White wine quality model - capstone project

Anna Win-Mason

29 May 2025

Introduction

This report forms part of the capstone certificate for the professional data science certificate through HarvardX. I have used a dataset of my own choice to explore a machine learning problem using different machine learning algorithms. I was motivated in looking at a chemistry related machine learning problem because of my background in chemical research prior to switching to starting a career in data science. Therefore, I chose to investigate the white wine quality dataset.(Cortez 2009a),(Cortez 2009b)

The white wine quality dataset contains physicochemical properties of wine plus a quality score. More details about this dataset are provided in the exploratory data analysis section. There are a number of different machine learning projects I could perform with this dataset. I could investigate classification machine learning methods, in which I could wrangle the wine quality dataset to include a category field of 'good' vs 'not good' for each wine. This category could then be determined by setting a quality score threshold for 'good'. However, I have decided to develop a wine quality score predictive model using regression techniques.

An overview of the key steps used in the project:

1. Download and wrangle the white wine quality dataset.
2. Exploratory data analysis to understand the variables and their distributions in the dataset.
3. Create a test set and training set to develop and validate the wine quality prediction score models.
4. Develop a wine quality prediction score model by starting with a regressive baseline algorithm, then explore additional machine learning algorithms in an iterative way to refine towards a final model.
5. Use the root mean square error (RMSE) as a benchmark metric to assess and compare the effectiveness of each algorithm.
6. Make wine quality score predictions with the final model.

Methods and analysis

White wine quality dataset

We start by downloading and preparing the white wine quality dataset for use in developing the machine learning algorithm with the below code.

```r
## Import and tidy the white Wine quality dataset

## sourced from https://archive.ics.uci.edu/dataset/186/wine+quality

# dataset citation: P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.
# Modeling wine preferences by data mining from physicochemical properties.
# In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

# import the white wine csv file
winequal_temp_w <- read_csv("winequality-white.csv")

# use tidyr::separate to separate the one character column into 12 columns of data
# remove spaces from the column names by adding an underscore, so that there no
# problems running the machine learning algorithms
# (rf had an error when there were spaces in the column names)

winequal_white <- winequal_temp_w %>%
  separate(col = `fixed acidity;volatile acidity;citric acid;residual sugar;chlorides;free sulfur dioxi
           into = c("fixed_acidity", "volatile_acidity", "citric_acid",
                    "residual_sugar", "chlorides", "free_sulfur_dioxide",
                    "total_sulfur_dioxide", "density", "pH", "sulphates",
                    "alcohol", "quality"),
           sep = ";")

# convert from character to integer
winequal_white <- winequal_white %>% mutate_if(is.character, as.numeric)
```

Exploratory data analysis of *winequal_white* dataset

Let's look into the *winequal_white* dataset in more detail. Each row of data in the *winequal_white* dataset corresponds to the physiochemical properties and quality score for one white variant of the Portuguese "Vinho Verde" wine.

```r
# the number of rows and columns in winequal_white
dim(winequal_white)
```

```
## [1] 4898   12
```

There are 12 numeric variables for each row of data in *winequal_white* detailed in table 1. This includes 11 physicochemical attributes of each wine. These attributes provide insights into the chemical composition and physical properties of the wine, which are ultimately responsible for its sensory quality.(Curada 2023)

Table 1: Definitions of *winequal_white* dataset variables, 11 input variables (physicochemical attributes) and 1 output variable (sensory attribute) of white wine.

Variables	Definitions
Fixed acidity	(grams per litre) amount of non-volatile acidity, primarily from tartaric acid. A more acidic wine can influence flavour
Volatile acidity	(grams per litre) amount of acetic acid, at high levels can lead to an unpleasant taste

Variables	Definitions
Citric acid	(grams per litre) amount of citric acid, found in grape juice it can add 'freshness' and flavor to wines
Residual sugar	(grams per litre) amount of sugar remaining after fermentation stops, plays a crucial role in determining wine sweetness
Chlorides	(grams per litre) amount of chloride salts, can influence the wine's salinity and balance
Free sulfur dioxide	(milligrams per litre) amount of unbound sulfur dioxide molecules, and is a common preservative, high levels can lead to unpleasant taste
Total sulfur dioxide	(milligrams per litre) total amount of sulfur dioxide, free and bound forms. Important for maintaining wine quality, but high levels can lead to unpleasant taste
Density	(grams per cubic centimetre) measures the mass per unit volume of wine, influenced by factors like alcohol and sugar content.
Sulphates	(grams per litre) amount of sulfate salts, they are naturally present in grapes and can be added. High levels may contribute to a bitter taste in the wine
Alcohol	(volume percentage) amount of alcohol by volume in the wine. Alcohol is a primary factor influencing the wine's flavour, body and complexity. Higher levels generally result in fuller-bodied wines with stronger aromas and flavours
Quality	sensory data as an integer score between 0 and 10

The summary statistics of each variable in the *winequal_white* dataset are shown below. Of note is that the quality scores range from 3 to 9, with and average of 5.9. Also the minumum alcohol by volume wine is 8%, and maximum is 14.2%.

```r
# summary of winequal_white dataset
summary(winequal_white)
```

```
##  fixed_acidity    volatile_acidity  citric_acid     residual_sugar
##  Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600
##  1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700   1st Qu.: 1.700
##  Median : 6.800   Median :0.2600   Median :0.3200   Median : 5.200
##  Mean   : 6.855   Mean   :0.2782   Mean   :0.3342   Mean   : 6.391
##  3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900   3rd Qu.: 9.900
##  Max.   :14.200   Max.   :1.1000   Max.   :1.6600   Max.   :65.800
##    chlorides       free_sulfur_dioxide total_sulfur_dioxide    density
##  Min.   :0.00900   Min.   :  2.00      Min.   :  9.0        Min.   :0.9871
##  1st Qu.:0.03600   1st Qu.: 23.00      1st Qu.:108.0        1st Qu.:0.9917
##  Median :0.04300   Median : 34.00      Median :134.0        Median :0.9937
##  Mean   :0.04577   Mean   : 35.31      Mean   :138.4        Mean   :0.9940
##  3rd Qu.:0.05000   3rd Qu.: 46.00      3rd Qu.:167.0        3rd Qu.:0.9961
##  Max.   :0.34600   Max.   :289.00      Max.   :440.0        Max.   :1.0390
##        pH          sulphates         alcohol         quality
##  Min.   :2.720   Min.   :0.2200   Min.   : 8.00   Min.   :3.000
##  1st Qu.:3.090   1st Qu.:0.4100   1st Qu.: 9.50   1st Qu.:5.000
##  Median :3.180   Median :0.4700   Median :10.40   Median :6.000
##  Mean   :3.188   Mean   :0.4898   Mean   :10.51   Mean   :5.878
##  3rd Qu.:3.280   3rd Qu.:0.5500   3rd Qu.:11.40   3rd Qu.:6.000
##  Max.   :3.820   Max.   :1.0800   Max.   :14.20   Max.   :9.000
```

