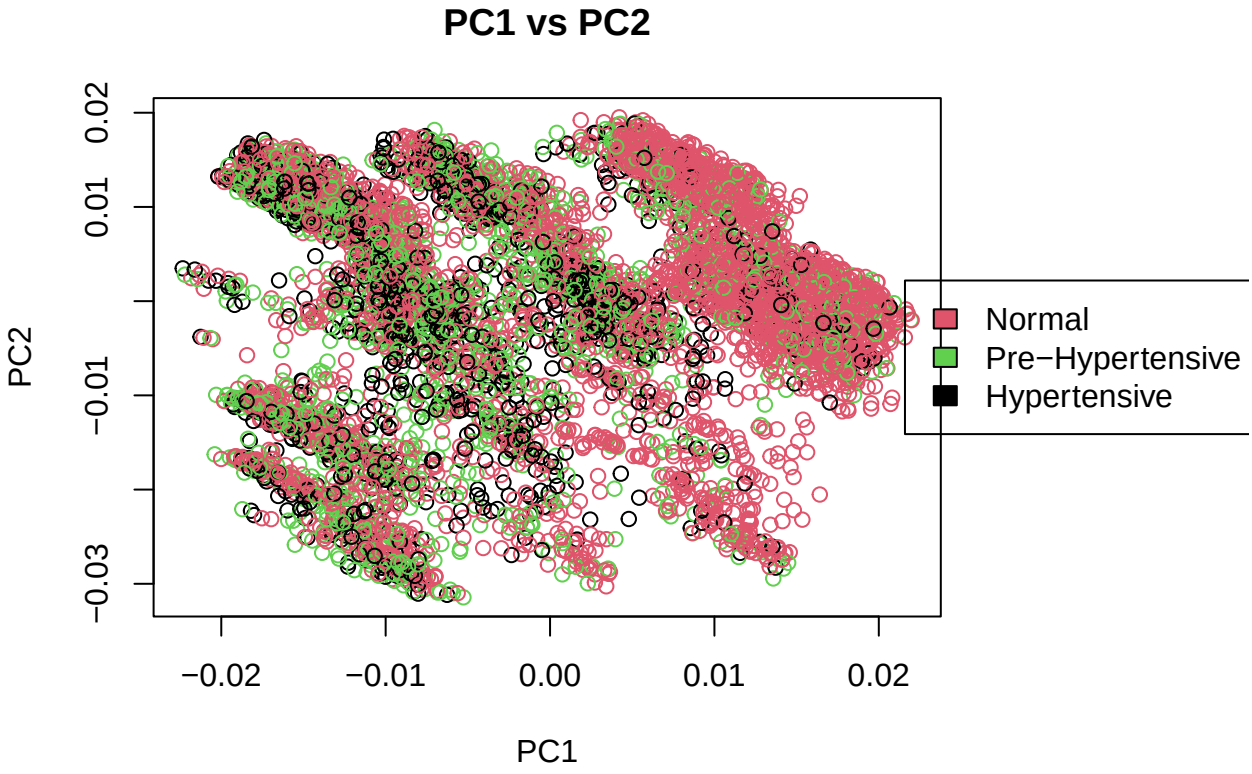


Correlation between Diastolic BP and other variables



Question 4: What does the data look like when visualized using PCA?

Lastly, to visualize any strong patterns in our dataset, we utilized Principal Components Analysis (PCA). PCA is a dimension reduction technique that allows us to easily capture the variability in the data. Here, we plot the first two principal components against each other. We see that there is a lot of overlapping of the three classes. However, there is a greater concentration of normal observations in the upper right corner.



5 Candidate Modeling Methods

With the exploratory data analysis complete, we chose to use four methods to determine the best classification model for blood pressure status classification. We aimed to use multinomial logistic regression, K-nearest neighbors, support vector machines (SVM), and random forest. For training and testing purposes, we used an 80/20 train-test split.

Multinomial Logistic Regression

Because there were three classes of blood pressure status within the dataset, we chose to do multinomial logistic regression, as logistic regression would be ideally used when there are only two classes. We first fit the multinomial logistic regression model using `multimod` from the package "nnet" with all the covariates. However, because the dataset is relatively unbalanced between the three classes, we also fit a multinomial logistic regression model with weights of 55%, 25% and 20% to "Normal", "Pre-Hypertensive" and "Hypertensive" respectively to account for the unbalancedness. In the first model with the unweighted fit, our test accuracy comes to be about 64.5%, and a multiclass AUC of 0.8333, while the second model with the weighted fit gives an accuracy of 59.48%, and a multiclass AUC of 0.5.

K Nearest Neighbors

K-nearest neighbors, or KNN, is a supervised learning method that uses proximity to group items into classifications with the understanding that the closer together items are, the more likely that they are in the same class. The parameter to be decided for this model was K, which is the number of nearest neighbors to include in the considering of a classification of a group. To decide what K would be optimal for our case, we started with cross validation, first with a resampling method through `trainControl()` from the `caret` package, and then by using `train()` to find the highest accuracy for each case of K. However, we ultimately found that as K increases, the accuracy converges to 57.4%, which was also shown from the accuracy vs. K-value plot that was made. We settled on a K-value of 17 in an effort to not sacrifice either too much bias or variance in the model, and because it is an odd value, it will prevent any ties from occurring. After deciding this, we applied 17 to the KNN model with `knn` from the `class` package, and the model yielded a test accuracy of 54.39%, and a multiclass AUC of 0.6667.

