appendix

11/9/2022

```
library(tidyverse)
library(class)
library(kknn)
library(ggplot2)
library(caret)
```

```
# can only use numerical ones in the knn model
train <- read.csv("train_df.csv") %>% select(-X, -race)
test <- read.csv("test_df.csv") %>% select(-X, -race)
```

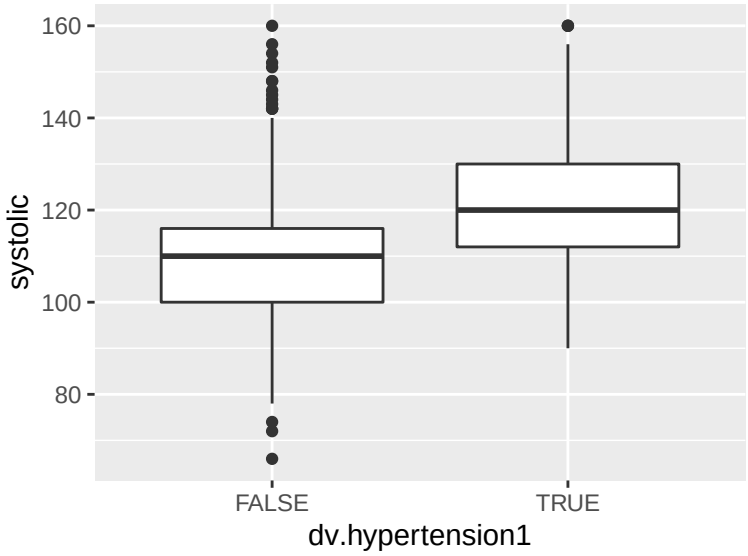
EDA

```
summary(train)
```

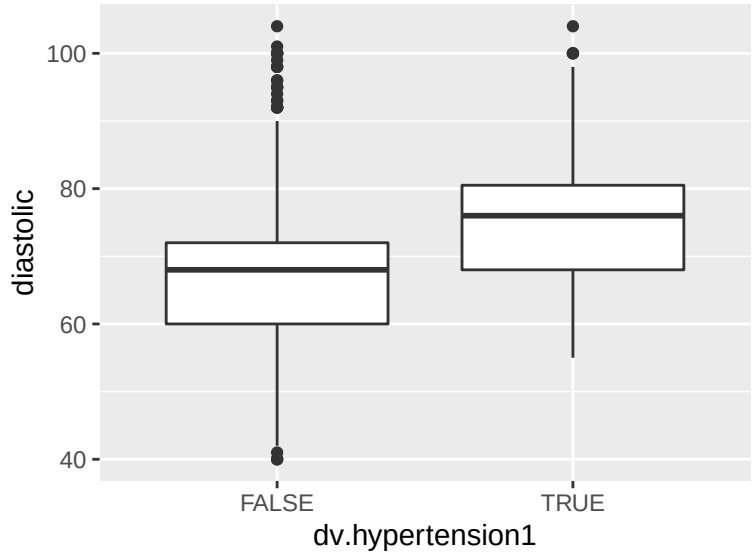
```
##       age         emosupport      financialsupport prenatalsupport
##  Min.   : 0.00   Mode :logical   Mode :logical    Mode :logical
##  1st Qu.:23.00   FALSE:323       FALSE:546        FALSE:633
##  Median :28.00   TRUE :5231      TRUE :5008       TRUE :4921
##  Mean   :27.11
##  3rd Qu.:31.00
##  Max.   :52.00
##  deliverysupport    psstotal        anxtotal      worryfambaby  exercise
##  Mode :logical   Min.   : 0.00   Min.   :20.00   Min.   :0.0   Mode :logical
##  FALSE:327       1st Qu.:28.00   1st Qu.:30.00   1st Qu.:4.0   FALSE:1568
##  TRUE :5227      Median :30.00   Median :34.00   Median :5.0   TRUE :3986
##                  Mean   :29.74   Mean   :35.34   Mean   :4.7
##                  3rd Qu.:32.00   3rd Qu.:40.00   3rd Qu.:5.0
##                  Max.   :50.00   Max.   :72.00   Max.   :9.0
##     systolic       diastolic     worryhealthcare worrysymptoms
##  Min.   : 66.0   Min.   : 40.0   Min.   :0.000   Min.   : 5.000
##  1st Qu.:100.0   1st Qu.: 60.0   1st Qu.:2.000   1st Qu.: 7.000
##  Median :110.0   Median : 68.0   Median :2.000   Median : 9.000
##  Mean   :109.2   Mean   : 67.2   Mean   :2.692   Mean   : 9.078
##  3rd Qu.:118.0   3rd Qu.: 72.0   3rd Qu.:3.000   3rd Qu.:10.000
##  Max.   :160.0   Max.   :104.0   Max.   :6.000   Max.   :18.000
##     ssqmean        prepreglbs    familypreeclampsia     income
##  Min.   :0.000   Min.   :  0.0   Min.   :1.000      Min.   : 0.000
##  1st Qu.:6.000   1st Qu.:125.0   1st Qu.:3.000      1st Qu.: 4.000
##  Median :6.583   Median :140.0   Median :3.000      Median :10.000
##  Mean   :6.198   Mean   :150.8   Mean   :2.786      Mean   : 7.899
##  3rd Qu.:7.000   3rd Qu.:168.0   3rd Qu.:3.000      3rd Qu.:12.000
```

```
##  Max.   :7.000   Max.   :368.0   Max.   :3.000      Max.   :14.000
##  dv.hypertension1  kidney1          lupus1        collagen1
##  Mode :logical    Mode :logical   Mode :logical   Mode :logical
##  FALSE:5371       FALSE:5451      FALSE:5544      FALSE:5460
##  TRUE :183        TRUE :103       TRUE :10        TRUE :94
##
##
##
##   crohns1          pcos1         discrimination     bornearly
##  Mode :logical   Mode :logical   Min.   : 0.000   Min.   :1.000
##  FALSE:5503      FALSE:5309      1st Qu.: 1.000   1st Qu.:3.000
##  TRUE :51        TRUE :245       Median : 1.000   Median :3.000
##                                  Mean   : 1.626   Mean   :2.796
##                                  3rd Qu.: 2.000   3rd Qu.:3.000
##                                  Max.   :11.000   Max.   :3.000
```