Data distributions of *winequal_white*

Now we will look at the distributions of a selection of three variables in the *winequal_white* dataset. First we will visualise the distribution of the actual quality score in the *winequal_white* dataset (figure 1). Here there are far fewer low quality and high quality wines, with the majority of wines scoring between 5 and 7, approximating a normal distribution around 6. Of note, the quality scores are integer values only, with no half scores.

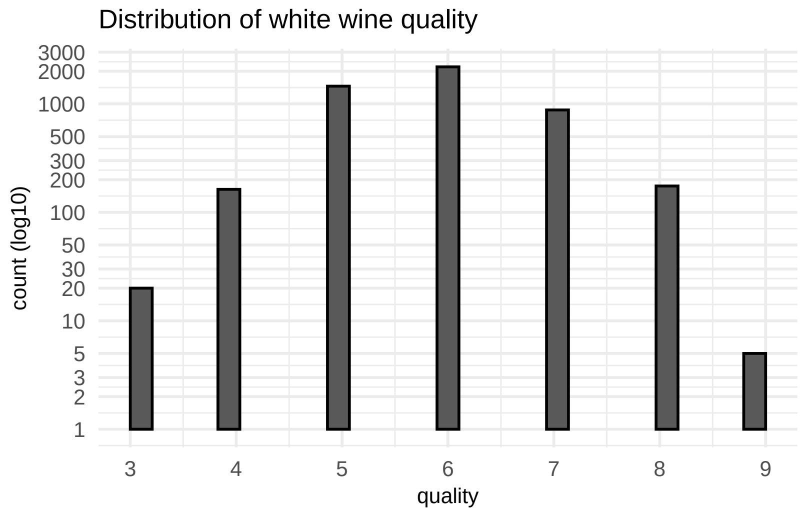


Figure 1: The distribution of the quality scores per wine in winequal_white

Next we'll look at the distribution of the alcohol variable in *winequal_white* (figure 2). In the description of the dataset, the amount of alcohol is a primary factor influencing taste. The wines in this dataset have an alcohol percentage that does not follow a normal distribution, with the majority of wines falling between about 9 to 12.5%.

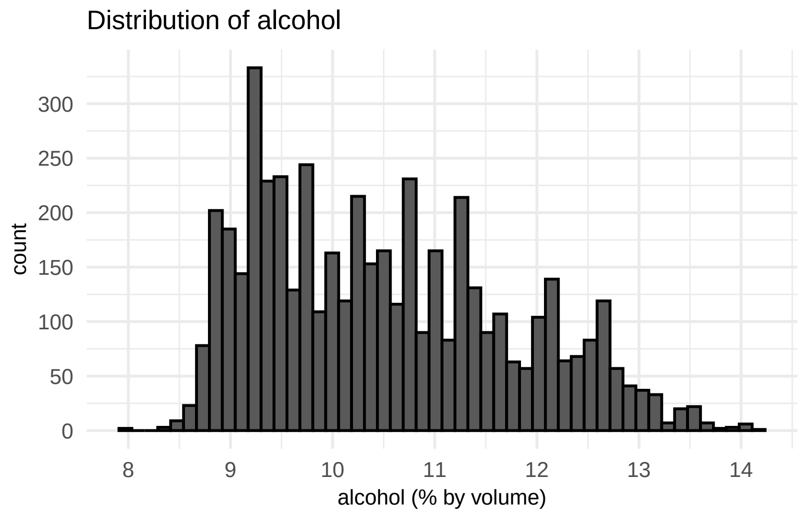


Figure 2: The distribution of the alcohol percentage per wine in winequal_white

The distribution of the density variable in *winequal_white* is shown in figure 3. The density of wine is affected by alcohol and sugar content. The wines in this dataset have a measure of density with a distribution around

0.994 g/cubic cm, ranging between 0.989 to 1.000 g/cubic cm.

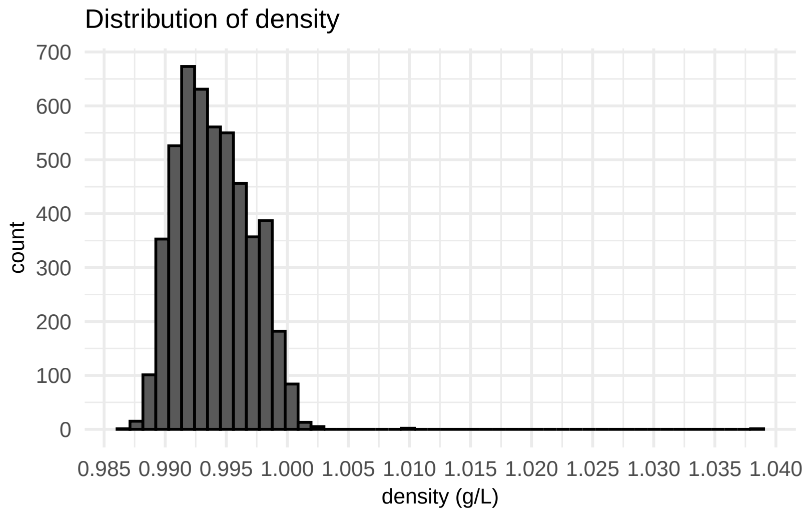


Figure 3: The distribution of the density per wine in winequal_white

Data content of *winequal_white* as motivation for the predicitive quality model

Given that the aim of this project is to develop a machine learning model to predict the quality of wines from using their physicochemical properties, we will endeavor to find the minimum number of predictors or features needed in the model. We can explore which variables have the most influence on the quality score through determining the correlation coefficient of the quality score with each of the 11 variables. In table 2, alcohol has the largest correlation coefficent at 0.436, followed by density at -0.307, then chlorides at -0.210. However, none of the variables has a strong correlation with the quality score as they are all under 0.5.