Support Vector Machines

Support vector machines are a supervised machine learning method based on the idea of maximum margin separation based on support vectors. A single SVM will do binary classification, so for our problem, we must extend this idea to do multiclass classification. Two approaches exist to address this type of classification problem. The first is one versus one classification. In this approach, we would construct $\binom{K}{2}$ SVMs where K is the number of classes. Each SVM does pairwise comparisons between two classes. A test observation is classified using each of those classifiers, then is assigned to a predicted class based on majority voting. The other approach is one versus all classification. In this case, K SVMs are constructed, each of which compares one of the K classes against the remaining K-1 classes. A test observation is classified to the class for which the confidence is the highest.

Since the `SVM` function in `R` handles multiclass classification problems using the one versus one approach, that is the implementation we used. After some cross validation procedures to go through the possible linear, radial, and polynomial kernels along with their respective hyperparameters (cost, gamma, and degree), we found that SVM with linear kernel and cost = 1 resulted in the best validation error. When performed on the test set, we obtained a test accuracy of 64.64% and a multiclass AUC of 0.6667. Since the classes are unbalanced, class weights were also tried, but did not yield improved results.

Random Forest

Random Forest is a tree-based method that is known for its property of being able to de-correlate many individual decision trees by selecting a random subset of predictors to construct a single tree. By doing so, random forest avoids issues with overfitting and multicollinearity. The main hyperparameter that needed to be tuned was the number of predictors used to fit each tree, which is the `mtry` argument in the `rf()` command in `R`. Using the `tuneRF()` command in `R`, we were shown that the best number of predictors to use was four. Other common practices include using $\sqrt{p}$ or $\frac{p}{3}$ predictors for each tree. Using `mtry = 4`, our test accuracy was 62.48% and the multiclass AUC was 0.8333. Since random forests are quite computationally complex, these models lack interpretability. Therefore, we use a variable importance plot to visualize each variable's contribution in classifying the data. We found that the most important feature was age, followed by years of residency in the United States, and family-income-to-poverty-line ratio.

6 Results, Conclusion, and Future Work

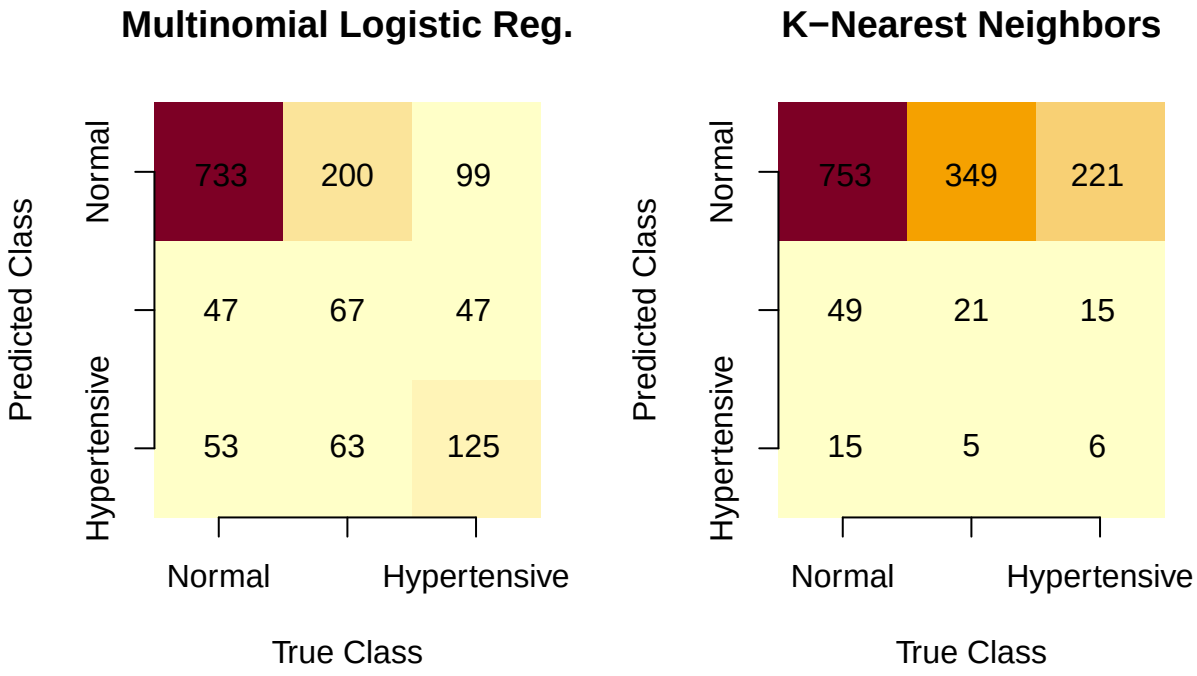
Table 1: **Results for the Four Candidate Models**