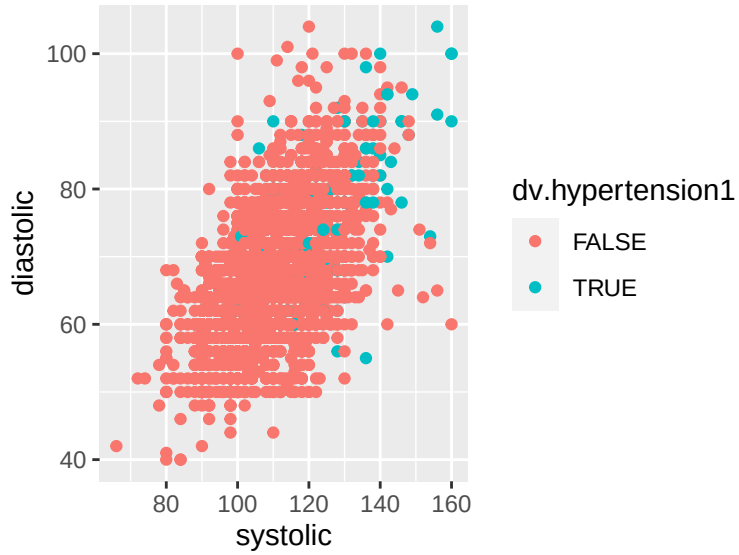
```
ggplot(data = train) + geom_boxplot(mapping = aes(x = dv.hypertension1, y = systolic))
```



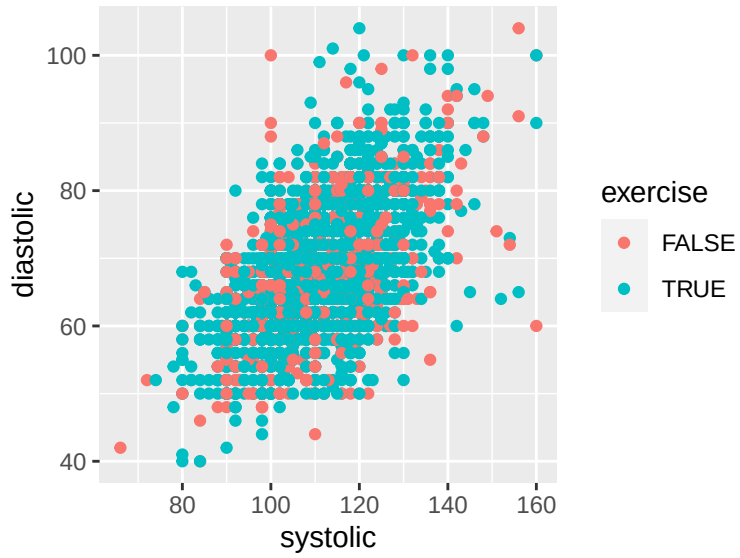

```
ggplot(data = train) + geom_boxplot(mapping = aes(x = dv.hypertension1, y = diastolic))
```



