Wine Quality Prediction

A project on predicting wine quality based on wine dataset

MD. Fahim Bakhtiar
Department of Computer Science and Engineering
BRAC University
Dhaka
fahimbakhtiar089@gmail.com

***Abstract*— In this paper, I describe a model of a prediction engine that predicts wine quality based on various features of wine available on a wine dataset. This paper is divided into parts describing our dataset, technologies used, algorithms, various intermediate steps and the prediction model.**

***Keywords*—wine ;machine learnnig; regression;random forest; supervised machine learning**

I. Introduction

Wine is many things to many people. It's a passion, career, hobby, beverage, meal accompaniment, investment or simply a fun way to take away from the drudgeries of everyday life. For others it's a placeholder, allowing them to remember events and there are those to simply enjoy its pleasures.

Considering the appeal of wine to its consumers, maintaining a steady quality of it is of utmost importance to wine businesses. In this project I have attempted to produce a model to understand the quality of a wine from its chemical compounds & physical features inhibiting into it.

Here I have maintained a very simple technique and have attempted to create a satisfactory outlook with the basic methods that were taught in the class.

II. METHODOLOGY

A. Data Collection

The dataset for this project was acquired from UCI [1], which is a massive repository for finding various kinds of datasets for machine learning purposes.

In this dataset there are 1599 samples of wine data with 12 features including *fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, & alcohol.*

The target here is to predicts the quality metric of a wine given these as features. In the dataset, we use 80% for the training and 20% for testing purpose. It is quite rare to use 50/50, 80/20 is quite a commonly occurring ratio. A better practice is to use: 60% for training, 20% for cross validation, 20% for testing.

B. Technologies Used

In this project, I used Python 3.6.3 as my development language , and Anaconda Navigator as my Integrated Development Environment. Some built-in libraries used in the project include preprocessing, make_pipeline and GridSearchCV that have been used for different operations and functions.

For the actual data analysis, I used some popular libraries that are widely used for a wide array of Machine Learning projects. These libraries include numpy, pandas, and sklearn.

NumPy is a fundamental package for scientific computing using Python. It contains powerful features like N-dimensional array object, broadcasting functions, linear algebra, Fourier transform, random number, etc. [2].

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It can be used for tabular data, ordered and unordered time-series data, arbitrary matrix data, and any other forms of observational and statistical datasets.

Scikit-learn is a Machine Learning library built on top of SciPy and contains powerful tools for machine learning and statistical modeling such as classification, regression, and clustering algorithms including Support Vector Machines, Gradient Boosting, K-means, and of course, Random Forests—which I will be using in this project.

C. Algorithm

In this project I have used Random Forest as the base model. This is a very popular machine learning model used for both classification and regression problems. In this problem the process of finding the root node and splitting the feature node can run side by side at the same time. Given enough trees in the forest, the random forest algorithm does not overfit the model.

The random forest algorithm generally works by first finding the k features from a total of m features where k < m. among them the best split point is calculated [4]. Using the best split, the nodes are then further split into daughter nodes. Until X number of nodes are reached, these steps are repeated.[3]

D. *Data processing*

In the data processing phase, the collected data from the given url is read by the function provided in the pandas library. Most of the data is numerical in nature, which is a good sign for data processing purpose. At this point the data is split into test and training set, setting an arbitrary value for random state & stratifying the sample by target variable so that the training set looks similar to the test set.

E. *Standardization*

Standardization is the process of subtracting the means from each feature then dividing by the feature standard deviations. Standardization is a common requirement for machine learning tasks. Many algorithms assume that all features are centered around zero and have approximately the same variance.

In this case we won't be directly invoking the scale function, instead we'll be using a feature in Scikit-Learn called the Transformer API**.** The Transformer API allows us to "fit" a preprocessing step using the training data the same way you'd fit a model, and then use the same transformation on future data sets.

Here's what that process looks like: First, fit the transformer on the training set (saving the means and standard deviations). Then apply the transformer to the training set (scaling the training data) and lastly apply the transformer to the test set (using the same means and standard deviations).

This makes our final estimate of model performance more realistic, and it allows to insert the preprocessing steps into a cross-validation pipeline.

F. *Tuning Hyperparameters*

There are two types of parameters: model parameters and hyperparameters. Models parameters can be learned directly from the data (i.e. regression coefficients), while hyperparameters cannot. hyperparameters express "higher-level" structural information about the model, and they are typically set before training the model.

Within each decision tree, the computer can empirically decide where to create branches based on either mean-squared-error (MSE) or mean-absolute-error (MAE). Therefore, the actual branch locations are model parameters.

However, the algorithm does not know which of the two criteria, MSE or MAE, that it should use. The algorithm also cannot decide how many trees to include in the forest. These are examples of hyperparameters that we must set : 'randomforestregressor__criterion', 'randomforestregressor__max_depth', 'randomforestregressor__max_features', 'randomforestregressor__max_leaf_nodes'.

G. *Cross Validation Pipeline*

Cross-validation is a process for reliably estimating the performance of a method for building a model by training and evaluating your model multiple times using the same method. Practically, that "method" is simply a set of hyperparameters in this context.

These are the steps for CV:

1. Split the data into *k* equal parts, or "folds" (typically *k*=10).
2. Train the model on *k*-1 folds (e.g. the first 9 folds).
3. Evaluate it on the remaining "hold-out" fold (e.g. the 10th fold).
4. Perform steps (2) and (3) *k* times, each time holding out a different fold.
5. Aggregate the performance across all *k* folds. This is the performance metric.

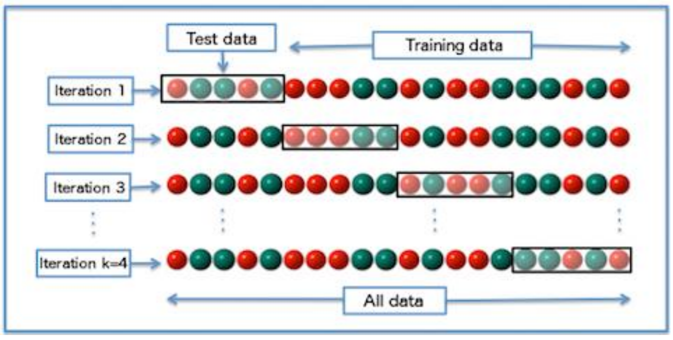

Fig. 1. K-Fold Cross-Validation diagram, courtesy of Wikipedia

Scikit-Learn makes it very simple to set this up, it provides the GridSearchCV function. GridSearchCV essentially performs cross-validation across the entire "grid" (all possible permutations) of hyperparameters.

III. RESULTS

A. *Prediction*

After running the predict function our model performance come to be around 0.45.

ACKNOWLEDGMENT

I would like to thank Dr. Iftekharul Mobin, Assistant Professor and Dr. Mohammad Zavid Parvez, Assistant Professor at the department of Computer Science and Engineering, BRAC University, for their continuous support, guidance and thoughtful feedback throughout the entire project. \

REFERENCES

[1] UCI Machine Learning Repository. Data available on mlr.cs.umass.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv.

[2] W. McKinney, “Python for data analysis: Data wrangling with Pandas, NumPy, and IPython." O'Reilly Media, Inc, 2012.

[3] Andreas C. Müller, Sarah Guido “Introduction to Machine Learning with Python A Guide for Data Scientists”.