

Mathematics in Machine Learning

Wine Quality

Gabriele Cuni
277957

Index

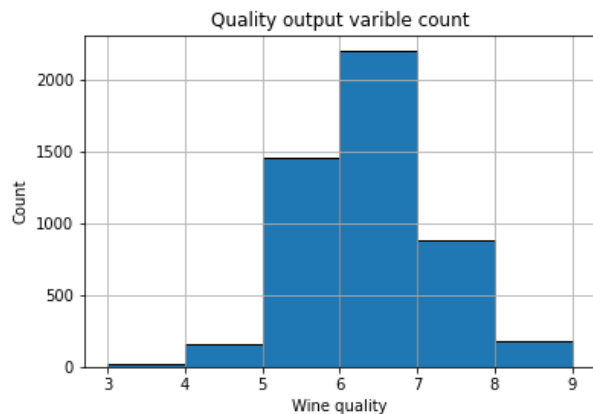
1. Data Exploration
 - 1.1. Dataset
 - 1.2. Target Variable Exploration
 - 1.3. Feature Exploration
 - 1.4. Standardization
 - 1.5. Correlation Matrix and Feature Selection
2. Model Validation
 - 2.1. Holdout Method
 - 2.2. K-Fold Method
3. Class Imbalance
 - 3.1. Random Oversampling
4. Model Selection
 - 4.1. Decision Tree
 - 4.2. Random Forest
 - 4.3. Adaptive Boosting
 - 4.4. Support Vector Machine
 - 4.5. K-Neighbors
5. Result
6. References

1.1 Dataset

The chosen wine dataset is populated with 4898 white wine samples, each one is characterized by 11 objective features and one output variable that is based on sensory data. The output is the median of at least 3 evaluations made by wine experts, each expert graded the wine quality between 0 (very bad) and 10 (very excellent).

The 11 objective features are: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol.

