Second Deliverable Project-ADM (ALGORITHMICS FOR DATA MINING)

Data Visualization of Italian wine constituents using freely available datasets of scikit-learn library in python

Author: Meysam Zamani Forooshani

Master MIRI Data Science, FIB, BarcelonaTech, UPC
ALGORITHMICS FOR DATA MINING(ADM)
April 27, 2019

Abstract

The present project is related to the dimensionality reduction. This allows to have a better understanding of the data, to perform visualizations and to save running time and memory during their treatment. At the end we will highlight the results of an analysis of 178 wines (instances) all grown in the same region in Italy, but from 3 different cultivars (3 different classes). Higher-quality wines were obtained in the years characterized by a reduction in rainfall and high temperature patterns. Teleconnections were also detected in different periods of the growing season, thus allowing for the development of wine-quality forecasting tools.[reference 1]

Principal component analysis was performed on color variables, and their correlations were discussed. The results have evidenced the important role played by the compounds since they seem to be the main contributors to the red color of these aged wines, and they have been demonstrated not to increase their yellow nuances. On the other hand, results highlight that, regardless of the time of aging, wines presenting more red and lower yellow nuances as well as a darker color are in general evaluated with higher quality scores by wine experts.[reference 2]

Wine is a complex matrix composed of water, ethanol and a variety of chemical compounds such as peptides, proteins, carbohydrates, thiols, and phenolic compounds. The latter ones can be classified into flavonoids (flavanols, flavonols, dihydroflavonols, and anthocyanins) and non-flavonoids (phenolic acids, phenols, and stilbenes). Anthocyanins and flavanols are particularly abundant in grape and wine and are essential to wine quality. Indeed, anthocyanins are the red pigments of grapes and are responsible for the color of red wines,whereas flavanols contribute to taste (especially astringency and bitterness). Due to the complexity of wine data obtained by usual instrumental techniques, it is not possible to resolve or quantify all the chemical constituents present in wine. Therefore, the combination of these techniques with analysis can reveal latent patterns in the data, which may enable classification of the samples in terms of varietal, geographical origin, aging, adulteration, etc[reference 3].

Nevertheless, to our knowledge, there are no reports of wine classification. In this work I decide to select a famous open-access dataset named Wine. It's a free-access dataset that we can find in the datasets offered by the scikit-learn library in python [reference 4].

The purpose of the present work is essential to reduce the dimension of the data by identifying the most relevant dimensions. This is called dimensionality reduction. The results of this allows to have a better understanding of the data, to perform visualizations and to save running time and memory during their treatment.

Key Words: dimensionality reduction, Data Visualization, wine constituents, wine classification, Principal Component Analysis, 2D visualization, 3D visualization.

1 Introduction

As the dimensions of data increases, the difficulty to visualize it and perform computations on also increases. When dealing with large dataset in statistics or classification for example, it is then essential to reduce the dimension of the data by identifying the most relevant dimensions. This is called dimensionality reduction. This allows to have a better understanding of the data, to perform visualizations and to save running time and memory during their treatment.

The main objectives of my project are to use reduction technics for a given dataset, to visualize these data in 3D or 2D and see how the reduction affects the quality of the classiffication of the data.

In section [2], the dataset I have chosen is described. For the cIassiffication, I use two different classifiers , both from the scikit-learn python library:

- *the KNeighbors classifier*
- *the Decision Tree classifier*

The classification scores obtained with these classifiers on the full dataset are reported in section [3].

In sections [4] and [5] I report, analyze and comments the results obtained by applying three popular dimensionality reduction methods:

- *the Principal Component Analysis*
- *the Kernel Principal Component Analysis*
- *the Linear Discriminant Analysis.*

The python code I wrote to obtain all these results and visualizations is reported in section [7] and also is available in my Google drive by link which is provided in references[reference 5].

2 Data sets Description and class definition

I decide to select a famous open-access dataset named Wine. It's a free-access dataset that we can find in the datasets offered by the scikit-learn library in python.

These data are the results of an analysis of 178 wines (instances) all grown in the same region in Italy, but from 3 different cultivars. The analysis is determined from the quantities of 13 constituents that are found in each wine (the 13 features). This dataset is then composed by 178 instances for a total of 3 different classes with the following repartition :

- Class 1 : 59 instances.
- Class 2 : 71 instances.
- Class 3 : 48 instances.

Since the instances are represented by 13 features, then 13 dimensions, a direct visualization is of course not possible. I will then use the dimensionality reduction methods as a feature selection procedures to reduce the dimension to 3 or 2 in section [4] and [5].

3 Classification on the dataset And constituents Analysis

In order to compare the effciency of the PCA and LDA feature reduction technics, I first apply the two classifiers on the dataset with the 13 features to obtain their scores. The KNeighbors classifier associates a class to a given instance depending on the class of the closest instances: the neighbors. The Tree classifier associates the class to a given instance after performing tests based on the values of the features at each nodes of a tree.

classifier 1: I use the functions KNeighborsClassifier and DecisionTreeClassifier from the scikit-learn library. I also use the function GridSearch to find the best parameters for each classifier, and a cross validation function to get reliable results for the 13 features set. This is done in my function scores with all features.

classifier 2: After running the function many times, it seems that the best parameter for the KNeighbors classifier is to consider as neighbor only the closest point(n neighbors=1) for a result of 72.5% of instances correctly classified. For the tree classifier I tried to find the best maximal depth of the tree, but the differences I observed between all the runs are not relevant. Therefore, I decided to use trees with a maximal depth equal to 8. In that case, 87% of instances are correctly classified.

classifier 3: Other parameters could be optimized for each classifiers (distance in KNeighbor, splitting strategy of nodes in tree, etc...), in order to obtain better scores, but this is not the aim of the project.

4 Results of Principal Component Analysis

One of the most popular methods to reduce the dimension is the PCA technic. The basic ideas are to remove the redundant dimensions and keep only the directions that capture most of the variance. This is done by using the covariance matrix of the data. Roughly speaking, the vector basis where this matrix is diagonal is found, and only the most important components are retained in this basis to reduce the dataset. Since the technical details and analysis of the PCA can be found in many references [reference 6], I only reports here the main steps:

- Calculate the covariance matrix of the data points (here size 13 * 178)
- Compute singular values and singular vectors of the matrix using the Singular Value Decomposition.
- Choose the first k vectors that will be the new k features.
- Transform the original data (13 features and 178 instances here) into the new data set composed by k new features and 178 instances

The singular values and vectors are generalization of eigen values and vectors for non-diagonalizable matrix. PCA can be done using eigen decomposition only, but the function PCA I use from scikit-learn is based on SVD.

I first apply the PCA reduction to visualize the dataset in 3D, so I choose to keep the most k = 3 relevant features. The results are presented on figure 1 and obtained with my function pca 3D visualization. We can observed that the first class (red) is better separated from the others, than second (green) and third (blue).

In figure 2 the dimension is reduced with PCA to k = 2 to observe the dataset in 2D using my function pca 2D visualization. As previously seen in 3D plots, the PCA allows to separate clearly the instances corresponding to class 1 (red).

It is interesting now to compute the scores obtained by the two classifiers on the reduced data sets. I report in table[1] the scores obtained for different PCA reduction sizes:

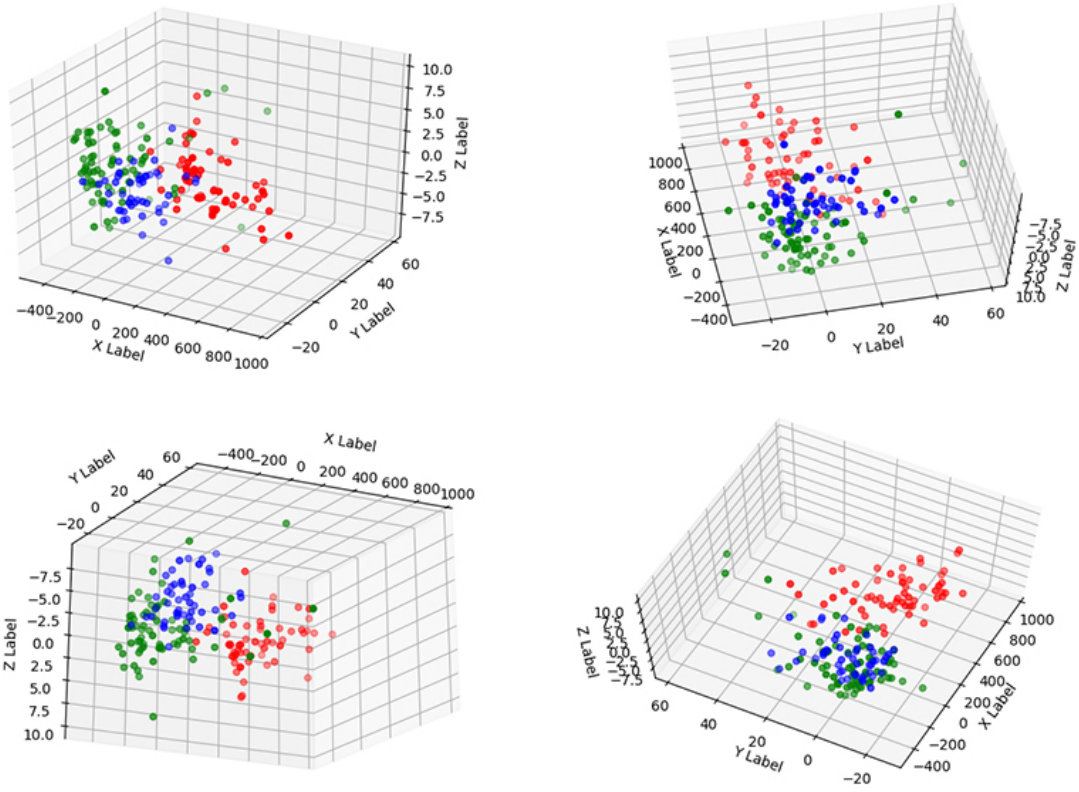

Figure 1: 3D visualization of the dataset after PCA feature reduction. Class 1:red, Class 2: green, Class 3: blue

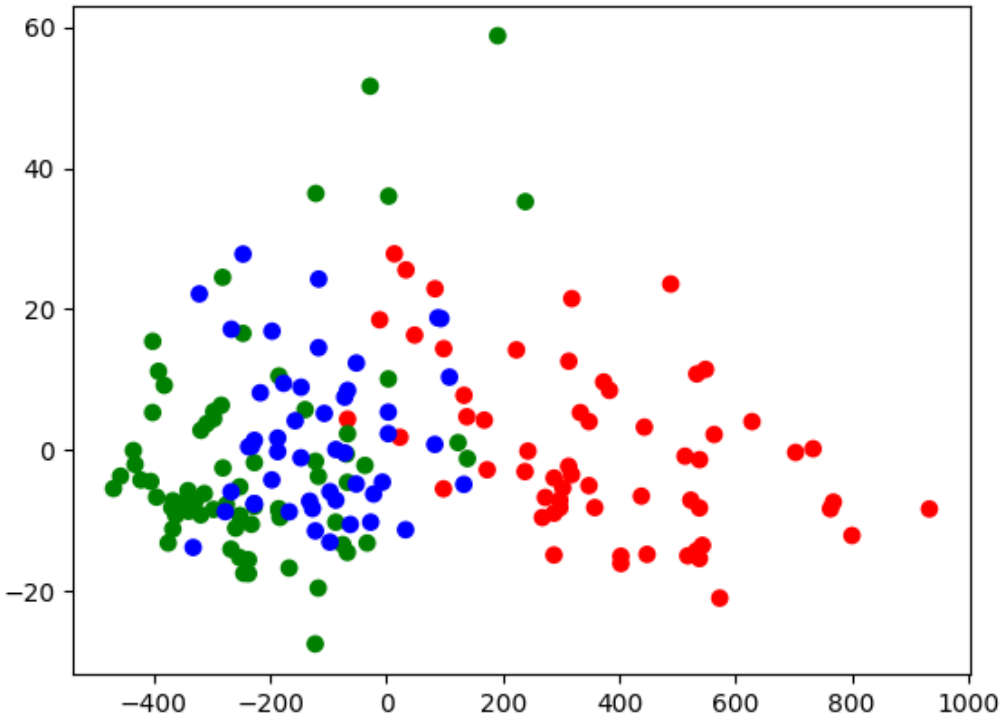

Figure 2: 2D visualization of the dataset after PCA feature reduction. Class 1:red, Class 2: green, Class 3: blue

PCA	k=2	k=3	k=5	k=7	k=9	full
KNeighbors	70%	70%	72%	72%	72%	72.5%
Decision Tree	71%	76%	86%	83%	83%	87%

Table 1: the scores obtained for different PCA reduction sizes

It is interesting to note that for the KNeighbors classifier, the score are mostly independent of the reduction size. With the decision tree classifier the scores are significantly decreased for large reduction, k = 2, 3. But for k $\geq$ 5 the scores are mostly the same than for the full dataset with 13 features. It means that reduction of the dimension of the problem by a factor 2 using PCA method will not significantly affects the quality of the results in that case. It will of course reduces the computational effort.