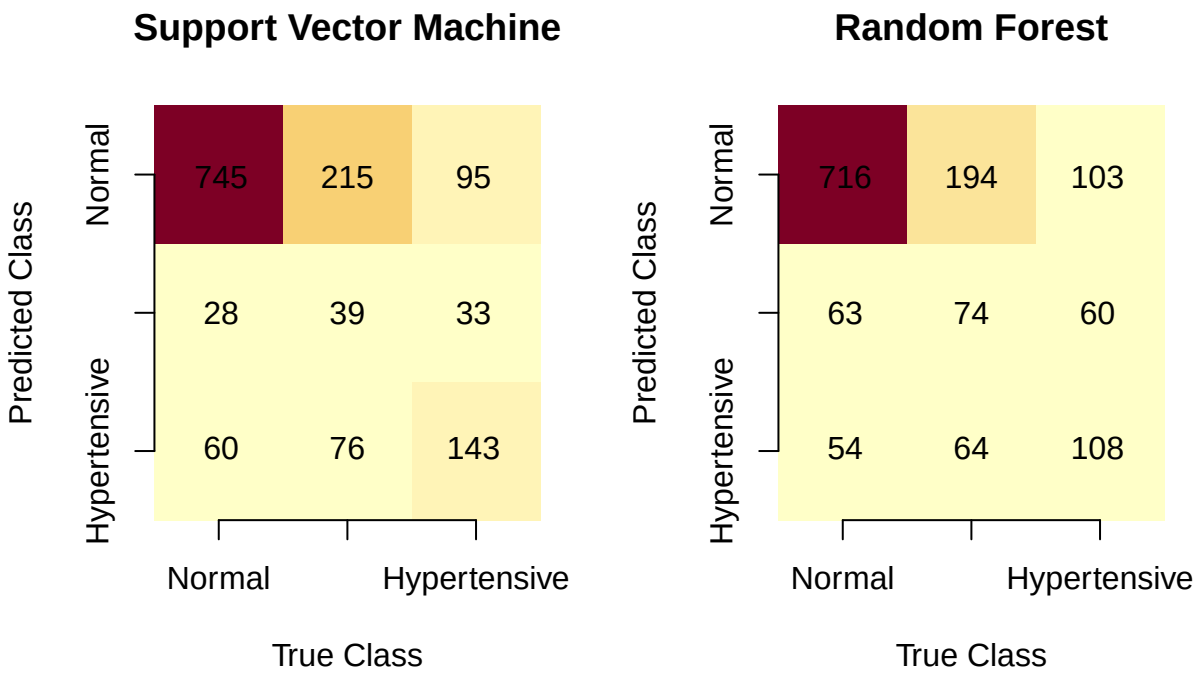
Performance Metrics	Multinomial Logistic Regression	KNN	SVM	Random Forest
Accuracy	64.5%	54.39%	64.64%	62.48%
Multiclass AUC	0.8333	0.6667	0.6667	0.8333

From the performance metrics of each method used, we can see that SVM has the highest test accuracy of 64.64%, but not by a substantial amount in comparison to the other methods. The multinomial logistic regression model follows closely behind with a test accuracy of 64.5%. Both the multinomial logistic regression model and the random forest model share the same multiclass AUC of 0.8333, while KNN and SVM share the same multiclass AUC of 0.6667.

All four methods seemed to have a tendency to classify most of the observations as "Normal", which most likely could have been due to the unbalanced nature of the dataset. A traditional baseline accuracy of a dataset with three classes would be 33% based on random guess, but for unbalanced data, a zero rate classifier, which classifies to the most frequent class, is more accurate to estimate a baseline accuracy. Hence, in the case of our dataset, the baseline accuracy according to the zero rate classifier would be 58%, so the test accuracies presented by the four different methods demonstrate how the accuracies from the models can be mainly attributed to the zero rate classifier.

Figure 1: **Confusion Matrices for Candidate Models**





Conclusion and Future Work

While the results from the models were decent, there were many factors that could have hindered the performance of our models, or perhaps they were never a suitable method to begin with. Multinomial logistic regression would perform more ideally if the variables did not have any kind of multicollinearity. However, with relationships between some predictors present in the model, such as the mentioned relationship between household income and the poverty ratio, or between age and marriage status, multicollinearity cannot be avoided. Therefore, multinomial logistic regression may not have been the best method to use on this dataset, despite how relatively well it performed in comparison to the other methods used. Although we did conduct our research with the random forest, a future opportunity to find a better classification model may include exploring the usage of boosting. Future work that could be done is two instead perform two separate regression problems to predict systolic and diastolic blood pressure first, then to use those two values to classify the observations into a class. Since systolic and diastolic BP are continuous variables and the cutoff values are very close, a small change in reading can easily change the blood pressure class to which an observation is assigned.

We find that the group of people most at risk of hypertension tend to be older and lower-income populations. Modifiable risk factors of hypertension include unhealthy diets containing excessive salt content and little to no vegetable or fruit intake, physical inactivity, drug and alcohol usage, and obesity. People who are typically older in age may get less physical activity as their mobility becomes more limited, and someone with lower income may opt for unhealthier eating options because of limited resources to have healthy foods, such as time, money and other factors. This leads to the rather unsurprising conclusion that one's social and environmental factors can make an impact on how likely they are to develop hypertension.

