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.

The number of distinct principal components is equal to the smaller of the number of original variables or the number of observations minus one.

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 the preceding components.

The resulting vectors are an uncorrelated orthogonal basis set.

Read in the data and perform basic exploratory analysis

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

```
df = pd.read_csv('/content/drive/MyDrive/Datasets/wine.data.csv')
df.head(10)
```

	Class	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavano pheno
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.
5	1	14.20	1.76	2.45	15.2	112	3.27	3.39	0.
6	1	14.39	1.87	2.45	14.6	96	2.50	2.52	0.
7	1	14.06	2.15	2.61	17.6	121	2.60	2.51	0.
8	1	14.83	1.64	2.17	14.0	97	2.80	2.98	0.
9	1	13.86	1.35	2.27	16.0	98	2.98	3.15	0.

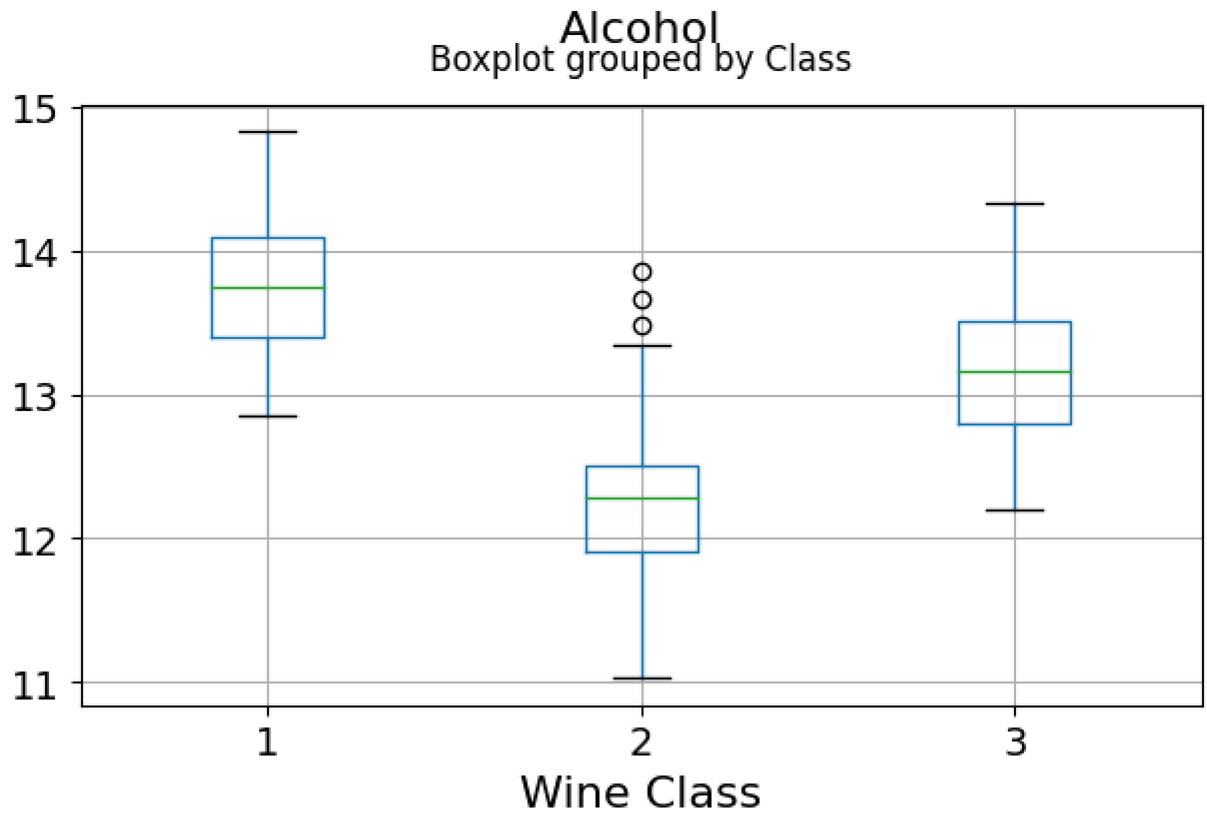
```
