# 1. Cleaning and Exploration

In [4]:

```
import pandas as pd
```

In [5]:

```
wine=pd.read_csv(r'C:\Users\DIVYANJALI\Downloads\Wine.csv')
wine.head()
```

Out[5]:

| | Alcohol | Malic acid | Ash | Alcalinity of ash | Magnesium | Total phenols | Flavanoids | Nonflavanoid phenols | Proanthocya |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 14.23 | 1.71 | 2.43 | 15.6 | 127 | 2.80 | 3.06 | 0.28 | |
| **1** | 13.20 | 1.78 | 2.14 | 11.2 | 100 | 2.65 | 2.76 | 0.26 | |
| **2** | 13.16 | 2.36 | 2.67 | 18.6 | 101 | 2.80 | 3.24 | 0.30 | |
| **3** | 14.37 | 1.95 | 2.50 | 16.8 | 113 | 3.85 | 3.49 | 0.24 | |
| **4** | 13.24 | 2.59 | 2.87 | 21.0 | 118 | 2.80 | 2.69 | 0.39 | |

In [6]:

```
wine.isna().sum()
```

Out[6]:

```
Alcohol                         0
Malic acid                      0
Ash                             0
Alcalinity of ash               0
Magnesium                       0
Total phenols                   0
Flavanoids                      0
Nonflavanoid phenols            0
Proanthocyanins                 0
Color intensity                 0
Hue                             0
OD280/OD315 of diluted wines    0
Proline                         0
dtype: int64
```

In [7]:

```
wine.duplicated()
```

Out[7]:

```
0      False
1      False
2      False
3      False
4      False
       ...
173    False
174    False
175    False
176    False
177    False
Length: 178, dtype: bool
```

In [9]:

```
wine.describe()
```

Out[9]:

| | Alcohol | Malic acid | Ash | Alcalinity of ash | Magnesium | Total phenols | Flavanoids | N |
|---|---|---|---|---|---|---|---|---|
| **count** | 178.000000 | 178.000000 | 178.000000 | 178.000000 | 178.000000 | 178.000000 | 178.000000 | |
| **mean** | 13.000618 | 2.336348 | 2.366517 | 19.494944 | 99.741573 | 2.295112 | 2.029270 | |
| **std** | 0.811827 | 1.117146 | 0.274344 | 3.339564 | 14.282484 | 0.625851 | 0.998859 | |
| **min** | 11.030000 | 0.740000 | 1.360000 | 10.600000 | 70.000000 | 0.980000 | 0.340000 | |
| **25%** | 12.362500 | 1.602500 | 2.210000 | 17.200000 | 88.000000 | 1.742500 | 1.205000 | |
| **50%** | 13.050000 | 1.865000 | 2.360000 | 19.500000 | 98.000000 | 2.355000 | 2.135000 | |
| **75%** | 13.677500 | 3.082500 | 2.557500 | 21.500000 | 107.000000 | 2.800000 | 2.875000 | |
| **max** | 14.830000 | 5.800000 | 3.230000 | 30.000000 | 162.000000 | 3.880000 | 5.080000 | |

In [10]:

```
wine.nunique()
```

Out[10]:

```
Alcohol                         126
Malic acid                      133
Ash                              79
Alcalinity of ash                63
Magnesium                        53
Total phenols                    97
Flavanoids                      132
Nonflavanoid phenols             39
Proanthocyanins                 101
Color intensity                 132
Hue                              78
OD280/OD315 of diluted wines    122
Proline                         121
dtype: int64
```

In [11]:

```
wine.dtypes
```

Out[11]:

```
Alcohol                         float64
Malic acid                      float64
Ash                             float64
Alcalinity of ash               float64
Magnesium                         int64
Total phenols                   float64
Flavanoids                      float64
Nonflavanoid phenols            float64
Proanthocyanins                 float64
Color intensity                 float64
Hue                             float64
OD280/OD315 of diluted wines    float64
Proline                           int64
dtype: object
```

In [12]:

```
wine.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 13 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Alcohol                       178 non-null    float64
 1   Malic acid                    178 non-null    float64
 2   Ash                           178 non-null    float64
 3   Alcalinity of ash             178 non-null    float64
 4   Magnesium                     178 non-null    int64  
 5   Total phenols                 178 non-null    float64
 6   Flavanoids                    178 non-null    float64
 7   Nonflavanoid phenols          178 non-null    float64
 8   Proanthocyanins               178 non-null    float64
 9   Color intensity               178 non-null    float64
 10  Hue                           178 non-null    float64
 11  OD280/OD315 of diluted wines  178 non-null    float64
 12  Proline                       178 non-null    int64  
dtypes: float64(11), int64(2)
memory usage: 18.2 KB
```

In [13]:

```
wine.shape
```

Out[13]:

```
(178, 13)
```

In [14]:

```
len(wine)
```

Out[14]:

```
178
```

In [15]:

```
wine.columns
```

Out[15]:

```
Index(['Alcohol', 'Malic acid', 'Ash', 'Alcalinity of ash', 'Magnesium',
       'Total phenols', 'Flavanoids', 'Nonflavanoid phenols',
       'Proanthocyanins', 'Color intensity', 'Hue',
       'OD280/OD315 of diluted wines', 'Proline'],
      dtype='object')
```

In [16]:

```
len(wine.columns)
```

Out[16]:

```
13
```

In [17]:

```
wine.size
```

Out[17]:

```
2314
```

In [18]:

```
wine.tail()
```

Out[18]:

| | Alcohol | Malic acid | Ash | Alcalinity of ash | Magnesium | Total phenols | Flavanoids | Nonflavanoid phenols | Proanthoc |
|---|---|---|---|---|---|---|---|---|---|
| **173** | 13.71 | 5.65 | 2.45 | 20.5 | 95 | 1.68 | 0.61 | 0.52 | |
| **174** | 13.40 | 3.91 | 2.48 | 23.0 | 102 | 1.80 | 0.75 | 0.43 | |
| **175** | 13.27 | 4.28 | 2.26 | 20.0 | 120 | 1.59 | 0.69 | 0.43 | |
| **176** | 13.17 | 2.59 | 2.37 | 20.0 | 120 | 1.65 | 0.68 | 0.53 | |
| **177** | 14.13 | 4.10 | 2.74 | 24.5 | 96 | 2.05 | 0.76 | 0.56 | |

In [19]:

```
wine.corr()
```

Out[19]:

| | Alcohol | Malic acid | Ash | Alcalinity of ash | Magnesium | Total phenols | Flavanoids |
|---|---|---|---|---|---|---|---|
| **Alcohol** | 1.000000 | 0.094397 | 0.211545 | -0.310235 | 0.270798 | 0.289101 | 0.236815 |
| **Malic acid** | 0.094397 | 1.000000 | 0.164045 | 0.288500 | -0.054575 | -0.335167 | -0.411007 |
| **Ash** | 0.211545 | 0.164045 | 1.000000 | 0.443367 | 0.286587 | 0.128980 | 0.115077 |
| **Alcalinity of ash** | -0.310235 | 0.288500 | 0.443367 | 1.000000 | -0.083333 | -0.321113 | -0.351370 |
| **Magnesium** | 0.270798 | -0.054575 | 0.286587 | -0.083333 | 1.000000 | 0.214401 | 0.195784 |
| **Total phenols** | 0.289101 | -0.335167 | 0.128980 | -0.321113 | 0.214401 | 1.000000 | 0.864564 |
| **Flavanoids** | 0.236815 | -0.411007 | 0.115077 | -0.351370 | 0.195784 | 0.864564 | 1.000000 |
| **Nonflavanoid phenols** | -0.155929 | 0.292977 | 0.186230 | 0.361922 | -0.256294 | -0.449935 | -0.537900 |
| **Proanthocyanins** | 0.136698 | -0.220746 | 0.009652 | -0.197327 | 0.236441 | 0.612413 | 0.652692 |
| **Color intensity** | 0.546364 | 0.248985 | 0.258887 | 0.018732 | 0.199950 | -0.055136 | -0.172379 |
| **Hue** | -0.071747 | -0.561296 | -0.074667 | -0.273955 | 0.055398 | 0.433681 | 0.543479 |
| **OD280/OD315 of diluted wines** | 0.072343 | -0.368710 | 0.003911 | -0.276769 | 0.066004 | 0.699949 | 0.787194 |
| **Proline** | 0.643720 | -0.192011 | 0.223626 | -0.440597 | 0.393351 | 0.498115 | 0.494193 |

