Fundamentals of Data Science Report

Wine Dataset Classification

Sara Marques

Index

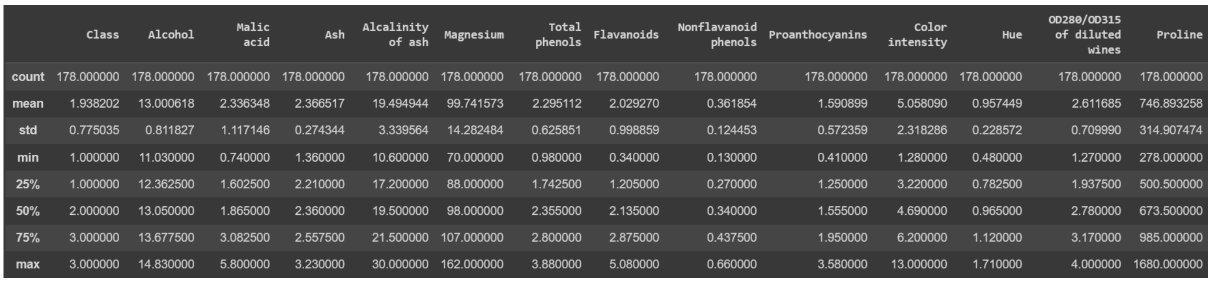
1. Introduction

The goal of this project is to perform classification on the wine dataset and evaluate the performance of the classification model.

2. Dataset

The wine data set consists of 178 samples with 13 numerical features attributes: Alcohol, Malic Acid, Ash, Alcalinity of ash, Magnesium, Total phenols, Flavanoids, Nonflavanoid phenols, Proanthocyanins, Color intensity, Hue, OD280/OD315 of diluted wines and Proline. The first feature is the target variable indicating the wine class which can be type 1, type 2 or type 3.

The first thing to do after downloading the dataset from the url using the pandas.read_csv() function is to do a statistical summary of the data. This can be done using the pandas.DataFrame.describe() function which prints a table with metrics such as count, mean, standard deviation, minimum, quartiles, and maximum for each numerical column in the dataset. We can then observe the range, spread, and central tendency of the different attributes.



	Class	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color intensity	Hue	OD280/OD315 of diluted wines	Proline
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000
mean	1.938202	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.029270	0.361854	1.590899	5.058090	0.957449	2.611685	746.893258
std	0.775035	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.998859	0.124453	0.572359	2.318286	0.228572	0.709990	314.907474
min	1.000000	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.340000	0.130000	0.410000	1.280000	0.480000	1.270000	278.000000
25%	1.000000	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.205000	0.270000	1.250000	3.220000	0.782500	1.937500	500.500000
50%	2.000000	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.135000	0.340000	1.555000	4.690000	0.965000	2.780000	673.500000
75%	3.000000	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.875000	0.437500	1.950000	6.200000	1.120000	3.170000	985.000000
max	3.000000	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.080000	0.660000	3.580000	13.000000	1.710000	4.000000	1680.000000

Fig. 1 - Statistical summary of the Wine dataset

3. PCA

After having the dataset loaded, we can reduce its dimensionality using the Principal Component Analysis. PCA is a technique for analyzing large datasets containing a high number of dimensions/features per observation, increasing the interpretability of data while preserving the maximum amount of information, and enabling the visualization of multidimensional data. In this case, we reduced the dimensionality of the dataset to 2 components using the sklearn.decomposition.PCA() function. After fitting the dataset into 2 components, we can use the PCA().components_ attribute which gives us the principal axes in feature space, representing the directions of maximum variance in the data and the PCA().explained_variance_ratio_ attribute which gives us the percentage of variance explained by each of the selected components. The result from PCA().explained_variance_ratio_ was the array [0.99809123, 0.00173592], so the total variance explained is 99.98% which means that the first two principal components explain the majority of the variance in this dataset. This is an indication of the total information represented compared to the original data.

