Predicting milk quality with machine learning techniques

Muxi Li

2022-12-10

The University of California, Santa Barbara, CA 93106

Email: muxi@ucsb.edu (mailto:muxi@ucsb.edu)

Abstract

Coffee is an integral part of many people's lives. Among them, latte is very popular among many people. As a significant part of latte, milk plays a decisive role in the quality. The goal of the project is to provide guiding opinions for consumers to evaluate milk by analyzing the characteristics of it. This article will use a series of models to model the milk data. By comparing the accuracy of different models on the training data set, We found that random forest and boosted trees models performed better. Due to overfitting, we finally choose the random forest model to model the test set data and evaluate the model accuracy.

1. Introduction

As the pace of society accelerates, coffee plays an increasingly important role. As an iced latte lover, I believe the quality of the milk plays a decisive role. We got the milk dataset from https://www.kaggle.com/ (https://www.kaggle.com/). We will use these data to build classification models and compare the advantages and disadvantages of different models. Let's start now!

2. Data and Packages

Packages

Let's start by loading packages. This article will use the following diagrams, model packages:

```
#input packages
library(ISLR)
library(ISLR2)
library(tidyverse)
library(tidymodels)
library(readr)
library(corrr)
library(corrplot)
library(discrim)
library(klaR)
library(tune)
library(rpart)
library(rpart.plot)
library(vip)
library(janitor)
library(randomForest)
library(xgboost)
library(kernlab)
library(kknn)
library(ggthemes)
library(ggplot2)
tidymodels_prefer()
```

Data

We now focus on the basic characteristics of the data. A series of characteristics of the milk data are depicted as follows:

pH: This Column defines PH alus of the milk which ranges from 3 to 9.5.

Temperature: This Column defines Temperature of the milk which ranges from 34’C to 90’C.

Taste: This Column defines Taste of the milk which is categorical data 0 (Bad) or 1 (Good).

Odor: This Column defines Odor of the milk which is categorical data 0 (Bad) or 1 (Good).

Fat: This Column defines Odor of the milk which is categorical data 0 (Low) or 1 (High).

Turbidity: This Column defines Turbidity of the milk which is categorical data 0 (Low) or 1 (High).

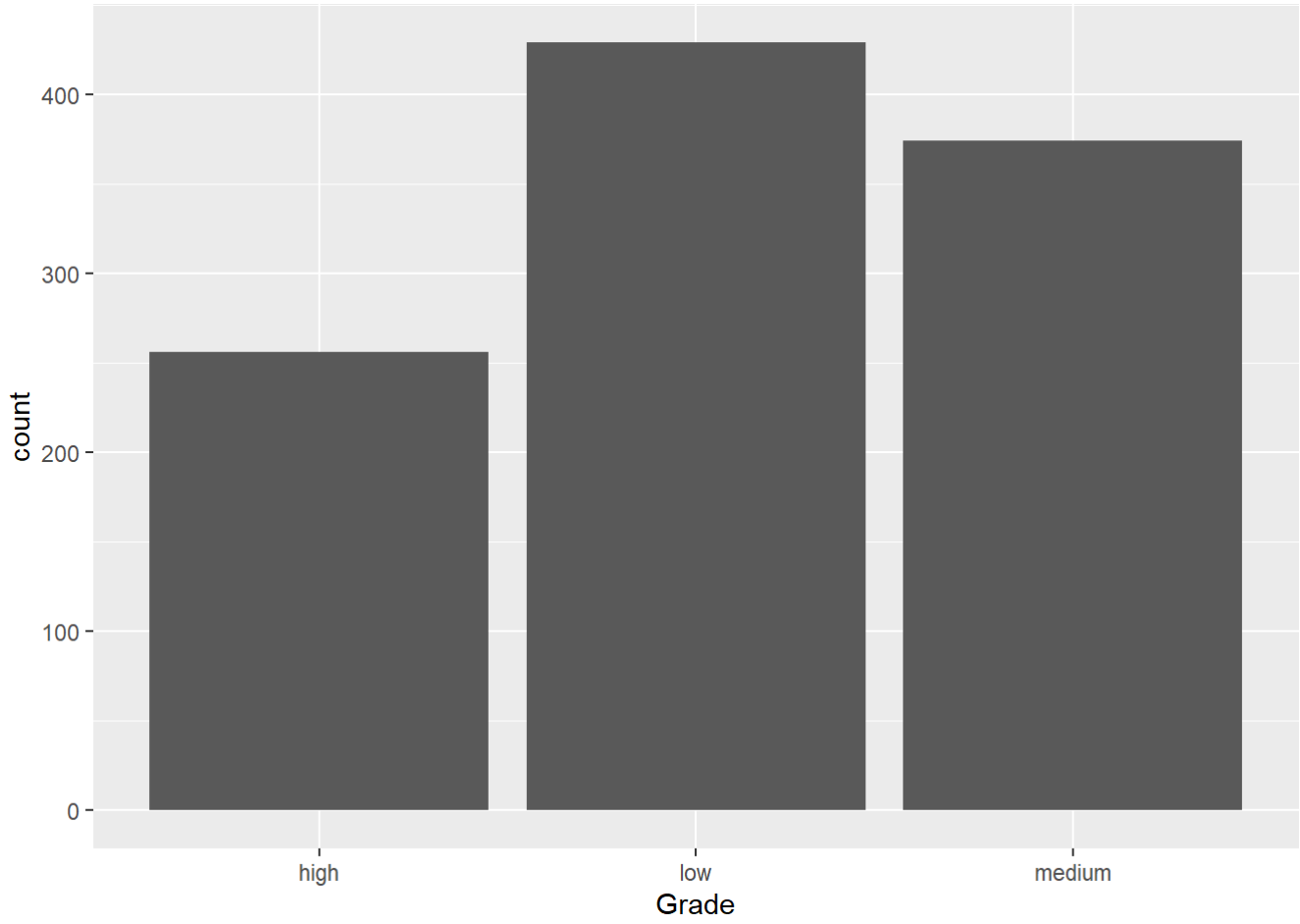
Colour: This Column defines Colour of the milk which ranges from 240 to 255.

Grade: This Column defines Grade (Target) of the milk which is categorical data Low (Bad) or Medium (Moderate) or High (Good).

From the definition we can easily see that, Taste, Odor, Fat, Turbidity and Colour are qualitative predictors. pH and Temperature are quantitative predictors. And Grade is qualitative response.

We first explore the distribution of our response variable – Grade.

```
# input data
milk=read_csv(file = "milk.csv",show_col_types = FALSE)
milk %>%
  ggplot(aes(x = Grade)) +
  geom_bar()
```



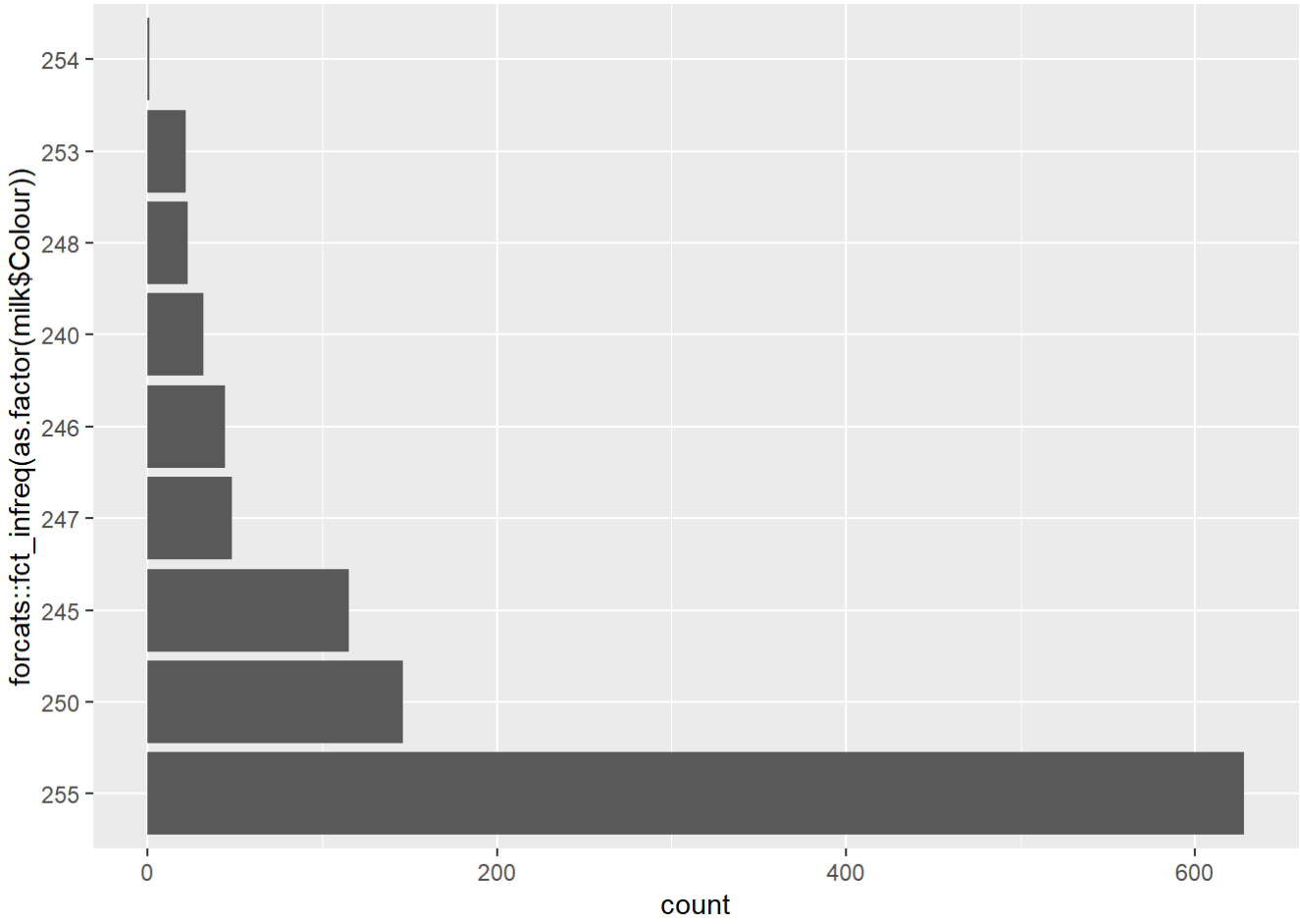
It is clear that the number of the three levels is relatively balanced. No rare class is noticed.

3. Exploratory Data Analysis

Let's quickly review our predictor variables. Two quantitative predictors and four binary qualitative predictors.

