PSTAT131 Final Project

Liangchen Xia

2022/5/30

Introduction

In recent years, with the prosperity and development of medical technology and the explosive increase of medical knowledge, the content that medical staff need to learn is also increasing. In the process of medication, it is difficult to ensure the appropriateness, economy, effectiveness and safety of medication. Unreasonable and unsafe drug use is common, and medical staff need to pay attention to the supervision of safe drug use both in patients with chronic diseases and acute diseases.
Therefore, how to use model technology to predict which drug a patient should use according to some characteristics and indicators of the patient is particularly important for the development of medicine. First of all, it can help reduce the workload of medical staff. When the condition is simple, it can automatically predict the corresponding treatment drugs. Secondly, it can help each other to verify whether the diagnosis of the medical staff is correct. In this context, this paper uses a common machine learning algorithm to effectively predict which drug a patient should use based on some patient indicators.

Loading Data and Packages

The data for this project comes from the kaggle competition platform see https://www.kaggle.com/datasets/prathamtripathi/drug-classification. The sample size is not large, with a total of 200 pieces of data, and there are five independent variables. The predictor variable is the category of the patient's medication we want to study, that is, the variable Drug The feature sets are:
Age–patient's age
Sex–patient's sex
BP–the blood pressure levels of patient
Na_to_K–the sodium to potassium ration in blood of patient
Cholesterol–the cholesterol levels of patient

```
library(tidyverse)
library(tidymodels)
library(corrplot)
library(caret)
library(janitor)
library(skimr)
library(patchwork)
library(lubridate)
library(ranger)
library(rlang)
library(ggplot2)
library(corrr)
library(klaR)
library(MASS)
```

```
library(discrim)
library(installr)
library(kernlab)
library(kknn)
tidymodels_prefer()

setwd('D:/Desktop/final project')
drug <- read_csv("drug200.csv")
```

Data Cleaning

1. checking missing values

We confirmed that none of the variables have missing values.

```
head(drug)
```

```
## # A tibble: 6 x 6
##     Age Sex   BP     Cholesterol Na_to_K Drug
##   <dbl> <chr> <chr>  <chr>         <dbl> <chr>
## 1    23 F     HIGH   HIGH          25.4  DrugY
## 2    47 M     LOW    HIGH          13.1  drugC
## 3    47 M     LOW    HIGH          10.1  drugC
## 4    28 F     NORMAL HIGH           7.80 drugX
## 5    61 F     LOW    HIGH          18.0  DrugY
## 6    22 F     NORMAL HIGH           8.61 drugX
```

```
#checking missing values
apply(is.na(drug), 2, sum)
```

```
##         Age         Sex          BP Cholesterol     Na_to_K        Drug
##           0           0           0           0           0           0
```

2. Clean names

Convert all uppercase letters of a variable to lowercase.

```
drug <- drug %>%
  clean_names()
```

3. Convert to factor or numeric

Because the categorical variables of the data are stored in character type, we need to convert the categorical variables into factor type and then into numerical type, and the predictor variable drug category into factor type.

```
drug0<-drug
drug0$sex<-as.numeric(as.factor(drug0$sex))
drug0$bp<-as.numeric(as.factor(drug0$bp))
drug0$cholesterol<-as.numeric(as.factor(drug0$cholesterol))
drug0$drug<-as.factor(drug0$drug)
str(drug0)
```

```
## spec_tbl_df [200 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ age        : num [1:200] 23 47 47 28 61 22 49 41 60 43 ...
##  $ sex        : num [1:200] 1 2 2 1 1 1 1 2 2 2 ...
##  $ bp         : num [1:200] 1 2 2 3 2 3 3 2 3 2 ...
##  $ cholesterol: num [1:200] 1 1 1 1 1 1 1 1 1 2 ...
##  $ na_to_k    : num [1:200] 25.4 13.1 10.1 7.8 18 ...
##  $ drug       : Factor w/ 5 levels "drugA","drugB",..: 5 3 3 4 5 4 5 3 5 5 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Age = col_double(),
##   ..   Sex = col_character(),
##   ..   BP = col_character(),
##   ..   Cholesterol = col_character(),
##   ..   Na_to_K = col_double(),
##   ..   Drug = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Data Split

The dataset contains a total of 200 samples,of which 80% of the data is used as training data about 159 and 20% of the data is used as test data about 41.
Stratified sampling was used based on drug.

```
set.seed(123)
drug_split <-drug0 %>%
  initial_split(drug0,prop = 0.8, strata = "drug")
drug_train <- training(drug_split)
drug_test <- testing(drug_split)
dim(drug_train) # 159 6
```

```
## [1] 159   6
```

```
dim(drug_test)  # 41  6
```

```
## [1] 41  6
```

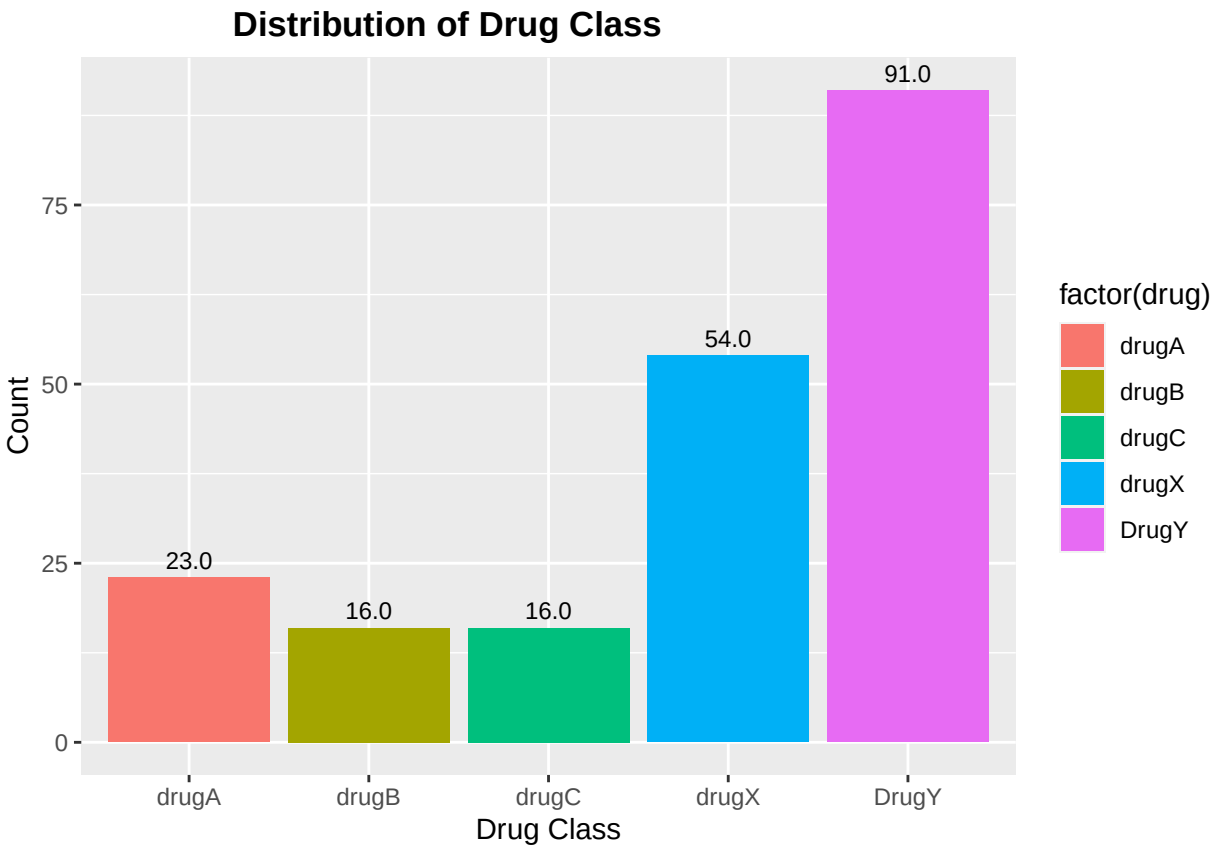
Exploratory Data Analysis

Distribution of Drug Class

By observing the proportion of drug class, we found that the proportion of drug Y is the highest, followed by drug X.