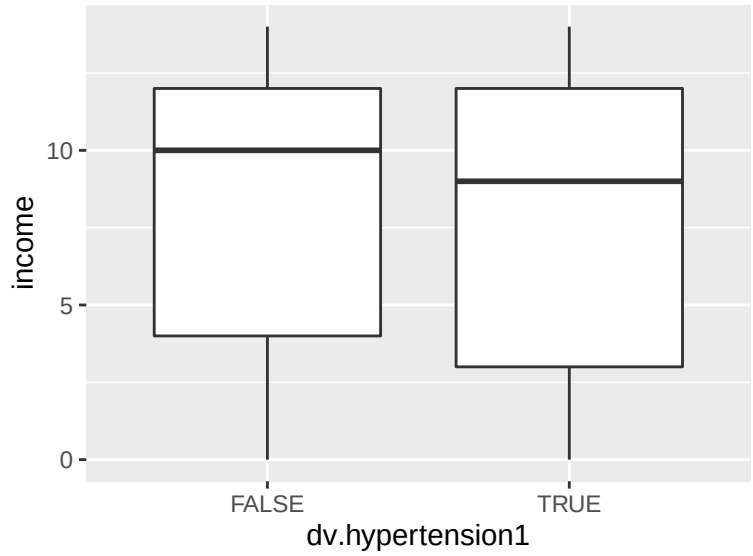

```
ggplot(data = train) + geom_point(mapping = aes(x = systolic, y = diastolic, color = dv.hypertension1))
```



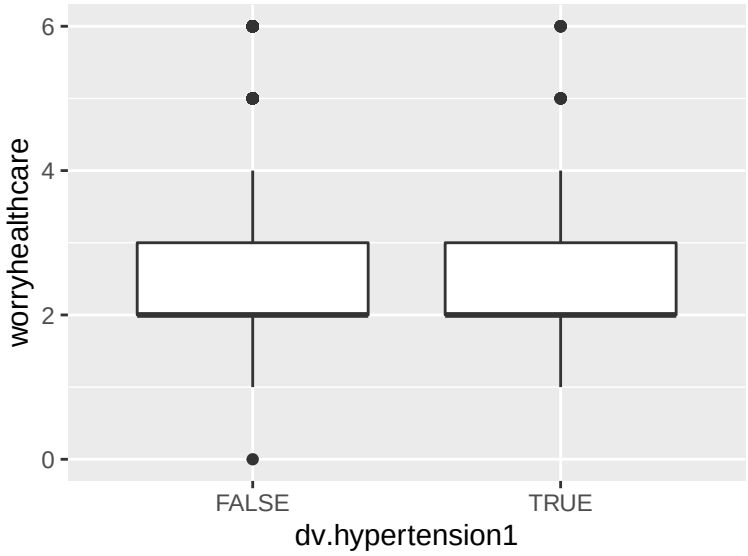

```
ggplot(data = train) + geom_point(mapping = aes(x = systolic, y = diastolic, color = exercise))
```



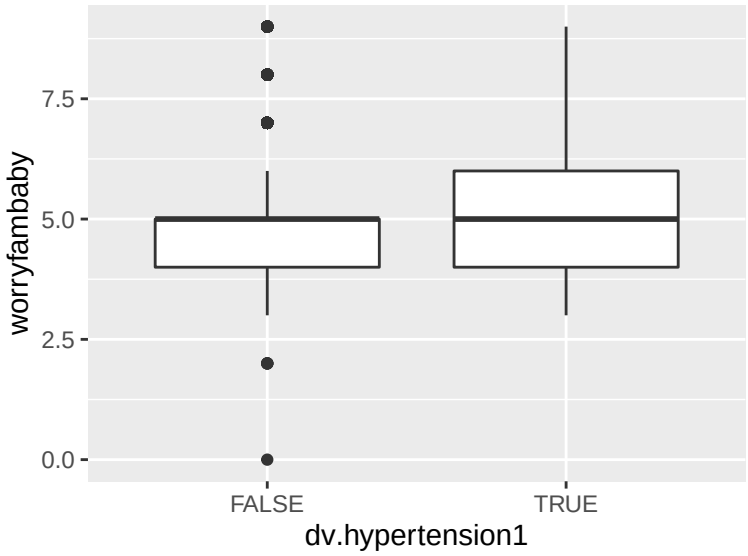

```
ggplot(data = train) + geom_boxplot(mapping = aes(x = dv.hypertension1, y = income))
```



