Carthage University
*** * ***
National Institute of Applied Science and Technology

Machine Learning Report

Study Field : Software Engineering

Grade : 5^{th} Grade

Subject :

Predicting Wine Types

Done by : **Mayssa Jaziri - Aycha Abid - Essia Ben Hamida**

Academic Year : 2022/2023

Contents

List of Figures

List of Tables

Acronyms

KNN K-Nearest Neighbors. 3, 5

SVM Support Vector Machine. 4

Introduction

This project is a walkthrough of wine classification using different machine learning models for supervised learning. It presents as well the evaluation and comparison of these different models.

Chapter 1

Project overview

Introduction

In this section we present wine classification context and the different models we are going to use to predict wine types.

1.1 Wine Classification Context

Wine certification is commonly assessed with the aid of physicochemical and sensory checks. Physicochemical laboratory examines automatically the different ingredients of the wine whilst sensory tests depend in particular on human experts. It has to be pressured that flavor is the least understood of the human senses hence wine classification is a hard task. Moreover, the relationship among the physicochemical and sensory evaluations is complicated and nonetheless now no longer completely understood.

1.2 Supervised Machine Learning Models used

The dataset we are working on is a combination of two datasets : White Wine Dataset and Red Wine Dataset. Hence, the problem we are dealing with is a supervised learning problem since we have the labels/classes of our data from the start. There are many machine learning classifiers we can use to predict the wine type. In our case we are applying and comparing the results between : Logistic Regression, K-Nearest Neighbors, Support Vector Machine (Linear Classifier), Support Vector Machine (RBF Classifier), Decision Tree and Random Forest.

1.2.1 Logistic Regression

Logistic regression is a powerful supervised machine learning algorithm used for binary classification problems . It is considered as linear regression used for classification purposes. Logistic regression essentially uses a logistic function defined below to model a binary output variable.

The main key difference between linear regression and logistic regression is that logistic regression's range is bounded between 0 and 1. In addition, as opposed to linear regression, logistic regression does not require a linear relationship between inputs and output variables.

