SS9864 Final Project

Gansen Deng, Chufan Wu

December 18, 2019

1 Introduction

1.1 Data description

Chronic kidney disease, also called chronic kidney failure, describes the gradual loss of kidney function. In our body, kidneys filter wastes and excess fluids from blood, which are then excreted in the urine. But when chronic kidney disease reaches an advanced stage, dangerous levels of fluid, electrolytes and wastes can build up in our body[1]. Therefore, it is important to control the chronic kidney disease before it reaches a dangerous level.

However, in the early stages of chronic kidney disease, we may have few signs or symptoms. Chronic kidney disease may not become apparent until our kidney function is significantly impaired. Therefore, detection and diagnosis of kidney disease in the early stage is very important for preventing the disease progressing to end-stage kidney failure, which is fatal without artificial filtering and kidney transplant.[1]

To detect whether people have chronic kidney disease without apparent symptoms, we need to make use of the information provided by other medical indexes, such as the amount of red blood cells in the body of patients. Based on these explanatory variables, we can build a classification model to predict the probability of a person getting the chronic kidney disease in the early stage. With this purpose, we use the chronic kidney disease data set from the UCI Machine Learning Repository to train our classification models. The data description of our dataset is displayed below:

	Description
age	Age in years(Numerical)
bp	Blood Pressure(Numerical: mm/Hg)
sg	Specific Gravity(Factor: (1.005,1.010,1.015,1.020,1.025))
al	Albumin(Factor: (0,1,2,3,4,5)
su	Sugar(Factor: (0,1,2,3,4,5))
rbc	Red Blood Cells(Factor: (normal,abnormal))
pc	Pus Cell(Factor: (normal,abnormal))
pcc	Pus Cell clumps(Factor((present,notpresent)))
ba	Bacteria(Factor: (present,notpresent))
bgr	Blood Glucose Random(Numerical: mgs/dl)
bu	Blood Urea(Numerical: mgs/dl)
sc	Serum Creatinine(Numerical: mgs/dl)
sod	Sodium(Numerical: mEq/L)
pot	Potassium(Numerical: mEq/L)
hemo	Hemoglobin(Numerical: gms)
pcv	Packed Cell Volume(Numerical)
wc	White Blood Cell Count(Numerical: cells/cumm)
rc	Red Blood Cell Count(Numerical: millions/cmm)
htn	Hypertension(Factor: (yes,no))
dm	Diabetes Mellitus(Factor: (yes,no))
cad	Coronary Artery Disease(Factor: (yes,no))
appet	Appetite(Factor: (good,poor)))
pe	Pedal Edema(Factor: (yes,no))
ane	Anemia(Factor: (yes,no))
class(Response Variable)	Class of chronic kidney disease(Factor: (ckd,notckd))

Table 1: Chronic Kidney Disease Data Set Description

1.2 Project objectives

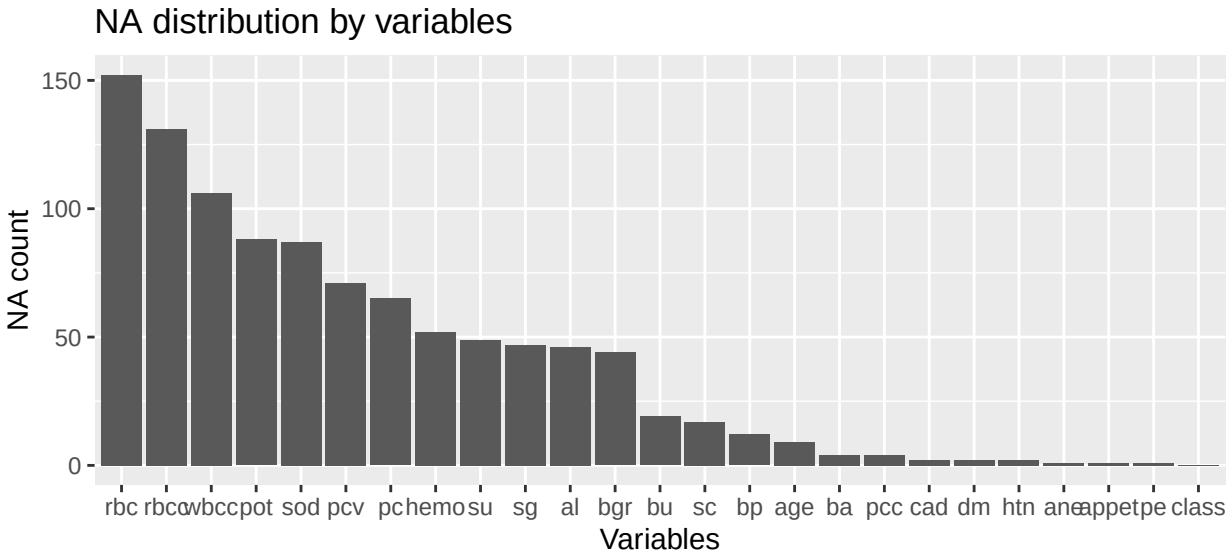
There are two main purposes of the analysis on this dataset. The first one is to build a chronic kidney disease detector to help hospitals diagnose **ckd** for patients. The other one is to find out which covariate has significant relationship with **ckd** and give patients health advice to protect them from **ckd**.

