Faculty of Mechanical Engineering and Robotics

Department of Robotics and Mechatronics

Python for Machine Learning and Data Science

Course for Mechatronic Engineering

Instruction 3:

Feature selection and extraction

and

Feature transformation

You will learn: which features can be useful, how to select and transform them, what feature scaling is, how PCA works, and how to remove statistical outliers.

Additional materials:

- Course lecture 2 [*obligatory*] http://galaxy.agh.edu.pl/~zdw/Materials/Python/LectureNotes/
- Matlab to Python handout http://galaxy.agh.edu.pl/~zdw/Materials/Python/Matlab%20-%20python%20handout.pdf

Learning outcomes supported by this instruction:
IMA1A_W07, IMA1A_U01, IMA1A_U05, IMA1A_U07, IMA1A_U14, IMA1A_K08

Course supervisor:
Ziemowit Dworakowski, zdw@agh.edu.pl

Instruction author:
Adam Machynia, machynia@agh.edu.pl

Introduction

This instruction will briefly introduce you to the topic of feature selection and transformation. You will learn how to scale features and why this is important. Finally, you will become familiar with PCA.

During this laboratory, we still use the *wine quality* dataset. Use the *red* part unless stated otherwise.

Correlation for feature selection

Correlation can be used to identify features suitable for making predictions. In the case of regression, we want to find features that are highly correlated with the target but not correlated with each other. An example of a correlation matrix is shown in Figure 1. **Starting from Task 3.1, work only on the training subset!**

Task 3.1: Load the dataset. Split it for train, validation, and test sets with a ratio of 0.6:0.2:0.2. Set the *random_state* parameter to 1. Plot the correlation matrix. Enable annotations, and choose a suitable colormap. In pairs, discuss which features would be useful for predicting the levels of density and pH. Justify your choices.

Hint: You do not need to split the target from the data:
data_train, data_test = train_test_split(data, ...).

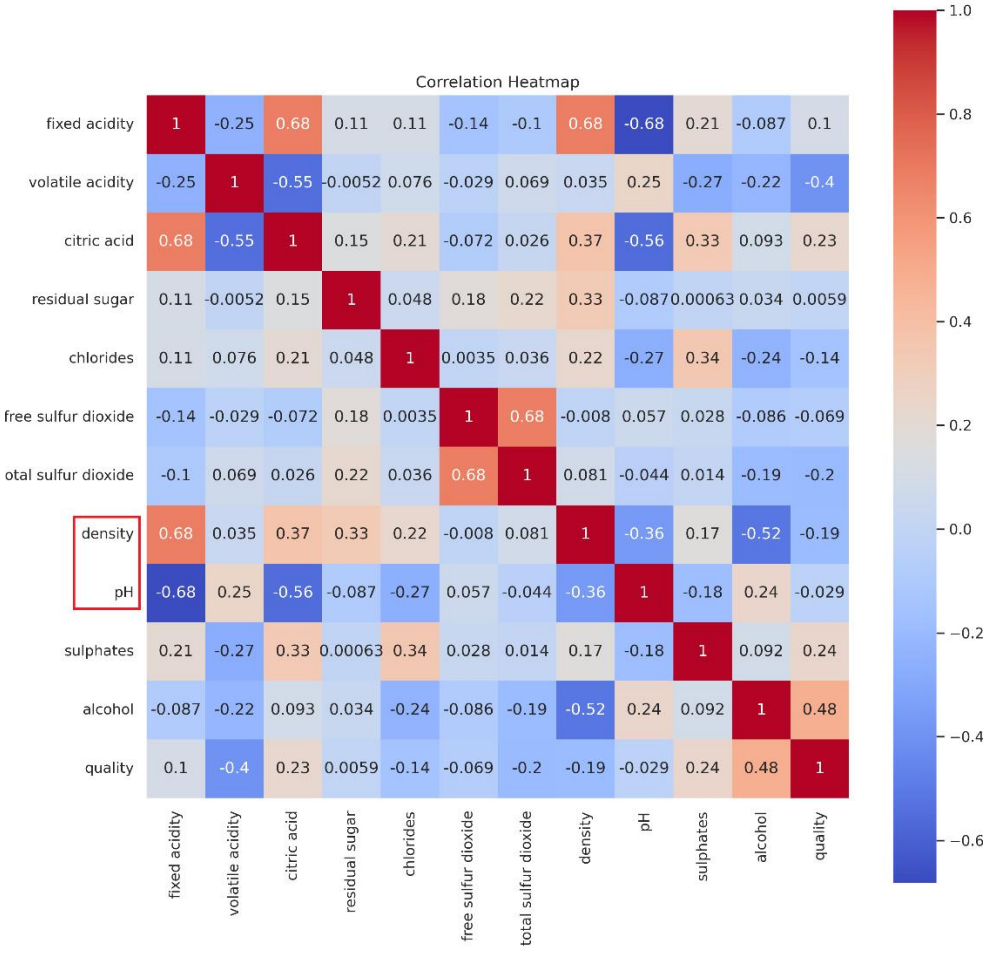

Fig 1 – Correlation matrix for red wine dataset.

Task 3.2: Repeat task 3.1 for the white wine dataset. Consider differences.

Statistical outliers removal

Sometimes we may want to exclude certain parts of the data based on statistical measures, for example, to remove outliers. Let's assume we want to exclude from our analysis 1% of observations with the lowest and highest pH. Here's how we can approach this task:

```
q_low = data_train['pH'].quantile(0.1) # get 1st percentile
q_hi  = data_train['pH'].quantile(0.99) # get 99th percentile

data_filtered = data_train[(data_train['pH'] < q_hi) & (data_train['pH'] > q_low)] #
filter data by percentiles
```

Task 3.3: Exclude 1% of observations with the lowest and highest pH. Compare the distribution of this feature before and after this operation.

Task 3.4: Implement the exclusion of specific parts of data based on other selected features. Consider which features might be reasonable and explore other statistical measures.

Feature scaling

Many machine learning models require standardization or scaling of the data, or at least perform better with these techniques. Generally, different ranges of feature values can cause significant issues if not handled properly. Consider gradient-based optimization algorithms dealing with two features: one varying slightly within a range from 0 to 1, and the other jumping from -1000 to 1000. This is where feature scaling becomes important.

Let's take a look at the distribution of sulphates relative to total sulfur dioxide in Figure 2. Notice the ranges of these two variables.

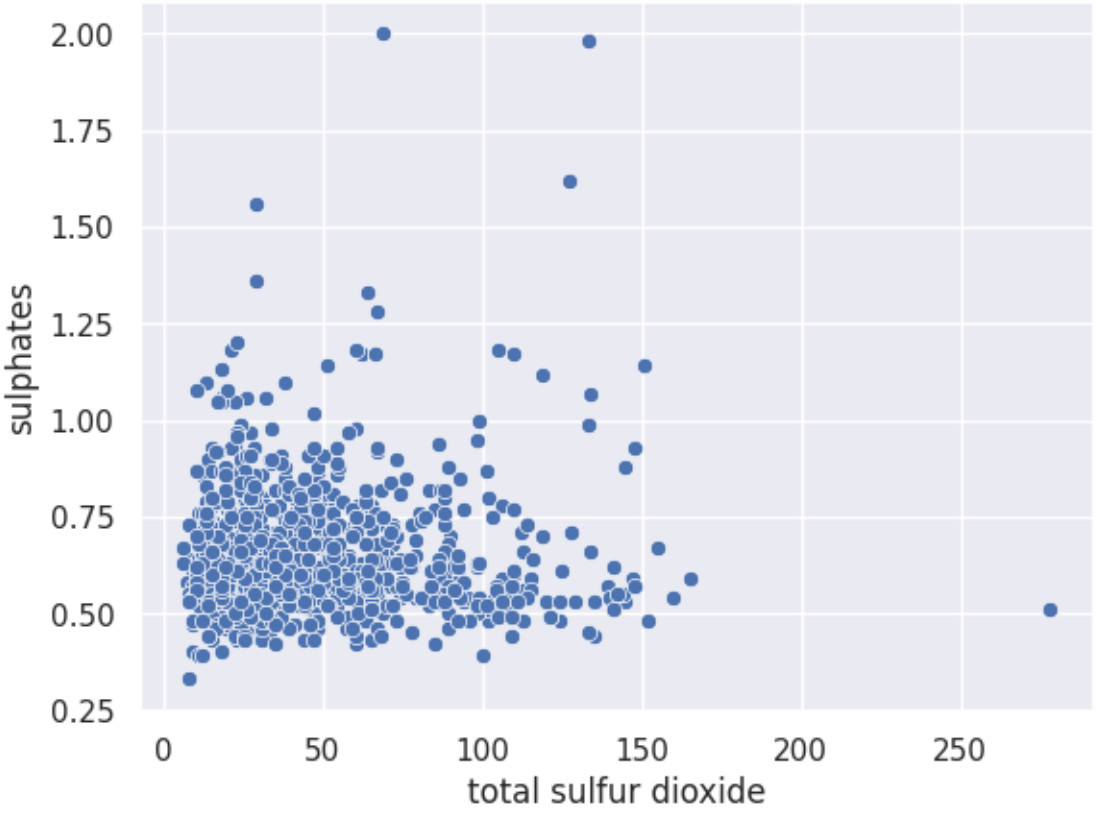

Fig 2 – Sulphates versus total sulfur dioxide.