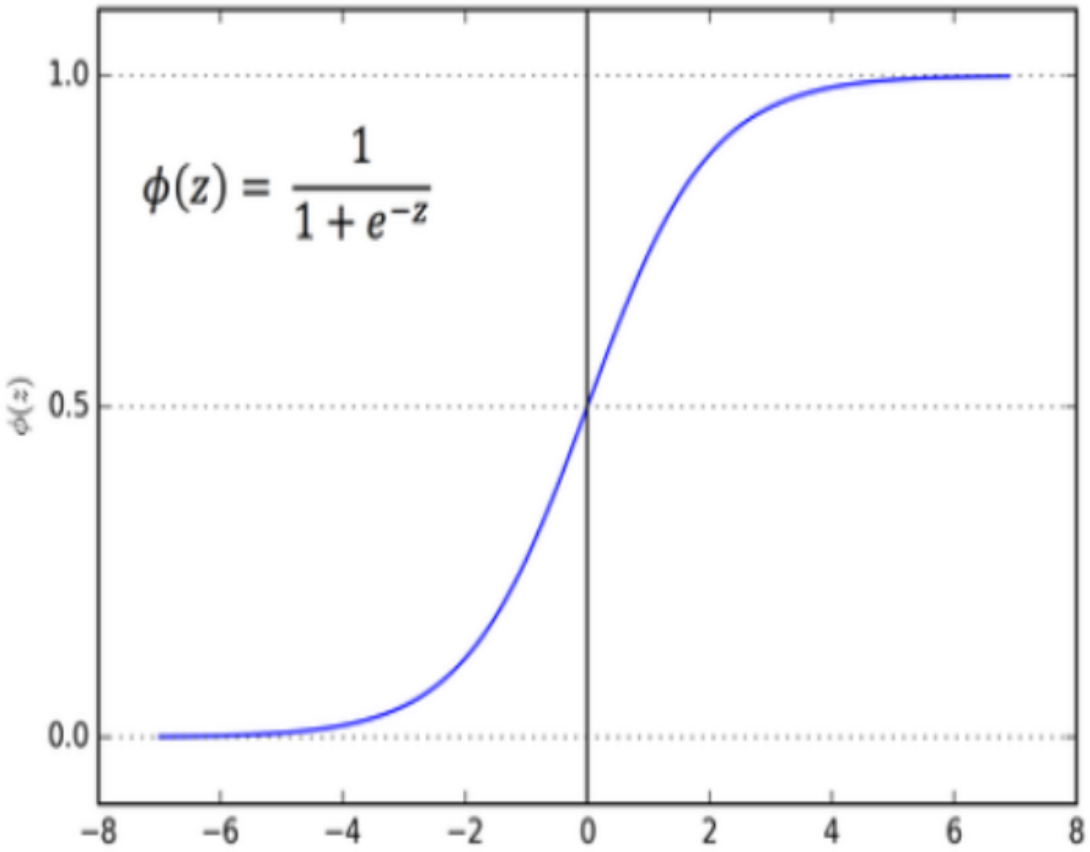

Figure 1.1: Logistic regression function

1.2.2 K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a non-parametric, supervised learning classifier for classification and regression purposes. It relies on distance/proximity and the assumption that similar points can be found near one another.

It is mostly used in classification problems and majority vote is the key method for labelling in this case. An example is shown in the next figure.

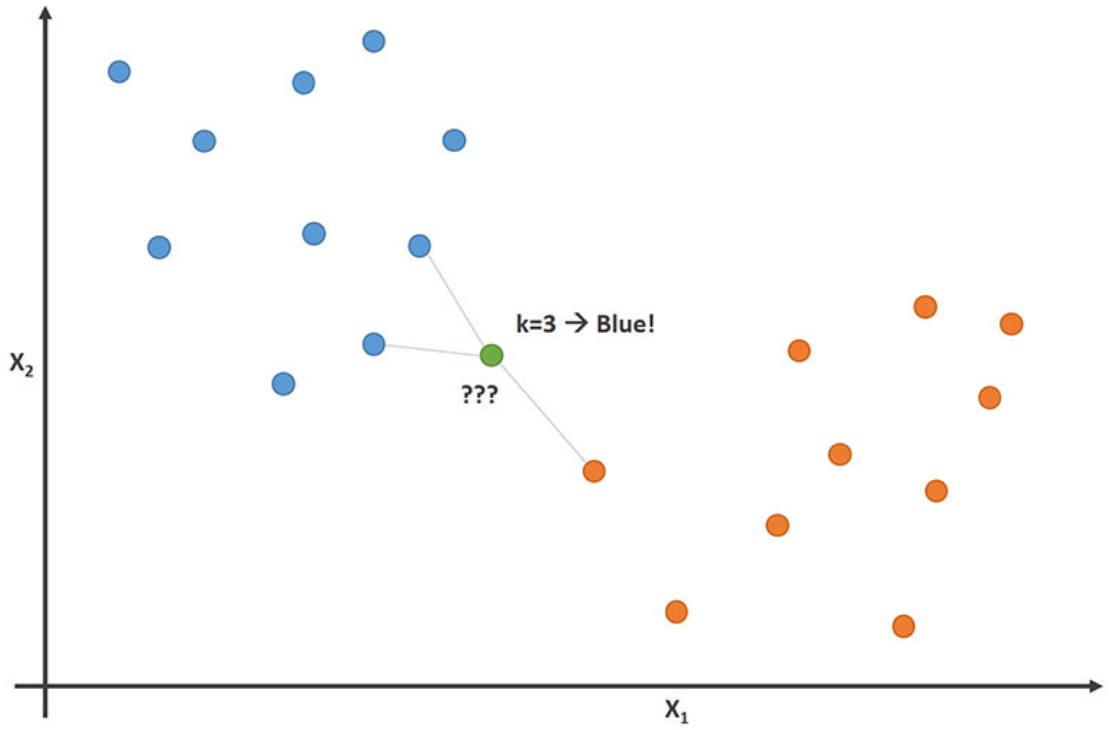

Figure 1.2: KNN classifier

1.2.3 Support Vector Machine (Linear Classifier)

Support Vector Machine (SVM) Linear Classifier is a linear model that is used for both classification and regression purposes. It can solve linear problems with the help of the creation of a line or a hyperplane seperating the data into different classes.

Support Vector Machine (SVM) aims to maximize the margin that come between classes.

