UnSupervised Learning Algorithm - Wine Color Prediction with Agglomerative Clustering

The agglomerative clustering is the most common type of hierarchical clustering used to group objects in clusters based on their similarity. It’s also known as AGNES (Agglomerative Nesting). The algorithm starts by treating each object as a singleton cluster. Next, pairs of clusters are successively merged until all clusters have been merged into one big cluster containing all objects. The result is a tree-based representation of the objects, named dendrogram.We are using a dataset which cointains chemical properties (volatile_acidity , total_sulphur_dioxide etc) to determine wine color

In [64]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
warnings.simplefilter('ignore')
```

In [65]:

```
data = pd.read_csv("data/Wine_Quality_Data.csv")
```

In [66]:

```
print(data.shape)
```

```
(6497, 13)
```

In [67]:

```
#Print no of integers, floats and strings
data.dtypes.value_counts()
```

Out[67]:

```
float64    11
int64       1
object      1
dtype: int64
```

In [68]:

```
#Data should be numerical
data.head()
```

Out[68]:

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur
0	7.4	0.70	0.00	1.9	0.076	11.0	
1	7.8	0.88	0.00	2.6	0.098	25.0	
2	7.8	0.76	0.04	2.3	0.092	15.0	
3	11.2	0.28	0.56	1.9	0.075	17.0	
4	7.4	0.70	0.00	1.9	0.076	11.0	

In [69]:

```
#Print no of entries for each color
data.color.value_counts()
```

Out[69]:

```
white    4898
red      1599
Name: color, dtype: int64
```

In [70]:

```
#Print % of each colors
data.color.value_counts(normalize=True)
```

Out[70]:

```
white    0.753886
red      0.246114
Name: color, dtype: float64
```

Preprocessing Steps

1. Select Features and apply feature tranformation/scaling.

In [71]:

```
#Removing Color and Quality from features.
float_columns = [x for x in data.columns if x not in ['color', 'quality']]

# The correlation matrix
corr_mat = data[float_columns].corr()

#Every feature with itself will have correlation of one and we need to remo
for x in range(len(float_columns)):
    corr_mat.iloc[x,x] = 0.0

# max correlations(fixed_acidity,volatile_acidity has max co-relation )
corr_mat.abs().max()
```

Out[71]:

```
fixed_acidity           0.458910
volatile_acidity        0.414476
citric_acid             0.377981
residual_sugar          0.552517
chlorides               0.395593
free_sulfur_dioxide     0.720934
total_sulfur_dioxide    0.720934
density                 0.686745
pH                      0.329808
sulphates               0.395593
alcohol                 0.686745
dtype: float64
```

In [72]:

```
#Calculate Skew Vlaues
#0- no skew
#+ve - right skew
#-Ve - left skew
skew_columns = (data[float_columns]
                .skew()
                .sort_values(ascending=False))
skew_columns
```

Out[72]:

```
chlorides               5.399828
sulphates               1.797270
fixed_acidity           1.723290
volatile_acidity        1.495097
residual_sugar          1.435404
free_sulfur_dioxide     1.220066
alcohol                 0.565718
density                 0.503602
citric_acid             0.471731
pH                      0.386839
total_sulfur_dioxide   -0.001177
dtype: float64
```

In [73]:

```
#Getting Skewed Columns and log tranforming it.
skew_columns = skew_columns.loc[skew_columns > 0.75]
# Perform log transform on skewed columns
for col in skew_columns.index.tolist():
    data[col] = np.log1p(data[col])
```

2. Normalize Features .

In [74]:

```
from sklearn.preprocessing import StandardScaler
data[float_columns] = StandardScaler().fit_transform(data[float_columns])
data.head(4)
```

Out[74]:

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur
0	0.229509	2.135767	-2.192833	-0.815173	0.624554	-1.193601	-
1	0.550261	3.012817	-2.192833	-0.498175	1.281999	-0.013944	-
2	0.550261	2.438032	-1.917553	-0.625740	1.104012	-0.754684	-
3	2.802728	-0.337109	1.661085	-0.815173	0.594352	-0.574982	-

Modeling with Agglomerative Hierarchial Clustering

In [75]:

