Forecasting the price of Italian Red Wine

Viktoriia Ilina

July 2021

Contents

1 Overview

Life is too short to drink bad wine - Johann Wolfgang von Goethe

Tuscany, the Promised Land for every wine devotee, and an ultimate destination of any path seeking best winy experience. A mesmerizing landscape full of tiny hilltop villages wrapped around by winding roads, monumental cypress trees and ambrosial vineyards. Indeed, a land made of folk tales and fables. And the spice, passion, scent, and vitality of wine is the essence that nourishes its soul with the mysterious power of Tuscan art of winemaking. Zeffiro Ciuffoletti, a renown historian, once came up with a perfect definition of the evolvement of Tuscan wine, saying: "Tuscany, in regards to wines, has no equal in the whole world, thanks to generous nature, and to a civilization of grapevine and of wine that has been decanted and refined over centuries[1]." Italy's most fruitful winery area, Tuscany is rightly glorious by its master reference, law-backed wines including 41 DOC (Denomination of Controlled Origin) and 11 DOCG (Denomination of Controlled and Guaranteed Origin), a category, that only Italy's best wines get rewarded with. Besides, there are further six of more flexible designations IGP/IGT, with the pan-regional Toscana IGP[2]. But, how does one find out the way to not get lost among all this diversity? The reading of wine label can be undoubtedly confusing and overwhelming for the layman. American oak, French oak, Hungarian oak, stainless steel, concrete, new, neutral, 15% of one and 85% of another…. What the heck does this all mean? And which factors determine the price of wine?

The main goal of this project is trying to make a robust prediction model of the price of a bottle of wine based on the data indicated on its label (name, producer, variety of grape, classification, alcohol content, year and type of aging) . The data was extracted from xtraWine, one of the best wine-shop in Italy 2021 by Gambero Rosso[3]. The dataset includes information on 803 red wines produced by 243 firms in the Tuscany region with a volume of 0.75 liters. All prices are shown in euros (VAT included). This report will present an exploratory analysis of the data, methodology and training the models, results and discussion.

[1] https://timelessitalytravels.com/2017/06/26/an-intriguing-history-of-tuscan-wine/

[2] https://www.wine-searcher.com/regions-tuscany

[3] https://www.gamberorosso.it/ristoranti/scheda-enoteca/xtrawine/

2 Exploratory data analysis

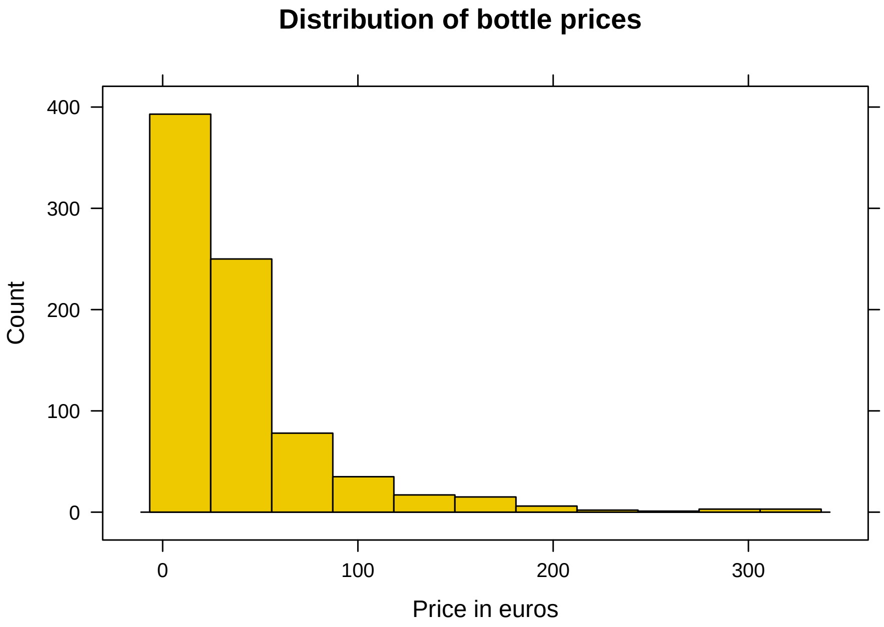
First of all, let's take a quick look at the data.

```
## # A tibble: 6 x 8
##   Name          Producer      Grape  Classification   Alcohol  Year Aging  Price
##   <chr>         <chr>         <chr>  <chr>            <chr>   <dbl> <chr>  <chr>
## 1 Fattoria dei~ Fattoria dei~ Sangi~ DOCG Brunello d~ 14.50    2015 Aged ~ 29.50
## 2 Campo Alle C~ Campo alle C~ Blend~ DOC Bolgheri Ro~ 14.00    2016 Aged ~ 16.90
## 3 Il Poggione ~ Il Poggione   Sangi~ DOCG Brunello d~ 14.50    2016 Aged ~ 38.80
## 4 Leonardo da ~ Cantine Leon~ Blend~ DOCG Chianti     13.00    2019 No oa~ 8.00
## 5 Guado al Tas~ Tenuta Guado~ Blend~ DOC Bolgheri Ro~ 14.50    2019 Aged ~ 18.90
## 6 Castello di ~ Castello di ~ Sangi~ DOCG Chianti Cl~ 13.50    2018 Aged ~ 15.49
```