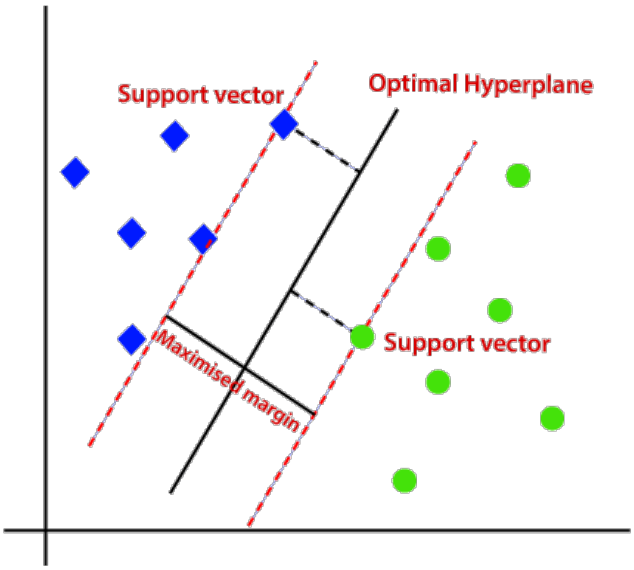

Figure 1.3: Linear SVM

1.2.4 Support Vector Machine (RBF Classifier)

RBF Kernel is popular since it is similar to K-Nearest Neighbors Algorithm. It has the advantages of KNN and overcomes the space complexity problem.

The RBF kernel function for two points X and Y computes the similarity or how close they are to each other. This kernel can be mathematically represented with the following formula:

$$K(X,Y) = exp(\frac{-\|\mathbf{X}-\mathbf{Y}\|^2}{2\theta^2})$$

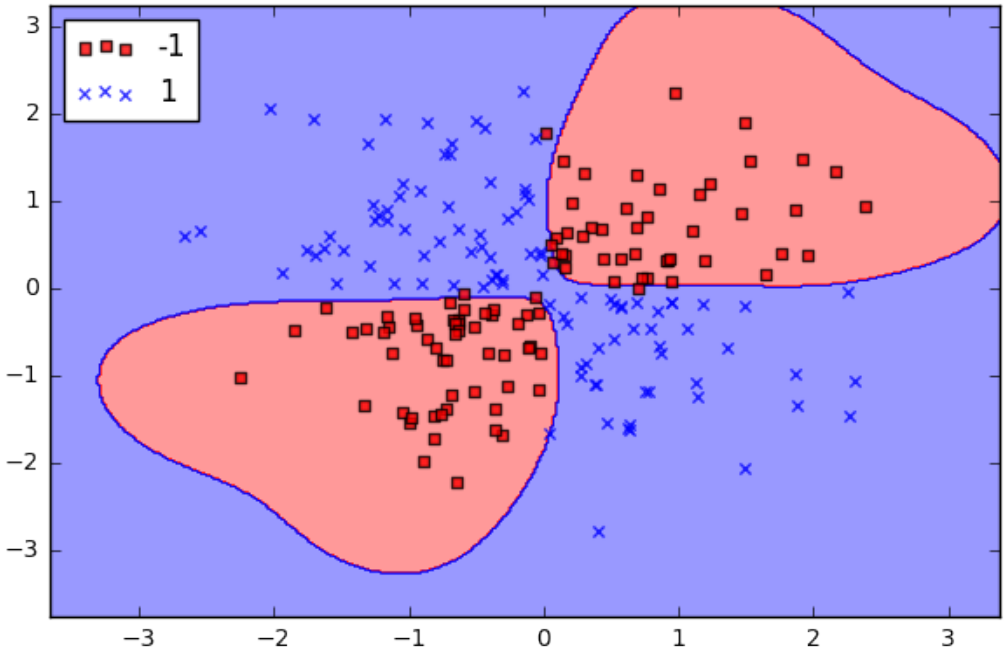

Figure 1.4: SVM with RBF

1.2.5 Decision Tree

A decision tree is a non-parametric supervised machine learning classifier that is utilized for both classification and regression problems. It has a hierarchical, tree structure, which consists of a root, branches, internal nodes and leaf nodes .

The internal nodes represent the features and each one has https://meet.google.com/xuk-erbt-sgrits own values (branches). As for the leaf branches, they represent the result. The idea is simple, decision tree aims to put the features that allows us to gain more information at first (at a high node) then moves to the next ones with less information.