I also use Kernel-PCA which is an extension of PCA which achieves non- linear dimensionality reduction. The graphical results are presented on figure 3 for k = 2; 3. For this dataset, the classification scores compared to PCA are not significantly improved as seen in table[2]:

Kernel PCA	k=2	k=3	k=5	k=7	k=9	full
KNeighbors	70%	70%	72.5%	72.5%	72.5%	72.5%
Decision Tree	71%	76%	86%	85%	84%	87%

Table 2: the scores obtained for different Kernel-PCA reduction sizes

Nevertheless once again, for k $\geq$ 5 the scores are the same than for the full 13 features.

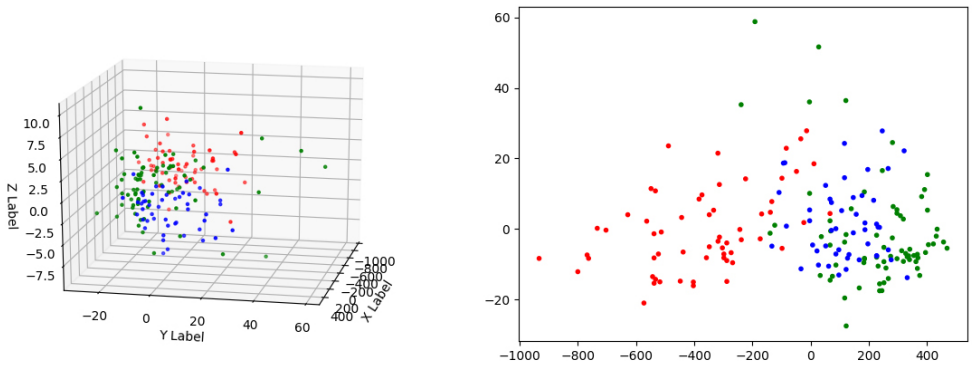

Figure 3: 3D and 2D visualizations of the dataset after Kernel PCA feature reduction. Class 1: red, Class 2: green, Class 3: blue

5 Results of Linear Discriminant Analysis

The LDA method can be used as a classifier or a dimension reduction algorithm[reference 7]. In this project I use LDA only for reduction and apply the two previous KNeighbors and tree classifier for consistency with the previous sections.

While the PCA works only with the features to find the dimensions that maximize the whole variance of the data set, the LDA introduces also the classes. Instead of computing the full covariance matrix, LDA computes the within-class and between-class covariance matrices. All these matrices are taken into account to find the best directions for separating the classes. This is done also through computing eigenvalues and eigenvectors and retaining the most relevants. One constraint is that the dimension of the new dataset is at most k = NbClasse - 1. In my case, the dimension after LDA is then k = 3 - 1 = 2.

I therefore apply the LDA reduction to visualize the dataset in 2D. The results obtained from my function lda 2D visualization are represented on figure 4. The color for each class are the same as in the PCA section.

These results are excellent, since we observe a clear separation between every classes and a real structure in the data. The classification scores are outstanding since both classifiers achieved 99% of data correctly classified! It's much better than the scores with all the features, and means that, in order to obtain better classification, LDA method can be a very good option for many problems.

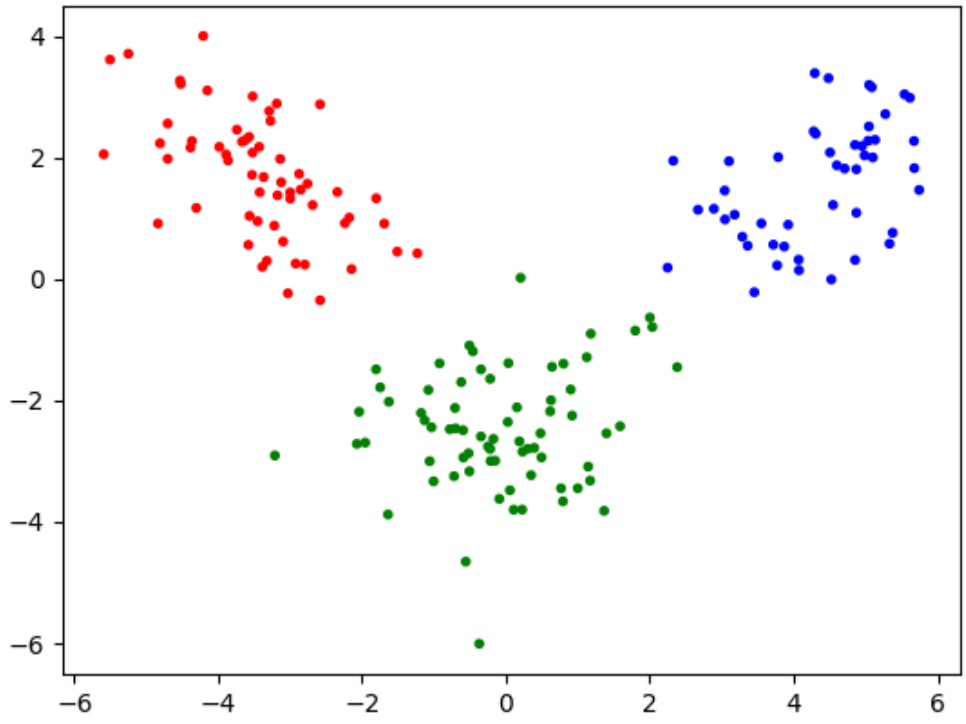

Figure 4: 2D visualization of the dataset after LDA feature reduction. Class 1: red, Class 2: green, Class 3: blue

6 Conclusion

In this project I perform dimensional reductions on a data set with 13 dimensions (features) in order to visualize the data in 2D and 3D. I use three popular methods: the PCA, the Kernel PCA and the LDA. I also evaluate the influence of the dimension reduction on the classification scores using two different classifiers, the KNeighours and the Tree classifier. The results are very interesting in my opinion. For the PCA and Kernel-PCA, the best classification scores for these classifiers can be obtained using only 5 dimensions instead of 13. Nevertheless no gain in score is observed.

For the LDA the results are outstanding since with only 2 dimensions, the separation between classes is clearly observed and the classification score is optimal (99%)! This will not be the case for general dataset of course but this method seems to be very powerfull.

As a conclusion, through this project I realized in practice the importance of the dimensional reduction of dataset, not only for visualization, but also for improving the treatment we want to do on the data.also for being more familiar to the topic,I have read another paper which i put it in my references[reference 8].

References

[1] Analysis of Italian Wine Quality Using Freely Available Meteorological Information, *American Journal of Enology and Viticulture, 57(3) Grifoni, D., Mancini, M., Maracchi, G., Orlandini, S., & Zipoli, G. (2006).*

[2] Pigment composition and color parameters of commercial Spanish red wine samples: linkage to quality perception, *European Food Research and Technology,* 232(5), 877887. *Senz-Navajas, M.-P., Echavarri, F., Ferreira, V., & Fernndez-Zurbano, P. (2011).*

[3] Exploration of liquid chromatographic-diode array data for Argentinean wines by extended multivariate curve resolution, *Chemometrics and Intelligent Laboratory Systems, Pisano, P. L., Silva, M. F., & Olivieri, A. C. (2014).*

[4] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html#sklearn.datasets.loa_wine

[5] The Python code is available in section [7] and also you can find it as a file in provided link as bellow: *https://drive.google.com/file/d/1xN6xkag6QZVtFJUVmejw6IrpmD6al8oe/view?usp=sharing*

[6] https://en.wikipedia.org/wiki/Principal_component_analysis

[7] https://en.wikipedia.org/wiki/Linear_discriminant_analysis

[8] The Graphical Feature Extraction of Star Plot for Wine Quality Classification, *First International Conference on Pervasive Computing, Signal Processing and Applications, Jing, L., Jin-Jia, W., Tao, Z., Chong-Xiao, M., & WenXue, H. (2010).*