Table 2: Correlation coefficients of 11 variables with quality score

ID	variable	correlation_coefficient
1	fixed_acidity	-0.1136628
2	volatile_acidity	-0.1947230
3	citric_acid	-0.0092091
4	residual_sugar	-0.0975768
5	chlorides	-0.2099344
6	free_sulfur_dioxide	0.0081581
7	total_sulfur_dioxide	-0.1747372
8	density	-0.3071233
9	pH	0.0994272
10	sulphates	0.0536779
11	alcohol	0.4355747

We can then visualise plots of the quality score versus the wine physicochemical properties with the largest correlation coefficient. I've chosen to present graphs for the top three largest correlation coefficients. Figure 4 presents the graph of quality versus alcohol for each wine in the *winequal_white* dataset. The relationship between the two variables is represented by slope of the regression line.

Figure 5 presents the graph of quality versus density for each wine in the *winequal_white* dataset.

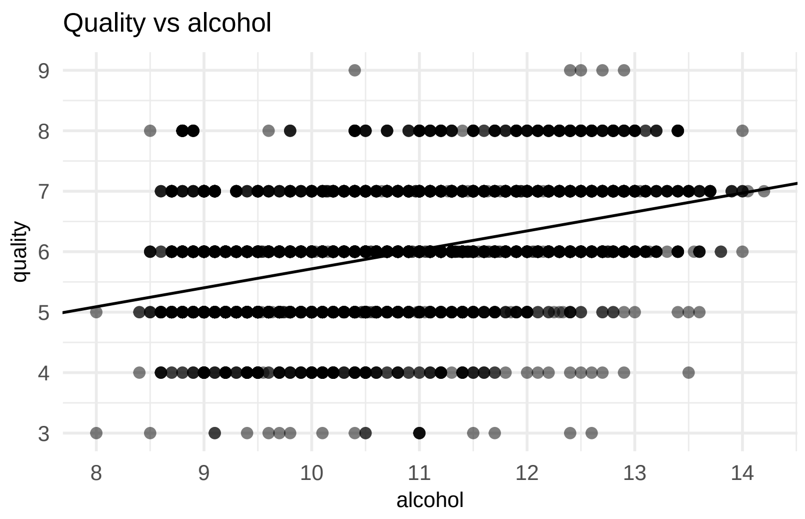


Figure 4: The quality score versus percentage of alcohol for each wine in winequal_white

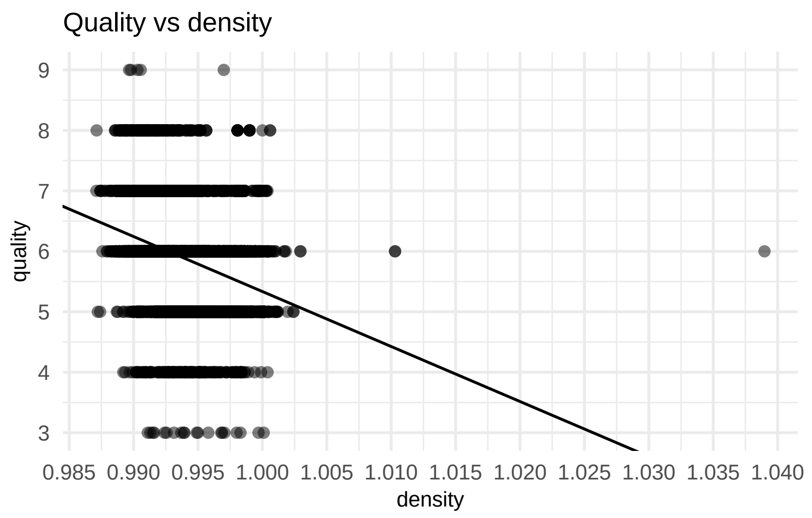


Figure 5: The quality score versus density for each wine in winequal_white

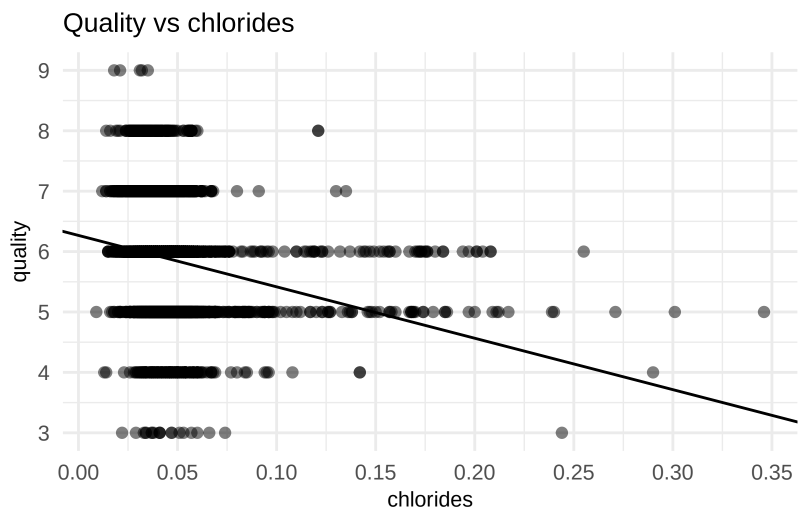


Figure 6: The quality score versus chlorides for each wine in winequal_white

Figure 6 presents the graph of quality versus chlorides for each wine in the *winequal_white* dataset.

Of all three graphs (alcohol, density and chlorides) the regression line for quality vs alcohol has the largest slope, this may indicate a stronger relationship between the variables. When developing the wine quality predictive model we will investigate which variables are needed to provide the most useful model. Noting that the correlation coefficient is a crude measure here, and trial and error may be the best way to test which variables are most useful in the algorithms.

Machine learning strategy

Table 3 shows the machine learning algorithms that I chose to explore to build the wine quality predictive model. These methods increase in computational complexity and resources. It may be that the first method, generalised linear regression, will be sufficient for the final model. Running a random forest algorithm does take a lot more computational time.

Table 3: Machine learning algorithms to develop wine quality predictive model

iteration	machine_learning_algorithm
1	Generalised linear regression
2	Regression trees
3	Random forests

Determing model accuracy

We need a metric to determine the accuracy of the predictions and to evaluate usefulness of the machine learning algorithms. One on the most common evaluation metrics is the root mean square error or RMSE.(Jonatasv 2024) RMSE measures the average difference between a model's actual and predicted values. Mathematically, it is the standard deviation of the differences. The lower the RMSE the less error in the model.

