AGH

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA
W KRAKOWIE

Tutorial 2 - Feature Selection & Extraction

March 23, 2020

Ioannis Manousaridis
Anastasios Tzelepakis
Jonathan Toctaguano Quezada

1. Introduction

1.1 Introduction to Google Colab

Here there is a brief introduction of how to use the code[1] that is written in Google Colab.

1. Follow the instructions strictly. Wrong names would not let you use the code.
2. First of all you need to have a google account.
3. Go to your drive. You can simple click the button “My Drive”.
4. Create a folder there and named **AGH**.
5. Open the folder and create a new folder there named **Pattern Recognition**.
6. Download the iris and the wdbc datasets. The first one is here and the second one here.
7. Place both of the files in the **AGH/Pattern Recogntion** path. The names should be iris.csv and wdbc.csv.
8. Open the code you want to run. We will present an example with **part_1 _wdbc_dataset** code.
9. Click on the file and open it with google colab.

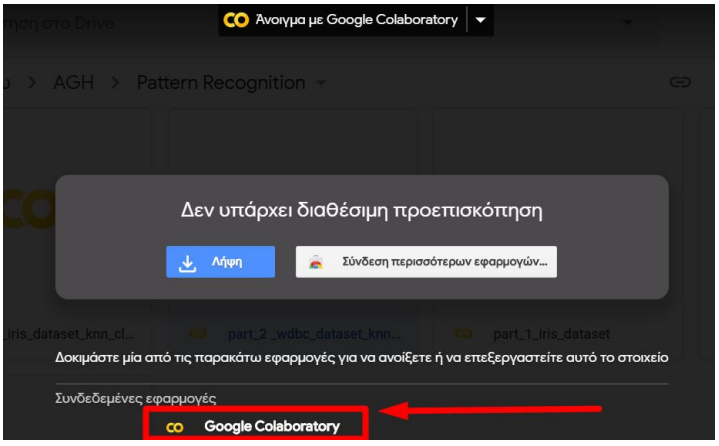

Figure 1: Opening Google Colab.

10. Then you have to give permission to Google to read your files. Just run the code by clicking the play button, go to the link, sign in with your google account, copy the authorization password and go back to the google colab code, place the password in the bar and press Enter. Below is a picture to help you.

[1] All the files can be found here: https://drive.google.com/drive/folders/14S5N30MVjvy0jjyaXh-VXa3gyDbvVHNP?usp=sharing

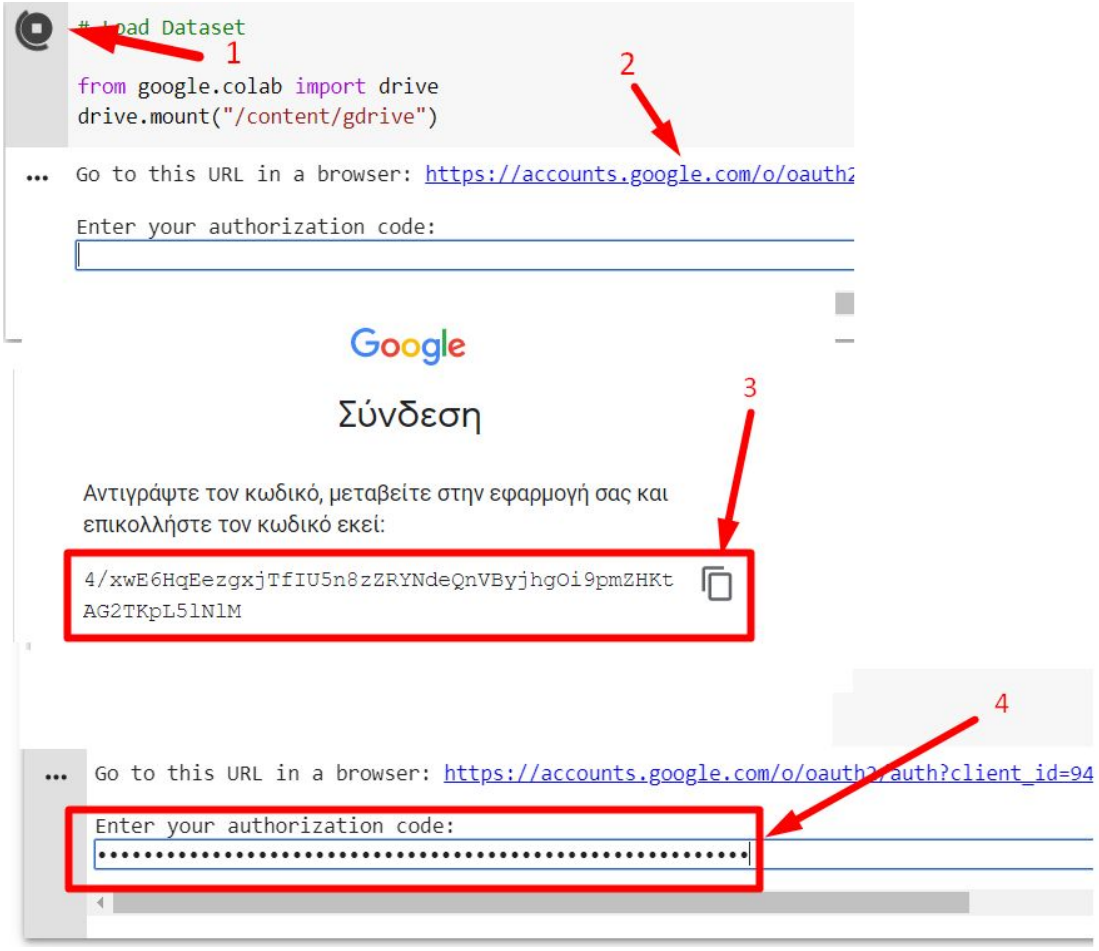

Figure 2: Giving permissions to Google Colab to access your Drive.