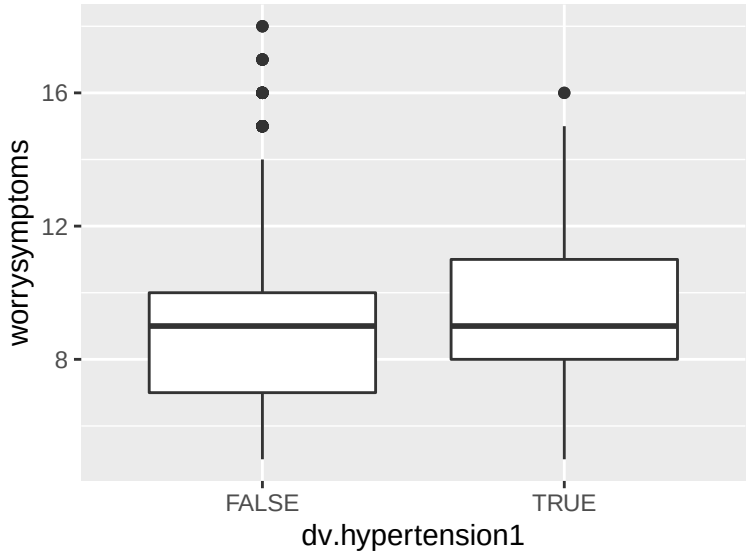

```
ggplot(data = train) + geom_boxplot(mapping = aes(x = dv.hypertension1, y = worryhealthcare))
```



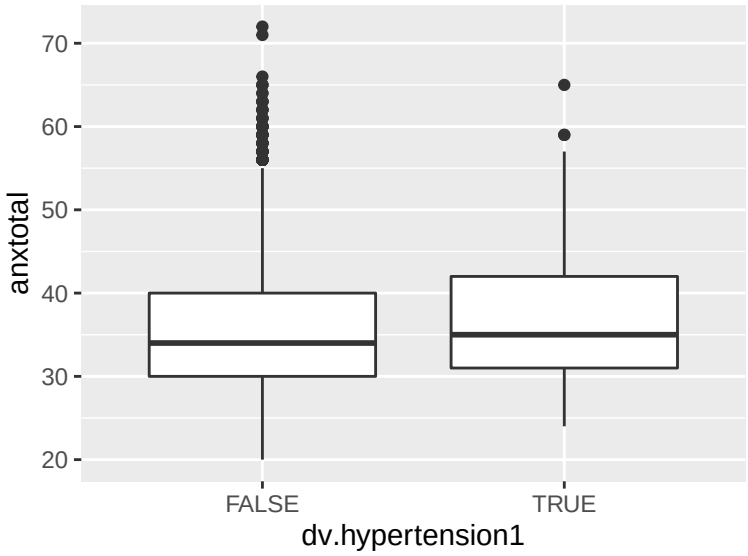

```
ggplot(data = train) + geom_boxplot(mapping = aes(x = dv.hypertension1, y = worryfambaby))
```



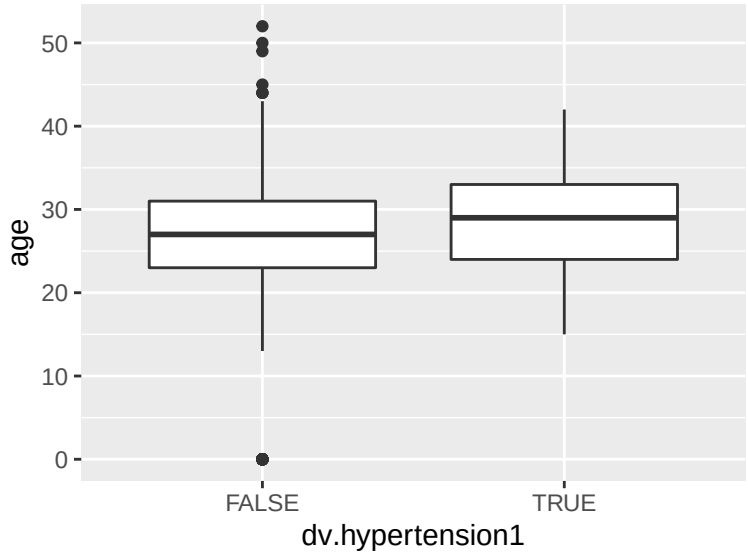

```
ggplot(data = train) + geom_boxplot(mapping = aes(x = dv.hypertension1, y = worrysymptoms))
```



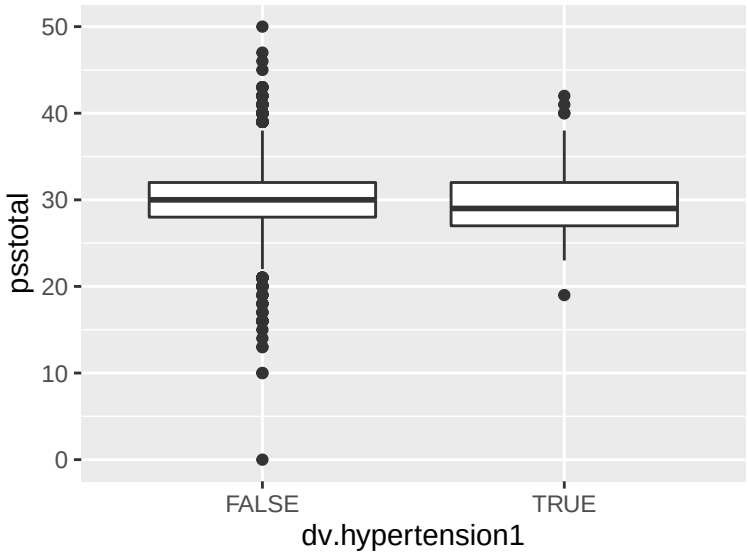

```
ggplot(data = train) + geom_boxplot(mapping = aes(x = dv.hypertension1, y = anxtotal))
```



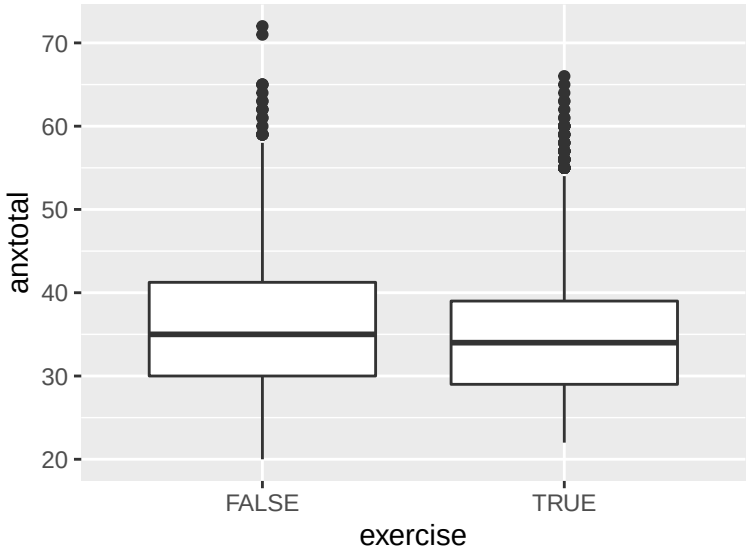

```
ggplot(data = train) + geom_boxplot(mapping = aes(x = dv.hypertension1, y = age))
```



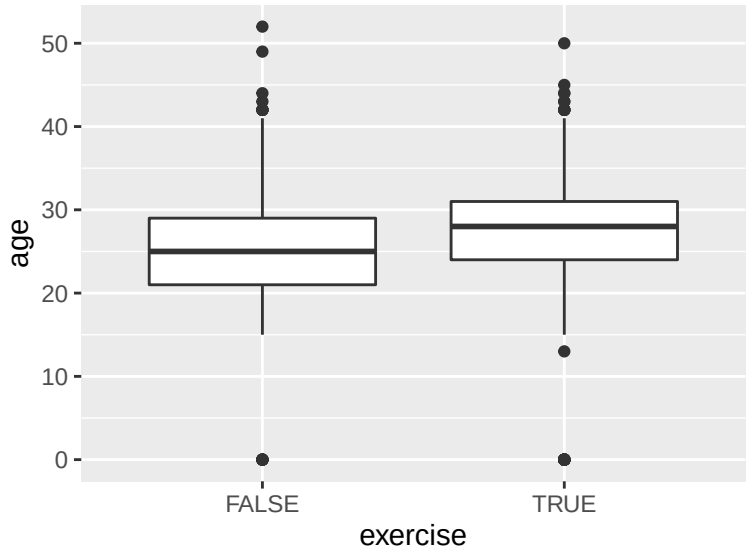

```
ggplot(data = train) + geom_boxplot(mapping = aes(x = dv.hypertension1, y = psstotal))
```