We can then visualize the reduced data in a two-dimensional space (x-axis: PCA Component 1 and y-axis: PCA Component 2), coloring each point with a different color corresponding to the different wine types: red (type 1), green (type 2) and blue (type 3). The result is shown below:

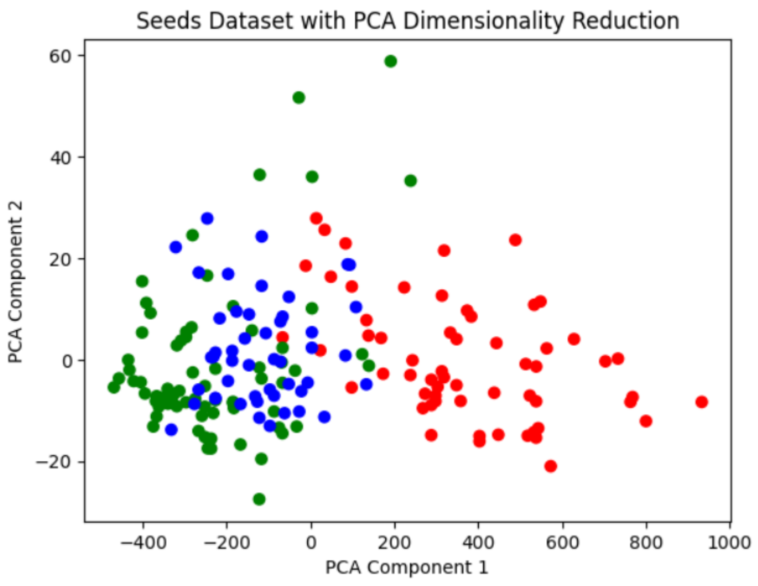


Fig. 2 - Reduced dataset plot

We can see that there is a clear differentiation of the 3 wine types especially between type 1 and 2. Type 1 is more to the right in the PCA Component 1 axis while type 2 is more to the left. Type 3 is in the middle of them.

4. Clustering

The next step was to cluster the dataset and evaluate the clustering results with classification labels. This was done first using the sklearn.cluster.AgglomerativeClustering() function which recursively merges pairs of clusters of sample data, and then using sklearn.cluster.KMeans() which is a centroid-based clustering algorithm, where we calculate the distance between each data point and a centroid to assign it to a cluster. The difference between the two algorithms is that K-Means uses a pre-specified number of clusters while Agglomerative Clustering seeks to build a hierarchy of clusters without having a fixed number of clusters.

Using first the Agglomerative Clustering algorithm without specifying the number of clusters and setting the parameter distance_treshold (linkage distance threshold at or above which clusters will not be merged) to 0, we can plot a dendrogram to see what should be the optimal number of clusters:

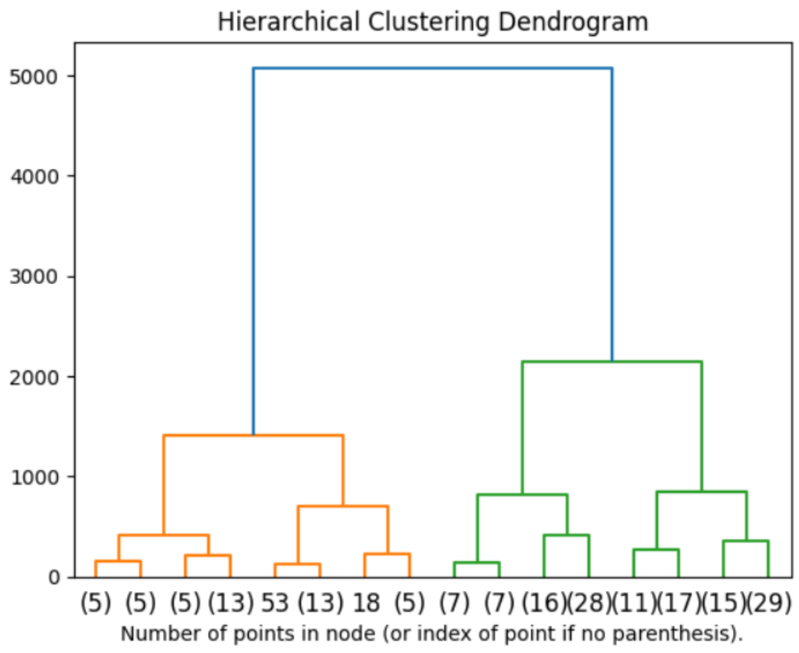


Fig 3. - Dendrogram plot of the Agglomerative Clustering algorithm on the Wine dataset