References

Centers for Disease Control and Prevention. (2017, January 26). National Health and Nutrition Examination Survey. Kaggle. Retrieved November 30, 2022, from https://www.kaggle.com/datasets/cdc/national-health-and-nutrition-examination-survey?sele ctdemographic.csv

Centers for Disease Control and Prevention. (n.d.). NHANES questionnaires, datasets, and related documentation. Centers for Disease Control and Prevention. Retrieved November 30, 2022, from https://wwwn.cdc.gov/nchs/nhanes/continuousnhanes/default.aspx?BeginYear=2013

World Health Organization. (n.d.). Hypertension. World Health Organization. Retrieved November 30, 2022, from https://www.who.int/news-room/fact-sheets/detail/hypertension#:~:text=What%20are%20the%20risk%20factors,and%20being%20overweight%20or%20obese.

Code Appendix

Data Cleaning

```
library(tidyverse)
library(RColorBrewer)
library(corrplot)
library(e1071)
library(gbm)
library(randomForest)
library(tree)
library(caret)
library(pROC)
library(vip)
library(varImp)
library(nnet)
library(class)
```

```
# loading all the data sets
demographic = read.csv("demographic.csv", header= T, sep = ",", check.names=FALSE)
diet = read.csv("diet.csv", header= T, sep = ",", check.names=FALSE)
examination = read.csv("examination.csv", header= T, sep = ",", check.names=FALSE)
labs = read.csv("labs.csv", header= T, sep = ",", check.names=FALSE)
medications = read.csv("medications.csv", header= T, sep = ",", check.names=FALSE)
questionnaire = read.csv("questionnaire.csv", header= T, sep = ",", check.names=FALSE)
```

```
# combine datasets
df_list = list(demographic, diet, examination, labs, questionnaire)

df = Reduce(function(x, y) merge(x, y, all=TRUE), df_list)
```

```
# create response variable, hypertension

df = df[df$BPXDI2 != 0, ]
df$hypertension = as.factor(ifelse(df$BPXSY2 < 120 & df$BPXDI2 <80, "Normal",
                          ifelse((df$BPXSY2 >= 120 & df$BPXSY2 <= 129) |
                                 (df$BPXDI2 >= 80 & df$BPXDI2 <= 89), "Pre-Hypertensive",
                                 ifelse((df$BPXSY2 >= 130) | (df$BPXDI2 >=90),
```

```
                                        "Hypertensive", "NA"))))

# drop any rows that are missing hypertension
df = df %>% drop_na(hypertension)
df = df[df$hypertension!="NA",]

df$hypertension = factor(df$hypertension, levels = c("Normal", "Pre-Hypertensive",
"Hypertensive"))

# remove other columns related to blood pressure/other blood pressure readings
df = df[, -which(names(df) %in% c("BPACSZ", "BPXDI1", "BPXDI3", "BPXDI4",
             "BPXML1", "BPXPLS", "BPXSY1", "BPXSY3",
             "BPXSY4", "PEASCST1"))]
```

```
# keep only pre-selected variables

df = df[, c("SEQN", "RIAGENDR", "RIDAGEYR", "RIDRETH3", "DMDMARTL", "DMDHHSIZ",
            "DMDBORN4",            "DMDHREDU", "DMDCITZN", "DMDYRSUS",
            "INDHHIN2", "INDFMPIR", "DBD895", "DBD905",
            "DBD910", "FSD032A", "HIQ011", "HSD010", "hypertension")]

# rename variables for easier workability
names(df) <- c("id", "gender", "age", "race", "marital", "householdnum", "birthcountry",
               "educationlvl", "citizen", "yrsinUS", "householdinc", "povratio", "foodhp",
               "ready2meals", "frznmeals", "foodinsecure", "healthinsurance",
               "self-ratedhealth", "hypertension")
```

```
# re-assign "refused" and "don't know" responses to NA

# if 77/99/7777/9999 then change to NA
df[df == 77 | df == 99 | df == 7777 | df == 9999] = NA

# put back all ages
df$age = age

# if 7 or 9 in columns  , then change to NA
for (col in c('educationlvl', 'citizen', 'healthinsurance',
              'self-ratedhealth','foodinsecure')) {
  df[, col] = ifelse(((df[, col] == 7) | (df[, col] == 9)), NA, df[,col])
  }
```

```
# how many observations are missing more than 20% of the responses?

sum(rowMeans(is.na(df)) > 0.20) # only 107, so lets remove those
df = df[rowMeans(is.na(df)) <= 0.20, ]

# Check prop of rows/columns that are missing data
colMeans(is.na(df))
```

