Wine_Quality_Numpy_Implementation

March 15, 2023

[79]:

```
#importing the required libraries/dependencies
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

0.1 Implementing the NumPy Methods

[80]:

```
#loading the downloaded csv file from Kaggle
#wine_dataset = pd.read_csv('/content/winequality-red.csv')
#load the Wine dataset into a pandas dataframe
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data"
columns = ["class", "alcohol", "malic_acid", "ash", "alcalinity_of_ash",␣
 ↪"magnesium", "total_phenols", "flavanoids", "nonflavanoid_phenols",␣
 ↪"proanthocyanins", "color_intensity", "hue", "od280/od315_of_diluted_wines",␣
 ↪"proline"]
wine_dataset = pd.read_csv(url, names=columns)
```

[81]:

```
#converting the wine_dataset into NumPy array
data = wine_dataset.values
```

[82]:

```
#knowing the shape of the wine dataset
shape = data.shape
print("Shape: ",shape)
```

```
Shape:  (178, 14)
```

[83]:

```
#calculating the median of all the 12 columms of the data
medians = np.median(data, axis=0)
print("Medians: ",medians)
```

```
Medians:  [2.000e+00 1.305e+01 1.865e+00 2.360e+00 1.950e+01 9.800e+01 2.355e+00
 2.135e+00 3.400e-01 1.555e+00 4.690e+00 9.650e-01 2.780e+00 6.735e+02]
```

[84]:

```
#calculating the mean of each colums
mean = np.mean(data, axis=0)
print("Mean is :",mean)
```

```
Mean is : [1.93820225e+00 1.30006180e+01 2.33634831e+00 2.36651685e+00
 1.94949438e+01 9.97415730e+01 2.29511236e+00 2.02926966e+00
 3.61853933e-01 1.59089888e+00 5.05808988e+00 9.57449438e-01
 2.61168539e+00 7.46893258e+02]
```

[85]:

```
#calculating the Standard Deviation of each column
standard_dvn = np.std(data ,axis=0)
print("Standard Deviation is :",standard_dvn)
```

```
Standard Deviation is : [7.72854859e-01 8.09542915e-01 1.11400363e+00
2.73572294e-01
 3.33016976e+00 1.42423077e+01 6.24090564e-01 9.96048950e-01
 1.24103260e-01 5.70748849e-01 2.31176466e+00 2.27928607e-01
 7.07993265e-01 3.14021657e+02]
```

[86]:

```
#calculating the minimum of each columns
minimum = np.min(data, axis=0)
print("The minimum value of each column are: ", minimum)
```

```
The minimum value of each column are:  [1.000e+00 1.103e+01 7.400e-01 1.360e+00
1.060e+01 7.000e+01 9.800e-01
 3.400e-01 1.300e-01 4.100e-01 1.280e+00 4.800e-01 1.270e+00 2.780e+02]
```

[87]:

```
#calculating the maximum of each columns
maximum = np.max(data,axis=0)
print("The maximum value of each column is :", maximum)
```

```
The maximum value of each column is : [3.000e+00 1.483e+01 5.800e+00 3.230e+00
3.000e+01 1.620e+02 3.880e+00
 5.080e+00 6.600e-01 3.580e+00 1.300e+01 1.710e+00 4.000e+00 1.680e+03]
```

[88]:

```
#calculate the range of each column
ranges = np.ptp(data, axis=0)
print("Ranges of all the columns are: ", ranges)
```

```
Ranges of all the columns are:  [2.000e+00 3.800e+00 5.060e+00 1.870e+00
1.940e+01 9.200e+01 2.900e+00
 4.740e+00 5.300e-01 3.170e+00 1.172e+01 1.230e+00 2.730e+00 1.402e+03]
```

[89]:

```
#Reshape the data into a 3D array
reshaped = data.reshape(178, 14, 1)
print("Reshaped Data: \n", reshaped)
```