11. Now you are ready to run the code.
12. You can run each cell separately.
13. Run the first cell to load the csv file.
14. Then click the code you want run.

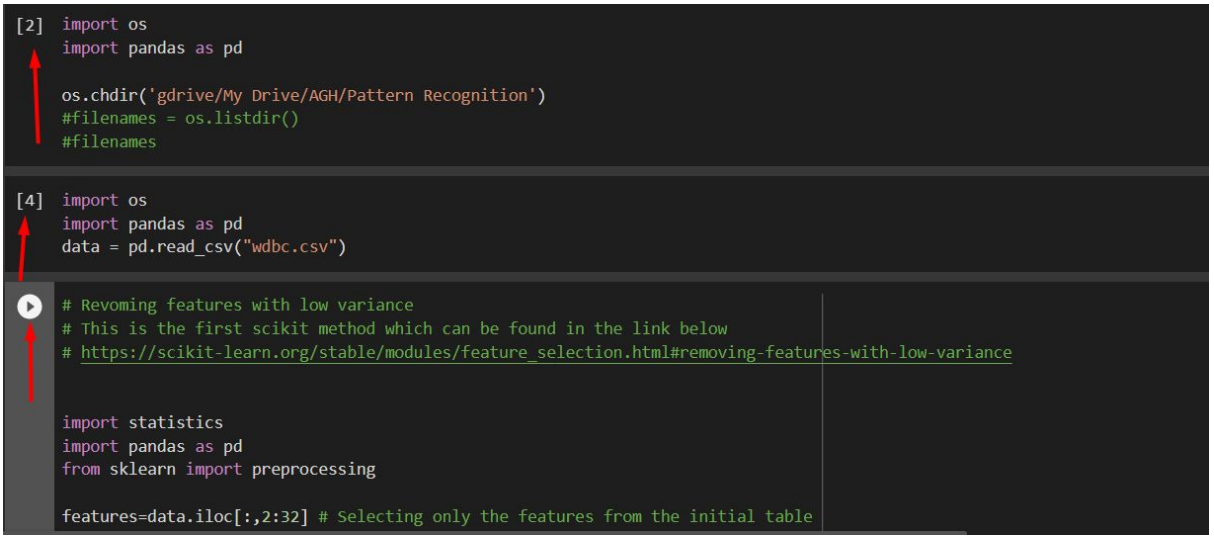

Figure 3: Executing code in cells

15. For every google colab you have to do a similar procedure.
16. Please don't change the code because it will change it to everyone. Either create new cells or you can copy all the files to your drive and work individually from there.

1.2 Structure of the project

In the main folder of pattern recognition the following files can be found:

- The two datasets that are used.
- The report
- All the algorithms.

The naming of the algorithms is in the following format:

Part_X_dataset_algorithms_&_methods_used

in which:

- X is the number of the part
- dataset, either iris or wdbc
- and algorithms_&_methods_used are the methods and the techniques which have been used in this specific code.

2. Datasets

For this tutorial two datasets are going to be used. The first one is the iris flower dataset[2]. The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features are measured from each sample: the length and the width of the sepals and petals, in centimeters. Based on the combination of these four features, it is possible to distinguish the species from each other. This dataset is very simple and it is used in order for the students to get familiar with google colab and some algorithms.

Also, the Breast Cancer Wisconsin (Diagnostic) dataset[3] was used for all the tasks and parts.This dataset consists of 569 instances and 30 attributes. This is the main dataset with which all the tests will be held.

[2] Iris flower dataset: https://en.wikipedia.org/wiki/Iris_flower_data_set
[3] Breast Cancer Wisconsin (Diagnostic) dataset: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

3. Part 1

3.1 Feature extraction and features selection

Feature selection techniques are used for several reasons:

- simplification of models to make them easier to interpret by researchers/users
- shorter training times
- to avoid the curse of dimensionality
- enhanced generalization by reducing overfitting [1]

Feature selection: A choice of the subset of the most informative features.
Feature extraction: New features are created by means of old ones.

3.2 Algorithms of scikit-learn for feature selection.

Below five ways are presented of how feature selection is possible with scikit-learn [3].

3.2.1 Removing features with low variance

Variance Threshold is a simple baseline approach to feature selection. It removes all features whose variance doesn't meet a specific threshold.