“

```r
# missing data imputation

# not married if NA
df$marital = ifelse(is.na(df$marital), 5, df$marital)

# not born in US
df$birthcountry = ifelse(is.na(df$birthcountry), 2, df$birthcountry)

# HS education if NA
df$educationlvl = ifelse(is.na(df$educationlvl), 3, df$educationlvl)

# not a citizen if NA
df$citizen = ifelse(is.na(df$citizen), 2, df$citizen)

# replace with age
df$yrsinUS = ifelse(is.na(df$yrsinUS), df$age, df$yrsinUS)

# median imputation
df$householdinc = ifelse(is.na(df$householdinc), median(df$householdinc, na.rm=TRUE),
                         df$householdinc)

# median imputation
df$povratio = ifelse(is.na(df$povratio), median(df$povratio, na.rm=TRUE), df$povratio)

# median imputation
df$foodhp = ifelse(is.na(df$foodhp), median(df$foodhp, na.rm=TRUE), df$foodhp)

# mean imputation
df$ready2meals = ifelse(is.na(df$ready2meals), mean(df$ready2meals, na.rm=TRUE),
                        df$ready2meals)

# mean imputation
df$frznmeals = ifelse(is.na(df$frznmeals), mean(df$frznmeals, na.rm=TRUE), df$frznmeals)

# sometimes true if NA
df$foodinsecure = ifelse(is.na(df$foodinsecure), 2, df$foodinsecure)

# uninsured if NA
df$healthinsurance = ifelse(is.na(df$healthinsurance), 2, df$healthinsurance)

# fair SRH if NA
df$`self-ratedhealth` = ifelse(is.na(df$`self-ratedhealth`), 4, df$`self-ratedhealth`)


# Check again
colMeans(is.na(df)) # yay

# saving dataframe before one hot encoding for EDA purposes
write.csv(df, "C:/Users/judyc/OneDrive - University of North Carolina at Chapel
          Hill/STOR 565/Project/data_before_OHE.csv", row.names = TRUE)
```

```r
# do one-hot encoding for categorical (gender, race, marital status,
#country of birth, citizenship status, health insurance)
```

```r
# Gender 1 = Male, Female otherwise
df$gender = ifelse(df$gender==2, 0, 1)

# Race: there are 6 categories, so 0 for all means other race/multi
# (there's no == 5 for some reason)
df$RACE_MexAm = ifelse(df$race == 1, 1, 0)
df$RACE_OtherHisp = ifelse(df$race == 2, 1, 0)
df$RACE_White = ifelse(df$race == 3, 1, 0)
df$RACE_Black = ifelse(df$race == 4, 1, 0)
df$RACE_Asian = ifelse(df$race == 6, 1, 0)

# Marital Status: 6 categories, 0 for all means living with partner
df$MARITAL_Married = ifelse(df$marital == 1, 1, 0)
df$MARITAL_Widowed = ifelse(df$marital == 2, 1, 0)
df$MARITAL_Divorced = ifelse(df$marital == 3, 1, 0)
df$MARITAL_Separated = ifelse(df$marital == 4, 1, 0)
df$MARITAL_NeverMarried = ifelse(df$marital == 5, 1, 0)


# Country of Birth: 1 = US, not US otherwise
df$USborn = ifelse(df$birthcountry==1, 1, 0)

# Citizenship status: 1 = yes, no otherwise
df$citizen = ifelse(df$citizen==2, 0, 1)

# Health insurance: 1 = covered, 0 otherwise
df$insured = ifelse(df$healthinsurance==1, 1, 0)

# reverse the scale for self-rated health so 0 is lowest, 6 is highest
df$`self-ratedhealth` = 7 - df$`self-ratedhealth`

# delete the columns we no longer need
df = df[, -which(names(df) %in% c("race", "marital", "birthcountry",
                                  "healthinsurance"))]
```

```r
# scaling/standardizing
library(caret)

# min-max scaling (all values between 0 and 1)
process = preProcess(df[, -c(1, 27)], method=c("range"))
norm_scale = predict(process, df[, -c(1, 27)])

df[, -c(1, 27)] = norm_scale

# move hypertension to be last column again
df = df %>% select(-hypertension, hypertension)
```

```r
# save
write.csv(df, "C:/Users/judyc/OneDrive - University of North Carolina at Chapel Hill/\
          STOR 565/Project/data_clean.csv", row.names = TRUE)
```