```
theme <- theme(plot.title = element_text(hjust = 0.3, face = "bold"))

drug%>%
  ggplot(aes(x = factor(drug),
             y = stat(count), fill = factor(drug),
             label = scales::comma(stat(count)))) +
  geom_bar(position = "dodge") +
```

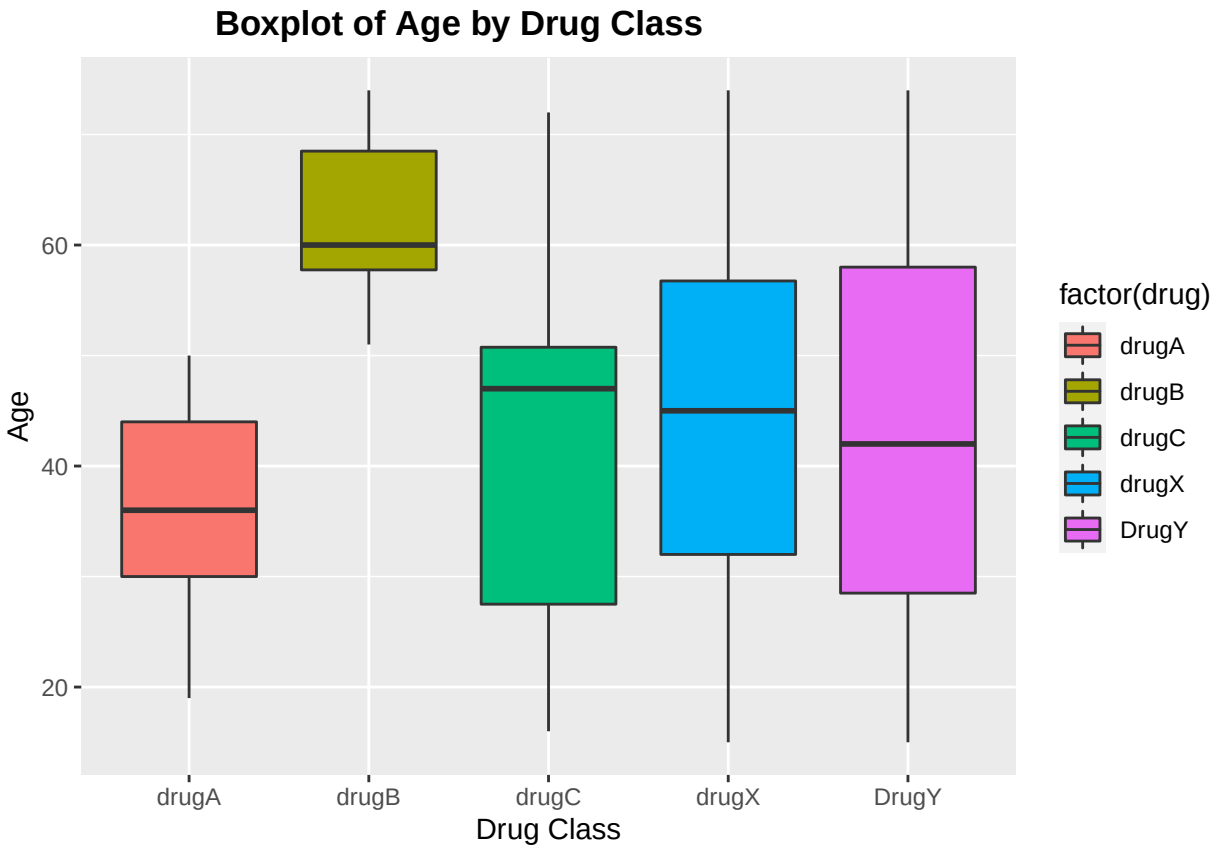
```
  geom_text(stat = 'count',
            position = position_dodge(.9),
            vjust = -0.5,
            size = 3) +
  scale_y_continuous(labels = scales::comma)+
  labs(x = 'Drug Class', y = 'Count') +
  ggtitle("Distribution of Drug Class") +
  theme
```



Distribution of Age by Drug Class

From the perspective of age distribution, the age of patients with drug B is significantly higher than that of other drugs, and it is mainly suitable for the elderly over 60.

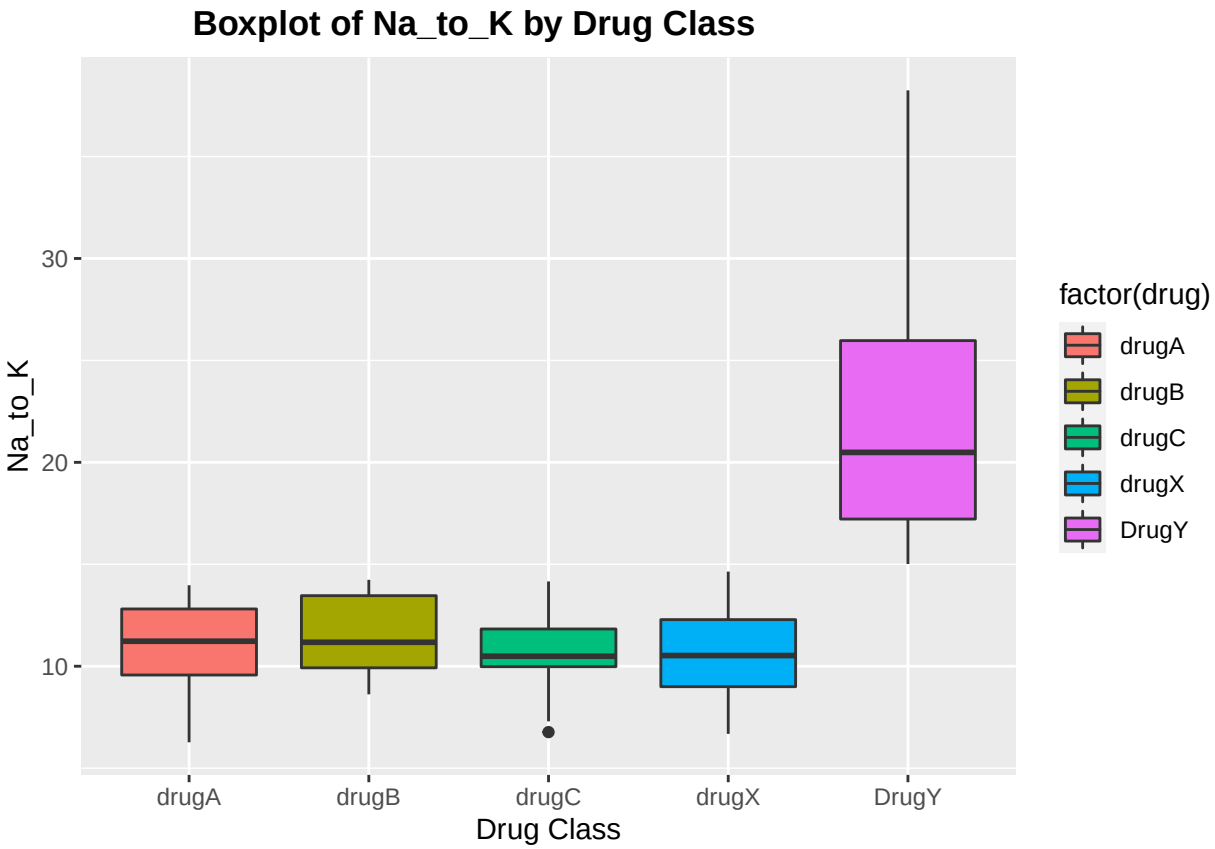
```
drug %>%
  ggplot(aes(x = as.factor(drug), y = age,fill=factor(drug))) +
  geom_boxplot() +
  labs(x = 'Drug Class', y = 'Age') +
  ggtitle("Boxplot of Age by Drug Class") +
  theme
```