```
Reshaped Data:
 [[[1.000e+00]
  [1.423e+01]
  [1.710e+00]
  …
  [1.040e+00]
  [3.920e+00]
  [1.065e+03]]

 [[1.000e+00]
  [1.320e+01]
  [1.780e+00]
  …
  [1.050e+00]
  [3.400e+00]
  [1.050e+03]]

 [[1.000e+00]
  [1.316e+01]
  [2.360e+00]
  …
  [1.030e+00]
  [3.170e+00]
  [1.185e+03]]

 …

 [[3.000e+00]
  [1.327e+01]
  [4.280e+00]
  …
  [5.900e-01]
  [1.560e+00]
  [8.350e+02]]

 [[3.000e+00]
  [1.317e+01]
  [2.590e+00]
  …
  [6.000e-01]
  [1.620e+00]
  [8.400e+02]]

 [[3.000e+00]
  [1.413e+01]
  [4.100e+00]
  …
  [6.100e-01]
  [1.600e+00]
  [5.600e+02]]]
```

[90]:

```
#calculate the sum of each row
row_sums = np.sum(data, axis=1)
print("Row Sums: ", row_sums)
```

```
Row Sums:  [1246.       1195.1      1342.82      1651.49       909.69      1616.23
 1437.46     1469.78     1192.82     1197.46     1671.6      1426.17
 1461.       1291.68     1702.52     1477.32     1458.1      1302.79
 1845.92     1013.48      959.93      927.08     1187.89     1160.99
  995.46     1013.93     1339.4      1428.11     1078.17     1182.46
 1445.95     1677.79     1145.51     1422.68     1257.84     1076.61
 1039.71     1253.74     1165.43      940.76      966.36     1179.24
 1248.91      836.44     1044.5      1248.08     1222.51     1139.88
 1219.71     1426.28     1292.52     1413.44     1355.12     1544.67
 1230.36     1293.9      1142.64     1425.85     1448.21      643.34
  825.08      597.72      773.27      560.86      507.9       827.85
  631.12      639.76      906.56      915.92     1021.236     556.83
  614.91     1189.78     1039.98      570.3       525.59      659.51
  931.7       622.62      413.33      852.49      761.33      656.85
  663.21      597.8       637.38      706.38      815.78      747.92
  608.74      588.06      625.29      428.88      493.06     1150.95
  809.17      560.66      801.81      547.32      855.95      697.05
  589.55      549.29      809.19      458.7       640.35      625.1
  453.9       826.97      723.56      465.1       762.32      572.49
  522.81      543.88      630.72      505.84      498.54      700.57
  774.55      652.56      527.67      524.49      519.77      516.27
  493.27      618.32      485.56      714.3       799.59      686.42
  713.58      760.09      783.64      841.02      861.56      671.31
  721.66      748.63      749.44      921.95      673.33      697.57
 1021.28      984.94      548.36      769.19      800.42      722.42
  684.17      652.03      601.6       831.01      795.69      878.19
  627.73     1039.27      827.59      772.9       665.12      841.84
  732.07      832.4       764.27      666.42      869.35      830.23
  916.87      806.27      656.41      611.059999  809.24      894.26
  911.74     1014.22     1016.97      720.6      ]
```

[91]:

```
#calculating the dot product of two columns
dot_product = np.dot(data[:, 0], data[:, 1])
print("Dot Product of Columns 1 and 2: ", dot_product)
```

```
Dot Product of Columns 1 and 2:  4448.66
```

[92]:

```
#calculating the index of the minimum values of the array
min_index = np.argmin(data, axis=0)
print("Index :", min_index)
```

```
Index : [  0 115 113  59  59  89 146 146  74  60 119 151 136  80]
```

[93]:

```
#calculating the index of the maximum values of the array
max_index = np.argmax(data, axis=0)
print("Index :", max_index)
```

```
Index : [130   8 123 121  73  95  52 121 105 110 158 115  22  18]
```

[94]:

```
#trasnposing the arrays of the dataset
transpose = np.transpose(data)
print("Transposed Array :", transpose)

#the shape changes too as we have taken the transpose
print("The new shape is:",transpose.shape)
```

```
Transposed Array : [[1.000e+00 1.000e+00 1.000e+00 … 3.000e+00 3.000e+00
3.000e+00]
 [1.423e+01 1.320e+01 1.316e+01 … 1.327e+01 1.317e+01 1.413e+01]
 [1.710e+00 1.780e+00 2.360e+00 … 4.280e+00 2.590e+00 4.100e+00]
 …
 [1.040e+00 1.050e+00 1.030e+00 … 5.900e-01 6.000e-01 6.100e-01]
 [3.920e+00 3.400e+00 3.170e+00 … 1.560e+00 1.620e+00 1.600e+00]
 [1.065e+03 1.050e+03 1.185e+03 … 8.350e+02 8.400e+02 5.600e+02]]
The new shape is: (14, 178)
```

0.2 Implementing few Matplotlib Methods

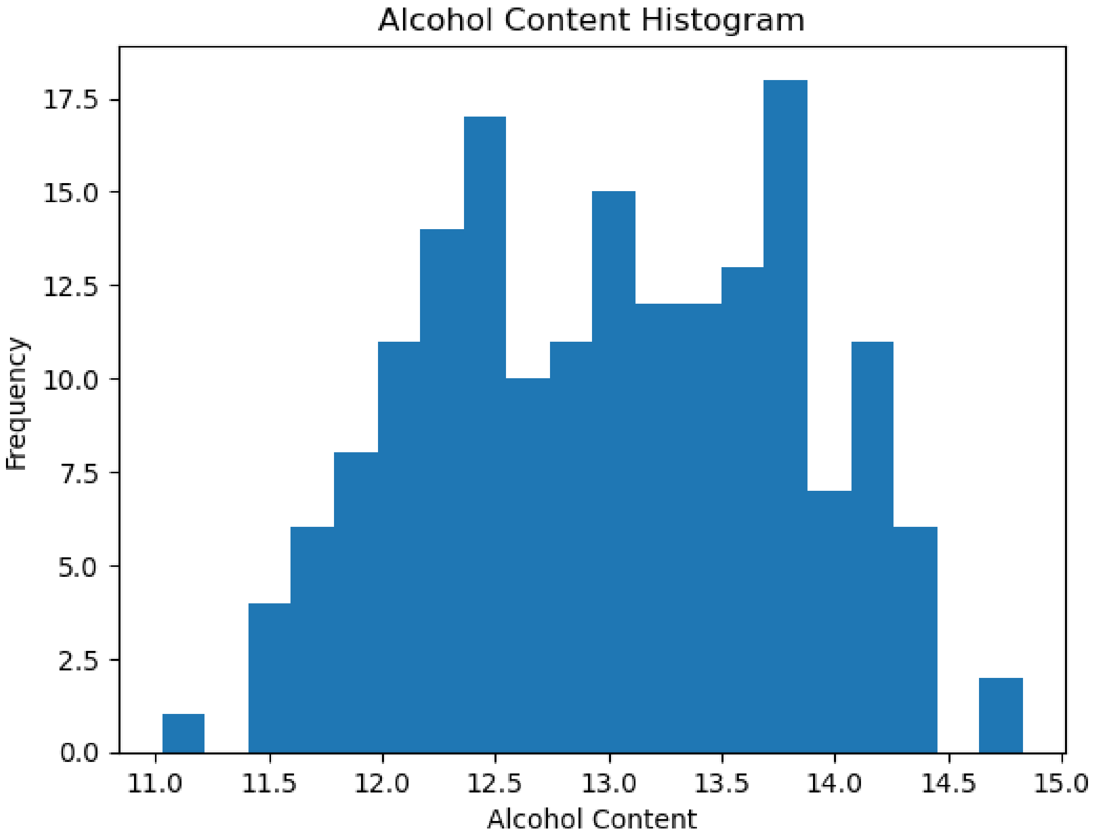
[95]:

```
#load the Wine dataset into a pandas dataframe
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data"
columns = ["class", "alcohol", "malic_acid", "ash", "alcalinity_of_ash",␣
 ↪"magnesium", "total_phenols", "flavanoids", "nonflavanoid_phenols",␣
 ↪"proanthocyanins", "color_intensity", "hue", "od280/od315_of_diluted_wines",␣
 ↪"proline"]
wine_df = pd.read_csv(url, names=columns)
```

Histogram

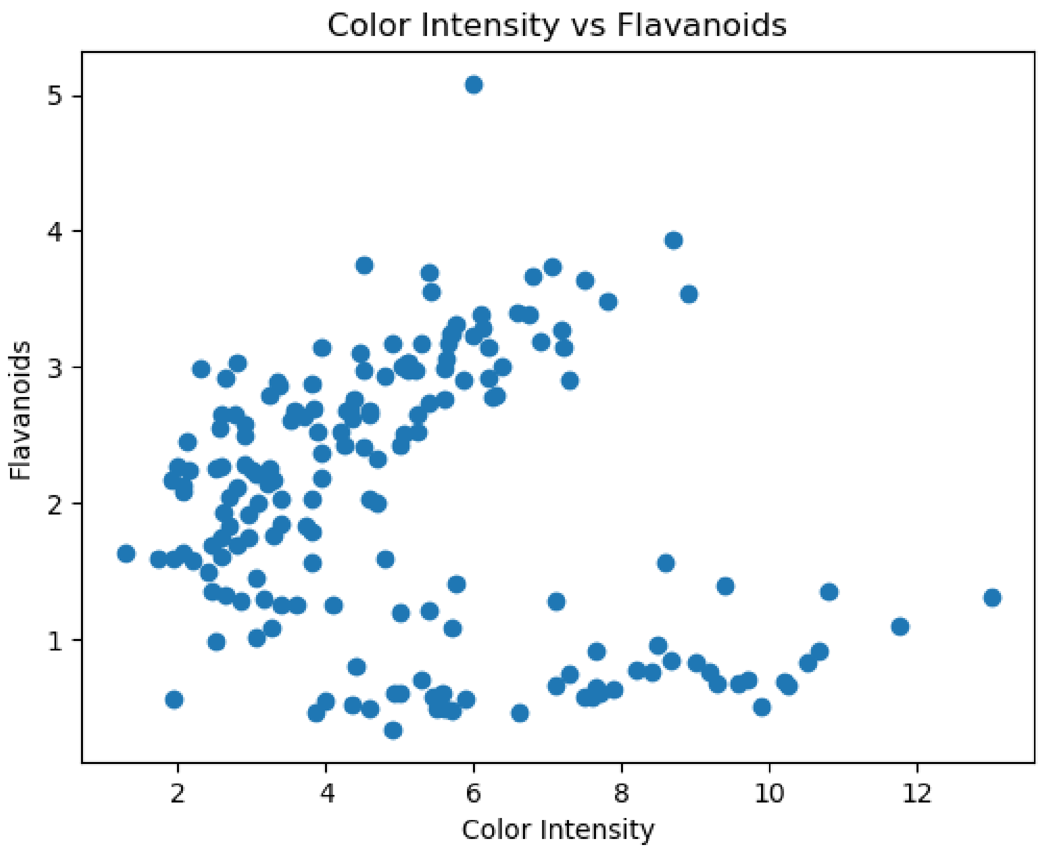
[96]:

```
#creating a histogram of alcohol content
plt.hist(wine_df["alcohol"], bins=20)
plt.title("Alcohol Content Histogram")
plt.xlabel("Alcohol Content")
plt.ylabel("Frequency")
plt.show()
```



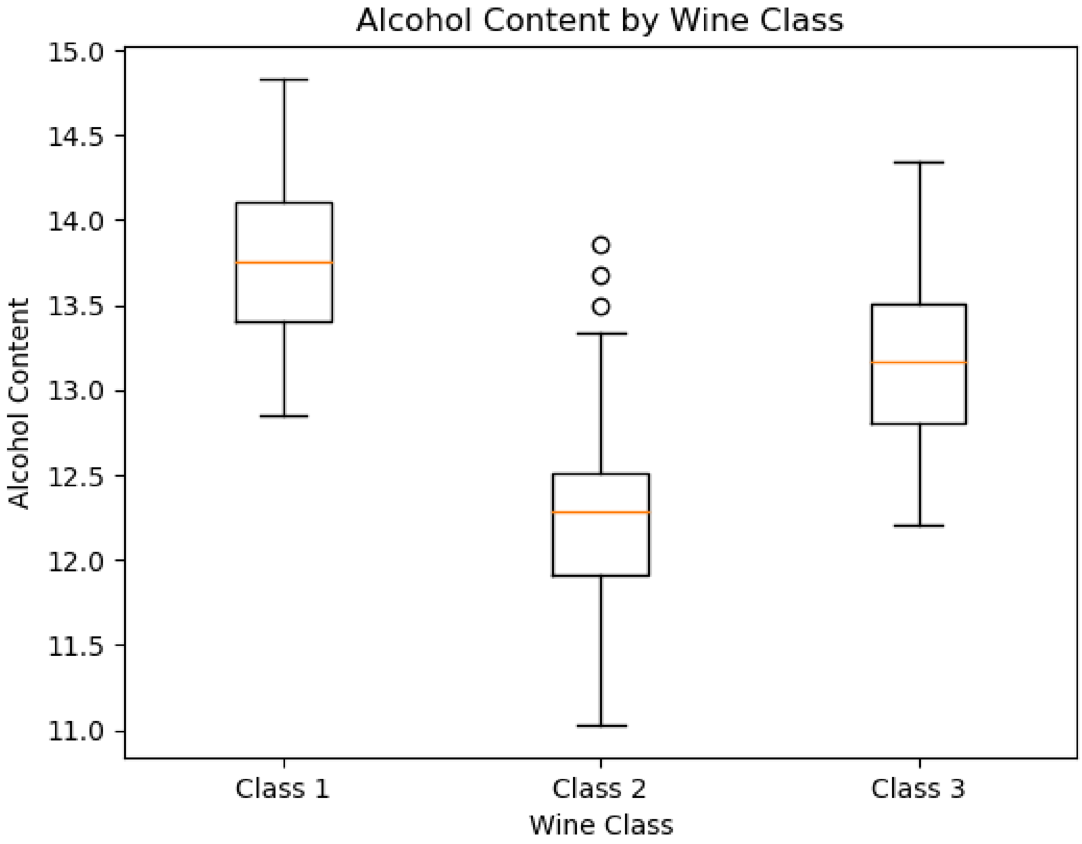
ScatterPlot

```
[97]: #creating a scatter plot of color intensity vs flavanoids
plt.scatter(wine_df["color_intensity"], wine_df["flavanoids"])
plt.title("Color Intensity vs Flavanoids")
plt.xlabel("Color Intensity")
plt.ylabel("Flavanoids")
plt.show()
```



BoxPlot

[98]:
```
#creating a boxplot of alcohol content by wine class
plt.boxplot([wine_df[wine_df["class"]==1]["alcohol"],␣
  ↪wine_df[wine_df["class"]==2]["alcohol"],␣
  ↪wine_df[wine_df["class"]==3]["alcohol"]])
plt.title("Alcohol Content by Wine Class")
plt.xlabel("Wine Class")
plt.ylabel("Alcohol Content")
plt.xticks([1, 2, 3], ["Class 1", "Class 2", "Class 3"])
plt.show()
```

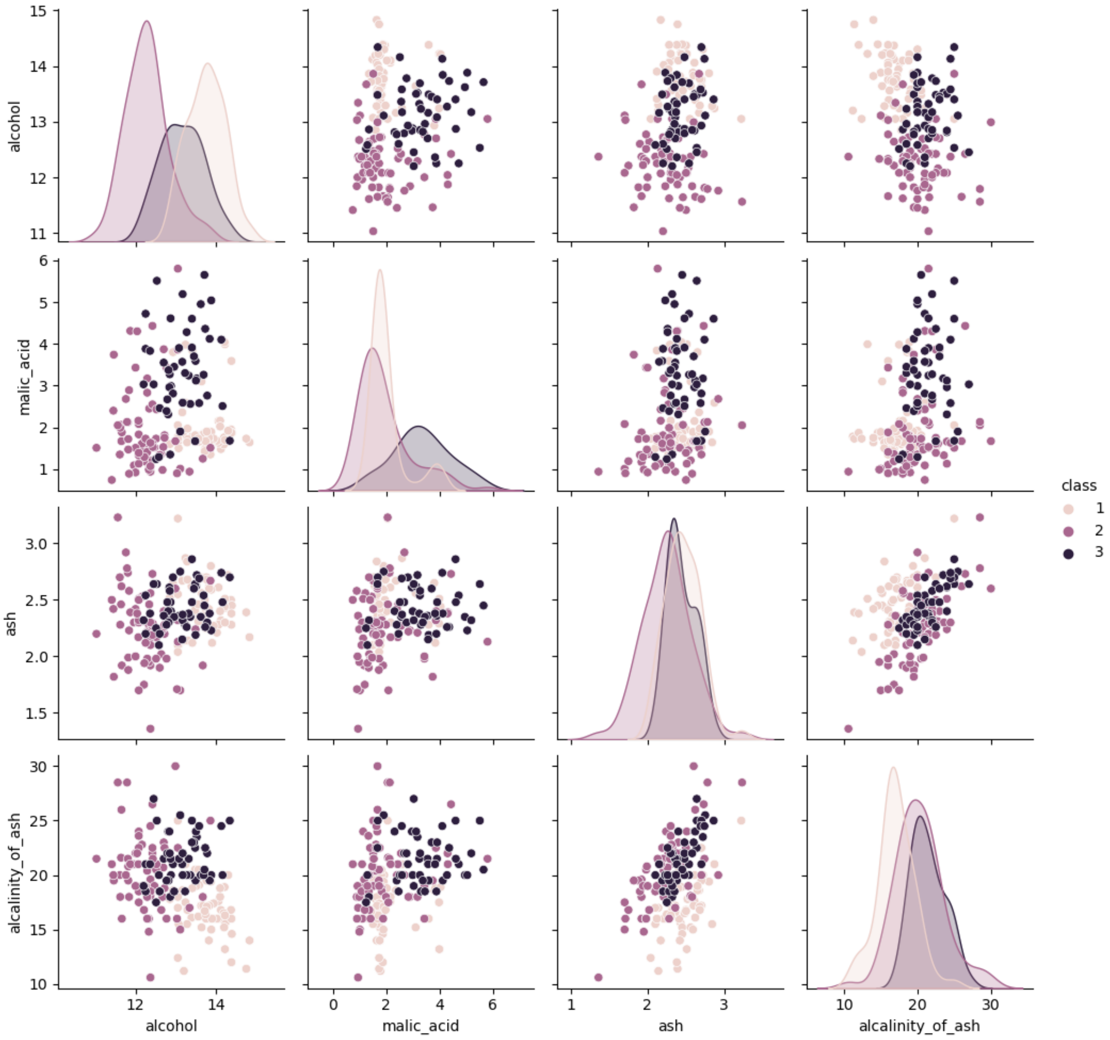