## 2. Univariate EDA

In [20]:

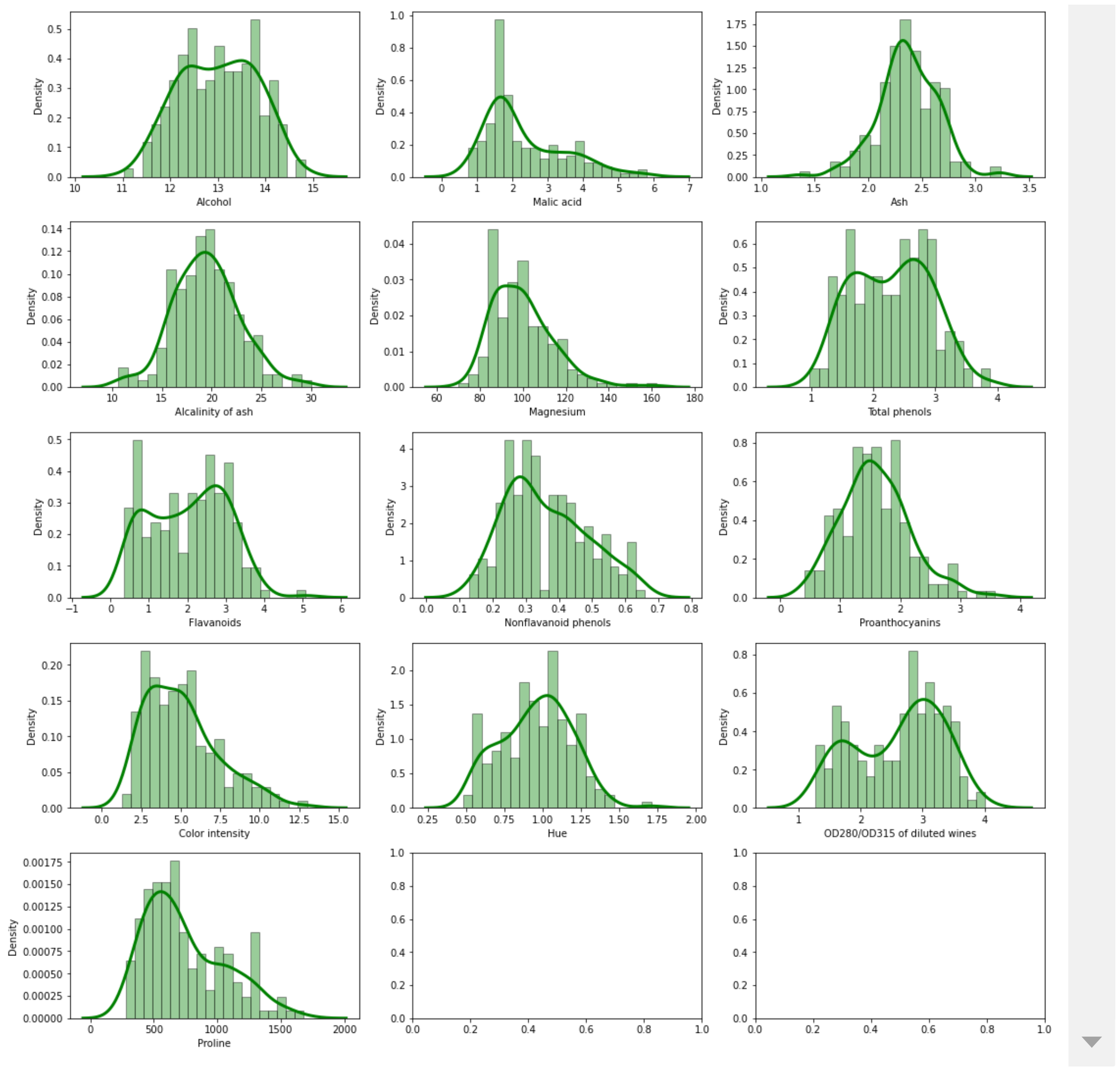
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
%matplotlib inline

import warnings
warnings.filterwarnings("ignore")
```

In [21]:

```
rows=5
cols=3
fig,ax=plt.subplots(nrows=rows,ncols=cols,figsize=(15,15))
col=wine.columns
index=0
for i in range(rows):
    for j in range(cols):
        sns.distplot(wine[col[index]], ax=ax[i][j],
                     color='green', bins=20,
                     hist_kws={'edgecolor':'black'},
                     kde_kws={'linewidth':3})
        index=index+1
        if index>=12:
            break
plt.tight_layout()

'''Here we can conclude that, Alkalinity of ash, Magnesium and Proanthocyanins approximatel
Alcohol, Malic acid, Total Phenols, Flavanoids, Hue, OD280/OD315 and Proline of diluted win
suggests the presence of 2 clusters.'''
```

Out[21]:

```
'Here we can conclude that, Alkalinity of ash, Magnesium and Proanthocyanins
approximately follows Normal distribution.\nAlcohol, Malic acid, Total Pheno
ls, Flavanoids, Hue, OD280/OD315 and Proline of diluted wines\nsuggests the
presence of 2 clusters.'
```



In [ ]:

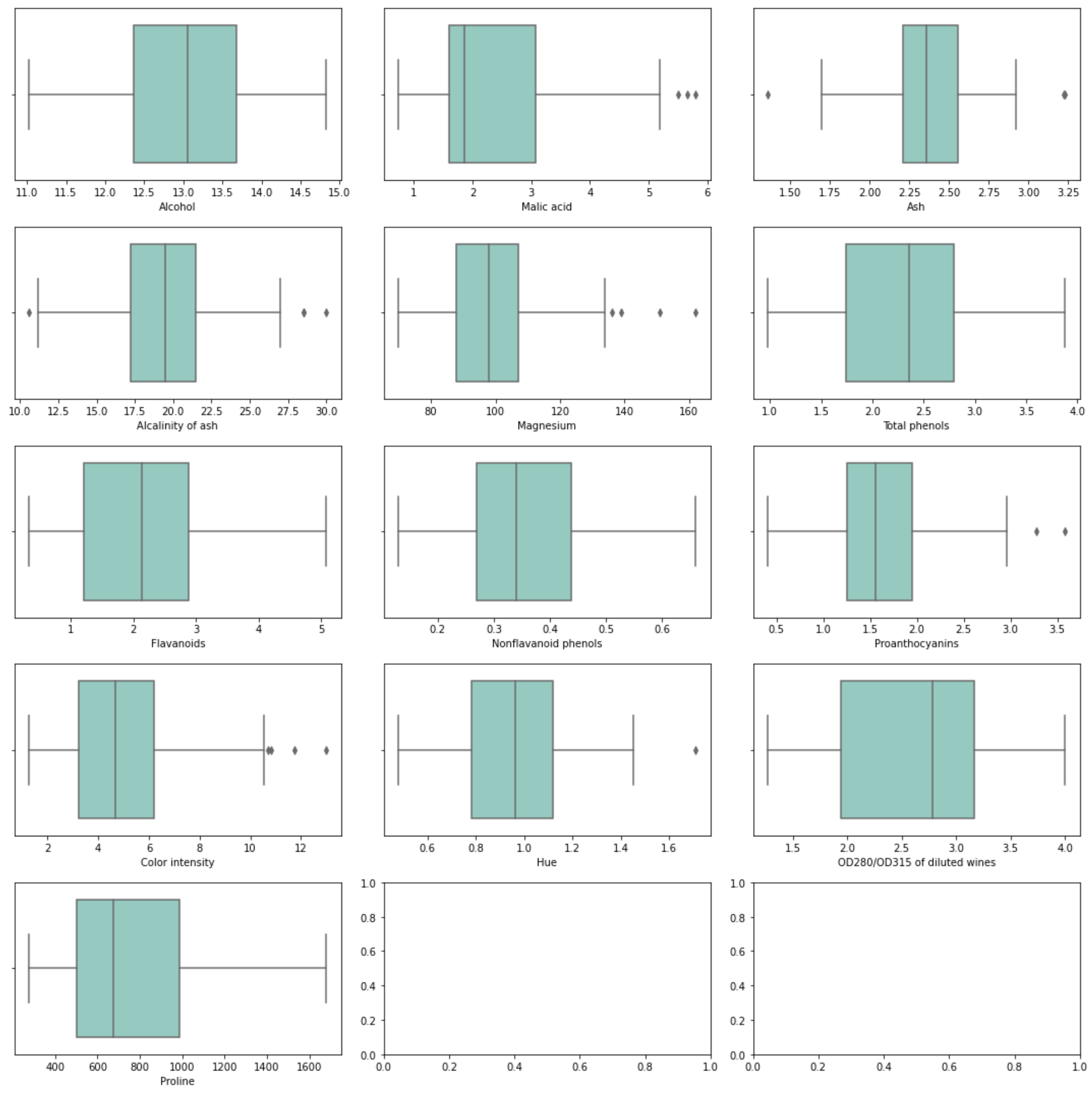
```
# Alcohol,OD280/OD315 of diluted wines and Ash are negatively skewed. The other variables a
```

In [22]:

```
rows=5
cols=3
fig,ax=plt.subplots(nrows=rows,ncols=cols,figsize=(15,15))
col=wine.columns
index=0
for i in range(rows):
    for j in range(cols):
        sns.boxplot(wine[col[index]],ax=ax[i][j], palette="Set3")
        index=index+1
        if index>=12:
            break
plt.tight_layout()

'''From the boxplots we observe that the variables
Malic acid, Ash, Alcalinity of Ash, Magnesium, Proanthocyanins, Color intensity, Hue
seem to have outliers'''
```

Out[22]:

```
'From the boxplots we observe that the variables \nMalic acid, Ash, Alcalini
ty of Ash, Magnesium, Proanthocyanins, Color intensity, Hue \nseem to have o
utliers'
```



In [ ]:

```
# From the boxplots we observe the variables Malic acid, Ash, Alcalinity of Ash, Magnesium,
# Color intensity, Hue seem to have outliers.
```

## 3. Multivariate EDA

In [ ]:

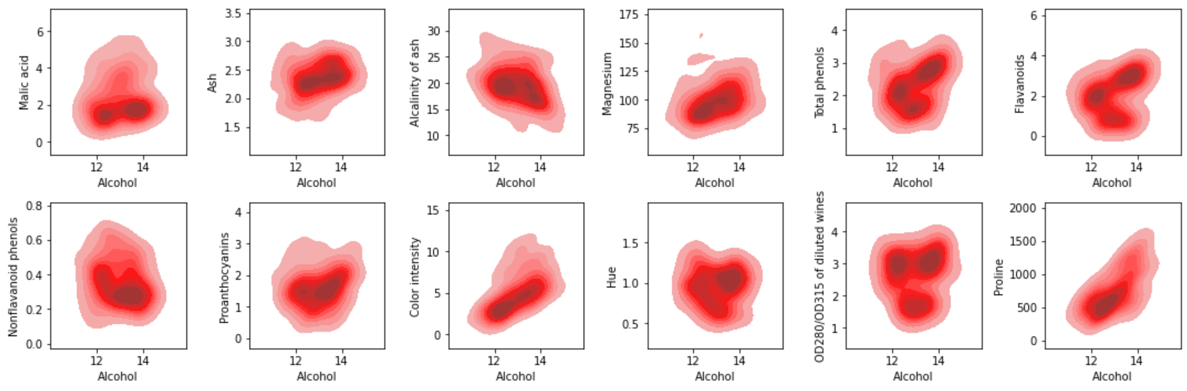
```
rows=2
cols=6
fig,ax=plt.subplots(nrows=rows,ncols=cols,figsize=(15,5))
col=wine.columns
index=0
for i in range(rows):
    for j in range(cols):
        sns.kdeplot(x=wine["Alcohol"],y=wine[col[index+1]],ax=ax[i][j],
                    shade=True, color='red')
        index=index+1
plt.tight_layout()
```

In [23]:

```
#Alcohol
rows=2
cols=6
fig,ax=plt.subplots(nrows=rows,ncols=cols,figsize=(15,5))
col=wine.columns
index=0
for i in range(rows):
    for j in range(cols):
        sns.kdeplot(x=wine["Alcohol"],y=wine[col[index+1]],ax=ax[i][j],
                    shade=True, color='red')
        index=index+1
plt.tight_layout()

'''From the bivariate Kdeplots it can be said that there might be 2 or 3 clusters.'''
```

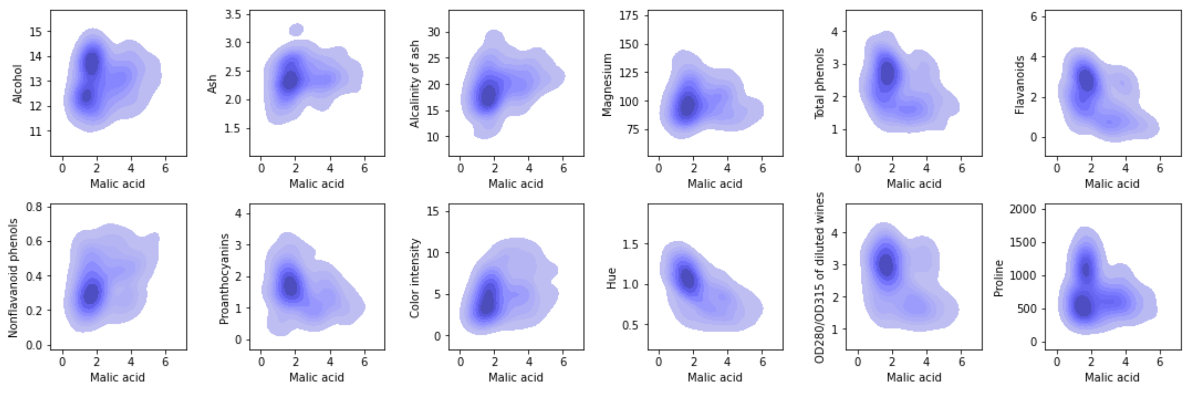
Out[23]:

```
'From the bivariate Kdeplots it can be said that there might be 2 or 3 clust
ers.'
```



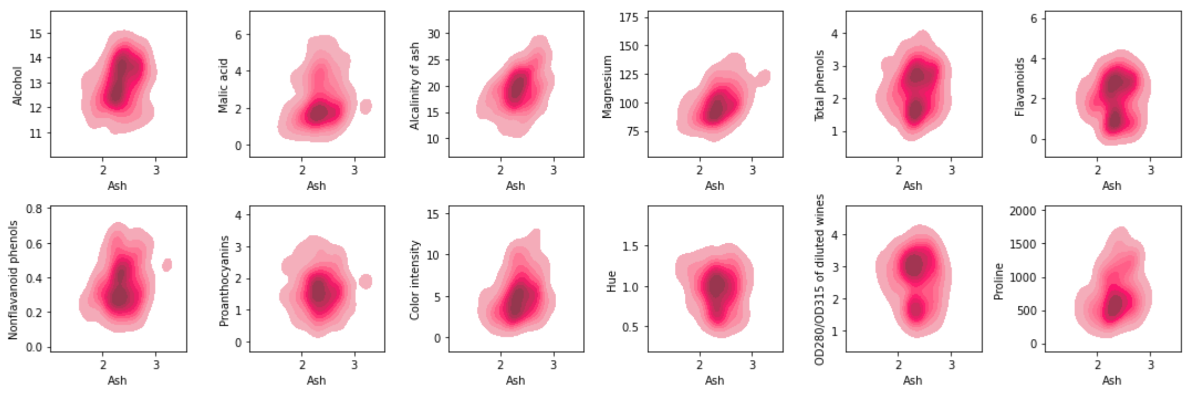
In [24]:

```
#Malic Acid
rows=2
cols=6
fig,ax=plt.subplots(nrows=rows,ncols=cols,figsize=(15,5))
col=wine.columns
index=0
for i in range(rows):
    for j in range(cols):
        if index==0:
            sns.kdeplot(x=wine["Malic acid"],y=wine[col[index]],ax=ax[i][j],
                    shade=True, color='blue')
        else:
            sns.kdeplot(x=wine["Malic acid"],y=wine[col[index+1]],ax=ax[i][j],
                    shade=True, color='blue')
        index=index+1
plt.tight_layout()
```



In [25]:

```
#Ash
rows=2
cols=6
fig,ax=plt.subplots(nrows=rows,ncols=cols,figsize=(15,5))
col=wine.columns
index=0
for i in range(rows):
    for j in range(cols):
        if index in range(0,2):
            sns.kdeplot(x=wine["Ash"],y=wine[col[index]],ax=ax[i][j],
                    shade=True, color='pink')
        else:
            sns.kdeplot(x=wine["Ash"],y=wine[col[index+1]],ax=ax[i][j],
                    shade=True, color='pink')
        index=index+1
plt.tight_layout()
```

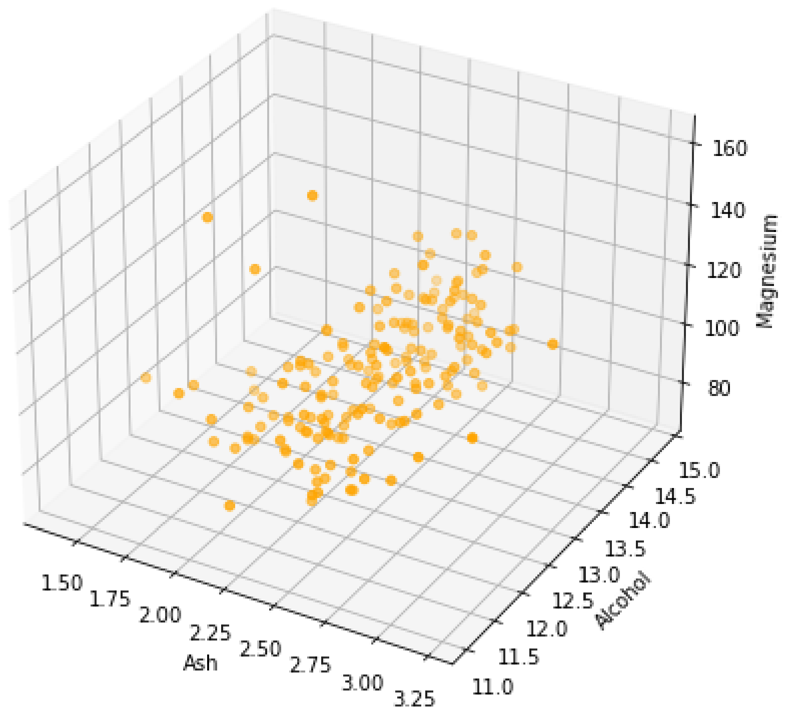


In [ ]:

```
# From the bivariate Kdeplots it can be said that there might be two or at maximum 3 cluste
```

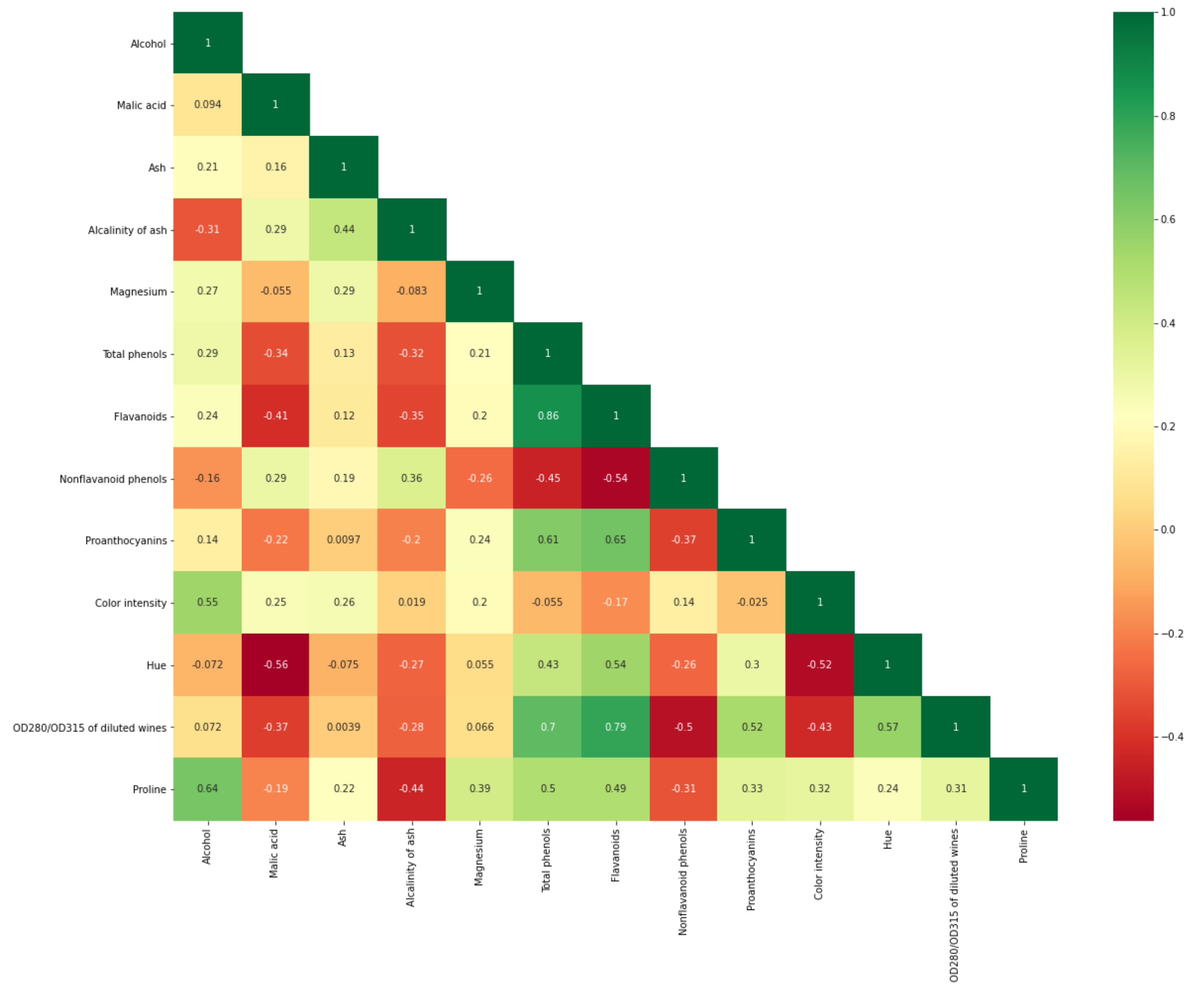
In [26]:

```
fig=plt.figure(figsize=(10,7))
ax=plt.axes(projection='3d')
ax.scatter(wine['Ash'],wine['Alcohol'],wine['Magnesium'], c='orange');
ax.set_xlabel('Ash')
ax.set_ylabel('Alcohol')
ax.set_zlabel("Magnesium");
```



In [27]:

```python
plt.figure(figsize=(20,15))
mask = np.triu(np.ones_like(wine.corr(), dtype=bool), k=1)
sns.heatmap(wine.corr(), annot=True, cmap='RdYlGn', mask=mask);

#there seems to be a high correlation between Flavanoids and total phenols
#and Flavanoids and OD280/OD315 of diluted wines.
```



In [28]:

```
for a in range(len(wine.corr().columns)):
    for b in range(a):
        if abs(wine.corr().iloc[a,b]>0.7):
            name=wine.corr().columns[a]
            print(name)
```