Distribution of the sodium to potassium ration by Drug Class

From the distribution of the sodium to potassium ration in blood of the patients, the sodium-potassium ratio of the patients taking the Y drug was significantly higher than that of the patients taking other drugs.

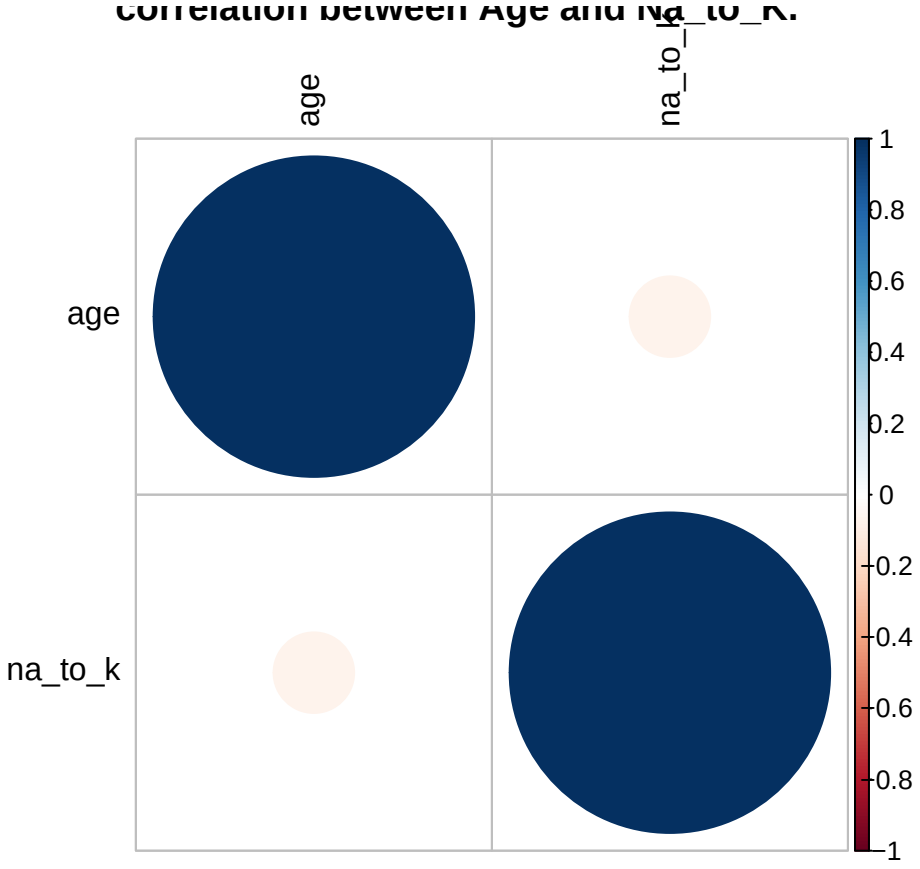
```
drug %>%
  ggplot(aes(x = factor(drug), y = na_to_k,fill=factor(drug))) +
  geom_boxplot() +
  labs(x = 'Drug Class', y = 'Na_to_K') +
  ggtitle("Boxplot of Na_to_K by Drug Class") +
  theme
```



correlation between Age and Na_to_K

We found a weak negative correlation between the sodium to potassium ratio in blood and age in patients.

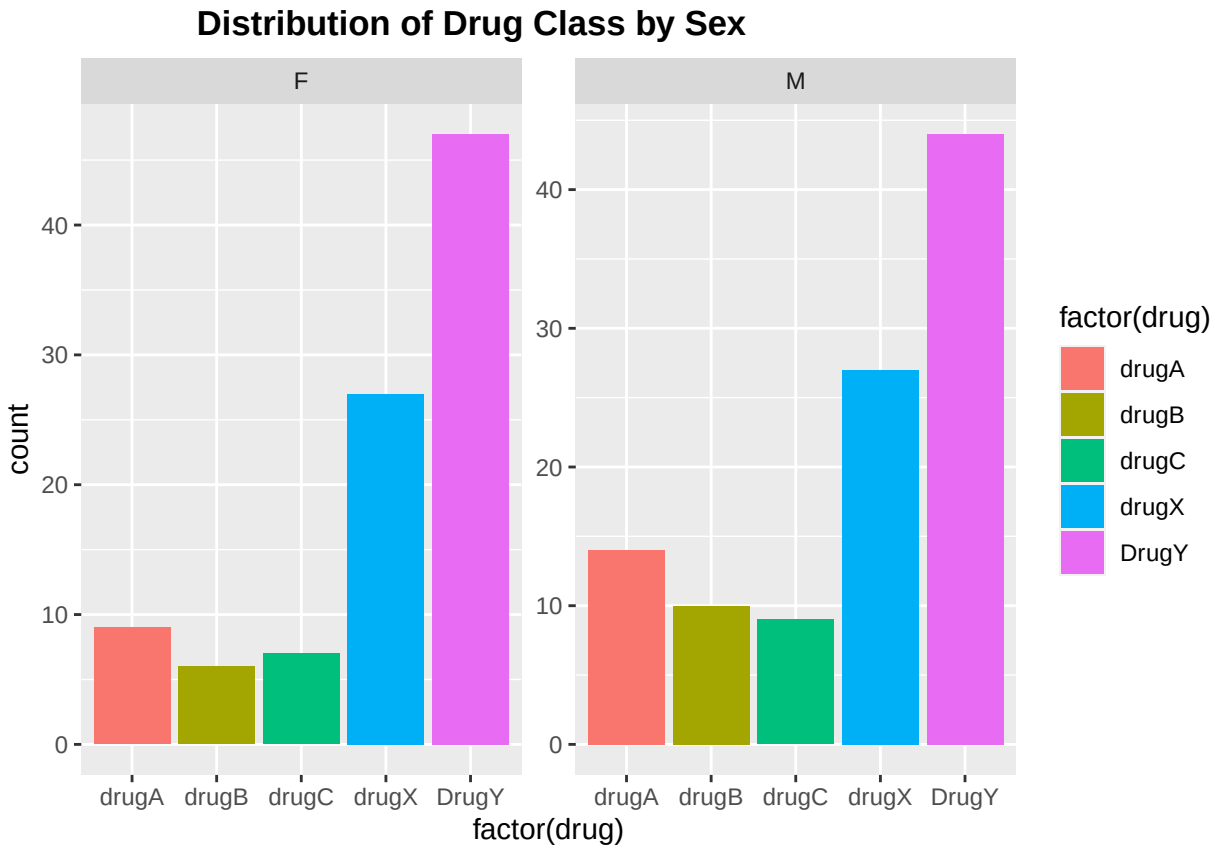
```
correlations <- cor(drug[,c('age','na_to_k')],method="pearson") %>%
corrplot( number.cex = .9, method = "circle", type = "full",tl.col = "black",title ='correlation betwee
```



Distribution of Drug Class by Sex

The distribution of drug classes in male patients was not significantly different from that in females

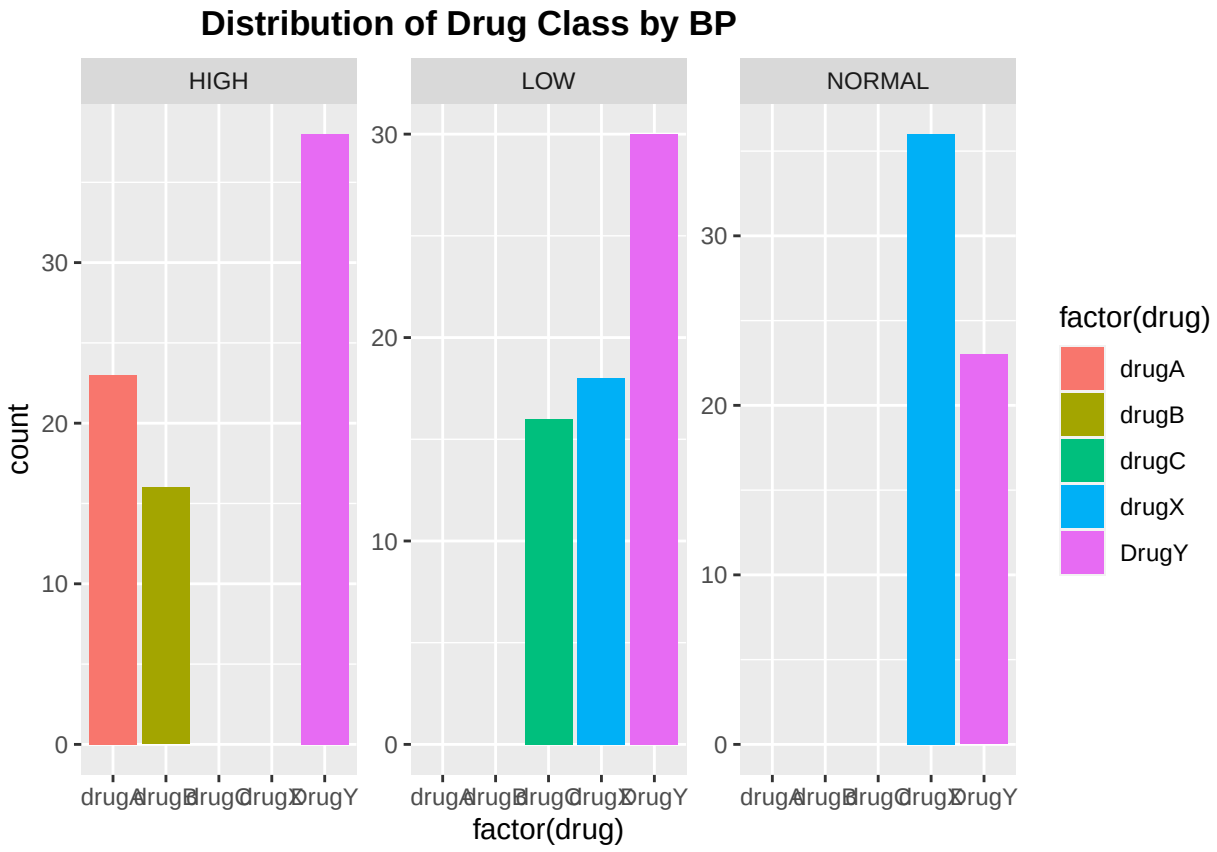
```
