Machine Learning Lab 8

K-Means Clustering and KNN

K-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.The algorithm starts by randomly choosing a centroid value for each cluster. After that the algorithm iteratively performs three steps:
(i) Find the Euclidean distance between each data instance and centroids of all the clusters
(ii) Assign the data instances to the cluster of the centroid with nearest distance
(iii) Calculate new centroid values based on the mean values of the coordinates of all the data instances from the corresponding cluster.

KNN is the abbreviation for K-Nearest Neighbours. KNN algorithm is one of the simplest classification algorithm and it is one of the most used learning algorithms. it is a non-parametric, lazy learning algorithm. With non-parametric we mean that it does not make any assumptions on the underlying data distribution. The output of a KNN algorithms is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its K nearest neighbors

The dataset

The dataset used to perform this experiment is the wine quality dataset, it is a combination of data on two types of wine variants, namely red wine and white wine, of the portuguese “Vinho Verde” wine. The dataset contains information on the parameters for fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol.

Experiment

In this experiment I used the sklearn’s K-Means and KNN algorithms to predict the quality of a wine.

Using the pandas library in I loaded the red wine and white wine datasets into the memory from their respective csv files and then merged the two datasets into one single pandas dataframe.

Using the pandas.Dataframe.describe() function in pandas I calculated the various statistical measures of each of the columns of the dataset.

For performing the experiment I started with scaling the features in the dataset. Since the features of the dataset each have a different scale we need to bring them to the same scale so that none of the features dominates the others features while training the models.

Next ran the K-Means algorithm on the data for the number of centroids in the range of 1 to 20 to find the the number where the elbow point occurred. As evident in the notebook it occurs at K=5 or 6, choosing K=5 I plotted the ingridients of the wine pair wise. From which we conclude that:

Cluster 1: Low pH, high sulphates, low alcohol
Cluster 2: High pH, low sulphates, high alcohol, low total sulpur dioxide
Cluster 3: Low alcohol, low sulphates, high total sulpur dioxide
Cluster 4: High alcohol, low pH, low total sulpur dioxide
Cluster 5: Low alcohol, low sulphates, low total sulphur dioxide

Thus we see that:
pH is high in cluster 2 and low in cluster 1.
Sulphates is high in cluster 1 and low in cluster 3.
Alcohol is high in cluster 2 & 4 and low in Rest of the clusters(1,3,5).
Total surfur dioxide is high in cluster 3 and low in cluster 4.

For the KNN I used a similar approach as K-Means by running the KNN algorithm over data with changing only the K value from 1 to 160 and as evident in the graph the value decrease as the K value increases. However it seems to plateau around 0.8.

The code and plots can be found in the accompanying jupyter notebook.

KMeans + KNN

November 1, 2018

1 Lab 8

2 K-Means Clustering and KNN

2.1 Submitted to: Prof. Sweetlin Hemlatha

2.2 Submitted by: Prateek Singh (15BCE1091)

In [1]:
```
import random
import numpy as np
import pandas as pd
import seaborn as sn

from sklearn.cluster import KMeans
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import precision_score, recall_score
from sklearn.neighbors import KNeighborsClassifier

import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
%matplotlib inline
```

In [2]:
```
red_wine_data = pd.read_csv('../Dataset/winequality-red.csv', sep=';')
white_wine_data = pd.read_csv('../Dataset/winequality-white.csv', sep=';')

wine_data = pd.concat([red_wine_data, white_wine_data])
bins = (2, 6.5, 10)
group_names = ['bad', 'good']
wine_data['quality'] = pd.cut(wine_data['quality'], bins = bins, labels = group_names)
wine_data.iloc[:, :11].describe()
```

Out[2]:
```
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    6497.000000       6497.000000  6497.000000     6497.000000
mean        7.215307          0.339666     0.318633        5.443235
std         1.296434          0.164636     0.145318        4.757804
min         3.800000          0.080000     0.000000        0.600000
```

```
25%        6.400000          0.230000     0.250000        1.800000
50%        7.000000          0.290000     0.310000        3.000000
75%        7.700000          0.400000     0.390000        8.100000
max       15.900000          1.580000     1.660000       65.800000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  6497.000000          6497.000000           6497.000000  6497.000000
mean      0.056034            30.525319            115.744574     0.994697
std       0.035034            17.749400             56.521855     0.002999
min       0.009000             1.000000              6.000000     0.987110
25%       0.038000            17.000000             77.000000     0.992340
50%       0.047000            29.000000            118.000000     0.994890
75%       0.065000            41.000000            156.000000     0.996990
max       0.611000           289.000000            440.000000     1.038980

                pH    sulphates      alcohol
count  6497.000000  6497.000000  6497.000000
mean      3.218501     0.531268    10.491801
std       0.160787     0.148806     1.192712
min       2.720000     0.220000     8.000000
25%       3.110000     0.430000     9.500000
50%       3.210000     0.510000    10.300000
75%       3.320000     0.600000    11.300000
max       4.010000     2.000000    14.900000
```

In [3]:
```
scaler = StandardScaler()

data = wine_data.iloc[:,:11].values
scaled_features = scaler.fit_transform(data)

wine_data_scaled = pd.DataFrame(scaled_features, index=wine_data.index, columns=wine_da
wine_data_scaled.head()
```