df.shape
```

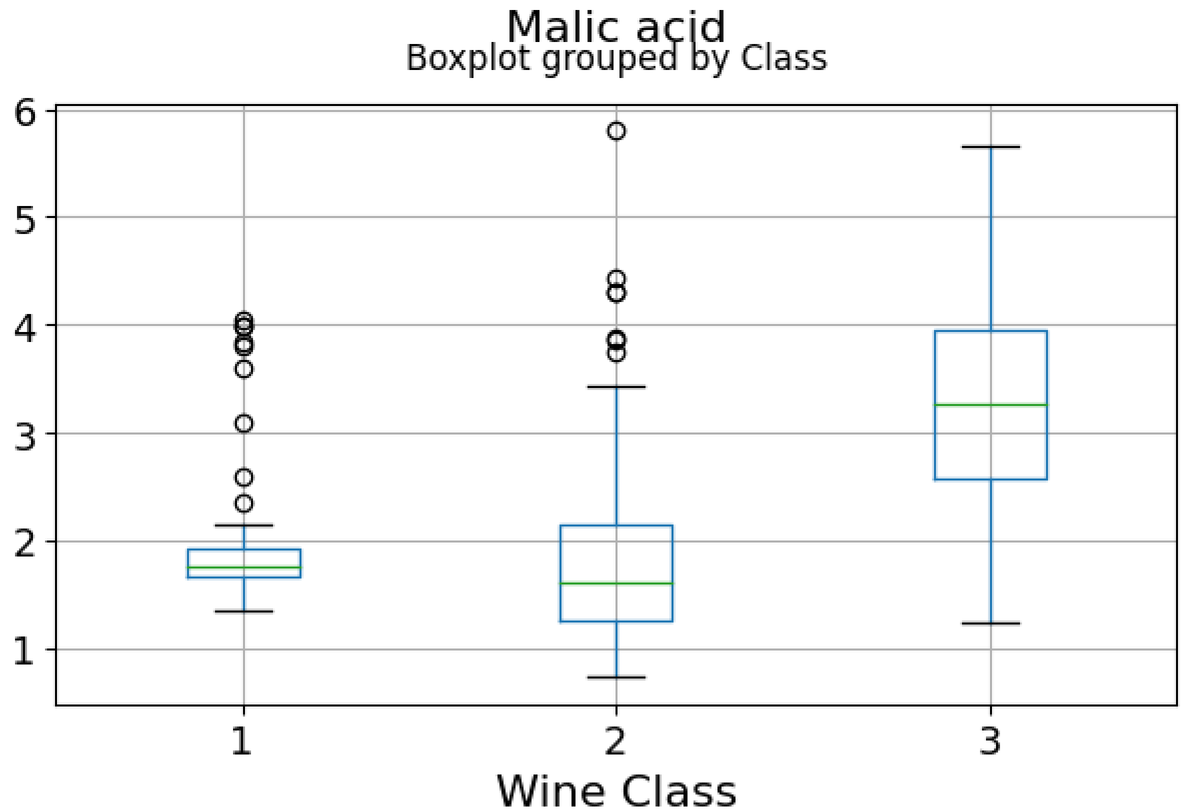
```
(178, 14)
```

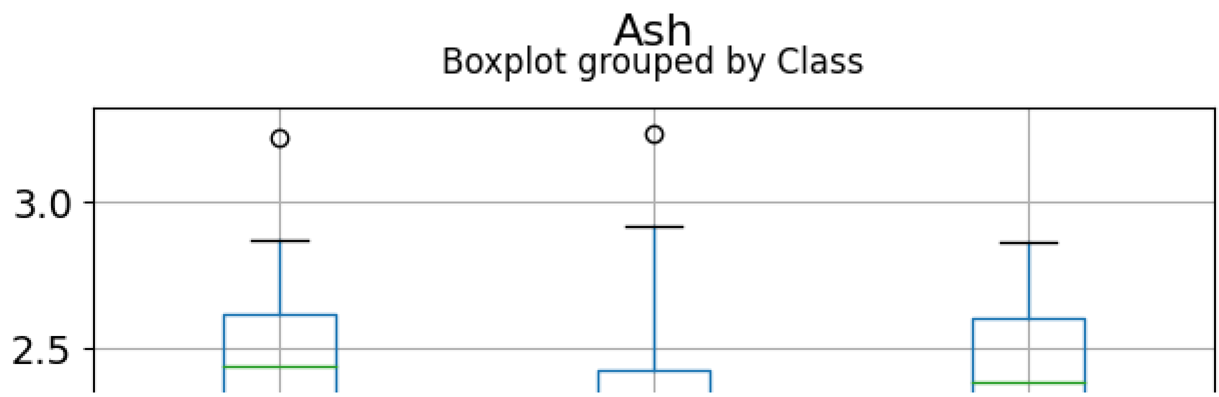
```
df.iloc[:,1:].describe()
```

	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoi
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.0000
mean	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.0292
std	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.9988
min	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.3400
25%	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.2050
50%	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.1350
75%	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.8750
max	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.0800

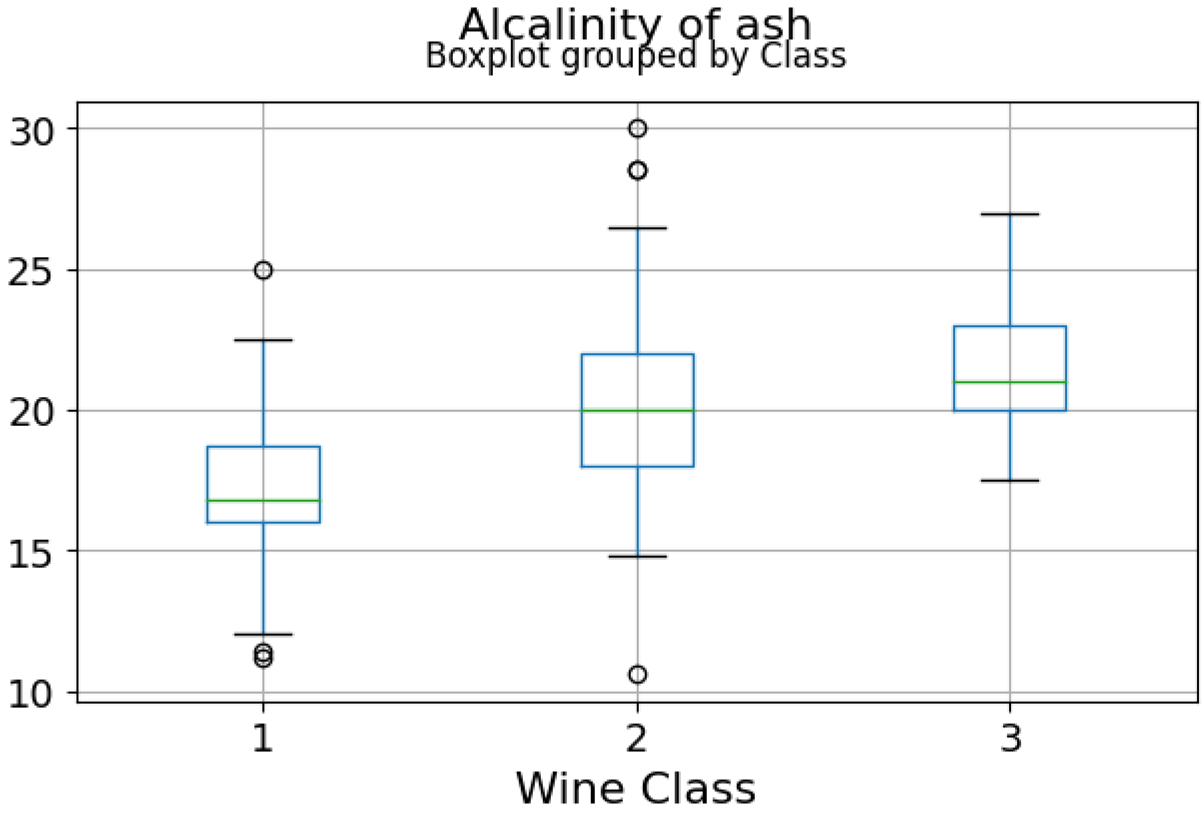
```
for c in df.columns[1:]:
    df.boxplot(c,by='Class',figsize=(7,4),fontsize=14)
    plt.title("{}\n".format(c),fontsize=16)
    plt.xlabel("Wine Class", fontsize=16)
```


Alcohol
Boxplot grouped by Class
15
14
13
12
11
1
2
3
Wine Class

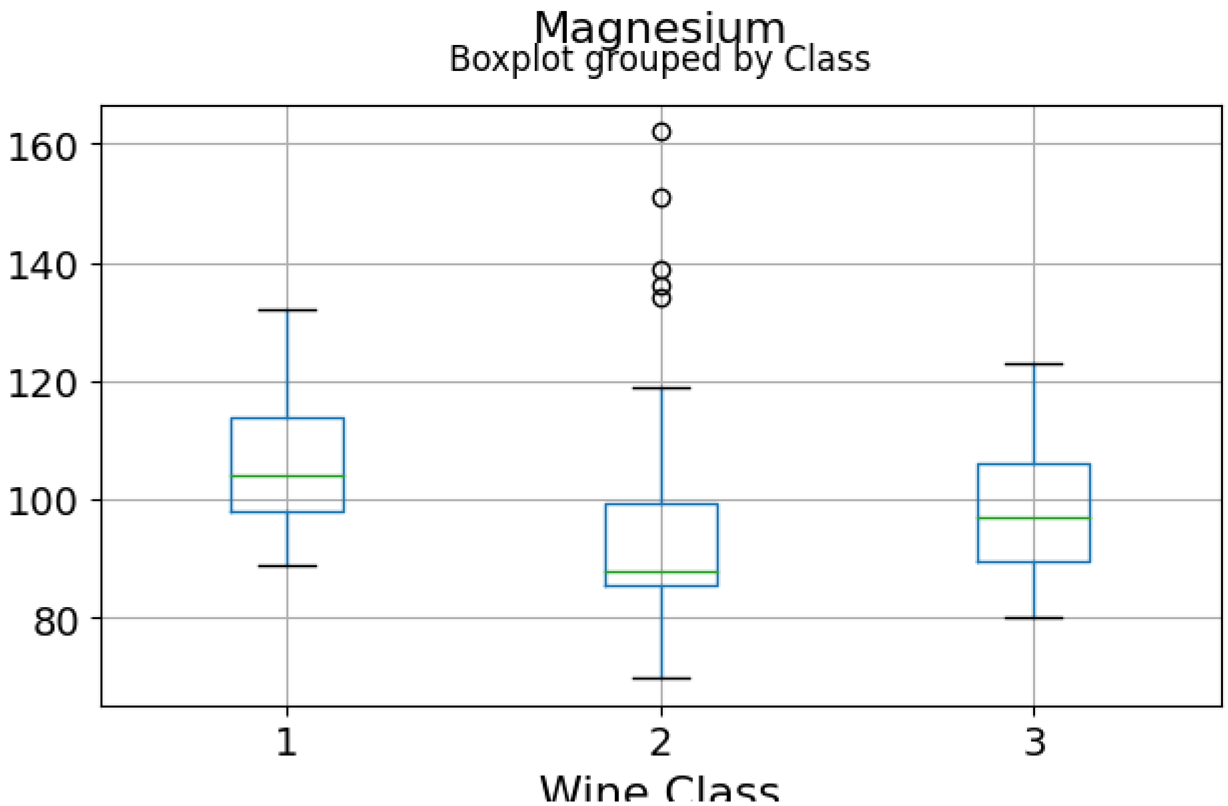

Malic acid
Boxplot grouped by Class
6
5
4
3
2
1
1
2
3
Wine Class


Ash
Boxplot grouped by Class
3.0
2.5

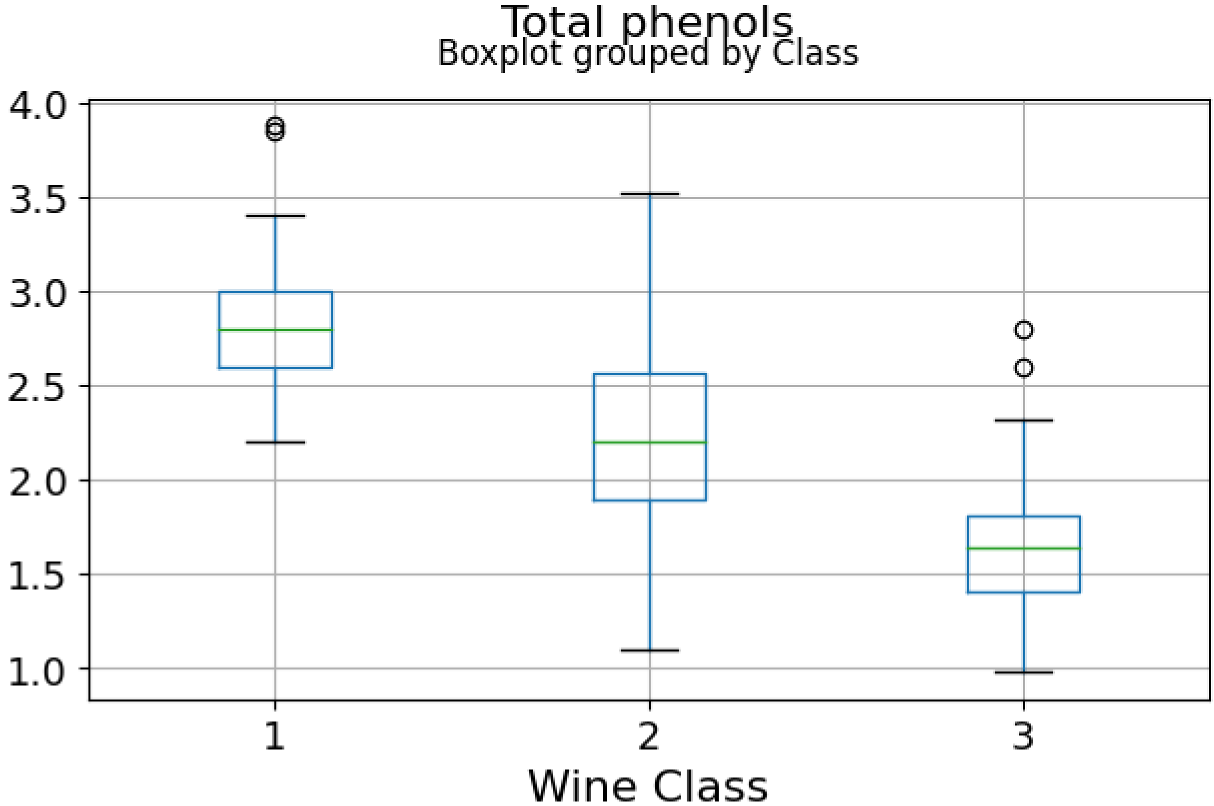
2.0
1.5
1
2
3
Wine Class