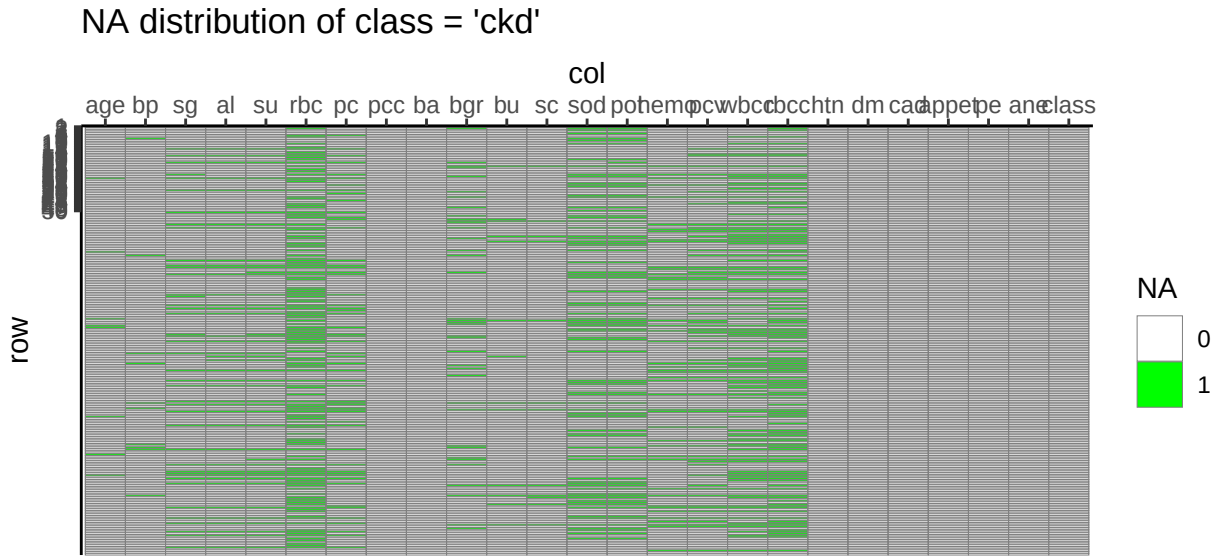
2 Missing values handling

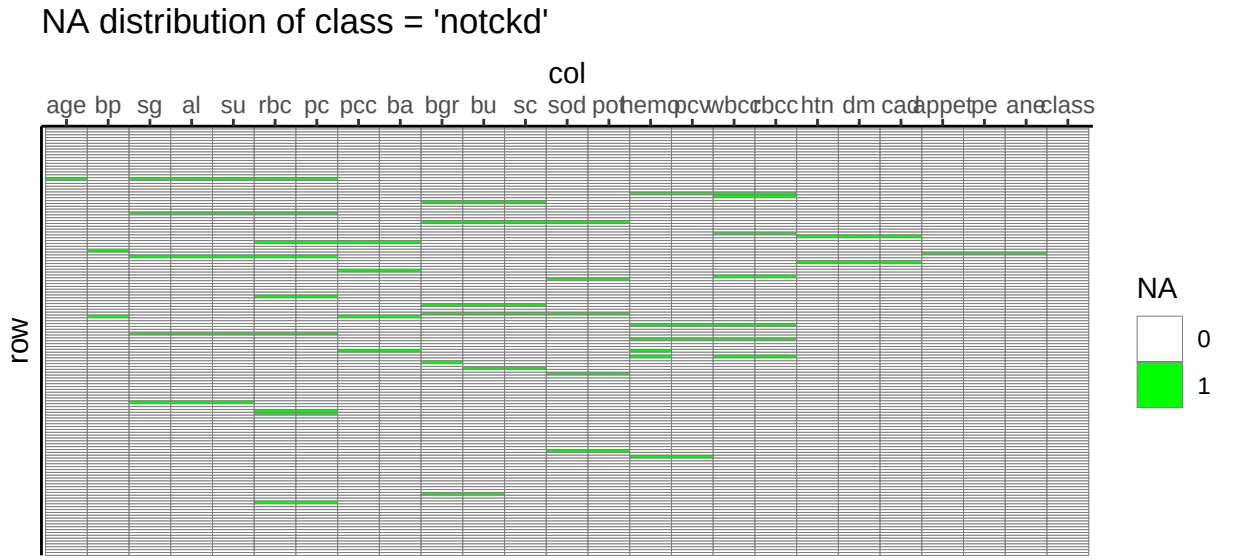
In the Chronic Kidney Disease dataset, there are a large proportion of observations with missing values. If we do the listwise deletion, then there would only be 39.5% (158/400) observations left. Thus, we think it's not appropriate to simply delete all observations with missing values and it's necessary for us to do some imputation in order to extract more information from the dataset.

2.1 The distribution of NA

In order to find an appropriate way to deal with missing values, we first plot the distribution of NA in this dataset. The plots are shown below:


NA distribution by variables
NA count
150
100
50
0
rbc rbcc wbcc pot sod pcv pc hemo su sg al bgr bu sc bp age ba pcc cad dm htn ane appet pe class
Variables


NA distribution of class = 'ckd'
col
age bp sg al su rbc pc pcc ba bgr bu sc sod pot hemo pcv wbcc rbcc htn dm cad appet pe ane class
row
NA
0
1



From the "NA distribution by variables" plot, we can see that the missing values in **rbc**, **rbcc** and **wbcc** are more than 100, which is more than 25%. Besides, from the following two plots we can see that most of the missing values occur in the observations with **class = ckd**. So we think imputing these variables might cause a big bias, especially when we know very few values of them for the observations with **class = ckd**.

Therefore, we decide to delete those 3 variables and do imputation on other variables.

2.2 Missing Data Mechanism

Before imputing the missing values, we first need to diagnose the missing data mechanism, since different NA handling approaches have different assumptions about the mechanism. The missing data mechanism can be divided into three different classes, which are MCAR(Missing completely at random), MAR(Missing at random), MNAR(Missing not at random)[2].

2.2.1 MCAR diagnosis

Based on the plots of NA distribution above, we can see that most of the missing values are in the **ckd** class, which means that the missing value is dependent on the response to some extent. Thus, we believe the missing data machinism of this dataset is not MCAR. Alternatively, we can perform the Little's test to check if the mechanism is MCAR[2]. The test result is shown below:

```
##   chi.square   df p.value
## 1   1731.071 1241       0
```

From the result we can see that the p-value of the test is 0. Thus, we should reject the null hypothesis that the missing data mechanism is MCAR.

2.2.2 MAR diagnosis

Unfortunately, no statistical test can be used to test if the data is MAR or MNAR. To diagnose if the mechanism is MAR or MNAR, we need to know the information about missing data[2], which we have no idea about in this problem. Therefore, we are going to make an assumption that the missing data mechanism in this problem is MAR and then we do data imputation based on this assumption.

2.3 Multiple imputation

By assuming the missing data mechanism is MAR, we can use multiple imputation to impute the missing data[3]. Multiple imputation is an iterative form of stochastic imputation. Instead of filling in a single value,

the distribution of the observed data is used to estimate multiple values that reflect the uncertainty around the true value. These values are then used in the analysis of interest and then we combine the results to evaluate the imputed values. Each imputed value includes a random component whose magnitude reflects the extent to which other variables in the imputation model cannot predict it's true values[4].

Therefore, in order to keep the information of the dataset as much as possible, we decide to use multiple imputation to impute our data. The analysis method we use for the imputation is predictive mean matching, which works for both categorical and continuous predictors.

In R, there are many powerful packages that can be used for multiple imputation and the package we use in this problem is **Hmisc**, which can automatically recognizes the variables types and uses bootstrap sample and predictive mean matching to impute missing values[5].

After imputing the data, we can get the R^2 values for the last round of imputations as follow. It measures the bias of the imputed data and higher the value, better are the values predicted[5].

```
##       age        bp        al        su        pc        sg       pcc
## 0.4004519 0.3599085 0.7319239 0.7510124 0.6026384 0.6538058 0.4968345
##        ba       bgr        bu        sc       sod       pot      hemo
## 0.3489563 0.6451389 0.7412032 0.6093740 0.5939785 0.3550599 0.8704963
##       pcv       htn        dm       cad     appet        pe       ane
## 0.8621372 0.6850208 0.6732389 0.4496494 0.4043171 0.5187983 0.5366865
```

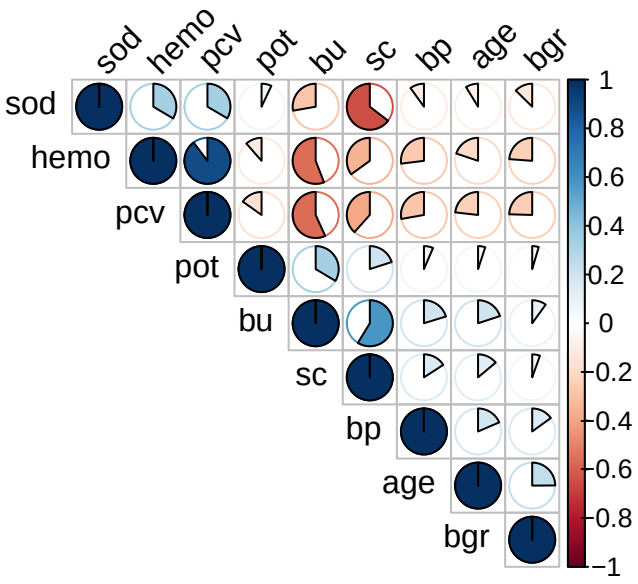
From the R^2 from those variables, we can see that though some variables have a R^2 of only about 0.4, most of the variables have a R^2 greater than 0.6. So we think it's appropriate to use this imputed data for our further analysis. Although some bias might be introduced inevitably, the bias would be much bigger if we don't do data imputation.

3 Exploratory Data Analysis

3.1 Correlation Table and PCA

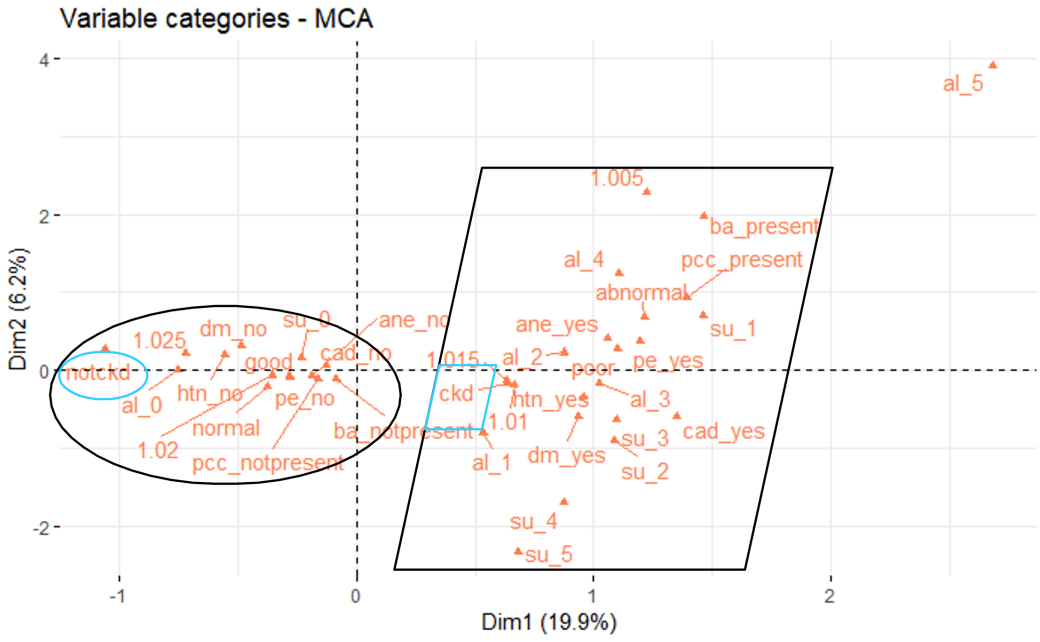
Before building the classification models, we first need to do some exploratory data analysis to examine the relationship between variables in our dataset. Since there are still 9 numeric variables after missing value handling, we decide to plot a correlation table first to check if these variables are highly correlated.

In the correlation table below, 4 pairs of variables have shown high correlation(>0.5). They are **bu-sc**, **bu-pcv**, **bu-hemo** and **sod-sc**. Considering there exists multicollinearity in this dataset, it occurs to us that we may use principle component analysis to reduce the dimension of our dataset. If the number of principle components are far less than 9(the number of numeric variables), we should replace original variables with principle components in order to decrease computational expenses. However, PCA provides nine principle components, which means it doesn't reduce dimension at all. Therefore, we are still going to build our models using the original variables for better interpretation.



3.2 Multiple Correspondence Analysis

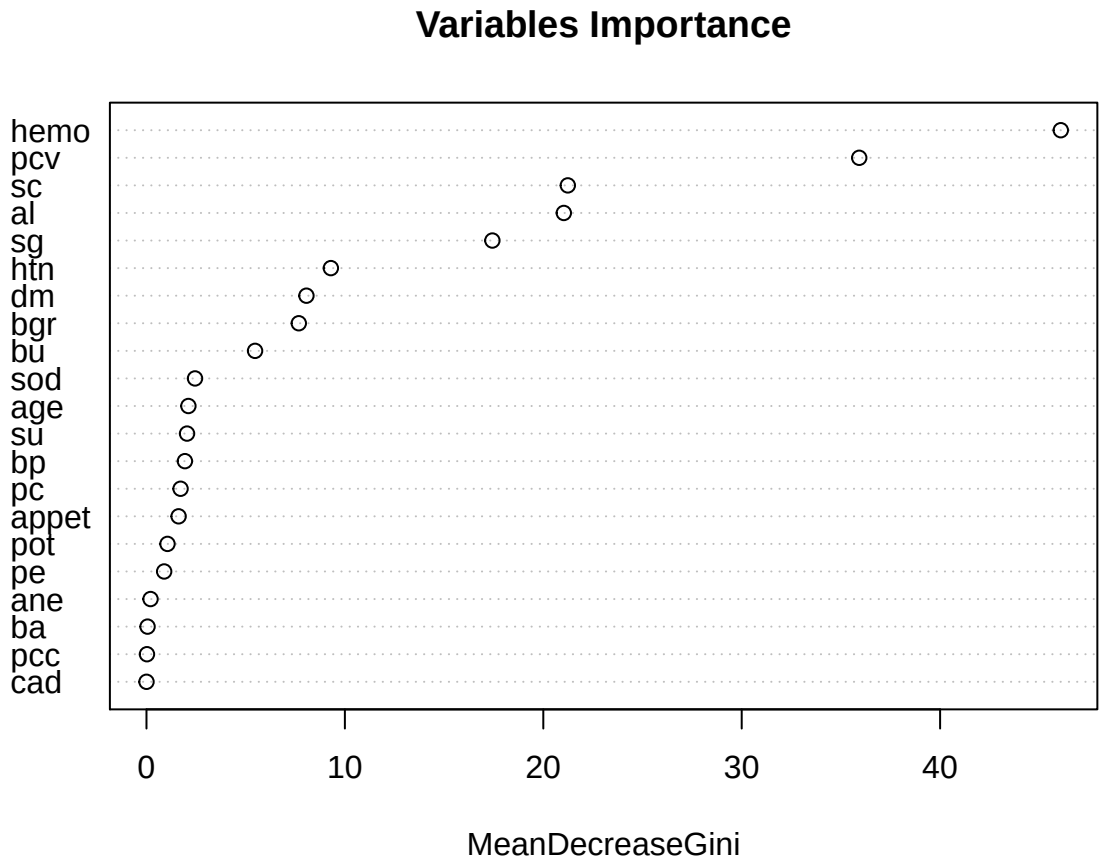
For the other 14 categorical variables, we use multiple correspondence analysis to detect their relationship with the response variable. The plot of MCA is a little messy, but it provides some useful information. In this plot below, each point is labelled with a variable name and its level. It is easy to find that the plot is segregated by two clusters of points and they represents two levels(**ckd**, **notckd**) of the response(**class**) respectively. This means the target variable's levels are distinctly different with each other. Thus, probably we are able to obtain some classification models with good prediction performance.



3.3 Variable Importance

Except the PCA and MCA, we also try to make a plot of variable importance by using the random forest model. In the plot below, variables are listed in a descending order in terms of its importance in the random

forest model. The top 5 most important variables are **hemo**, **pcv**, **sc**, **al** and **sg**. This result will be helpful for the variable selection when we are building classification models.



To sum up, through the exploratory data analysis above, we obtain three conclusions as follow:
(1) For numeric variables, there is no need to replace them by principle components.
(2) The two levels of the response(**ckd**, **notckd**) are distinctly different.
(3) The top five most important variables are **hemo**, **pcv**, **sc**, **al** and **sg**.

4 Building classification models

Before fitting models, we first randomly split our dataset into training set(70%) and testing set(30%), where training set is used to train the models and testing set is used to evaluate the models.

4.1 Logistic regression

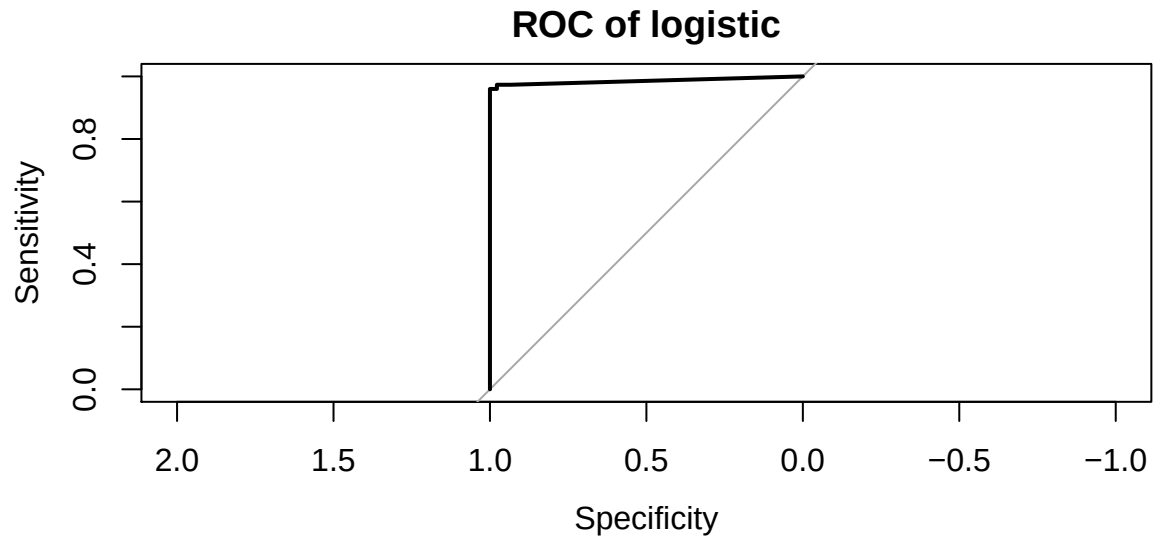
Considering that we have a binary response in this dataset, so we first try to fit a logistic regression model with all the covariates. Then we do stepwise regression using AIC to do variable selection. The final model we get is as follow:

```
##
## Call:
## glm(formula = class ~ sg + bgr + sod + hemo + pcv + htn + appet,
##     family = "binomial", data = kidney_train)
##
```

```
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -2.139e-04  -2.100e-08   2.100e-08   2.100e-08   1.820e-04
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.289e+03  3.269e+05   0.007    0.994
## sg1.01       1.434e+02  1.170e+05   0.001    0.999
## sg1.015      1.562e+02  1.171e+05   0.001    0.999
## sg1.02      -4.055e+01  1.161e+05   0.000    1.000
## sg1.025      5.073e-01  1.160e+05   0.000    1.000
## bgr          5.275e-01  9.681e+01   0.005    0.996
## sod         -1.112e+01  1.724e+03  -0.006    0.995
## hemo        -4.531e+01  5.645e+03  -0.008    0.994
## pcv         -5.693e+00  8.637e+02  -0.007    0.995
## htnyes       1.300e+02  2.705e+04   0.005    0.996
## appetpoor    9.253e+01  3.093e+04   0.003    0.998
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3.7048e+02  on 279  degrees of freedom
## Residual deviance: 1.6814e-07  on 269  degrees of freedom
## AIC: 22
##
## Number of Fisher Scoring iterations: 25
```