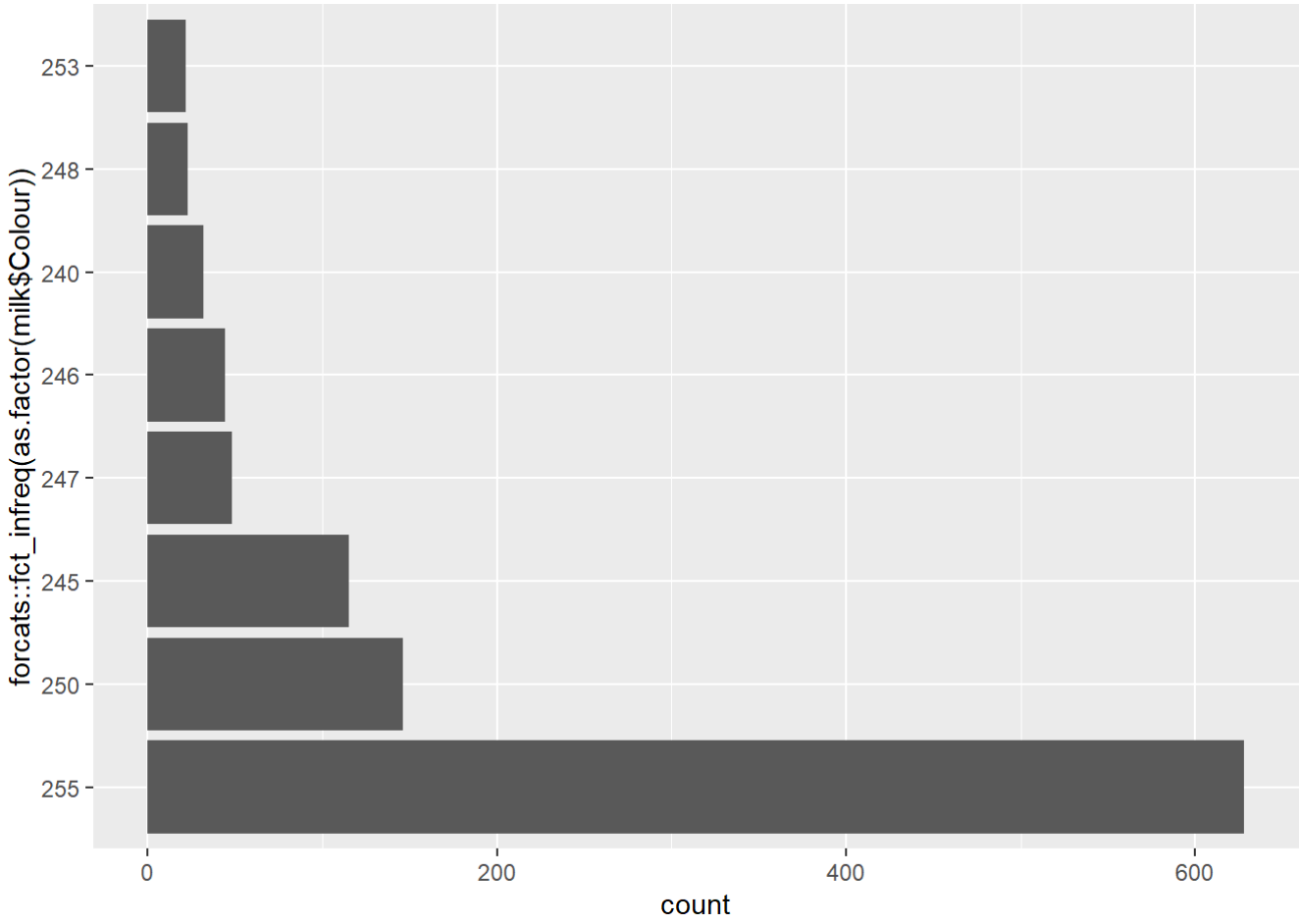
Let's now check the "Colour" distribution.

```
#show the distribution of Colour
milk %>%
  ggplot(aes(x = forcats::fct_infreq(as.factor(milk$Colour)))) +
  geom_bar() +
  coord_flip()
```



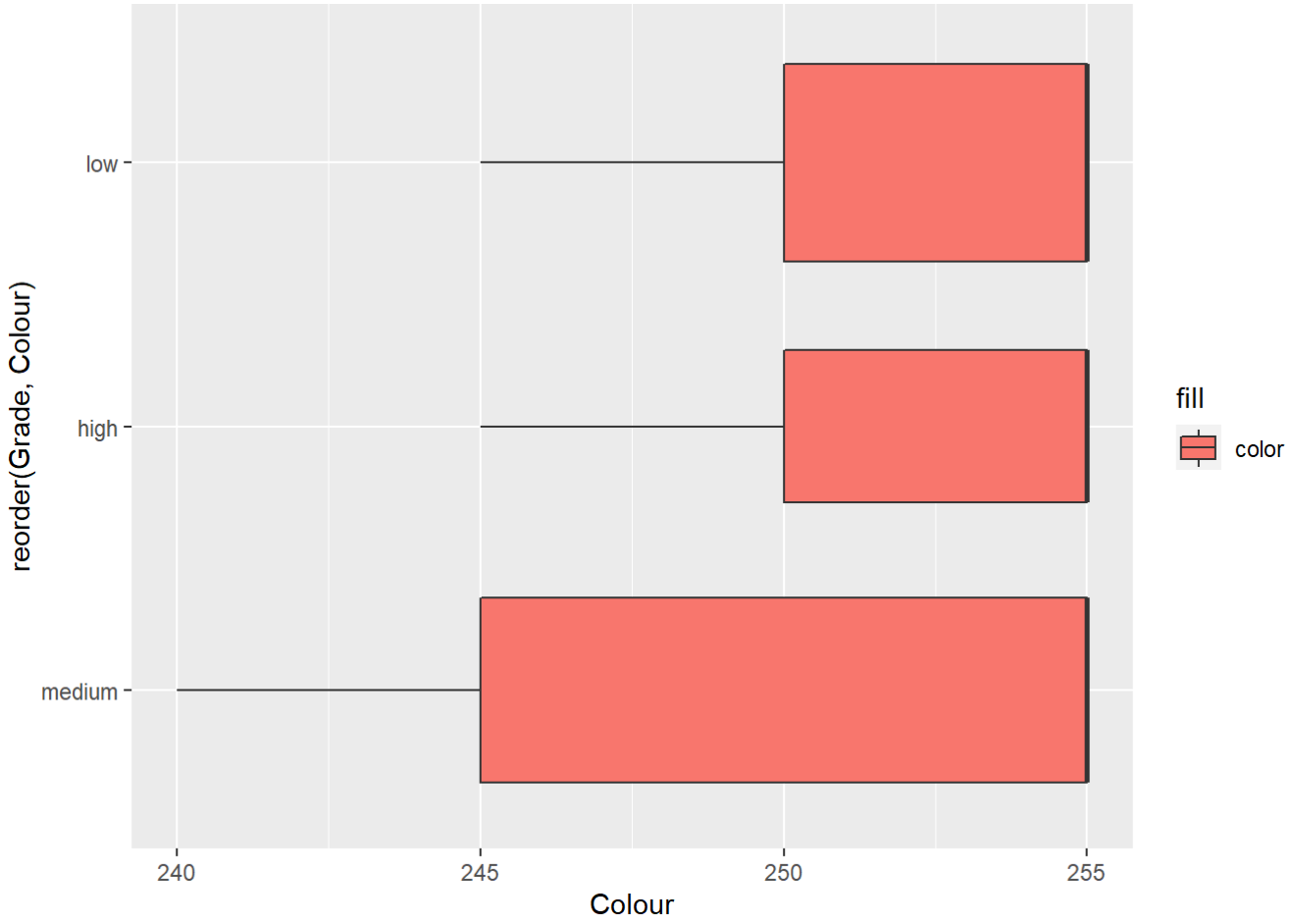
Everything looks reasonable except for the colour:254. I take the rare situation of colour:254 as the outlier. Now let's remove it and check our data again.

```
#check the distribution of Colour
milk=milk %>% filter(Colour!="254")
milk %>%
  ggplot(aes(x = forcats::fct_infreq(as.factor(milk$Colour)))) +
  geom_bar() +
  coord_flip()
```



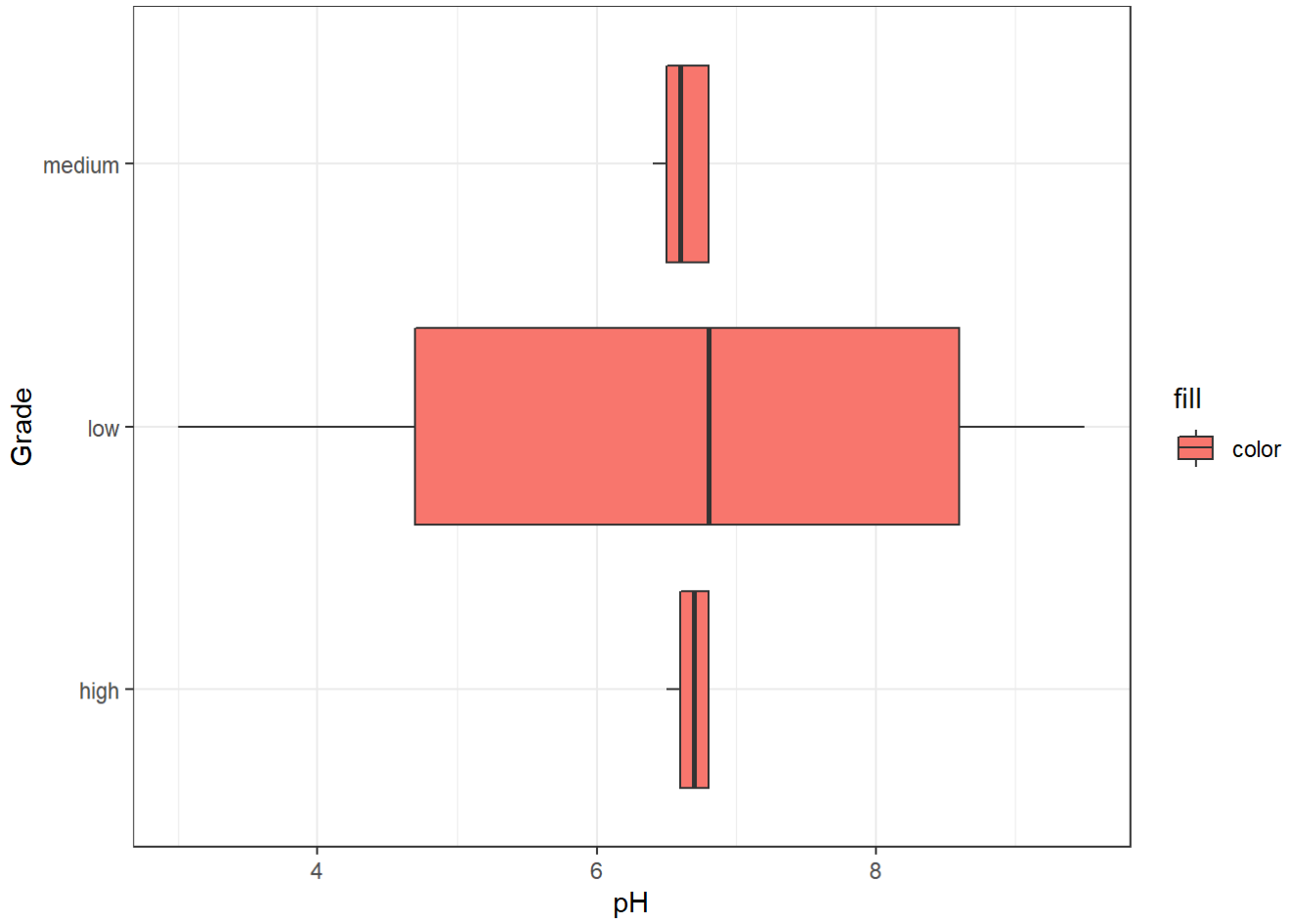
Also, let's check the boxplot.

```
#check the boxplot
ggplot(milk, aes(reorder(Grade, Colour), Colour,fill = 'color')) +
  geom_boxplot(varwidth = TRUE) +
  coord_flip()
```