Out[3]:
```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0       0.142473          2.188833    -2.192833       -0.744778   0.569958
1       0.451036          3.282235    -2.192833       -0.597640   1.197975
2       0.451036          2.553300    -1.917553       -0.660699   1.026697
3       3.073817         -0.362438     1.661085       -0.744778   0.541412
4       0.142473          2.188833    -2.192833       -0.744778   0.569958

   free sulfur dioxide  total sulfur dioxide   density        pH  sulphates  \
0            -1.100140             -1.446359  1.034993  1.813090   0.193097
1            -0.311320             -0.862469  0.701486 -0.115073   0.999579
2            -0.874763             -1.092486  0.768188  0.258120   0.797958
3            -0.762074             -0.986324  1.101694 -0.363868   0.327510
4            -1.100140             -1.446359  1.034993  1.813090   0.193097

    alcohol
```

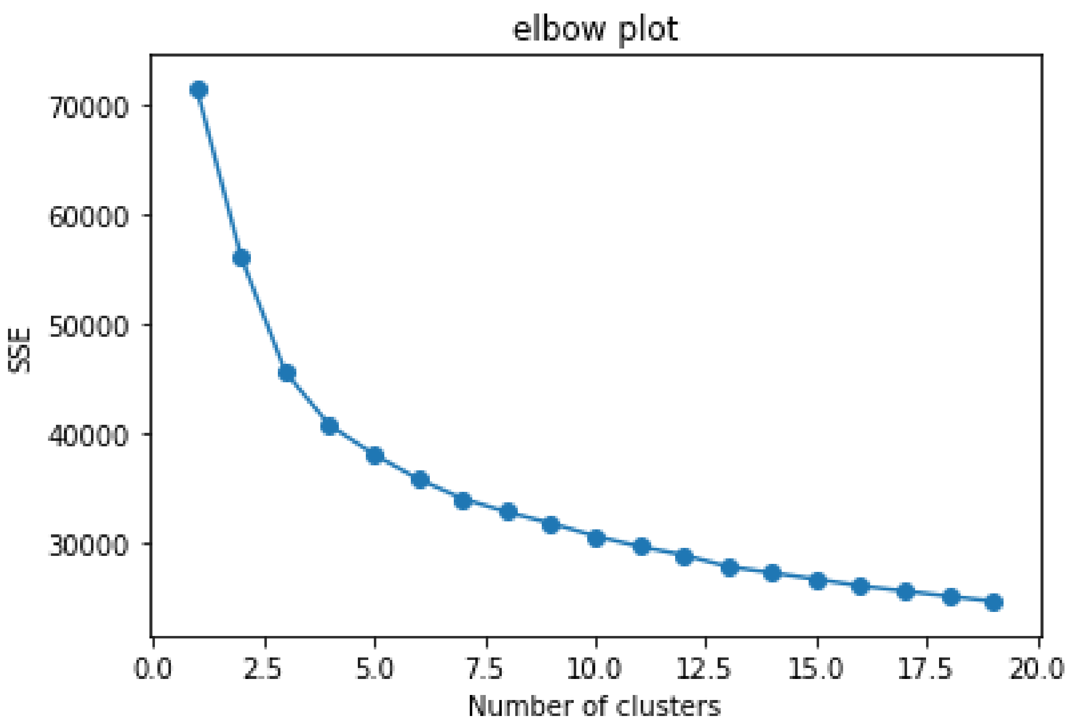
```
0 -0.915464
1 -0.580068
2 -0.580068
3 -0.580068
4 -0.915464
```

Determining the Elbow point using K means clustering

```
In [4]: sse = {}

        for k in range(1, 20):
            kmeans = KMeans(n_clusters=k, max_iter=1000).fit(wine_data_scaled)
            wine_data["clusters"] = kmeans.labels_
            sse[k] = kmeans.inertia_

        plt.figure()
        plt.plot(list(sse.keys()), list(sse.values()))
        plt.scatter(list(sse.keys()), list(sse.values()))
        plt.title('elbow plot')
        plt.xlabel("Number of clusters")
        plt.ylabel("SSE")
        plt.show()
```



From the above we can observe that K=5/6 is the perfect choice of K Thus the plot shows that the number of groups to choose is 5. Hence lets run K means algorithm for k=5 and find clusters in the data