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}\left(\hat{y}_{u,i} - y_{u,i}\right)^2}$$

RMSE can be defined by the following R code:

```r
# RMSE written out as an R function.
# RMSE is the sqrt of the mean of (true_ratings - predicted_ratings) squared
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

Model validation

To develop the wine quality predictive model using machine learning methods, it is best practice to take the *winequal_white* dataset and split it further into a *winequal_white*(80%) and a *winequal_white* (20%). This is needed to avoid over-training the data. For each method iteration we will use the train_set to develop the algorithm, then use the test_set to make ratings predictions using the algorithm, with the prediction accuracy evaluated using RMSE.

```r
# Create training and test datasets from winequal_white
# for use in developing machine learning algorithm

# set the seed with rounding
set.seed(1, sample.kind="Rounding")

# create the partition index using caret::createDataPartion function
winequal_test_index <- createDataPartition(y = winequal_white$quality, times = 1,
                                           p = 0.2, list = FALSE)

# use the partition index to create the train_set and test_set
winequal_train_set <- winequal_white[-winequal_test_index,] # 80% of winequal_white data
winequal_test_set <- winequal_white[winequal_test_index,] # 20% of winequal_white data
```

Building the predictive model

In developing the final model, We will use the *train* function from the *caret* package to train the algorithms. The *train* function behaves like a wrapper for a variety of machine learning methods, and will come in handy later on with more complex algorithms as it allows for parameter tuning and cross-validation in a sytematic manner.(Irizarry 2019a)

Generalised linear regression

We will start with a generalised linear regression model to develop a baseline algorithm for the predictive model.(Irizarry 2019b) Initially we will try just using 5 physicochemical properties as predictors in the model. I have chosen these based on likely prpoerties that will affect the taste. We will use the *winequal_train_set* to train the algorithm, and the *winequal_test_set* to make the predictions. The RMSE for just 5 predictors, which is also calculated on the *winequal_test_set*, is 0.7702957.

```r
# Baseline model with generalised linear regression (glm) with 5 predictors

# using caret::train to calculate the algorithm
# predict quality using 5 predictors
train_5pred_glm <- train(quality ~  alcohol + chlorides + residual_sugar +
                           volatile_acidity +
                           total_sulfur_dioxide,
                         method = "glm",
                         data = winequal_train_set)

# use predict function to make quality score predictions using trained algorithm
predict_train_5pred_glm <- predict(train_5pred_glm, winequal_test_set, type = "raw")

# calculate RMSE to evaluate the algorithm
rmse_5pred_glm <- RMSE(predict_train_5pred_glm, winequal_test_set$quality)
# print RMSE
rmse_5pred_glm
```

```
## [1] 0.7702957
```

Next we will test if using more predictors in the model will improve the RMSE. We will choose the 8 most likely properties as predictors based on their descriptions in table 1, and whether these are most likely to affect wine taste. Here there has been an improvement in algorithm, as the RMSE using 8 predictors has improved by decreasing to 0.769237.

```r
# Baseline model with generalised linear regression (glm) with 8 predictors

# using caret::train to calculate the algorithm
# predict quality using 8 predictors
train_8pred_glm <- train(quality ~  alcohol + chlorides + residual_sugar +
                           citric_acid + volatile_acidity + fixed_acidity +
                           total_sulfur_dioxide + sulphates,
                         method = "glm", # use glm algorithm
                         data = winequal_train_set) # on winequal train set

# use predict function to make quality score predictions using trained algorithm
predict_train_8pred_glm <- predict(train_8pred_glm, winequal_test_set, type = "raw")

# calculate RMSE to evaluate the algorithm
rmse_8pred_glm <- RMSE(predict_train_8pred_glm, winequal_test_set$quality)
# print RMSE
rmse_8pred_glm
```

```
## [1] 0.7692373
```

Let's see what happens when we use all 11 physicochemical properties of wine as predictors in the glm algorithm. Here, the performance of the algorithm has been improved further by using all the available properties of wine as predictors. This makes since considering that all 11 of the properties of wine had some value for the correlation coefficient with the wine quality score (table 2).

```r
# Baseline model with generalised linear regression (glm) with all predictors

# using caret::train to calculate the algorithm
train_allpred_glm <- train(quality ~  ., # predict quality using all other predictors
                           method = "glm", # use glm algorithm
                           data = winequal_train_set) # on winequal train set

# use predict function to make quality score predictions using trained algorithm
predict_train_allpred_glm <- predict(train_allpred_glm, winequal_test_set, type = "raw")

# calculate RMSE to evaluate the algorithm
rmse_allpred_glm <- RMSE(predict_train_allpred_glm, winequal_test_set$quality)
# print RMSE
rmse_allpred_glm
```