drug %>%
  ggplot(aes(x = factor(drug),
             y = stat(count), fill = factor(drug))) +
  geom_bar( ) +
  facet_wrap(~sex, scales = "free_y") +
  labs(
    title = "Distribution of Drug Class by Sex"
  )+
  theme
```



Distribution of Drug Class by blood pressure levels

Different blood pressure levels, the drug distribution for patients is significantly different. For example, high blood pressure levels patients and low blood pressure levels patients are mainly suitable for drug Y, while normal blood pressure levels patients are mainly suitable for drug X.

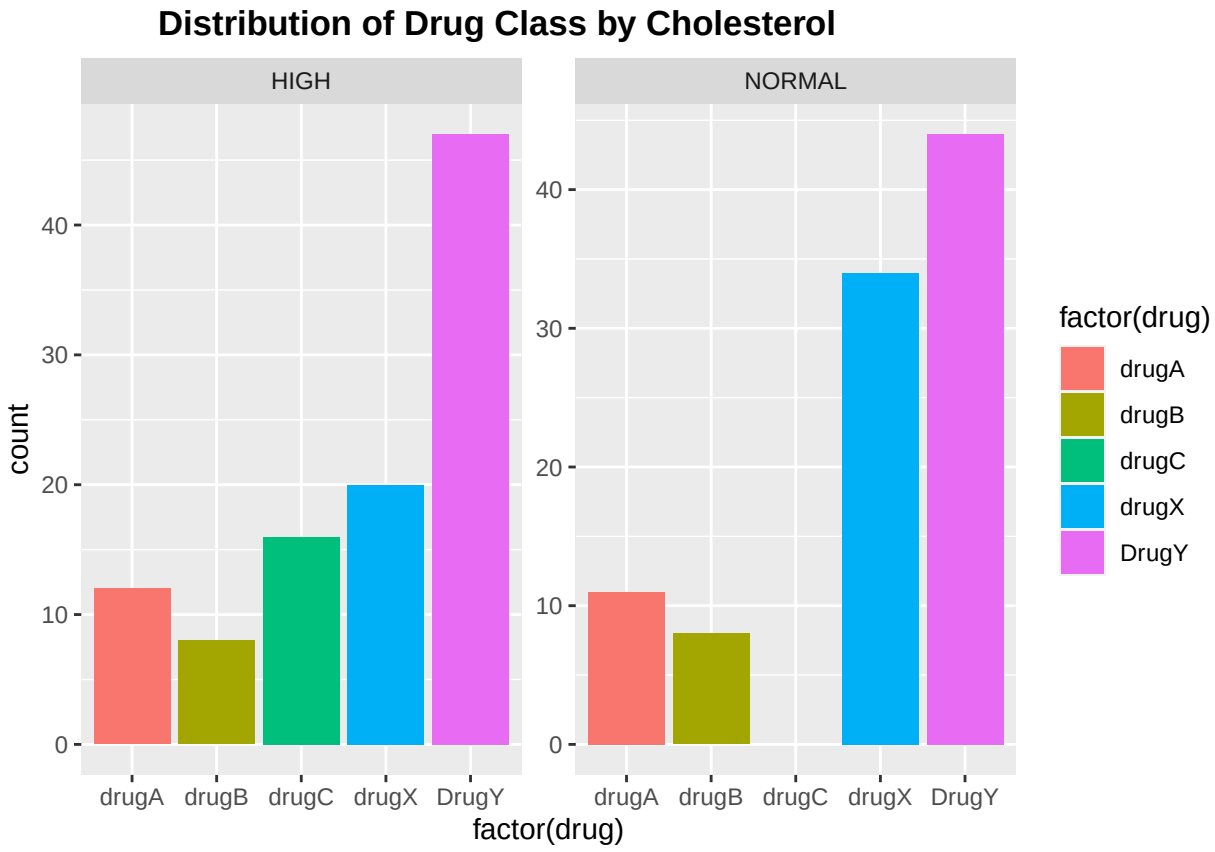
```
drug %>%
  ggplot(aes(x = factor(drug),
             y = stat(count), fill = factor(drug))) +
  geom_bar( ) +
  facet_wrap(~bp, scales = "free_y") +
  labs(
    title = "Distribution of Drug Class by BP"
  )+
  theme
```



Distribution of Drug Class by Cholesterol

Patients with normal cholesterol did not use drug C, and patients with high cholesterol were significantly more likely to use drug C than normal patients.