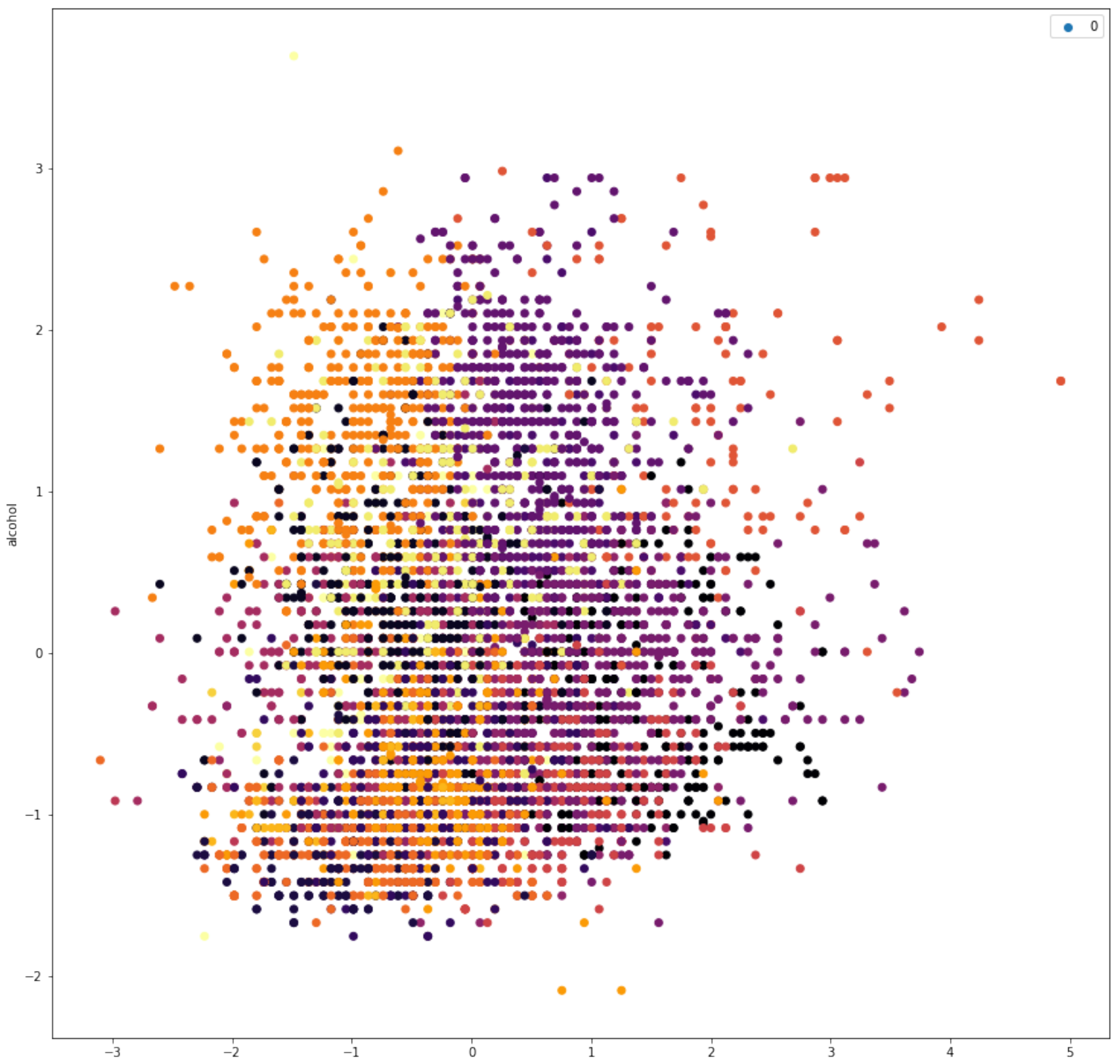
2.3 Making Pair wise profiling plots and labelling wines with respect to its ingredients

plotting alcohol vs pH clusters

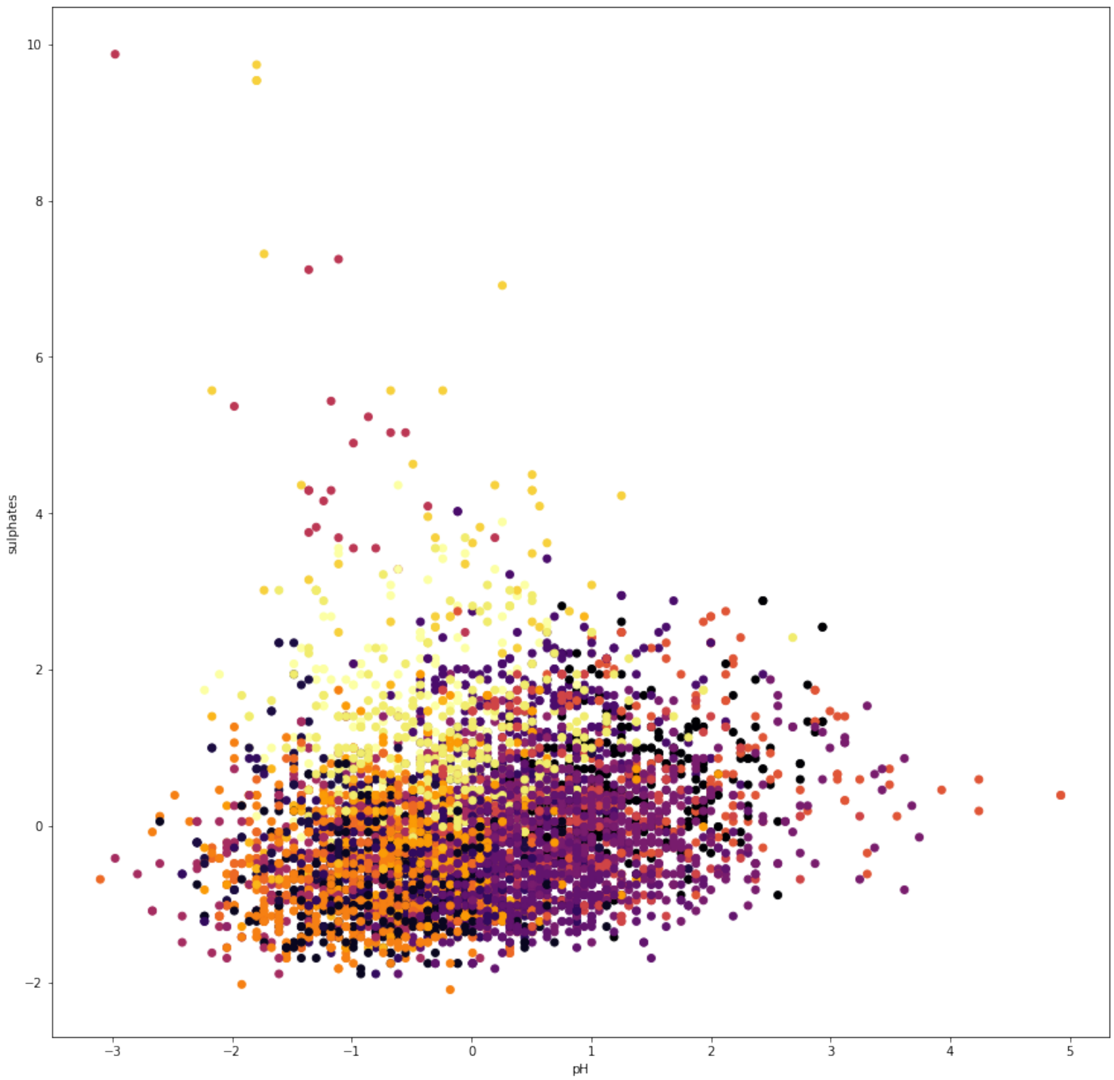
In [5]:
```
wine_data.clusters.unique()
```

Out[5]:
```
array([ 0, 18,  4, 16,  9,  7,  3, 11,  6, 15, 17,  8,  1,  5, 13,  2, 10,
       12, 14])
```

In [6]:
```
plt.figure(figsize=(15,15))
plt.scatter(wine_data_scaled.pH,
            wine_data_scaled.alcohol,
            c=wine_data.clusters,
            cmap='inferno')
plt.legend(wine_data.clusters.unique())
plt.xlabel('pH')
plt.ylabel('alcohol')
plt.show()
```

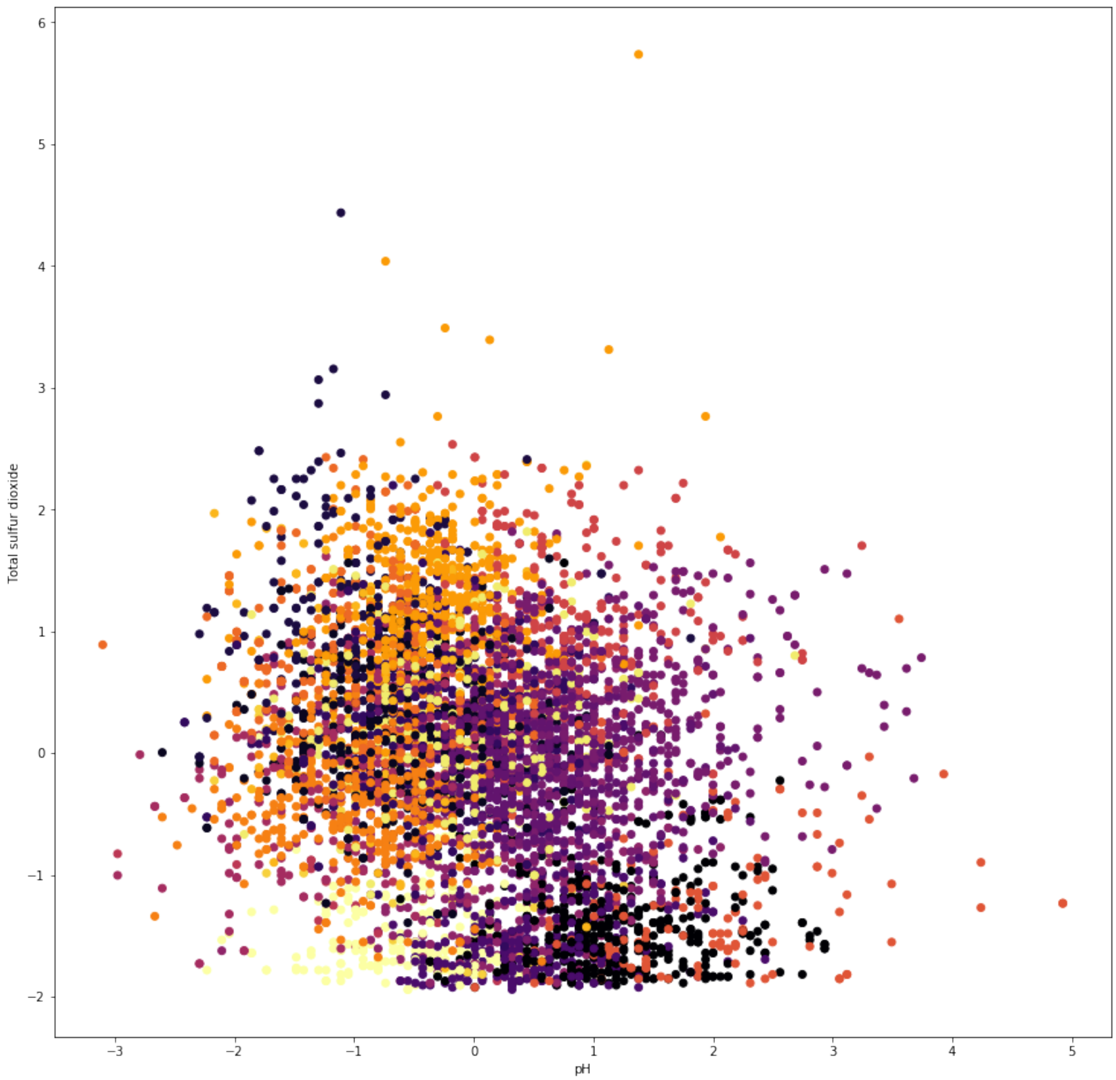


In [7]:
```
plt.figure(figsize=(15,15))
plt.scatter(wine_data_scaled.pH,
            wine_data_scaled.sulphates,
            c=wine_data.clusters,
            cmap='inferno')
plt.xlabel('pH')
plt.ylabel('sulphates')
plt.show()
```

