Wine Data Analysis

BAN 620 – DATA MINING

PROFESSOR ROBERT TREICHLER

CALIFORNIA STATE UNIVERSITY EAST BAY

SPRING SEMESTER 2020

AMAN SOLANKI | BILAL ADNAN | MANDEEP SINH | KRISHNA PATEL

Aman Solanki, Bilal Adnan,
Mandeep Singh, Krishna Patel

Table of Contents

Aman Solanki, Bilal Adnan,
Mandeep Singh, Krishna Patel

Introduction

The Wine data used for the project contains approximately 12,000 commercially available wines and their chemical compositions. Along with the details about the composition of the wines the data also provides information on the number of cases when the wine was purchased by wine distribution companies after sampling a wine.

The purpose of this project is to explore and analyze the compositional properties of various wines and build a model to *predict the number of cases* of wine that will be purchased by the wine distribution companies after sampling a wine.

We aim to make good use of multi-variate techniques already in place to accomplish our goal. It would be in our best interest to conclude our project with a model that is delivers:

1. More Accuracy
2. Less Mean Squared Error
3. Computationally Inexpensive to deploy

Aman Solanki, Bilal Adnan,
Mandeep Singh, Krishna Patel

Dataset Description

Below is the short description of the dataset.

Total Datapoints	≈12000
Total attributes	16 (including index variable)
Input variables (Predictors)	14 (excluding index variable)
Outcome variable	1 (Target)

Description of the provided Wine Dataset

Column	Description
INDEX	wine Identification Variable
TARGET	number of Wine Cases ordered
FixedAcidity	most acids involved with wine or fixed or nonvolatile (do not evaporate readily)
VolatileAcidity	the amount of acetic acid in wine
CitricAcid	citric acid amount in wine
ResidualSugar	the amount of sugar remaining after fermentation stops
Chlorides	the amount of salt in the wine
FreeSulfurDioxide	the free form of SO2 exists in equilibrium between molecular SO2 (as a dissolved gas) and bisulfite ion
TotalSulfurDioxide	amount of free and bound forms of S02
Density	the density of water is close to that of water depending on the percent alcohol and sugar content
pH	describes how acidic or basic a wine is on a scale from 0 (very acidic) to 14 (very basic)
Sulphates	a wine additive which can contribute to sulfur dioxide gas (S02) levels
Alcohol	the percent alcohol content of the wine
LabelAppeal	score indicating the appeal of label design for consumers.
AcidIndex	proprietary method of testing total acidity of wine by using a weighted average
STARS	Wine Rating

Data Exploration

The dataset contains approximately 12000 entries. After using quick data exploration commands (*describe* and *info*), we realized that there were multiple missing values.

Feature Name	Missing Values
ResidualSugar	616
Chlorides	638
FreeSulfurDioxide	647
TotalSulfurDioxide	682
pH	395
Sulphates	1210
Alcohol	653
STARS	3359

Approximately half of the features in the entire dataset had 5%-10% missing values. The above missing values summary table clearly gives an idea that STARS has approximately 30% missing values. This makes sense once a product is purchased, every consumer generally does not participate in customer satisfaction survey.

Univariate Analysis: Detecting Outliers

Looking at the various plots we could not find any data points that could be considered as a definite outlier. We noticed that the data contained negative values. It was astonishing to see features in a Wine dataset with negative values. It was more surprising to see negative alcohol content for some entries in the dataset. The possible approaches to fix the negative values were:

1. Replace the negative values with *null* and then impute those values alongside other missing values.
2. Drop entries containing negative values.
3. Replace columns with negative values with absolute values.