0.3 Implementing methods of the Seaborn Library

Pairplot

[99]:

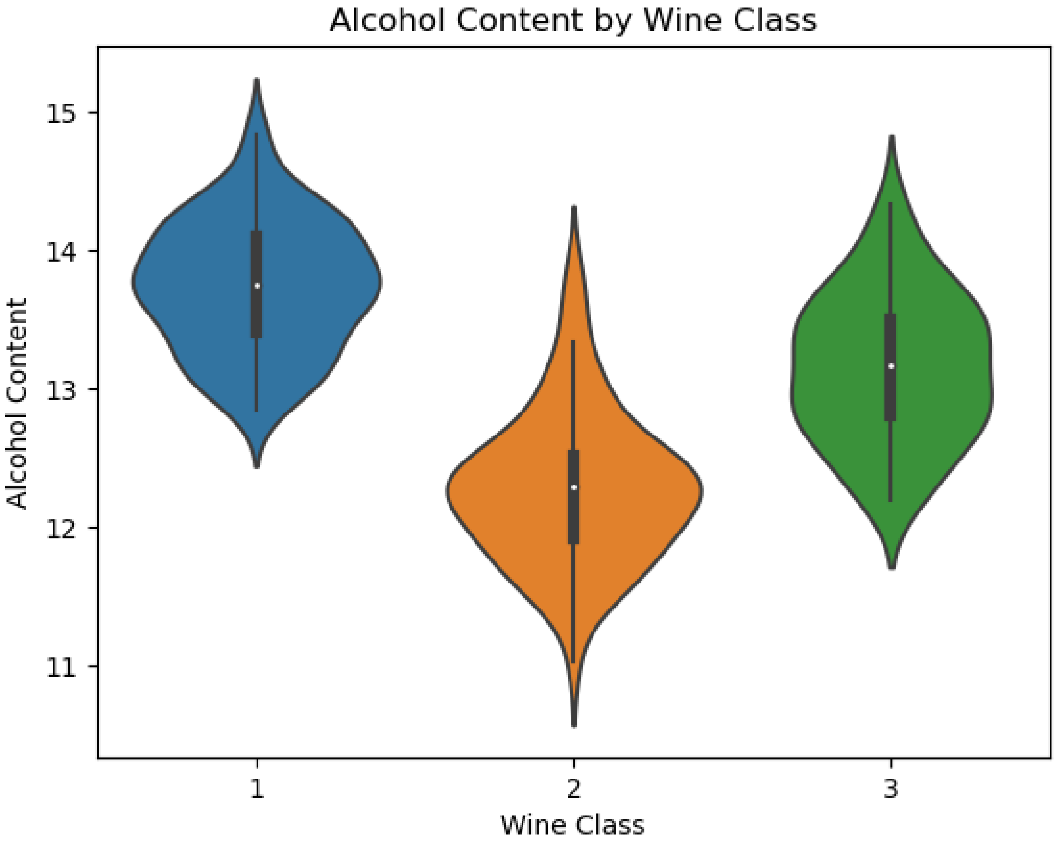
```
#creating a pairplot of the first four variables
sns.pairplot(wine_df.iloc[:, 0:5], hue="class")
plt.show()
```



Violin Plot

[100]:

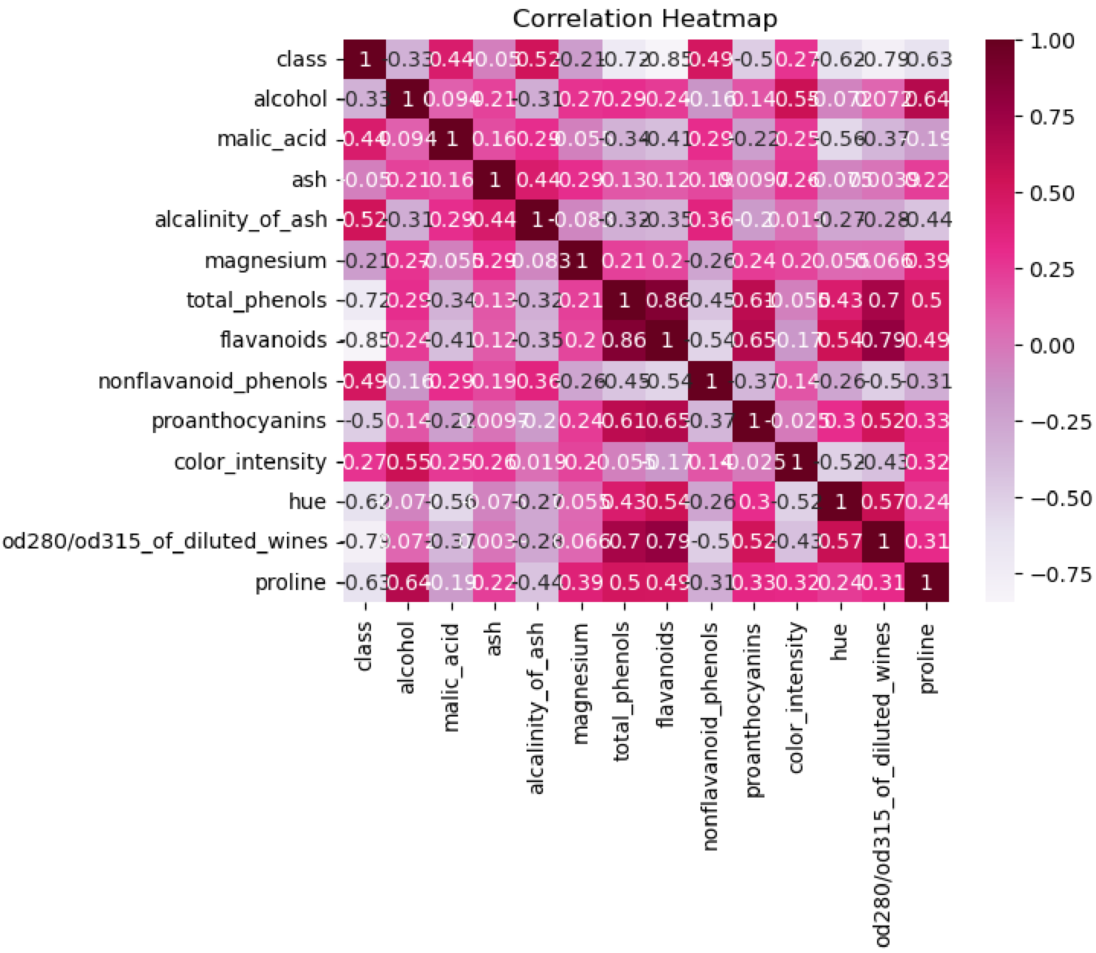
```
#creating a violin plot for the 3 different wine classes
sns.violinplot(x="class", y="alcohol", data=wine_df)
plt.title("Alcohol Content by Wine Class")
plt.xlabel("Wine Class")
plt.ylabel("Alcohol Content")
plt.show()
```



Heatmap

[101]:
```
#calculating the correlation matrix
corr_matrix = wine_df.corr()

# Creating a heatmap of the correlation matrix using Seaborn and using␣
 ↪Purple-Red(PuRd) color combination for the Heatmap
sns.heatmap(corr_matrix, annot=True, cmap="PuRd")
plt.title("Correlation Heatmap")
plt.show()
```



0.4 References/Citations:

- https://www.python.org/ (wherever stuck in python)
- https://numpy.org/doc/stable/user/index.html#user (implementing functions on NumPy arrays)
- https://www.w3schools.com/python/pandas/pandas_plotting.asp (to gain knowledge about plots)
- https://seaborn.pydata.org/ (to get to know about the various plots)
- https://scikit-learn.org/stable/datasets/toy_dataset.html (to access the Wine dataset)
- https://archive.ics.uci.edu/ml/datasets.php (Wine Dataset)