```
## [1] 0.7585215
```

By reducing the number of predictors in the model down to 5, this increases the RMSE (0.770295) compared to the glm model with 8 predictors. However, just 5 predictors is still more accurate than using all 11 predictors, as measured by RMSE. We will use this information going forward when investigating more computationally complex algorithms.

Summary of GLM models Table 4 depicts the results of the generalised linear model (GLM) performance as measured by RMSE. The algorithm using all chemical properties of wine as predictors was the best. Therefore, for the subsequent machine learning methods I will only develop algorithms using all 11 predictors.

Table 4: Generalised linear model iterations in order of best performance measured by RMSE

iteration	model_type	RMSE
1	GLM with 11 predictors	0.7585215
2	GLM with 8 predictors	0.7692373
3	GLM with 5 predictors	0.7702957

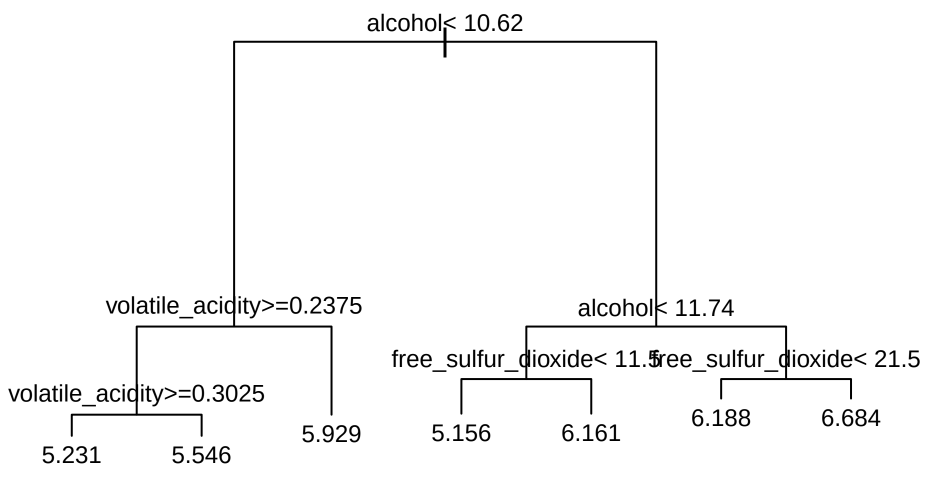
Regression trees

The next machine learning method we'll look at is regression trees.(Irizarry 2019b) This is a decision making algorithm which can be used to predict continuous valued outputs. The algorithm repeatedly splits the dataset into parts by selecting partitions based on predictors and at certain nodes that minimise the RMSE.(Prasad 2021) The idea is that the leaf node represents a predicted value for the target variable; in this case, the predicted quality score.

No tuning First we'll build a simple regression tree using the *rpart* function from the rpart package to fit a model without tuning any parameters. We'll start with using all 11 predictors as shown with the code below. Here the RMSE of 0.759828 is a bit higher than that of the best GLM algorithm. However, we can improve the regression tree further with tuning the parameters.

```r
# fit regression tree model using all predictors using rpart function
fit_regression_tree <- rpart(quality ~ ., data = winequal_train_set)
```

```r
# plot the regression tree model
plot(fit_regression_tree, margin = 0.1)
text(fit_regression_tree, cex = 0.75)
```



```r
# use predict function to make predictions using fitted model
predict_regression_tree <- predict(fit_regression_tree, winequal_test_set)

# calculate RMSE using caret::RMSE
rmse_rt_allpreds <- RMSE(predict_regression_tree, winequal_test_set$quality)
# print RMSE
rmse_rt_allpreds
```

```
## [1] 0.7598277
```

Tuning and cross validation To avoid overfitting the model and to optimise the parameters we can improve the regression tree algorithm performance with caret::train. This will allow us to use cross validation to tune the model. The most common parameters to tune for a regression tree are *cp*, which is the complexity parameter, and *maxdepth*, which is the maximum depth of any node of the final tree.(Hui Lin 2023) The train function allows training of both these parameters, with the method *rpart* to tune *cp* and the method *rpart2* to tune *maxdepth*.(Hui Lin 2023)

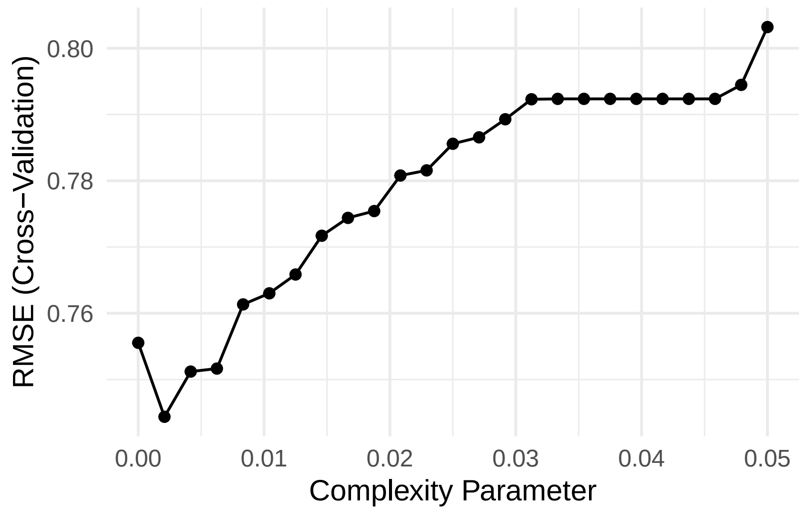
First we will tune *cp* with *rpart*. As the cross validation is a random process, we will also set a seed for reproducibility. I've chosen to cross validate with 10 samples and tune *cp* with a sequence 25 values between 0 and 0.05. Here, the RMSE is 0.752454.

```r
# Fit the regression tree model on the training set
# With tuning  the cp parameter and cross validation

#set seed for reproducibility
set.seed(1, sample.kind = "Rounding")

# use caret:train to tune cp
train_rpart_cv10 <- train(
  quality ~., # all 11 predictors
  data = winequal_train_set, # train set
  method = "rpart", # using rpart for cp
  trControl = trainControl("cv", number = 10), # cross validation of 10 samples
  tuneGrid = data.frame(cp = seq(0, 0.05, len = 25))) # sequence of 25 cp to test

# plot the tuning of RMSE vs cp
ggplot(train_rpart_cv10) +
   theme_minimal()
```



```r
# print the value of best cp with lowest RMSE
train_rpart_cv10$bestTune
```

```
##            cp
## 2 0.002083333
```

```r
# use tuned model to predict quality against test set
predict_rpart_cv10 <- predict(train_rpart_cv10, winequal_test_set)

# calculate RMSE
rmse_rpart_cv10 <- RMSE(predict_rpart_cv10, winequal_test_set$quality)
# print RMSE
rmse_rpart_cv10
```

```
## [1] 0.7524548
```

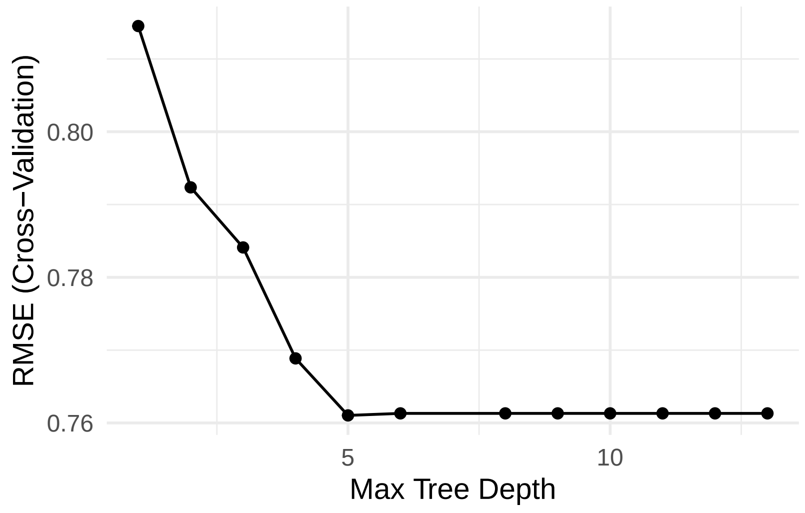
Next we will tune *maxdepth* using *rpart2* using similar methods as for *cp*. Here, the RMSE is 0.759828.