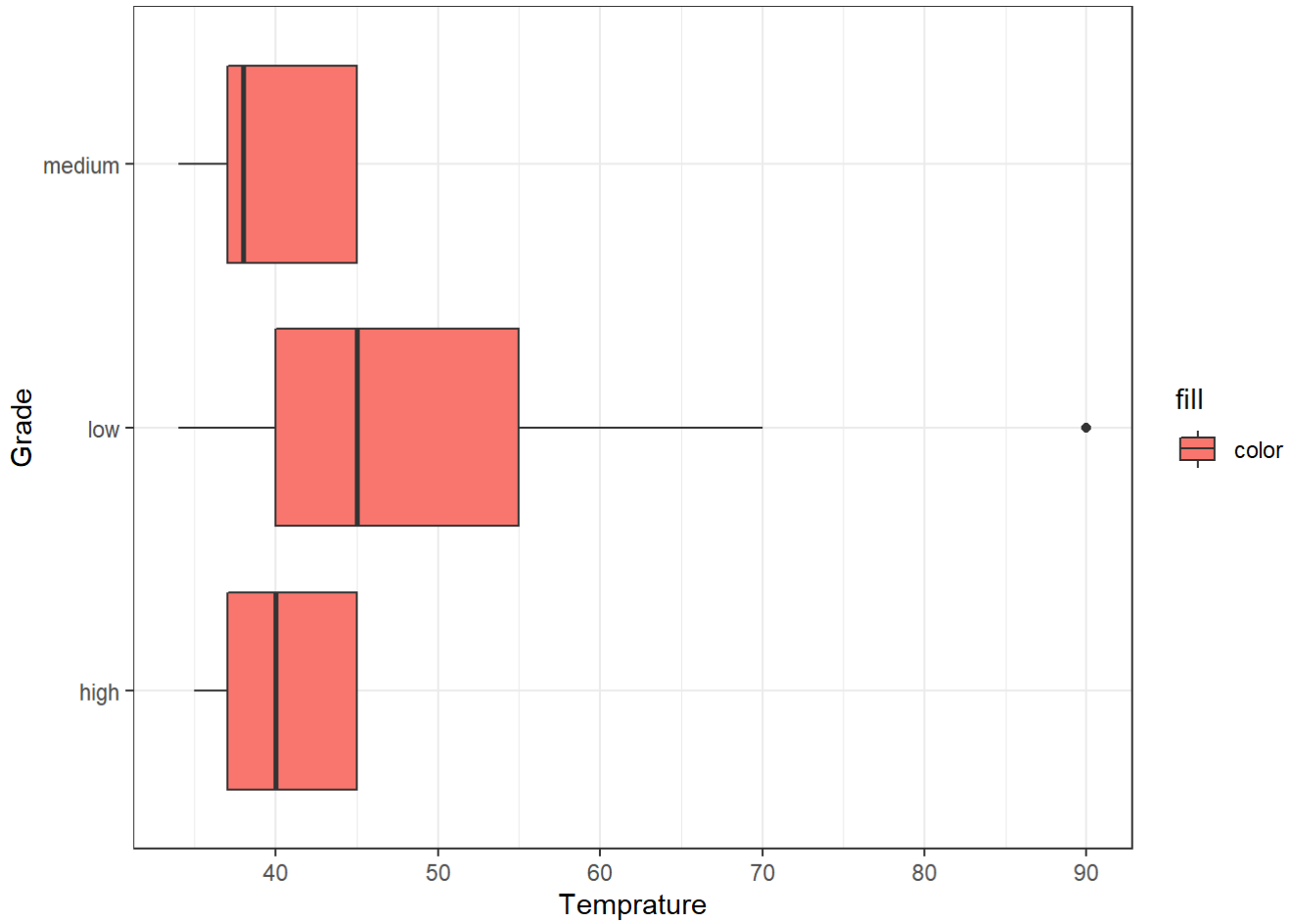


Now let's check the boxplot of pH and temperature.

```
#check the boxplot of pH
milk %>%
  ggplot(aes(x = pH, y = Grade, fill = 'color')) +
  geom_boxplot() +
  labs(y = "Grade", x = "pH") +
  theme_bw()
```



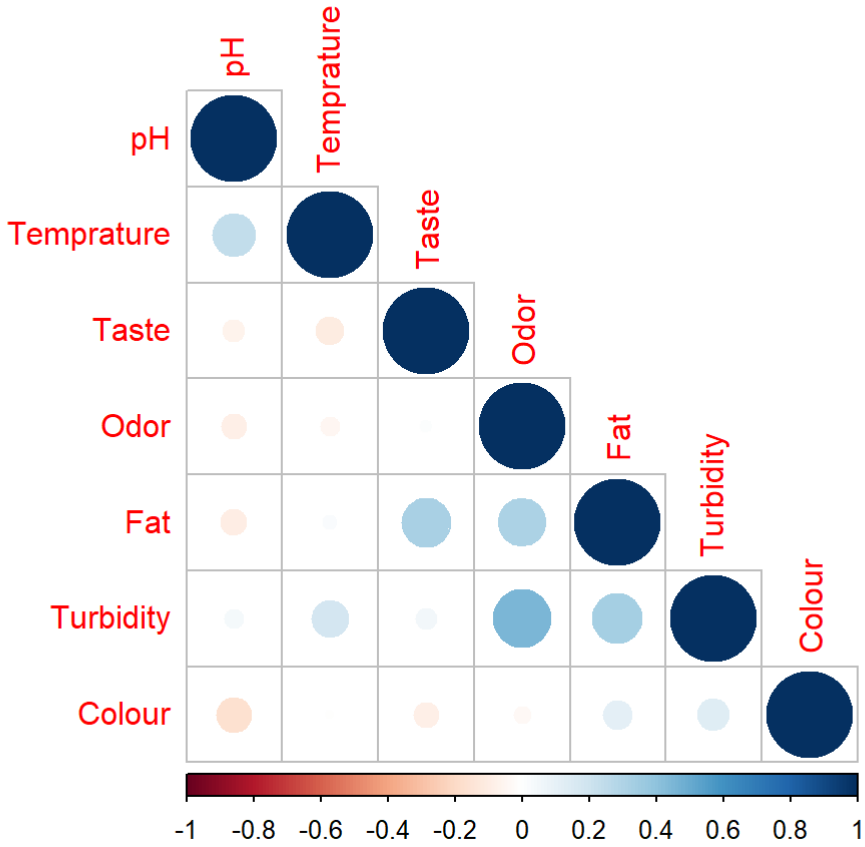

```
#check the boxplot of Temperature
milk %>%
  ggplot(aes(x = Temprature, y = Grade, fill = 'color')) +
  geom_boxplot() +
  labs(y = "Grade", x = "Temprature") +
  theme_bw()
```



Though Temperature:90 looks like an outlier. We find that there are totally 18 samples. We'd better not hasty remove them.

Moreover, it is necessary to check whether there is strong correlation between predictor variables.

```
#Use corrplot to check correlation
milk %>%
  select(where(is.numeric)) %>%
  cor(use = "complete.obs") %>%
  corrplot(type = "lower")
```



No strong correlation was found.

For the next work, we will convert qualitative predictors and grades to factors.

```
#convert qualitative predictors and grades to factors
milk=milk %>%
  mutate(Taste = factor(Taste),Odor = factor(Odor),Fat = factor(Fat),Turbidity =factor(Turbid
ity),Colour = factor(Colour))%>%
  mutate(Grade = factor(Grade, levels = c("high","medium", "low")))
```

4. Data splitting and cross-validation

Initial Split

Here, we set the scale to 0.7 to group the data. Stratified sampling was used as the Grade distribution. Check the

number of two group data.

```
#set seed
set.seed(1215)
#split data using Stratified sampling
milk_split=initial_split(milk, prop = 0.7, strata = Grade)
milk_train=training(milk_split)
milk_test=testing(milk_split)
# check the number of samples in two groups
dim(milk_train)
```

```
## [1] 739   8
```

```
dim(milk_test)
```

```
## [1] 319   8
```

V-fold cross-validation

For the next hyperparameter tuning, we can use k-Fold Cross-Validation here.

```
set.seed(1215)
#k-Fold Cross-Validation
milk_folds=vfold_cv(data = milk_train, v = 10, strata = Grade)
```

5. Model fitting

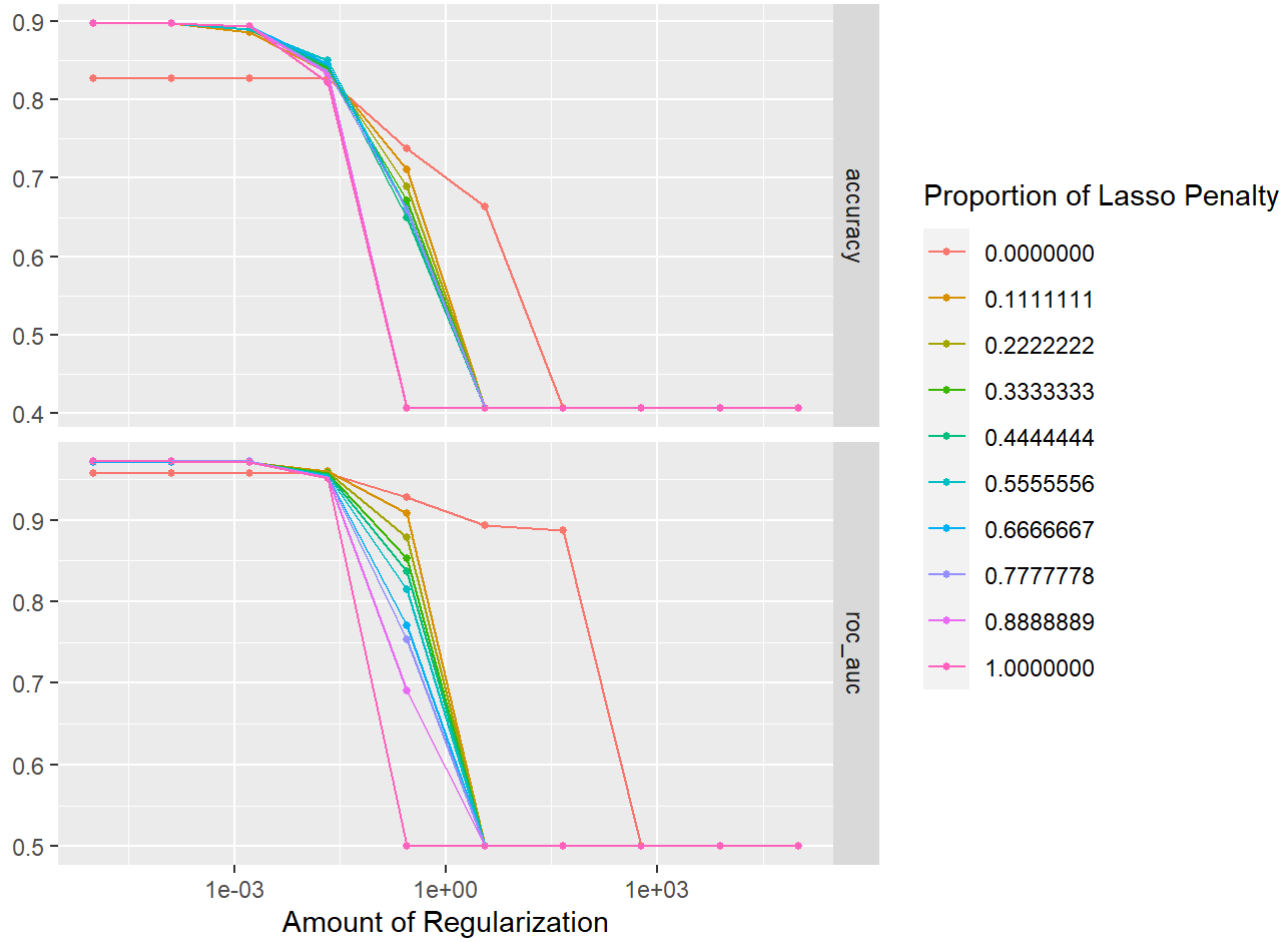
Recipe

As there are five qualitative preditors, we first dummy all them and nomalize all predictors. These parameters play a vital role in the predictive analysis of the milk thus all is needed here.

```
# create the recipe
milk_recipe=recipe(Grade ~ ., data = milk_train) %>%
  # dummy all nominal predictors.
  step_dummy(all_nominal_predictors()) %>%
  # nomalize all predictors
  step_normalize(all_predictors())
```

Elastic Net

To begin with, we first fit the Elastic Net Tuning.



Smaller values of penalty and smaller values of mixture tend to result in higher ROC-AUC and accuracy values.