From the above dendrogram plot, we can find a horizontal rectangle with max-height that does not cross any vertical dendrogram line:

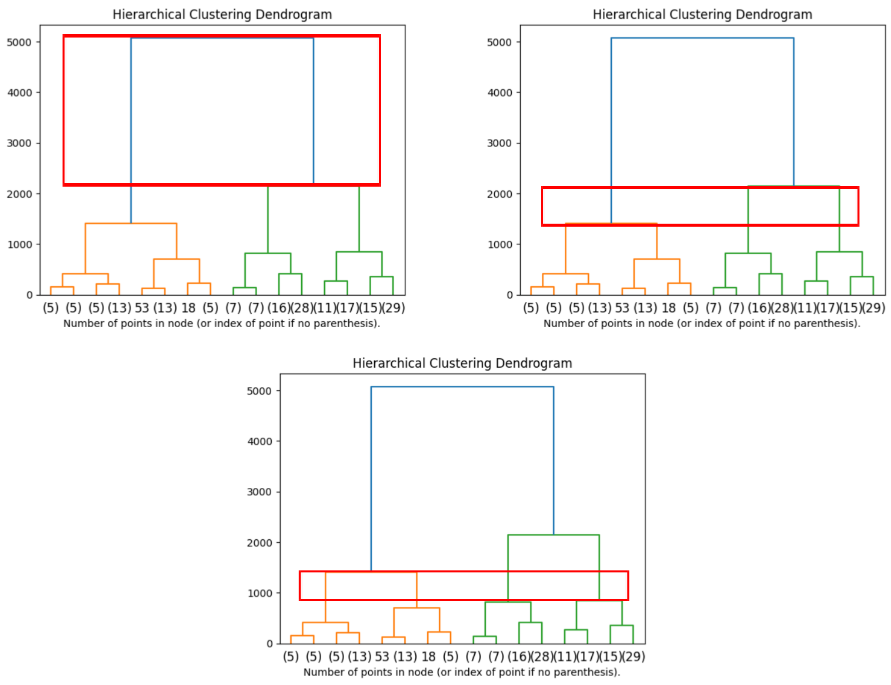


Fig. 4 - Height comparison in each rectangle corresponding to 2, 3 and 4 clusters respectively

The portion in the dendrogram in which a rectangle has the maximum height can be cut in the first image so the optimal number of clusters will be 2. This can make sense because of what we saw in Fig 2. - there was a type of wine (type 3) that wasn't that distinct from the others, so its data points can be distributed by the other two clusters. The rectangle with the next maximum height corresponds to 3 clusters which is the number of different wine classes in the dataset.

We then compared the results using K-Means with 3 clusters on the original dataset with 13 attributes and on the reduced dataset with PCA. The comparison was done using the Rand index score which measures the similarity between the obtained clusters and the actual wine classes. A higher ARI value indicates better clustering performance. Performing this metric on both normal and reduced datasets, we observed that the rand scores are the same and relatively high (71.9%) which makes sense because like I said before, the 2 components resulting from the PCA transformation cover almost 100% of the data information.

5. Splitting the dataset

After that the dataset was split into training and testing sets using the sklearn.model_selection.train_test_split() function which splits the dataset array into two random subsets. The parameter test_size was set to 0.20 (20% of the dataset is for test and 80% for train) and the parameter random_state was set to 10. This parameter controls the shuffling applied to the data before applying the split. The training set will be used to train the classification model (learn the underlying patterns and relationships), while the testing set will be used to evaluate its performance.

6. Classification

After splitting into training and testing sets, the training set is used to build a model that will be able to classify the testing set. For the classification there were tested many machine learning models both linear and nonlinear, all from the sklearn library. Below there is a brief explanation of each one of them and which parameters were used.

Logistic Regression - this model uses the sigmoid function to return the probability of a label. It is widely used when the classification problem is binary. The sigmoid function generates a probability output. By comparing the probability with a pre-defined threshold, the object is assigned to a label accordingly. There were used the default parameters.