Exploratory Data Analysis

```
pre_data = read.csv("data_before_OHE.csv", header= T, sep = ",", check.names=FALSE)
df = read.csv("data_clean.csv", header= T, sep = ",", check.names=FALSE)
examination = read.csv("examination.csv", header= T, sep = ",", check.names=FALSE)

df$hypertension = factor(df$hypertension, levels = c("Normal", "Pre-Hypertensive",
                                                     "Hypertensive"))

pre_data = pre_data[,-1]
df = df[,-1]

df  = rename(df, "SRH" = "self-ratedhealth")
pre_data = rename(pre_data, "SRH" = "self-ratedhealth")

exam = examination[, c("SEQN", "BPXSY2", "BPXDI2")]
exam = exam[exam$BPXDI2!=0,] %>% drop_na(BPXSY2)

BPdf = merge(pre_data, exam, by.x = "id", by.y = "SEQN")
BPdf2 = merge(df, exam, by.x = "id", by.y = "SEQN")
```

```
# What does the distribution of systolic and diastolic blood pressures look like?
ggplot(BPdf, aes(BPXSY2, BPXDI2)) + geom_point(aes(color=hypertension)) +
  xlab("Systolic Blood Pressure") +
  ylab("Diastolic Blood Pressure") + geom_hline(yintercept = 80,
                                                linetype="solid", col = "red", size = 1) +
  geom_hline(yintercept = 90, linetype="dashed", col = "red", size = 1) +
  geom_vline(xintercept = 120, linetype="solid", col = "blue", size = 1) +
  geom_vline(xintercept = 130, linetype="dashed", col = "blue", size = 1)

# What are the outliers? (diastolic BP less than 15 or systolic BP greater than 225)
BPdf[(BPdf$BPXDI2 <=15) | (BPdf$BPXSY2 >= 225),]
```

```
# checking if classes are balanced
nrow(df[df$hypertension=="Normal",])
nrow(df[df$hypertension=="Pre-Hypertensive",])
nrow(df[df$hypertension=="Hypertensive",])

# plot
ggplot(df, aes(hypertension)) + geom_bar(fill="lightblue") + xlab("Blood Pressure Status") +
  ylab("Count") +
  theme_classic()
```

```
# what are the correlations between the variables?

# after one hot encoding
B = cor(as.matrix(df[, -c(1, 27)]))
corrplot(B, method = 'color', order = 'alphabet', type="lower")


# what are the correlations with systolic and diastolic blood pressure?
```

```
Acor = cor(x = BPdf2$BPXSY2, y = BPdf2[2:26])
corrplot(Acor, tl.srt = 80, tl.col = "black", cl.length=3,
         title = "Correlation between Systolic BP and other variables",
         mar = c(5, 10, 10, 10))

Bcor = cor(x = BPdf2$BPXDI2, y = BPdf2[2:26])
corrplot(Bcor, tl.srt = 80, tl.col="black", cl.length=3,
         title = "Correlation between Diastolic BP and other variables",
         mar = c(5, 10, 10, 10))
```

```
# PCA visualization

BP_dat = t(df[, c(2:26)])
colnames(BP_dat) = df$hypertension

dim(BP_dat)
unique(colnames(BP_dat))

# PCA - take SVD to get solution
X = t(scale(t(BP_dat), center = TRUE, scale=FALSE))
sv = svd(t(X))
U = sv$u
V = sv$v
D = sv$d
Z = t(X)%*%V


# PC scatter plots colored by true labels
par(mar=c(5, 4, 4, 8), xpd=TRUE)

cols = as.numeric(as.factor(colnames(BP_dat)))

K = 3
pclabs = c("PC1", "PC2", "PC3", "PC4")
for (i in 1:K) {
  j = i + 1
  plot(U[,i], U[, j], type = "n", xlab = pclabs[i], ylab=pclabs[j])
  points(U[,i], U[, j], col=cols)
  legend(x = "right",inset=c(-0.32, 0), legend = c("Normal", "Pre-Hypertensive",
                                                   "Hypertensive"),
         fill = cols)

}
```

Modeling

```
# splitting data into train and test sets

set.seed(1) # for reproducibility

# roughly 80/20 split
```

```
sample = sample.int(7170, 5736, replace = F)
df_train = df[sample, -1]
df_test = df[-sample, -1]

X_train = df_train[, -26]
y_train = df_train$hypertension

X_test = df_test[, -26]
y_test = df_test$hypertension
```

Multinomial Logistic Regression

```
# Training the multinomial model
multimod1 <- multinom(hypertension ~ ., data = df_train)

# Checking the model
summary(multimod1)

# Predicting the values for train dataset
df_train$multinompred1 <- predict(multimod1, newdata = df_train, "class")

# Building classification table
tab1 <- table(df_train$hypertension, df_train$multinompred1)

# Calculating accuracy - sum of diagonal elements divided by total obs
round((sum(diag(tab1))/sum(tab1))*100,2)

# Predicting the class for test dataset
df_test$multinompred1 <- predict(multimod1, newdata = df_test, "class")
tab2 <- table(df_test$multinompred1, df_test$hypertension)
round((sum(diag(tab2))/sum(tab2))*100,2)

tab1
tab2

#62.64% accuracy for training set, 64.5% accuracy for testing set
```

```
# making model with weights this time

sample = sample.int(7170, 5736, replace = F)
df_train = data_clean[sample, ]
df_test = data_clean[-sample, ]

# Making weights since the data is unbalanced

df_train$wts <- ifelse(df_train$hypertension == "Normal", .55,
                       ifelse(df_train$hypertension == "Pre-Hypertensive", .25, .20))
df_test$wts <- ifelse(df_test$hypertension == "Normal", .55,
                       ifelse(df_test$hypertension == "Pre-Hypertensive", .25, .20))

multimod2 <- multinom(hypertension ~ . -wts, data = df_train, weights = wts)
summary(multimod2)
```

```r
# Predicting the values for train dataset
df_train$multinompred2 <- predict(multimod2, newdata = df_train, "class")

# Building classification table
tab3 <- table(df_train$hypertension, df_train$multinompred2)

# Calculating accuracy - sum of diagonal elements divided by total obs
round((sum(diag(tab3))/sum(tab3))*100,2)

# Predicting the class for test dataset
df_test$multinompred2 <- predict(multimod2, newdata = df_test, "class")
tab4 <- table(df_test$multinompred2, df_test$hypertension)
round((sum(diag(tab4))/sum(tab4))*100,2)

tab3
tab4

#59.78% accuracy for training set, 59.48% accuracy for testing set
```

```r
#roc for test multinom without weights
real_values = matrix(c("Normal", "Pre-Hypertensive", "Hypertensive"), nc=1)
pred.mat = matrix(c(125, 99, 47, 53, 733, 47, 63, 200, 67), nc=3)
colnames(pred.mat) <- c("Normal", "Pre-Hypertensive", "Hypertensive")

multiclass.roc(real_values, pred.mat)

#roc for test multinom with weights
pred.mat = matrix(c(36, 188, 11, 19, 803, 10, 18, 335, 14), nc=3)
colnames(pred.mat) <- c("Normal", "Pre-Hypertensive", "Hypertensive")

multiclass.roc(real_values, pred.mat)
```