From the summary output, we can see that the residual deviance of this model is $2.8214e-05$ and the corresponding p-value is very close to 1. It means that this model is actually correctly specified. However, if we look at the p-values of those predictors, we can find that none of them show significant relationships with the response. Considering that in logistic regression, including more covariates might decrease the precision of estimated effect[6], the reason for those large p-values might be that those covariates are highly correlated with each other(i.e. they are the measurements of similar things). Therefore, the significant relationship between some predictors and the target variable is cancelled in the logistic model.

To verify our guess, we make prediction on the testing set using this model and we get the ROC curve and corresponding AUC value as follow:



```
## Area under the curve: 0.9855
```

We find that the AUC value for this prediction is 0.9855, which shows that this model has a very good prediction performance on the testing data and it also verifies our guess.

4.2 Logistic regression with single predictor

Although we have already gotten a pretty good prediction result using the previous model, we cannot see the effect of those covariates clearly with so many covariates in the model, especially for those important covariates. Thus, we fit another logistic regression model with only one predictor, **hemo**, which is also the most important predictor shown in the random forest model. The model we get is shown below:

```
##
## Call:
## glm(formula = class ~ hemo, family = "binomial", data = kidney_train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.64874  -0.26816   0.03266   0.19286   2.81916
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  22.1019     2.9759   7.427 1.11e-13 ***
## hemo         -1.6184     0.2165  -7.477 7.61e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 370.48  on 279  degrees of freedom
## Residual deviance: 128.51  on 278  degrees of freedom
## AIC: 132.51
##
## Number of Fisher Scoring iterations: 7
```

From the model we can see that **hemo** is extremely significant in this new model. Its coefficient is 1.502, which means that the odds of **ckd** would increase $exp(1.502) - 1 = 3.490661$ times when **hemo** increases by one unit.

Similarly, we can also fit a logistic regression model with only **pcv** to get its effect on **ckd** disease.

```
##
## Call:
## glm(formula = class ~ pcv, family = "binomial", data = kidney_train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.55857  -0.32175   0.04761   0.27776   3.11184
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 19.84983    2.71213   7.319 2.50e-13 ***
## pcv         -0.47469    0.06474  -7.332 2.26e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

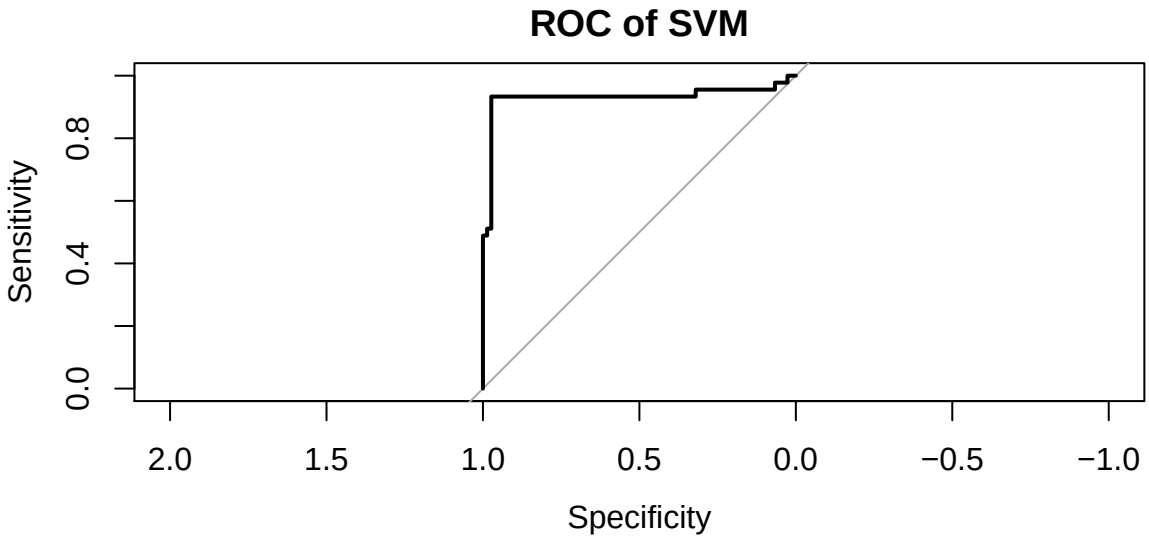
```
##
##     Null deviance: 370.48  on 279  degrees of freedom
## Residual deviance: 147.33  on 278  degrees of freedom
## AIC: 151.33
##
## Number of Fisher Scoring iterations: 7
```

From the summary output of this model, we can see that the relationship between **pcv** and **ckd** is also very significant. Besides, its coefficient is 0.40310, which means that the odds of **ckd** would increase $exp(0.40310) - 1 = 0.4964565$ times when **pcv** increases by one unit.

Furthermore, we also try to make prediction using these two models and the auc values we get are 0.9696 and 0.9427 respectively. Thus, we think these two models are very meaningful, since for hospital with limited resource or patients with limited money, having a test on only one of these two covariates might be good enough to diagnose the chronic kidney disease.

4.3 SVM

In the end, we also try to apply SVM(Support Vector Machine) on our dataset. We have the ROC curve below showing that the prediction performance of the SVM model is also good, with its auc value reaching 0.931, though it is a little bit worse than the logistic model in terms of auc. We think the reason for that might be that we cannot split the **ckd** patients and **notckd** patients completely using a hyperplane and thus SVM is not so effective in this case.




