finalproject_IST687

Nadia Paquin

2023-03-15

Introduction

Chronic kidney disease is a common diabetic complication that is characterized by slow, irreversible loss of kidney function. Because of its irreversible nature, diagnosing CKD as early as possible is critical in preventing further loss of function and potentially severe outcomes. Current medicinal practice uses GFR, or glomerular filtration rates, as a metric to diagnose CKD. Once GFR levels are at the critical point needed for a diagnosis however, the patient has already experienced a loss of kidney function. Thus, there is room for further research on different metrics we can use to predict and diagnose the onset of CKD at earlier stages. In this study, I hope to use a data set published by the University of Irvine to investigate a long list of other variables. This data set has 25 metrics (see Appendix for full list) for 400 patients with and without CKD. I hope to identify variables with strong associations to CKD and create a model which can be used for diagnoses.

Methods

1. Clean the data, convert categorical data to binary, fill in NA values

2. Perform simple visualizations using base packages
3. Identify which variables are of interest with a correlation matrix

4. Perform predictive analysis (regression, decision tree)

Results and Conclusions

Because my data has 25 variables, I ran a correlation matrix in order to see which variables had the highest correlation with CKD presence. From my matrix I found 8 variables to have correlation coefficients greater than 0.5 (positive and negative): sugar, albumin, red blood cells (categorical), hemoglobin, packed cell volume, red blood cell count, hypertension, and diabetes mellitus. I used these eight variables to build my models.

I conducted a linear regression first, using all variables with high correlation to predict the presence of CKD. The results yield a model with an adjusted R square value of 0.77, and F score $F(8,391)= 167.8$ ($p<2.2e-16$). These are both suggestive of a robust model. Looking at each individual predictor, all are $p<0.005$ with the exception of x. With largely significant predictors and an overall strong model, I have a pretty solid predictive analysis of CKD onset.

In order to check for multicollinearity between my predictors, I used the vif() function which yields the variance inflation factor (VIF) for each coefficient. The VIF is a statistical tool used with regression modeling to measure how much the variance of a regression coefficient is affected by collinearity. When VIF=1, no collinearity is present. VIF>10, collinearity is high. In my analysis, I saw VIFs around 1-2, with the exception of 'hemo' and 'pcv' at around 5. This suggests that there is minimal collinearity between most of our variables and minimal amounts between hemo and pcv. Because my variables don't seem to be affecting each other too heavily, I continued on to the next analysis.

Next, I created a decision tree (see code for visualization). Here, I split my data in two parts - 60% of my data goes towards my training set and 40% goes towards my testing set. This Rpart decision tree yielded a reported accuracy of 0.96 in the results. When tested on my tested set, it yielded an accuracy of 0.97, with sensitivity of 0.97 and specificity of 0.97. Overall this is a more robust predictive model to diagnose CKD.

R-code and Analysis:

Setting up

```
#Loading libraries
library(tidyverse)
library(imputeTS)
library(car)
library(caret)
library(rpart)
library(rpart.plot)
library(zoo)
library(ggplot2)
```

If I had an older version of R, I would use the Kaggle library to load my data:

```
# Install the kaggle package (if not already installed)
install.packages("kaggle")

# Load the kaggle package
library(kaggle)

# Download the ckdisease dataset
kaggle_datasets_download("mansoordaku/ckdisease")
```

Since R 4.2.1 is incompatible with Kaggle at the moment, I'll just download the CSV and load it from a local source. This file was downloaded in CSV format from Kaggle at https://www.kaggle.com/datasets/mansoordaku/ckdisease?resource=download made available by UCI's dataset archive.

```
#Loading my data.
raw <- data.frame(read.csv('kidney_disease.csv'))
```