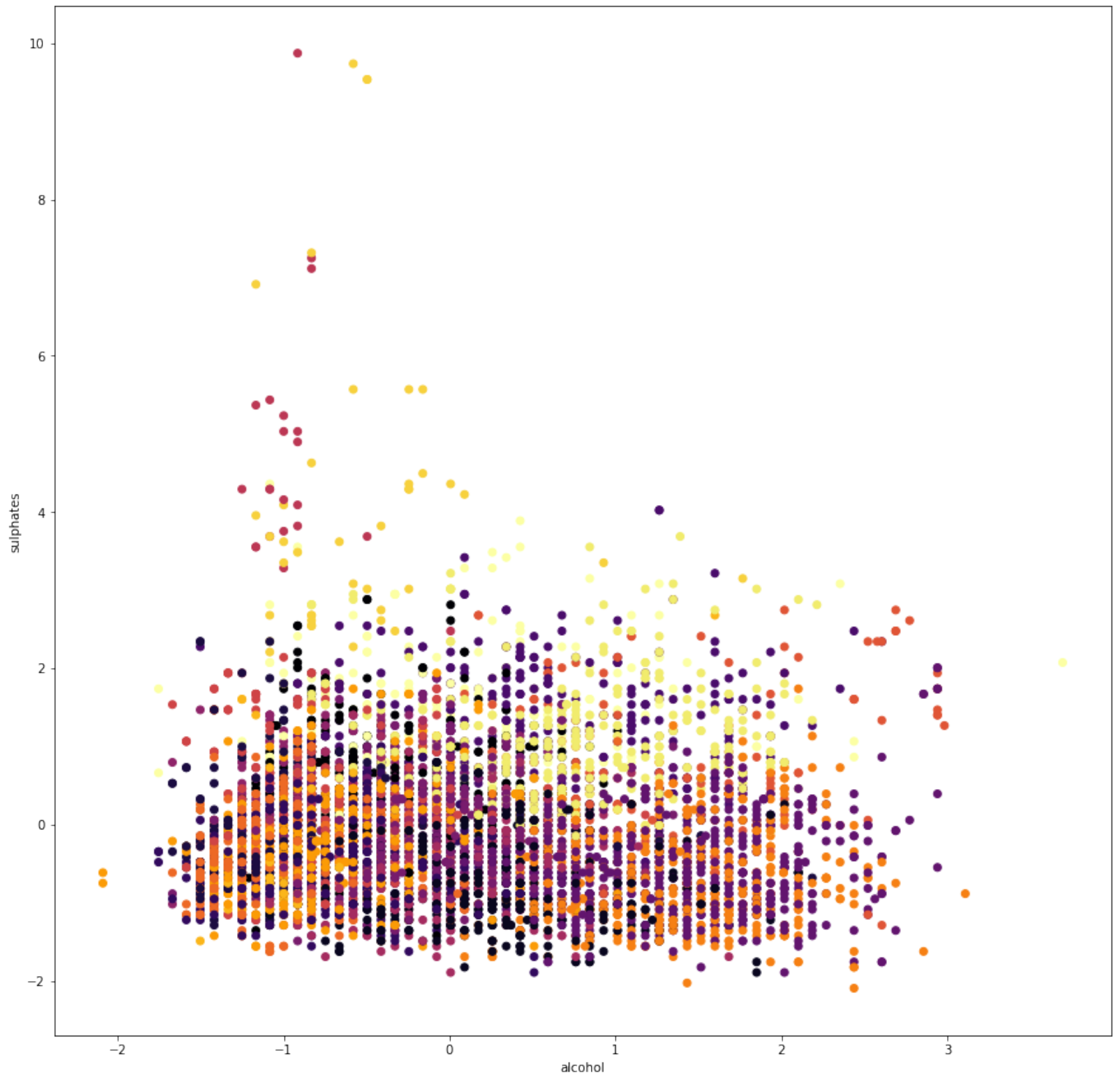


In [8]:
```
plt.figure(figsize=(15,15))
plt.scatter(wine_data_scaled.pH,
```

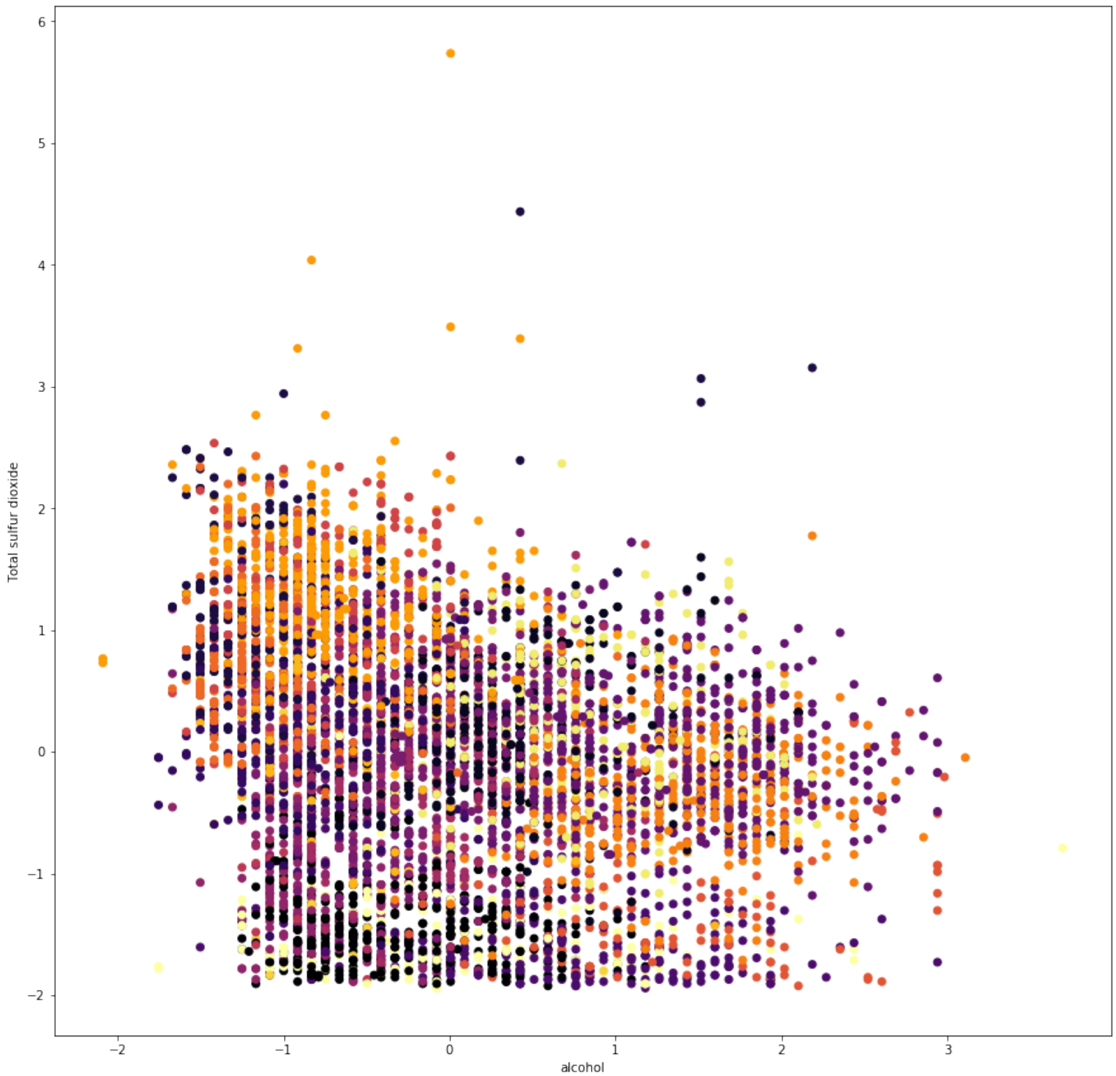
```
                wine_data_scaled['total sulfur dioxide'],
                c=wine_data.clusters,
                cmap='inferno')
plt.xlabel('pH')
plt.ylabel('Total sulfur dioxide')
plt.show()
```