```
## Area under the curve: 0.931
```

5 Application

5.1 Chronic kidney disease detector

After fitting models above, we can see that the first logistic regression model has the best prediction performance. Therefore, we decide to build the chronic kidney disease detector using that model.

So the model we are going to use is a model with covariates **age**, **sg**, **bgr**, **hemo**, **pcv**, **dm** and **appet**, which is the model we get after stepwise logistic regression.

However, since we can only get a predicted probability from the model, we still need to determine a cutoff to give the prediction result. In order to get the optimum cutoff that maximize prediction accuracy, we use

5-fold cross validation to split the training data into 5 folds. For each validation, we build the model using 4 folds of the data and then make prediction on the other fold to get the prediction accuracy. Then we can get 5 accuracy for every given cutoff and thus we can use the **optimize** function in R to find the cutoff that maximizes the average accuracy.

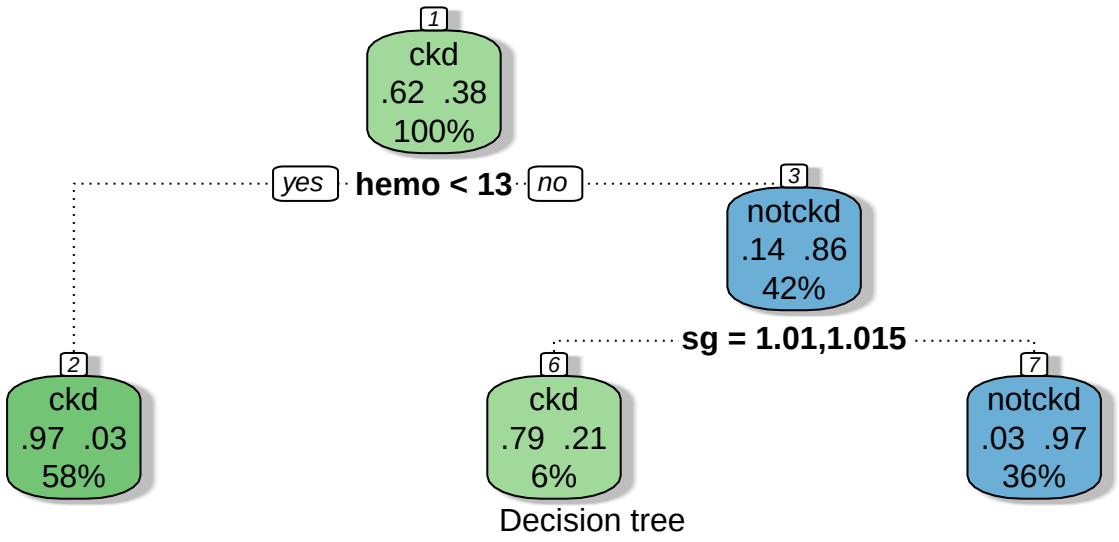
After performing above steps, we obtain that the optimum cutoff for this prediction problem is 0.976, which gives an average accuracy of 96.1%. So we try to use this cutoff to make prediction on the testing data and the result we get is as follow:

```
## Confusion Matrix and Statistics
##
##          actual
## predicted  0  1
##         0 44  3
##         1  1 72
##
##                Accuracy : 0.9667
##                  95% CI : (0.9169, 0.9908)
##     No Information Rate : 0.625
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9295
##
##  Mcnemar's Test P-Value : 0.6171
##
##             Sensitivity : 0.9600
##             Specificity : 0.9778
##          Pos Pred Value : 0.9863
##          Neg Pred Value : 0.9362
##              Prevalence : 0.6250
##          Detection Rate : 0.6000
##    Detection Prevalence : 0.6083
##       Balanced Accuracy : 0.9689
##
##        'Positive' Class : 1
##
```

From the result, we can see that the detector not only has a good performance in accuracy, but also in sensitivity and specificity. Therefore, we believe it's suitable for hospitals to use this detector to help diagnose the chronic kidney disease for patients.

5.2 Health advice for avoiding chronic kidney disease

We also want to offer some recommendations to patients in order to help them avoid chronic kidney disease. We also fit a decision tree model to explain the data in a more intuitive way. The decision tree model we get is as follow:



Decision tree

From the model we can see that for a patient with **hemo** value less than 13 or **sg** value equal to 1.01 or 1.015, he/she is very likely to have the chronic kidney disease.

Thus, if a patient want to stay away from chronic kidney disease, he/she had better increase the intake of iron-rich foods to increase the hemoglobin level[7]. In addition, a patient is also recommended to go to spa or take some other actions to increase his/her blood specific gravity.

6 Summary

6.1 Conclusion

Based on the analysis above, we conclude that **hemo** is the most important covariate for **ckd**, the odds of **ckd** would increase 3.490661 times when **hemo** increases by one unit. In addition, to diagnose the chronic kidney disease, it is not essential to measure all covariates in the dataset. Basically, we can get a pretty good diagnostic accuracy if we collect the information of **age**, **sg**, **bgr**, **hemo**, **pcv**, **dm** and **appet**. Actually, even if we only know about **hemo** or **pcv**, we can still obtain a good diagnostic result, since these covariates are highly correlated with each other and they are the measurement of the similar things.

6.2 Further discussion

In this problem, we utilize the stepwise logistic regression model and cutoff optimization to build a chronic kidney disease detector that has a test accuracy of 96.67%. Although the result is pretty good already, applying a neural network model might be able to improve the performance of the detector. It's not included in this report since building a neural network model is computationally expensive and we don't think it's necessary to increase our prediction accuracy by 1 percent or 2 by using neural network model.

Additionally, as for dealing with the large proportion of missing values in our original dataset, although multiple imputation is an excellent imputation method that can minimize the bias by adding random components, some bias might still be introduced inevitably. What's more, before performing multiple imputation, we assume that the missing data mechanism of this dataset is MAR, which could also be incorrect. Therefore, the result we get from the models might be deviated from the reality to some extent, and should be applied carefully.

7 References

1. Facts about chronic kidney disease [Internet]. National Kidney Foundation. 2019. Available from: https://www.kidney.org/atoz/content/about-chronic-kidney-disease

2. GRACE-MARTIN K. How to diagnose the missing data mechanism [Internet]. The Analysis Factor. 2018. Available from: https://www.theanalysisfactor.com/missing-data-mechanism/

3. Jakobsen JC, Gluud C, Wetterslev J, Winkel P. When and how should multiple imputation be used for handling missing data in randomised clinical trials–a practical guide with flowcharts. BMC medical research methodology. 2017;17(1):162.

4. HOME [Internet]. IDRE Stats. Available from: https://stats.idre.ucla.edu/stata/seminars/mi_in_stata_pt1_new/

5. Content AV. Tutorial on 5 powerful packages used for imputing missing values in r [Internet]. Analytics Vidhya. 2019. Available from: https://www.analyticsvidhya.com/blog/2016/03/tutorial-powerful-packages-imputing-missing-values/

6. Lachin JM. Biostatistical methods: The assessment of relative risks. Vol. 509. John Wiley & Sons; 2009.

7. Davis CP. Hemoglobin ranges: Normal, symptoms of high and low levels [Internet]. MedicineNet. MedicineNet; 2019. Available from: https://www.medicinenet.com/hemoglobin/article.htm

8 Appendix

8.1 Data preprocessing