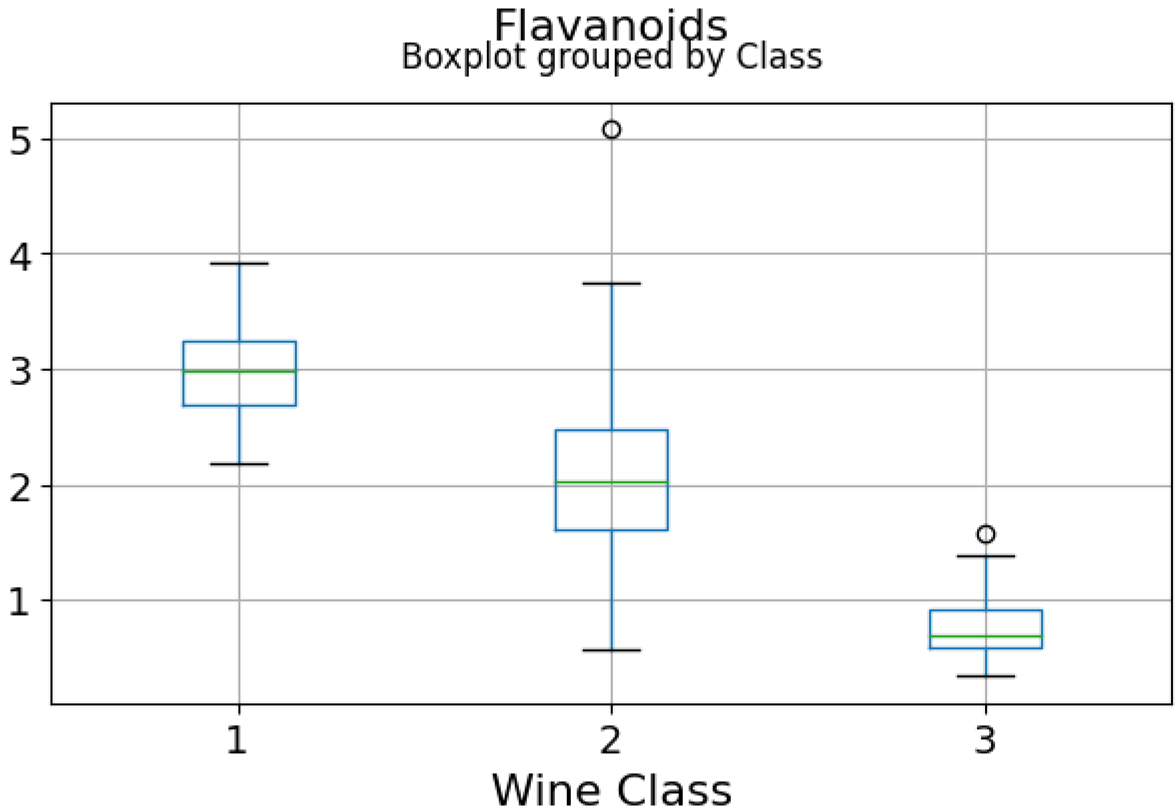


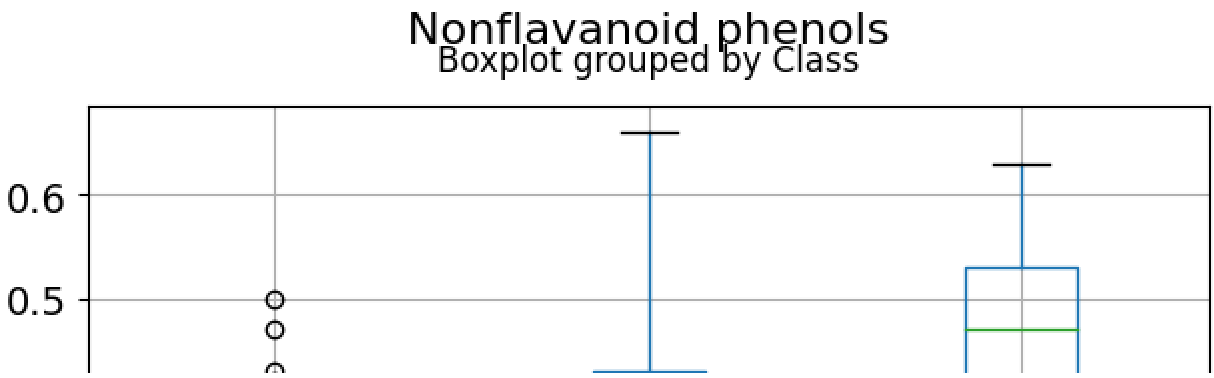
Alcalinity of ash
Boxplot grouped by Class
30
25
20
15
10
1
2
3
Wine Class

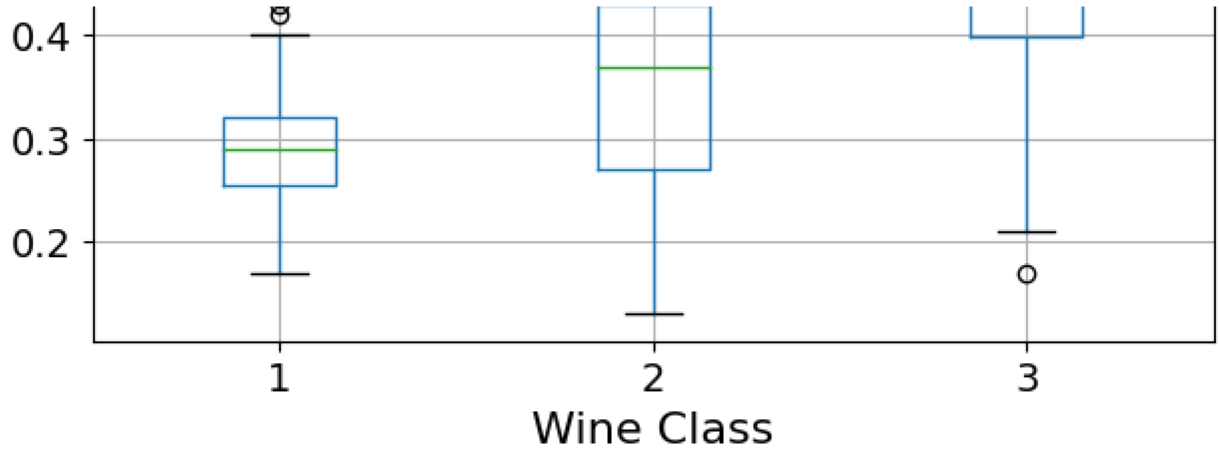


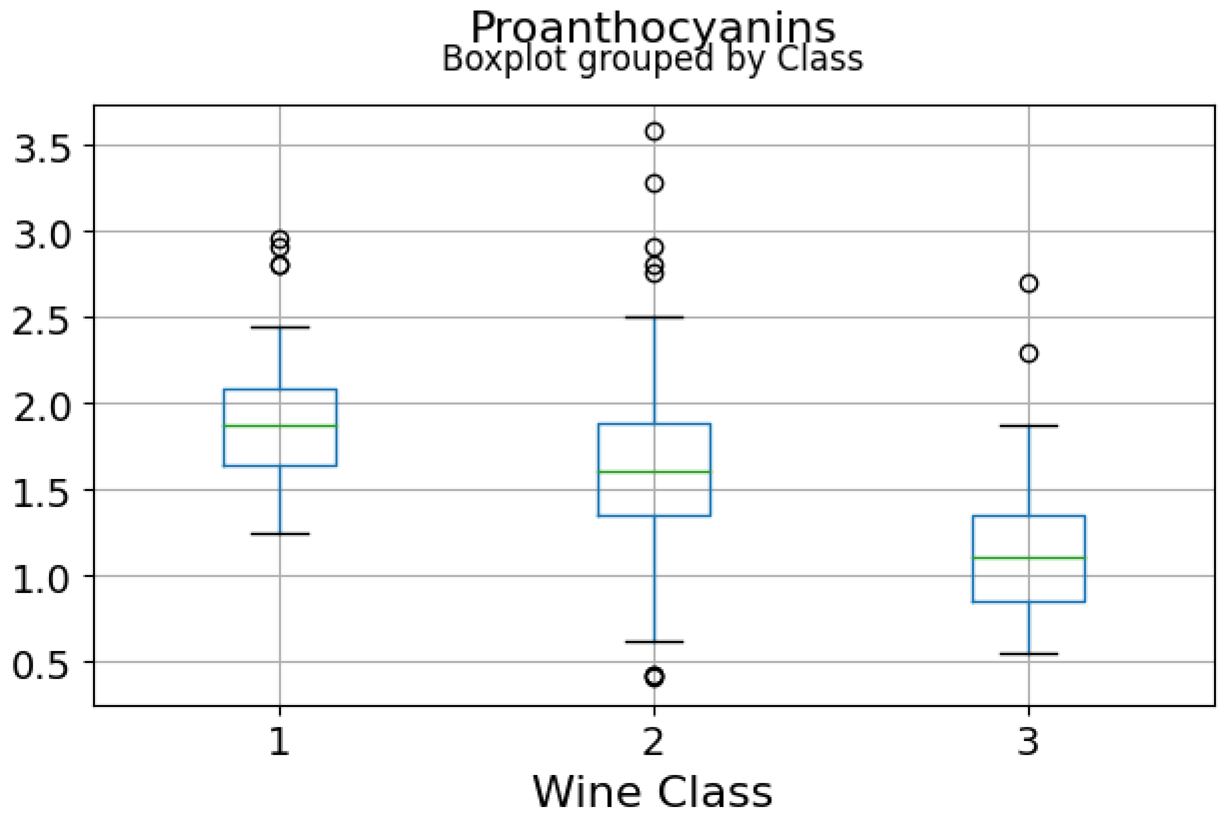
Magnesium
Boxplot grouped by Class
160
140
120
100
80
1
2
3
Wine Class