```
from sklearn.cluster import AgglomerativeClustering
### BEGIN SOLUTION
ag = AgglomerativeClustering(n_clusters=2, linkage='ward', compute_full_tre
ag = ag.fit(data[float_columns])
labels = ag.labels_
print(labels)
```

```
[1 1 1 ... 0 0 0]
```

In [76]:

```
#Assigning labels generated by K-means to our original dataset
data['agglom'] = labels
data.head(5)
```

Out[76]:

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur
0	0.229509	2.135767	-2.192833	-0.815173	0.624554	-1.193601	-
1	0.550261	3.012817	-2.192833	-0.498175	1.281999	-0.013944	-
2	0.550261	2.438032	-1.917553	-0.625740	1.104012	-0.754684	-
3	2.802728	-0.337109	1.661085	-0.815173	0.594352	-0.574982	-
4	0.229509	2.135767	-2.192833	-0.815173	0.624554	-1.193601	-

In [77]:

```
data.groupby('agglom').mean()
```

Out[77]:

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total
agglom							
0	-0.274576	-0.387664	0.089931	0.203975	-0.371151	0.320495	
1	0.768043	1.084371	-0.251555	-0.570559	1.038182	-0.896487	

In [78]:

```
#Without giving labels K-Means has created two clusters and lets examine ho
(data[['color','agglom']]
 .groupby(['agglom','color'])
 .size()
 .to_frame()
 .rename(columns={0:'number'}))
```

Out[78]:

		number
agglom	**color**	
0	**red**	31
	white	4755
1	**red**	1568
	white	143

Agglomerative Clustering could classify into two clusters one with having red as majority and another with having white has majority.

How to Plot dendrogram created by agglomerative clustering?

In [79]:

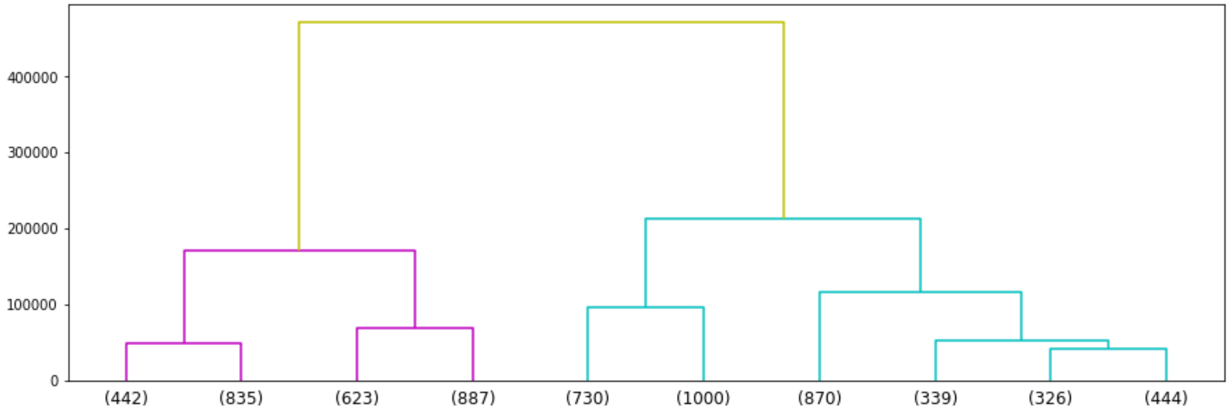
```
# Import the cluster hierarchy module from SciPy to obtain the linkage and
from scipy.cluster import hierarchy

Z = hierarchy.linkage(ag.children_, method='ward')

fig, ax = plt.subplots(figsize=(15,5))

hierarchy.set_link_color_palette(['m', 'c', 'y', 'k'])

den = hierarchy.dendrogram(Z, orientation='top',
                           p=10, truncate_mode='lastp',
                           show_leaf_counts=True, ax=ax,
                           above_threshold_color='y')
### END SOLUTION
```



Summary

Without providing labels Agglomerative Clustering can classify wine to red and white with one cluster having majority red and another with white .A dendrogram is a diagram representing a tree. The figure factory called create_dendrogram performs hierarchical clustering on data and represents the resulting tree

In [ ]: