

# Wine Quality Prediction / ML

Peter Chu

2022-11-27

## Table of Contents

1. Abstract
2. Main Body
3. Exploratory Data Analysis
4. Data Splitting And Folding
5. Model Creation And Training
6. Results Of Models, Best Model Selection, And Applying Best Model
7. Conclusion
8. References
9. Appendix

## Abstract

In this project I want to be able to predict the quality of wine based on its creation process. This involved residual sugar from the fermentation process, added citric acid, added sulfate, and much more. It is a very complex process, but with these values known it is possible to predict the quality of the wine. It involved using many models and ultimately a random forest model performed the best on the training data set.

## Main Body

I used the dataset ‘Red Wine Quality’ which was published from the UCI Machine Learning Repository. It has 1599 observations of 12 variables. I ended up creating a new categorical value called ‘fqual’ which categorically quantifies the wine quality based off of its numerical quality rating. I ended up choosing a 70/30 data split and I believe that since we have a large number of observations, a 70% split is enough data to accurately train our models on. The main reason why I chose to do analysis /project on this dataset is because I go to UCSB. It was once ranked the #1 party school for very obvious reasons. Young adults with access to alcohol leads to loud weekends. However, many elitist snobs often look down at this kind of drinking the older people get. They believe in being “fancy” and drink wine. Therefore I decided to do some analysis on wine with the firm belief that it doesn’t matter what you drink as long as you reach your preferred level of drunk.

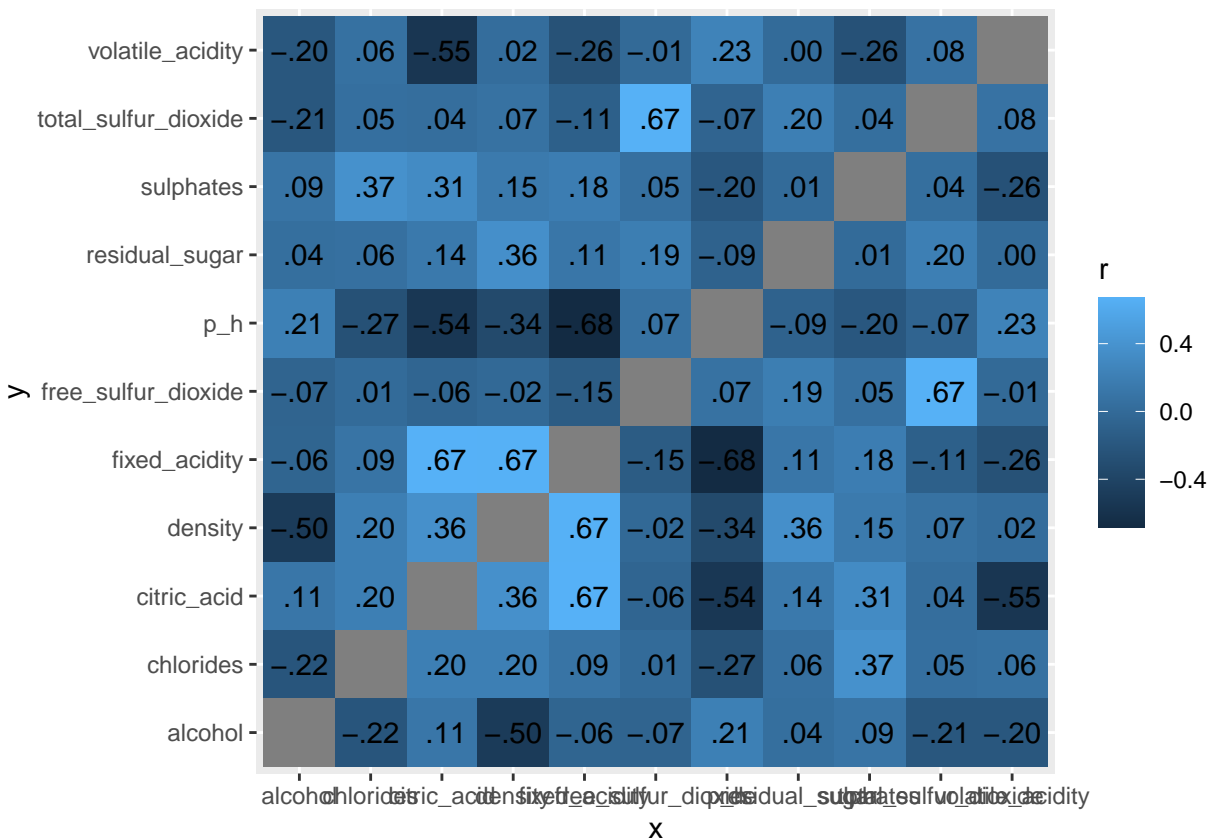
For the analysis I started by splitting the data into a 70/30 split after creating a new categorical variable and dropping the numerical one it was derived from. I then did a 10 fold cross validation because we have a large data set we can create more accurate models by doing so. To start off the initial data analysis, I created

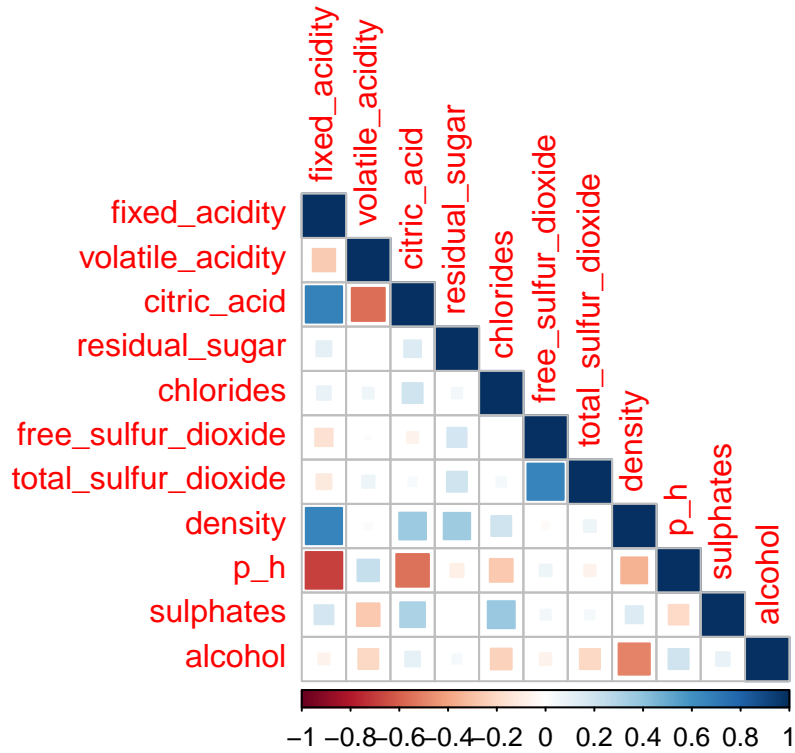
a correlation matrix of all the variables and histograms to catch predictors with large outliers. Afterwards I scaled such predictors and created an interaction term between two of them. Then I tested the training data on a elastic net, SVM, basic tree, random forest, and boosted tree models. From this is became clear that the random forest model was the best. I then used the model on the testing data set and achieved a similar roc\_auc. A heatmap matrix and roc curve plots further confirm that the model performed well.

## Exploratory Data Analysis

Let us first look a correlation matrices of all predictors.

```
## Correlation computed with
## * Method: 'pearson'
## * Missing treated using: 'pairwise.complete.obs'
```





Although the table is a little hard to read, we can conclude from the correlation matrix that fixed\_acidity has high correlation among itself, density, and citric acid. In addition, total\_sulfur\_dioxide has high correlation with free\_sulfur\_dioxide which makes sense as free\_sulfur\_dioxide is a part of the total\_sulfur\_dioxide. Interesting, p\_h has a high negative correlation with fixed acidity and citric acid. volatile\_acidity also has a high negative correlation between volatile\_acidity and citric acid, and alcohol and density.

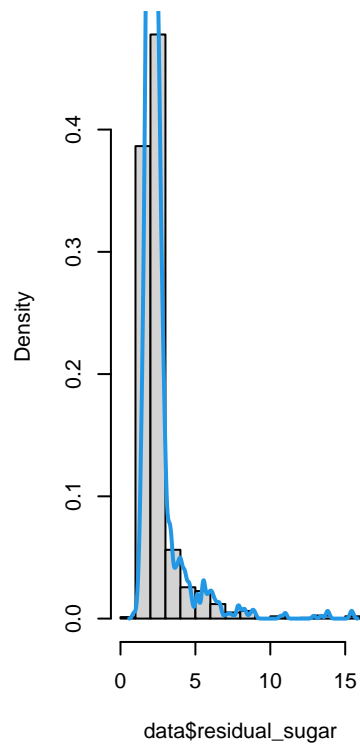
Let us see if there are any outliers for each variable which we need to scale later on.

fixed_acidity	volatile_acidity	citric_acid
Min. : 4.60	Min. :0.1200	Min. :0.000
1st Qu.: 7.10	1st Qu.:0.3900	1st Qu.:0.090
Median : 7.90	Median :0.5200	Median :0.260
Mean : 8.32	Mean :0.5278	Mean :0.271
3rd Qu.: 9.20	3rd Qu.:0.6400	3rd Qu.:0.420
Max. :15.90	Max. :1.5800	Max. :1.000

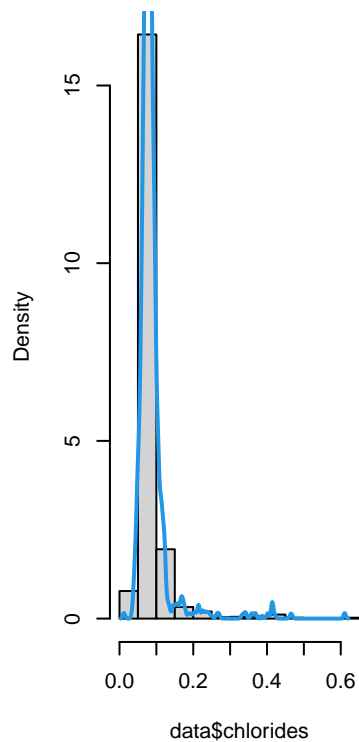
residual_sugar	chlorides	free_sulfur_dioxide
Min. : 0.900	Min. :0.01200	Min. : 1.00
1st Qu.: 1.900	1st Qu.:0.07000	1st Qu.: 7.00
Median : 2.200	Median :0.07900	Median :14.00
Mean : 2.539	Mean :0.08747	Mean :15.87
3rd Qu.: 2.600	3rd Qu.:0.09000	3rd Qu.:21.00
Max. :15.500	Max. :0.61100	Max. :72.00

From our summary it appears that residual sugar, chlorides, free\_sulfur\_dioxide, total\_sulfur\_dioxide, sulphates are skewed heavily. Taking a look at their histograms confirms this.

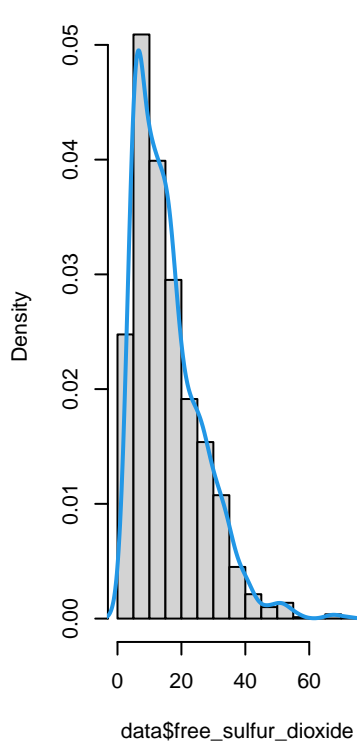
**Histogram of data\$residual\_sugar**



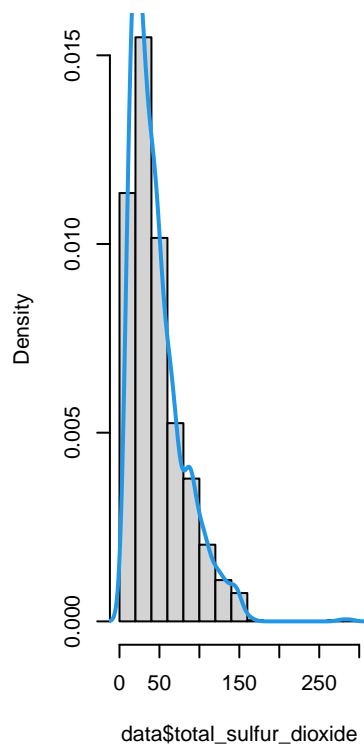
**Histogram of data\$chlorides**



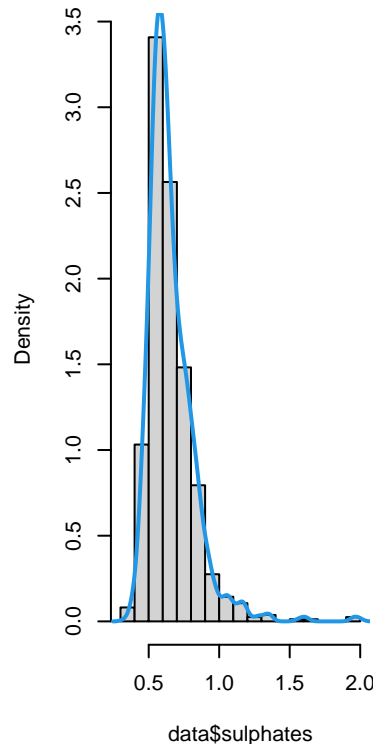
**histogram of data\$free\_sulfur\_dioxide**



histogram of data\$total\_sulfur\_di



Histogram of data\$sulphates



Therefore when we create a recipe for our models to use, we will scale these predictors.

## Data Splitting And Folding

I chose to split the data 70/30, so the training set will have 70% of the data and the testing will have 30%. this was done to ensure that we have enough data to adequately train our models before testing them. I also chose to stratify sample on the variable fqual as this is our variable of interest. After all this, I folded the data into 10 folds on the strata fqual again so that each fold will have enough of each wine quality rating. If this was not done, then some folds could have more ratings of Best which could skew our models. A recipe for the training data was made which involved scaling the aforementioned predictors with high outliers and making interaction terms between sugar and chlorides. This was done as the residual sugar creates the sweetness of the wine while chlorides make the bitterness. Thus they interact in the process of making wine.

```
#Set seed to get reproducible results
set.seed(100)

#Split data into a 70/30 split on strata fqual
data_split <- initial_split(data, strata = fqual, prop = 0.7)
data_train <- training(data_split)
data_test <- testing(data_split)

#fold data on strata fqual 10 times
data_train_fold <- vfold_cv(data_train, v = 10, strata = fqual)

#Create recipe
```

```
data_rec <- recipe(fqual ~ fixed_acidity + volatile_acidity + citric_acid + residual_sugar + chlorides +
  step_dummy(all_nominal_predictors()) %>%
  step_scale(residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, sulphates) %>%
  step_interact(residual_sugar ~ sulphates)
```

## Model Creation And Training

### Elastic Net Model

We will first start with an elastic net model which uses the penalties from both Lasso and Ridge regression. Elastic net should be used as we have groups of highly correlated independent variables. For example, total\_sulfur\_dioxide and free\_sulfur\_dioxide form a highly correlated group that is independent from the highly correlated group of fixed\_acidity and citric\_acid. We will be tuning the hyperparameters mixture and penalty to capture a wide variety of models. Below is a graph of all the models and a table of summary statistics of the best performing models.

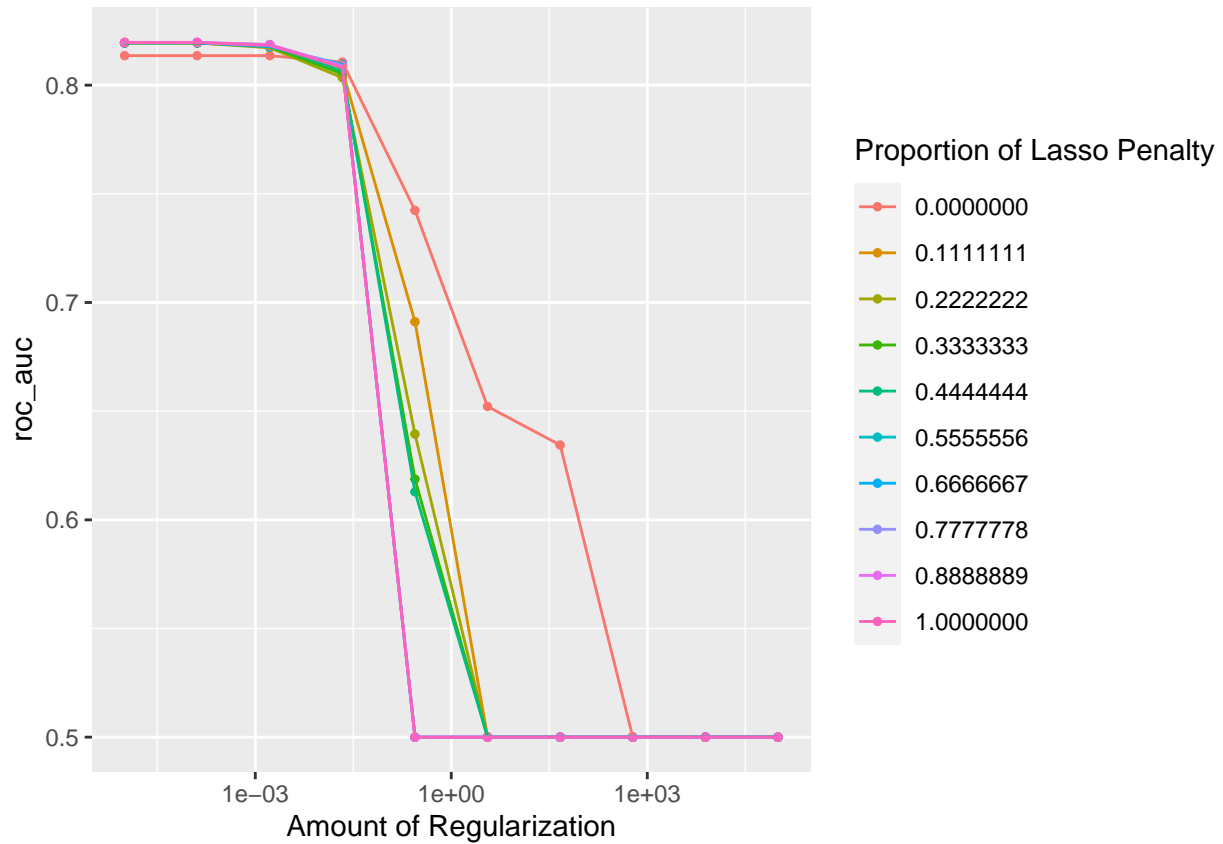


Table 3: First Few Rows Of Best Performing Elastic Net Models And Their Statistics

Metric	Value	Standard_Error	Penalty_Value	Mixture_Value
roc_auc	0.8197134	0.0212351	0.0001292	0.8888889
roc_auc	0.8196900	0.0212157	0.0001292	1.0000000
roc_auc	0.8196870	0.0212409	0.0000100	0.8888889
roc_auc	0.8196743	0.0212242	0.0000100	1.0000000

Metric	Value	Standard_Error	Penalty_Value	Mixture_Value
roc_auc	0.8196298	0.0212162	0.0000100	0.7777778
roc_auc	0.8196298	0.0212162	0.0001292	0.7777778

As we can see from our table, the best Elastic Net model had an roc\_auc of 0.819 with a standard error of 0.021. It used the penalty value of 0.0001292 and a mixture value of 0.888.

## SVM Model

Next we will use a SVM model. We use SVM as we want to predict the quality of wine which has 3 possible classes. Therefore, SVM is used to help distinguish wine into one of these classes based on our predictors' values. We will tune the hyperparameter of cost.