1.2 Target Variable Exploration

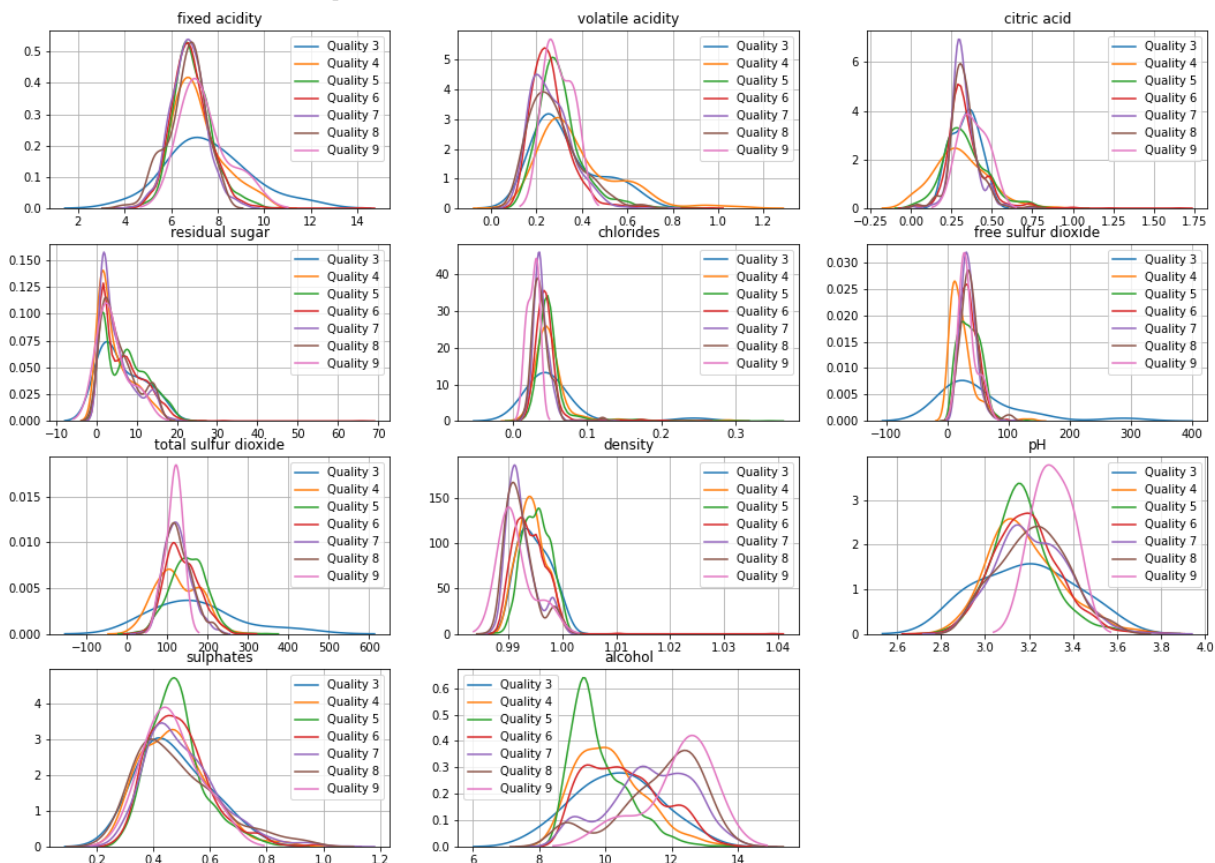


The output variable has a range between zero and ten, but in practice there are no wines of quality: 0,1,2 and 10.

Precisely there are 20 wines of quality 3, 163 wines of quality 4, 1457 wines of quality 5, 2198 wines of quality 6, 880 wines of quality 7, 175 wines of quality 8 and only 5 wines of quality 9.

As you can see the dataset is quite imbalanced, therefore the random oversampling technique will be used in order to lighten the imbalance effect on the classification models.

1.3 Feature Exploration



These are the distribution of each feature made by the kernel density estimate (KDE) plot which is a method for visualizing the distribution of observations in a dataset, analogous to a histogram. KDE represents the data using a continuous probability density curve in one dimension. From these graphs the feature Total Sulfur Dioxide

seems to be very good to infer the wine quality. This information will be taken into account when some features will be dropped in order to reduce correlation between features.

1.4 Standardization

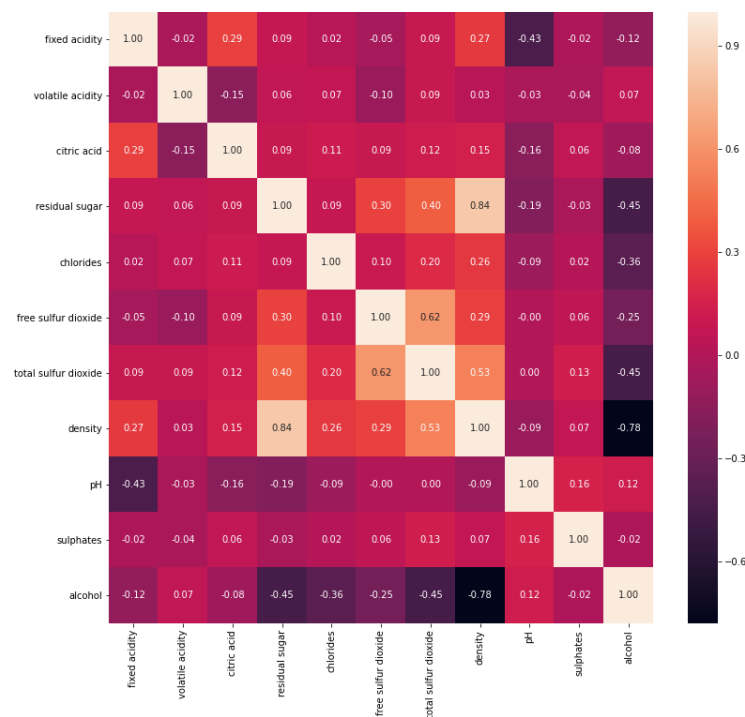
Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data. For instance many elements used in the objective function of a learning algorithm (such as the RBF kernel of Support Vector Machines) assume that all features are centered around 0 and have variance in the same order. If a feature has a variance that is orders of magnitude larger than others, it might dominate the objective function and make the estimator unable to learn from other features correctly as expected.

