APSSDC

Andhra Pradesh State Skill Development Corporation

(https://apssdc.in)

Day09 Dimensionality Reduction

Principal Component Analysis (PCA)

Brief primer and history

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation (https://en.wikipedia.org/wiki/Orthogonal_transformation) to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated (https://en.wikipedia.org/wiki/Correlation_and_dependence) 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 (https://en.wikipedia.org/wiki/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 (https://en.wikipedia.org/wiki/Orthogonal) the preceding components. The resulting vectors are an uncorrelated orthogonal basis set (https://en.wikipedia.org/wiki/Orthogonal_basis_set).

PCA is sensitive to the relative scaling of the original variables.

PCA was invented in 1901 by Karl Pearson (https://en.wikipedia.org/wiki/Karl_Pearson) as an analogue of the principal axis theorem in mechanics; it was later independently developed and named by Harold Hotelling (https://en.wikipedia.org/wiki/Harold_Hotelling) in the 1930s.

Dataset_Link (https://raw.githubusercontent.com/AP-State-Skill-Development-Corporation/Datasets/master/wine.data.csv)

class of wine Wine dataset --> Reducing the dimensions --> unsupervised

Data Exploration

In [3]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:

```
df = pd.read_csv('https://raw.githubusercontent.com/AP-State-Skill-Development-Corporation/
```

In [5]:

```
df.head()
```

Out[5]:

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

In [6]:

```
df.shape
```

Out[6]:

```
(178, 14)
```

In [7]:

```
df.columns
```

Out[7]:

```
Index(['Class', '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')
```

0. Class - The type of wine, into one of three classes, 1 (59 obs), 2(71 obs), and 3 (48 obs)
1. Alcohol -
2. Malic acid
3. Ash
4. Alcalinity of ash
5. Magnesium
6. Total phenols
7. Flavanoids
8. Nonflavanoid phenols
9. Proanthocyanins
10. Color intensity
11. Hue

12. OD280/OD315 of diluted wines
13. Proline

In [8]:

```
df['Class'].value_counts() ## Class is the output
```

Out[8]:

```
2    71
1    59
3    48
Name: Class, dtype: int64
```

In [9]:

```
df.isnull().sum()
```

Out[9]:

```
Class                         0
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 [10]:

```
df.duplicated().sum()
```

Out[10]:

```
0
```

In [11]:

```
df.dtypes
```

Out[11]:

```
Class                            int64
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]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Class                         178 non-null    int64  
 1   Alcohol                       178 non-null    float64
 2   Malic acid                    178 non-null    float64
 3   Ash                           178 non-null    float64
 4   Alcalinity of ash             178 non-null    float64
 5   Magnesium                     178 non-null    int64  
 6   Total phenols                 178 non-null    float64
 7   Flavanoids                    178 non-null    float64
 8   Nonflavanoid phenols          178 non-null    float64
 9   Proanthocyanins               178 non-null    float64
 10  Color intensity               178 non-null    float64
 11  Hue                           178 non-null    float64
 12  OD280/OD315 of diluted wines  178 non-null    float64
 13  Proline                       178 non-null    int64  
dtypes: float64(11), int64(3)
memory usage: 19.6 KB
```

In [13]:

```
df.describe()
```

Out[13]:

	Class	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Fl
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	17
mean	1.938202	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	
std	0.775035	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	
min	1.000000	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	
25%	1.000000	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	
50%	2.000000	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	
75%	3.000000	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	
max	3.000000	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	

- First step "decorrelation"
- Second step reduces dimension

In [15]:

```
data = df.drop('Class', axis = 'columns')
data.head()
```

...

In [16]:

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

In [17]:

```
decorr = StandardScaler()

Decodata = decorr.fit_transform(data)
```

In [25]:

```
help(pca)
```

```
Help on PCA in module sklearn.decomposition._pca object:

class PCA(sklearn.decomposition._base._BasePCA)
 |  PCA(n_components=None, *, copy=True, whiten=False, svd_solver='auto',
tol=0.0, iterated_power='auto', random_state=None)
 |
 |  Principal component analysis (PCA).
 |
 |  Linear dimensionality reduction using Singular Value Decomposition of
the
 |  data to project it to a lower dimensional space. The input data is cen
tered
 |  but not scaled for each feature before applying the SVD.
 |
 |  It uses the LAPACK implementation of the full SVD or a randomized trun
cated
 |  SVD by the method of Halko et al. 2009, depending on the shape of the
input
 |  data and the number of components to extract.
```

In [18]:

```
from sklearn.decomposition import PCA
```

In [33]:

```
pca = PCA()
```

In [34]:

```
pca.fit(Decodata)
```

Out[34]:

```
PCA()
```

In [35]:

```
len(pca.explained_variance_ratio_)
```

Out[35]:

```
13
```

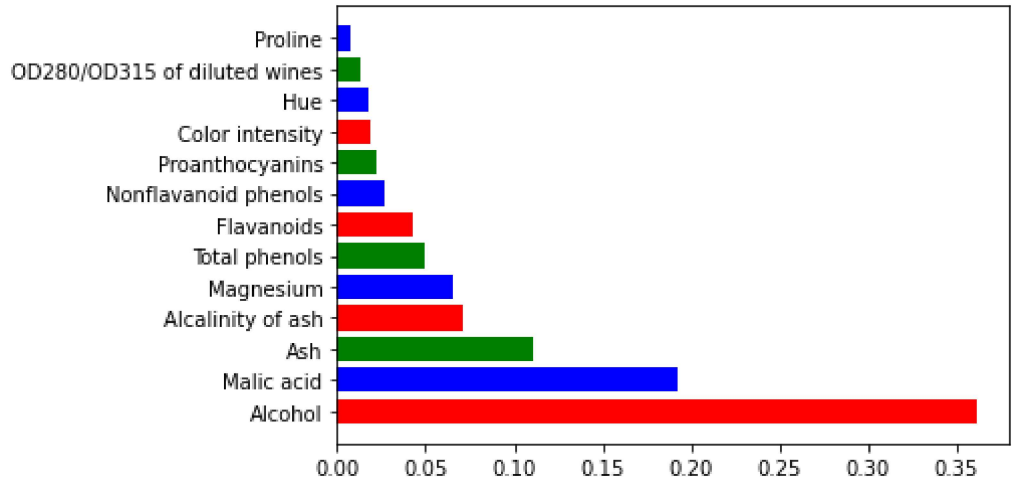
In [39]:

```
import matplotlib.pyplot as plt

plt.barh(data.columns, pca.explained_variance_ratio_, color = ['r','b','g','r','b','g','r',
```

Out[39]:

```
<BarContainer object of 13 artists>
```



In [38]:

```
help(plt.barh)
```

```
Help on function barh in module matplotlib.pyplot:

barh(y, width, height=0.8, left=None, *, align='center', **kwargs)
    Make a horizontal bar plot.

    The bars are positioned at *y* with the given *align*\ment. Their
    dimensions are given by *width* and *height*. The horizontal baseline
    is *left* (default 0).

    Each of *y*, *width*, *height*, and *left* may either be a scalar
    applying to all bars, or it may be a sequence of length N providing a
    separate value for each bar.

    Parameters
    ----------
    y : scalar or array-like
        The y coordinates of the bars. See also *align* for the
        alignment of the bars to the coordinates.
```

In [44]:

```
plt.scatter(Decodata[:,0], Decodata[:,1], c = df['Class'])
```

Out[44]:

```
<matplotlib.collections.PathCollection at 0x2471fcbc8b0>
```