```
Flavanoids
OD280/OD315 of diluted wines
```

- Removing outliers

In [29]:

```
def outliers(df, ft):
    Q1=df[ft].quantile(0.25)
    Q3=df[ft].quantile(0.75)
    IQR=Q3-Q1
    lower_bound=Q1-(IQR*1.5)
    upper_bound=Q3+(IQR*1.5)
    ls=df.index[ (df[ft]<lower_bound)| (df[ft]>upper_bound) ]
    return ls
```

In [30]:

```
index_list=[]
for feature in wine.keys():
    index_list.extend(outliers(wine, feature))
print(index_list, end='')
```

```
[123, 137, 173, 25, 59, 121, 59, 73, 121, 127, 69, 73, 78, 95, 95, 110, 151,
158, 159, 166, 115]
```

In [31]:

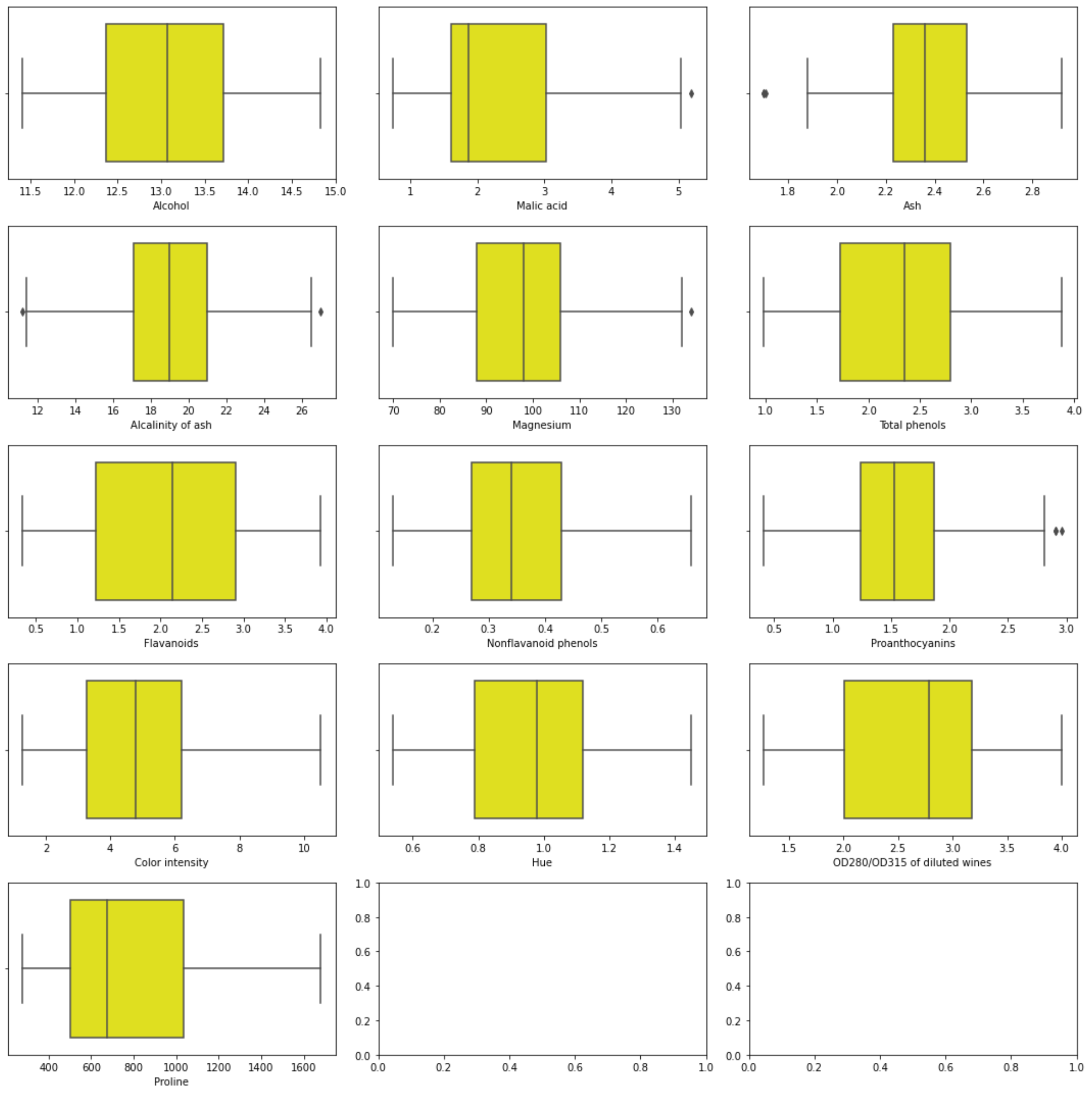
```
def remove(df, ls):
    ls=sorted(set(ls))
    df=df.drop(ls)
    return df
wine_cleaned=remove(wine, index_list)
wine_cleaned.shape
```

Out[31]:

```
(161, 13)
```

In [32]:

```
rows=5
cols=3
fig,ax=plt.subplots(nrows=rows,ncols=cols,figsize=(15,15))
col=wine_cleaned.columns
index=0
for i in range(rows):
    for j in range(cols):
        sns.boxplot(wine_cleaned[col[index]],ax=ax[i][j], color='yellow')
        index=index+1
        if index>=12:
            break
plt.tight_layout()
```

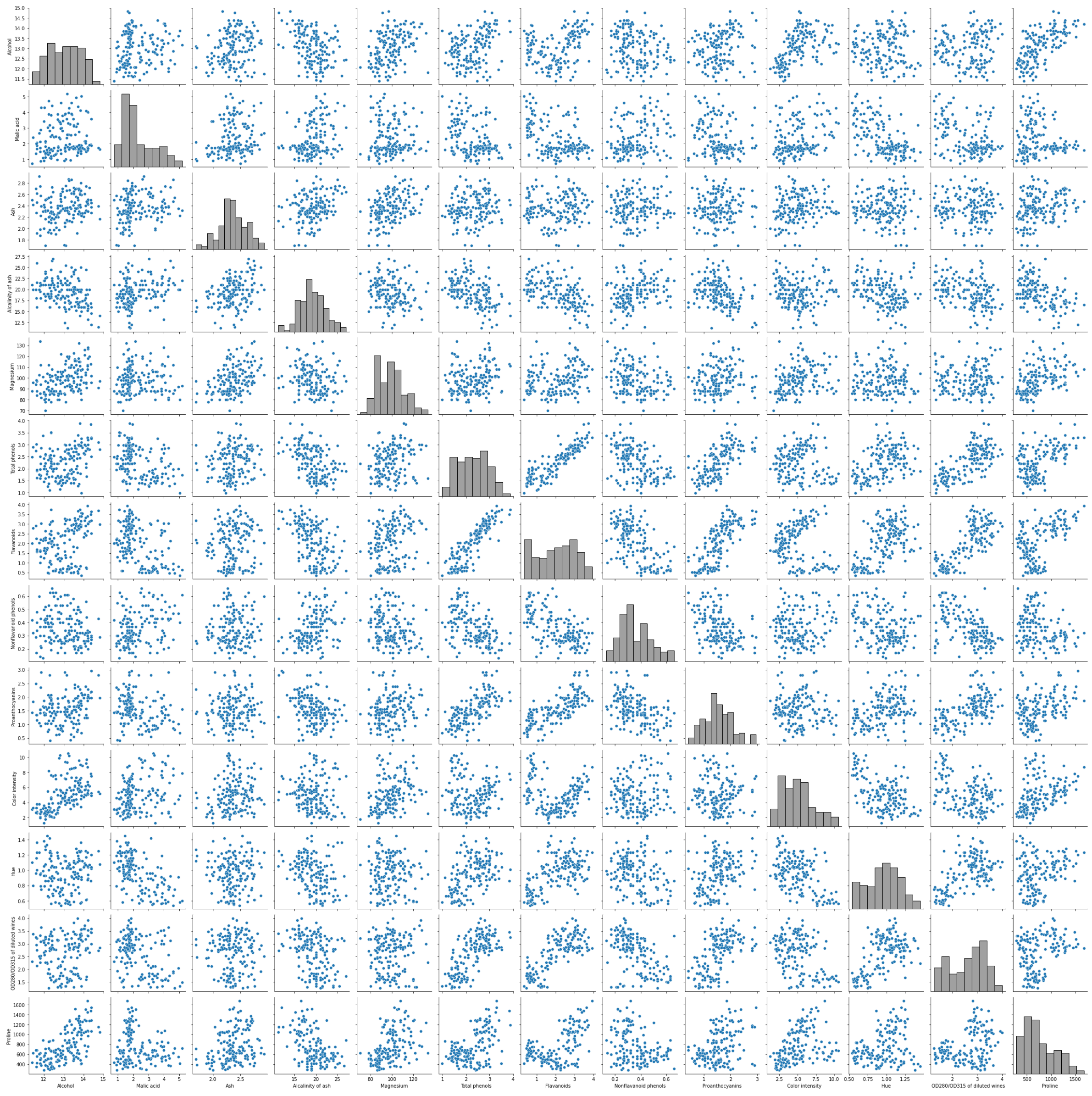


In [33]:

```
sns.pairplot(wine_cleaned, diag_kws={'color':'grey'})
```

Out[33]:

```
<seaborn.axisgrid.PairGrid at 0x1db533469a0>
```



## Principal Component Analysis

In [34]:

```
x=wine_cleaned

#standardisation
from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
x_scaled=pd.DataFrame(sc.fit_transform(x), columns=x.columns)
x_scaled.head(2)
```

Out[34]:

| | Alcohol | Malic acid | Ash | Alcalinity of ash | Magnesium | Total phenols | Flavanoids | Nonflavanoid phenols |
|---|---|---|---|---|---|---|---|---|
| **0** | 1.495711 | -0.571130 | 0.277425 | -1.208619 | 2.309259 | 0.808420 | 1.034224 | -0.643463 |
| **1** | 0.200698 | -0.504145 | -0.928625 | -2.672685 | 0.118914 | 0.570475 | 0.728906 | -0.808165 |

In [35]:

```
#PCA
from sklearn.decomposition import PCA
pca=PCA()
x_pca1=pd.DataFrame(pca.fit_transform(x_scaled), columns=x_scaled.columns)
x_pca1.head(2)
```

Out[35]:

| | Alcohol | Malic acid | Ash | Alcalinity of ash | Magnesium | Total phenols | Flavanoids | Nonflavanoid phenols |
|---|---|---|---|---|---|---|---|---|
| **0** | 3.482298 | -1.494604 | 0.133281 | -0.171033 | 0.650584 | -0.512913 | -0.733783 | 0.342958 |
| **1** | 2.278164 | 0.375183 | -1.873077 | -0.672753 | 0.530476 | -1.217015 | -0.049324 | -0.893003 |

In [36]:

```
pca.explained_variance_ratio_

#The first 5 Principal Components are capturing around 80% of the variance so we can replac
#with the new 5 features having 80% of the information.
#So, we have reduced the 11 dimensions to only 5 dimensions while retaining most of the inf
```

Out[36]:

```
array([0.38853448, 0.20487624, 0.09547051, 0.06919384, 0.05789018,
       0.04289108, 0.03510666, 0.0259047 , 0.02394086, 0.01980589,
       0.0175546 , 0.01308515, 0.00574581])
```

In [37]:

```
#Dimentionality Reduction
pca=PCA(n_components=5)
x_pca2=pd.DataFrame(pca.fit_transform(x_scaled), columns=['PC1','PC2','PC3','PC4','PC5'])
x_pca2.head(2)
```

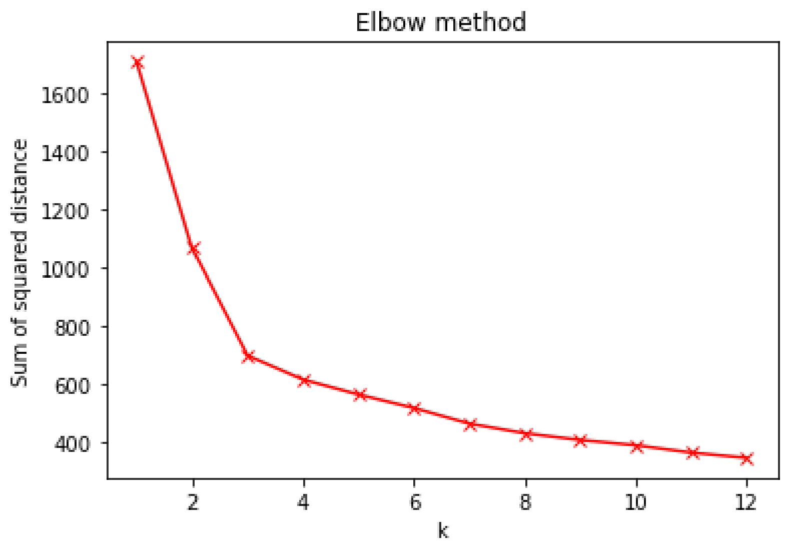
Out[37]:

| | PC1 | PC2 | PC3 | PC4 | PC5 |
|---|---|---|---|---|---|
| **0** | 3.482298 | -1.494604 | 0.133281 | -0.171033 | 0.650584 |
| **1** | 2.278164 | 0.375183 | -1.873077 | -0.672753 | 0.530476 |

**1. K means clustering**

In [38]:

```
from sklearn.cluster import KMeans
ssd=[]
K=range(1,13)
for k in K:
    km=KMeans(n_clusters=k)
    km=km.fit(x_pca2)
    ssd.append(km.inertia_)

plt.plot(K, ssd, 'rx-');
plt.xlabel('k');
plt.ylabel('Sum of squared distance');
plt.title('Elbow method');
```



In [ ]:

```
# Thus, we choose no. of clusters to be 3.
```

In [40]:

```
km=KMeans(n_clusters=3,init='k-means++', random_state=0)
km.fit(x_pca2)
```

Out[40]:

```
KMeans(n_clusters=3, random_state=0)
```

In [41]:

```
centroids=km.cluster_centers_
centroids
```

Out[41]:

```
array([[ 2.38884698, -0.88560974, -0.02554146, -0.08793572, -0.0877621 ],
       [-2.78775617, -1.2489108 , -0.29375436,  0.08181669,  0.03935514],
       [-0.22593271,  1.85459226,  0.25345433,  0.02445725,  0.05722795]])
```

In [42]:

```
km.inertia_
```

Out[42]:

```
699.8420783063123
```

In [43]:

```
labels=km.labels_
labels
```

Out[43]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1])
```

In [44]:

```
from collections import Counter
Counter(labels)
```

Out[44]:

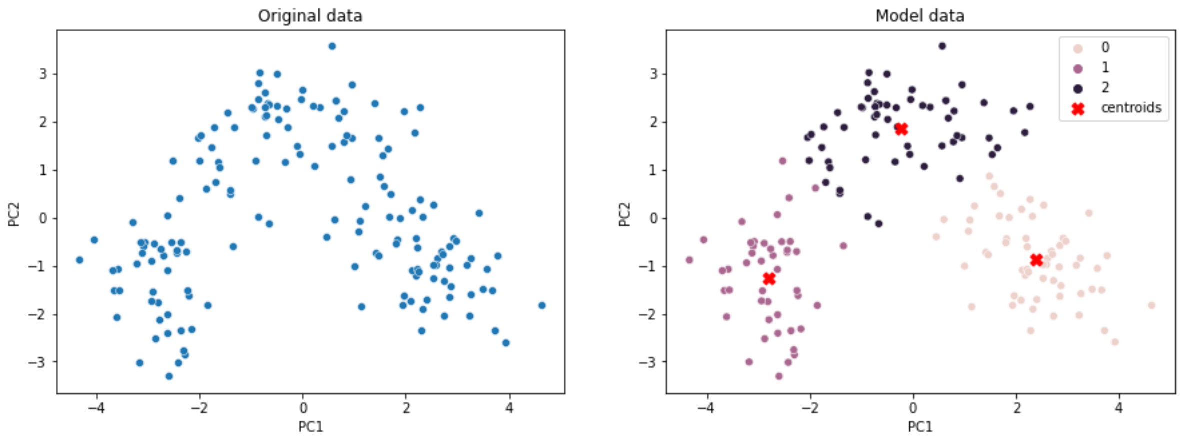
```
Counter({0: 58, 2: 58, 1: 45})
```

In [45]:

```
#Visualising the clusters using scatter plot

plt.figure(figsize=(15,5))
ax1=plt.subplot(1,2,1)
ax1.set_title('Original data')
sns.scatterplot(data = x_pca2, x='PC1', y='PC2', ax=ax1);
ax2=plt.subplot(1,2,2)
ax2.set_title('Model data')
sns.scatterplot(data = x_pca2, x='PC1', y='PC2', hue=km.labels_, ax=ax2);
plt.scatter(centroids[:, 0], centroids[:, 1], marker="X", c="r", s=80, label="centroids");
plt.legend()
```