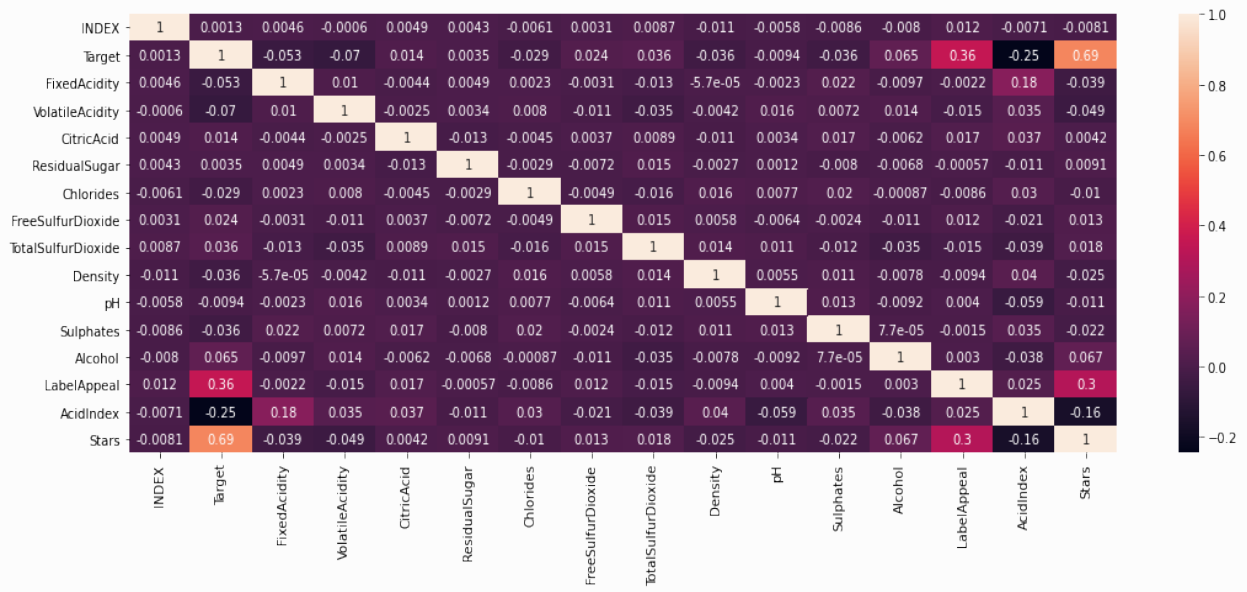
Feature Name	Negative Values
Fixed Acidity	1621
Volatile Acidity	2827
Citric Acid	2966
Residual Sugar	3136
Chlorides	3197
Free SulfurDioxide	3036
Total SulfurDioxide	2504
Sulphates	2361
Alcohol	118

We collectively chose to replace the column with negative values with their absolute values. We assumed that the negative sign before the number were added because of a human data entry error. An example of the code snippet we used to accomplish this milestone:

```
no_negative_df['FixedAcidity'] = abs(no_negative_df['FixedAcidity'])
```

Bivariate Analysis

Bivariate analysis aims to understand the relationship between two variables.



The above correlation heatmap was created after replacing the negative values with absolute values and after imputing the missing values with an *Iterative Imputer*. The heatmap hints a strong relationship between Target Variable with Stars Variable. This makes perfect sense, because in real world, an average product rating would directly be proportional to the sale of that item. For our dataset, wines with higher rating should technically have more case orders.

Data Preparation

Fix missing value:

Missing data can reduce the statistical power of a study and can produce biased estimates, leading to invalid conclusions. To impute missing value, we have used an *Iterative* ***Imputer*** which is a multivariate imputer that estimates each feature from all the others. Scikit Learn defines that Iterative Imputer uses a 'strategy for imputing missing values by modeling each feature with missing values as a function of other features in a round-robin fashion.'

We chose this because for a Wine dataset, it would be ideal to impute missing values by considering other chemicals present. For example, if a wine has less alcohol content, it would make perfect sense to conclude that it would also have less acid index.

Data Normalization

Our data attributes have values on different scales which may lead to poor data models while performing data mining operations.

Data summary before normalization.

	INDEX	TARGET	FixedAcidity	VolatileAcidity	CitricAcid	ResidualSugar	Chlorides	FreeSulfurDioxide	TotalSulfurDioxide	Density
count	12795.000000	12795.000000	12795.000000	12795.000000	12795.000000	12795.000000	12795.000000	12795.000000	12795.000000	12795.000000
mean	8069.980305	3.029074	8.063251	0.641086	0.686315	23.367862	0.222552	106.680346	204.338256	0.994203
std	4656.905107	1.926368	4.996119	0.555614	0.606005	24.337802	0.228375	105.334103	158.728015	0.026538
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.888090
25%	4037.500000	2.000000	5.600000	0.250000	0.280000	4.000000	0.046000	29.000000	102.000000	0.987720
50%	8110.000000	3.000000	7.000000	0.410000	0.440000	14.200000	0.120000	61.000000	160.000000	0.994490
75%	12106.500000	4.000000	9.800000	0.910000	0.970000	37.200000	0.353000	164.000000	251.000000	1.000515
max	16129.000000	8.000000	34.400000	3.680000	3.860000	141.150000	1.351000	623.000000	1057.000000	1.099240

Hence to bring all the attributes on the same scale we have used MinMaxScaler.