For further analysis, delete irrelevant column "Name" and convert other columns' content to the proper format. Thereafter, check our dataset.

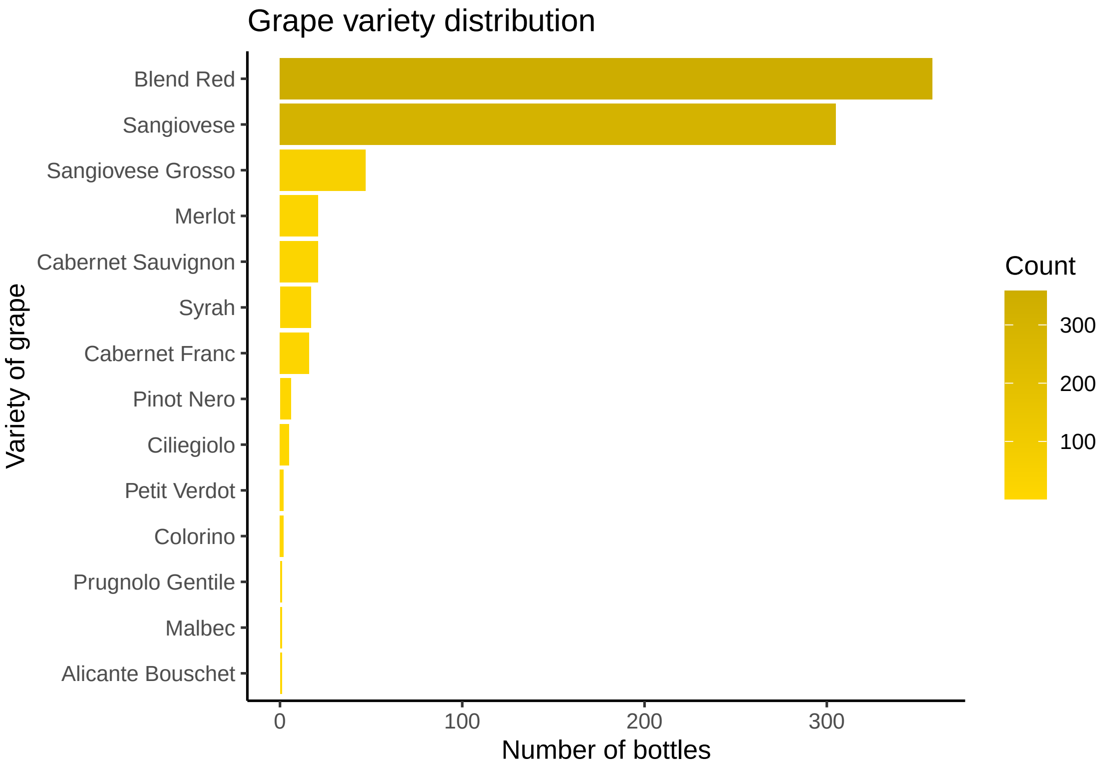
```
##                        Producer                  Grape
##  Banfi                    : 19   Blend Red         :358
##  Cantine Leonardo da Vinci: 19   Sangiovese        :305
##  Ricasoli 1141            : 10   Sangiovese Grosso : 47
##  Tenute Folonari          : 10   Cabernet Sauvignon: 21
##  Brancaia                 :  9   Merlot            : 21
##  Carpineto                :  9   Syrah             : 17
##  (Other)                  :727   (Other)           : 34
##                              Classification    Alcohol           Year
##  IGT Toscano o Toscana              :225    Min.   :11.00   2016   :191
##  DOCG Brunello di Montalcino        :120    1st Qu.:13.50   2018   :169
##  DOCG Chianti Classico              : 51    Median :14.00   2017   :134
##  DOCG Chianti Classico Riserva      : 45    Mean   :14.05   2015   :116
##  DOC Rosso di Montalcino            : 41    3rd Qu.:14.50   2019   :111
##  DOCG Brunello di Montalcino Riserva: 39    Max.   :16.00   2012   : 20
##  (Other)                            :282                    (Other): 62
##                    Aging         Price
##  Aged in barrique    :356   Min.   :  6.10
##  Aged in big casks   : 35   1st Qu.: 14.93
##  Aged in wood        :334   Median : 25.62
##  No oak/wood         : 75   Mean   : 40.57
##  With passage on wood:  3   3rd Qu.: 48.65
##                             Max.   :324.52
##
```