```
drug %>%
  ggplot(aes(x = factor(drug),
             y = stat(count), fill = factor(drug))) +
  geom_bar( ) +
  facet_wrap(~cholesterol, scales = "free_y") +
  labs(
    title = "Distribution of Drug Class by Cholesterol"
  )+
  theme
```



Model Building

Data prepare

We first normalize the continuous variables of the training set and test set respectively.

```
train<-drug_train
train[,c('age','na_to_k')] <- scale(train[,c('age','na_to_k')])
str(train)
#View(train)
test<-drug_test
test[,c('age','na_to_k')] <- scale(test[,c('age','na_to_k')])
str(test)
```

Model select

I decided to run cross fold validation but not repeat on the following four models.
1.Boosted Tree Model
2.SVM model
3.Nearest Neighbors model
4.Random forest model

Building the Recipe and Tweaking The Data Due to the small amount of our data, when designing cross fold validation, after many attempts, we finally found that v=3 is the best value. Once the V value is

greater than 3, it will appear that the categories cannot appear at the same time and cannot appear in the cross-validation model. Validation caused the model to report an error.

```
train_folds <- vfold_cv(train, v = 3,repeats = 3)
recipe<-recipe(drug ~ age+sex+bp+cholesterol+na_to_k, data = train)
recipe
```

Boosted Tree Model I tuned min_n and mtry, set mode to "classification" (because my outcome is a factor variable), and used the xgboost engine.

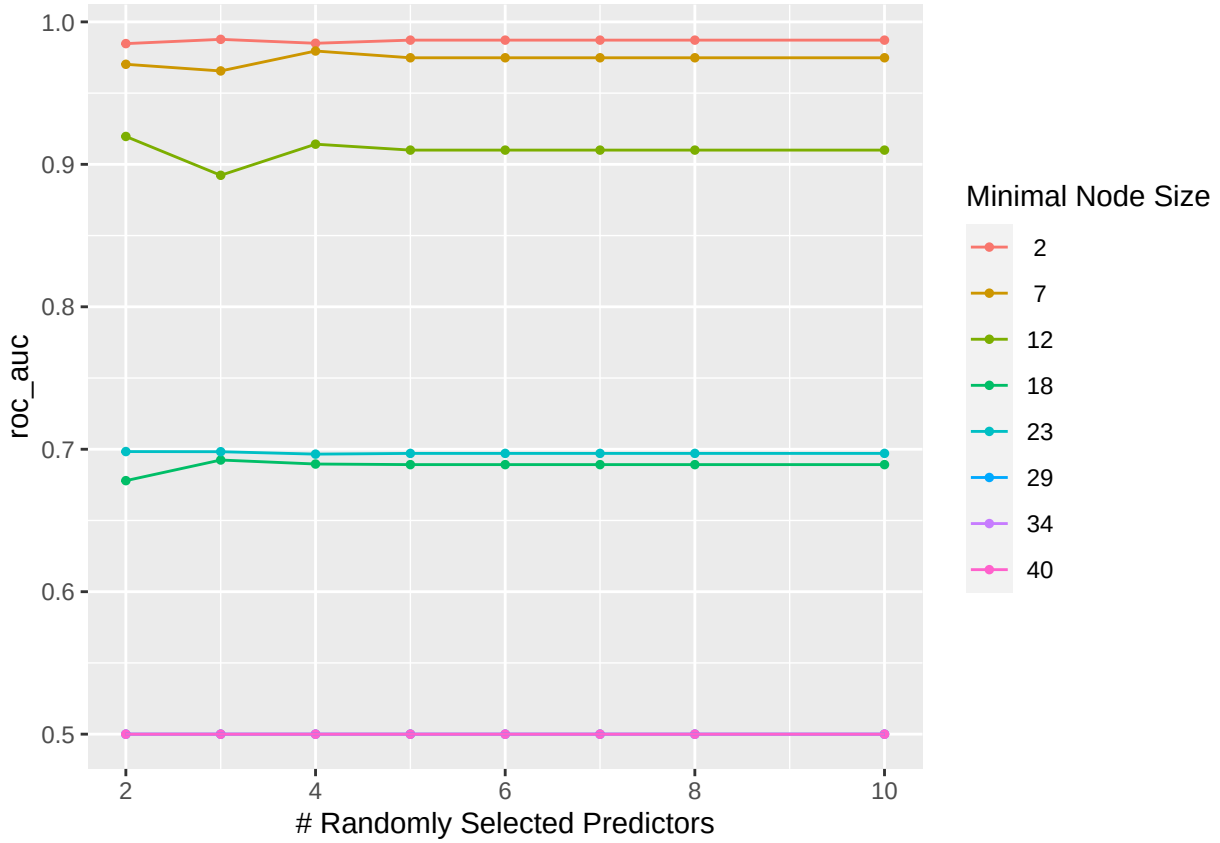
```
bt_model <- boost_tree(mode = "classification",
                       min_n = tune(),
                       mtry = tune()
                       ) %>%
  set_engine("xgboost")%>%
  translate()

bt_workflow <- workflow() %>%
  add_model(bt_model) %>%
  add_recipe(recipe)

bt_params <- extract_parameter_set_dials(bt_model) %>%
  update(mtry = mtry(range= c(2, 10)),
         min_n = min_n(range= c(2, 40))
  )

# define grid
bt_grid <- grid_regular(bt_params, levels = 8)
```

Then, I executed my model by tuning and fitting. This process took 10 minutes, it is quickly.

```
set.seed(123)
bt_res <- bt_workflow %>%
  tune_grid(resamples = train_folds,
            grid = bt_grid)
#save(bt_res, bt_workflow, file = "D:/Desktop/final project/bt_res2.rda")
```

Taking a quick peak at the autoplot() function and show_best() based on roc_auc metric.We found that when node=2, its mean roc_auc value is close to 98.8%

```
set.seed(123)
load("D:/Desktop/final project/bt_res2.rda")
autoplot(bt_res, metric = "roc_auc")  ##node=2 is best
```




```
show_best(bt_res, metric = "roc_auc") %>% select(-.estimator, -.config) #2 2
```

```
## # A tibble: 5 x 6
##    mtry min_n .metric  mean     n std_err
##   <int> <int> <chr>   <dbl> <int>   <dbl>
## 1     3     2 roc_auc 0.988     9 0.00492
## 2     5     2 roc_auc 0.987     9 0.00430
## 3     6     2 roc_auc 0.987     9 0.00430
## 4     7     2 roc_auc 0.987     9 0.00430
## 5     8     2 roc_auc 0.987     9 0.00430
```

We'll create a workflow that has tuned in the name, so we can identify it. We'll finalize the workflow by taking the parameters from the best model (the Boosted Tree regression model) using the select_best() function.