K- Nearest Neighbors

```r
#KNN

sample = sample.int(7170, 5736, replace = F)
df_train = data_clean[sample, ]
df_test = data_clean[-sample, ]

#removing NA values - KNN won't work with NA
knn_train <- na.omit(df_train)
knn_test <- na.omit(df_test)
y_train <- as.factor(knn_train$hypertension)
y_test <- as.factor(knn_test$hypertension)
knn_train <- knn_train[, -27]
knn_test <- knn_test[, -27]


ctrl <- trainControl(method = "cv", number = 10)

set.seed(1)
```

```
#looking at k's
knnmod <- train(y = y_train, x = knn_train, method = "knn", metric = "Accuracy",
                tuneGrid = expand.grid(k = 1:100),trControl = ctrl)
knnmod

#chose K value of 17 for models, since the higher the K value, accuracy converges to
# 0.5754889, and R will always choose the largest K in the range you set.

#finding training accuracy
set.seed(1)
knn.predtrain <- knn(train=knn_train, test=knn_train, cl=y_train, k=17)

knntb1 <- table(knn.predtrain, y_train)
knntb1

#finding testing accuracy
set.seed(1)
knn.predtest <- knn(train=knn_train, test=knn_test, cl=y_train, k=17)

knntb2 <- table(knn.predtest, y_test)
knntb2

round((sum(diag(knntb1))/sum(knntb1))*100,2)
round((sum(diag(knntb2))/sum(knntb2))*100,2)

#KNN with k = 17 gives training accuracy of 58.04, and testing accuracy of 55.23
```

```
#roc for test KNN
real_values = matrix(c("Normal", "Pre-Hypertensive", "Hypertensive"), nc=1)
pred.mat = matrix(c(6, 15, 5, 221, 753, 349, 15, 49, 21), nc=3)
colnames(pred.mat) <- c("Normal", "Pre-Hypertensive", "Hypertensive")

multiclass.roc(real_values, pred.mat)
```

Support Vector Machine

```
set.seed(1)


# Cross Validation Procedures: kernel, cost, gamma

# tuning linear kernel
tune.out.linear = tune(svm, hypertension~., data = df_train, kernel="linear",
                       ranges = list(
                         cost = c(0.01, 0.1, 1, 5, 10)))
summary(tune.out.linear)

bestlinearmod = tune.out.linear$best.model
summary(bestlinearmod)


# tuning radial kernel
tune.out.radial = tune(svm , hypertension ~ ., data = df_train, kernel = "radial",
```

```
                  ranges = list (
                    cost = c(0.1, 1, 5),
                    gamma = c(0.5, 1, 2)
                                ))

summary(tune.out.radial)
bestradialmod = tune.out.radial$best.model
summary(bestradialmod)


# tuning polynomial kernel
tune.out.poly = tune(svm , hypertension ~ ., data = df_train,
                        kernel = "polynomial",
                  ranges = list (
                    cost = c(0.1, 1, 5),
                    degree = c(2, 3, 4)
                                ))

summary(tune.out.poly)
bestpolymod = tune.out.poly$best.model
summary(bestpolymod)
```

```
# linear kernel with cost 1 was best
real_values = matrix(c("Normal", "Pre-Hypertensive", "Hypertensive"), nc=1)

svm.pred.linear =  predict(bestlinearmod, X_test)
table(pred = svm.pred.linear, true = y_test)
confusionMatrix(y_test, svm.pred.linear)

linear.pred.mat = matrix(c(745, 215, 95, 28, 39, 33, 60, 76, 143), byrow=TRUE, nc=3)
colnames(linear.pred.mat) <- c("Normal", "Pre-Hypertensive", "Hypertensive")
multiclass.roc(real_values, linear.pred.mat)
```