Let's check out the struture of the raw data:

```
str(raw)
```

```
## 'data.frame':    400 obs. of  26 variables:
##  $ id             : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ age            : num  48 7 62 48 51 60 68 24 52 53 ...
##  $ bp             : num  80 50 80 70 80 90 70 NA 100 90 ...
##  $ sg             : num  1.02 1.02 1.01 1 1.01 ...
##  $ al             : num  1 4 2 4 2 3 0 2 3 2 ...
##  $ su             : num  0 0 3 0 0 0 0 4 0 0 ...
##  $ rbc            : chr  "" "" "normal" "normal" ...
##  $ pc             : chr  "normal" "normal" "normal" "abnormal" ...
##  $ pcc            : chr  "notpresent" "notpresent" "notpresent" "present" ...
##  $ ba             : chr  "notpresent" "notpresent" "notpresent" "notpresent" ...
##  $ bgr            : num  121 NA 423 117 106 74 100 410 138 70 ...
##  $ bu             : num  36 18 53 56 26 25 54 31 60 107 ...
##  $ sc             : num  1.2 0.8 1.8 3.8 1.4 1.1 24 1.1 1.9 7.2 ...
##  $ sod            : num  NA NA NA 111 NA 142 104 NA NA 114 ...
```

```
##  $ pot           : num  NA NA NA 2.5 NA 3.2 4 NA NA 3.7 ...
##  $ hemo          : num  15.4 11.3 9.6 11.2 11.6 12.2 12.4 12.4 10.8 9.5 ...
##  $ pcv           : chr  "44" "38" "31" "32" ...
##  $ wc            : chr  "7800" "6000" "7500" "6700" ...
##  $ rc            : chr  "5.2" "" "" "3.9" ...
##  $ htn           : chr  "yes" "no" "no" "yes" ...
##  $ dm            : chr  "yes" "no" "yes" "no" ...
##  $ cad           : chr  "no" "no" "no" "no" ...
##  $ appet         : chr  "good" "good" "poor" "poor" ...
##  $ pe            : chr  "no" "no" "no" "yes" ...
##  $ ane           : chr  "no" "no" "yes" "yes" ...
##  $ classification: chr  "ckd" "ckd" "ckd" "ckd" ...
```

Our data frame is 400x25 (plus a surrogate ID key). 11 of the columns are numerical, while 14 are character strings.

Cleaning

There are a couple things we can fix right away:
- There are three columns labeled as 'chr' but really should be numerical. Among these are: 'pcv', 'wc', 'rc'.
- There are empty strings in many columns when they should be NA instead.

Let's get rid of empty strings and replace them with NAs. We'll also replace '?' with NAs (these show up in columns pcv, wc, and rc).

```
raw[raw == ""] <- NA
raw[raw == "\t?"] <- NA
```

We'll also get rid of any non-numeric data entries. Some entries have letters and symbols, so we'll use gsub to take them out and leave just the numbers.

```
raw$pcv <- gsub("[^0-9]+", "", raw$pcv)
raw$rc <- gsub("[^0-9]+", "", raw$rc)
raw$wc <- gsub("[^0-9]+", "", raw$wc)
raw$classification <- gsub("ckd\t", "ckd", raw$classification)
raw$dm <-  gsub("\t", "", raw$dm)
raw$dm <- gsub(" ", "", raw$dm)
raw$cad <- gsub("\t", "", raw$cad)
```

Let's convert these character columns to numeric

```
raw$pcv <- as.numeric(raw$pcv)
raw$rc <- as.numeric(raw$rc)
raw$wc <- as.numeric(raw$wc)
```

Let's convert the character factors to binary 0 and 1s

```
numerical <- raw %>% mutate(rbcnum = ifelse(is.na(rbc)==TRUE, NA, ifelse(rbc=='normal',0,1))) %>%
  mutate(pcnum = ifelse(is.na(pc)==TRUE, NA, ifelse(pc=='normal',0,1))) %>%
  mutate(pccnum = ifelse(is.na(pcc)==TRUE, NA, ifelse(pcc=='notpresent',0,1))) %>%
  mutate(banum = ifelse(is.na(ba)==TRUE, NA, ifelse(ba=='notpresent',0,1))) %>%
  mutate(htnnum = ifelse(is.na(htn)==TRUE, NA, ifelse(htn=='no',0,1))) %>%
  mutate(dmnum = ifelse(is.na(dm)==TRUE, NA, ifelse(dm=='no',0,1))) %>%
  mutate(cadnum = ifelse(is.na(cad)==TRUE, NA, ifelse(cad=='no',0,1))) %>%
  mutate(anenum = ifelse(is.na(ane)==TRUE, NA, ifelse(ane=='no',0,1))) %>%
  mutate(penum = ifelse(is.na(pe)==TRUE, NA, ifelse(pe=='no',0,1))) %>%
  mutate(appetnum = ifelse(is.na(appet)==TRUE, NA, ifelse(appet=='good',0,1))) %>%
```

```
  mutate(classification_num = ifelse(is.na(classification)==TRUE, NA, ifelse(classification=='notckd',0
  select(id, age, bp, sg, al, su, rbcnum, pcnum, pccnum, banum, bgr, bu, sc, sod, pot, hemo, pcv, wc, r
```

For the purposes of this project we are going to fill in all of our NA values

```
interpolated <- data.frame(apply(numerical, 2, function(x) na_interpolation(x)))
```

Let's peak at the new structure to see if everything is going as planned:

```
str(interpolated)
```

```
## 'data.frame':    400 obs. of  26 variables:
##  $ id                : num  0 1 2 3 4 5 6 7 8 9 ...
##  $ age               : num  48 7 62 48 51 60 68 24 52 53 ...
##  $ bp                : num  80 50 80 70 80 90 70 85 100 90 ...
##  $ sg                : num  1.02 1.02 1.01 1 1.01 ...
##  $ al                : num  1 4 2 4 2 3 0 2 3 2 ...
##  $ su                : num  0 0 3 0 0 0 0 4 0 0 ...
##  $ rbcnum            : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ pcnum             : num  0 0 0 1 0 0 0 1 1 1 ...
##  $ pccnum            : num  0 0 0 1 0 0 0 0 1 1 ...
##  $ banum             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ bgr               : num  121 272 423 117 106 74 100 410 138 70 ...
##  $ bu                : num  36 18 53 56 26 25 54 31 60 107 ...
##  $ sc                : num  1.2 0.8 1.8 3.8 1.4 1.1 24 1.1 1.9 7.2 ...
##  $ sod               : num  111 111 111 111 126 ...
##  $ pot               : num  2.5 2.5 2.5 2.5 2.85 3.2 4 3.9 3.8 3.7 ...
##  $ hemo              : num  15.4 11.3 9.6 11.2 11.6 12.2 12.4 12.4 10.8 9.5 ...
##  $ pcv               : num  44 38 31 32 35 39 36 44 33 29 ...
##  $ wc                : num  7800 6000 7500 6700 7300 7800 7350 6900 9600 12100 ...
##  $ rc                : num  52 47.7 43.3 39 46 ...
##  $ htnnum            : num  1 0 0 1 0 1 0 0 1 1 ...
##  $ dmnum             : num  1 0 1 0 0 1 0 1 1 1 ...
##  $ cadnum            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ appetnum          : num  0 0 1 1 0 0 0 0 0 1 ...
##  $ penum             : num  0 0 0 1 0 1 0 1 0 0 ...
##  $ anenum            : num  0 0 1 1 0 0 0 0 1 1 ...
##  $ classification_num: num  1 1 1 1 1 1 1 1 1 1 ...
```

Simple Visuaizations

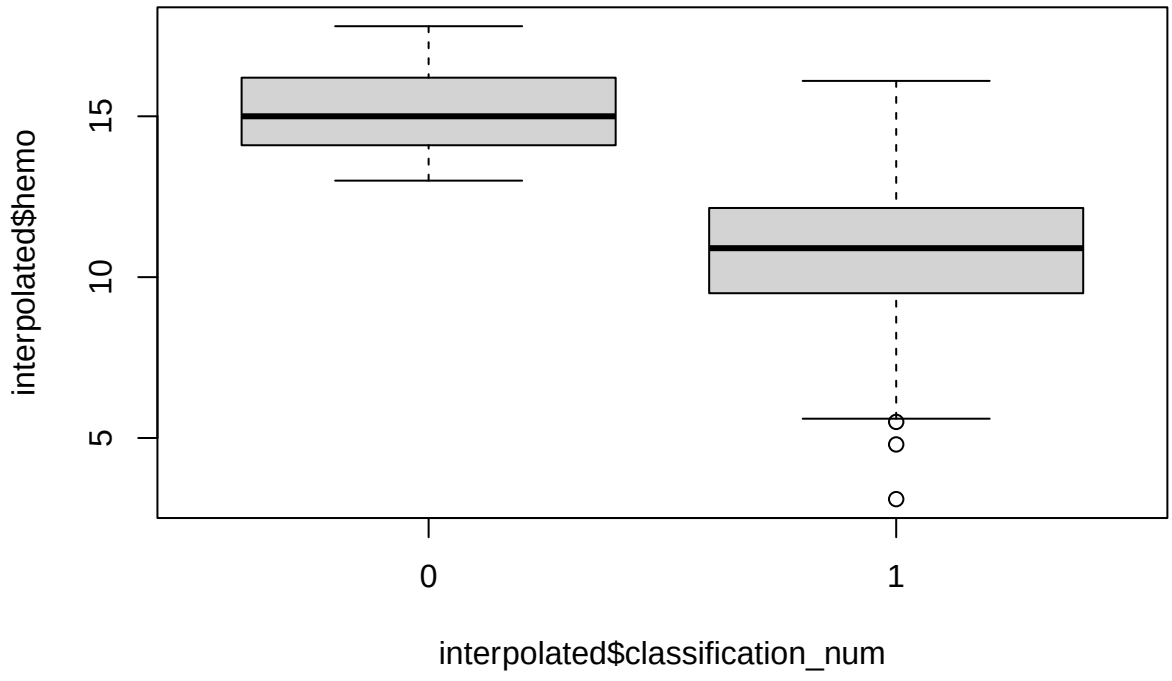
An example of quick graphs I made to visualize parts of the data.

```
hist(interpolated$age)
```


```
table(interpolated$classification_num, interpolated$dmnum)
```

```
##
##       0   1
##   0 150   0
##   1 113 137
```

```
boxplot(interpolated$hemo~interpolated$classification_num)
```



Correlations, linear model, decision tree

Finding which variables will be most promising to make a prediction model out of.

```
correlation <- data.frame(cor(interpolated))
correlation %>% select(classification_num) %>% filter(abs(classification_num)>0.5)
```

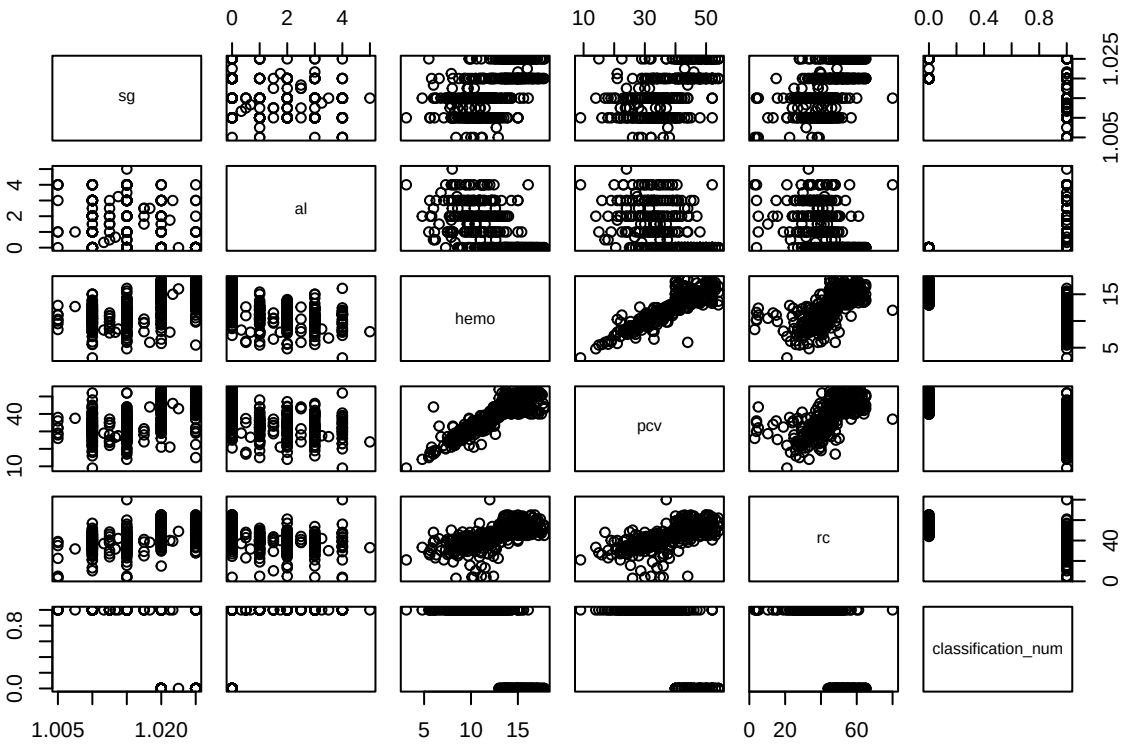
```
##                     classification_num
## id                          -0.8385281
## sg                          -0.7293099
## al                           0.6220511
## rbcnum                       0.5117182
## hemo                        -0.7611562
## pcv                         -0.7324147
## rc                          -0.6620775
## htnnum                       0.5904376
## dmnum                        0.5590595
## classification_num           1.0000000
```

Making a subset of the interpolated numerical data with just the variables that yield a correlation of $>|0.5|$.

```
subset1 <- interpolated %>% select(sg, al, rbcnum, hemo, pcv, rc, htnnum, dmnum, classification_num)
subset2 <- interpolated %>% select(sg, al, hemo, pcv, rc, classification_num) #same as subset 1 but lac
```