```r
# Fit the regression tree model on the training set
# With tuning the maxdepth parameter and cross validation

#set seed for reproducibility
set.seed(1, sample.kind = "Rounding")

# use caret:train to tune maxdepth
train_rpart2_cv10 <- train(
  quality ~., # all 11 predictors
  data = winequal_train_set, # train set
  method = "rpart2", # rpart2 for tuning maxdepth
  tuneLength = 12, # 12 values
  trControl = trainControl("cv", number = 10)) # cross validation of 10 samples

# plot the tuning of RMSE vs maxdepth
ggplot(train_rpart2_cv10) +
  theme_minimal()
```



```r
# print the best value of maxdepth with lowest RMSE
train_rpart2_cv10$bestTune
```

```
##   maxdepth
## 5        5
```

```r
# use tuned model to predict quality against test set
predict_rpart2_cv10 <- predict(train_rpart2_cv10, winequal_test_set)

# calculate RMSE
rmse_rpart2_cv10 <- RMSE(predict_rpart2_cv10, winequal_test_set$quality)
# print RMSE
rmse_rpart2_cv10
```

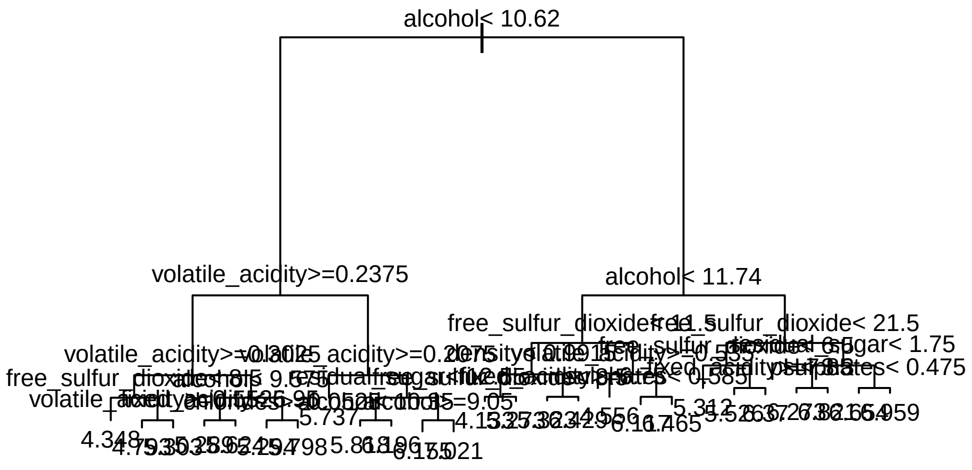
```
## [1] 0.7598277
```

Then we put both tuned parameters into the rpart function to fit the tuned regression tree model. With this final regression tree model the RMSE is the lowest so far at 0.751895.

```
# fit the regression tree model against the train set using
# the tuned cp and maxdepth parameters from above

fit_rpart_tuned <- rpart(quality ~ .,
                         data = winequal_train_set,
                         maxdepth = train_rpart2_cv10$bestTune$maxdepth,
                         cp = train_rpart_cv10$bestTune$cp)

# plot the tuned regression tree model
plot(fit_rpart_tuned, margin = 0.1)
text(fit_rpart_tuned, cex = 0.75)
```



```
# use tuned fitted model to predict quality against test set
predict_rpart_tuned <- predict(fit_rpart_tuned, winequal_test_set)

# calculate RMSE
rmse_rpart_tuned <- RMSE(predict_rpart_tuned, winequal_test_set$quality)
# print RMSE
rmse_rpart_tuned
```

```
## [1] 0.7518948
```

Random forests

The last machine learning method we will investigate for our wine quality score predictive model is Random forests.(Irizarry 2019b) These random forest algorithms are a popular machine learning approach as the techniques they employ address the downsides of regression trees.(Irizarry 2019b) Random forest methods average multiple regression trees to make a forest of trees with randomness, thereby improving prediction performance.

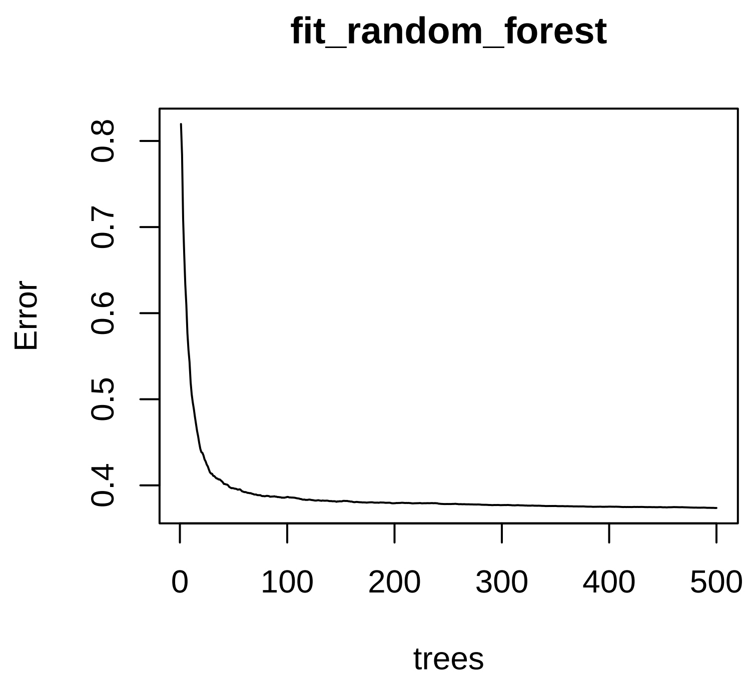
No tuning or cross validation First we will use the randomForest function from the randomForest package, to provide a baseline model for this algorithm. Here, we plot the fitted model which shows that the error has minimised around 400 trees. We then use the fitted model to predict the quality score against the test set. The RMSE for this model of 0.598826 is the lowest for far. Therefore, even without tuning the parameter of the model the random forest algorithm has the best predictive performance.