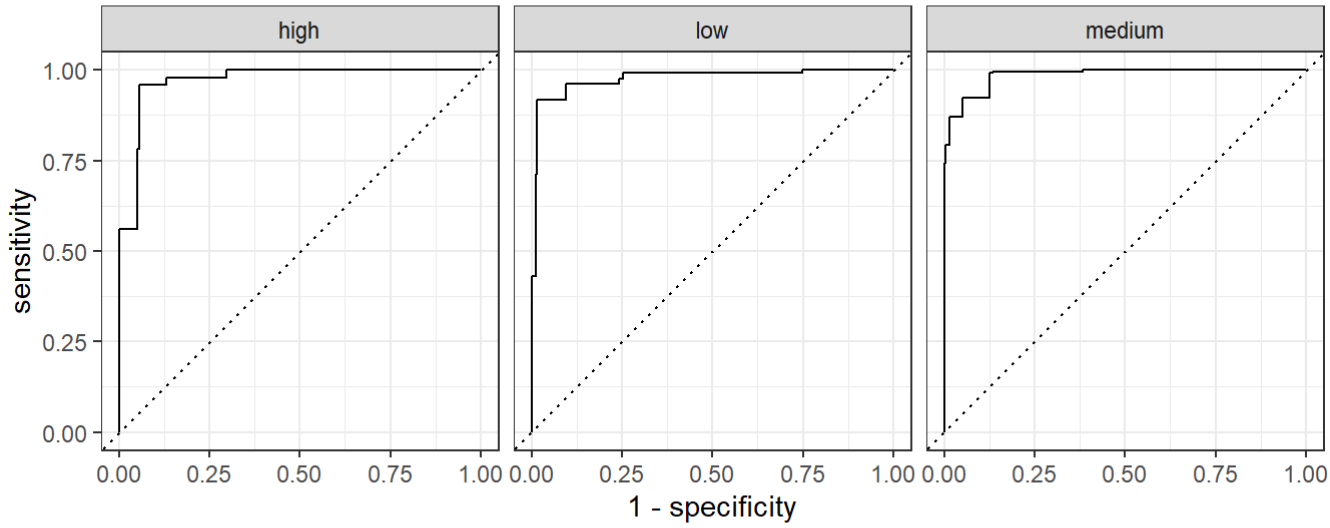
Now, we could choose the model that has the optimal roc_auc and fit the model to the training set and evaluate its performance on the training set.

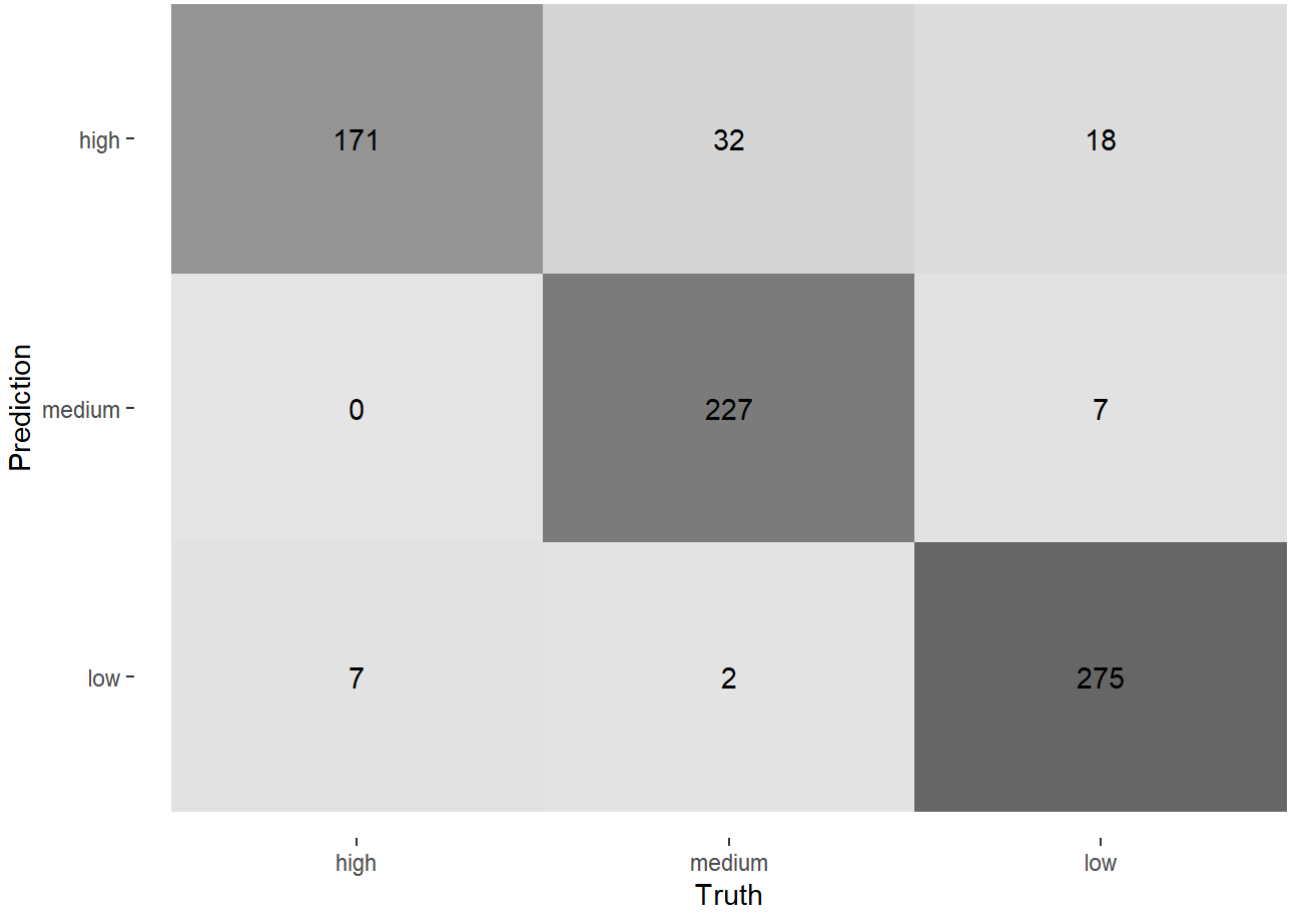
```
#select the best model
best_model=select_best(tune_res, metric = "roc_auc")
#fit the best model
en_final=finalize_workflow(en_workflow, best_model)
en_final_fit=fit(en_final, data = milk_train)
```

We could now evalute whether Elastic net perform well. We now check the overall ROC AUC on the training set,

plots of the different ROC curves, one per level of the outcome and the heat map of the confusion matrix.

```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.976
```

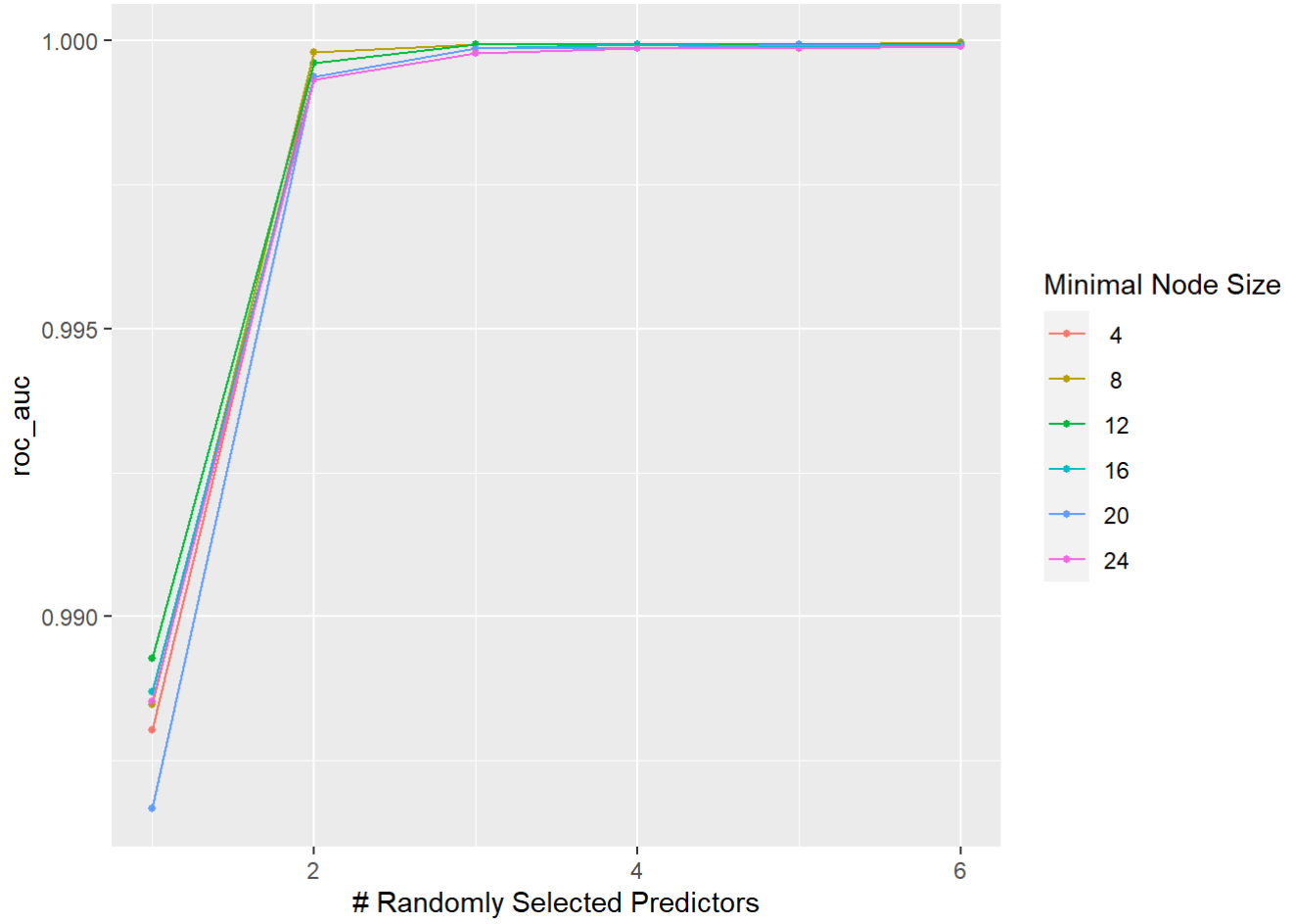




The roc_auc is pretty good in this case. Now let's see other models.

Random Forest

In our Random Forest model, we tuned two different parameters: mtry - The number of predictors that would be randomly sampled and given to the tree to make its decisions and min_n - the minimum number of data values needed to create another split. As the number of predictors increase, so did the accuracy. As the number of trees increased, the ROC AUC also typically increased.



Now, we could choose the best Random Forest model that has the optimal roc_auc and fit the model to the training set and evaluate its performance on the training set.

```
# choose the best model
rf_best_para=select_best(rf_res, metric = "roc_auc")
# fit the best model
rf_final=finalize_workflow(rf_wf, rf_best_para)
rf_final_fit=fit(rf_final, data = milk_train)
augment(rf_final_fit, new_data = milk_train) %>%
  accuracy(truth = Grade, estimate = .pred_class)
```

```
## # A tibble: 1 × 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass         1
```

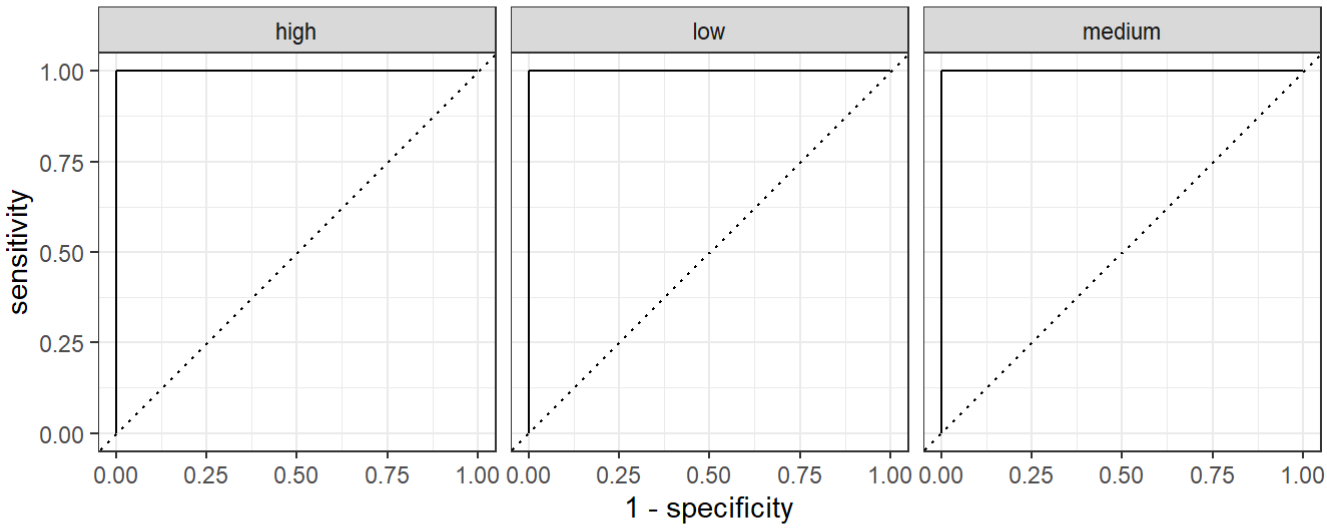
We could now evalute whether Random Forests model perform well. We now check the overall ROC AUC on the training set,

plots of the different ROC curves, one per level of the outcome and the heat map of the confusion matrix.