Data summary after normalization

	FixedAcidity	VolatileAcidity	CitricAcid	ResidualSugar	Chlorides	FreeSulfurDioxide	TotalSulfurDioxide	Density	pH	Sulphates
count	12795.000000	12795.000000	12795.000000	12795.000000	12795.000000	12795.000000	12795.000000	12795.000000	12795.000000	12795.000000
mean	0.234397	0.174208	0.177802	0.165553	0.164731	0.171237	0.193319	0.502547	0.482773	0.199703
std	0.145236	0.150982	0.156996	0.172425	0.169042	0.169076	0.150168	0.125681	0.118433	0.147141
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.162791	0.067935	0.072539	0.028339	0.034049	0.046549	0.096500	0.471845	0.440708	0.106132
50%	0.203488	0.111413	0.113990	0.100602	0.088823	0.097913	0.151372	0.503907	0.481416	0.150943
75%	0.284884	0.247283	0.251295	0.263549	0.261288	0.263242	0.237465	0.532441	0.525664	0.235849
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Data splitting

We have apportioned the data into training, validation, and test sets, with a 50-30-20 split.

```
Training:    (6397, 14)
Validation:  (3838, 14)
Test:        (2560, 14)
```

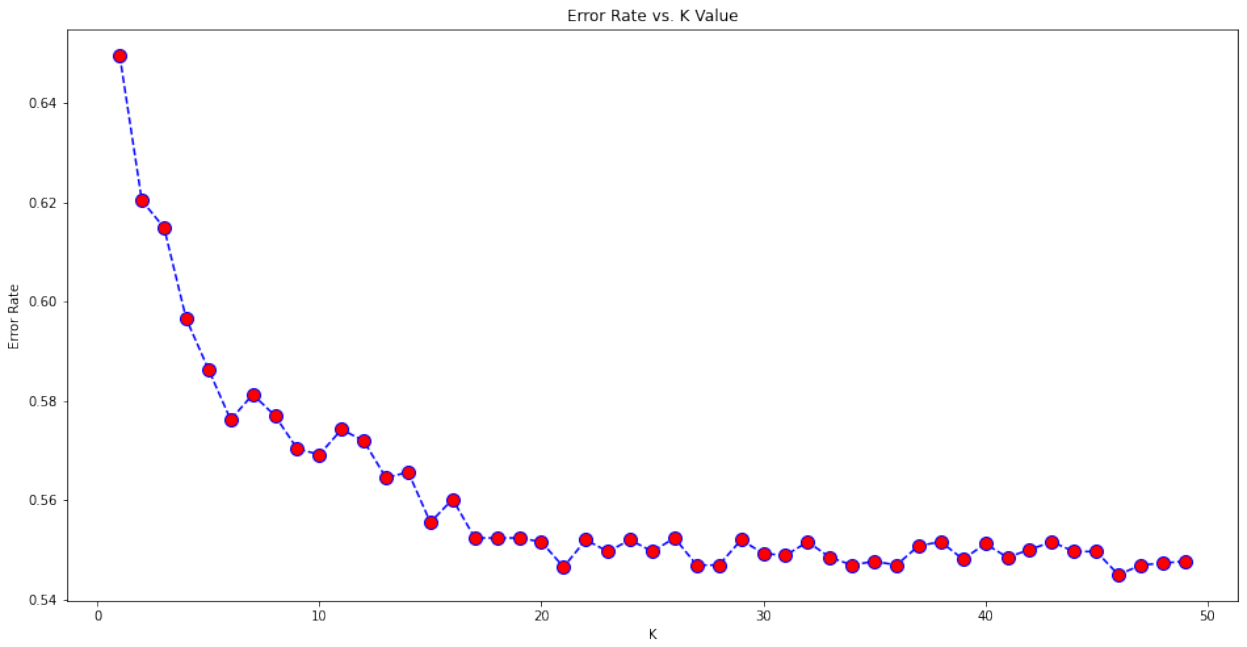
Build Models (Let's just do it)

After preprocessing the data, we identified the algorithms that are applicable to predict the target variable. Some of the elements affecting the choice of a model were:

- The accuracy of the model.
- The interpretability of the model.
- The complexity of the model.
- The scalability of the model.

KNN (KNeighborsClassifier)

The k-nearest neighbors (KNN) algorithm is a supervised machine learning algorithm that can be used to solve both classification and regression problems. It assumes that similar things exist in close proximity therefore similar things are near to each other. For KNN, we acquired the necessary hyperparameters by using an Error Rate vs K values line plot. Considering the results in the plot below we decided to tune the KNeighborsClassifier with n_neighbors=46 because it had the least error rate.



