Mathematics in Machine Learning
Wine Quality

Gabriele Cuni

Index

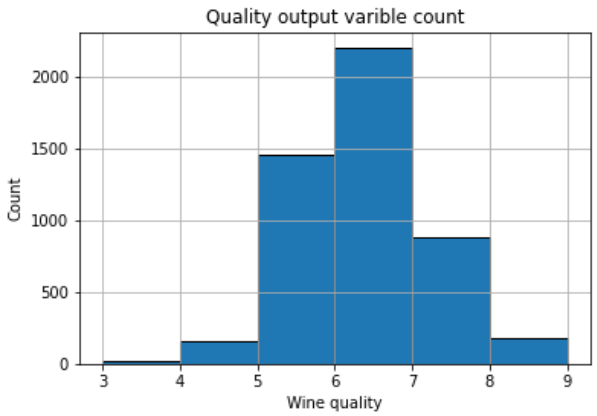
1.1 Dataset

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.

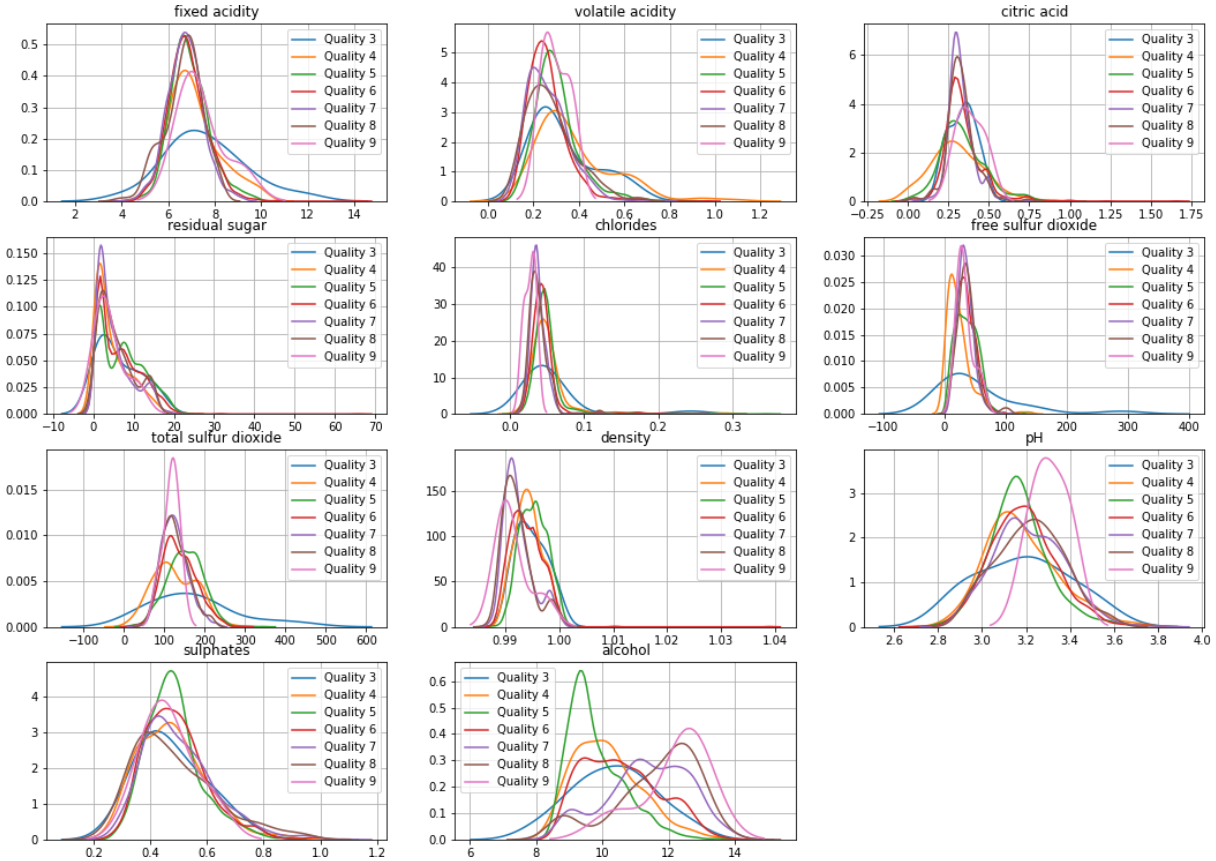
1.2 Target Variable Exploration



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 5.

