Wine assignment

Name: Nick Glas

Student number: 670516

Assignment/dataset: Wine dataset (homework 2)

Date: 21/05/2023

Git URL: https://github.com/nickglas/wine\_assignment

Contents

[Taken steps 3](#_Toc135573857)

[Explanation source code 4](#_Toc135573858)

[Results 14](#_Toc135573859)

# Taken steps

To begin the project, I imported the Wine dataset from scikit-learn and took the time to read through the documentation to understand its structure and variables. The dataset provides information about different attributes of wine samples, including their chemical properties and cultivator labels.

First, I constructed a pipeline to streamline the preprocessing steps. The pipeline included standard scaling of numeric features using StandardScaler and the creation of new features through the FeatureCreator function. I experimented with various classifiers such as LogisticRegression, RandomForestClassifier, SVC, GradientBoostingClassifier, and GaussianNB by assigning them to the classifier step in the pipeline.

In the Wine dataset I used, I didn't have to worry about missing values, outliers, or dealing with categorical features. That's because all the information we needed was already there and complete. We didn't have any missing pieces of data that we needed to guess or fill in. Similarly, there weren't any strange or unusual values that would have messed up our analysis. Everything seemed to be in the right range. Lastly, the dataset was already set up in a way that the computer could understand it easily. We didn't have to do any special tricks to handle words or categories. The dataset was already in a format that the computer could work with. So, in simple terms, we didn't have any missing information, weird numbers, or special language to deal with in this dataset.

During the dataset analysis, I made a few important steps to help me better understand and predict the wine types. Firstly, we made sure that all the numbers were on the same scale. This way, we could compare them more easily and avoid any biases that could come from using different measurement units. Secondly, I picked out the most important features that helped us predict the wine types accurately. We focused on the things that had the biggest impact and ignored the less important details. Lastly, I created new features by combining or using additional information. This allowed us to see the data from different angles and discover new patterns. By doing all this, we improved our analysis and made it easier to predict the wine types accurately.

To evaluate the performance of the pipeline with different classifiers, I calculated the accuracy, R2 score, precision, recall, and displayed the confusion matrix. We used the score function to obtain accuracy and R2 scores, while the precision\_score, recall\_score, and confusion\_matrix functions from scikit-learn allowed us to calculate precision, recall, and visualize the confusion matrix. Furthermore, I Used seaborn and matplotlib to create a heatmap representation of the confusion matrix for better visualization.

The pipeline approach provided an organized and efficient way to preprocess the data, engineer new features, select features, and compare multiple classifiers. By using the pipeline, I could easily swap out different classifiers and observe their performance. Overall, the code allowed for a systematic exploration and evaluation of the Wine dataset, aiding in the prediction of wine cultivators and providing insights into feature importance and model performance.

To conclude, by importing the dataset, reading the documentation, preprocessing the data using a pipeline, and exploring different classifiers, I was able to predict the cultivator of each wine sample accurately by using different classifiers. The project provided valuable insights into feature selection, model evaluation, and the performance of different classifiers on the Wine dataset.

# Explanation source code

In this chapter, we will take a closer look at the source code and explain how it works. We will go through the code step by step to understand how it functions. By doing this, we will learn how the code achieves its goals and what each line of code does. Our goal is to make the code easier to understand and help you gain insights into how the code works.

I started by creating a Python file where I could experiment with the basic functionality of the sklearn pipelines. I wanted to understand how pipelines work and how they can streamline the machine learning workflow. I imported the necessary libraries, including sklearn, and began by exploring the dataset I wanted to work with. I read the documentation to gain insights into the dataset's structure and the available features.

I firstly started of creating the script and importing the necessary libraries. After doing this, I declared some global variables like the dataset itself, the features and the feature names as well.

A screen shot of a computer

Description automatically generated with medium confidence

The FeatureCreator step in the script creates new features from the existing ones in the dataset. It looks for hidden patterns or relationships in the data and uses them to generate additional information that can help make better predictions. By adding these new features, we can get a better understanding of the data and improve the accuracy of our model.

The OutlierRemoval step focuses on finding and handling unusual or extreme data points called outliers. These outliers can sometimes disrupt the patterns in the data and affect the model's performance. The step identifies these outliers and takes appropriate actions to address them, so that the model can focus on the typical data patterns and make more reliable predictions.

A screen shot of a computer program

Description automatically generated with low confidence

The updateDataSet function allows for updating the original dataset (X) and the feature names (feature\_names) with new data or feature names. The function uses global variables (feature\_names and X) to store and update the original dataset and feature names. By using the global keyword, it ensures that the changes made inside the function are reflected outside of the function scope.

A screen shot of a computer

Description automatically generated with medium confidence

This function takes the wine dataset as input and creates two new features based on existing features. It uses three relevant features: 'color\_intensity', 'total\_phenols', and 'flavanoids'.

1. Color Intensity Category: The function categorizes the color intensity values into three categories: 'Low', 'Medium', and 'High'. It compares each color intensity value to certain thresholds (5.0 and 10.0) and assigns the corresponding category based on whether the value is below or above these thresholds.
2. Flavanoid Proportion: The function calculates the proportion of flavanoids in each sample. It divides the flavanoid values by the total phenols values to determine the relative concentration of flavanoids.

After creating these new features, the function appends them to the original dataset, creating a new dataset with the additional features. It also updates the feature names by adding 'color\_intensity\_category' and 'flavanoid\_proportion' to the existing feature names.

Finally, the function returns the new dataset and the updated feature names.

A screen shot of a computer code

Description automatically generated with low confidence

These methods get called in the main function of the test script. This is just the main entry point of the script when it gets called into the python shell.

A picture containing text, screenshot, font

Description automatically generated

I have made progress in organizing my code in a way that makes sense to me. To make things even clearer, I decided to use a Jupyter notebook. This notebook allows me to explain my code in a more detailed and interactive manner and include visualizations. This way, when I or someone else looks at the code later, it will be easier to understand what each part does and how it fits together. The Jupyter notebook helps me combine my code with explanations, making it more straightforward and accessible for anyone who reads it.

In the notebook version of my code, I made a significant change to the structure and organization. The main difference is that I restructured the code to work seamlessly with a concept called a pipeline. A pipeline is a powerful tool in machine learning that allows me to streamline and automate various steps in the data processing and model training process.

I firstly created a small header where my information, like my name, student number etc.. are described.

A black screen with white text

Description automatically generated with low confidence

This homework assignment was created using version 3.10.7 of Python. The code was thoroughly tested on Anaconda, specifically version 3.9.13. This testing ensured that the code worked well and produced the expected outcomes. I used visual studio code for my IDE and all output will be shown in the visual studio code format.

I created a special environment for my project called a virtual environment using a tool called venv. It's like a separate space where I can install and use specific versions of Python and its packages just for this project. This helps keep everything organized and prevents any conflicts with other projects or the main Python setup on my computer.

To set up my project and use the necessary tools, I installed some packages using the Anaconda kernel. These packages include sklearn, numpy, seaborn, and matplotlib. Sklearn is a powerful library for machine learning and data analysis, while numpy provides efficient numerical operations. Seaborn is a visualization library that helps me create appealing and informative plots, and matplotlib is another popular plotting library that offers a wide range of customization options.

A black screen with white text

Description automatically generated with low confidence

After installing the necessary libraries, I imported the right classes and function into the notebook environment.

A screen shot of a computer program

Description automatically generated with medium confidence

Just like the other script, let's start by importing the dataset and setting some global variables. This will allow me to access and manipulate the data throughout our project. Once the dataset is imported, I can proceed to print the features and take a closer look at the imported wine dataset.

A screenshot of a computer program

Description automatically generated with medium confidence

We can see the results of the print statements in the following log. The features names are clearly visible and we can see that there is data inside of the train variable, indicating that the train\_test\_split method did his job.

A screen shot of a computer

Description automatically generated with medium confidence

Now, let's move on to the creation of a class that will be responsible for generating two new features. This class, named 'FeatureCreator,' is designed to inherit the functionality of the BaseEstimator and TransformerMixin classes from the sklearn library. With the FeatureCreator class, we will be able to generate two additional features: color\_intensity\_category and flavanoid\_proportion. The color\_intensity\_category feature is a categorical variable that represents the intensity of color in the wine samples. It is assigned a single value, which can be 1, 2, or 3, indicating the low, medium, or high category. A value of 1 represents the lowest intensity, while 3 represents the highest intensity.

On the other hand, the flavanoid\_proportion feature is a calculated variable that measures the relative concentration of flavanoids in each wine sample. It is obtained by dividing the flavanoids by the total phenols. This ratio provides valuable information about the proportion of flavanoids present in relation to the overall phenolic compounds in the wine. By incorporating the FeatureCreator class into our pipeline, we can integrate these newly created features into our dataset, enriching it with additional information that can potentially improve the performance of our models and provide deeper insights into the wine dataset. This is only creating the class. The implementation in the pipeline will come at a later stage.

A screen shot of a computer program

Description automatically generated with low confidence

One of my assignments was to identify the top three predictive features in my dataset, I implemented three different methods: univariate\_feature\_selection, recursive\_feature\_elimination, and feature\_importance. I conducted some research to understand the concepts and workings of these methods.

* Univariate feature selection involves evaluating each feature in the dataset individually to determine its importance in predicting the target variable. It assesses the relationship between each feature and the target variable independently, allowing us to identify the most influential features.

A screen shot of a computer program

Description automatically generated with low confidence

* Recursive feature elimination, on the other hand, is a technique that iteratively eliminates less important features from the dataset. It starts by training a model, assessing the importance of each feature, and removing the least important one. This process continues until the desired number of features is reached, resulting in a subset of the most relevant features.

A screen shot of a computer program

Description automatically generated with low confidence

* Feature importance is a measure or score that indicates the significance of each feature in predicting the target variable. It helps us understand the relative contribution or influence of each feature towards the outcome we are trying to predict. By identifying the most important features, we gain insights into the crucial factors or attributes that drive the target variable's behavior.

A screen shot of a computer code

Description automatically generated with low confidence

These methods use the global X and y variables, which were defined in previous code snippets, to perform their calculations and evaluations. By employing these feature selection techniques, we can gain a deeper understanding of our dataset and focus on the most informative features for building accurate predictive models.

The following code snippet serves as the main entry point for this file. When the script is executed, this function is automatically called. To view all the output and results generated by the code, it is recommended to open the debug/log window as a scrollable element (This is required in visual studio code). This ensures that no information is missed or truncated, allowing you to view the complete output of the script.

Let me explain the code for the main function.

1. In the first row I create a variable called numeric\_features. This creates a slice object, which is used for extended slicing. We can only do this because our dataset contains numeric values



1. I Create a ColumnTransformer object with the numeric\_features variable. We used the standard scaler in the pipeline to perform feature scaling on the input data. The standard scaler standardizes features by subtracting the mean and dividing them by the standard deviation. Another reason is to avoid Bias. Standardization helps prevent this bias and ensures that all features are equally considered during model training.

A picture containing text, font, screenshot

Description automatically generated

1. The next step is to create the classifier types and the pipeline. The first argument is the scaller arg. This argument uses the same scaler as the column\_transformer\_scaler. The next step is to implement the preprocessing step. This step uses column\_transformer\_scaler we created before. Once again this is done to perform feature scaling on the input data. The next step is to create new features. This step is done in the feature\_engineering section. It uses the FeatureCreator class we created beforehand. Look at the code snippet above for further explanation.

A screen shot of a computer program

Description automatically generated with low confidence

1. I used the following functions to determine the most important features in my dataset: univariate\_feature\_selection, feature\_importance, and recursive\_feature\_elimination.

A screen shot of a computer code

Description automatically generated with low confidence

1. Finally, in the pipeline, we need to select a classifier to make predictions. There are several options available, such as RandomForestClassifier, LogisticRegression, SVC (Support Vector Machines), GradientBoostingClassifier, and GaussianNB (Naive Bayes). In step 3 I created a classifiers array holding different classifiers. I loop over these values and import the right classifier. I set this new classifier, applied the data and get the scores for each classifier.

A screen shot of a computer program

Description automatically generated with low confidence



# Results

In this chapter we will take a look at my assignments and the results that I got from executing the code snippets. My assignments are:

* Pre-process necessary features: I performed necessary data preprocessing steps, such as scaling numeric features to ensure comparability and handling any missing values or outliers. However, for this particular dataset, some of these preprocessing steps were not required.
* Design 2 new features: I created two additional features called "color\_intensity\_category" and "flavanoid\_proportion" using the FeatureCreator class. These new features provided additional information and insights into the dataset.

A black background with white text

Description automatically generated with low confidence

* Predict the cultivator of each wine measured: I employed various classification algorithms, including RandomForestClassifier, LogisticRegression, SVC, GradientBoostingClassifier, and GaussianNB, to predict the cultivator of each wine sample.

A picture containing text, font, screenshot, typography

Description automatically generated

* Find the top 3 predictive features according to 3 different methods of measuring predictiveness: I utilized three different feature selection methods, namely univariate\_feature\_selection, recursive\_feature\_elimination, and feature\_importance, to determine the top three features that had the most significant impact on predicting the cultivator.

A screen shot of a computer

Description automatically generated with medium confidence

* Report score/accuracy in at least 2 different formats: To assess the performance of the classifiers, I reported the accuracy scores using different formats. I also added a heatmap/confusion matrix for each classifier.

A screenshot of a graph

Description automatically generated with medium confidence

A screenshot of a graph

Description automatically generated with medium confidence

A screenshot of a graph

Description automatically generated with medium confidence

A screenshot of a graph

Description automatically generated with medium confidence

A screenshot of a graph

Description automatically generated with medium confidence

At last, I print a message stating which classifier did the best.