```
set.seed(123)
bt_workflow_tuned <- bt_workflow %>%
  finalize_workflow(select_best(bt_res, metric = "roc_auc"))
bt_final <- fit(bt_workflow_tuned, train)
```

SVM Model I tuned cost, set mode to "classification" (because my outcome is a factor variable), and used the kernlab engine.

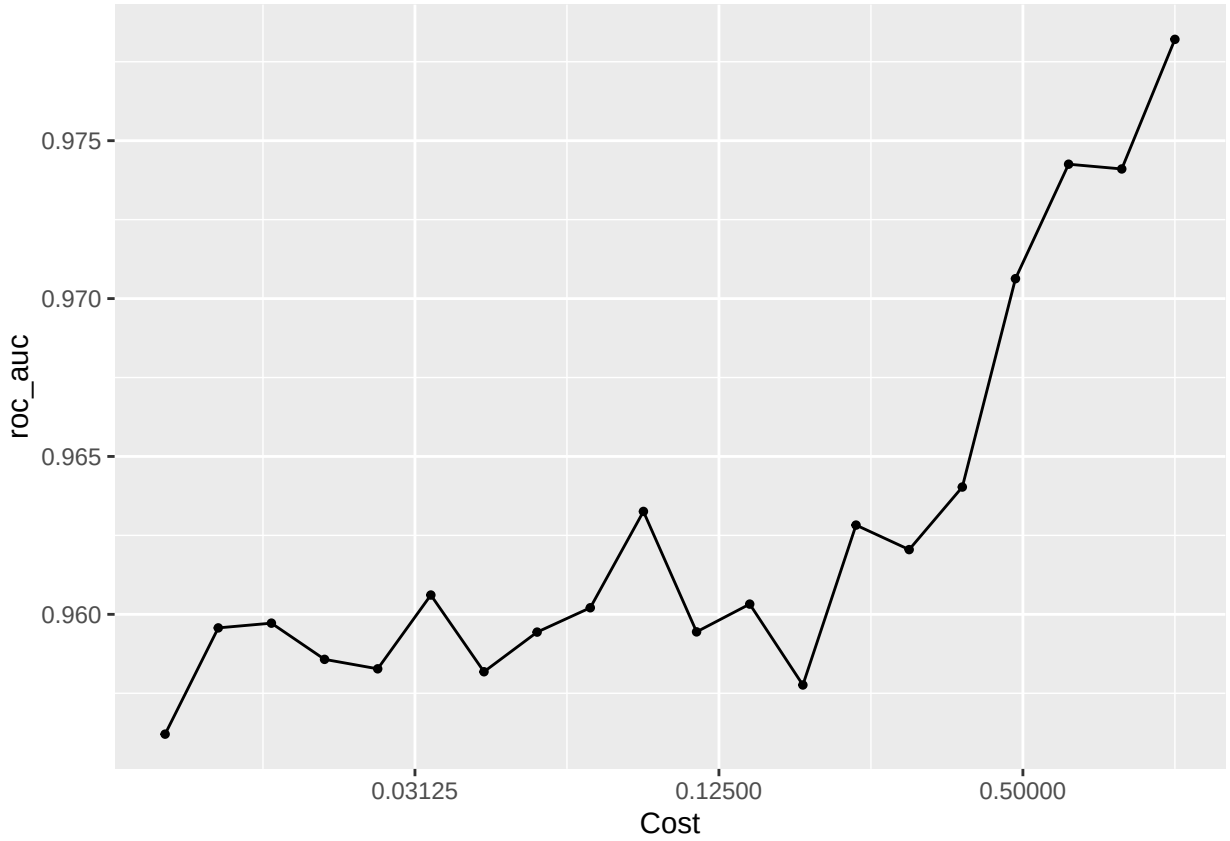
```
svm_model <- svm_rbf( cost  = tune()
                      ) %>%
  set_engine("kernlab")%>%
  set_mode('classification')%>%
  translate()

svm_workflow <- workflow() %>%
  add_model(svm_model) %>%
  add_recipe(recipe)

svm_grid <- tibble(cost = 10^seq(-2, 0, length.out = 20))
```

Then, I executed my model by tuning and fitting. This process took 5 minutes, it is quickly.

```
set.seed(123)
svm_res <- svm_workflow %>%
  tune_grid(resamples =train_folds,
            grid = svm_grid)
#save(svm_res, bt_workflow, file = "D:/Desktop/final project/svm_res2.rda")
```

Taking a quick peak at the autoplot() function and show_best() based on roc_auc metric.We found that when cost=1, its mean roc_auc value is close to 97.8%

```
set.seed(123)
load("D:/Desktop/final project/svm_res2.rda")
autoplot(svm_res, metric = "roc_auc")
```




```
show_best(svm_res, metric = "roc_auc") %>% select(-.estimator, -.config) #cost=1
```

```
## # A tibble: 5 x 5
##    cost .metric  mean     n std_err
##   <dbl> <chr>   <dbl> <int>   <dbl>
## 1 1     roc_auc 0.978     9 0.00427
## 2 0.616 roc_auc 0.974     9 0.00443
## 3 0.785 roc_auc 0.974     9 0.00410
## 4 0.483 roc_auc 0.971     9 0.00471
## 5 0.379 roc_auc 0.964     9 0.00535
```

We'll create a workflow that has tuned in the name, so we can identify it. We'll finalize the workflow by taking the parameters from the best model (the svm model) using the select_best() function.

```
set.seed(123)
svm_workflow_tuned <- svm_workflow %>% finalize_workflow(select_best(svm_res, metric = "roc_auc"))
svm_final <- fit(svm_workflow_tuned, train)
```

Nearest Neighbors Model I tuned neighbors, set mode to "classification" (because my outcome is a factor variable), and used the kknn engine.