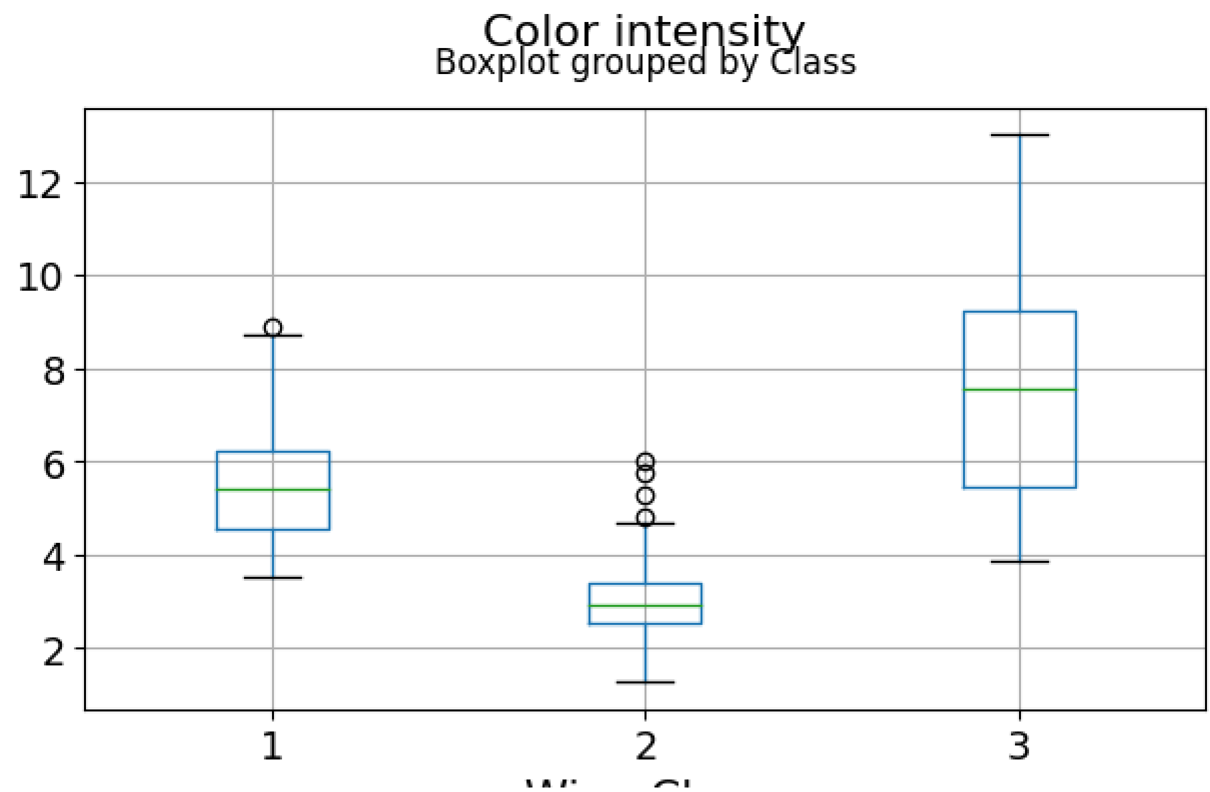

Total phenols
Boxplot grouped by Class
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1
2
3
Wine Class


Flavanoids
Boxplot grouped by Class
5
4
3
2
1
1
2
3
Wine Class

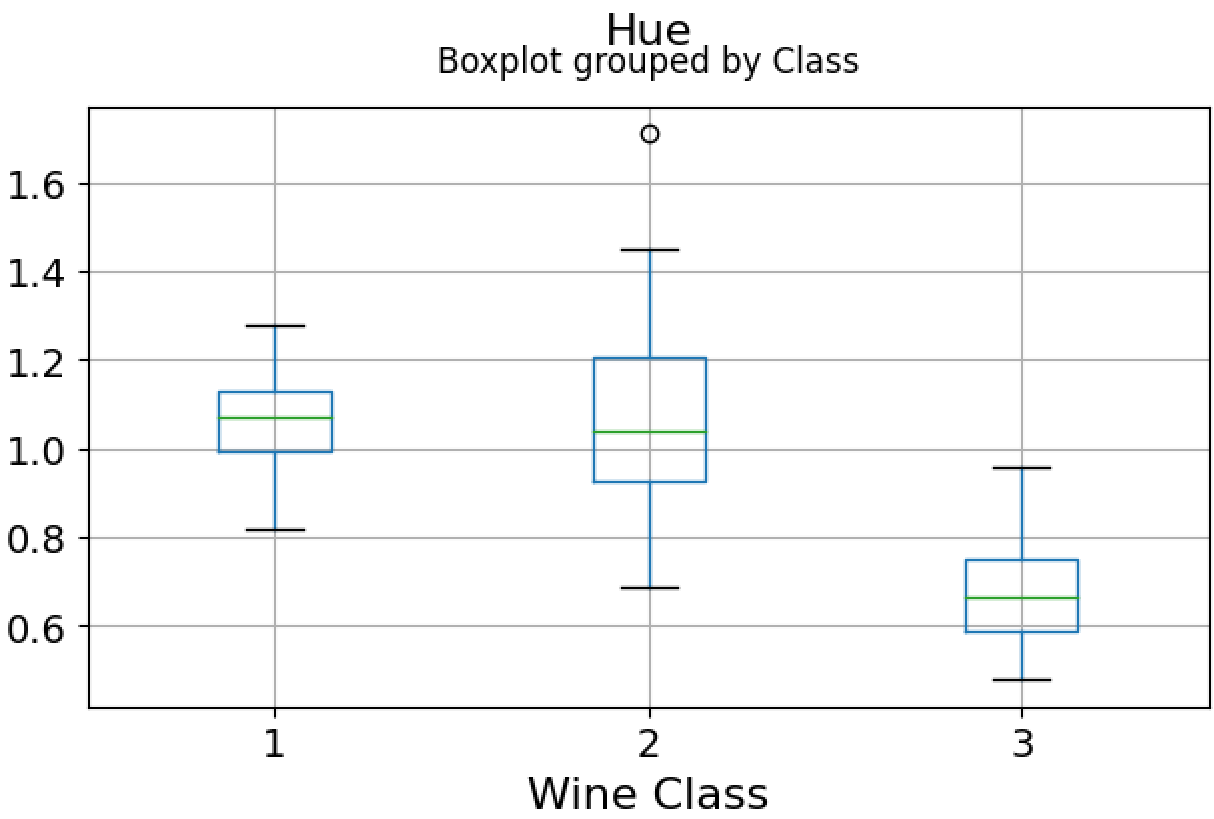

Nonflavanoid phenols
Boxplot grouped by Class
0.6
0.5

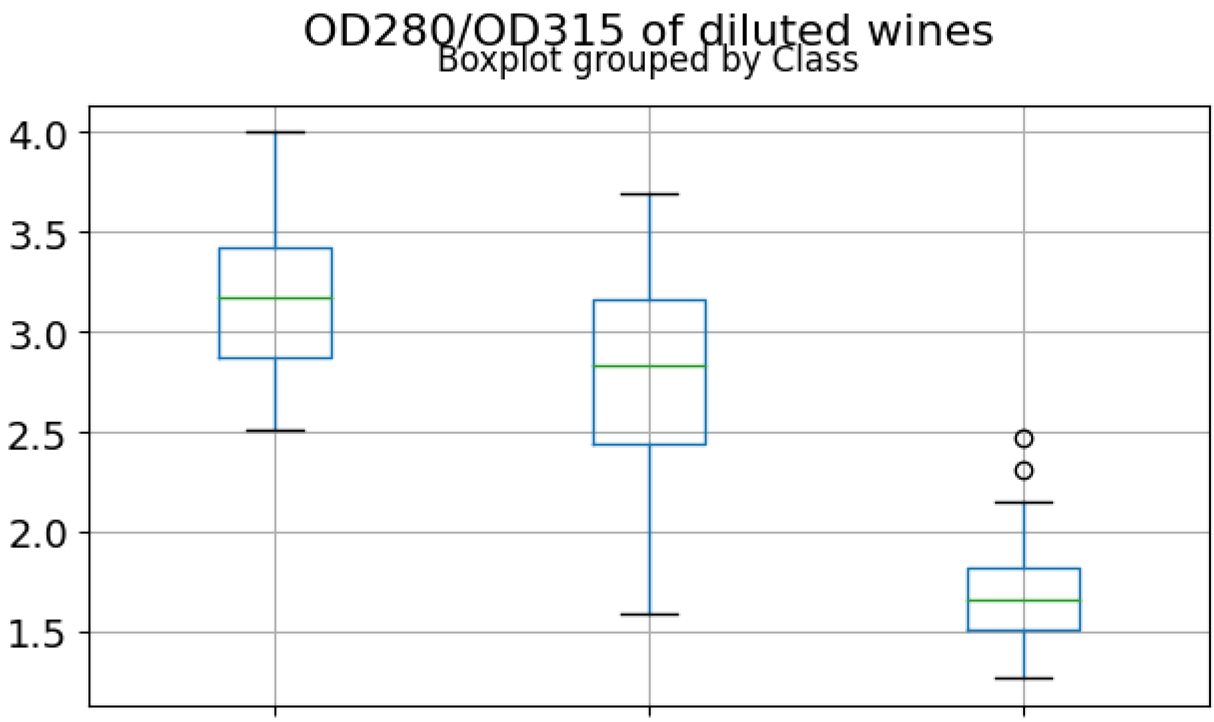

0.4
0.3
0.2
1
2
3
Wine Class


Proanthocyanins
Boxplot grouped by Class
3.5
3.0
2.5
2.0
1.5
1.0
0.5
1
2
3
Wine Class

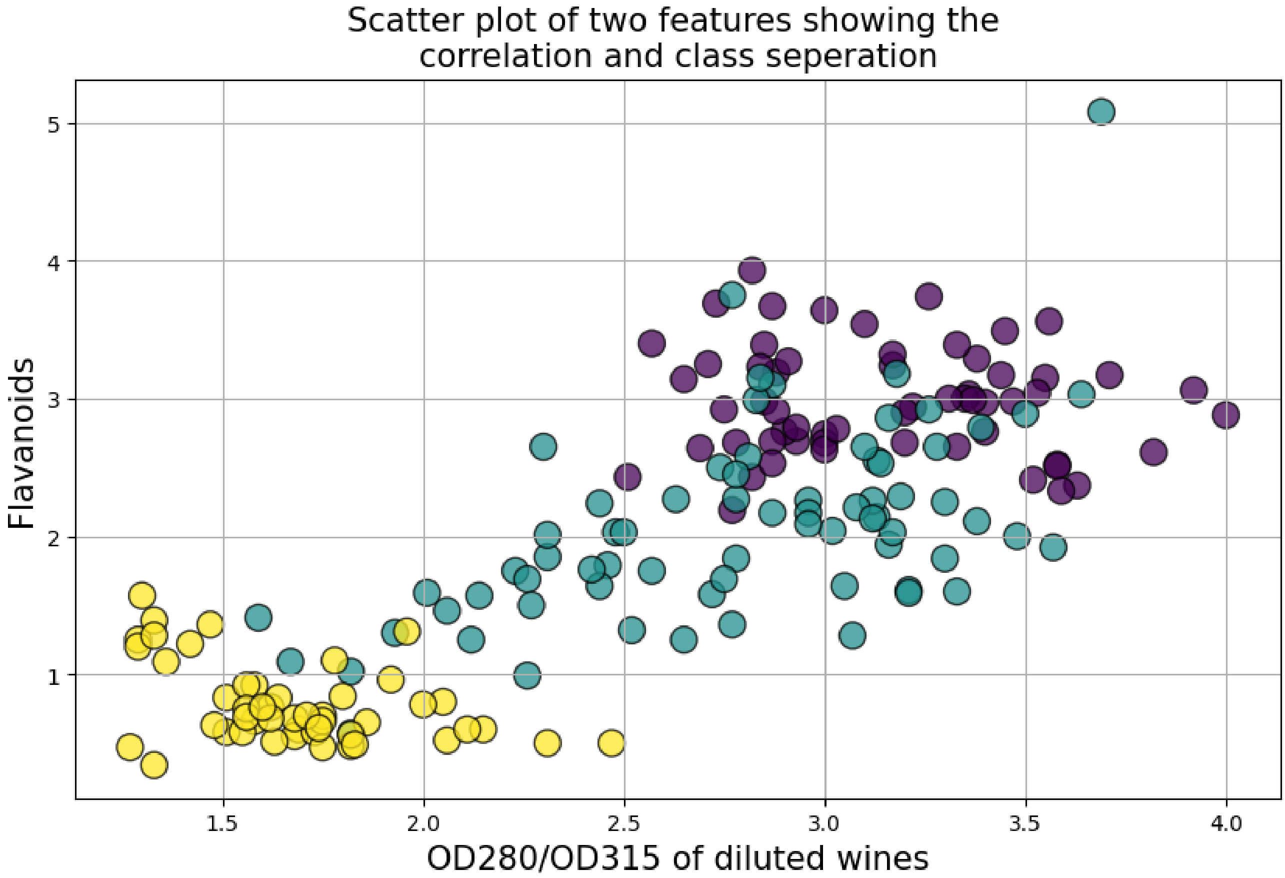

Color intensity
Boxplot grouped by Class
12
10
8
6
4
2
1
2
3

Wine Class






```
plt.figure(figsize=(10,6))
plt.scatter(df['OD280/OD315 of diluted wines'],df['Flavanoids'],c=df['Class'],edgecolors=
plt.grid(True)
plt.title("Scatter plot of two features showing the \ncorrelation and class seperation",f
plt.xlabel("OD280/OD315 of diluted wines",fontsize=15)
plt.ylabel("Flavanoids",fontsize=15)
plt.show()
```