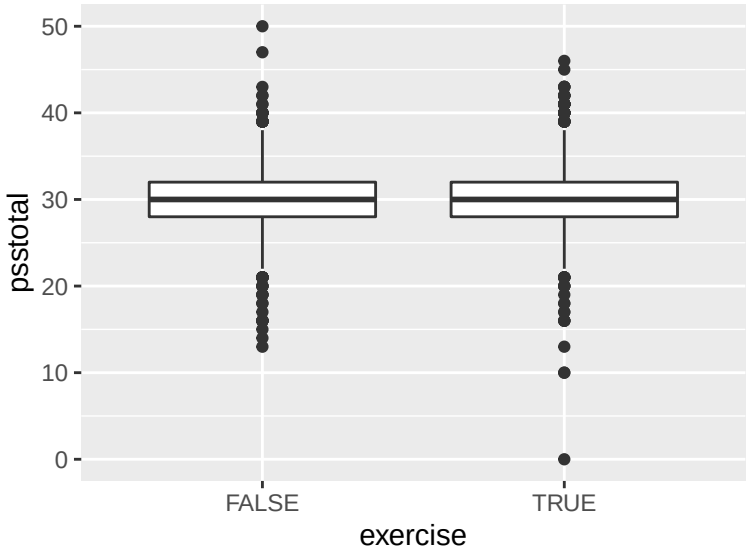

```
ggplot(data = train) + geom_boxplot(mapping = aes(x = exercise, y = anxtotal))
```




```
ggplot(data = train) + geom_boxplot(mapping = aes(x = exercise, y = age))
```




```
ggplot(data = train) + geom_boxplot(mapping = aes(x = exercise, y = psstotal))
```



scale the data for knn

```
train_x <- train %>% select(-dv.hypertension1)
train_label <- train %>% .$dv.hypertension1
test_x <- test %>% select(-dv.hypertension1)
test_label <- test %>% .$dv.hypertension1
```

```
mean_train = colMeans(train_x)
std_train = sqrt(diag(var(train_x)))
# training data
train_x = scale(train_x, center = mean_train, scale = std_train)
# test data
test_x = scale(test_x, center = mean_train, scale = std_train)
```

K-Fold CV

```
Kfold_CV_knn <- function(K,K_knn,train,train_label){
  fold_size <- floor(nrow(train)/K)
  cv_error <- rep(0,K)
  sensitives <- rep(0,K)
  for(i in 1:K){
    # select K-1 folds
    if(i!=K){
      CV_test_rows = ((i-1)*fold_size+1):(i*fold_size)
    }else{
      CV_test_rows = ((i-1)*fold_size+1):nrow(train)
    }
    CV_train <- train[-CV_test_rows,]
    CV_test <- train[CV_test_rows,]
```

```
    # normalize training and testing using mean and sd
    mean_CV_train <- colMeans(CV_train)
    sd_CV_train <- apply(CV_train,2,sd)

    CV_train <- scale(CV_train,center = mean_CV_train,scale = sd_CV_train)
    CV_test <- scale(CV_test,center = mean_CV_train,scale = sd_CV_train)
    # Fit
    pred_CV_test <- knn(CV_train,CV_test,train_label[-CV_test_rows],k = K_knn)
    # Calculate CV error
    cv_error[i] <- mean(pred_CV_test!=train_label[CV_test_rows])
    cm <- confusionMatrix(data = as.factor(pred_CV_test), reference = as.factor(train_label[CV_test_rows
                      positive = "TRUE")
    sensitives[i] = cm$byClass["Sensitivity"]
  }
  senses[i] = mean(sensitives)
  return(mean(cv_error))
}
```

```
K_fold <- 10
K_knn <- 1:50
cv_error <- rep(0,length(K_knn))
senses <- rep(0,length(K_knn))
for(i in 1:length(K_knn)){
  cv_error[i] <- Kfold_CV_knn(K = K_fold, K_knn = K_knn[i],train = train_x,train_label = train_label)
}
```

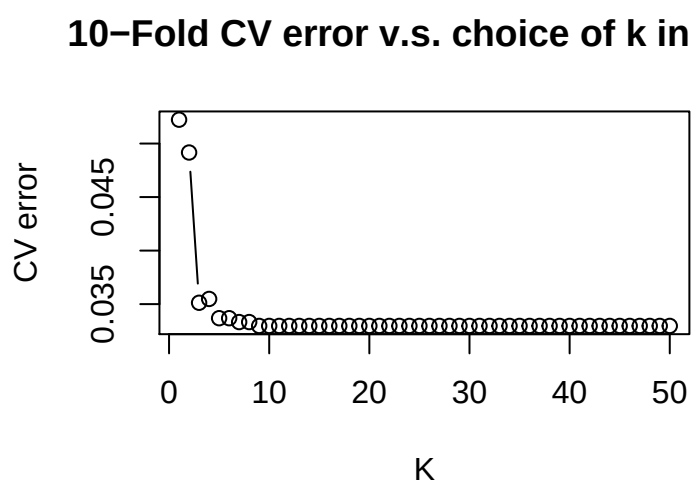
```
min(cv_error)
```

```
## [1] 0.03296266
```

```
best_k = which(cv_error == min(cv_error))
best_k
```

```
##  [1]  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
## [26] 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
```

```
plot(cv_error~K_knn,type='b',main = '10-Fold CV error v.s. choice of k in KNN',xlab = 'K',ylab = 'CV er
```




```
pred_train <- knn(train_x, train_x, train_label,
                  k = 2)
pred_test <- knn(train_x, test_x, train_label, k = 2)
```

```
tp <- 6
fn <- 65
fp <- 54

(recall <- tp/(tp+fn))
```

```
## [1] 0.08450704
```

```
(precision <- tp/(tp + fp))
```

```
## [1] 0.1
```

```
(f1 <- 2*precision*recall/(precision+recall))
```

```
## [1] 0.09160305
```

```
#confusionMatrix(pred_train, as.factor(train_label), positive = "TRUE")
#mean(pred_train == train_label)

confusionMatrix(pred_test, as.factor(test_label), positive = "TRUE")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE  2259   63
##      TRUE     50    8
```

```
##
##                Accuracy : 0.9525
##                  95% CI : (0.9432, 0.9607)
##     No Information Rate : 0.9702
##     P-Value [Acc > NIR] : 1.000
##
##                   Kappa : 0.0999
##
##  Mcnemar's Test P-Value : 0.259
##
##             Sensitivity : 0.112676
##             Specificity : 0.978346
##          Pos Pred Value : 0.137931
##          Neg Pred Value : 0.972868
##              Prevalence : 0.029832
##          Detection Rate : 0.003361
##    Detection Prevalence : 0.024370
##       Balanced Accuracy : 0.545511
##
##        'Positive' Class : TRUE
##
```

```
mean(pred_test == test_label)
```

```
## [1] 0.952521
```

try weighted KNN

https://search.r-project.org/CRAN/refmans/kknn/html/kknn.html

```
Kfold_CV_kknn <- function(K,K_knn,train,train_label, kern){
  fold_size <- floor(nrow(train)/K)
  cv_error <- rep(0,K)
  sensitive <- rep(0,K)
  for(i in 1:K){
    # select K-1 folds
    if(i!=K){
      CV_test_rows = ((i-1)*fold_size+1):(i*fold_size)
    }else{
      CV_test_rows = ((i-1)*fold_size+1):nrow(train)
    }
    CV_train = train[-CV_test_rows,]
    CV_test = train[CV_test_rows,]
    # Fit knn
    fit.kknn = kknn(dv.hypertension1 ~., train = CV_train, test = CV_test,k = K_knn,
                    kernel = kern, distance = 2)
    pred_CV_test <- fit.kknn$fitted.values
    # Calculate error
    cv_error[i] = mean(pred_CV_test!=train_label[CV_test_rows])
    cm <- confusionMatrix(data = pred_CV_test, reference = train_label[CV_test_rows],
                      positive = "TRUE")
    sensitive[i] = cm$byClass["Sensitivity"]
```

```
  }
  return(mean(sensitive))
}
```

```
K_fold <- 5
K_knn <- 3:25
kernels <- c("triangular", "epanechnikov", "optimal", "gaussian", "rectangular")
sensitives <- rep(0,length(K_knn))
train$dv.hypertension1 <- as.factor(train$dv.hypertension1)
for(kerns in kernels) {
  sensitives <- rep(0,length(K_knn))
  for(i in 1:length(K_knn)){
    kval<-K_knn[i]
    sensitives[i] <- Kfold_CV_kknn(K = K_fold, K_knn = kval,train = train,
                                   train_label = train$dv.hypertension1, kern=kerns)
  }
  best_k <- which(sensitives == max(sensitives))
}
```

```
knn.fit <- kknn(dv.hypertension1~., train, test,k=3, kernel = "optimal", distance = 2)
confusionMatrix(data = knn.fit$fitted.values, reference = as.factor(test$dv.hypertension1),
                      positive = "TRUE")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE  2265   63
##      TRUE     44    8
##
##                Accuracy : 0.955
##                  95% CI : (0.9459, 0.963)
##     No Information Rate : 0.9702
##     P-Value [Acc > NIR] : 0.99998
##
##                   Kappa : 0.1076
##
##  Mcnemar's Test P-Value : 0.08184
##
##             Sensitivity : 0.112676
##             Specificity : 0.980944
##          Pos Pred Value : 0.153846
##          Neg Pred Value : 0.972938
##              Prevalence : 0.029832
##          Detection Rate : 0.003361
##    Detection Prevalence : 0.021849
##       Balanced Accuracy : 0.546810
##
##        'Positive' Class : TRUE
##
```