7 The python code

```
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.decomposition import FastICA
from sklearn.decomposition import KernelPCA

from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn import datasets

from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import matplotlib
import numpy as np

def pca_3D_visualisation():
wine=datasets.load_wine()
pca = PCA(n_components=3)
wine_pca=pca.fit_transform(wine.data)
x=[]
y=[]
z=[]
label=[]
label.append(wine.target)
colors=['red', 'green', 'blue']
for i in wine_pca:
x.append(i[0])
y.append(i[1])
z.append(i[2])
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(x, y, z, c=label, marker='o',cmap=matplotlib.colors.ListedColormap
(colors))
ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')
plt.show()
def pca_2D_visualisation():
wine=datasets.load_wine()
pca = PCA(n_components=3)
wine_pca=pca.fit_transform(wine.data, wine.target)
x=[]
y=[]
#z=[]
label=[]
label.append(wine.target)
colors=['red', 'green', 'blue']
for i in wine_pca:
x.append(i[0])
y.append(i[1])
#z.append(i[2])
plt.scatter(x, y, c=label, marker='.', cmap=matplotlib.colors.ListedColormap
(colors))
plt.show()
```

```
def get_best_classification(tree_param, kneig_param):
wine = datasets.load_wine()
tree=DecisionTreeClassifier()
kneig=KNeighborsClassifier()
grid_tree=GridSearchCV(tree, tree_param, cv=10)
grid_kneig=GridSearchCV(kneig, kneig_param, cv=10)
grid_tree.fit(wine.data, wine.target)
grid_kneig.fit(wine.data, wine.target)
results={'tree best param':grid_tree.best_params_, '
tree best score':grid_tree.best_score_, 'kneig best param':
grid_kneig.best_params_, 'kneig best score': grid_kneig.best_score_}
return(results)

def scores_with_all_features(tree_param, kneig_param):
wine = datasets.load_wine()
tree = DecisionTreeClassifier()
kneig = KNeighborsClassifier()
grid_tree = GridSearchCV(tree, tree_param, cv=5)
grid_kneig = GridSearchCV(kneig, kneig_param, cv=5)
grid_tree.fit(wine.data, wine.target)
grid_kneig.fit(wine.data, wine.target)
tree = DecisionTreeClassifier(max_depth=grid_tree.best_params_['max_depth'])
cross_tree=cross_val_score(tree,wine.data, wine.target, cv=5)
kneig = KNeighborsClassifier(n_neighbors=grid_kneig.best_params_['n_neighbors'])
cross_kneig=cross_val_score(kneig, wine.data, wine.target, cv=5)
print(sum(cross_tree)/len(cross_tree))
print(sum(cross_kneig) / len(cross_kneig))
return({'tree best param':grid_tree.best_params_,
'kneig best param':grid_kneig.best_params_})

def lda_2D_visualisation():
wine=datasets.load_wine()
lda = LDA(n_components=3)
wine_lda=lda.fit_transform(wine.data, wine.target)
x=[]
y=[]
#z=[]
label=[]
label.append(wine.target)
colors=['red', 'green', 'blue']
for i in wine_lda:
x.append(i[0])
y.append(i[1])
#z.append(i[2])
plt.scatter(x, y, c=label, marker='.', cmap=matplotlib.colors.ListedColormap
(colors))
plt.show()
```

```
#Not relevant results, I decided not to use this method in my report
def ica_3D_visualisation():
wine=datasets.load_wine()

ica = FastICA(n_components=3)
wine_ica=ica.fit_transform(wine.data)
x=[]
y=[]
z=[]
label=[]
label.append(wine.target)
colors=['red', 'green', 'blue']
for i in wine_ica:
x.append(i[0])
y.append(i[1])
z.append(i[2])
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(x, y, z, c=label, marker='.',cmap=matplotlib.colors.ListedColormap
(colors))
ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')
plt.show()

#Not relevant results, I decided not to use this method in my report
def ica_2D_visualisation():
wine=datasets.load_wine()
ica = FastICA(n_components=3)
wine_ica=ica.fit_transform(wine.data, wine.target)
x=[]
y=[]
#z=[]
label=[]
label.append(wine.target)
colors=['red', 'green', 'blue']
for i in wine_ica:
x.append(i[0])
y.append(i[1])
#z.append(i[2])
plt.scatter(x, y, c=label, marker='.', cmap=matplotlib.colors.ListedColormap
(colors))
plt.show()

def ker_3D_visualisation():
wine=datasets.load_wine()
ker = KernelPCA(n_components=3)
wine_ker=ker.fit_transform(wine.data)
x=[]
```

```
y=[]
z=[]
label=[]
label.append(wine.target)
colors=['red', 'green', 'blue']
for i in wine_ker:
x.append(i[0])
y.append(i[1])
z.append(i[2])
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(x, y, z, c=label, marker='.',cmap=matplotlib.colors.ListedColormap
(colors))
ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')
plt.show()

def ker_2D_visualisation():
wine=datasets.load_wine()
ker = KernelPCA(n_components=3)
wine_ker=ker.fit_transform(wine.data, wine.target)
x=[]
y=[]
#z=[]
label=[]
label.append(wine.target)
colors=['red', 'green', 'blue']
for i in wine_ker:
x.append(i[0])
y.append(i[1])
#z.append(i[2])
plt.scatter(x, y, c=label, marker='.', cmap=matplotlib.colors.ListedColormap
(colors))
plt.show()
```