Decision Trees - this model builds tree branches in a hierarchy approach and each branch can be considered as an if-else statement. The branches develop by partitioning the dataset into subsets based on most important features. Final classification happens at the leaves of the decision tree. There were used the parameters criterion set to “gini” (function to measure the quality of a split) and random_state set to 10.

Support Vector Machine (SVM) - this model finds the best way to classify the data based on the position in relation to a border between positive class and negative class. This border is known as the hyperplane which maximizes the distance between data points from different classes. Similar to decision tree and random forest, support vector machine can be used in both classification and regression, SVC (support vector classifier) is for classification problems. There were used the parameters kernel set to “linear” (kernel type to be used in the algorithm) and C set to 0.025 (regularization parameter).

Random Forest - this model is a collection of decision trees. It is a common type of ensemble method which aggregate results from multiple predictors. Random forest additionally utilizes bagging techniques that allow each tree trained on a random sampling of original dataset and takes the majority vote from trees. Compared to the decision tree, it has better generalization but less interpretable, because of more layers added to the model. There were used the parameters max_depth (maximum depth of the tree), n_estimators set to 10 (number of trees in the forest) and max_features set to 1 (number of features to consider when looking for the best split). I tried varying the max_depth parameter from 1 to 10 and the best result was obtained with max_depth equal to 4.

Naive Bayes - this model is based on Bayes’ Theorem - an approach to calculate conditional probability based on prior knowledge, and the naive assumption that each feature is independent of each other. The biggest advantage of Naive Bayes is that, while most machine learning algorithms rely on a large amount of training data, it performs relatively well even when the training data size is small. Gaussian Naive Bayes is a type of Naive Bayes classifier that follows the normal distribution. There were used the default parameters.

K-Nearest Neighbour (KNN) - this model represents each data point in a n dimensional space — which is defined by n features. It calculates the distance between one point to another, then assigns the label of unobserved data based on the labels of nearest observed data points. There were used the parameters n_neighbors (number of neighbors to use) and metric (metric to use for distance computation). I tried with the number of neighbors varying from 1 to 10 and with 3 different metrics (“euclidean”, “manhattan” and “cosine”). The best result was obtained with the cosine metric and n_neighbors set to 2 and the manhattan metric and n_neighbors set to 5.

Multi-layer Perceptron Classifier - this model is a supervised learning algorithm where the mapping between inputs and output is non-linear. It has input and output layers, and one or more hidden layers with many neurons stacked together. There were used the parameters alpha set to 1 (strength of the L2 regularization term) and max_iter set to 1000 (maximum number of iterations).

7. Results

The performance of each model was measured by the percentage of correctly predicted samples obtained with the testing set and were saved in an array called scores and then plotted using matplotlib. The results are shown below:

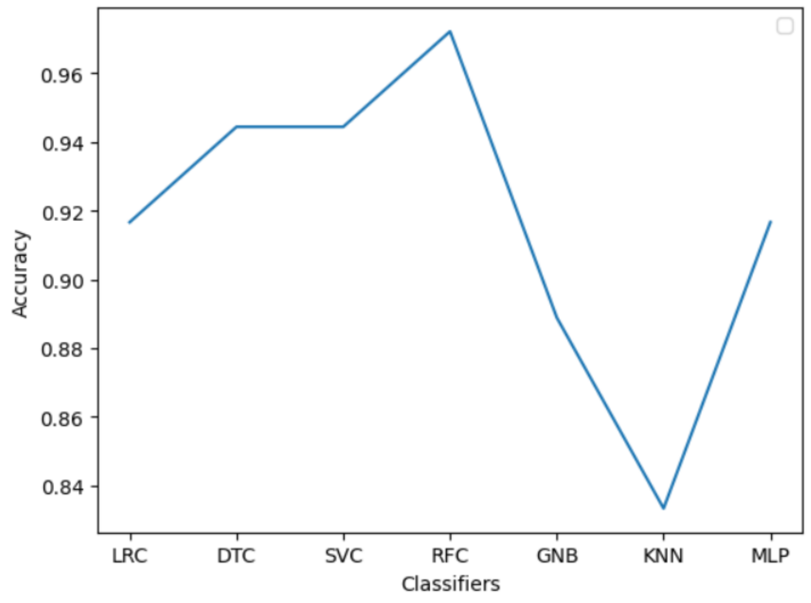
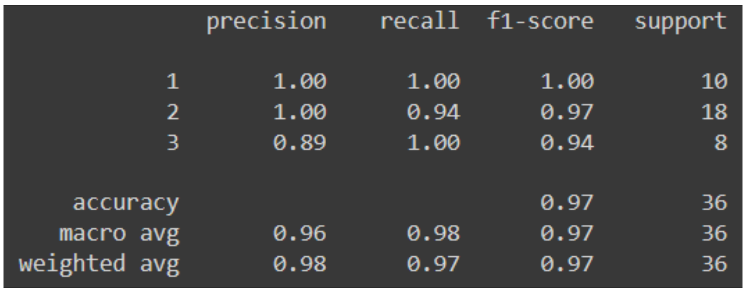