Out[45]:

```
<matplotlib.legend.Legend at 0x1db5e0c4250>
```

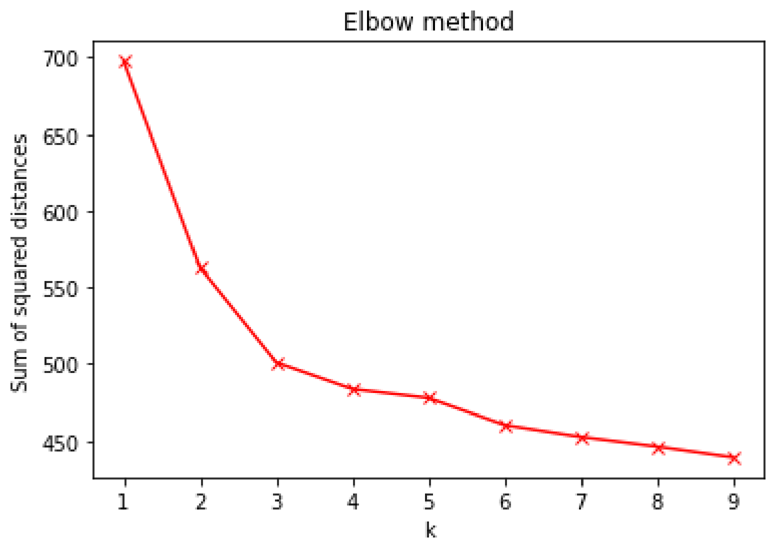


In [47]:

```
from sklearn_extra.cluster import KMedoids
ssd=[]
for k in range(1,10):
    kmed=KMedoids(n_clusters=k, max_iter=300, random_state=1)
    kmed.fit(scaled_wine)
    ssd.append(kmed.inertia_)

plt.plot(range(1,10), ssd, 'rx-');
plt.xlabel('k')
plt.ylabel("Sum of squared distances");
plt.title("Elbow method")
```

Out[47]:

```
Text(0.5, 1.0, 'Elbow method')
```



**2. K Medoids**

In [48]:

```
from sklearn.decomposition import PCA
pca=PCA(n_components=2)
pc=pd.DataFrame(pca.fit_transform(scaled_wine), columns=['PC1', 'PC2'])
pc.head()
```

Out[48]:

| | PC1 | PC2 |
|---|---|---|
| **0** | 3.316751 | -1.443463 |
| **1** | 2.209465 | 0.333393 |
| **2** | 2.516740 | -1.031151 |
| **3** | 3.757066 | -2.756372 |
| **4** | 1.008908 | -0.869831 |

In [58]:

```
kmed=KMedoids(n_clusters=3, random_state=0)
kmed.fit(pc)
'''wine['Labels']=kmed.predict(scaled_wine)
wine.head(2)'''
```

Out[58]:

```
"wine['Labels']=kmed.predict(scaled_wine)\nwine.head(2)"
```

In [59]:

```
centroids=kmed.cluster_centers_
centroids
```

Out[59]:

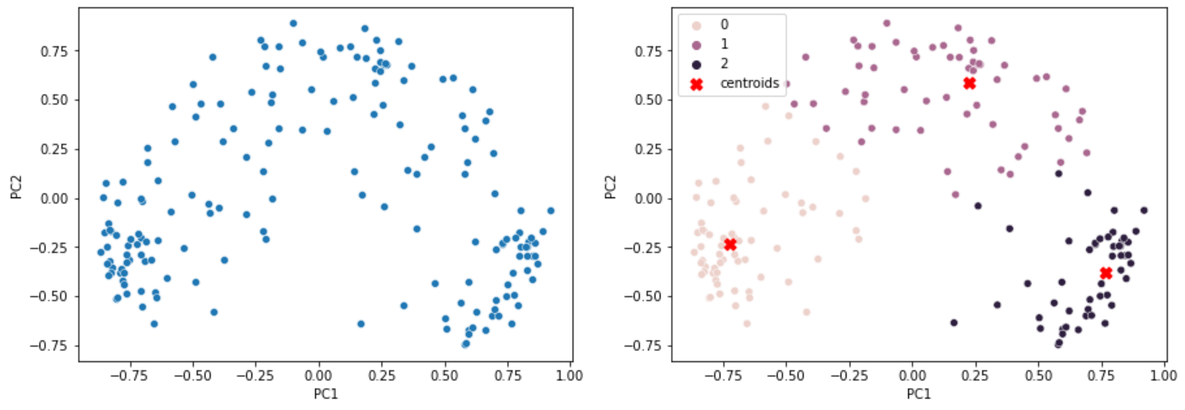
```
array([[-0.72328949, -0.23681583],
       [ 0.2229686 ,  0.58496601],
       [ 0.76943894, -0.38281809]])
```

In [60]:

```
plt.figure(figsize=(15,5))
ax1=plt.subplot(1,2,1)
sns.scatterplot(data = pc, x='PC1', y='PC2', ax=ax1);

ax2=plt.subplot(1,2,2)
sns.scatterplot(data = pc, x='PC1', y='PC2',
                ax=ax2, hue=kmed.labels_)
plt.scatter(centroids[:, 0], centroids[:, 1],
            marker="X", c="r", s=80, label="centroids");
plt.legend()
```

Out[60]:

```
<matplotlib.legend.Legend at 0x1db5f8395e0>
```

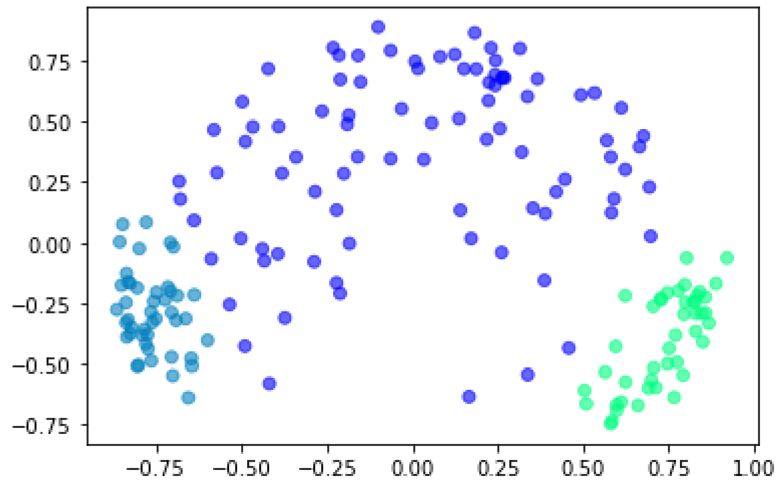


**3. Gaussian Mixture clustering**

In [61]:

```
from sklearn.preprocessing import StandardScaler, normalize
sc=StandardScaler()
scaled_wine=sc.fit_transform(wine)

normalised_wine=pd.DataFrame(normalize(scaled_wine))

pca=PCA(n_components=2)
pc=pd.DataFrame(pca.fit_transform(normalised_wine), columns=['PC1', 'PC2'])
print(pc.head(2))

from sklearn.mixture import GaussianMixture
gmm=GaussianMixture(n_components=3)
gmm.fit(pc)
labels=GaussianMixture(n_components=3).fit_predict(pc)

plt.scatter(pc['PC1'], pc['PC2'],
           c = labels, cmap =plt.cm.winter, alpha = 0.6)
plt.show()
```

```
        PC1       PC2
0 -0.832433 -0.318834
1 -0.639443  0.091947
```