As an example, suppose that we have a dataset with boolean features, and we want to remove all features that are either one or zero (on or off) in more than 80% of the samples. Boolean features are Bernoulli random variables, and the variance of such variables is given by

$$Var(X) = p * (1 - p) \qquad (3.1)$$

so we can select using the threshold 0.8 * (1 - 0.8). Example code below:

```
>>> from sklearn.feature_selection import VarianceThreshold
>>> X = [[0, 0, 1], [0, 1, 0], [1, 0, 0], [0, 1, 1], [0, 1, 0], [0, 1,
1]]
>>> sel = VarianceThreshold(threshold=(.8 * (1 - .8)))
>>> sel.fit_transform(X)
array([[0, 1],
       [1, 0],
       [0, 0],
       [1, 1],
       [1, 0],
```

```
       [1, 1]])
```

3.2.2 Univariate feature selection

Univariate feature selection works by selecting the best features based on univariate statistical tests. It can be seen as a preprocessing step to an estimator. Scikit-learn exposes feature selection routines as objects that implement the transform method:

- **SelectKBest** removes all but the highest scoring features.
- **SelectPercentile** removes all but a user-specified highest scoring percentage of features.
- using common univariate statistical tests for each feature: false positive rate **SelectFpr**, false discovery rate **SelectFdr**, or family wise error **SelectFwe**.

- **GenericUnivariateSelect** allows to perform univariate feature selection with a configurable strategy. This allows to select the best univariate selection strategy with hyper-parameter search estimator.

Example code below:

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.feature_selection import SelectKBest
>>> from sklearn.feature_selection import chi2
>>> X, y = load_iris(return_X_y=True)
>>> X.shape
(150, 4)
>>> X_new = SelectKBest(chi2, k=2).fit_transform(X, y)
>>> X_new.shape
(150, 2)
```

3.2.3. Recursive feature elimination

Given an external estimator that assigns weights to features (e.g., the coefficients of a linear model), recursive feature elimination (RFE) is to select features by recursively considering smaller and smaller sets of features. First, the estimator is trained on the initial set of features and the importance of each feature is obtained either through a coef_ attribute or through a feature_importances_ attribute. Then, the least important features are pruned from current set of features.That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached.

3.2.4 Feature selection using SelectFromModel

SelectFromModel is a meta-transformer that can be used along with any estimator that has a coef_ or feature_importances_ attribute after fitting. The features are considered unimportant and removed, if the corresponding coef_ or feature_importances_ values are below the

provided threshold parameter. Apart from specifying the threshold numerically, there are built-in heuristics for finding a threshold using a string argument. Available heuristics are “mean”, “median” and float multiples of these like “0.1*mean”.

3.2.5. Feature selection as part of a pipeline

Feature selection is usually used as a pre-processing step before doing the actual learning. The recommended way to do this in scikit-learn is to use a sklearn.pipeline.Pipeline. Example code below:

```
clf = Pipeline([
  ('feature_selection', SelectFromModel(LinearSVC(penalty="l1"))),
  ('classification', RandomForestClassifier())
])
clf.fit(X, y)
```

3.3 Examples of scikit-learn with datasets

Below there is a presentation of the selection algorithms applied to the two datasets.

3.3.1 Iris dataset code

The code for this part which is needed is named **part_1_iris_dataset**. Two methods were used which are[4] :

3.3.1.1 Removing features with low variance:

The threshold was set to 0.06 and the weights of the features can been seen in the Figure 4.

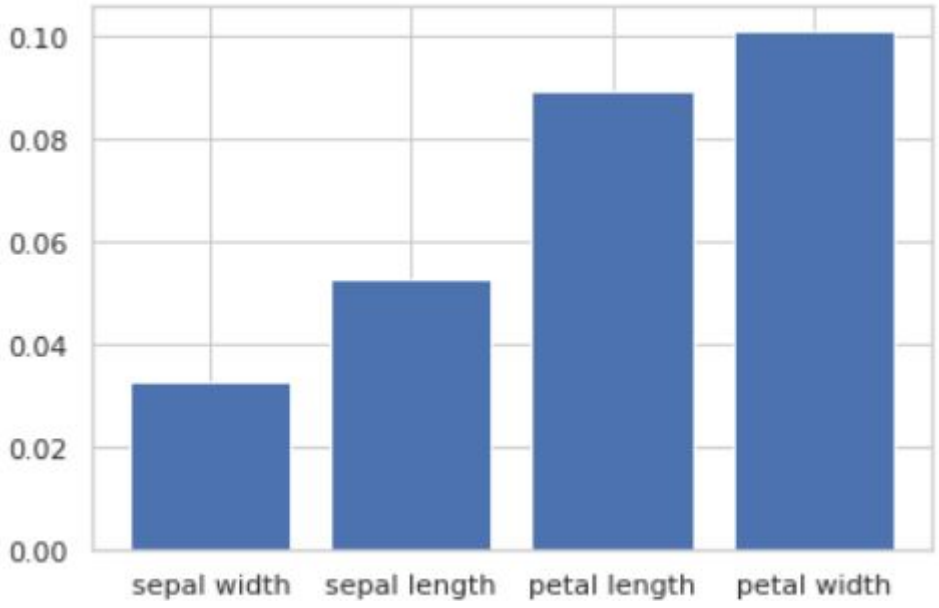

Figure 4: Ordered histogram of the weights of individual features for iris dataset using the Revoming features with low variance method.

[4] Iris code here: https://colab.research.google.com/drive/1Svw6fcQzAKoeFNhOT-oc1kfGrxfdmcj4#scrollTo=dYTRxewwMfxM

3.3.1.2 Univariate feature selection

For this method the SelectKBest algorithm was used. The function that was used to determine the scores of its feature is the chi2. The values of the scores for the attributes can be seen in the figure 5.

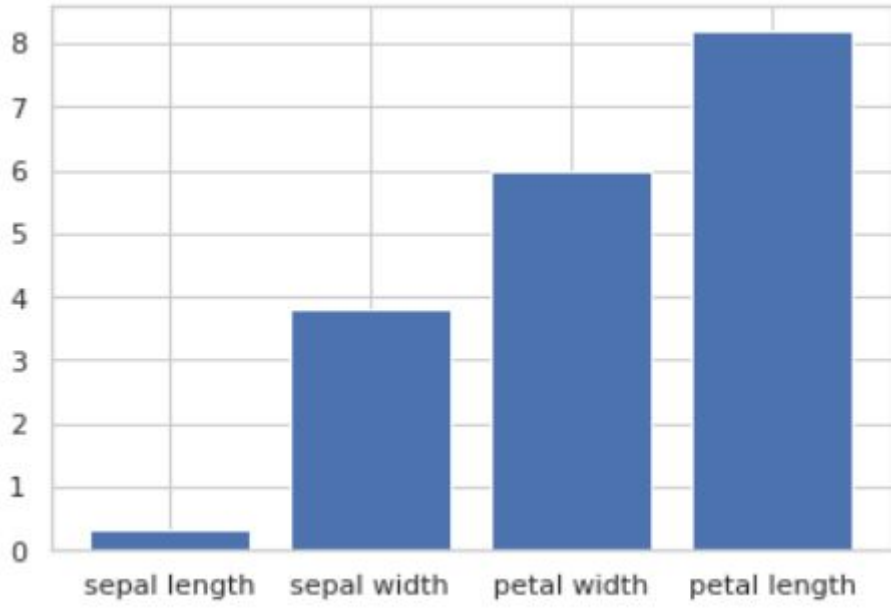

Figure 5: Ordered histogram of the weights of individual features for iris dataset using the Univariate feature selection method.

3.3.2 Wdbc dataset

The code for this part which is needed is named **part_1 _wdbc_dataset**. Two methods were used which are:

3.3.1.1 Removing features with low variance:

The threshold was set to 0.025 and the weights of the features can been seen in the Figure 6.

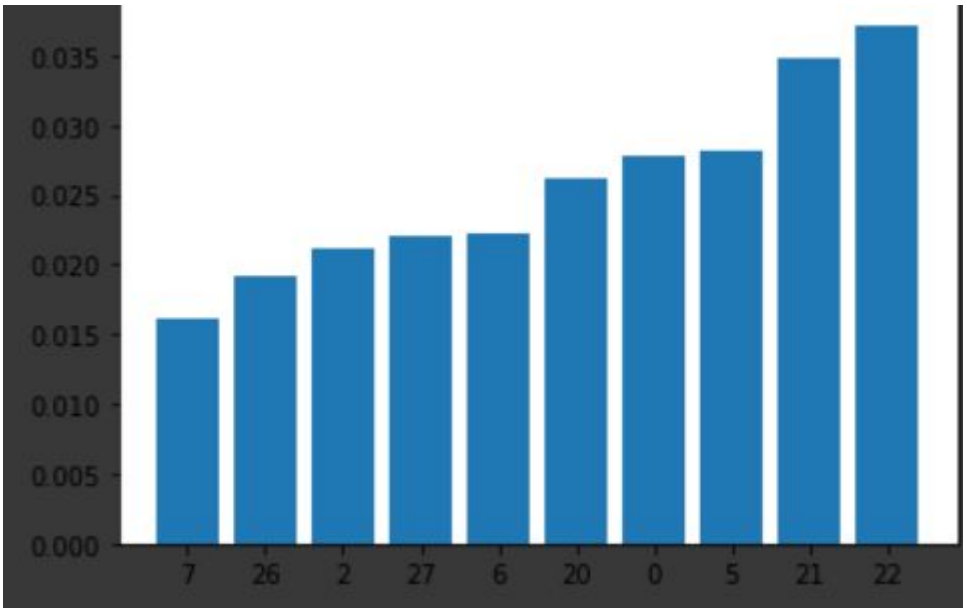

Figure 6: Ordered histogram of the weights of individual features for wdbc dataset using the revoming features with low variance method.

3.3.1.2 Univariate feature selection

For this method the SelectKBest algorithm was used. The function that was used to determine the scores of its feature is the chi2. The values of the scores for the attributes can be seen in the figure 7.

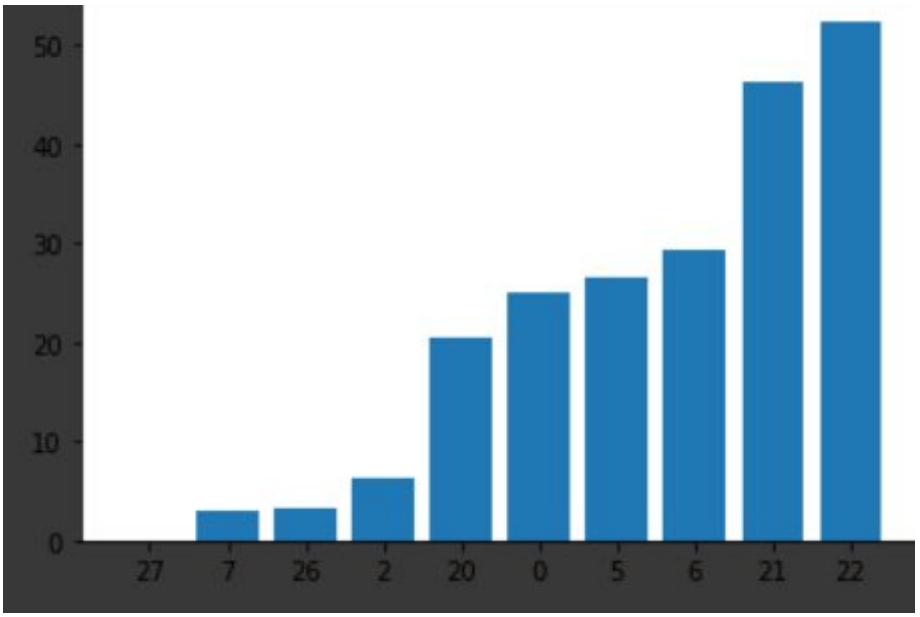

Figure 7: Ordered histogram of the weights of individual features for wdbc dataset using the Univariate feature selection method.

4. Part 2

In this part the students should complete the table: (k = 1,3,5; m = N, 5,2) for the two datasets. However the iris dataset has only 4 attributes so the table has to be altered to (k=1,3,5; m=N,3,2).

4.1 Iris dataset

For this part the **part_2_iris_dataset_knn_classifier** code has to used. In the code is possible to determine the number of neighbours and the number of best features to be kept. In the end the train and the test score are printed.

4.1 Wdbc dataset

For this part the **part_2_wdbc_dataset_knn_classifier** code has to used. In the code is possible to determine the number of neighbours and the number of best features to be kept. In the end the train and the test score are printed.

5. Part 3

5.1 PCA - presentation

"Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors (each being a linear combination of the variables and containing n observations) are an uncorrelated orthogonal basis set. PCA is sensitive to the relative scaling of the original variables [4]*."*

5.2 PCA - example

Here is an example code in python which uses the PCA transformation.

```
from sklearn.decomposition import PCA

# Defining PCA as our Feature Selection Model.The  variable
# n_components determines the number of PCA components
```

```
pca = PCA(n_components=2)

# Applying PCA to our Data, where X is the Dataset
principalComponents = pca.fit_transform(X)
```

This algorithm will transform the features from figure 9 to the figure 10, in which only two features exists.

	ID	Diagnosis	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28	V29	V30
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	1.0950	0.9053	8.589	153.40	0.006399	0.04904	0.05373	0.01587	0.03003	0.006193	25.380	17.33	184.60	2019.0	0.16220	0.66560	0.7119	0.2654	0.4601	0.11890
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	0.5435	0.7339	3.398	74.08	0.005225	0.01308	0.01860	0.01340	0.01389	0.003532	24.990	23.41	158.80	1956.0	0.12380	0.18660	0.2416	0.1860	0.2750	0.08902
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	0.7456	0.7869	4.585	94.03	0.006150	0.04006	0.03832	0.02058	0.02250	0.004571	23.570	25.53	152.50	1709.0	0.14440	0.42450	0.4504	0.2430	0.3613	0.08758
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	0.4956	1.1560	3.445	27.23	0.009110	0.07458	0.05661	0.01867	0.05963	0.009208	14.910	26.50	98.87	567.7	0.20980	0.86630	0.6869	0.2575	0.6638	0.17300
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	0.7572	0.7813	5.438	94.44	0.011490	0.02461	0.05688	0.01885	0.01756	0.005115	22.540	16.67	152.20	1575.0	0.13740	0.20500	0.4000	0.1625	0.2364	0.07678
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	1.1760	1.2560	7.673	158.70	0.010300	0.02891	0.05198	0.02454	0.01114	0.004239	25.450	26.40	166.10	2027.0	0.14100	0.21130	0.4107	0.2216	0.2060	0.07115
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	0.7655	2.4630	5.203	99.04	0.005769	0.02423	0.03950	0.01678	0.01898	0.002498	23.690	38.25	155.00	1731.0	0.11660	0.19220	0.3215	0.1628	0.2572	0.06637
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	0.4564	1.0750	3.425	48.55	0.005903	0.03731	0.04730	0.01557	0.01318	0.003892	18.980	34.12	126.70	1124.0	0.11390	0.30940	0.3403	0.1418	0.2218	0.07820
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	0.7260	1.5950	5.772	86.22	0.006522	0.06158	0.07117	0.01664	0.02324	0.006185	25.740	39.42	184.60	1821.0	0.16500	0.86810	0.9387	0.2650	0.4087	0.12400
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	0.3857	1.4280	2.548	19.15	0.007189	0.00466	0.00000	0.00000	0.02676	0.002783	9.456	30.37	59.16	268.6	0.08996	0.06444	0.0000	0.0000	0.2871	0.07039

569 rows × 32 columns

Figure 9: Original data with all attributes.

	PCA1	PCA2	Diagnosis
0	1.387021	0.426895	M
1	0.462308	-0.556947	M
2	0.954621	-0.109701	M
3	1.000816	1.525089	M
4	0.626828	-0.302471	M
...	...	...	...
564	1.002840	-0.474785	M
565	0.620757	-0.517200	M
566	0.226311	-0.287946	M
567	1.677834	0.335946	M
568	-0.905068	-0.104109	B

569 rows × 3 columns

Figure 10: Data for two principal components.

5.3 Mathematical approach of PCA procedure

In this chapter a mathematical example of PCA will be presented. For simplicity reasons the dataset in figure 11 will be used. The procedure is the same for more complicated datasets [5].

	X	Y	Z
X1	1	0	-1
X2	0	1	-1
X3	-1	1	0
X4	0	-1	1
X5	-1	0	1
X6	1	-1	0

Figure 11: The dataset for the mathematical example of PCA

First of all, the matrix $\mathbf{X^T X}$ needs to be defined.

$$X = [x_1 \;\; x_2 \;\; \dots \;\; x_n]^T = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix}$$

$$X^T X = \begin{bmatrix} 1 & 0 & -1 & 0 & -1 & 1 \\ 0 & 1 & 1 & -1 & 0 & -1 \\ -1 & -1 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 4 & -2 & -2 \\ -2 & 4 & -2 \\ -2 & -2 & 4 \end{bmatrix}$$

After calculating the matrix X^TX, the eigenvalues have to be found. The eigenvalues are the roots of the characteristic polynomial.

$$|X^T X - \lambda I\ | = 0 \Longrightarrow \begin{vmatrix} 4-\lambda & -2 & -2 \\ -2 & 4-\lambda & -2 \\ -2 & -2 & 4-\lambda \end{vmatrix} = 0 \Longrightarrow (\lambda - 6)\cdot\lambda\cdot(6-\lambda) = 0$$

The roots are $\boldsymbol{\lambda_{1,2} = 6}$, $\boldsymbol{\lambda_3 = 0}$. From these values only the highest values are kept. Then the eigenvectors have to be found for these values. For $\mathbf{u_1}$:

$$(X^T X - \lambda_1 I_3)\underline{u_1} = 0 \Longrightarrow \begin{bmatrix} -2 & -2 & -2 \\ -2 & -2 & -2 \\ -2 & -2 & -2 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = 0 \Longrightarrow x + y + z = 0 \Longrightarrow z = -x - y$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = x\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + y\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + z\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \overset{(4)}{\Rightarrow} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = x\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + y\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} - (x+y)\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ z \end{bmatrix} = x\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} + y\begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}$$

This equation gives a big family of eigenvectors. The norm should be 1 and thus as $\mathbf{u_1}$ is selected the following:

$$\underline{u_1} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

The two eigenvectors should be vertical:

$$\underline{u_1} \perp \underline{u_2} \Longrightarrow \underline{u_2}^T\underline{u_1} = 0 \Longrightarrow \frac{1}{\sqrt{2}}[x \quad y \quad z]\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} = 0 \Longrightarrow x - z = 0 \Longrightarrow x = z$$

Also, $z = -x - y$. That's why : $y = -2x$. Finally:

$$\underline{u_2} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x \\ -2x \\ x \end{bmatrix} \overset{x=1}{\Longrightarrow} \underline{u_2} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \Longrightarrow \underline{u_2} = \frac{1}{\sqrt{6}}\begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

The final components are calculated by the following formula:

$$x_i' = \begin{bmatrix} \underline{u_1}^T x_i \\ \underline{u_2}^T x_i \end{bmatrix}$$

$$x_1' = \begin{bmatrix} \frac{2}{\sqrt{2}} \\ 0 \end{bmatrix}, x_2' = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{3}{\sqrt{6}} \end{bmatrix}, x_3' = \begin{bmatrix} \frac{-1}{\sqrt{2}} \\ -\frac{3}{\sqrt{6}} \end{bmatrix}, x_4' = \begin{bmatrix} \frac{-1}{\sqrt{2}} \\ \frac{3}{\sqrt{6}} \end{bmatrix}, x_5' = \begin{bmatrix} -2\sqrt{2} \\ 0 \end{bmatrix}, x_6' = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{3}{\sqrt{6}} \end{bmatrix}$$

Also, it should be noted that the procedure of PCA is not lossless. The lost information can be calculated by the following formula.

$$Lost\ information = \frac{\sum Not\ used\ eigenvalues}{\sum All\ eigenvalues} = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3}$$

In this example $\lambda_1 = 0$ and thus the loss of information is zero.