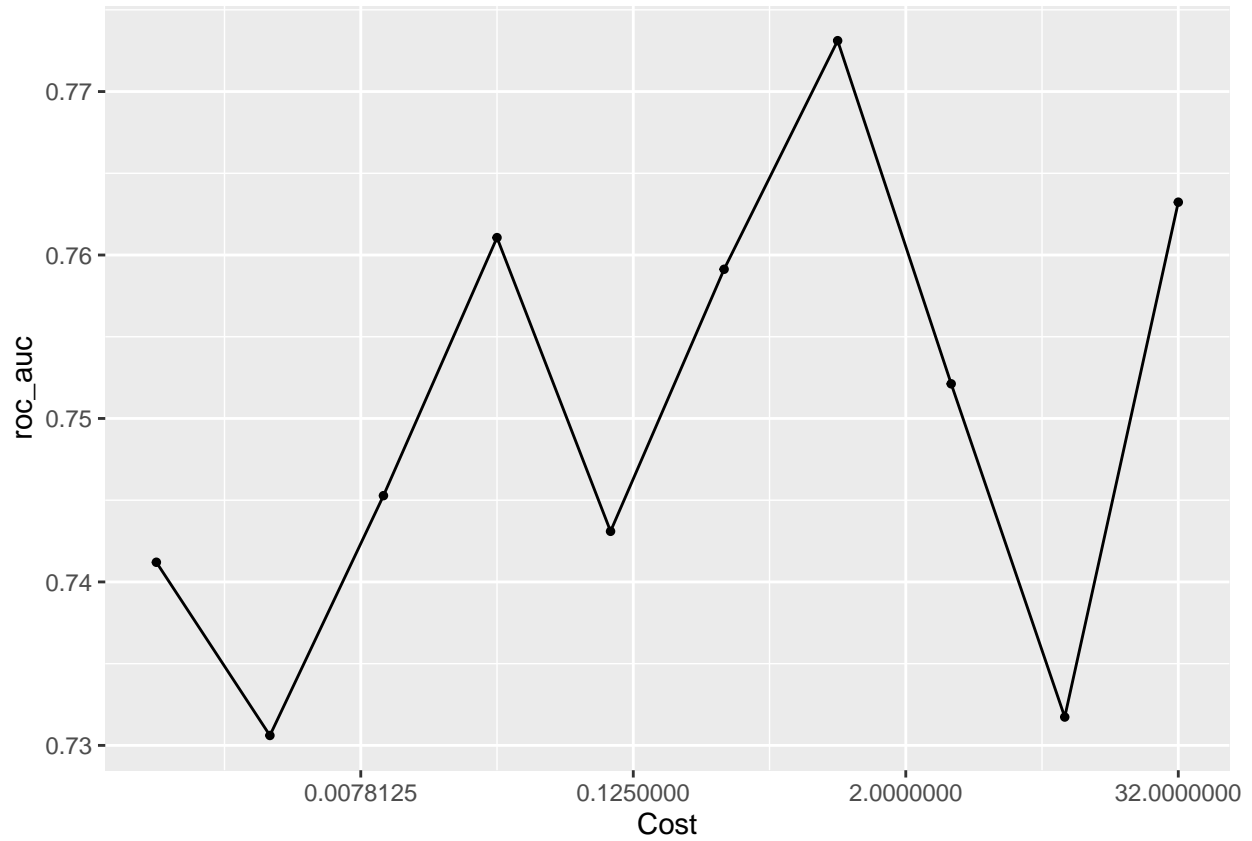


Table 4: First Few Rows Of Best Performing SVM Models And Their Statistics

Metric	ROC_AUC	Standard_Error	Cost_Value
roc_auc	0.7731128	0.0204784	1.0000000
roc_auc	0.7632320	0.0162996	32.0000000
roc_auc	0.7610650	0.0127777	0.0312500
roc_auc	0.7591280	0.0140259	0.3149803
roc_auc	0.7521204	0.0195804	3.1748021
roc_auc	0.7452783	0.0088354	0.0098431

From our table we can see that our best SVM model had an accuracy of 0.773 with a standard error of 0.020. However, this model did not outperform our best elastic net model based only on accuracy. Therefore, we will not use it.

### Basic Tree With Tuned Hyperparameters

Now we will show a basic tree its tree graph. Since we have 11 predictors there a lot of ways we can input them. Thus we want to see how different inputs can lead to different predictions / results. We will tune on the cost\_complexity hyperparameter from -1 to 3