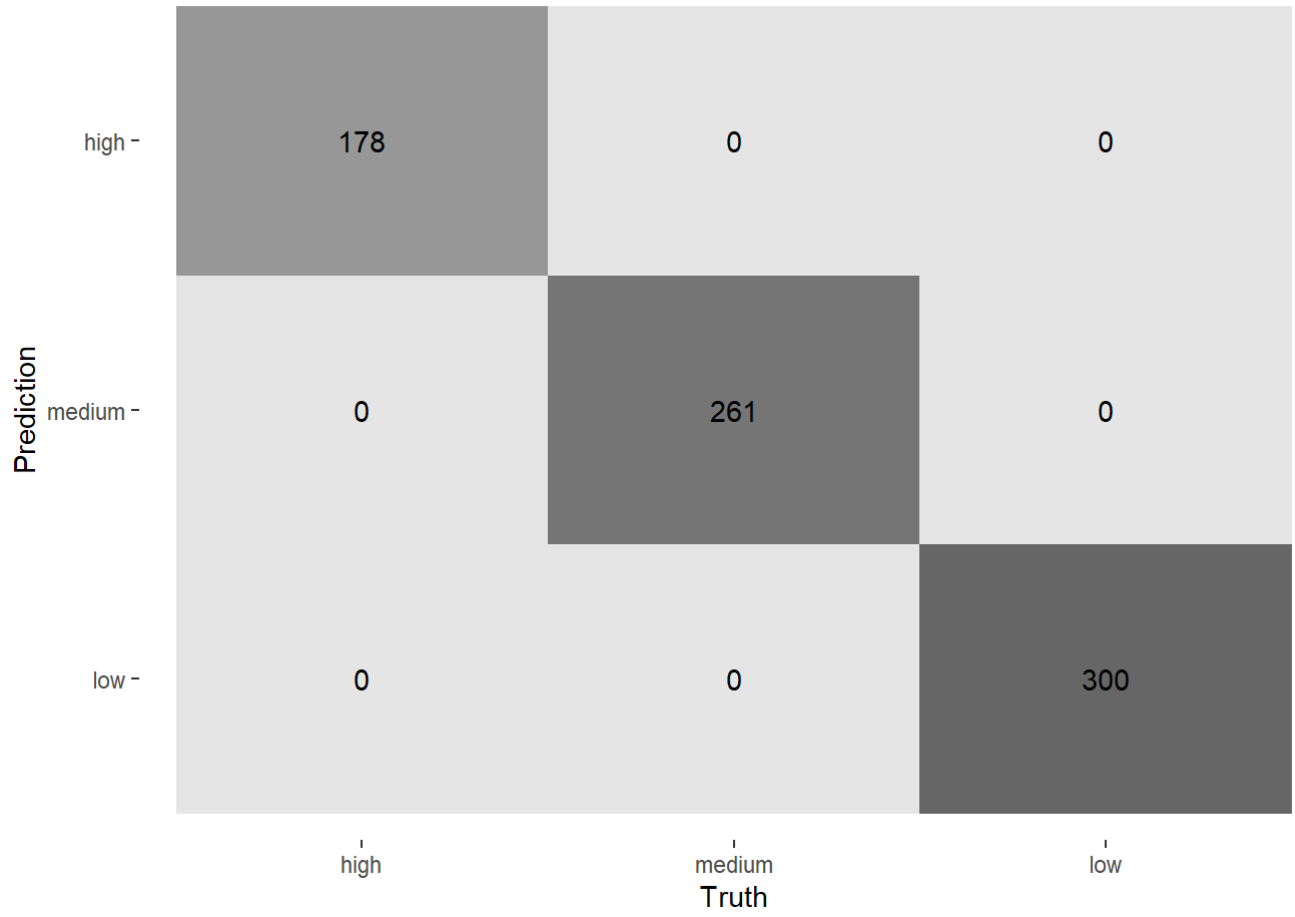
```
# calculate the roc auc
predicted_data=augment(rf_final_fit, new_data = milk_train) %>%
  select(Grade, starts_with(".pred"))
predicted_data %>% roc_auc(Grade, .pred_high:.pred_low)
```

```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till          1
```

```
# show the autoplot
rf_roc_auc=predicted_data %>% roc_auc(Grade, .pred_high:.pred_low)
predicted_data %>% roc_curve(Grade, .pred_high:.pred_low) %>%
  autoplot()
```



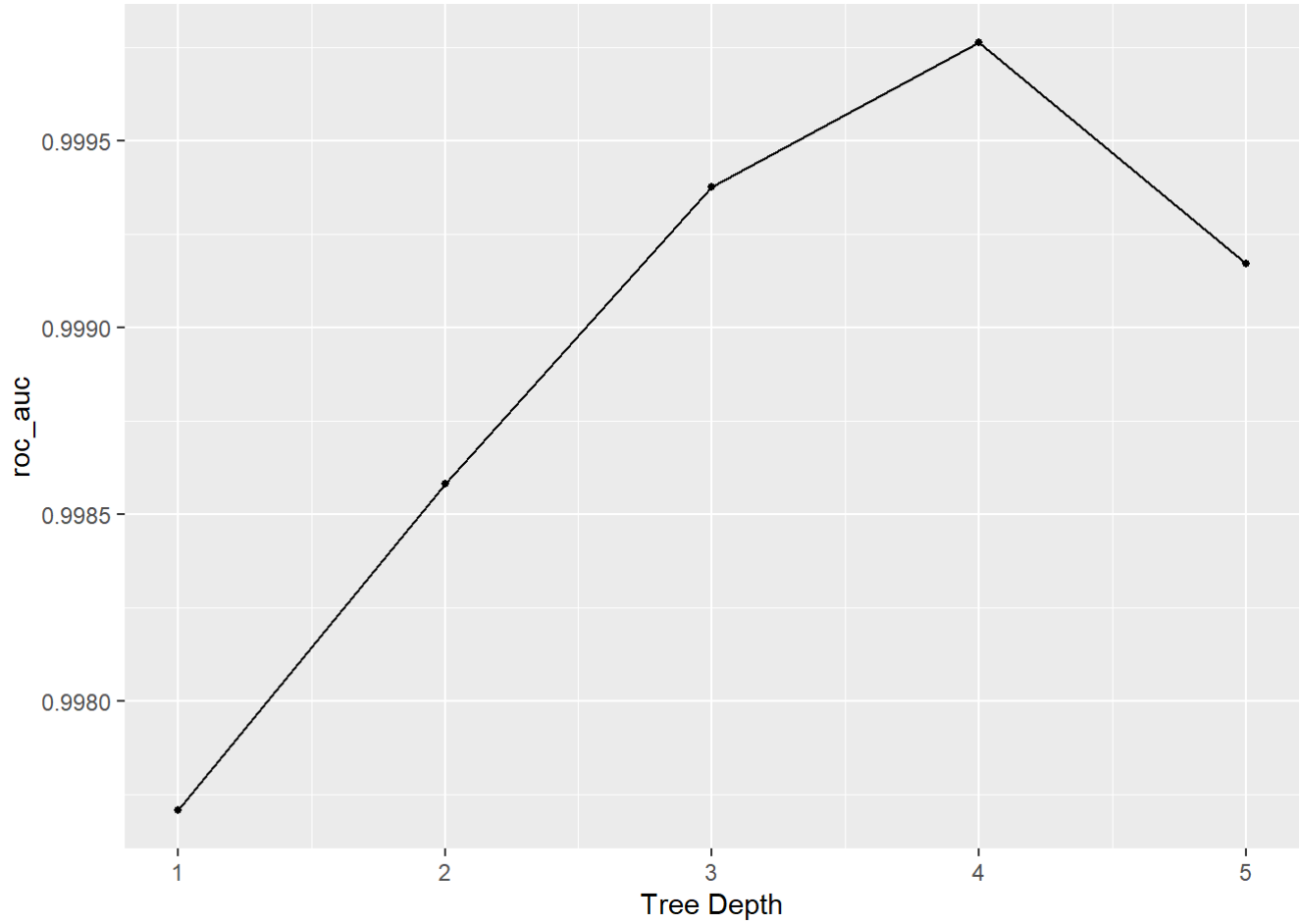

```
# show the heat map of the confusion matrix
predicted_data %>%
  conf_mat(truth = Grade, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```



The roc_auc is pretty good in this case. Now let's see other models.

Boosted Tree

In our Boosted Tree model, we tuned one parameter: tree depth - The number of the maximum depth of the tree. The roc auc curve reaches highest at h=4 and then decreases.



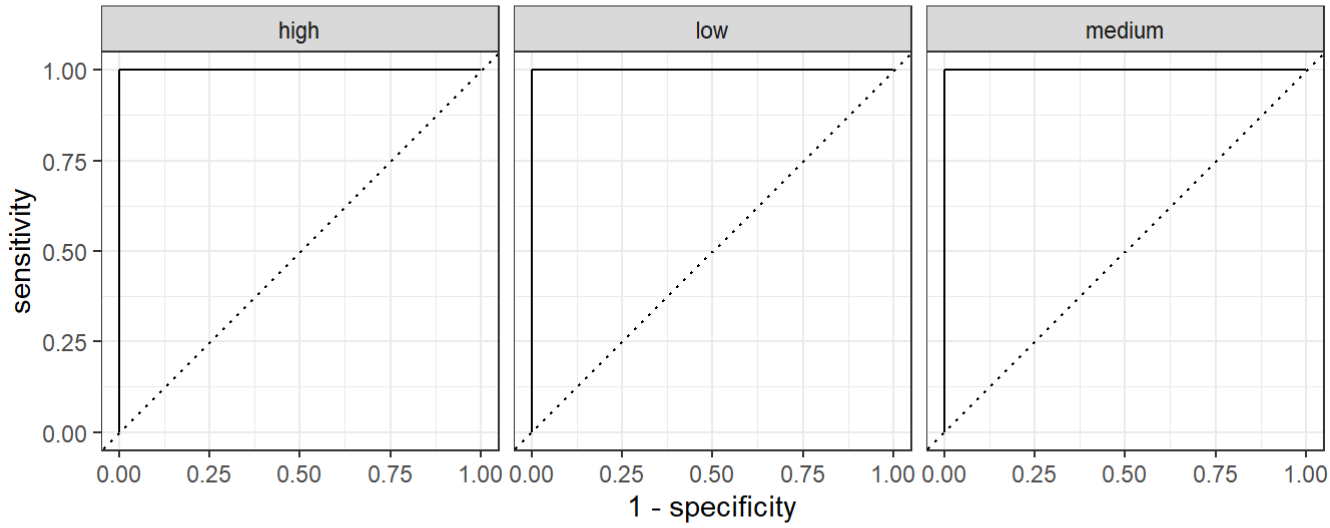
Now, we could choose the best Boosted Tree model that has the optimal roc_auc and fit the model to the training set and evaluate its performance on the training set.

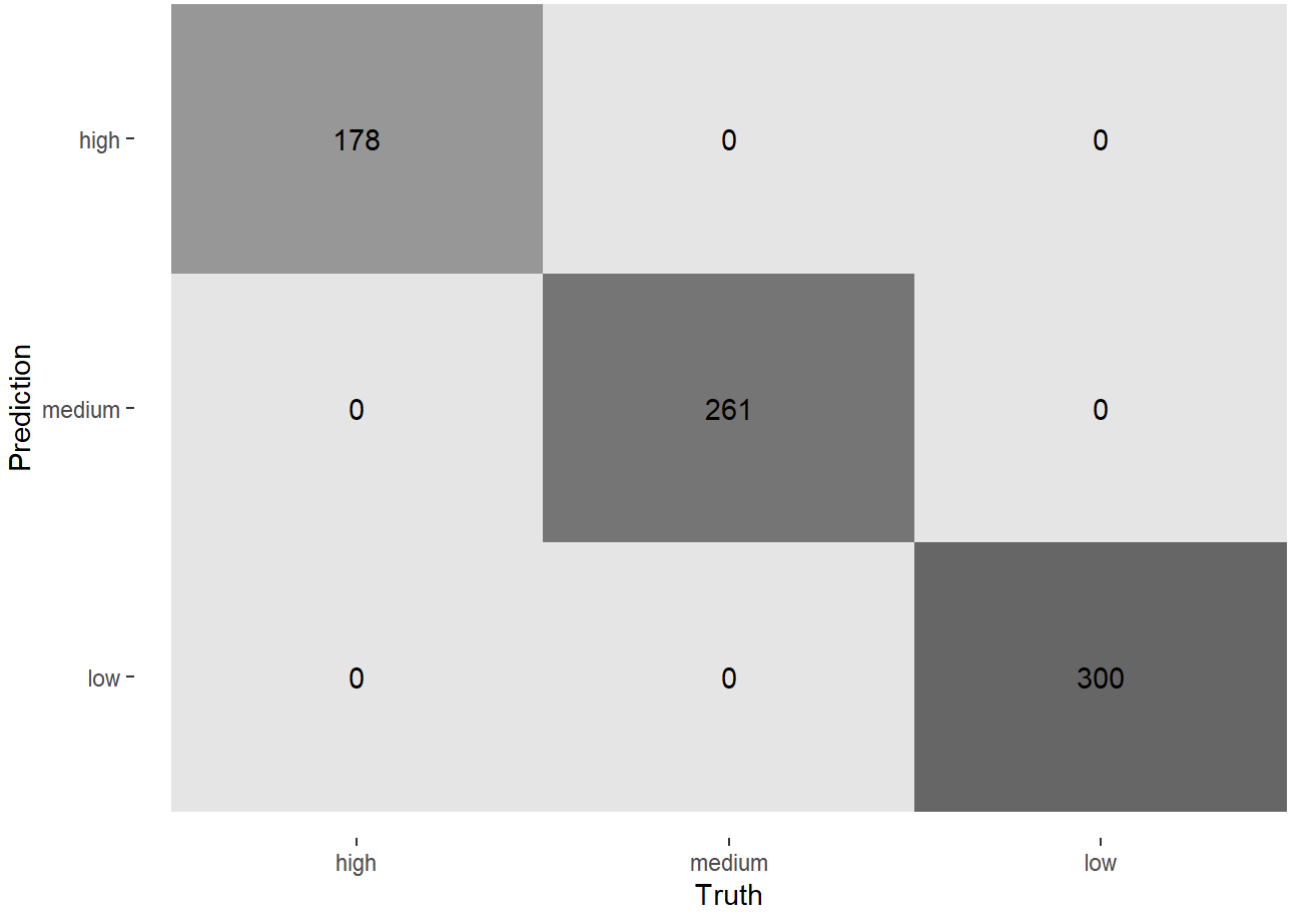
```
## # A tibble: 1 × 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass         1
```

We could now evalute whether Boosted Tree model perform well. We now check the overall ROC AUC on the training set,

plots of the different ROC curves, one per level of the outcome and the heat map of the confusion matrix.