upsampled data

```
train_one_hot <- read.csv("train_hot_X_y.csv") %>% select(-X)
test_one_hot <- read.csv("test_hot_X_y.csv") %>% select(-X)

data <- train_one_hot %>% select(-dv.hypertension)
test_data <- test_one_hot %>% select(-dv.hypertension)
```

```
K_knn <- 1:50
senses <- rep(0, length(K_knn))
Kfold_CV_knn1 <- function(K,K_knn,train,train_label){
  fold_size <- floor(nrow(train)/K)
  cv_error <- rep(0,K)
  sensitives <- rep(0,K)
  for(i in 1:K){
    # select K-1 folds
    if(i!=K){
      CV_test_rows <- ((i-1)*fold_size+1):(i*fold_size)
    }else{
      CV_test_rows <- ((i-1)*fold_size+1):nrow(train)
    }
    CV_train = train[-CV_test_rows,]
    CV_test = train[CV_test_rows,]
    # normalize the CV_train and CV_test
    mean_CV_train <- colMeans(CV_train)
    sd_CV_train <- apply(CV_train,2,sd)

    CV_train <- scale(CV_train,center = mean_CV_train,scale = sd_CV_train)
    CV_test <- scale(CV_test,center = mean_CV_train,scale = sd_CV_train)
    # Fit knn
    pred_CV_test <- knn(CV_train,CV_test,train_label[-CV_test_rows],k = K_knn)
    # Calculate CV error
    cv_error[i] <- mean(pred_CV_test!=train_label[CV_test_rows])
    cm <- confusionMatrix(data = as.factor(pred_CV_test),
                          reference = as.factor(train_label[CV_test_rows]), positive = "yes")
    sensitives[i] <- cm$byClass["Sensitivity"]
  }
  senses[i] <- mean(sensitives)
  return(mean(cv_error))
}
```

```
K_fold <- 10
K_knn <- 1:50
cv_error <- rep(0,length(K_knn))
for(i in 1:length(K_knn)){
  cv_error[i] <- Kfold_CV_knn1(K = K_fold, K_knn = K_knn[i],train = data,
                               train_label = train_one_hot$dv.hypertension)
}
```

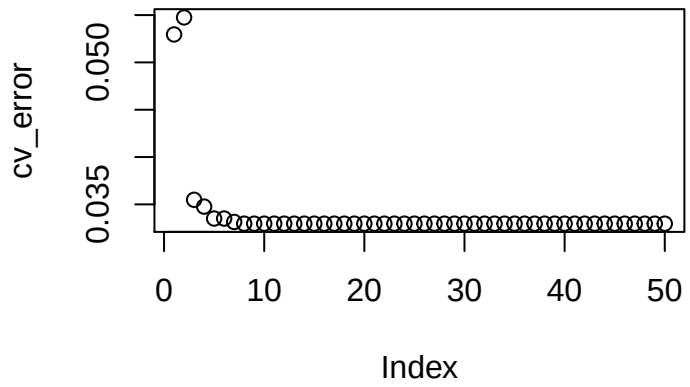
```
print(min(cv_error))
```

```
## [1] 0.03296266
```

```
best_k = which(cv_error == min(cv_error))
print(best_k)
```

```
##  [1]  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
## [26] 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
```

```
plot(cv_error)
```




```
kfit <- knn(train = data, test = test_data, cl = train_one_hot$dv.hypertension, k = 2)
confusionMatrix(data = kfit, reference = test_one_hot$dv.hypertension,
                      positive = "yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  2250   65
##        yes   59    6
##
##                Accuracy : 0.9479
##                  95% CI : (0.9382, 0.9565)
##     No Information Rate : 0.9702
##     P-Value [Acc > NIR] : 1.0000
##
##                   Kappa : 0.0615
##
##  Mcnemar's Test P-Value : 0.6534
##
##             Sensitivity : 0.084507
##             Specificity : 0.974448
##          Pos Pred Value : 0.092308
##          Neg Pred Value : 0.971922
##              Prevalence : 0.029832
```

```
##          Detection Rate : 0.002521
##    Detection Prevalence : 0.027311
##       Balanced Accuracy : 0.529477
##
##        'Positive' Class : yes
##
```