The standard score of a sample x is calculated as:

$$z = \frac{(x-u)}{s}$$

where u is the mean of the training samples, and s is the standard deviation of the training samples. Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set [13].

1.5 Correlation Matrix and Feature Selection



The above Correlation Matrix shows that given a threshold of ± 0.50 there are a lot of correlated features:

- total sulfur dioxide and free sulfur dioxide
- density and residual sugar
- density and total sulfur dioxide
- alcohol and density

Given the above list the features Density and Free Sulfur Dioxide are dropped from the dataset, in order to delete the high correlated pairs. Between Total and Free Sulfur Dioxide I chose to drop the Free one which is the less informative as it was seen in the feature exploration section.

2.1 Model Validation: Holdout Method

The simplest way to estimate the true error of a predictor h is by sampling an additional set of examples, independent of the training set, and using the empirical error on this validation set as our estimator. [8]

In this project I have split the available examples into three sets.

The first set is used for training the algorithms and the second is used as a validation set for model selection. After I select the best model, I test the performance of the output predictor on the third set, which is often called the test set. The number obtained is used as an estimator of the true error of the learned predictor and it is stored in the tables available in the result section.

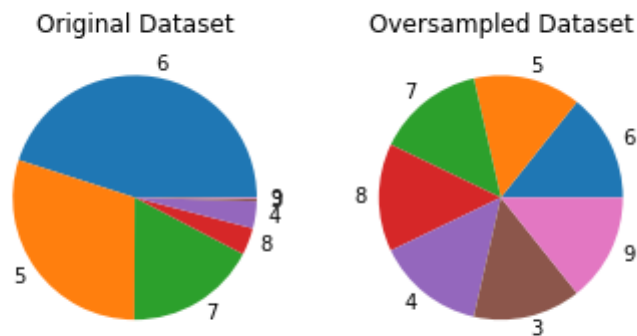
The train set is made of 3673 samples, the validation set is made of 918 samples and the test set is made of 307 samples.

2.2 Model Validation: K-Fold Method

Other than the simpler holdout method, the models are validated in parallel with the 5-Fold cross validation method in order to compare the different results in terms of the model selected.

In the k-fold cross validation method the original training set is partitioned into k subsets (5 subsets in this project). For each fold, the algorithm is trained on the union of the other folds and then the error of its output is estimated using the fold. Finally, the average of all these errors is the estimate of the true error. [8] The k-Fold cross validation method is used for model selection, and once the best parameters are chosen, the algorithm is retrained using these parameters on the entire training set and the accuracy result written in the result table is computed on the test set.

3.1 Class Imbalance: Random Oversampling



The first graph shows that the original dataset is quite imbalanced as mentioned also by the dataset summary delivered within the dataset itself. This imbalance can make it hard for some classifier to classify correctly the samples, especially the samples that are under represented in the dataset. Given this reasoning I have made a balanced dataset, as you can see in the second graph, by randomly sampling samples from the original dataset. Take into account that only the train set is oversampled, instead validation and test set are taken original without oversampling otherwise the model would have found identical samples in both the training set and the test set.

Experiments are made both on the original dataset and on the balanced dataset and results are compared.

4 Model Selection

The experiments in order to choose the model are done both on the original imbalanced dataset and on the balanced dataset, for each dataset there is a initial experiment where all the models (Decision Tree, Random Forest, AdaBoost, Support Vector Machine and K-Neighbor) are tested on the validation set with the default parameters given by Sklearn. After this preliminary study only the two best performing models for each dataset are fine tuned by a grid search in order to find the best performing hyper parameters both with the holdout and k-Fold Method.

From the preliminary study can be seen (In the result tables) that Decision Trees and Random Forest are always the best two performing models, therefore Random Forest is selected for the following fine tuning, instead the Decision Tree model is not chosen in order to fine tune a model outside of the family of model based on Trees.

4.1 Decision Tree

Decision Trees are a supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

The aim is learning a decision tree that on one hand fits the data well while on the other hand is not too large (This is the prior knowledge). Given this reasoning the max depth parameter of the Trees in the project has a maximum value equal to 20 in order to reduce the overfitting according to this formula [8]:

$$L_D(h) \leq L_S(h) + \sqrt{\frac{(n+1)\log_2(d+3) + \log(\frac{2}{\delta})}{2m}}$$

where n is the number of nodes of the tree, d is the number of features, m is the number of train samples and δ is the probability of getting a non-representative sample.