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 or more dimensions.

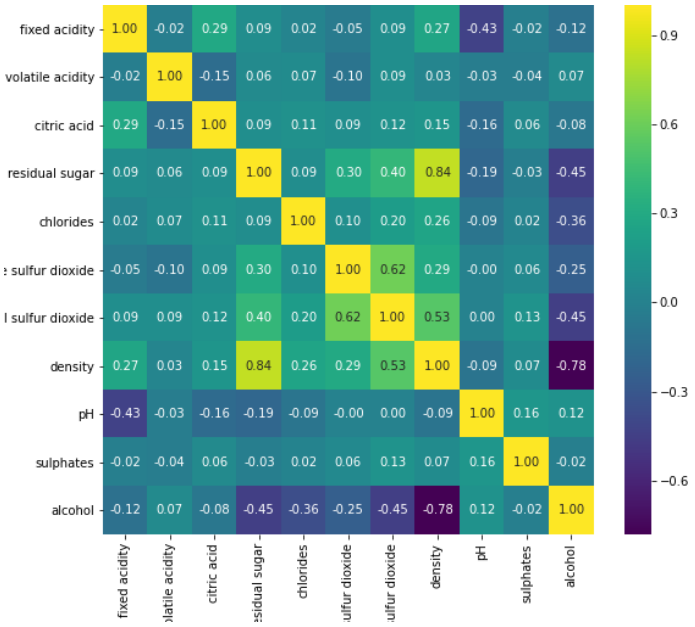
1.4 Standardization

Standardize features by removing the mean and scaling to unit variance. 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. 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.

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 reasoning the features Density and Free Sulfur Dioxide are dropped from the dataset, in order to delete the high correlated pairs.

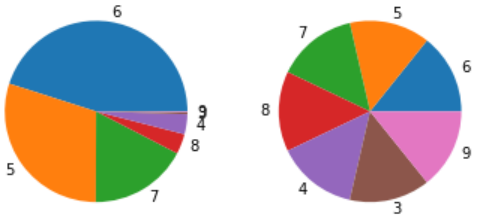
2.1 Model Validation: Holdout Method

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.
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.
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.

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 rappresentante 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.
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 chose 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, Gradient Boosting,

Support Vector Machine and K-Neighbor) are tested on the validation set with the default hyper 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.

4.1 Decision Tree

Decision Trees are a non-parametric 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.

A tree with k leaves can shatter a set of k instances and the VC dimension of decision trees with k leaves is k. Hence, if we allow decision trees of arbitrary size, we obtain a hypothesis class of infinite VC dimension. Such an approach can easily lead to overfitting. Given the above reasoning the max depth parameters in the project has a maximum equal to 20.

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

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.

4.3 Adaptive Boosting

An Adaptive Boosting classifier is a meta-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.

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

$min_{\mathbf{w},b,\xi}(\lambda||\mathbf{w}||^2 + \frac{1}{m}\sum_{i=1}^{m} \xi_i)$ such that $\forall i,\ y_i(< \mathbf{w}, \mathbf{x}_i > + b) \geq 1 - \xi_i$ 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 allows more errors in exchange of a bigger margin, while higher values can be used where it's needed to be less permissive regarding misclassification, 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 **x** and **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 $K(\mathbf{x}, \mathbf{x}') =< \psi(\mathbf{x}), \psi(\mathbf{x}') >$ 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.

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]: $(\sum_{i=1}^{n} w \cdot |x_i - y_i|^p)^{\frac{1}{p}}$

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.

5 Results

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#sklearn.metrics.DistanceMetric

[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