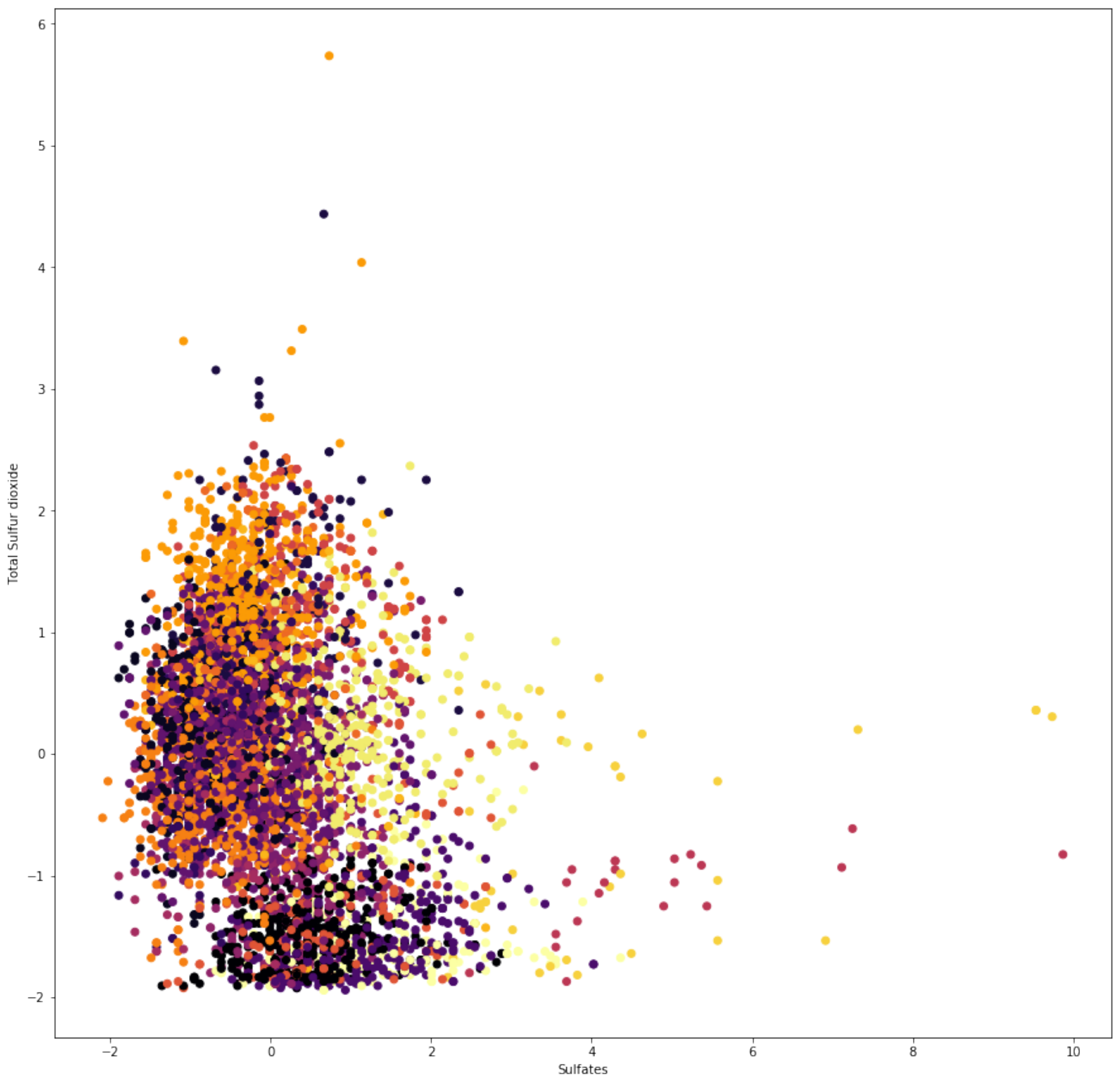
In [10]:
```
plt.figure(figsize=(15,15))
plt.scatter(wine_data_scaled.alcohol,
            wine_data_scaled.sulphates,
            c=wine_data.clusters,
            cmap='inferno')
plt.xlabel('alcohol')
plt.ylabel('sulphates')
plt.show()
```




```
In [11]: plt.figure(figsize=(15,15))
         plt.scatter(wine_data_scaled.alcohol,
                     wine_data_scaled['total sulfur dioxide'],
                     c=wine_data.clusters,
                     cmap='inferno')
         plt.xlabel('alcohol')
         plt.ylabel('Total sulfur dioxide')
         plt.show()
```



In [12]:
```
plt.figure(figsize=(15,15))
plt.scatter(wine_data_scaled.sulphates,
            wine_data_scaled['total sulfur dioxide'],
            c=wine_data.clusters,
            cmap='inferno')
plt.xlabel('Sulfates')
plt.ylabel('Total Sulfur dioxide')
plt.show()
```



3 KNN Classifier

In [59]:
```
red_wine_data = pd.read_csv('../Dataset/winequality-red.csv', sep=';')
white_wine_data = pd.read_csv('../Dataset/winequality-white.csv', sep=';')

wine_data = pd.concat([red_wine_data, white_wine_data])
bins = (2, 5, 6, 10)
group_names = ['bad', 'medium', 'good']
wine_data['quality'] = pd.cut(wine_data['quality'], bins = bins, labels = group_names
wine_data.iloc[:, :11].describe()
```