Trees can use different implementations of the Gain function which evaluate the gain of a split of the tree according to the i th features.

I have use the default function proposed by sklearn [10] which is the Gini Index [8]:

$C(a) = 2a(1 - a)$. Given the training error before splitting on feature i is $C(P_s[y = 1])$ and the training error after splitting on feature is:

$P_s[x_i = 1] \cdot C(P_s[y = 1|x_i = 1]) + P_s[x_i = 0] \cdot C(P_s[y = 1|x_i = 0])$ Therefore the gain to be the difference between the two.

Decision Tree performs well on the preliminary model selection but it is discarded in favor of Random Forest that performs better and it is also composed of trees.

4.2 Random Forest

Another way to reduce the danger of overfitting is by constructing an ensemble of trees. A random forest is a classifier consisting of a collection of decision trees, where each tree is constructed by applying an algorithm A on the training set S and an additional random vector, θ , where θ is sampled i.i.d. from some distribution. The prediction of the random forest is obtained by a majority vote over the predictions of the individual trees. Random forest is the model that best performs on the preliminary test with the default parameters both on the original and on the balanced dataset, therefore it is selected for further model fine tune.

The best parameters found on the original dataset with the holdout sets were: number of trees equal to 100 and depth equal to 17, with a test accuracy of 69.38%.

The best parameters found on the original dataset with the K-Fold method were: number of trees equal to 200 and depth equal to 10 , with a test accuracy of 62.45%.

The best parameters found on the balanced dataset with the holdout sets were: number of trees equal to 200 and depth equal to 19, with a test accuracy of 69.71% .

The best parameters found on the balanced dataset with the K-Fold method were the same as those found with the holdout method, therefore the accuracy is the same on the test set: 69.71%.

4.3 Adaptive Boosting

An Adaptive Boosting classifier is an assembled estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

The boosting approach addresses two major issues: the first is the bias-complexity tradeoff over which this learner has smooth control, the second issue that boosting addresses is the computational complexity of learning because it amplifies the accuracy of weak learners.

In this project the Adaptive Boosting assembled method is used with the default sklearn parameter in the preliminary study but it did not get good results both on the original and balanced dataset therefore it was not used in the subsequent fine tuning model validation. The model that the Adaptive Boosting method was trying to improve was the Decision Tree [12].

4.4 Support Vector Machine

The SVM algorithmic paradigm tackles the sample complexity challenge by searching for large margin separators, which is a certain type of prior knowledge. Roughly speaking, a hyper space separates a training set with a large margin if all the examples are not only on the correct side of the separating hyperplane but also far away from it.

The first theoretical implementation is the hard margin SVM in which data needs to be linearly separable to allow the algorithm to converge. The Hard-SVM rule is:

$$\operatorname{argmax}_{(\mathbf{w}, b): \|\mathbf{w}\|=1} \min_{i \in [m]} y_i (< \mathbf{w}, \mathbf{x}_i > + b)$$

The Hard-SVM formulation assumes that the training set is linearly separable, which is a rather strong assumption given that in the real world, the vast majority of the problems are not linearly separable. The Soft-SVM rule is:

$$\min_{\mathbf{w}, b, \xi} (\lambda \|\mathbf{w}\|^2 + \frac{1}{m} \sum_{i=1}^m \xi_i) \text{ such that } \forall i, y_i (< \mathbf{w}, \mathbf{x}_i > + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0$$

where the sklearn documentation [2] gives us $\lambda = \frac{1}{2}$ and $\frac{1}{m} = C$.

C is one of the hyper parameters evaluated during the model evaluation grid search. Smaller values of C allow more errors (increase the Approximation Error) in exchange for a bigger margin (less overfitting), while higher values decrease the Approximation Error with a higher risk of overfitting.

Notice that the sample complexity for both types of SVM does not depend on the dimension of the domain but rather on parameters such as the maximal norms of \mathbf{x} and \mathbf{w} .