The distribution of bottle prices in the dataset is right-skewed, with most wines costing less than 50 euros, but with a long right tail of much more expensive wines (the highest price is 324.52 euro).



The Italian wine making sector is characterised by a high number of small or mid-sized players, mostly cooperatives or family owned companies[4], so it is not surprising that 58 producers have just one wine bottle, while the maximum count of the bottles belongs to two brends (Banfi Cantine Leonardo da Vinci) - 19 each.

87% of Tuscan wine is red with a staggering dominance of the local star Sangiovese (63% of the total vineyard surface area)[5]. So, it's quite natural that in the dataset most of introduced wines contain or are fully made of this variety.



[4] https://www.lexology.com/library/detail.aspx?g=dc3ec3b9-ef49-4f6a-9f56-b0ec873815aa

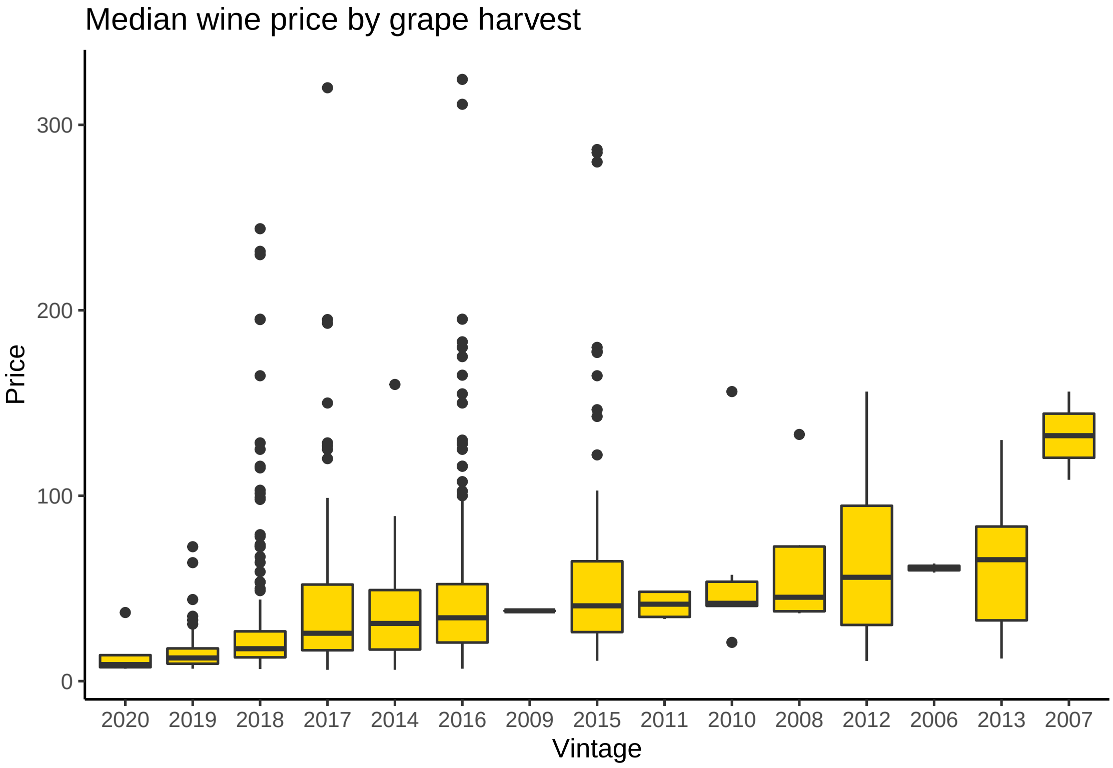
[5] https://www.vivino.com/wine-news/everything-you-need-to-know-about-tuscan-wine-in-600-words