Fig. 4 - Accuracy of each model

The results are all relatively good with the best one being obtained by the Random Forest Classifier with maximum depth set to 4 (accuracy 97.2%). The worst one was the K-nearest neighbors algorithm with 3 neighbors (accuracy 83.3%).

8. ChatGPT

After obtaining the results we can ask ChatGPT to do the same procedure and compare how this information differs. When asked to perform a statistical summary of the data, ChatGPT provides exactly the same code I wrote in the notebook. To reduce the dimensionality of the wine data and visualize it, it also provides the same code as the one done in the lab classes. Next, to Cluster the dataset and evaluate clustering results with classification labels, ChatGPT also used the K-means algorithm to create the clusters and to evaluate it, it uses the already build-in sklearn.metric.adjusted_rand_score() but does the same as the one implemented in the lab classes. The next step was to split the dataset into training and testing sets and ChatGPT also provides the same code as I have. For the last step, performing classification ChatGPT uses the Logistic Regression classifier (probably because of its simplicity and interpretability) with default parameters like I did and to evaluate its result, besides using the score() function uses the sklearn.metrics.classification_report() function which builds a text report showing the main classification metrics including precision, recall, and F1-score for each class. Using this function on our best and worst models we obtain respectively:

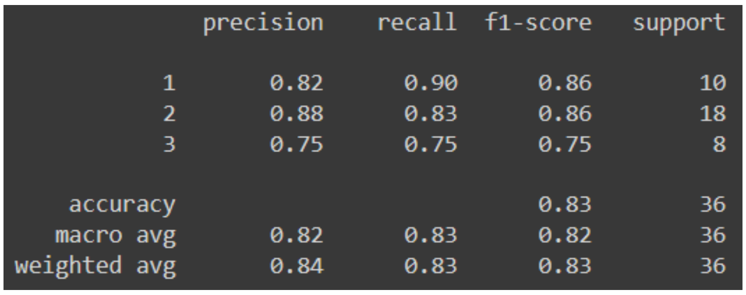


```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        10
           2       1.00      0.94      0.97        18
           3       0.89      1.00      0.94         8

    accuracy                           0.97        36
   macro avg       0.96      0.98      0.97        36
weighted avg       0.98      0.97      0.97        36
```

Fig. 5 - Classification report of the best model (Random Forest Classifier)



```
              precision    recall  f1-score   support

           1       0.82      0.90      0.86        10
           2       0.88      0.83      0.86        18
           3       0.75      0.75      0.75         8

    accuracy                           0.83        36
   macro avg       0.82      0.83      0.82        36
weighted avg       0.84      0.83      0.83        36
```

Fig. 6 - Classification report of the worst model (KNN Classifier)

9. Conclusion

In conclusion, the classification task was successfully performed using several algorithms. Every model tried got a reasonable accuracy on the testing set (above 80%) which means that each one of them can effectively predict the wine classes based on the given features.

Overall, this project demonstrates the application of dimensionality reduction and classification techniques to analyze and classify the wine dataset. The obtained results provide valuable insights into the wine samples' classification and can be further utilized for wine quality assessment or other related applications.

10. Code

The link to the Google Colabs Notebook with the code is provided below:

https://colab.research.google.com/drive/17xf5cgbPezgqNo8fengV1A3OdVwwdbw_?usp=sharing