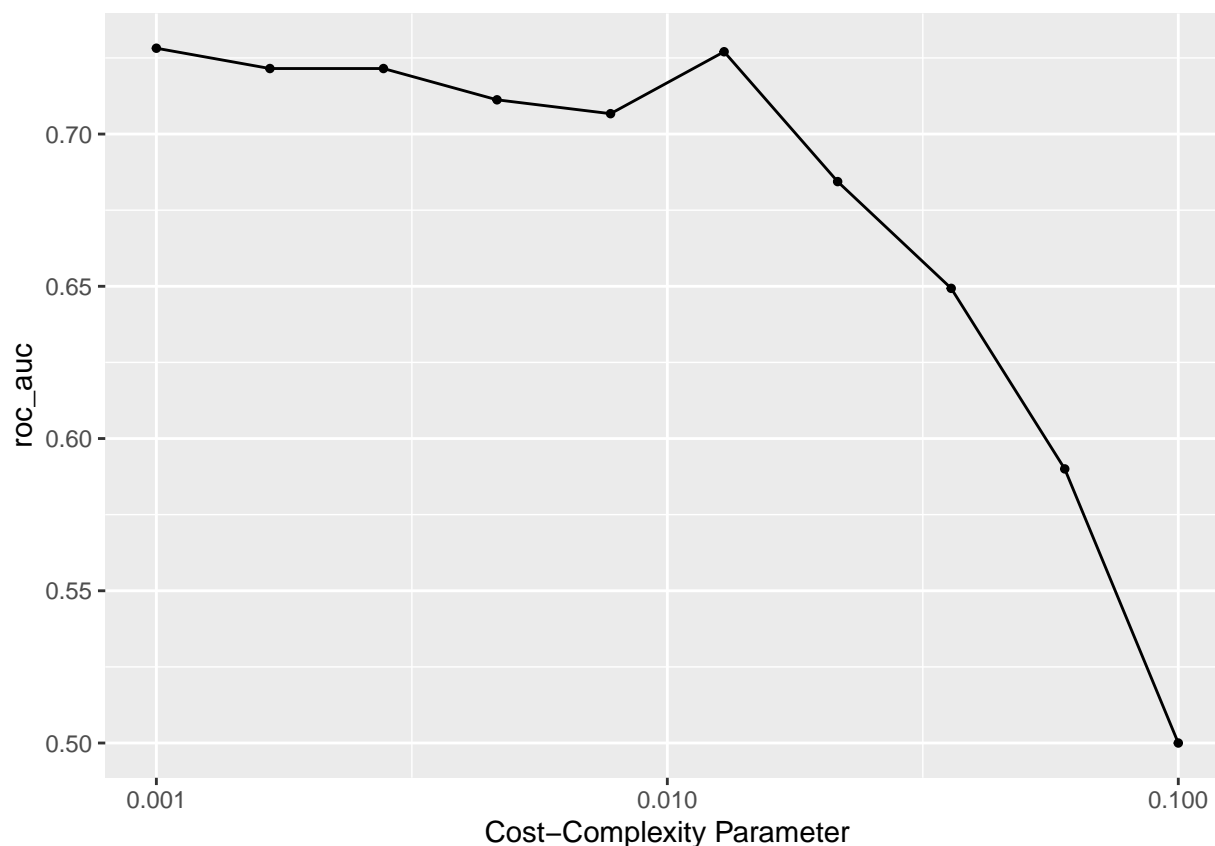


Table 5: First Few Rows of Best Performing Tree Models and Their Statistics

Metric	Value	Standard_Error	Cost_Complexity_Value
roc_auc	0.7281918	0.0205565	0.0010000
roc_auc	0.7270298	0.0210289	0.0129155
roc_auc	0.7215154	0.0234001	0.0016681
roc_auc	0.7215154	0.0234001	0.0027826
roc_auc	0.7112431	0.0220622	0.0046416
roc_auc	0.7066879	0.0213707	0.0077426

As we can see our best performing tree model had a ROC\_AUC of 0.728 with a standard error of 0.02 and cost complexity of 0.001. Similar to our SVM model, it did not outperform our elastic net model in terms of ROC\_AUC. Even though it did not outperform our elastic net model, let's take a look at what the decision tree would look like.





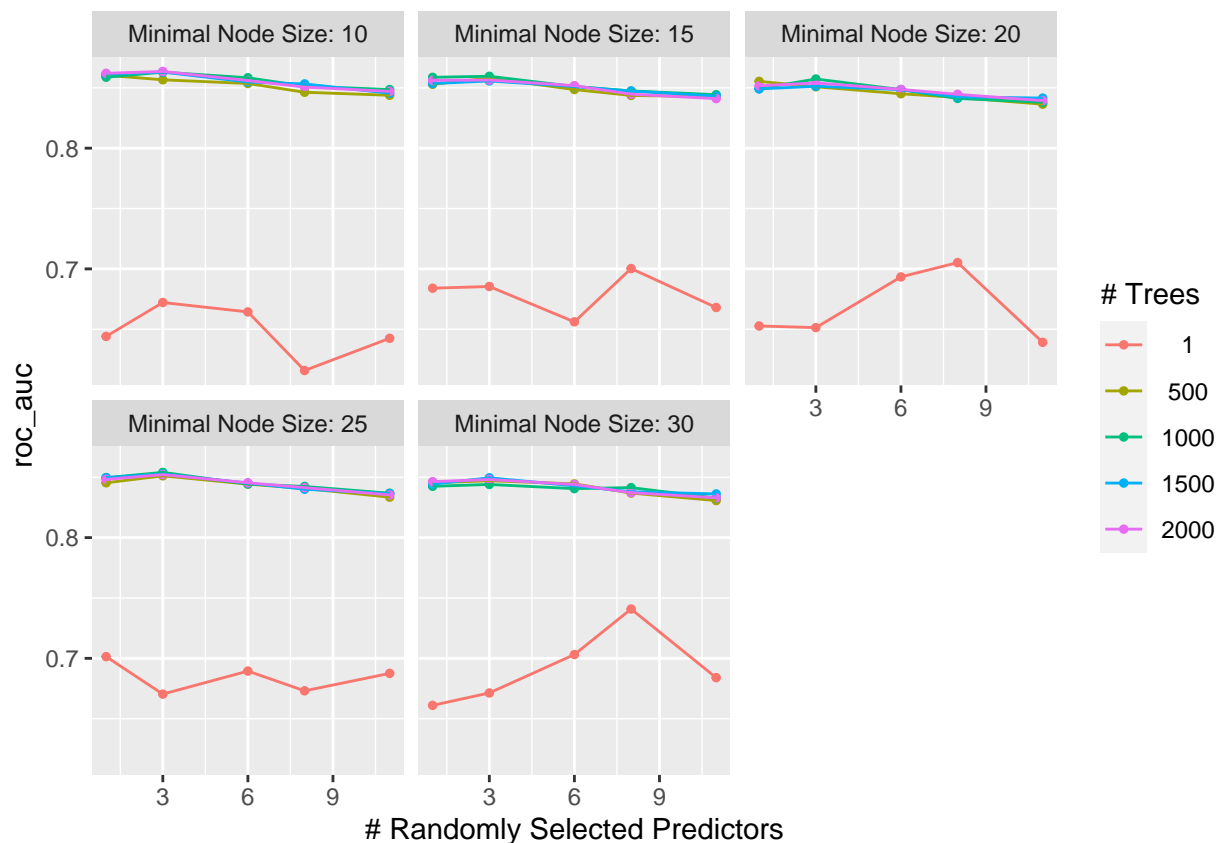


Table 6: First Few Rows of Best Performing Random Forest Models and Their Statistics

Metric	Value	Standard_Error	mtry_Value	trees_Value	min_n_Value
roc_auc	0.8635622	0.0177428	3	2000	10
roc_auc	0.8630427	0.0172771	3	1500	10
roc_auc	0.8629743	0.0176220	3	1000	10
roc_auc	0.8620333	0.0164444	1	2000	10
roc_auc	0.8613143	0.0159637	1	1500	10
roc_auc	0.8604194	0.0171058	1	500	10

As we can see our more complex tree with tuned hyperparameters outperformed the elastic net model. It had a roc\_auc of 0.863, standard error of 0.017 with a mtry value of 3, trees values of 2000, and min\_n value of 10. Since  $mtry \neq 11$  we know that our model is not a bagging model. Therefore our new current best model is this random forest model.

### Boosted Tree Model

Our last model is a boosted tree model. Similar to our random forest model, since we have a large number of observations using a boosted tree is not a bad idea. We tune the hyperparameter `trees` from 10 to 2000. Let us see how it performs.