69% of wines in the compilation have an alcohol content of at least 14% ABV, which proves the popular assumption of wine being a low alcohol drink, wrong. According to Liv-ex (the global marketplace for the wine trade), alcohol in wine is rising - red wines from Tuscany had higher alcohol levels on average in the decade between 2010 and 2019 than they did in 1990s[6]. This is facilitated by the climate changes and special techniques aimed to encourage ripening(planting at higher densities with low-yielding clones, short pruning, green harvesting and so on)[7].

Most of the wines in the selection have a vintage from 2015 to 2019.

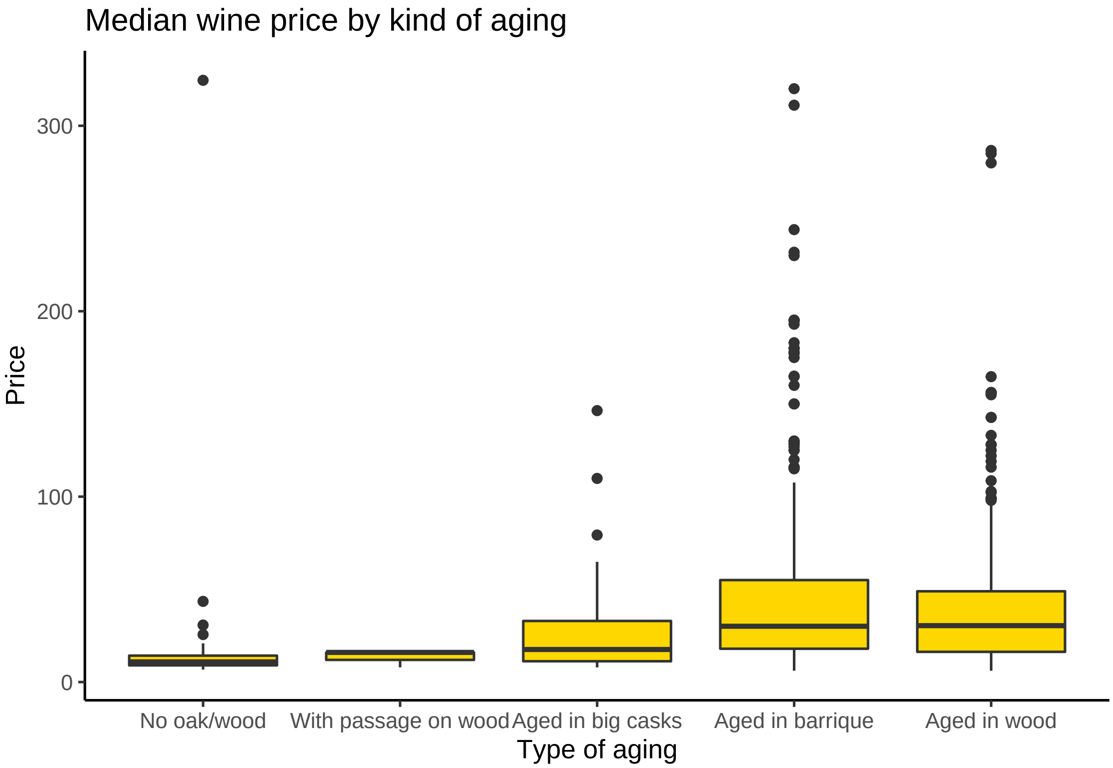
```
## # A tibble: 15 x 2
##    Year  Count
##    <fct> <int>
##  1 2016    191
##  2 2018    169
##  3 2017    134
##  4 2015    116
##  5 2019    111
##  6 2012     20
##  7 2013     18
##  8 2014     16
##  9 2020      9
## 10 2010      6
## 11 2008      4
## 12 2011      4
## 13 2006      2
## 14 2007      2
## 15 2009      1
```

As depicted in the chart below, overall, the younger wines are cheaper than their vintage kindred.