5.4 Bar Plots of the eigenvalues

In the bar plots that are in figures 12 and 13 the eigenvalues of the PCAs are presented.

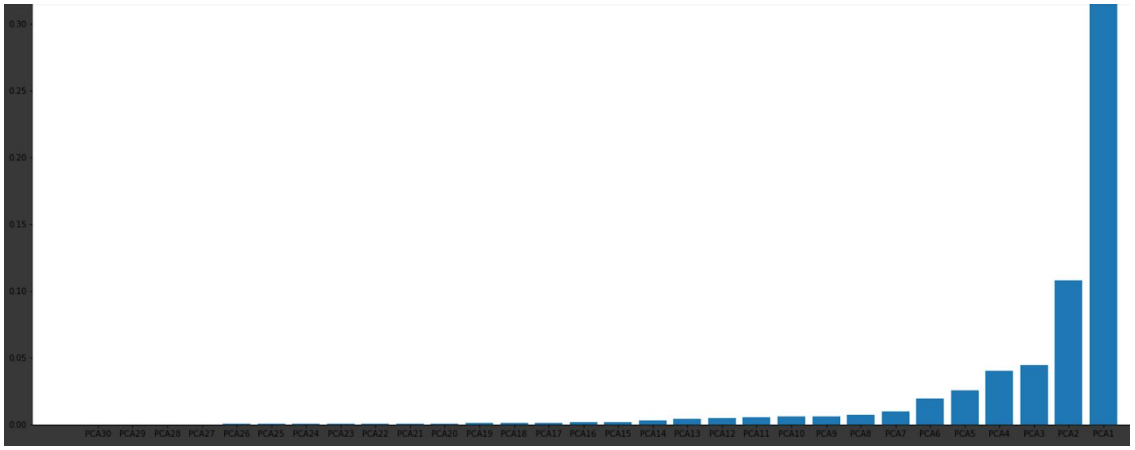

Figure 12: Eigenvalues of all principal components.

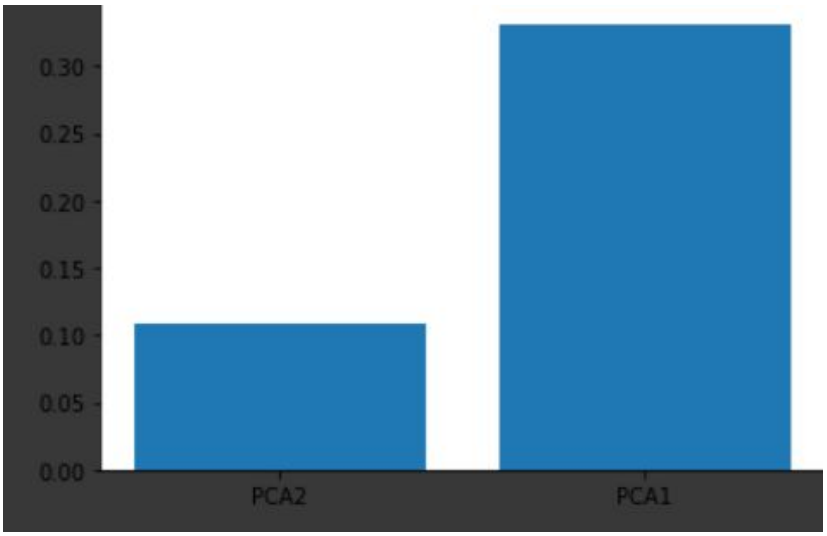

Figure 13: The eigenvalues for the two most important principal components.

In the PCA method for two components, only the two highest PCA values are kept. The rest are eliminated.

6. Part - 5

In this part the students should complete the table: (k = 1,3,5; m = N, 5,2) for the two datasets. However the iris dataset has only 4 attributes so the table has to be altered to (k=1,3,5; m=N,3,2).

6.1 Iris dataset

For this part the **part_5_6_Iris_dataset_PCA_and_knn_classifier** code has to used. In the code is possible to determine the number of neighbours and the number of best features to be kept. In the end the train and the test score are printed.

6.2 Wdbc dataset

For this part the **part_5_6_wdbc_dataset_PCA_and_knn_classifier** code has to used. In the code is possible to determine the number of neighbours and the number of best features to be kept. In the end the train and the test score are printed.

7. Part - 6

7.1 Visualization of data and outliers

In order for the outliers to be calculated and eliminated, the Z-score function from the library stats it is used. This function works in the following way:

- For each column, first it computes the Z-score of each value in the column, relative to the column mean and standard deviation.
- Then is takes the absolute of Z-score because the direction does not matter, only if it is below the threshold.
- all(axis=1) ensures that for each row, all column satisfy the constraint.
- Finally, result of this condition is used to index the dataframe.

In the figure 14, a comparison with the help of visualization is being held. On the upper left side, it is the scatter plot of the iris set with the outliers and on the upper right side it is the same but without the outliers. The difference is clear. In the first picture with the outliers, the x-axis ranges from -0.8 to 0.8 and in the y-axis from -0.5 to 0.5. By eliminating the outliers the x-axis ranges from -0.6 to 0.6 and in the y-axis from -0.25 to 0.25.

Regarding the wdbc dataset, on the down left side there is a scatter plot with outliers and on the down left side without the outliers. In the first occasion the x-axis ranges from -1.0 to 2.0

and the y-axis from -1.0 to 1.5. By eliminating the outliers the x-axis ranges from -1.0 to 1.0 and the y-axis from -1.0 to 0.6.

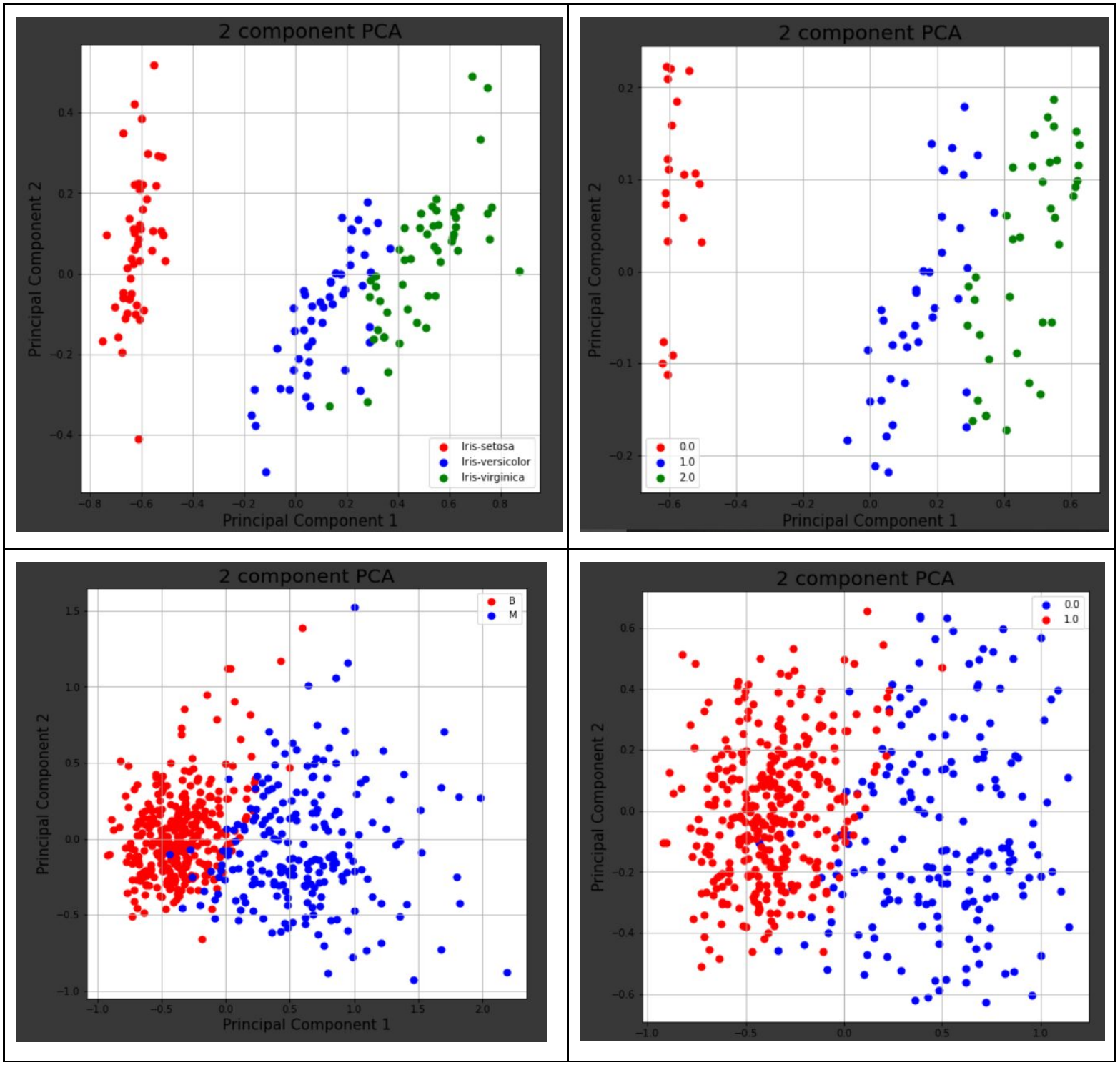

FIgure 14: The scatter plots for the two datasets with outliers on the left and without on the right. The upper plots are for the iris flower dataset and the down for the wdbc dataset.

8. Bibliography

[1] Gareth James; Daniela Witten; Trevor Hastie; Robert Tibshirani (2013). An Introduction to Statistical Learning. Springer. p. 204.

[2] Feature selection and extraction, Witold Dzwinel, March 2020, Lecture notes in Pattern Recognition and Machine learning

[3] Feature Selection, Scikit Learn, 2019 https://scikit-learn.org/stable/modules/feature_selection.htm

[4] Pearson, K. (1901). "On Lines and Planes of Closest Fit to Systems of Points in Space". Philosophical Magazine. 2 (11): 559–572

[5] Preprocessing, PCA, ISOMAP, A. Simeonidis, 2020, Lectures for Pattern Recognition .