Just to take a quick look at the relationship between all of our notable variables. Here we see some patterns that are likely a product of the na.approx method I used while generating data.

```
pairs(subset2)
```



Basic linear model using all variables

```
fullmodel <- lm(classification_num ~ .,  data=subset1)
summary(fullmodel)
```

```
##
## Call:
## lm(formula = classification_num ~ ., data = subset1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.71971 -0.16942 -0.04404  0.14786  0.62391
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  28.922153   2.778747  10.408  < 2e-16 ***
## sg          -27.219588   2.757430  -9.871  < 2e-16 ***
## al            0.048970   0.011410   4.292 2.24e-05 ***
## rbcnum        0.170717   0.033413   5.109 5.07e-07 ***
## hemo         -0.035128   0.009309  -3.774 0.000186 ***
## pcv          -0.003890   0.002933  -1.326 0.185475
## rc           -0.004415   0.001481  -2.981 0.003052 **
## htnnum        0.099098   0.034592   2.865 0.004399 **
## dmnum         0.097018   0.032308   3.003 0.002846 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2325 on 391 degrees of freedom
## Multiple R-squared:  0.7745, Adjusted R-squared:  0.7699
## F-statistic: 167.8 on 8 and 391 DF,  p-value: < 2.2e-16
```

Checking collinearity between variables.

```
vif(fullmodel)
```

```
##       sg       al   rbcnum     hemo      pcv       rc   htnnum    dmnum
## 1.757012 1.693729 1.291510 5.243578 5.007404 2.109952 2.057440 1.738674
```

Building training and testing subsets for an rpart model

```
trainList <- createDataPartition(y=subset1$classification_num, p=0.6, list=FALSE)
trainset <- subset1[trainList,]
testset <- subset1[-trainList,]
```

Building a decision tree with caret

```
tree <- train(classification_num ~., data=trainset, method='rpart')
```

```
## Warning in train.default(x, y, weights = w, ...): You are trying to do
## regression and your outcome only has two possible values Are you trying to do
## classification? If so, use a 2 level factor as your outcome column.
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

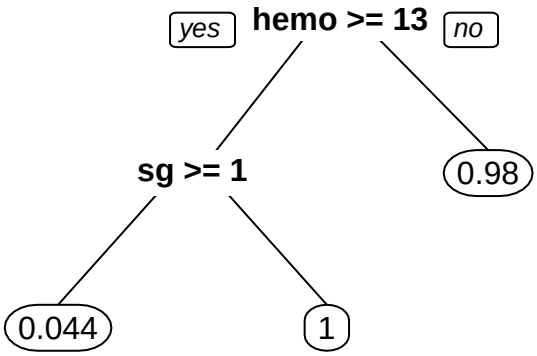
```
tree
```

```
## CART
##
## 240 samples
##   8 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
```

```
## Summary of sample sizes: 240, 240, 240, 240, 240, 240, ...
## Resampling results across tuning parameters:
##
##   cp          RMSE       Rsquared   MAE
##   0.02182902  0.1947966  0.8298139  0.05957095
##   0.11980159  0.2047578  0.8104052  0.06915636
##   0.75949582  0.3203550  0.7021464  0.20055612
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 0.02182902.
```

Visualising the decision tree

```
prp(tree$finalModel)
```




```
predictValues <- predict(tree, newdata=testset)
predictValues <- factor(round(predictValues, digits=1))

testset$classification_num <- factor(testset$classification_num)

confusion <- confusionMatrix(predictValues, testset$classification_num)
confusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 61  2
##          1  0 97
##
##                Accuracy : 0.9875
##                  95% CI : (0.9556, 0.9985)
##     No Information Rate : 0.6188
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9737
##
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 1.0000
##             Specificity : 0.9798
##          Pos Pred Value : 0.9683
##          Neg Pred Value : 1.0000
##              Prevalence : 0.3812
```

```
##          Detection Rate : 0.3812
##    Detection Prevalence : 0.3937
##       Balanced Accuracy : 0.9899
##
##        'Positive' Class : 0
##
```

Appendix

Variables:
age - age
bp - blood pressure
sg - specific gravity
al - albumin
su - sugar
rbc - red blood cells
pc - pus cell
pcc - pus cell clumps
ba - bacteria
bgr - blood glucose random
bu - blood urea
sc - serum creatinine
sod - sodium
pot - potassium
hemo - hemoglobin
pcv - packed cell volume
wc - white blood cell count
rc - red blood cell count
htn - hypertension
dm - diabetes mellitus
cad - coronary artery disease
appet - appetite
pe - pedal edema
ane - anemia
class - class