Random Forest Classifier

```
set.seed(1)

rf5 = randomForest(hypertension~., data = df_train, mtry = 5, importance = TRUE)
rf4 = randomForest(hypertension~., data = df_train, mtry = 4, importance = TRUE)
rf8 = randomForest(hypertension~., data = df_train, mtry = 8, importance = TRUE)
rf2 = randomForest(hypertension~., data = df_train, mtry = 2, importance = TRUE)

rf4.preds = predict(rf4, df_test, type="response")
rf5.preds = predict(rf5, df_test, type="response")
rf8.preds = predict(rf8, df_test, type="response")
rf2.preds = predict(rf2, df_test, type="response")

sum(rf2.preds == y_test)/1434
sum(rf4.preds == y_test)/1434
sum(rf5.preds == y_test)/1434
sum(rf8.preds == y_test)/1434

table(pred = rf2.preds, true = y_test)
```

```
table(pred = rf4.preds, true = y_test)
table(pred = rf5.preds, true = y_test)
table(pred = rf8.preds, true = y_test)

# CV for choosing mtry
set.seed(1)
tuneRF(df_train[,-26], df_train$hypertension, mtryStart = 1, stepFactor = 0.9,
       improve = 0.0001, plot = TRUE)


varImpPlot(rf4, type = 2, main = "Variable Importance for 4 Predictors")
varImpPlot(rf5, type = 2, main = "Variable Importance for 5 Predictors")
varImpPlot(rf8, type = 2,main = "Variable Importance for 8 Predictors")
```

```
real_values = matrix(c("Normal", "Pre-Hypertensive", "Hypertensive"), nc=1)

table(pred = rf4.preds, true = y_test)

pred.mat.rf = matrix(c(716, 194, 103, 63, 72, 60, 54, 64, 108), byrow=TRUE, nc=3)

colnames(pred.mat.rf) <- c("Normal", "Pre-Hypertensive", "Hypertensive")

multiclass.roc(real_values, pred.mat.rf)
```

```
# plots for confusion matrices
par(mfrow = c(2, 2))


# Mutlinomial Logistic Reg. confusion matrix

z1 = c(733, 200, 99, 47, 67, 47, 53, 63, 125)

z1 = matrix(z1, ncol=3)
colnames(z1) = c("Hypertensive", "Pre-Hypertensive", "Normal")
rownames(z1) = c("Normal", "Pre-Hypertensive", "Hypertensive")

image(z1[,ncol(z1):1], axes=FALSE)

axis(2, at = seq(0, 1, length=length(colnames(z1))), labels=colnames(z1))
axis(1, at = seq(0, 1, length=length(colnames(z1))), labels=rownames(z1))


text(0.01,0.01,"53")
text(0.01,0.5,"47")
text(0.01,0.99,"733")

text(0.5,0.01,"63")
text(0.5,0.5,"67")
text(0.5,0.99,"200")

text(0.99,0.01,"125")
text(0.99,0.5,"47")
text(0.99,0.99,"99")
```

```
title("Multinomial Logistic Reg.")
mtext('True Class', side = 1, outer = FALSE, line = 3)
mtext('Predicted Class', side = 2, outer = FALSE, line = 3)

# KNN confusion matrix

z2 = c(753, 349, 221, 49, 21, 15, 15, 5, 6)

z2 = matrix(z2, ncol=3)
colnames(z2) = c("Hypertensive", "Pre-Hypertensive", "Normal")
rownames(z2) = c("Normal", "Pre-Hypertensive", "Hypertensive")

image(z2[,ncol(z2):1], axes=FALSE)

axis(2, at = seq(0, 1, length=length(colnames(z2))), labels=colnames(z2))
axis(1, at = seq(0, 1, length=length(colnames(z2))), labels=rownames(z2))


text(0.01,0.01,"15")
text(0.01,0.5,"49")
text(0.01,0.99,"753")

text(0.5,0.01,"5")
text(0.5,0.5,"21")
text(0.5,0.99,"349")

text(0.99,0.01,"6")
text(0.99,0.5,"15")
text(0.99,0.99,"221")

title("K-Nearest Neighbors")
mtext('True Class', side = 1, outer = FALSE, line = 3)
mtext('Predicted Class', side = 2, outer = FALSE, line = 3)


# Support Vector Machine confusion matrix

z = c(745, 28, 60, 215, 39, 76, 95, 33, 143)

z = matrix(z, ncol=3, byrow=TRUE)
colnames(z) = c("Hypertensive", "Pre-Hypertensive", "Normal")
rownames(z) = c("Normal", "Pre-Hypertensive", "Hypertensive")

image(z[,ncol(z):1], axes=FALSE)

axis(2, at = seq(0, 1, length=length(colnames(z))), labels=colnames(z))
axis(1, at = seq(0, 1, length=length(colnames(z))), labels=rownames(z))


text(0.01,0.01,"60")
text(0.01,0.5,"28")
text(0.01,0.99,"745")
```

```
text(0.5,0.01,"76")
text(0.5,0.5,"39")
text(0.5,0.99,"215")

text(0.99,0.01,"143")
text(0.99,0.5,"33")
text(0.99,0.99,"95")

title("Support Vector Machine")
mtext('True Class', side = 1, outer = FALSE, line = 3)
mtext('Predicted Class', side = 2, outer = FALSE, line = 3)

# Random Forest confusion matrix

z1 = c(716, 194, 103, 63, 72, 60, 54, 64, 108)

z1 = matrix(z1, ncol=3)
colnames(z1) = c("Hypertensive", "Pre-Hypertensive", "Normal")
rownames(z1) = c("Normal", "Pre-Hypertensive", "Hypertensive")

image(z1[,ncol(z1):1], axes=FALSE)

axis(2, at = seq(0, 1, length=length(colnames(z))), labels=colnames(z))
axis(1, at = seq(0, 1, length=length(colnames(z))), labels=rownames(z))


text(0.01,0.01,"54")
text(0.01,0.5,"63")
text(0.01,0.99,"716")

text(0.5,0.01,"64")
text(0.5,0.5,"74")
text(0.5,0.99,"194")

text(0.99,0.01,"108")
text(0.99,0.5,"60")
text(0.99,0.99,"103")

title("Random Forest")

mtext('True Class', side = 1, outer = FALSE, line = 3)
mtext('Predicted Class', side = 2, outer = FALSE, line = 3)
```