Task 3.5: Check the feature value ranges and their distributions. Determine which features' distributions are similar and which differ. Consider the possible sources of similarities and differences in feature distributions.

Hint: check what happens when you type:
data.hist() or *data.boxplot()*.

The most common feature scaling approaches are standardization and Min-Max scaling. Standardization, also known as Z-score normalization, scales features to a standard normal distribution with a mean of zero and a standard deviation of one. In other words, it removes the mean and scales to unit variance, as shown in the equation:

$$z = \frac{x - \mu}{\sigma}$$

where x is the original value, z is the rescaled value, μ is the mean, and σ is the standard deviation.
Min-Max scaling, also called normalization, scales all values to a specific range, which is commonly from 0 to 1. This is done according to the following equation:

$$x_{minmax} = \frac{x - \mathrm{x}_{min}}{\mathrm{x}_{max} - \mathrm{x}_{min}}$$

where x is the original value, x_{minmax} is the rescaled value, x_{min} is the minimum in the set, x_{max} is the maximum in the set. These two operations can be easily applied using the scikit-learn library. You can see the results of these transformations in Figure 3.

Generally, we do not scale target variables alongside other features, so don't forget to drop the quality column before scaling.

```
quality = data['quality']
data = data.drop(columns = ['quality'])

from sklearn.preprocessing import (
    MinMaxScaler,
    StandardScaler,
)

standard_scaler = StandardScaler().set_output(transform="pandas")
data_standardized = standard_scaler.fit_transform(data)

minmax_scaler = MinMaxScaler().set_output(transform="pandas")
data_normalized = minmax_scaler.fit_transform(data)
```

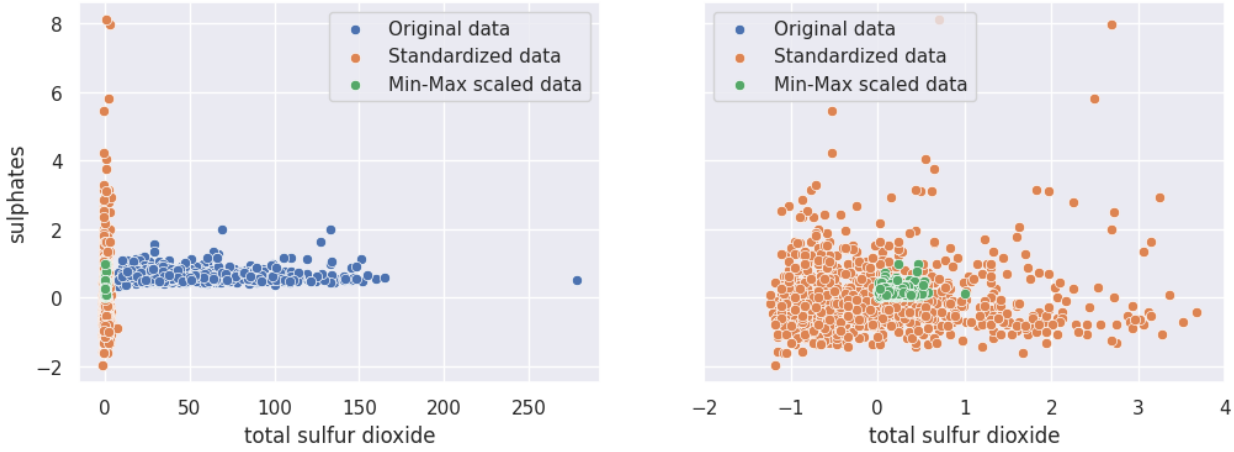

Fig 3 – Sulphates versus total sulfur dioxide: original, standardized, and Min-Max scaled data.

Task 3.6: Perform standardization and Min-Max scaling on the training data. Compare feature distributions as in the previous task. Consider in which cases standardization should be used and when Min-Max scaling is more appropriate.

Task 3.7: Consider the impact of the outliers on feature scaling. Repeat Task 3.6 after removing statistical outliers. You may drop 1 or 2 percentiles from the top and bottom of the selected features.

Principal component analysis (PCA)

Principal Component Analysis (PCA) is used to reduce the dimensionality of the data (have you ever heard of the curse of dimensionality?). However, it not only limits the number of features but also transforms them into another space. This new coordinate system is defined by the so-called principal components, which form the axes of this coordinate system. Principal components are determined to maximize variance, ensuring that the first few components are the most informative. By retaining only the first few components, we obtain a projection into a lower-dimensional space. PCA can be easily performed with scikit-learn.

```
from sklearn.decomposition import PCA
pca = PCA().set_output(transform="pandas").fit(data)
data_pca = pca.transform(data)
```

The results may look like in Figure 4.