```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till          1
```


high
low
medium
sensitivity
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
1 - specificity

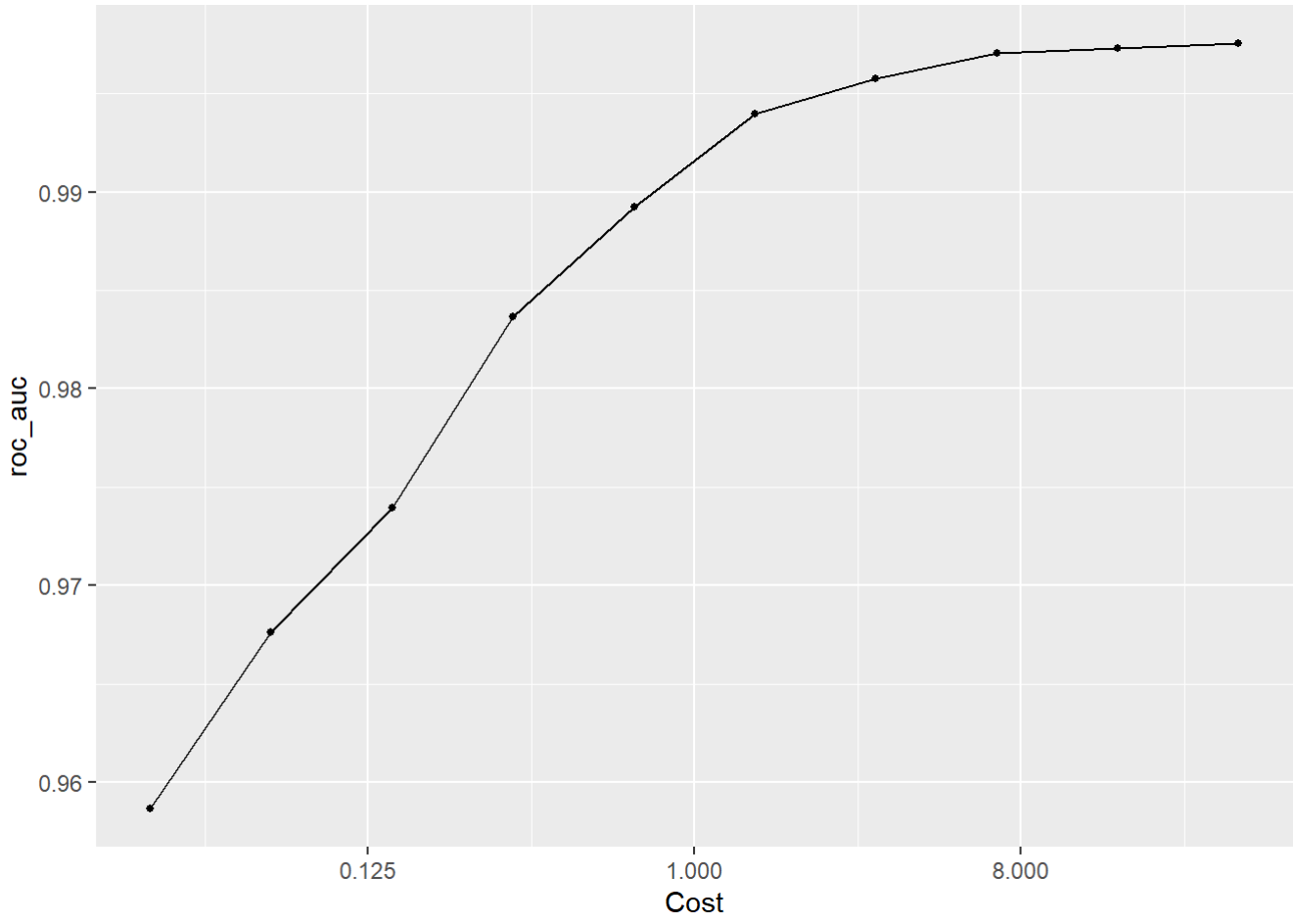

high
medium
low
Prediction
178
0
0
0
261
0
0
0
300
high
medium
low
Truth

SVM

In our SVM, we tuned one parameter: cost - the price for misclassifications. As the value of cost increased, the ROC AUC also typically increased.

```
# set the model
svm_spec=svm_rbf() %>%
  set_mode("classification") %>%
  set_engine("kernlab")
# set the workflow
svm_wf=workflow() %>%
  add_model(svm_spec %>% set_args(cost = tune())) %>%
  add_recipe(milk_recipe)
```

```
# set the tunning grid
para_grid=grid_regular(cost(range = c(-5, 5)), levels = c(cost = 10))
# fit the model
svm_res=tune_grid(
  svm_wf,
  resamples = milk_folds,
  grid = para_grid,
  metrics = metric_set(roc_auc)
)
# show the autoplot
autoplot(svm_res)
```



Now, we could choose the best SVM model that has the optimal roc_auc and fit the model to the training set and evaluate its performance on the training set.

```
# choose the best model
svm_best_para=select_best(svm_res, metric = "roc_auc")
# fit the model
svm_final=finalize_workflow(svm_wf, svm_best_para)
svm_final_fit=fit(svm_final, data = milk_train)
augment(svm_final_fit, new_data = milk_train) %>%
  accuracy(truth = Grade, estimate = .pred_class)
```

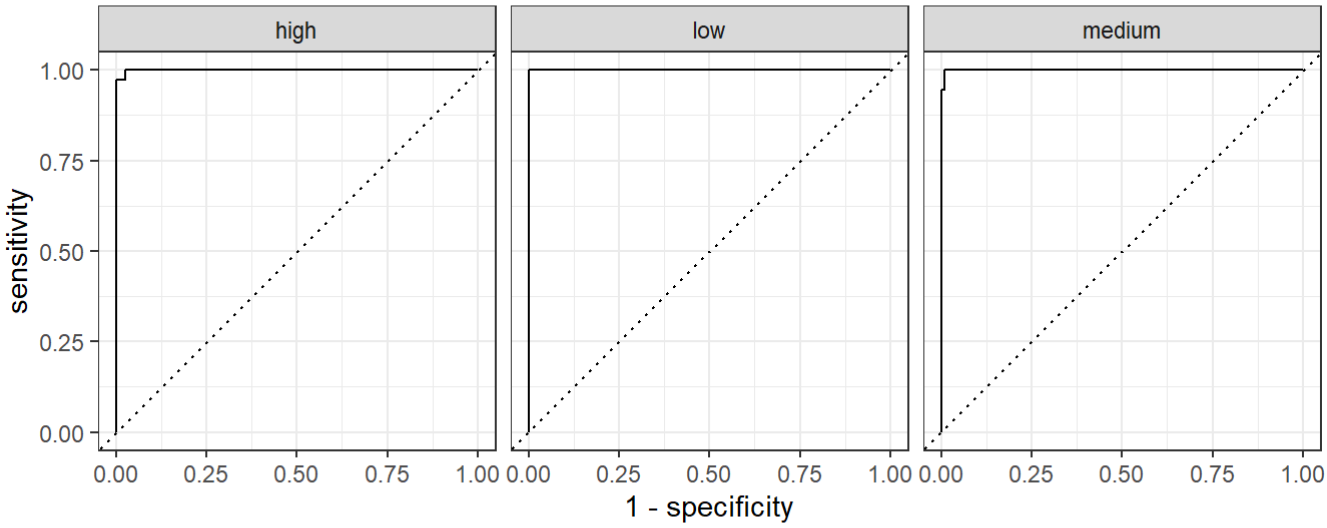
```
## # A tibble: 1 × 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.981
```

We could now evalute whether SVM perform well. We now check the overall ROC AUC on the training set, plots of the different ROC curves, one per level of the outcome and the heat map of the confusion matrix.

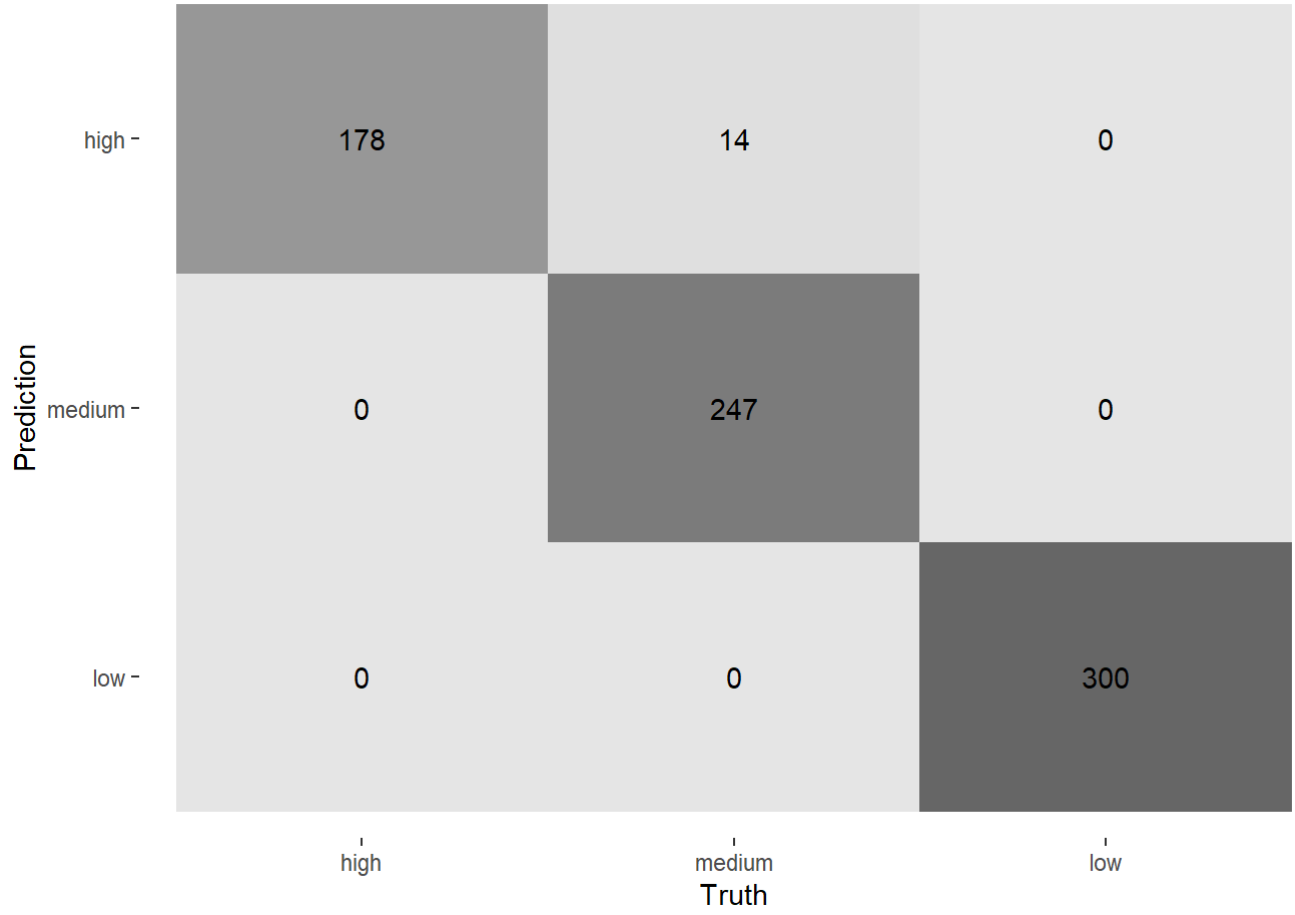
```
# calculate the roc auc
predicted_data=augment(svm_final_fit, new_data = milk_train) %>%
  select(Grade, starts_with(".pred"))
predicted_data %>% roc_auc(Grade, .pred_high:.pred_low)
```