```
# Read the data
kidney_data = read_csv("Original_data.csv", na = c("", "NA", "?"))
kidney_data = read_csv("chronic_kidney_disease.csv", na = c("", "NA", "?"))
as.tibble(kidney_data)
# make colnames meaningful
colnames(kidney_data) = gsub("'", "", colnames(kidney_data))
#-------------------------------------------------------------------------
# Change variables' class
factor_no = c(3, 4, 5, 6, 7, 8, 9, 19:25) + 1
factor_name = colnames(kidney_data)[factor_no]
numeric_name = colnames(kidney_data)[-c(1, factor_no)]
kidney_data = kidney_data %>% mutate_each_(funs(factor), factor_name) %>%
  mutate_each_(funs(as.numeric), numeric_name)

# Remove column ID
data1 = kidney_data[, -1]

# Listwise deletion
data2 = drop_na(data1)
```

8.2 Missing values handling

```
# Packages that deal with missing data
library(mice)
library(VIM)

# Get the distribution of NA
mice_plot <- aggr(data1, col=c('navyblue','yellow'),
```

```
                    numbers=TRUE, sortVars=TRUE,
                    labels=names(data1), cex.axis= 0.3,
                    gap=3, ylab=c("Missing data","Pattern"))
# Get the count of NA in each variable
NA_count = arrange(mice_plot$missings, Count)

# Plot the distribution of NA by variable
plot_NA_count = ggplot(NA_count, aes(x = reorder(Variable, -Count), y = Count)) +
  geom_bar(position="dodge", stat="identity") + xlab("Variables") +
  ylab("NA count") + ggtitle("NA distribution by variables")

# Plot the distribution of NA in dataframe
NA_dist = as.data.frame(ifelse(is.na(data1), 1, 0))
colnames(NA_dist) = colnames(data1)
NA_dist$row = 1:nrow(NA_dist)
NA_dist1 = NA_dist[1:250, ]
NA_dist2 = NA_dist[251:400, ]

NA_dist = gather(NA_dist, col, value, -row) %>%
  mutate(col = factor(col, levels=colnames(NA_dist)))
NA_dist1 = gather(NA_dist1, col, value, -row) %>%
  mutate(col = factor(col, levels=colnames(NA_dist1)))
NA_dist2 = gather(NA_dist2, col, value, -row) %>%
  mutate(col = factor(col, levels=colnames(NA_dist2)))

plot_NA_dist1 = ggplot(NA_dist1, aes(col, row, fill=factor(value))) +
  geom_tile(colour="grey50") +
  scale_fill_manual(values=c("1"="green", "0"="white")) +
  scale_y_reverse(breaks=1:50, expand=c(0,0)) +
  scale_x_discrete(position="top") +
  labs(fill="NA") +
  theme_classic() + ggtitle("NA distribution of class = 'ckd'")

plot_NA_dist2 = ggplot(NA_dist2, aes(col, row, fill=factor(value))) +
  geom_tile(colour="grey50") +
  scale_fill_manual(values=c("1"="green", "0"="white")) +
  scale_y_reverse(breaks=1:50, expand=c(0,0)) +
  scale_x_discrete(position="top") +
  labs(fill="NA") +
  theme_classic() + ggtitle("NA distribution of class = 'notckd'")

# Drop rbc, rbcc and wbcc
drop.cols <- c('rbc', 'rbcc', 'wbcc')
data3 = data1 %>% select(-one_of(drop.cols))


# Diagnose the missing data mechanism
## Check MCAR
library(BaylorEdPsych)
little_test = LittleMCAR(as.data.frame(data3))

# Multiple Imputation
data4 = data3
```

```r
al5 = (data4$al == 5)
al5[is.na(al5)] <- FALSE
data4[al5,]$al = 4
data4$al = factor(data4$al)

su5 = (data4$su == 5)
su5[is.na(su5)] <- FALSE
data4[su5,]$su = 4
data4$su = factor(data4$su)

library(Hmisc)
data_imputed = aregImpute(class ~ age + bp + al + su + pc + sg + pcc + ba + bgr + bu +
             sc + sod + pot + hemo + pcv + htn + dm + cad + appet + pe +
             ane,
                               data = data4, n.impute = 5, type = 'pmm')
fill_data <- function(impute = data_imputed, data = data4, im = 5) {
  cbind.data.frame(impute.transcan(x = impute,
                                   imputation = im,
                                   data = data,
                                   list.out = TRUE,
                                   pr = FALSE))
}
kidney_imputed <- fill_data()
levels(kidney_imputed$al) = c(levels(kidney_imputed$al), 5)
levels(kidney_imputed$su) = c(levels(kidney_imputed$su), 5)
kidney_imputed[al5,]$al = 5
kidney_imputed[su5,]$su = 5