[6] https://www.decanter.com/learn/are-alcohol-levels-in-wine-rising-data-460879/
[7] https://www.winemag.com/2015/04/23/hot-italian-wines-is-15-abv-the-new-14/

Aging is the phase between alcoholic fermentation and bottling allowing the production of a wine with more rich and complex organoleptic qualities. Type of container directly affects the result of aging , along with oxygen, temperature, humidity, light and keeping practices, such as topping ups and rackings[8]. Below we can see an obvious interconnection between kind of aging and price of the bottle.



3 Methodology and analysis

Non-symmetric distribution of bottle prices with horde of outliers in the dataset may be the reason of bad performance of the models. So, It might make sense to categorize a continuous variable using several data independent cut-offs and reclassify the problem as the multi-class (multinomial) classification. Common machine learning algorithms that can be used for this task include: k-Nearest Neighbors, Decision Trees, Naïve Bayes, Random Forest, Gradient Boosting, Artificial neural networks and so on. Below we will use three of them.

Go through our transformed dataset:

```
##                             Producer      Grape              Classification
## 1                  Fattoria dei Barbi Sangiovese DOCG Brunello di Montalcino
## 2                   Campo alle Comete  Blend Red           DOC Bolgheri Rosso
## 3                         Il Poggione Sangiovese DOCG Brunello di Montalcino
## 4           Cantine Leonardo da Vinci  Blend Red                 DOCG Chianti
## 5 Tenuta Guado al Tasso (Antinori)  Blend Red           DOC Bolgheri Rosso
## 6 Castello di Fonterutoli (Mazzei) Sangiovese       DOCG Chianti Classico
##   Alcohol Year            Aging Category
## 1    14.5 2015 Aged in barrique    25-50
## 2      14 2016 Aged in barrique    10-25
## 3    14.5 2016 Aged in barrique    25-50
## 4      13 2019      No oak/wood      <10
## 5    14.5 2019 Aged in barrique    10-25
## 6    13.5 2018     Aged in wood    10-25
```

[8] http://www.diwinetaste.com/dwt/en2007066.php

Before proceeding to the models and algorithms, we should split our dataset into training (80%) and testing (20%) sets.

To get the base accuracy of the dataset, we will use a very popular supervised classification algorithm - Naive Bayes. This algorithm is based on the Bayes' theorem, that describes the probability of an event based on prior knowledge of the conditions that might be relevant to the event[9]. The Bayes' rule can be expressed in the following formula:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Where:

- $P(A|B)$ is the probability of event A occuring, given event B has occured;
- $P(B|A)$ is the probability of event B occuring, given event A has occured;
- $P(A)$ is the probability of event A;
- $P(B)$ is the probability of event B.

Train and test the model:

```
set.seed(120) # set the seed to make the prediction reproducible

# Fitting the model

model_naive <- naiveBayes(Category ~ ., data = train)

# Predicting on test data

prediction_naive <- predict(model_naive, newdata = test)

# Model evaluation

confusionMatrix(test$Category, prediction_naive)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <10 10-25 25-50 50-75 75-100 100-200 >200
##    <10      12     3     0     0      0       0    0
##    10-25     7    48     7     3      0       0    0
##    25-50     0    11    27     3      1       1    0
##    50-75     0     3     6     8      1       0    1
##    75-100    0     1     0     1      4       1    0
##    100-200   0     2     2     3      0       2    1
##    >200      0     1     0     1      0       0    0
##
## Overall Statistics
##
##                Accuracy : 0.6273
##                  95% CI : (0.5477, 0.7021)
##     No Information Rate : 0.4286
##     P-Value [Acc > NIR] : 3.048e-07
##
##                   Kappa : 0.4888
##
```

[9] https://corporatefinanceinstitute.com/resources/knowledge/other/bayes-theorem/

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: <10 Class: 10-25 Class: 25-50 Class: 50-75
## Sensitivity             0.63158       0.6957       0.6429      0.42105
## Specificity             0.97887       0.8152       0.8655      0.92254
## Pos Pred Value          0.80000       0.7385       0.6279      0.42105
## Neg Pred Value          0.95205       0.7812       0.8729      0.92254
## Prevalence              0.11801       0.4286       0.2609      0.11801
## Detection Rate          0.07453       0.2981       0.1677      0.04969
## Detection Prevalence    0.09317       0.4037       0.2671      0.11801
## Balanced Accuracy       0.80523       0.7554       0.7542      0.67179
##                      Class: 75-100 Class: 100-200 Class: >200
## Sensitivity                0.66667        0.50000     0.00000
## Specificity                0.98065        0.94904     0.98742
## Pos Pred Value             0.57143        0.20000     0.00000
## Neg Pred Value             0.98701        0.98675     0.98742
## Prevalence                 0.03727        0.02484     0.01242
## Detection Rate             0.02484        0.01242     0.00000
## Detection Prevalence       0.04348        0.06211     0.01242
## Balanced Accuracy          0.82366        0.72452     0.49371
```

The model achieved 62.73% accuracy with a p-value close to 0. It's better than a random choice, nevertheless very far from ideal.

Let's try to achieve more adequate predictions using an another model - Support Vector Machines(SVM). The objective of SVM is to find a hyperplane that maximizes the separation of the data points to their actual classes in an n-dimensional space. The data points which are at the minimum distance to the hyperplane i.e, closest points are called Support Vectors[10]. For fitting of this model we should choose two hypermeters: "C" for controlling error and "Gamma" for giving curvature weight of the decision boundary.

Here:

```
set.seed(2) # set the seed to make the prediction reproducible

# Training the model

model_svm <- svm(Category ~ ., data=train,
  method="C-classification", kernal="radial",
  gamma=0.1, cost=3)

# Predicting on test data

prediction_svm <- predict(model_svm, test)

# Model evaluation

confusionMatrix(test$Category, prediction_svm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction <10 10-25 25-50 50-75 75-100 100-200 >200
##    <10      10     5     0     0      0       0    0
```

[10] https://www.analyticsvidhya.com/blog/2021/05/multiclass-classification-using-svm/

```
##    10-25     1    60     4     0      0       0    0
##    25-50     1    13    28     1      0       0    0
##    50-75     0     5     6     7      0       1    0
##    75-100    0     1     1     1      2       2    0
##    100-200   0     1     6     2      0       1    0
##    >200      0     1     1     0      0       0    0
##
## Overall Statistics
##
##                Accuracy : 0.6708
##                  95% CI : (0.5925, 0.7427)
##     No Information Rate : 0.5342
##     P-Value [Acc > NIR] : 0.0002953
##
##                   Kappa : 0.5236
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: <10 Class: 10-25 Class: 25-50 Class: 50-75
## Sensitivity             0.83333       0.6977       0.6087      0.63636
## Specificity             0.96644       0.9333       0.8696      0.92000
## Pos Pred Value          0.66667       0.9231       0.6512      0.36842
## Neg Pred Value          0.98630       0.7292       0.8475      0.97183
## Prevalence              0.07453       0.5342       0.2857      0.06832
## Detection Rate          0.06211       0.3727       0.1739      0.04348
## Detection Prevalence    0.09317       0.4037       0.2671      0.11801
## Balanced Accuracy       0.89989       0.8155       0.7391      0.77818
##                      Class: 75-100 Class: 100-200 Class: >200
## Sensitivity                1.00000       0.250000          NA
## Specificity                0.96855       0.942675     0.98758
## Pos Pred Value             0.28571       0.100000          NA
## Neg Pred Value             1.00000       0.980132          NA
## Prevalence                 0.01242       0.024845     0.00000
## Detection Rate             0.01242       0.006211     0.00000
## Detection Prevalence       0.04348       0.062112     0.01242
## Balanced Accuracy          0.98428       0.596338          NA
```

Well, forecast accuracy has increased by 5%. Unfortunately. it's still a different result from what had been expected.

The last model that we'll try to implement is eXtreme Gradient Boosting(XGB). XGB is an efficient open-source application of the stochastic gradient boosting ensemble algorithm[11]. This algorithm found widespread use due to memory efficiency, time saving and model perfomance.

```
# XGBoost requires the classes to be in an integer format, starting with 0.
# So, we have to convert "Category" factor to the proper format

Categories <- my_data$Category
label <- as.integer(my_data$Category)-1
my_data$Category <- NULL

#Encoding all other variables to dummy variables

my_data <- one_hot(as.data.table(my_data))
```

[11] https://machinelearningmastery.com/extreme-gradient-boosting-ensemble-in-python/

```r
# Splitting the data for training and testing

set.seed(2) # set the seed to make the partion reproducible

n <- nrow(my_data)
index <- sample(n, floor(0.8*n))
train <- as.matrix(my_data[index,])
train_label <- label[index]
test <- as.matrix(my_data[-index,])
test_label <- label[-index]

# Creating the xgb.DMatrix objects

xgb.train <- xgb.DMatrix(data = train,label = train_label)
xgb.test <- xgb.DMatrix(data = test,label = test_label)

# Define the parameters for multinomial classification

num_class <- length(levels(Categories))
xgb_params <- list(
  objective="multi:softprob",
  eval_metric="mlogloss",
  num_class=num_class
)

set.seed(120) # set the seed to make the prediction reproducible

# Training the XGBoost model

xgb.fit <- xgb.train(
  params = xgb_params,
  data = xgb.train,
  nrounds = 100,
  early_stopping_rounds = 10,
  watchlist = list(val1 = xgb.train,val2 = xgb.test),
  verbose = 0
)

# Predicting on test data

xgb.pred <- predict(xgb.fit, test, reshape=T)
xgb.pred <- as.data.frame(xgb.pred)
colnames(xgb.pred) <- levels(Categories)

xgb.pred$prediction <- apply(xgb.pred, 1, function(x) colnames(xgb.pred)[which.max(x)])
xgb.pred$label <- levels(Categories)[test_label+1]

# Model evaluation

confusionMatrix(factor(xgb.pred$prediction),
                factor(xgb.pred$label),
                mode = "everything")
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction <10 >200 10-25 100-200 25-50 50-75 75-100
##    <10       6    0     0       0     1     0      0
##    >200      0    0     0       0     0     0      0
##    10-25     9    1    58       6    15     8      2
##    100-200   0    0     0       1     2     0      1
##    25-50     0    1     4       5    20     9      1
##    50-75     0    0     0       2     2     4      1
##    75-100    0    0     1       0     0     0      1
##
## Overall Statistics
##
##                Accuracy : 0.559
##                  95% CI : (0.4787, 0.6371)
##     No Information Rate : 0.3913
##     P-Value [Acc > NIR] : 1.231e-05
##
##                   Kappa : 0.355
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: <10 Class: >200 Class: 10-25 Class: 100-200
## Sensitivity             0.40000     0.00000       0.9206       0.071429
## Specificity             0.99315     1.00000       0.5816       0.979592
## Pos Pred Value          0.85714         NaN       0.5859       0.250000
## Neg Pred Value          0.94156     0.98758       0.9194       0.917197
## Precision               0.85714          NA       0.5859       0.250000
## Recall                  0.40000     0.00000       0.9206       0.071429
## F1                      0.54545          NA       0.7160       0.111111
## Prevalence              0.09317     0.01242       0.3913       0.086957
## Detection Rate          0.03727     0.00000       0.3602       0.006211
## Detection Prevalence    0.04348     0.00000       0.6149       0.024845
## Balanced Accuracy       0.69658     0.50000       0.7511       0.525510
##                      Class: 25-50 Class: 50-75 Class: 75-100
## Sensitivity                0.5000      0.19048      0.166667
## Specificity                0.8347      0.96429      0.993548
## Pos Pred Value             0.5000      0.44444      0.500000
## Neg Pred Value             0.8347      0.88816      0.968553
## Precision                  0.5000      0.44444      0.500000
## Recall                     0.5000      0.19048      0.166667
## F1                         0.5000      0.26667      0.250000
## Prevalence                 0.2484      0.13043      0.037267
## Detection Rate             0.1242      0.02484      0.006211
## Detection Prevalence       0.2484      0.05590      0.012422
## Balanced Accuracy          0.6674      0.57738      0.580108
```

Surprisingly, this algorithm performed the worst prediction power.

4 Results and discussion

The main goal of this project was trying to make a robust prediction model of the price of a bottle of wine based on the data indicated on its label. Unfortunately, we must acknowledge that this attempt failed. The best percentage of correct predictions for the test data, driven by Support Vector Machines algorithm, only managed to hit the mark of 67%, which is a barely feasible result. Thus, label information is not reliable predictor of bottle price.

However, future work could likely improve predictive performance in the following ways:

- expanding the sample beyond one region to the whole country or a number of countries;
- trying to solve this problem as a regression task;
- exploring more factors such as awards, packaging, reviews of the sommeliers, number of produced bottles, philosophy(organic/no), ageing potential and so on;
- evaluating the performance of the model via Accuracy, Precision, Recall and F1 Score metrics;
- using other machine learning algorithms (e.g. KNN).