```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.999
```

```
svm_roc_auc=predicted_data %>% roc_auc(Grade, .pred_high:.pred_low)
# show the autoplot
predicted_data %>% roc_curve(Grade, .pred_high:.pred_low) %>%
  autoplot()
```




```
# show the heat map of the confusion matrix
predicted_data %>%
  conf_mat(truth = Grade, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```



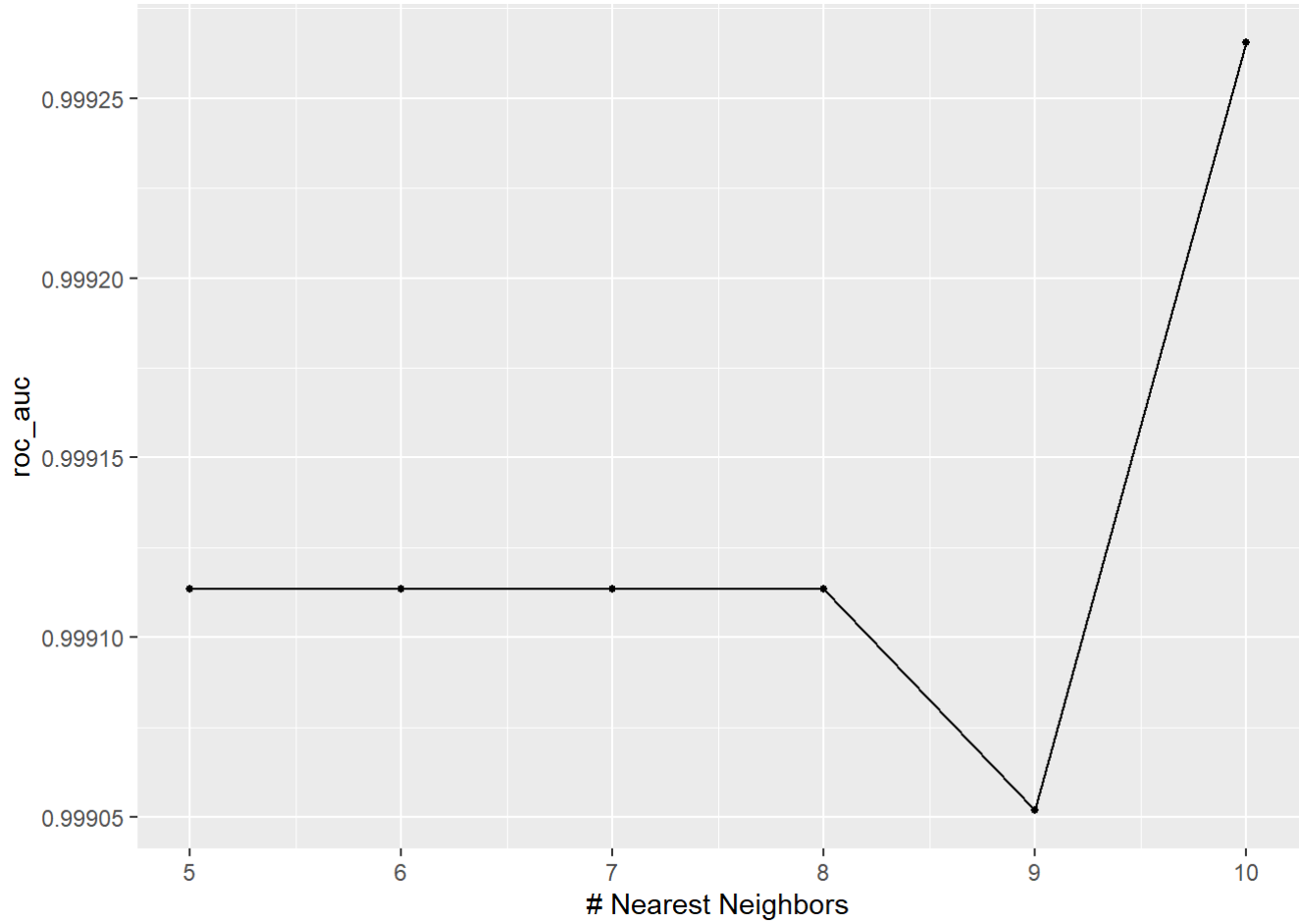
The roc_auc is pretty good in this case. Now let's see other models.

KNN

In our KNN model, we tuned one parameter: neighbors - the number of neighbors to consider. The roc auc curve reaches highest at k=10 which may cause overfitting. Let's see more details.

```
# set the model
knn_spec=nearest_neighbor(neighbors = tune()) %>%
  set_mode("classification") %>%
  set_engine("kknn")
# set the workflow
knn_wf=workflow() %>%
  add_model(knn_spec) %>%
  add_recipe(milk_recipe)
```

```
# set the tunning grid
para_grid=grid_regular(neighbors(range = c(5, 10)), levels = c(neighbors = 6))
# fit the model
knn_res=tune_grid(
  knn_wf,
  resamples = milk_folds,
  grid = para_grid,
  metrics = metric_set(roc_auc)
)
# show the autoplot
autoplot(knn_res)
```



Now, we could choose the best KNN model that has the optimal roc_auc and fit the model to the training set and evaluate its performance on the training set.

```
# choose the best model
knn_best_para=select_best(knn_res, metric = "roc_auc")
# fit the best model
knn_final=finalize_workflow(knn_wf, knn_best_para)
knn_final_fit=fit(knn_final, data = milk_train)
augment(knn_final_fit, new_data = milk_train) %>%
  accuracy(truth = Grade, estimate = .pred_class)
```

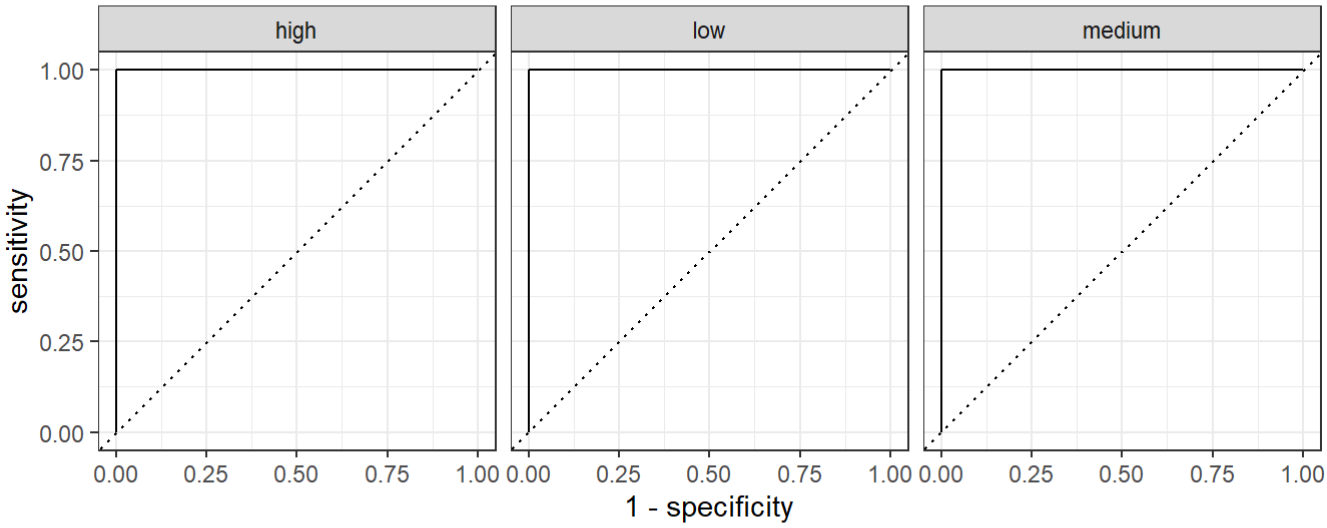
```
## # A tibble: 1 × 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.996
```

We could now evalute whether KNN perform well. We now check the overall ROC AUC on the training set, plots of the different ROC curves, one per level of the outcome and the heat map of the confusion matrix.