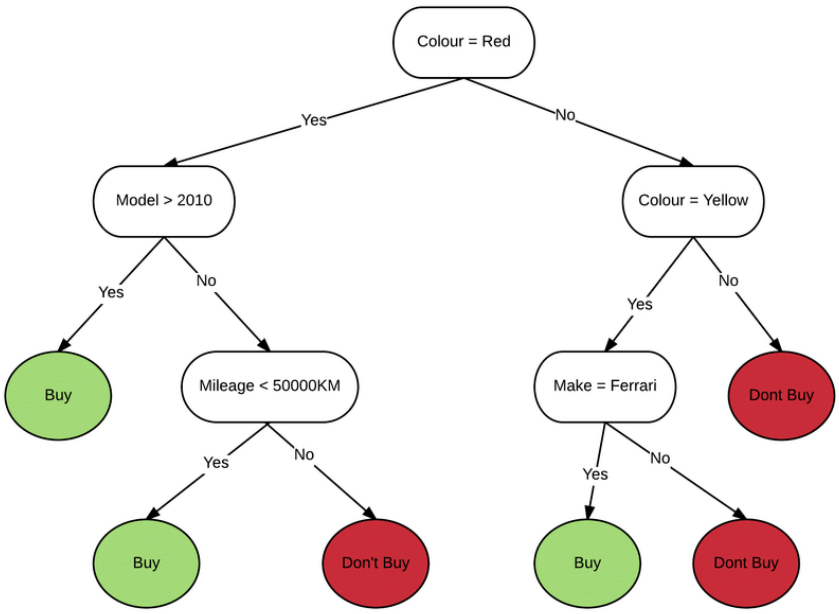

Figure 1.5: Decision Tree

1.2.6 Random Forest

Random forest, like its name implies, is composed of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest generates a class prediction and the class with the most votes becomes our model's prediction.

Random Forest model falls under the category of ensemble learning methods and its strength is based on the low correlation between the decision trees it is composed of.

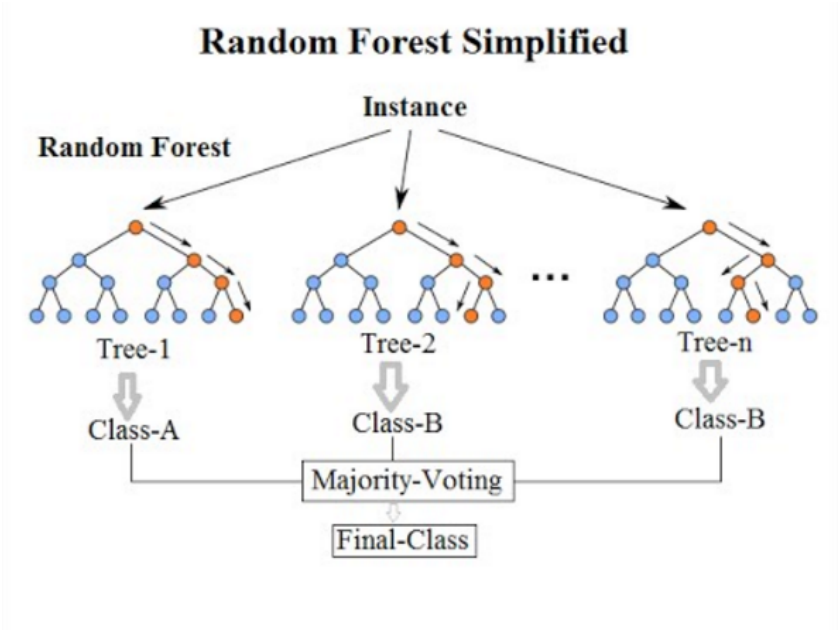

Figure 1.6: Random Forest

Chapter 2

Project walkthrough

Introduction

This section is a small walkthrough of the project consisting of Loading Data, Visualizing and understanding it, Pre-processing it, Training our Models and evaluating them at last.

2.1 Loading Data

As mentioned before, our dataset consists of the combination of two datasets: Red Wine Dataset and White Wine Dataset. Both of these datasets are available on the UCI Machine Learning Repository. The first step is to load the red wine and white wine datasets then we combine them together using Pandas and we add the color feature to the new dataset.