The other hyper parameter evaluated during the model evaluation grid search is γ that is a Kernel parameter and must be greater than zero. This parameter is there because the experiment are done with a SVM classifier with a RBF kernel that stands for Radial Basis Function also called Gaussian Kernel [3] which is defined by the kernel function: $\exp(-\gamma ||x - x'||^2)$.

This kernel function $K(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$ implements the inner product in the new feature space therefore ψ doesn't have to be explicitly applied on the data. Thanks to this trick called kernel trick the computation of the learning task is feasible.

Support Vector Machine performed well in the preliminary test on the original dataset but not on the balanced dataset. Therefore it was fine tuned only on the original dataset both with the holdout method and the K-Fold method.

The parameters found with the holdout method were C equal to 10 and γ equal to 1, with a test accuracy of 64.17%.

The parameters found with the K-Fold method were C equal to 100 and γ equal to 0.01, with a test accuracy of 53.31%.

4.5 K-Neighbors

Nearest Neighbor algorithms are among the simplest of all machine learning algorithms. The idea is to memorize the training set and then to predict the label of any new instance on the basis of the labels of its closest neighbors in the training set.

Note that, in contrast with the algorithmic paradigms that are determined by some hypothesis class, the Nearest Neighbor method figures out a label on any test point without searching for a predictor within some predefined class of functions. Being so, it is often successful in classification situations where the decision boundary is very irregular.

In order to select the neighbor points a distance metric is used and by default sklearn

used the Minkowski metric [6] which is defined as [7]:
$$\left(\sum_{i=1}^n w \cdot |x_i - y_i|^p \right)^{\frac{1}{p}}$$

The w parameter is equal to one because the hyper parameter weights of the classifier is set on "uniform" by default. [6].

The p and k hyper parameters of the classifier are evaluated during the model evaluation grid search.

K-Neighbors performed well in the preliminary test on the balanced dataset but not on the original dataset. Therefore it was fine tuned only on the balanced dataset both with the holdout method and the K-Fold method.

The parameters found with the holdout method were: number of neighbors equal to 5 and P equal to 2, with a test accuracy of 55.05%.

The parameters found with the K-Fold method were: number of neighbors equal to 5 and P equal to 3, with a test accuracy of 57.01%.

5 Results

Original Dataset	Preliminary Test Validation Accuracy	Oversampled Dataset	Preliminary Test Validation Accuracy
Decision Tree	58.93 %	Decision Tree	59.69 %
Random Forest	67.76 %	Random Forest	66.99 %
AdaBoost	34.20 %	AdaBoost	34.64 %
SVC	54.03 %	SVC	43.25 %
KNeighbors	53.15 %	KNeighbors	52.40 %

Original Dataset	Hold Out Test Accuracy	K-Fold Test Accuracy	Oversampled Dataset	Hold Out Test Accuracy	K-Fold Test Accuracy
Random Forest	69.38 %	62.45 %	Random Forest	69.70 %	69.71 %
SVC	64.16 %	53.31 %	KNeighbors	55.05 %	57.01 %

The best model seems to be the Random Forest with the best parameters found on the balanced dataset with the K-Fold method. With the best parameters depth equal to 19 and number of the trees equal to 200, the accuracy on the test set was 69.71%.

This is a very good accuracy given that the output variable is not binary and the work on this dataset done in [1] reports an accuracy of 64.60% obtained thanks to a SVM model.

6 References

- [1] 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. ISSN: 0167-9236.
- [2] <https://scikit-learn.org/stable/modules/svm.html#svm-mathematical-formulation>
- [3] <https://scikit-learn.org/stable/modules/svm.html>
- [4] <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC>
- [5] <https://scikit-learn.org/stable/modules/neighbors.html>
- [6] <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>
- [7] <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.DistanceMetric.html>
- [8] Understanding Machine Learning: From Theory to Algorithms 2014 by Shai Shalev-Shwartz and Shai Ben-David
- [9] <https://scikit-learn.org/stable/modules/tree.html>
- [10] <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>
- [11] <https://scikit-learn.org/stable/modules/tree.html#tree-mathematical-formulation>
- [12] <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html>
- [13] <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>