# The kidney imputed data is the cleaned data
```

8.3 Exploratory data analysis

```r
# EDA
# Correlation table
library(corrplot)
kidney_imputed_numeric = dplyr::select_if(kidney_imputed, is.numeric)
res = cor(kidney_imputed_numeric)
corrplot(res, method = "pie", type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45)

# PCA
pca = prcomp(kidney_imputed_numeric, center = TRUE, scale = TRUE)
summary(pca)

# MCA
library(FactoMineR)
library(factoextra)
kidney_imputed_factor = dplyr::select_if(kidney_imputed, is.factor)
res.mca = MCA(kidney_imputed_factor) # multiple correspondence analysis
mca_plot = fviz_mca_var(res.mca, repel = TRUE, # Avoid text overlapping (slow)
                        ggtheme = theme_minimal(), col.var = "coral")

# Random Forest
```

```r
set.seed(200)
rf = randomForest(class ~ ., data = kidney_imputed, importance = TRUE)
varimp_plot = varImpPlot(rf,type=2)
```

8.4 Modelling

```r
# Logistic regression
## Split the data into training set and testing set
kidney_imputed$class = ifelse(kidney_imputed$class == "ckd", 1, 0)
set.seed(222)
index = sample(nrow(kidney_imputed), 0.7*nrow(kidney_imputed))
kidney_train = kidney_imputed[index, ]
kidney_test = kidney_imputed[-index, ]

## Fit a logistic regression model including all variables
model_logistic = glm(class ~ ., data = kidney_train, family = 'binomial')
## Use AIC to do the stepwise regression
AIC_logisitc = step(model_logistic)
summary(AIC_logisitc)
pchisq(AIC_logisitc$deviance, df=AIC_logisitc$df.residual, lower.tail=FALSE)

## Draw the ROC curve
prob_logit0=predict(AIC_logisitc,kidney_test,type='response')
library(pROC)
roc_logit0=roc(response=kidney_test$class, predictor=prob_logit0)
plot(roc_logit0, main = "ROC of logistic")
auc(roc_logit0)

## Fit another logistic regression model with hemo only
model_logistic1 = glm(class ~ hemo, data = kidney_train, family = 'binomial')
summary(model_logistic1)

## Fit another logistic regression model with pcv only
model_logistic2 = glm(class ~ pcv, data = kidney_train, family = 'binomial')
summary(model_logistic2)

## Make prediction with hemo model
prob_logit1=predict(model_logistic1,kidney_test,type='response')
## Draw the ROC curve
library(pROC)
roc_logit1=roc(response=kidney_test$class, predictor=prob_logit1)
plot(roc_logit1, main="ROC curve with Hemo only")
auc(roc_logit1)

## Make prediction with pcv model
prob_logit2=predict(model_logistic2,kidney_test,type='response')
## Draw the ROC curve
library(pROC)
roc_logit2=roc(response=kidney_test$class, predictor=prob_logit2)
plot(roc_logit2, main="ROC curve with PCV only")
auc(roc_logit2)
```

```r
# SVM
set.seed(222)
train = sample(1:400, 280, replace = FALSE)
kidney_imputed_train = kidney_imputed[train, ]
kidney_imputed_test = kidney_imputed[-train, ]
fit = svm(class ~ ., data = kidney_imputed_train,
          scale = FALSE, kernel = "radial", cost = 5)
predict_svm = predict(fit, newdata = kidney_imputed_test)
conf_mat_svm = table(predicted = predict_svm, actual = kidney_imputed_test$class)
svm_matrix = confusionMatrix(conf_mat_svm)

svm_data = kidney_imputed %>% mutate_each_(funs = as.numeric, 1)
svm_data$class = svm_data$class - 1
fit_2 = svm(class ~ ., data = svm_data[train, ],
            scale = FALSE, kernel = "radial", cost = 5)
predict_svm_2 = predict(fit_2, newdata = svm_data[-train, ])
roc_svm = roc(svm_data[-train, ]$class, predict_svm_2, plot = TRUE, col = "orange")
auc_svm = auc(roc_svm)
```

8.5 Cutoff optimization

```r
# Use 5-fold CV to find the best cutoff that maximize accuracy
## Prepare the data for cross validation
set.seed(850)
rand_index = sample(nrow(kidney_train))
kidney_cv = kidney_train[rand_index,]
folds = cut(1:nrow(kidney_train),breaks=5,labels=FALSE)

## A function that compute average accuracy
logit_cut = function(c){
  acc_logit = numeric(5)
  for(i in 1:5){
    test_index = which(folds==i)
    cv_test = kidney_cv[test_index,]
    cv_train = kidney_cv[-test_index,]
    fit_logit = glm(class ~ age + sg + bgr + hemo + pcv + dm + appet,
                    data = cv_train, family = 'binomial')
    prob_logit = predict(fit_logit,newdata=cv_test,type="response")
    pred_logit = ifelse(prob_logit > c, 1, 0)
    conf_mat_logit = table(predicted = pred_logit, actual = cv_test$class)
    cof_table = confusionMatrix(conf_mat_logit, positive = "1")
    acc_logit[i] = cof_table$overall[1]
  }
  return (mean(acc_logit))
}
## Find the optimum cutoff
cut_max=optimize(logit_cut,interval=c(0,1), maximum = T, tol = 0.00001)

## Use that cutoff to make prediction
pred_logit0=ifelse(prob_logit0>0.976, 1, 0)
## The confusion matrix
library(caret)
conf_mat_logit0 = table(predicted = pred_logit0, actual = kidney_test$class)
```

```r
confusionMatrix(conf_mat_logit0, positive = "1")
```

8.6 Decision tree

```r
# Decision tree
library(rpart)    #Decision tree
library(rattle)   #Fancy tree plot
## Build a tree model and plot it
kidney_tree0=rpart(class~.,data=kidney_imputed,method='class')
fancyRpartPlot(kidney_tree0, sub = "Decision tree")
```