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	color
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5	red
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5	red
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5	red
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6	red
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5	red
...	...	...	...	...	...	...	...	...	...	...	...	...	...
6492	6.2	0.21	0.29	1.6	0.039	24.0	92.0	0.99114	3.27	0.50	11.2	6	white
6493	6.6	0.32	0.36	8.0	0.047	57.0	168.0	0.99490	3.15	0.46	9.6	5	white
6494	6.5	0.24	0.19	1.2	0.041	30.0	111.0	0.99254	2.99	0.46	9.4	6	white
6495	5.5	0.29	0.30	1.1	0.022	20.0	110.0	0.98869	3.34	0.38	12.8	7	white
6496	6.0	0.21	0.38	0.8	0.020	22.0	98.0	0.98941	3.26	0.32	11.8	6	white

6497 rows × 13 columns

Figure 2.1: Wine Dataset

The dataset is composed of 6497 instances, their 12 features and labels refering to the color's class we are trying to predict. The 12 features are:

- fixed acidity
- volatile acidity

- citric acid
- residual sugar
- chlorides
- free sulfur dioxide
- total sulfur dioxide
- density
- pH
- sulphates
- alcohol
- quality

2.2 Data Visualization

The general description on the dataset shows that the different features present means and standard deviations with noticeable variations between one and another. From here arises the need for standardizing variables in the pre-processing stage.

```
wine_df.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000
mean	7.215307	0.339666	0.318633	5.443235	0.056034	30.525319	115.744574	0.994697	3.218501	0.531268	10.491801	5.818378
std	1.296434	0.164636	0.145318	4.757804	0.035034	17.749400	56.521855	0.002999	0.160787	0.148806	1.192712	0.873255
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000	0.220000	8.000000	3.000000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000	0.992340	3.110000	0.430000	9.500000	5.000000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000	0.994890	3.210000	0.510000	10.300000	6.000000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000	0.996990	3.320000	0.600000	11.300000	6.000000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000	2.000000	14.900000	9.000000

Figure 2.2: Description of Dataset

The next step is visualizing how the data is divided between the classes: White wine instances are noticeably more numerous than red wine instances as shown in the following figure.

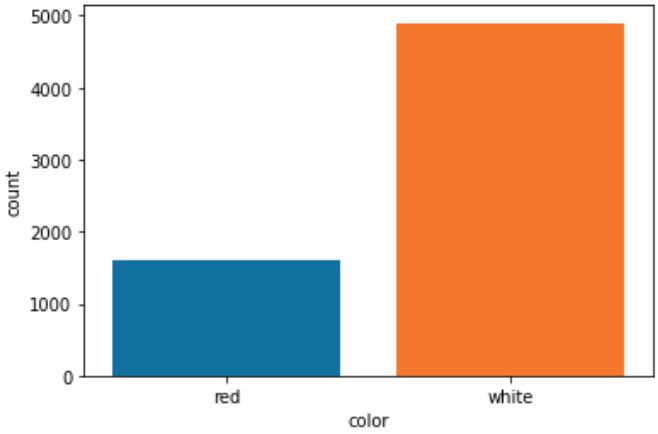

Figure 2.3: Distribution of classes in the dataset

Another interesting thing to check is the correlation between the color and the other features: fixed acidity, volatile acidity as well total sulfur dioxide seem to play a great role in defining the class of wine (red wine or white wine).

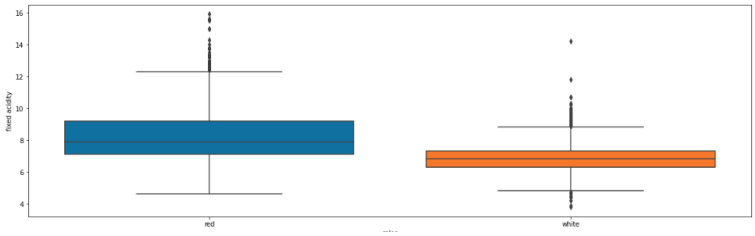

Figure 2.4: Relation between color of wine and fixed acidity

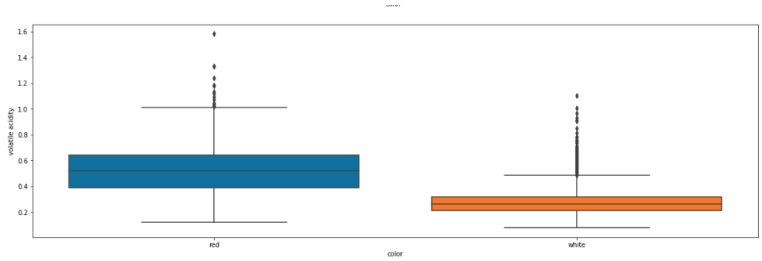

Figure 2.5: Relation between color of wine and volatile acidity

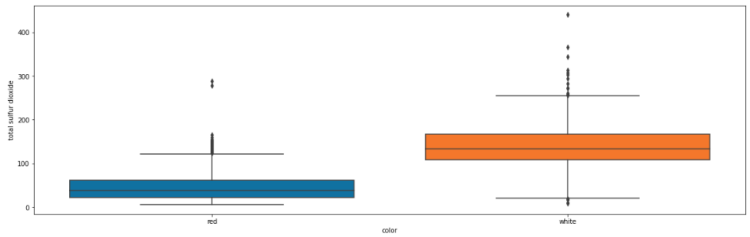

Figure 2.6: Relation between color of wine and total sulfur dioxide

One last thing to mention: it appears to be a strong correlation between the total sulfur dioxide and the free sulfur dioxide as shown in the correlation matrix below.

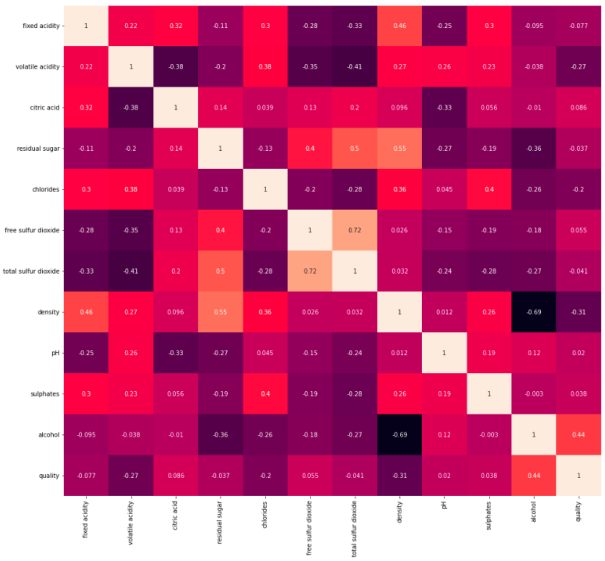

Figure 2.7: Correlation Matrix

2.3 Pre-processing

This step is fundamental in the machine learning project workflow. We started by the conversion of numerical quality values into 'bad' or 'good' while supposing the wine that has a quality higher than 6.5 is good or bad otherwise.

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	color
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	bad	red
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	bad	red
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	bad	red
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	bad	red
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	bad	red
...	...	...	...	...	...	...	...	...	...	...	...	...	...
6492	6.2	0.21	0.29	1.6	0.039	24.0	92.0	0.99114	3.27	0.50	11.2	bad	white
6493	6.6	0.32	0.36	8.0	0.047	57.0	168.0	0.99490	3.15	0.46	9.6	bad	white
6494	6.5	0.24	0.19	1.2	0.041	30.0	111.0	0.99254	2.99	0.46	9.4	bad	white
6495	5.5	0.29	0.30	1.1	0.022	20.0	110.0	0.98869	3.34	0.38	12.8	good	white
6496	6.0	0.21	0.38	0.8	0.020	22.0	98.0	0.98941	3.26	0.32	11.8	bad	white

6497 rows × 13 columns

Figure 2.8: Converted quality values

Since we are dealing with a classification problem with two classes, it is quite important to convert the quality labels and the color labels into ones and zeros instead of having them as sequence of characters: For the quality, we assign zero for 'bad' quality and one otherwise. As for the color, it is zero for 'red' and one otherwise.

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	color
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	0	0
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	0	0
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	0	0
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	0	0
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
6492	6.2	0.21	0.29	1.6	0.039	24.0	92.0	0.99114	3.27	0.50	11.2	0	1
6493	6.6	0.32	0.36	8.0	0.047	57.0	168.0	0.99490	3.15	0.46	9.6	0	1
6494	6.5	0.24	0.19	1.2	0.041	30.0	111.0	0.99254	2.99	0.46	9.4	0	1
6495	5.5	0.29	0.30	1.1	0.022	20.0	110.0	0.98869	3.34	0.38	12.8	1	1
6496	6.0	0.21	0.38	0.8	0.020	22.0	98.0	0.98941	3.26	0.32	11.8	0	1

6497 rows × 13 columns

Figure 2.9: Converted quality and color values

The final step of pre-processing is standardizing the data after splitting it into a train set (80% of the original dataset) and test set (20% of the original dataset). Both datasets pass then through StandardScaler of sklearn's preprocessing package that trains on the train set features then transforms the test set features.