Support Vector Machine (SVC)

SVM is a supervised machine learning algorithm that can be used for classification or regression problems. It chooses the decision boundary which is the most distant from the points nearest to the decision boundary from both classes which is called the kernel trick to transform data and then based on these transformations it finds an optimal boundary between the possible outputs. For Support Vector Machine Model, the hyperparameter were acquired using GridSearchCV. The kernel gave the following improved parameters:

```
Improved parameters:  {'C': 10.0, 'gamma': 1.0}
```

The accuracy of the model went up by 1% after using the improved parameters to train the model.

LassoCV

Lasso is a regression analysis method that performs both variable selection and regularization to enhance the prediction accuracy and interpretability of the statistical model it produces.
For Lasso Regression Model, default hyperparameters were used. (normalize=True, cv=5)

Ridge

Ridge regression is a way to create a parsimonious model when the number of predictor variables in a set exceeds the number of observations, or when a data set has multicollinearity (correlations between predictor variables). For Ridge Regression Model, default hyperparameters were used. (normalize=True, alpha=1.0)

Random Forest Classifier

Random forests consist of a large number of individual decision trees that operate as an ensemble. Each tree in the random forest gives out a class prediction and the class with the most votes becomes our model's prediction. For Random Forest Classifier Model, the hyperparameters were acquired using AdaBoostClassifier and GridSearchCV.

Decision Tree Classifier

Decision Tree algorithm is a supervised machine learning algorithm. It can be used for solving regression and classification problems as well. It creates a training model that can be used to predict the class or value of the target variable by learning simple decision rules inferred from the training data. For Decision Tree Classifier Model, the hyperparameter were acquired using AdaBoostClassifier and GridSearchCV.

MLP Classifier

A multilayer perceptron (MLP) is a class of feedforward artificial neural networks (ANN). It utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can separate data that is not linearly separable. For MLP Classifier Model, the hyperparameter were acquired using GridSearchCV.

Gradient Boosting Classifier

Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models to create a strong predictive model. It can be combined with other models to improve accuracy. For Gradient Boosting Classifier Model, we tried to acquire hyperparameters using AdaBoostClassifier and GridSearchCV, but it took several hours to run. We trained our set using no additional hyperparameters.

Extra Tree Classifier

An extra-trees classifier. This class implements a meta estimator that fits several randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. For Extra Tree Classifier Model, the hyperparameter were acquired using AdaBoostClassifier and GridSearchCV.

Results

Models and their corresponding MSE scores and Accuracy Scores (sorted by MSE):

ID	Model Name	MSE Score	Accuracy Score
8	Gradient Boosting	1.18	59.14%
5	Random Forest	1.25	58.79%
6	Decision Tree	1.5	55.55%
7	MLPClassifier	1.58	55.70%
3	Lasso	1.75	-
4	Ridge	2.13	-
9	Extra Tree Classifier	2.15	55.51%
2	SVC	2.34	48.40%
1	KNN	2.5	45.51%

Actual data and predicted data using different models.

Actual	KNN	SVC	Lasso	Ridge	Random Forest	Decision Tree	MLPClassifier	Gradient Boosting	Extra Tree
4.0	3.0	4.0	3.262933	3.120608	4.0	4.0	3.0	3.0	4.0
5.0	4.0	5.0	3.619470	3.460845	5.0	5.0	5.0	4.0	4.0
3.0	0.0	0.0	1.722692	2.285066	3.0	3.0	4.0	4.0	0.0
6.0	5.0	5.0	4.742942	3.976265	5.0	5.0	5.0	5.0	5.0
0.0	0.0	0.0	1.231434	1.917329	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...
6.0	6.0	6.0	5.371140	4.424551	6.0	6.0	6.0	6.0	6.0
0.0	3.0	4.0	2.367835	2.554405	4.0	4.0	3.0	4.0	4.0
0.0	0.0	0.0	2.508976	2.902660	0.0	0.0	0.0	0.0	0.0
4.0	5.0	5.0	6.208523	4.804378	5.0	5.0	5.0	6.0	5.0
5.0	4.0	4.0	3.367932	3.184718	4.0	4.0	4.0	4.0	4.0

Conclusion

Several models were used to predict our target variable but we found that Gradient Boosting had the least Mean Squared Error amongst all our models. Most of the models were trained using custom hyperparameters (Adaboost and GridSeachCV) which would result to more accurate predictions. Despite of not being trained using any improved hyperparameter, our winning model had the least Mean Squared Error and the highest Accuracy Score amongst all the other models. Our second contender was Random Forest model. It was trained using improved hyperparameters.

Even though Gradient Boosting model had the highest accuracy, we would recommended our Random Forest model because we believe that model was not computationally expensive to train and had nominal difference in the Mean Squared Error and Accuracy Score.