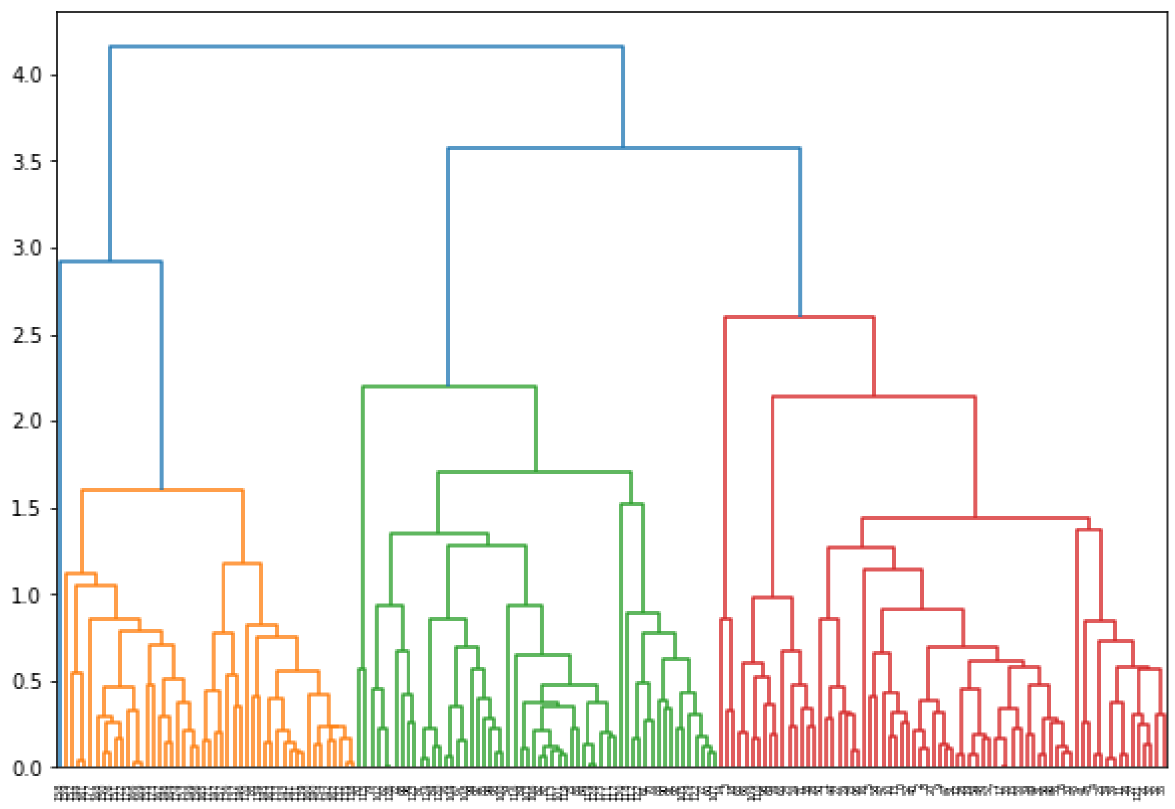


**4. Hierarchical Clustering — Agglomerative Clustering :**

In [62]:

```
from sklearn.preprocessing import StandardScaler, normalize
sc=StandardScaler()
scaled_wine=sc.fit_transform(wine)

pca=PCA(n_components=2)
pc=pd.DataFrame(pca.fit_transform(scaled_wine), columns=['PC1', 'PC2'])
print(pc.head(2))

from scipy.cluster.hierarchy import dendrogram, linkage
plt.figure(figsize=(30,25))
link=linkage(pc, method='centroid')
plt.figure(figsize=(10, 7))
dendrogram(link);
```

```
        PC1       PC2
0  3.316751 -1.443463
1  2.209465  0.333393

<Figure size 2160x1800 with 0 Axes>
```

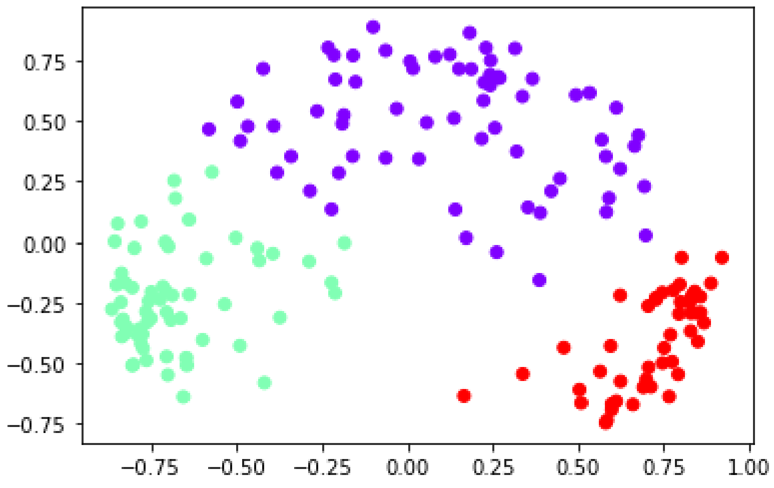


In [69]:

```
from sklearn.cluster import AgglomerativeClustering
agg=AgglomerativeClustering(n_clusters=3, affinity='euclidean', linkage='ward')
agg.fit_predict(pc)
plt.scatter(pc['PC1'], pc['PC2'], c=agg.labels_, cmap='rainbow')
```

Out[69]:

```
<matplotlib.collections.PathCollection at 0x1db5ff68ca0>
```



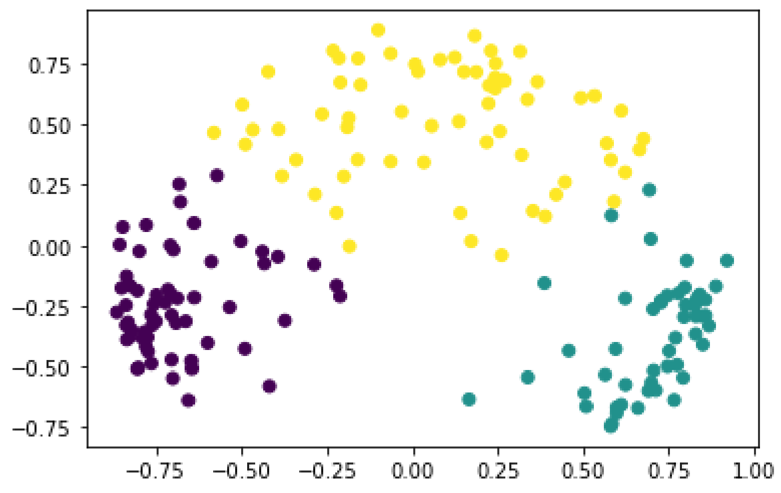
**5. Spectral Clustering**

In [70]:

```python
from sklearn.preprocessing import StandardScaler, normalize
sc=StandardScaler()
scaled_wine=sc.fit_transform(wine)

normalised_wine=normalize(scaled_wine)

from sklearn.decomposition import PCA
pca=PCA(n_components=2)
pc=pd.DataFrame(pca.fit_transform(normalised_wine), columns=['PC1', 'PC2'])

from sklearn.cluster import SpectralClustering
sp=SpectralClustering(n_clusters=3, affinity ='rbf')

labels = sp.fit_predict(normalised_wine)

plt.scatter(pc['PC1'],pc['PC2'], c=labels)
```

Out[70]:

```
<matplotlib.collections.PathCollection at 0x1db5ffd40a0>
```



0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00