```
knn_model <- nearest_neighbor(mode = "classification",
                              neighbors = tune()
) %>%
```

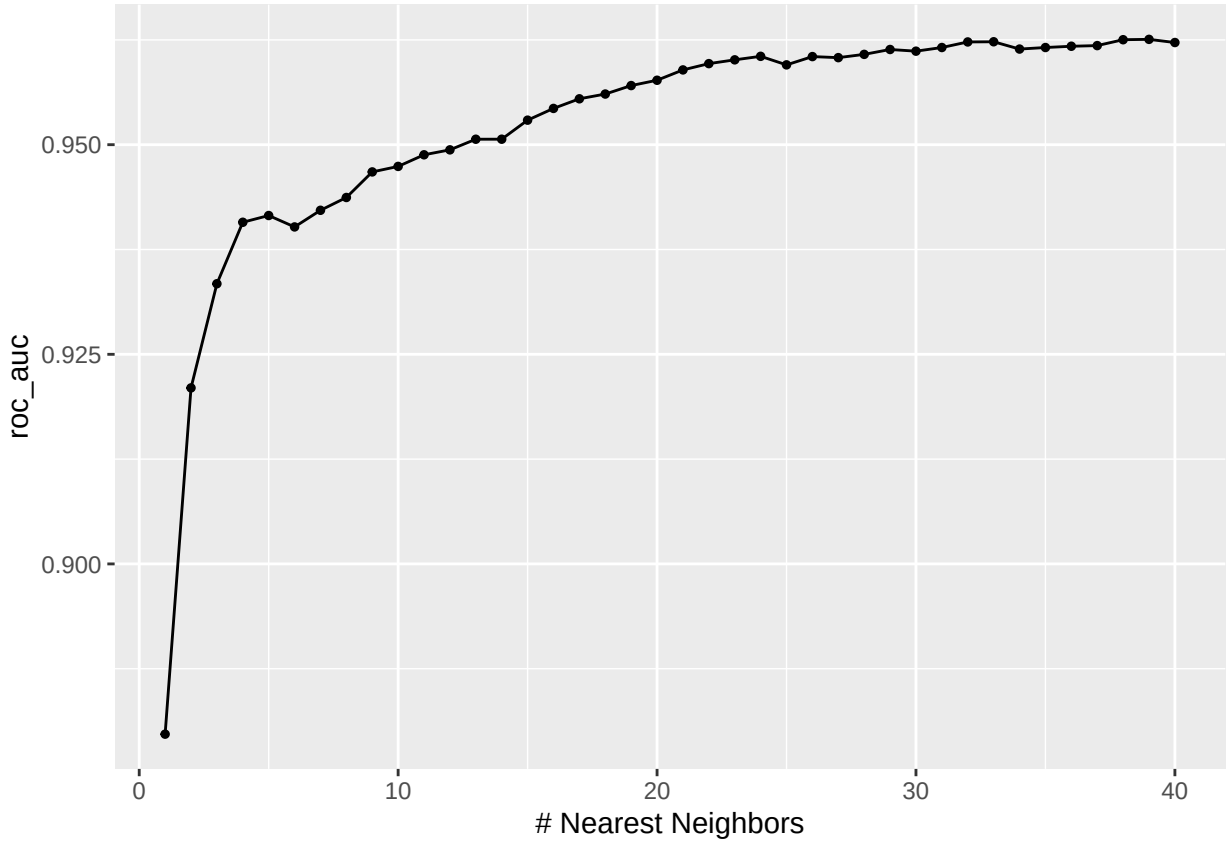
```
  set_engine("kknn")

knn_workflow <- workflow() %>%
  add_model(knn_model) %>%
  add_recipe(recipe)

# define grid
knn_grid <- tibble(neighbors = seq(1, 40))
```

Then, I executed my model by tuning and fitting. This process took 2 minutes, it is quickly.

```
set.seed(123)
knn_res <- knn_workflow %>%
  tune_grid(resamples = train_folds,
            grid = knn_grid
        )
#save(knn_res, bt_workflow, file = "D:/Desktop/final project/knn_res2.rda")
```

Taking a quick peak at the autoplot() function and show_best() based on roc_auc metric.We found that as K increases,roc_auc will also increase, but when the value of K reaches 20, the value of roc_auc will no longer change significantly,its mean roc_auc value is close to 95.8%.

```
set.seed(123)
load("D:/Desktop/final project/knn_res2.rda")
autoplot(knn_res, metric = "roc_auc")  ##
```



```
show_best(knn_res, metric = "roc_auc") %>% select(-.estimator, -.config)  #k=38
```

```
## # A tibble: 5 x 5
##   neighbors .metric  mean     n std_err
##       <int> <chr>   <dbl> <int>   <dbl>
## 1        39 roc_auc 0.963     9 0.00418
## 2        38 roc_auc 0.963     9 0.00420
## 3        33 roc_auc 0.962     9 0.00372
## 4        32 roc_auc 0.962     9 0.00362
## 5        40 roc_auc 0.962     9 0.00381
```

We'll create a workflow that has tuned in the name, so we can identify it. We'll finalize the workflow by taking the parameters from the best model (the knn model) using the select_best() function.

```
set.seed(123)
knn_workflow_tuned <- knn_workflow %>%
  finalize_workflow(select_best(knn_res, metric = "roc_auc"))
knn_final <- fit(knn_workflow_tuned, train)
```

Random Forest Model I tuned min_n, set mode to "classification" (because my outcome is a factor variable), and used the ranger engine.

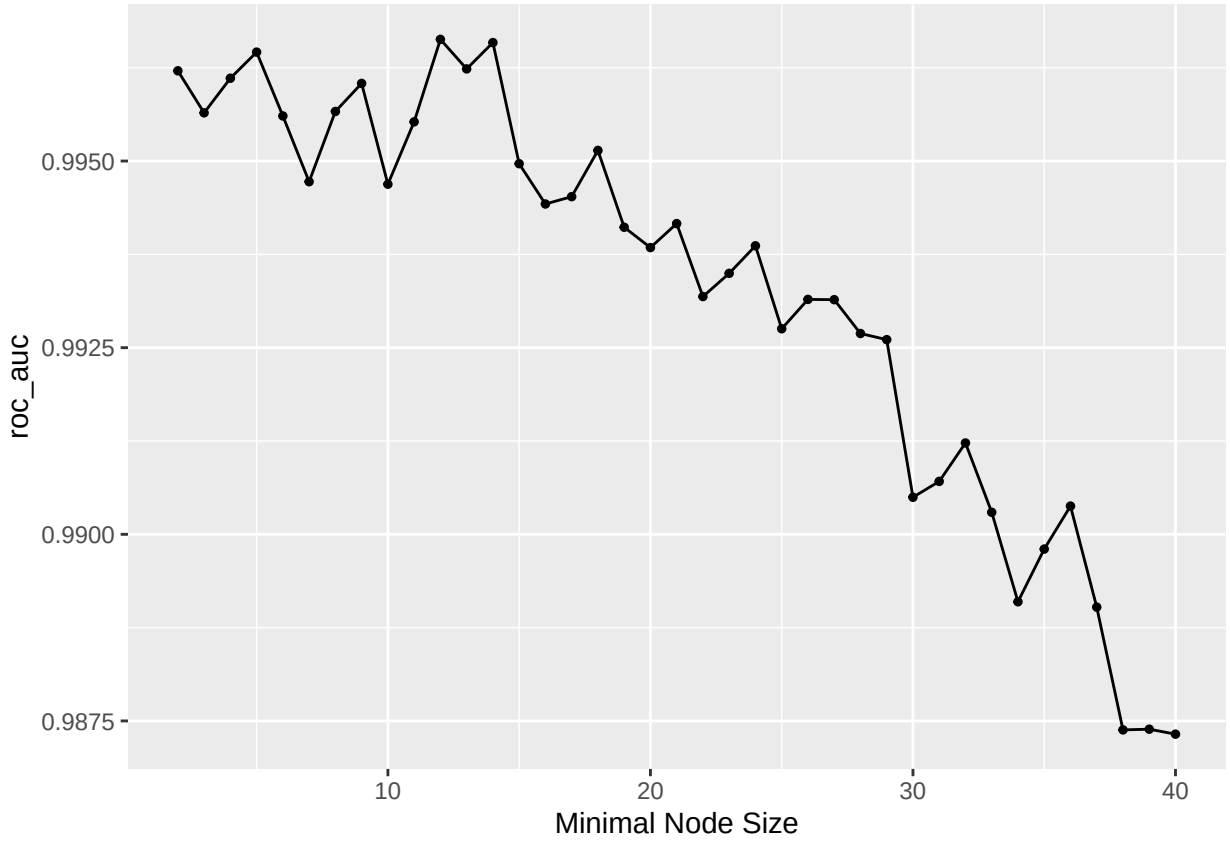
```
rf_model <- rand_forest(mode = "classification",
                        min_n = tune()
                        ) %>%
  set_engine("ranger")

rf_workflow <- workflow() %>%
  add_model(rf_model) %>%
  add_recipe(recipe)

# define grid  data is small,so only tune min_n
rf_grid <- tibble(min_n = seq(2, 40) )
```

Then, I executed my model by tuning and fitting. This process took 5 minutes, it is quickly.

```
set.seed(123)
rf_res <- rf_workflow %>%
  tune_grid(resamples = train_folds,
            grid = rf_grid
           )
#save(rf_res, bt_workflow, file = "D:/Desktop/final project/rf_res2.rda")
```

Taking a quick peak at the autoplot() function and show_best() based on roc_auc metric.We found that When node is 12, roc_auc has a maximum value, its mean roc_auc value is close to 99.7%, and then roc_auc shows a decreasing trend with the increase of node.

```
set.seed(123)
load("D:/Desktop/final project/rf_res2.rda")
autoplot(rf_res, metric = "roc_auc")  ##
```




```
show_best(rf_res, metric = "roc_auc") %>% select(-.estimator, -.config)
```

```
## # A tibble: 5 x 5
##   min_n .metric  mean     n std_err
##   <int> <chr>   <dbl> <int>   <dbl>
## 1    12 roc_auc 0.997     9 0.00201
## 2    14 roc_auc 0.997     9 0.00188
## 3     5 roc_auc 0.996     9 0.00231
## 4    13 roc_auc 0.996     9 0.00234
## 5     2 roc_auc 0.996     9 0.00230
```

Final Model Building

Let's continue with the Random Forest Model being the model that performed best,it's mean roc_auc value is highest,closing to 99.7% .

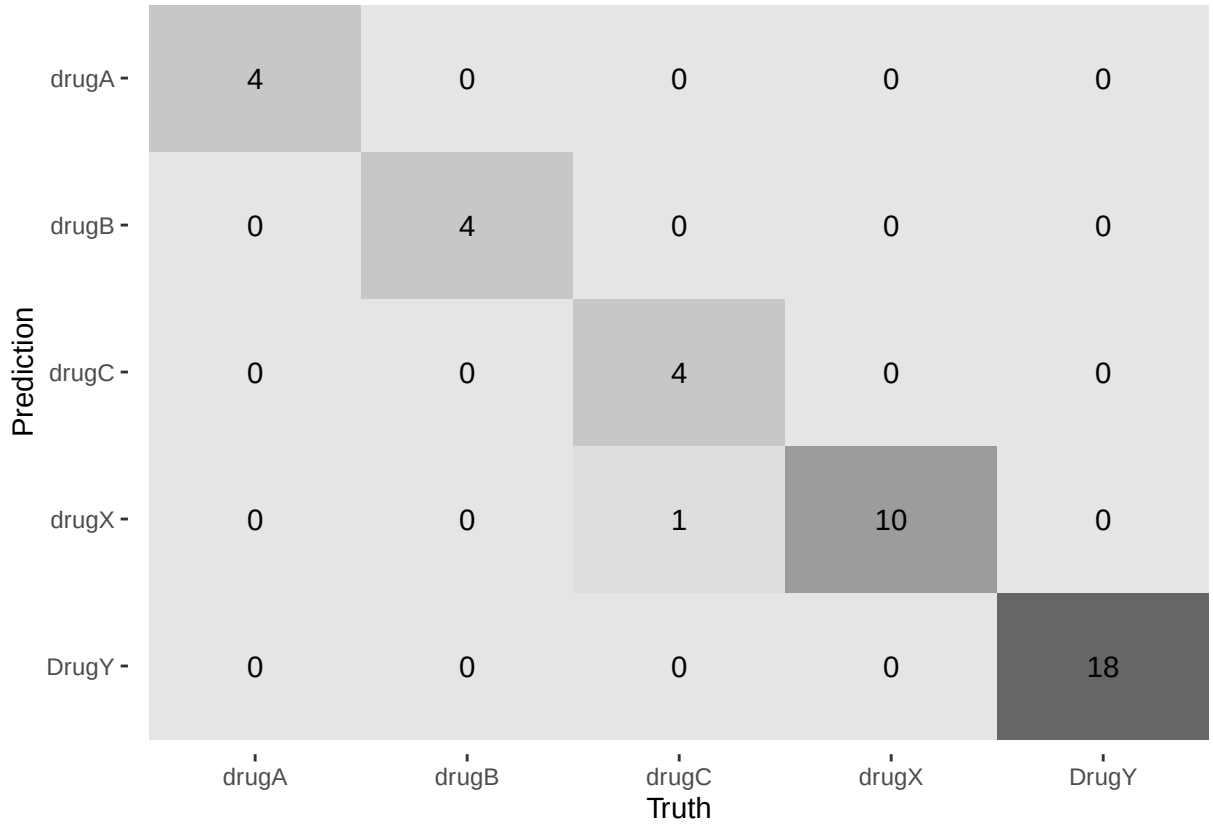
We'll create a workflow that has tuned in the name, so we can identify it. We'll finalize the workflow by taking the parameters from the best model (the random forest model) using the select_best() function.

```
set.seed(123)
rf_workflow_tuned <- rf_workflow %>%
  finalize_workflow(select_best(rf_res, metric = "roc_auc"))
rf_final <- fit(rf_workflow_tuned, train)
```

Analysis of The Test Set

Lets fit the model to the testing data set and plot confusion matrix and calculate accuracy.The test results show that, except for drug C, which has prediction errors, the predictions of other drug types have no errors, and the overall prediction accuracy rate is 97.5%,the model predicts well.

```
set.seed(123)
augment(rf_final, new_data = test) %>%
  conf_mat(truth = drug, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```




```
#calculate accuracy
rf_acc <- augment(rf_final, new_data = test) %>%
  accuracy(truth = drug, estimate = .pred_class)
rf_acc # 0.975
```

Conclusion

In order to study how to correctly predict which drug a patient should take, we first performed an exploratory analysis of variables, knowing that cholesterol levels, sodium-potassium ratios, and age had important effects on the types of drugs used.
Then We selected four commonly machine learning models, namely boosted tree,SVM, Nearest Neighbors, and random forest, and adjusted the parameters of each model. And then used the validation set to evaluate the effect of each model. According to the performance of roc_auc value, we believe that the random forest

model have the best prediction effect,its mean roc_auc value is close to 99.7%. The boosted tree model and svm model also performed well,the roc_auc value is close to 98.7% and 97.8% respectively. The nearest neighbors model is the worst performer among them,it's roc_auc value is close to 95.8%.
Finally, we use the test set to test the best performing random forest model. The test results show that, except for drug C, which has prediction errors, the predictions of other drug types have no errors, and the overall prediction accuracy rate is 97.5%,the model predicts well.
Overall, although the Drug dataset has only 200 samples, it still provides different patient information and corresponding drug usage, and has achieved fruitful results in how to predict which drug patients should use based on patient information.