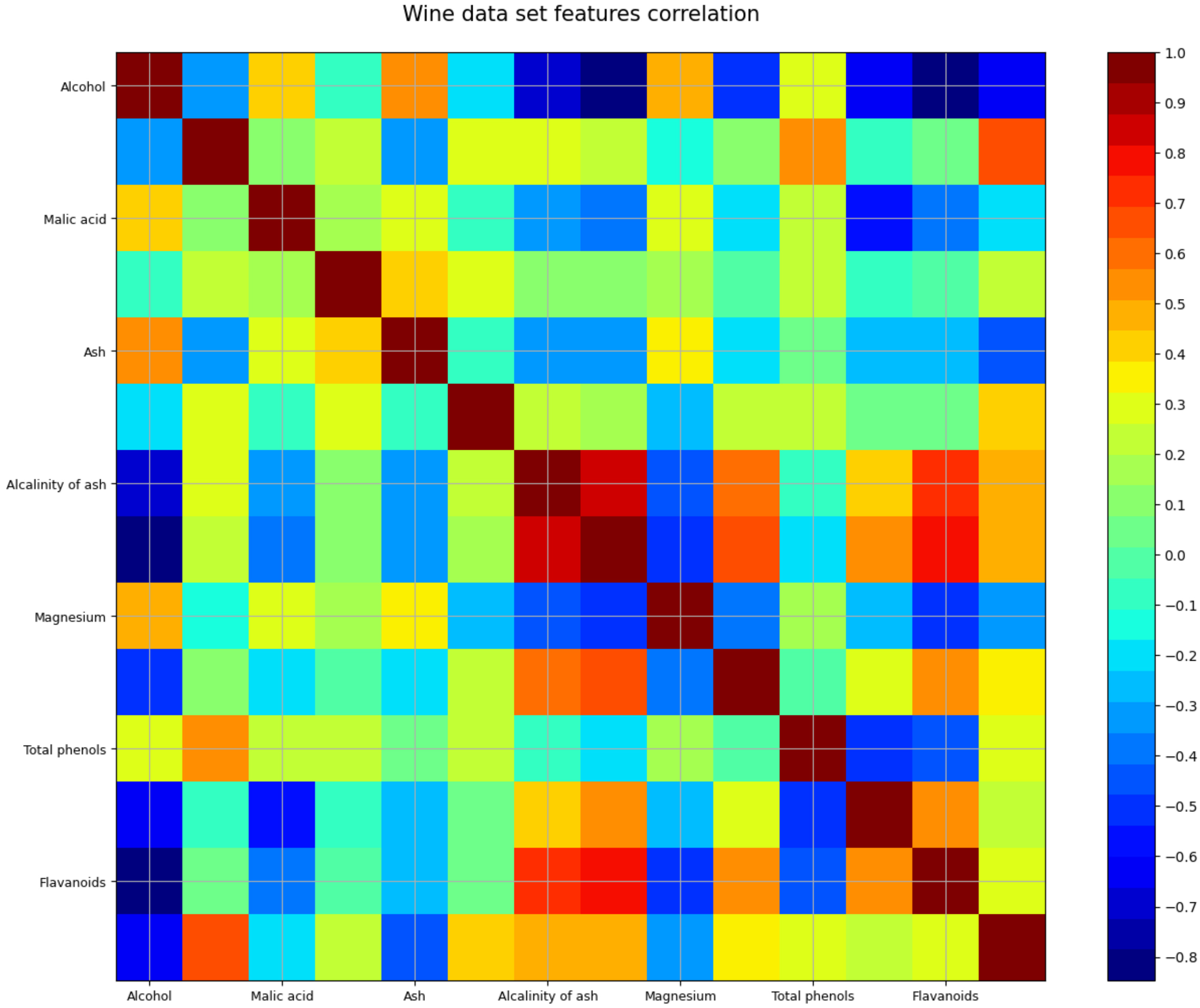



```
def correlation_matrix(df):
    from matplotlib import pyplot as plt
    from matplotlib import cm as cm

    fig = plt.figure(figsize=(16,12))
    ax1 = fig.add_subplot(111)
    cmap = cm.get_cmap('jet', 30)
    cax = ax1.imshow(df.corr(), interpolation="nearest", cmap=cmap)
    ax1.grid(True)
    plt.title('Wine data set features correlation\n',fontsize=15)
    labels=df.columns
    ax1.set_xticklabels(labels,fontsize=9)
    ax1.set_yticklabels(labels,fontsize=9)
    # Add colorbar, make sure to specify tick locations to match desired ticklabels
    fig.colorbar(cax, ticks=[0.1*i for i in range(-11,11)])
    plt.show()

correlation_matrix(df)
```

```
<ipython-input-27-ff6b588b834f>:7: MatplotlibDeprecationWarning: The get_cmap functio
  cmap = cm.get_cmap('jet', 30)
<ipython-input-27-ff6b588b834f>:12: UserWarning: FixedFormatter should only be used t
  ax1.set_xticklabels(labels,fontsize=9)
<ipython-input-27-ff6b588b834f>:13: UserWarning: FixedFormatter should only be used t
  ax1.set_yticklabels(labels,fontsize=9)
```



Data scaling

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
```

```
X = df.drop('Class',axis=1)
y = df['Class']
```

```
X = scaler.fit_transform(X)
```

```
dfx = pd.DataFrame(data=X,columns=df.columns[1:])
dfx.head(10)
```

	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflava phe
0	1.518613	-0.562250	0.232053	-1.169593	1.913905	0.808997	1.034819	-0.65
1	0.246290	-0.499413	-0.827996	-2.490847	0.018145	0.568648	0.733629	-0.82
2	0.196879	0.021231	1.109334	-0.268738	0.088358	0.808997	1.215533	-0.49
3	1.691550	-0.346811	0.487926	-0.809251	0.930918	2.491446	1.466525	-0.98
4	0.295700	0.227694	1.840403	0.451946	1.281985	0.808997	0.663351	0.22
5	1.481555	-0.517367	0.305159	-1.289707	0.860705	1.562093	1.366128	-0.17
6	1.716255	-0.418624	0.305159	-1.469878	-0.262708	0.328298	0.492677	-0.49
7	1.308617	-0.167278	0.890014	-0.569023	1.492625	0.488531	0.482637	-0.41
8	2.259772	-0.625086	-0.718336	-1.650049	-0.192495	0.808997	0.954502	-0.57
9	1.061565	-0.885409	-0.352802	-1.049479	-0.122282	1.097417	1.125176	-1.14

```
dfx.describe()
```

	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	To phen
count	1.780000e+02	1.780000e+02	1.780000e+02	1.780000e+02	1.780000e+02	178.000
mean	-8.382808e-16	-1.197544e-16	-8.370333e-16	-3.991813e-17	-3.991813e-17	0.000
std	1.002821e+00	1.002821e+00	1.002821e+00	1.002821e+00	1.002821e+00	1.002
min	-2.434235e+00	-1.432983e+00	-3.679162e+00	-2.671018e+00	-2.088255e+00	-2.107
25%	-7.882448e-01	-6.587486e-01	-5.721225e-01	-6.891372e-01	-8.244151e-01	-0.885
50%	6.099988e-02	-4.231120e-01	-2.382132e-02	1.518295e-03	-1.222817e-01	0.095
75%	8.361286e-01	6.697929e-01	6.981085e-01	6.020883e-01	5.096384e-01	0.808
max	2.259772e+00	3.109192e+00	3.156325e+00	3.154511e+00	4.371372e+00	2.539

PCA class import and analysis