```
X_train
array([[-0.55217795, -0.91147336, -0.26072779, ..., -0.14150511,
         1.01626334,  1.98971593],
       [ 0.46852561, -0.24180106, -0.1924072 , ..., -1.01196191,
        -1.26008699, -0.50258431],
       [-1.33733453,  0.2147937 ,  0.42247814, ..., -0.9450037 ,
         0.00455208, -0.50258431],
       ...,
       [-0.63069361, -0.78971476, -0.26072779, ..., -0.87804548,
         0.42609844, -0.50258431],
       [ 1.33219785, -0.42443896,  0.14919577, ..., -0.34237976,
         0.08886136, -0.50258431],
       [ 1.41071351, -1.03323196,  1.17400466, ..., -0.14150511,
         0.00455208,  1.98971593]])

X_test
array([[-0.31663097, -0.66795616, -0.05576601, ...,  0.72895169,
         0.34178917,  1.98971593],
       [-0.94475624, -0.91147336, -0.39736897, ..., -1.14587834,
        -1.34439626, -0.50258431],
       [ 0.31149429, -0.72883546, -0.26072779, ..., -1.54762764,
        -1.09146844, -0.50258431],
       ...,
       [-0.15959966, -0.36355966,  0.49079873, ..., -0.00758868,
         0.00455208, -0.50258431],
       [-0.23811531, -1.09411126,  0.35415755, ...,  0.39416061,
         1.26919116, -0.50258431],
       [-0.55217795,  0.30611265, -0.60233075, ..., -1.21283656,
        -0.41699427, -0.50258431]])
```

Figure 2.10: Standardized data

2.4 Training the models

After pre-proprocessing and splitting our data into a train set and a test set like the following:

- 80% for training
- 20% for testing

The training was done on the training set using Logistic Regression, K-Nearest Neighbors, Support Vector Machine (Linear Classifier), Support Vector Machine (RBF Classifier), Decision Tree and Random Forest. The training generated the following accuracies:

Training

```
In [122]: #Get and train all of the models
          model = models(X_train,Y_train)

[0]Logistic Regression Training Accuracy: 0.993650182797768
[1]K Nearest Neighbor Training Accuracy: 0.9942274389070618
[2]Support Vector Machine (Linear Classifier) Training Accuracy: 0.9946122763132577
[3]Support Vector Machine (RBF Classifier) Training Accuracy: 0.996921300750433
[4]Decision Tree Classifier Training Accuracy: 0.9998075812969021
[5]Random Forest Classifier Training Accuracy: 0.9996151625938041
```

Figure 2.11: Training accuracies

2.5 Models evaluation and comparison

The following table sums up the different metrics that were calculated to compare between the models:

Model	Accuracy	Accuracy cross-validation	Precision	Recall	F-score
Logistic Regression	0.996	0.979	0.997	0.998	0.998
KNN	0.997	0.944	0.998	0.998	0.998
Linear SVM	0.998	0.987	0.999	0.998	0.998
RBF SVM	0.997	0.936	0.998	0.999	0.999
Decision Tree	0.982	0.986	0.991	0.985	0.985
Random Forest	0.995	0.994	0.997	0.997	0.997

Overall, we can say that the Random Forest classifier generated the best predictions.

Conclusion and perspectives

This project is a personal work for a Machine Learning Lab for school.

At first we presented the project overview: the wine classification context and the models we used.

Next we moved on to the project walkthrough starting from Loading our Data, visualizing it, pre-processing the data, training the models and finally evaluating their results and comparing them.