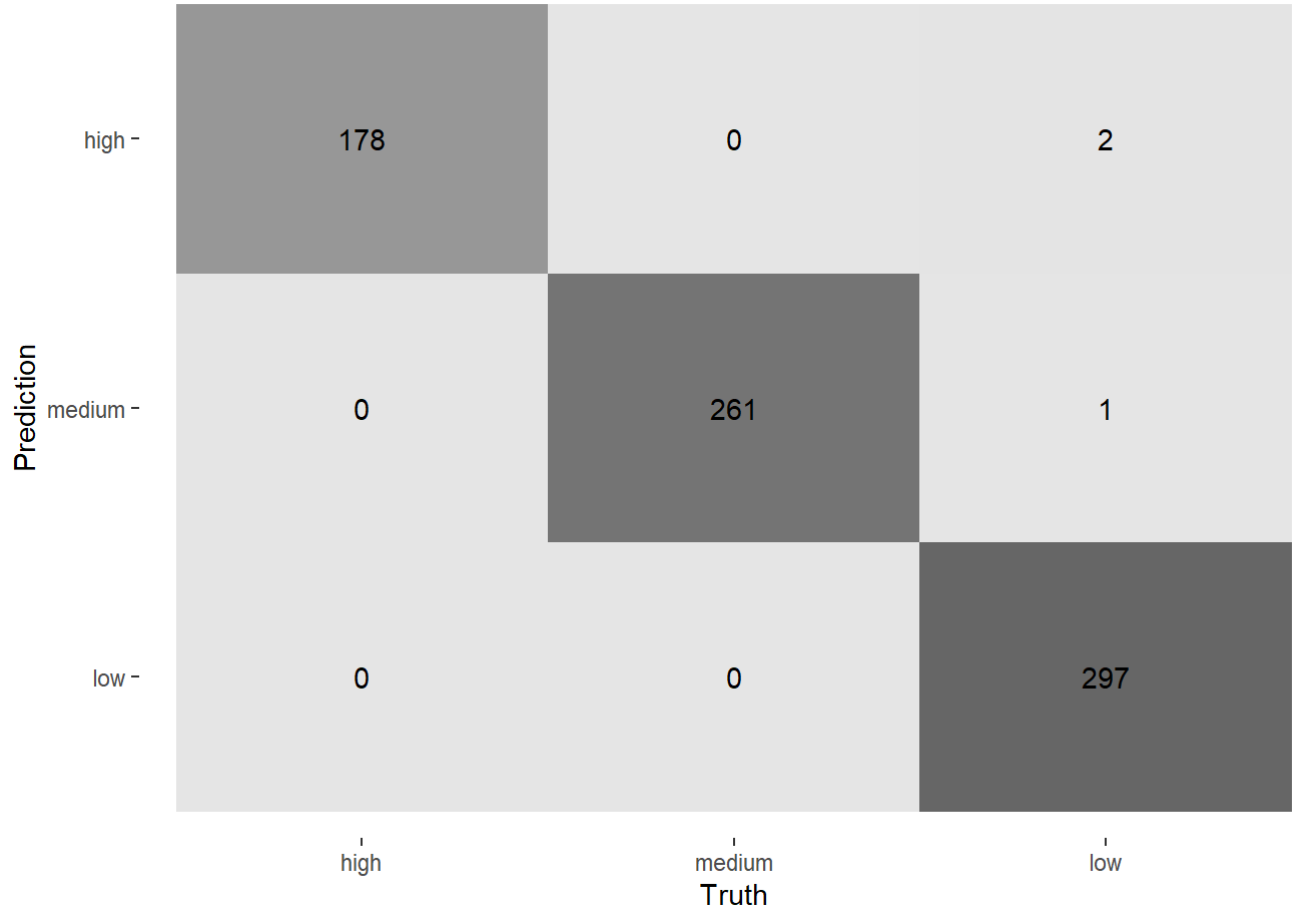
```
# calculate the roc auc
predicted_data=augment(knn_final_fit, new_data = milk_train) %>%
  select(Grade, starts_with(".pred"))
predicted_data %>% roc_auc(Grade, .pred_high:.pred_low)
```

```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till       1.00
```

```
knn_roc_auc=predicted_data %>% roc_auc(Grade, .pred_high:.pred_low)
# show the autoplot
predicted_data %>% roc_curve(Grade, .pred_high:.pred_low) %>%
  autoplot()
```




```
# show the heat map of the confusion matrix
predicted_data %>%
  conf_mat(truth = Grade, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```



The roc_auc is pretty good in this case. Next, we would like to compare all models and decide which one to use.

6. Final Model

We set the metric to be roc auc. Let's all see roc auc of all five models.

```
# summary of roc auc of all models
roc_auc=c(en_roc_auc$.estimate, rf_roc_auc$.estimate, boost_roc_auc$.estimate, svm_roc_auc$.e
stimate,knn_roc_auc$.estimate)
models=c("Elastic net", "Random Forest", "Boosted Trees", "Support vector machine","K-nearest
neighbors")
# create a tibble
results=tibble(roc_auc = roc_auc, models = models)
results %>%
  arrange(-roc_auc)
```

```
## # A tibble: 5 × 2
##   roc_auc models
##     <dbl> <chr>
## 1   1     Random Forest
## 2   1     Boosted Trees
## 3   1.00  K-nearest neighbors
## 4   0.999 Support vector machine
## 5   0.976 Elastic net
```

It is clear that all models perform quite good. I believe this result is due to our outcome variable is perfectly classified by all predictors. As both Random Forest model and Boosted Trees model achieve wonderful results, considering overfitting of Boosted Trees model, I prefer to choose Random Forests model as our final model.

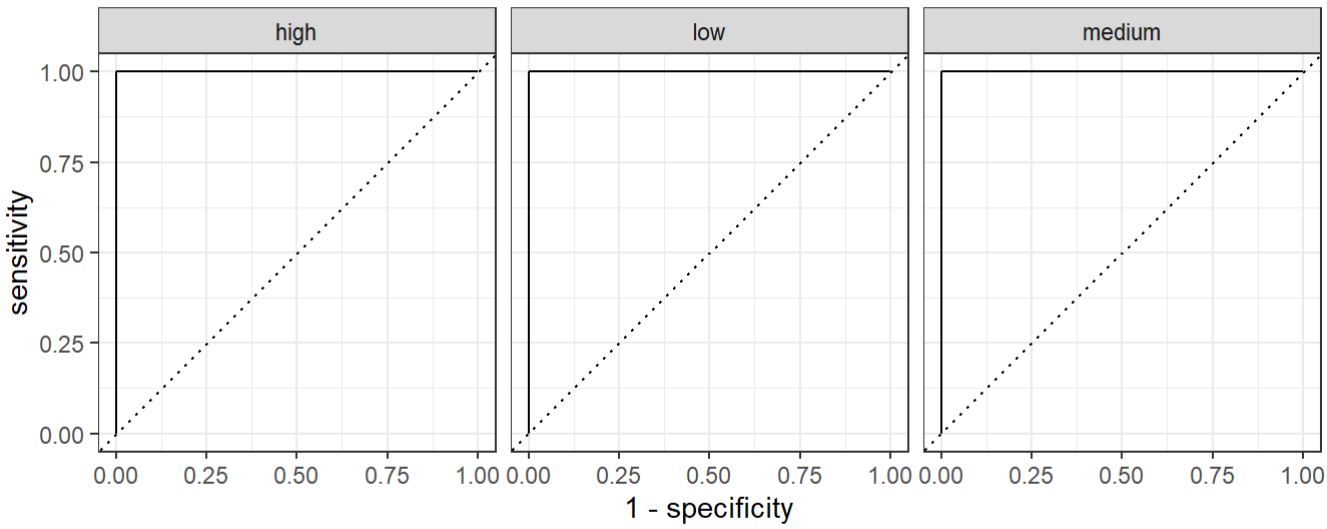
Fitting to Testing Data

Now, we could fit the best Random Forests model that has the optimal roc_auc to the testing set and evaluate its performance on the it.

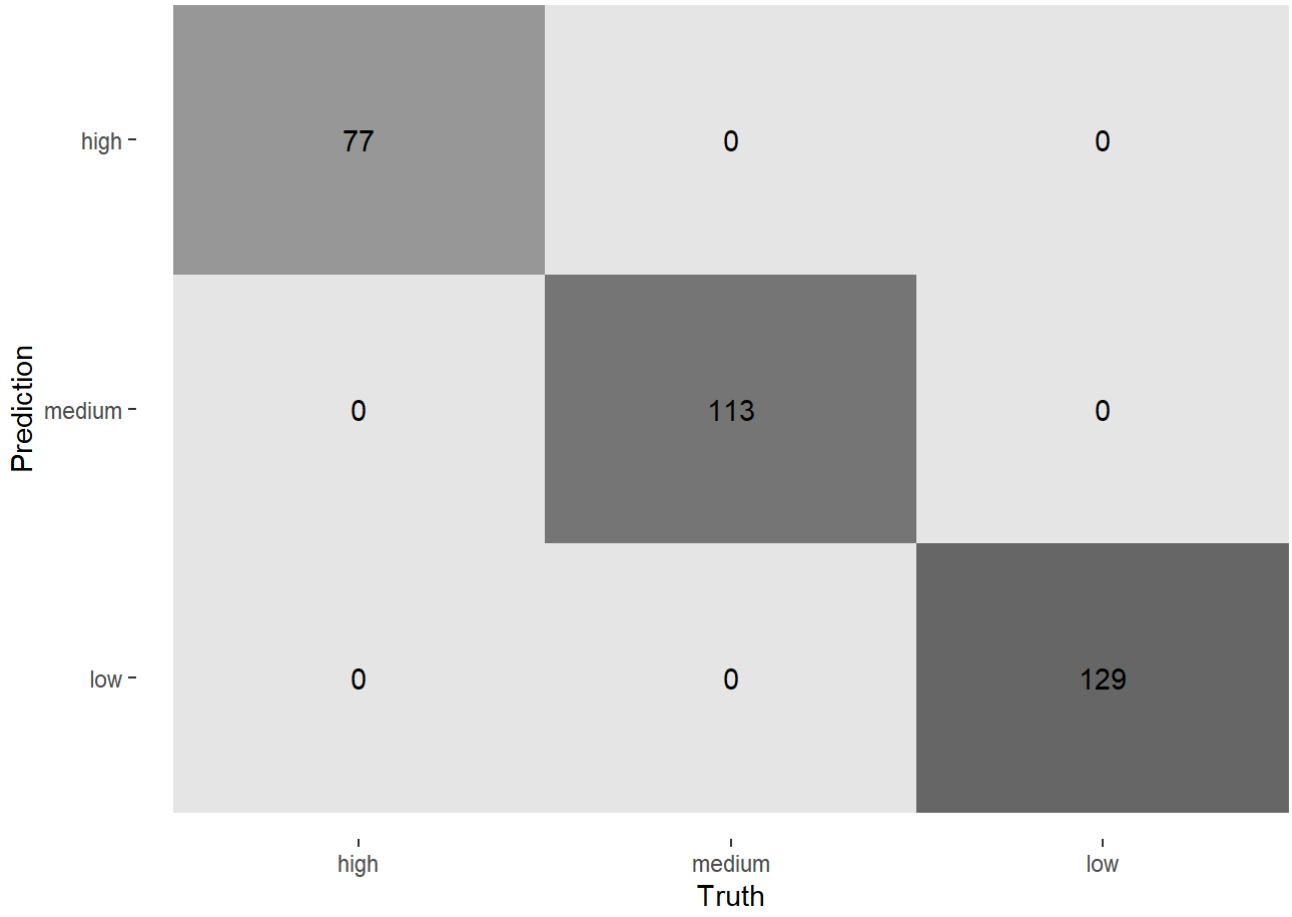
```
# fit the best Random Forests model
predicted_data=augment(rf_final_fit, new_data = milk_test) %>%
  select(Grade, starts_with(".pred"))
predicted_data %>% roc_auc(Grade, .pred_high:.pred_low)
```

```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till          1
```

```
# show the autoplot
predicted_data %>% roc_curve(Grade, .pred_high:.pred_low) %>%
  autoplot()
```



```
# show the heat map of the confusion matrix
predicted_data %>%
  conf_mat(truth = Grade, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```



7. Conclusion

This article fits the several models in R to assess its roc auc of quality classification using the milk data from kaggle. According to the experimental findings, the Random Forest model can classify the quality of milk fairly well

and it is also interpretable, making it an alternative model for predicting similar daily product. We also examine the ROC curves and the heat map of the confusion matrix. As the original oucome data is perfectly classified.

Everything goes well and we even obtain wonderful result – 1 roc auc. At the beginning, I'm worried about overfitting problem. Now I am more confident about the model with the support of inside relationship of our data.

Morever, by tweaking the model's input parameters and cleaning up the original data to remove outliers, follow-up can

enhance the model.