```
from sklearn.decomposition import PCA


pca = PCA(n_components=None)


dfx_pca = pca.fit(dfx)


plt.figure(figsize=(10,6))
plt.scatter(x=[i+1 for i in range(len(dfx_pca.explained_variance_ratio_))],
            y=dfx_pca.explained_variance_ratio_,
           s=200, alpha=0.75,c='orange',edgecolor='k')
plt.grid(True)
plt.title("Explained variance ratio of the \nfitted principal component vector\n",fontsiz
plt.xlabel("Principal components",fontsize=15)
plt.xticks([i+1 for i in range(len(dfx_pca.explained_variance_ratio_))],fontsize=15)
plt.yticks(fontsize=15)
plt.ylabel("Explained variance ratio",fontsize=15)
plt.show()
```

Explained variance ratio of the fitted principal component vector

Showing better class separation using principal components

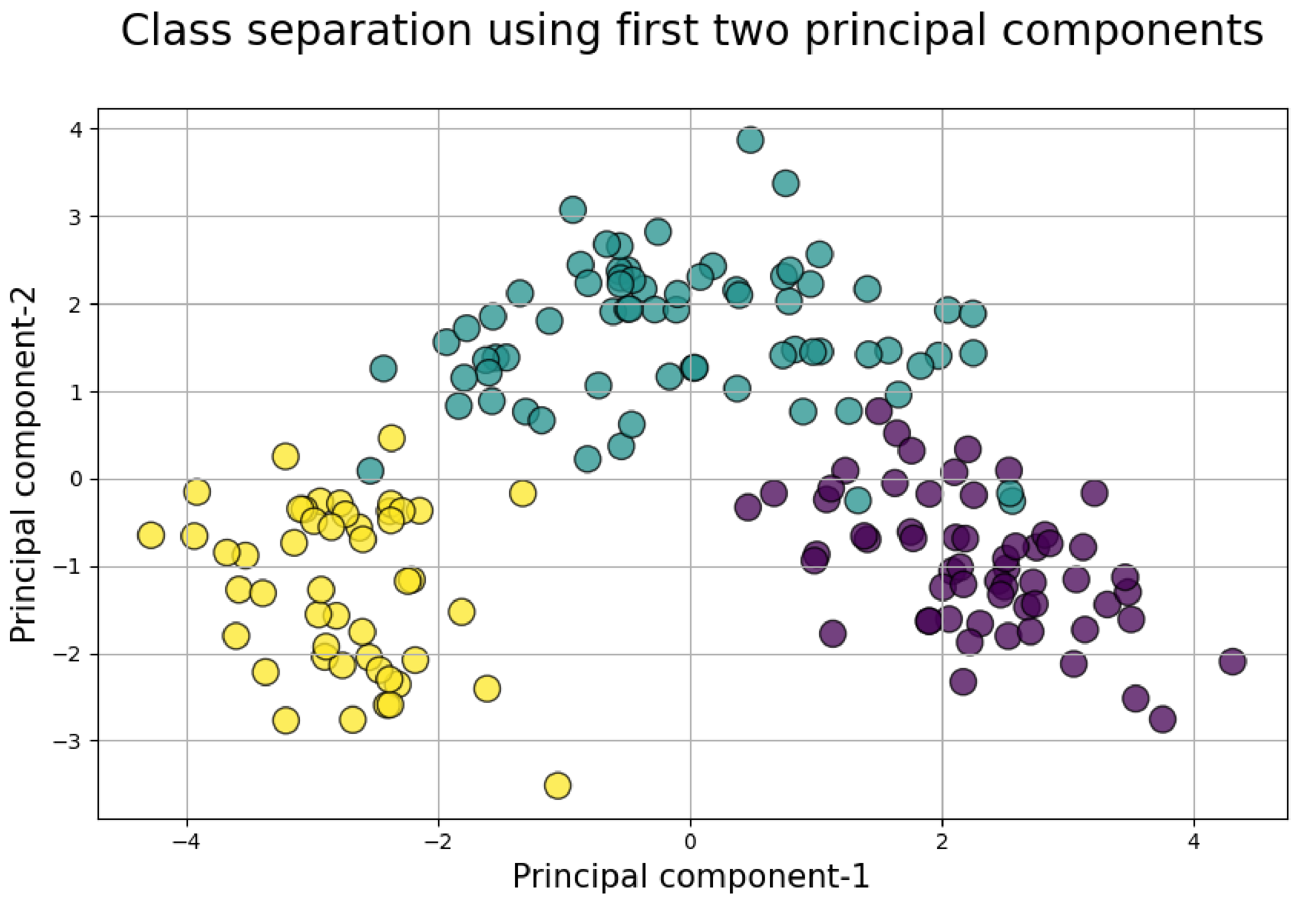
```
## Transform the scaled data set using the fitted PCA object
dfx_trans = pca.transform(dfx)
```

```
## Put it in a data frame
dfx_trans = pd.DataFrame(data=dfx_trans)
dfx_trans.head(10)
```

	0	1	2	3	4	5	6	7	
0	3.316751	-1.443463	-0.165739	-0.215631	0.693043	-0.223880	0.596427	0.065139	0
1	2.209465	0.333393	-2.026457	-0.291358	-0.257655	-0.927120	0.053776	1.024416	-0
2	2.516740	-1.031151	0.982819	0.724902	-0.251033	0.549276	0.424205	-0.344216	-1
3	3.757066	-2.756372	-0.176192	0.567983	-0.311842	0.114431	-0.383337	0.643593	0
4	1.008908	-0.869831	2.026688	-0.409766	0.298458	-0.406520	0.444074	0.416700	0
5	3.050254	-2.122401	-0.629396	-0.515637	-0.632019	0.123431	0.401654	0.394893	-0
6	2.449090	-1.174850	-0.977095	-0.065831	-1.027762	-0.620121	0.052891	-0.371934	-0
7	2.059437	-1.608963	0.146282	-1.192608	0.076903	-1.439806	0.032376	0.232979	0
8	2.510874	-0.918071	-1.770969	0.056270	-0.892257	-0.129181	0.125285	-0.499578	0
9	2.753628	-0.789438	-0.984247	0.349382	-0.468553	0.163392	-0.874352	0.150580	0

Plot the first two columns of this transformed data set with the color set to original ground truth class label

```
plt.figure(figsize=(10,6))
plt.scatter(dfx_trans[0],dfx_trans[1],c=df['Class'],edgecolors='k',alpha=0.75,s=150)
plt.grid(True)
plt.title("Class separation using first two principal components\n",fontsize=20)
plt.xlabel("Principal component-1",fontsize=15)
plt.ylabel("Principal component-2",fontsize=15)
plt.show()
```


Class separation using first two principal components
Principal component-1
Principal component-2
4
3
2
1
0
−1
−2
−3
−4
−2
0
2
4