```r
# initial random forest model with no tuning

# set the seed for reproducibility
set.seed(1, sample.kind = "Rounding")

# using randomForest function to fit the model against train data
fit_random_forest <- randomForest(quality ~., data = winequal_train_set)

# plot the fitted model
plot(fit_random_forest)
```



```r
# use fitted model to predict quality against test set
predict_random_forest <- predict(fit_random_forest, winequal_test_set)
```

```r
# calculate RMSE
rmse_random_forest <- RMSE(predict_random_forest, winequal_test_set$quality)
# print RMSE
rmse_random_forest
```

```
## [1] 0.5988259
```

Now we will look to see if tuning the parameters of the random forest algorithm will improve performance. For this we will once again use the caret package.(Irizarry 2019a) There are two parameters that are most likely to have the biggest affect on the performance of our random forest model (Brownlee 2020), which are:

1) *mtry* - the number of variables randomly sampled as candidates at each split, and
2) *ntree* - the numbers of trees to grow

Only *mtry* can be tuned with train::caret, and the value of *ntree* is largely limited by computational time.(Brownlee 2020). Therefore, I set up a series of *mtry* tuning experiments with different values for *ntree* and 5 and 10-fold cross validation to test how long the algorithm would take to run (table 5). Then I fit a random forest model using the randomForest function with each tuned *mtry*, used the predict function against the test set, and calculated the RMSE. This way I could decide on the best trade off for computational cost to run the algorithm compared to model performance.

Table 5: Random forest tuning experiments testing computational time vs algorithm performance

iteration	cross_validation	mtry	ntree	computational_time	best_mtry	RMSE
1	10-fold	seq(1, 10, 1)	500	22 mins	4	0.597867
2	5-fold	seq(1, 6, 1)	300	3.5 mins	2	0.598555
3	5-fold	seq(1, 6, 1)	400	4.5 mins	3	0.598502
4	5-fold	seq(1, 6, 1)	500	5.5 mins	5	0.601563

The results of these experiments are depicted in table 5. Here, you can see that the random forest model with the lowest RMSE (0.597867) and therefore best performance was iteration 1, using *ntree* of 500, and 10-fold cross validation to get *mtry* of 4. However, tuning *mtry* in iteration 1 took 22 minutes, and is not practical to include in this report. The model with the next best performance according to the RMSE measure used *mtry* from iteration 3, which took 4.5 minutes to tune. Therefore I chose this to be the final model and have included the code for this random forest tuning and model fitting below (note that this code will take about 5-6 minutes to run, including tuning *mtry* and fitting the model).

```r
# random forest algorithm tuning mtry, 4.5 mins to tune mtry, 1 min to fit model

# set the seed for reproducibility
set.seed(1, sample.kind = "Rounding")

# use 5 fold cross validation
control <- trainControl(method="cv", number = 5)
# assign the values of mtry to tune for
grid <- data.frame(mtry = seq(1, 6, 1))
# caret::train
train_rf_4 <-  train(quality ~., # predict quality
                     method = "rf", # using random forest algorithm
                     data = winequal_train_set, # on the train set
```

```
                      ntree = 400, # number of trees
                      trControl = control, # cross validation
                      tuneGrid = grid) # tune for mtry

# print the tuning of the model, RMSE, Rsquared and MAE
train_rf_4
```

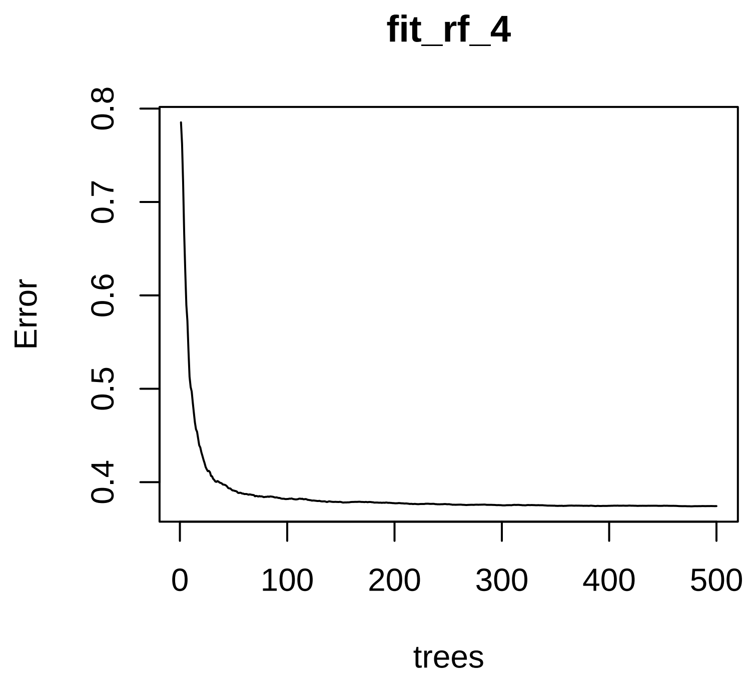
```
## Random Forest
##
## 3918 samples
##   11 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 3134, 3134, 3134, 3134, 3136
## Resampling results across tuning parameters:
##
##   mtry  RMSE       Rsquared   MAE
##   1     0.6334290  0.5105399  0.4714813
##   2     0.6269315  0.5107539  0.4612476
##   3     0.6265567  0.5079660  0.4581663
##   4     0.6271270  0.5055403  0.4579729
##   5     0.6259604  0.5066483  0.4570453
##   6     0.6282462  0.5023221  0.4582308
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 5.
```

```
# print best tuned mtry that minimises RMSE
train_rf_4$bestTune$mtry
```

```
## [1] 5
```

```
# fit the randomForest model with best tuned mtry
fit_rf_4 <- randomForest(quality ~., data = winequal_train_set,
                         mtry = train_rf_4$bestTune$mtry)

# plot the best tuned fitted model
plot(fit_rf_4)
```



```r
# use tuned fitted model to predict quality against test set
predict_fit_rf_4 <- predict(fit_rf_4, winequal_test_set)

# calculate RMSE
rmse_rf_4 <- RMSE(predict_fit_rf_4, winequal_test_set$quality)
# print RMSE
rmse_rf_4
```

```
## [1] 0.5985022
```

Results

We have developed a random forest algorithm for the final wine quality score predictive model. The RMSEs are shown in table 6 for each the best iteration of each machine learning method investigated. Here, you can see that the Random forest algorithm clearly outperforms the other two machine learning methods, as it has the lowest RMSE.

Table 6: Performance of machine learning algorithms for wine quality predictive model as measured by RMSE

iteration	machine_learning_algorithm	parameters	RMSE
1	Generalised linear regression	all predictors	0.7585215
2	Regression trees	cp = 0.002, maxdepth = 5	0.7518948
3	Random forests	mtry = 3, ntree = 400	0.5985022

Now that we have the final model, we can apply this to the whole of the white wine quality dataset, *winequal_white*.