Out[59]:
```
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    6497.000000       6497.000000  6497.000000     6497.000000
```

```
mean        7.215307          0.339666     0.318633        5.443235
std         1.296434          0.164636     0.145318        4.757804
min         3.800000          0.080000     0.000000        0.600000
25%         6.400000          0.230000     0.250000        1.800000
50%         7.000000          0.290000     0.310000        3.000000
75%         7.700000          0.400000     0.390000        8.100000
max        15.900000          1.580000     1.660000       65.800000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  6497.000000          6497.000000           6497.000000  6497.000000
mean      0.056034            30.525319            115.744574     0.994697
std       0.035034            17.749400             56.521855     0.002999
min       0.009000             1.000000              6.000000     0.987110
25%       0.038000            17.000000             77.000000     0.992340
50%       0.047000            29.000000            118.000000     0.994890
75%       0.065000            41.000000            156.000000     0.996990
max       0.611000           289.000000            440.000000     1.038980

                pH    sulphates      alcohol
count  6497.000000  6497.000000  6497.000000
mean      3.218501     0.531268    10.491801
std       0.160787     0.148806     1.192712
min       2.720000     0.220000     8.000000
25%       3.110000     0.430000     9.500000
50%       3.210000     0.510000    10.300000
75%       3.320000     0.600000    11.300000
max       4.010000     2.000000    14.900000
```

In [60]:
```python
scaler = StandardScaler()

data = wine_data.iloc[:,:11].values
scaled_features = scaler.fit_transform(data)

wine_data_scaled = pd.DataFrame(scaled_features, index=wine_data.index, columns=wine_d
wine_data_scaled.head()
```

Out[60]:
```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0       0.142473          2.188833    -2.192833       -0.744778   0.569958
1       0.451036          3.282235    -2.192833       -0.597640   1.197975
2       0.451036          2.553300    -1.917553       -0.660699   1.026697
3       3.073817         -0.362438     1.661085       -0.744778   0.541412
4       0.142473          2.188833    -2.192833       -0.744778   0.569958

   free sulfur dioxide  total sulfur dioxide   density        pH  sulphates  \
0            -1.100140             -1.446359  1.034993  1.813090   0.193097
1            -0.311320             -0.862469  0.701486 -0.115073   0.999579
2            -0.874763             -1.092486  0.768188  0.258120   0.797958
3            -0.762074             -0.986324  1.101694 -0.363868   0.327510
```

```
4           -1.100140           -1.446359  1.034993  1.813090   0.193097

    alcohol
0 -0.915464
1 -0.580068
2 -0.580068
3 -0.580068
4 -0.915464
```

In [83]:
```
X_train, X_test, Y_train, Y_test = train_test_split(wine_data.iloc[:, :11],
                                                    wine_data.quality,
                                                    test_size=0.2,
                                                    random_state=42)
```

In [84]:
```
knn = KNeighborsClassifier(20)
knn.fit(X_train, Y_train)
```

Out[84]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=20, p=2,
           weights='uniform')
```

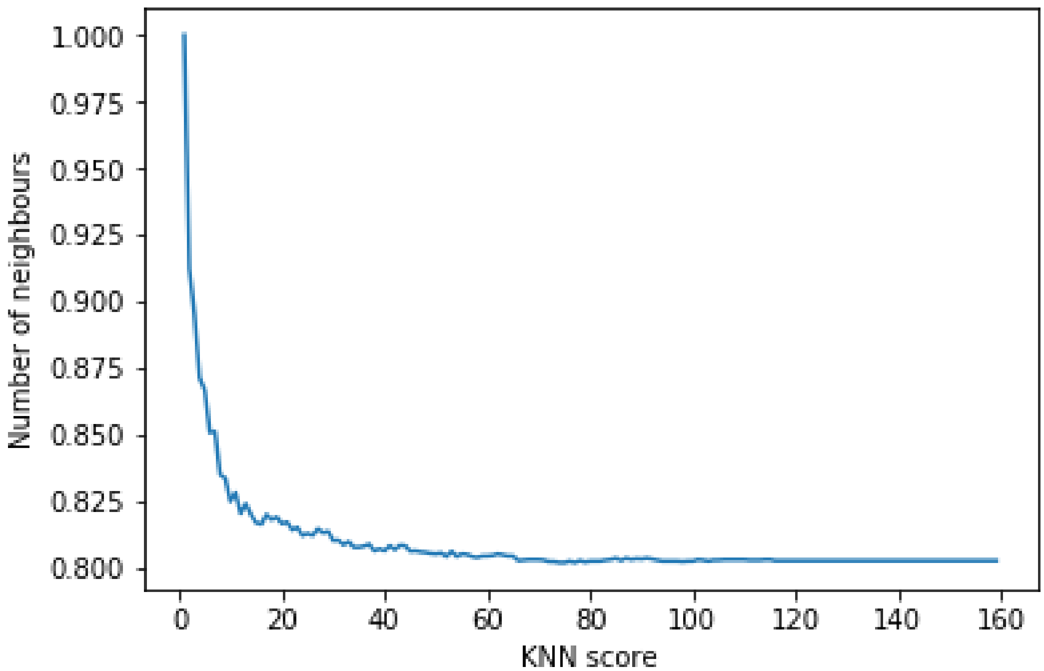
In [85]:
```
knn.score(X_train, Y_train)
```

Out[85]:
```
0.8166249759476621
```

In [89]:
```
knn_score = []

for i in range(1,160):
    knn = KNeighborsClassifier(i)
    knn.fit(X_train, Y_train)
    knn_score.append(knn.score(X_train, Y_train))

plt.plot(range(1, 160), knn_score)
plt.xlabel("KNN score")
plt.ylabel("Number of neighbours")
plt.show()
```


1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
Number of neighbours
0
20
40
60
80
100
120
140
160
KNN score