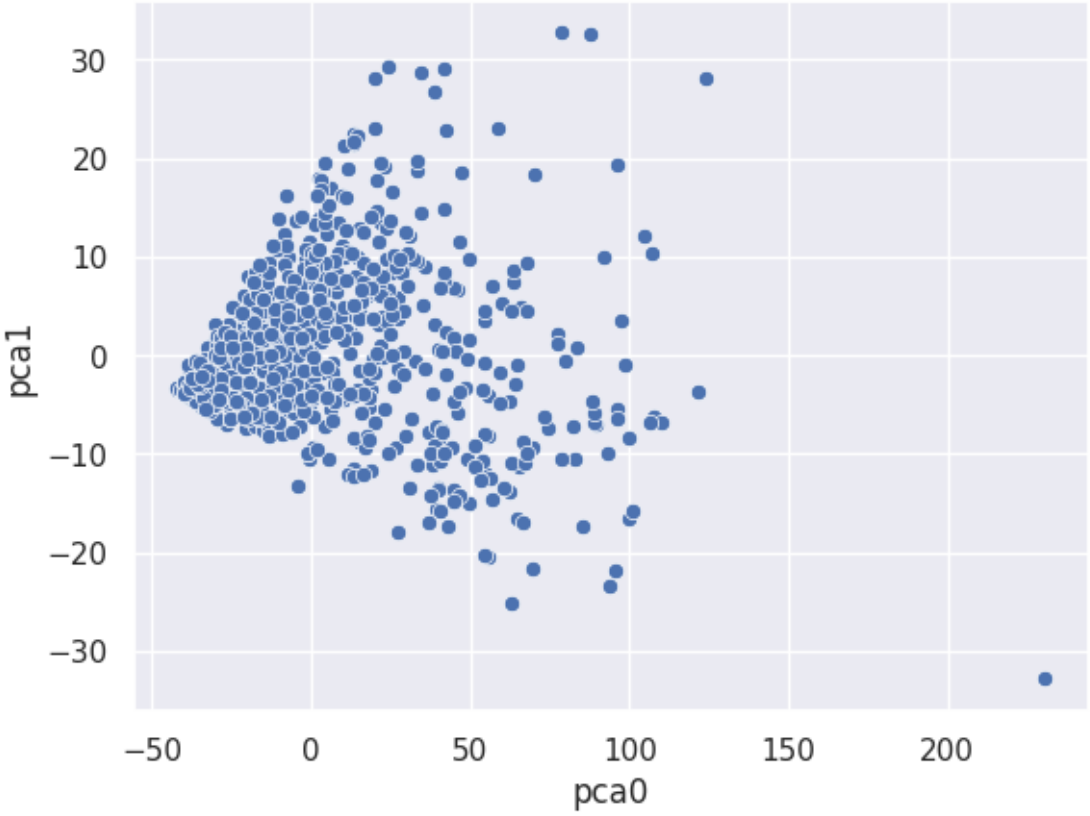

Figure 4 – Example of PCA results.

Task 3.8: Perform PCA. Plot the first two principal components , and then plot the last two components. Explore other combinations as well.

Task 3.9: Apply PCA to the data after feature scaling. How does this impact the PCA results?

Task 3.10: First, use the data with outliers removed. Apply standardization or normalization, and then use PCA. Are there any differences in the results? Did you filter only the selected features or all of them?

Target preparation and encoding

Sometimes, we need to perform additional operations on data before processing it further. Let's assume we want to classify red and white wines. In this case, we should add a type label to the datasets and merge the datasets for red and white wines.

Task 3.11: You have already loaded the red wine dataset. Now, load the white wine data. Prompt the built-in Colab AI generator to add type labels to the datasets and merge them. You should obtain an additional column in your data with labels: *red* or *white*. Ensure the task is completed correctly and save the prompts used.

Task 3.12: Let's further assume we want the dataset to be well-balanced regarding the type of wine. Repeat Task 3.11, ensuring the same number of observations for each wine type in the final dataset. Make sure the task is completed correctly and save the prompts used.

The AI generator probably works well and is easy to use. Your entire Colab notebook and the previously written code provide the context, so the generator's answers can be precise.

If the dataset contains categorical variables, we can use them to analyze the dataset regarding these categories or map them to numerical values. Encoding can be easily done using *OrdinalEncoder* or *OneHotEncoder* from *sklearn.preprocessing*.

```
from sklearn.preprocessing import OrdinalEncoder
ordinal_encoder = OrdinalEncoder()
data['type'] = ordinal_encoder.fit_transform(data[['type']])
```

Task 3.13: Encode label *type* to numerical values.

Since it contains only two values, you can use *OrdinalEncoder* to obtain a binary variable. But what if there were more than two possible types?

Additional tasks

Task 3.14: Create a scatter plot to compare the distribution of unscaled, standardized, and Min-Max scaled data. You may take inspiration from Figure 3.

Task 3.15: Complete all tasks for the white wine dataset.