```r
#Final model is the random forest as it has the lowest RMSE

#Therefore applying the model to the dataset:
fit_final_model_rf <- randomForest(quality ~., data = winequal_white,
                                   mtry = train_rf_4$bestTune$mtry)


# adding predicted quality score to the whole white wine dataset
final_model_rf <- winequal_white %>%
  mutate(predict_quality = predict(fit_final_model_rf, winequal_white))
```

A plot of the actual quality vs predicted quality score for each wine in the *winequal_white* dataset is shown in figure 7. Here we can see that the range of predicted quality scores is smaller than the actual range of quality. Since we performed regression modelling on the dataset, we got back continuous predictive scores, even though the actual scores were integers. We can see that the predictive scores have a spread of about 1. Therefore, the random forest final model has provided a reasonable predictive model of wine quality scores.

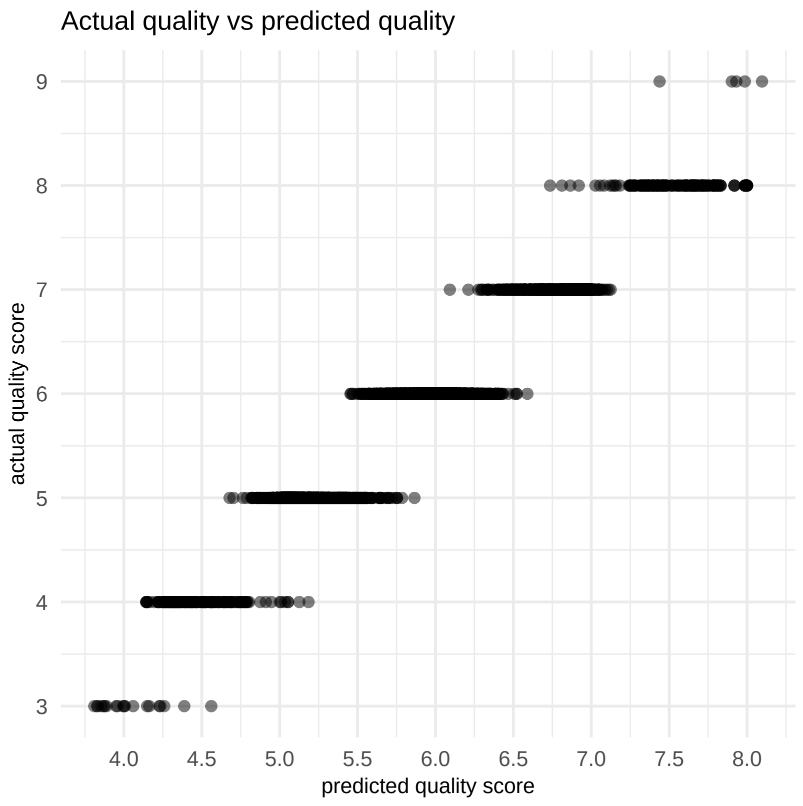


Figure 7: Actual quality scores vs predicted quality scores for each wine in the dataset

Conclusion

I have explored machine learning methods to develop a wine quality score predictive model using the white wine quality datset. I found that the random forest algorithm provided the best performance using RMSE as the comparative metric. Due to scope creep, I limited this project to only looking at three algorithms.

However, if I had more time and resources available I would continue this project by looking at additional algorithms, such as Bayesian methods(Irizarry 2019b) or support vector regression(Tibrewal 2023).

References

Brownlee, Jason. 2020. "Tune Machine Learning Algorithms in r (Random Forest Case Study)." Machine Learning Mastery. https://machinelearningmastery.com/tune-machine-learning-algorithms-in-r/.

Cortez, Cerdeira, P. 2009a. "Modeling Wine Preferences by Data Mining from Physicochemical Properties." *Decision Support Systems* 47 (4): 547–53.

———. 2009b. "Wine Quality [Dataset]." UCI machine learning repository. https://archive.ics.uci.edu/dataset/186/wine+quality.

Curada, John Paul. 2023. "Red Wine Quality: Simple and Clean Practice Dataset for Regression or Classification Modelling." Kaggle. https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009/discussion/456651.

Hui Lin, Ming Li. 2023. "Practitioner's Guide to Data Science." In. https://scientistcafe.com/ids/regression-and-decision-tree-basic#regression-and-decision-tree-basic.

Irizarry, Rafael A. 2019a. "Introduction to Data Science: Data Analysis and Prediction Algorithms with r." In, 551–56. https://leanpub.com/datasciencebook. https://rafalab.dfci.harvard.edu/dsbook/caret.html#caret.

———. 2019b. "Introduction to Data Science: Data Analysis and Prediction Algorithms with r." In, 557–600. https://leanpub.com/datasciencebook. https://rafalab.dfci.harvard.edu/dsbook/examples-of-algorithms.html#examples-of-algorithms.

Jonatasv. 2024. "Metrics Evaluation: MSE, RMSE, MAE and MAPE." Medium. https://medium.com/@jonatasv/metrics-evaluation-mse-rmse-mae-and-mape-317cab85a26b.

Prasad, Ashwin. 2021. "Regression Trees | Decision Tree for Regression | Machine Learning." Medium. https://medium.com/analytics-vidhya/regression-trees-decision-tree-for-regression-machine-learning-e4d7525d8047.

Tibrewal, Tasmay Pankaj. 2023. "Support Vector Machines (SVM): An Intuitive Explanation." Medium. https://medium.com/low-code-for-advanced-data-science/support-vector-machines-svm-an-intuitive-explanation-b084d6238106.