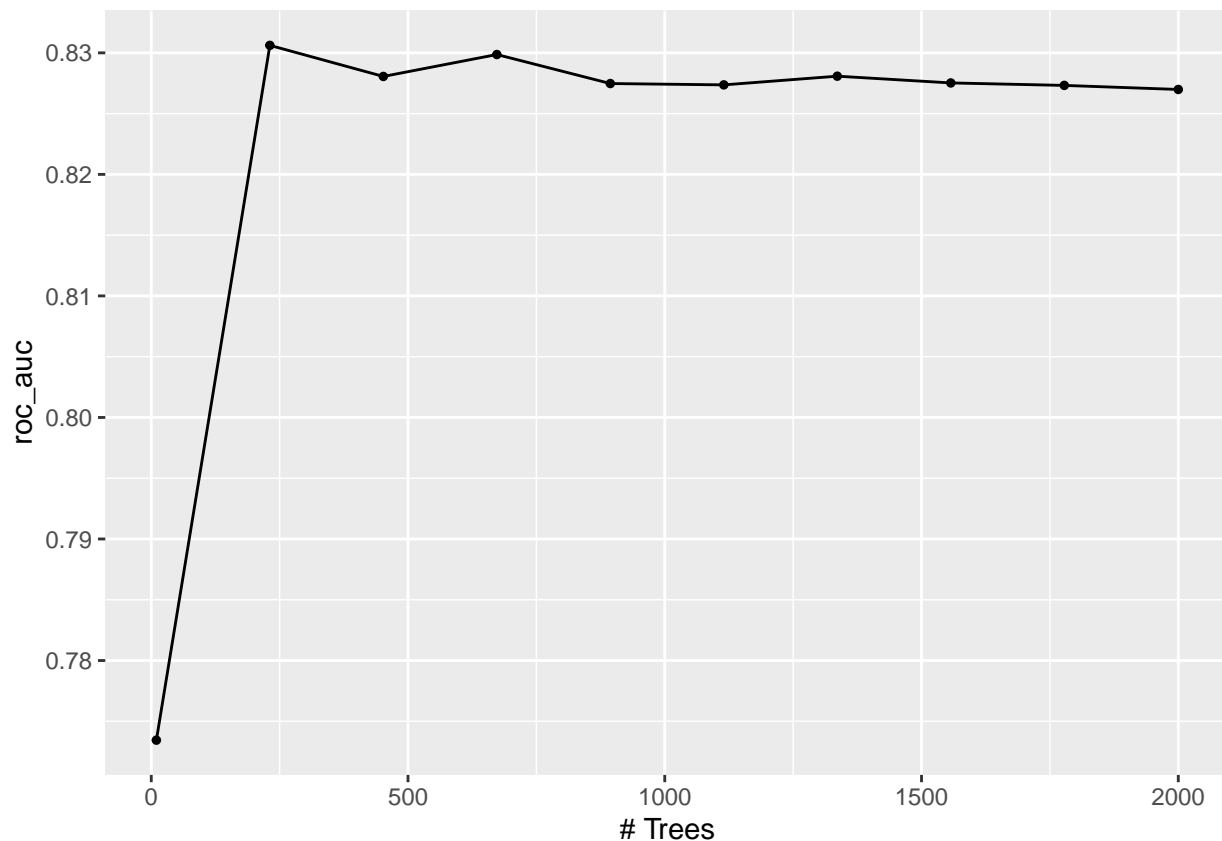


Table 7: First Few Rows of Best Performing Boosted Tree Models and Their Statistics

Metric	Value	Standard_Error	trees_Value
roc_auc	0.8306177	0.0190000	231
roc_auc	0.8298615	0.0203476	673
roc_auc	0.8280812	0.0224096	1336
roc_auc	0.8280581	0.0205470	452
roc_auc	0.8275245	0.0226205	1557
roc_auc	0.8274754	0.0216183	894

With our last model, its best performing model had a roc\_auc of 0.83 with a standard error of 0.01 and a trees value of 231. Compared to our random forest model in terms of roc\_auc it performed worse. Therefore we will finish our model testing with our random forest model performing the best.

## Results Of Models, Best Model Selection, And Applying Best Model

Let us recap the roc\_auc of all our models.

Models	ROC_AUC_Values	Standard_Error
Random Forest and Bagging	0.8635622	0.0177428
Boosted Tree	0.8306177	0.0190000
Elastic Net	0.8197134	0.0212351

Models	ROC_AUC_Values	Standard_Error
SVM	0.7731128	0.0204784
Basic Tree	0.7281918	0.0205565

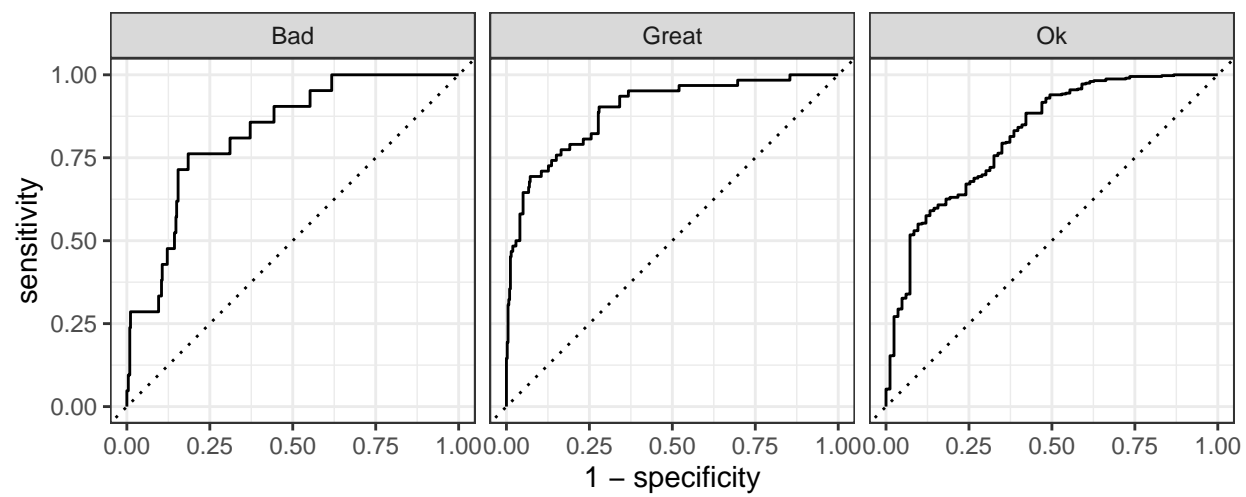
As we can see, again, the random forest had the best roc\_auc and lowest standard error. Let's see the values of the best model's hyperparameters

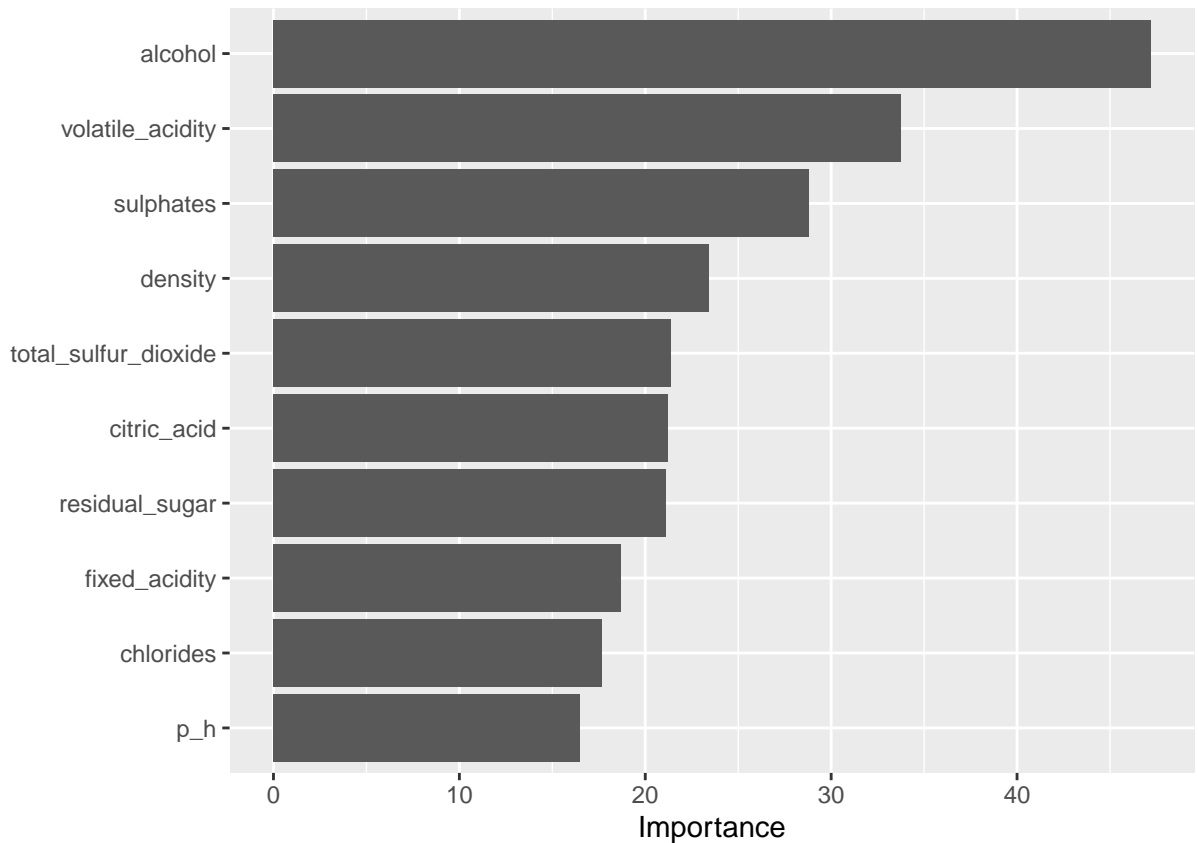
mtry	trees	min_n	.config
3	2000	10	Preprocessor1_Model022

Now let us proceed with fitting this model and using it against the testing data.

Metric	Testing_AUC_ROC	Training_ROC_AUC
roc_auc	0.8413422	0.8635622

Prediction	Bad -	0	0	0
	Ok -	20	391	33
	Great -	1	7	29
		Bad	Ok	Great
		Truth		





The final model had a slightly lower AUC\_ROC by around 0.03. Overall it performed well. From the conf matrix we can see that it correctly predicted almost all of the values. It seemed to predict Great quality wine the worse, but predicted Ok and Bad wine well. From all 3 ROC curves we can see how up and left they are which indicates that the models performed well. Our VIP plot goes along with what I have learned about wine making throughout this project.

Therefore, our final model had a final ROC\_AUC of 0.838. This means that our model favors a random positive example higher than a negative random example 83.8% of the time. This is clearly much better than 50/50 guessing.

## Conclusion

Overall I believe my model performed well. We took several possible models, trained them on the training data and got a roc\_auc of 0.838. Our final model was a random forest model with mtry value of 3, trees value of 2000, and min\_n value of 10. I believe that if we added more interaction terms or scaled more / less predictors we would have had a more accurate model.

## References

All homeworks for code All labs for concepts and code <https://winefolly.com/deep-dive/how-is-red-wine-made/> for how wine is made which led to the conclusion of interaction between residual\_sugar and sulphates. <https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009> for the dataset

## Appendix

```
library(tidymodels)
library(tidyverse)
library(ISLR)
library(ISLR2)
library(janitor)
library(Rmisc)
library(ggplot2)
library(ISLR)
library(glmnet)
library(rpart.plot)
library(vip)
library(randomForest)
library(corrplot)
library(corr)
library(xgboost)
library(dials)
library(dplyr)
library(ranger)
library(discrim)
library(MASS)
library(kernlab)
library(Hmisc)
library(knitr)
tidymodels_prefer()

data <- read_csv('C:/Users/peter/Desktop/231Final/wine.csv')

data <- clean_names(data)

data$fqual <- with(data, ifelse(data$quality > 6, 'Great',
                              ifelse(data$quality > 4, 'Ok',
                              ifelse(data$quality > 2, 'Bad'))))

data$fqual <- ordered(as.factor(data$fqual), levels = c('Bad','Ok','Great'))

data <- data %>% select(-quality)

cor_data <- data %>%
  select(-fqual) %>%
  correlate()

cor_data %>%
  stretch() %>%
  ggplot(aes(x,y, fill = r)) + geom_tile() + geom_text(aes(label = as.character(fashion(r))))

data %>%
  select(where(is.numeric)) %>%
  cor() %>%
  corrplot(type = 'lower', diag = TRUE,
          method = 'square')
```

```

summDat <- summary(data %>% select(-fqual))
kable(summDat)

par(mfrow = c(1,3))
hist(data$residual_sugar, prob = TRUE)
lines(density(data$residual_sugar), col = 4, lwd = 2)

hist(data$chlorides, prob = TRUE)
lines(density(data$chlorides), col = 4, lwd = 2)

hist(data$free_sulfur_dioxide, prob = TRUE)
lines(density(data$free_sulfur_dioxide), col = 4, lwd = 2)

hist(data$total_sulfur_dioxide, prob = TRUE)
lines(density(data$total_sulfur_dioxide), col = 4, lwd = 2)

hist(data$sulphates, prob = TRUE)
lines(density(data$sulphates), col = 4, lwd = 2)

set.seed(100)

data_split <- initial_split(data, strata = fqual, prop = 0.7)
data_train <- training(data_split)
data_test <- testing(data_split)

data_train_fold <- vfold_cv(data_train, v = 10, strata = fqual)

data_rec <- recipe(fqual ~ fixed_acidity + volatile_acidity + citric_acid + residual_sugar + chlorides +
  step_dummy(all_nominal_predictors()) %>%
  step_scale(residual_sugar, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, sulphates) %>%
  step_interact(residual_sugar ~ sulphates)

mn_model <- multinom_reg(penalty = tune(), mixture = tune()) %>%
  set_engine('glmnet') %>%
  set_mode('classification')

tuned_mn_wf <- workflow() %>%
  add_recipe(data_rec) %>%
  add_model(mn_model)

degree_grid <- grid_regular(penalty(range = c(-5,5)),mixture(range = c(0,1)), levels = 10)

#tune_mn_res <- tune_grid(object = tuned_mn_wf, resamples = data_train_fold, grid = degree_grid, metric =
#Save values to avoid rerunning everytime
#save(tune_mn_res, file = 'tune_mn_res.rda')

load(file = 'tune_mn_res.rda')

autoplot(tune_mn_res)

```



```

mn_metrics <- collect_metrics(tune_mn_res) %>% dplyr::arrange(desc(mean))

#Create Table
df <- data.frame(Metric = mn_metrics$.metric, Value = mn_metrics$mean, Standard_Error = mn_metrics$std_err)

kable(head(df), caption = 'First Few Rows Of Best Performing Elastic Net Models And Their Statistics')

best_mn <- df[1,2]
mn_se <- df[1,3]

svm_linear_spec <- svm_poly(degree = 1) %>%
  set_mode("classification") %>%
  set_engine("kernlab", scaled = FALSE)

svm_linear_wf <- workflow() %>%
  add_model(svm_linear_spec %>% set_args(cost = tune())) %>%
  add_formula(fequal ~ fixed_acidity + volatile_acidity + citric_acid + residual_sugar + chlorides + free_sulfur_dioxide)

param_grid <- grid_regular(cost(), levels = 10)

#tune_svm_res <- tune_grid(svm_linear_wf, resamples = data_train_fold, grid = param_grid, metrics = metric_s)

#Save results
#save(tune_svm_res, file = 'tune_svm_res.rda')

#Load Results
load(file = 'tune_svm_res.rda')

autoplot(tune_svm_res)

svm_metrics <- collect_metrics(tune_svm_res) %>% dplyr::arrange(desc(mean))

#Create Table
df <- data.frame(Metric = svm_metrics$.metric, ROC_AUC = svm_metrics$mean, Standard_Error = svm_metrics$std_err)
kable(head(df), caption = 'First Few Rows Of Best Performing SVM Models And Their Statistics')

best_svm <- df[1,2]
svm_se <- df[1,3]

tree_spec <- decision_tree() %>%
  set_engine('rpart') %>%
  set_mode('classification')

tree_wf <- workflow() %>%
  add_model(tree_spec %>% set_args(cost_complexity = tune())) %>%
  add_formula(fequal ~ fixed_acidity + volatile_acidity + citric_acid + residual_sugar + chlorides + free_sulfur_dioxide)

tree_param <- grid_regular(cost_complexity(range = c(-3,-1)), levels = 10)

#tune_tree_res <- tune_grid(tree_wf, resamples = data_train_fold, grid = tree_param, metrics = metric_s)

#Save Results
#save(tune_tree_res, file = 'tune_tree_res.rda')

```

```

#load results
load(file = 'tune_tree_res.rda')

autoplot(tune_tree_res)

tree_metrics <- collect_metrics(tune_tree_res) %>% dplyr::arrange(desc(mean))

#Create Table
df <- data.frame(Metric = tree_metrics$.metric, Value = tree_metrics$mean, Standard_Error = tree_metrics$std_err)

kable(head(df), caption = 'First Few Rows of Best Performing Tree Models and Their Statistics')

best_tree <- df[1,2]
tree_se <- df[1,3]

best_tree <- select_best(tune_tree_res, metric = 'roc_auc')

tree_final <- finalize_workflow(tree_wf, best_tree)

tree_final_fit <- fit(tree_final, data = data_train)

tree_final_fit %>%
  extract_fit_engine() %>%
  rpart.plot()

bagging_spec <- rand_forest() %>%
  set_engine("ranger", importance = 'impurity') %>%
  set_mode("classification")

rf_wf <- workflow() %>%
  add_model(bagging_spec %>% set_args(mtry = tune(), trees = tune(), min_n = tune(), importance = 'impurity'))
  add_formula(fqual ~ fixed_acidity + volatile_acidity + citric_acid + residual_sugar + chlorides + free_sulfur_dioxide)

rf_param <- grid_regular(mtry(range = c(1,11)), trees(range = c(1,2000)), min_n(range = c(10,30)), level_1 = 5, level_2 = 5)

tune_rf_res <- tune_grid(rf_wf, resamples = data_train_fold, grid = rf_param, metrics = metric_set(roc_auc, accuracy))

#Save results
#save(tune_rf_res, file = 'tune_rf_res.rda')

#load results
load(file = 'tune_rf_res.rda')
autoplot(tune_rf_res)

rf_metrics <- collect_metrics(tune_rf_res) %>% dplyr::arrange(desc(mean))

#Create Table
df <- data.frame(Metric = rf_metrics$.metric, Value = rf_metrics$mean, Standard_Error = rf_metrics$std_err)

kable(head(df), caption = 'First Few Rows of Best Performing Random Forest Models and Their Statistics')

best_rf <- df[1,2]

```

```

rf_se <- df[1,3]

boost_spec <- boost_tree(trees = tune()) %>%
  set_engine('xgboost') %>%
  set_mode('classification')

boost_wf <- workflow() %>%
  add_model(boost_spec) %>%
  add_recipe(data_rec)

boost_param <- grid_regular(trees(range = c(10,2000)), levels = 10)

#tune_boost_res <- tune_grid(boost_wf, resamples = data_train_fold, grid = boost_param, metrics = metri

#Save results
#save(tune_boost_res, file = 'tune_boost_res.rda')

#load results
load(file = 'tune_boost_res.rda')

autoplot(tune_boost_res)

boost_metrics <- collect_metrics(tune_boost_res) %>% dplyr::arrange(-mean)

#Create Table
df <- data.frame(Metric = boost_metrics$.metric, Value = boost_metrics$mean, Standard_Error = boost_metr

kable(head(df), caption = 'First Few Rows of Best Performing Boosted Tree Models and Their Statistics')

best_boost <- df[1,2]
boost_se <- df[1,3]

models <- c('Elastic Net', 'SVM', 'Basic Tree', 'Random Forest and Bagging', 'Boosted Tree')
vals <- c(best_mn, best_svm, best_tree, best_rf, best_boost)
se <- c(mn_se, svm_se, tree_se, rf_se, boost_se)
#Create table

df <- data.frame(Models = models, ROC_AUC = vals, Standard_Error = se)
kable(df %>% dplyr::arrange(-ROC_AUC))

#Create table
best_vals <- select_best(tune_rf_res)
kable(best_vals)

rf_final <- finalize_workflow(rf_wf, best_vals)

rf_final_fit <- fit(rf_final, data = data_train)

final_res <- augment(rf_final_fit, new_data = data_test) %>%
  roc_auc(fqual, estimate = .pred_Bad:.pred_Great)

```

```

#Create table
df <- data.frame(Metric = final_res$.metric, Testing_AUC_ROC = final_res$.estimate, Training_ROC_AUC = 1)
kable(df)

augment(rf_final_fit, new_data = data_test) %>%
  conf_mat(fqual, estimate = .pred_class) %>%
  autoplot(type = 'heatmap')

augment(rf_final_fit, new_data = data_test) %>%
  roc_curve(fqual, .pred_Bad:.pred_Great) %>%
  autoplot()

vip::vip(rf_final_fit)%>% extract_fit_parsnip()

```