Exercise work – Part 1

Data Analysis and Knowledge Discovery

Name: Joonas Syysvirta

Student number: 502603

Wine set: Red wines

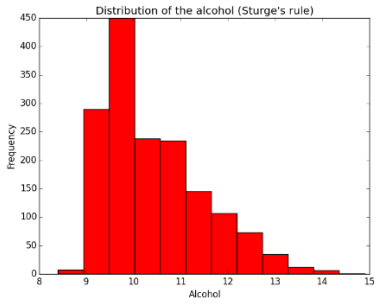
Preparing the data

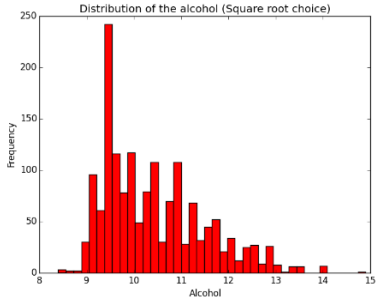
I started working on the data set by reading the *winequality-red.csv* file with Python's *csv.reader* method. I then stored the data by first creating separate arrays for each of the attributes/features (12 total), and then storing each attribute's values into these arrays. I also created an array for the whole data set, by reading each row of the set and parsing each sample into the array.

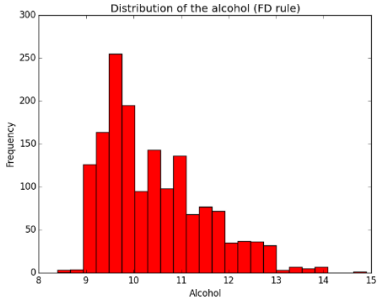
1. Histograms

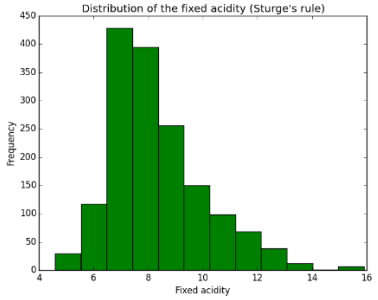
Now that I had every value of each attribute's data in a (named) array data structure, I created a method that takes one of these attribute data lists as a parameter and plots a histogram for that attribute's data, by using *matplotlib*'s `hist()` method.

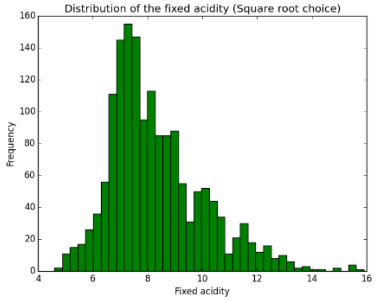
To determine the number of bins to use, I've used the following three methods: ***Sturge's rule***, ***square-root choice*** and ***Freedman-Diaconis' rule***. I have created separate methods for each of these (using the functions in the slides). The *Freedman-Diaconis' rule* needs the interquartile range of the attribute data to count the number of bins, so I created a separate function that counts the IQR for a given attribute. Below I've included the three histograms for six different attributes.

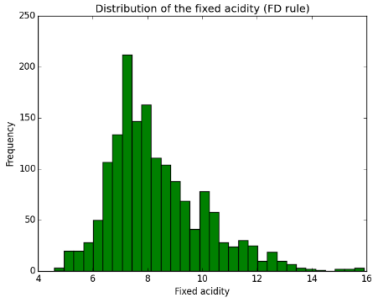


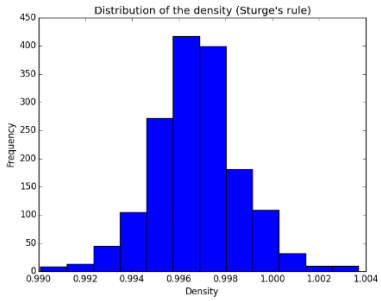


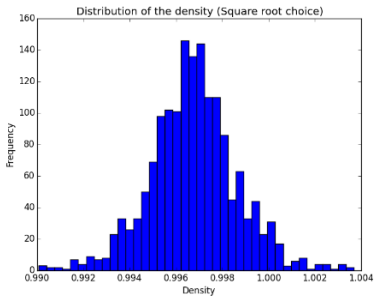


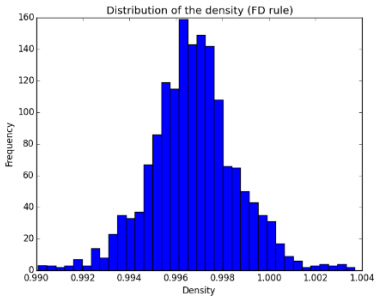


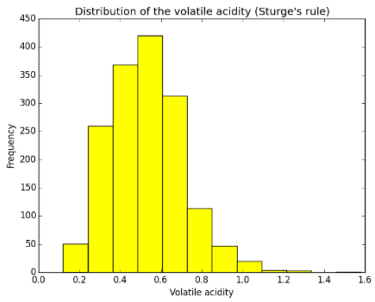


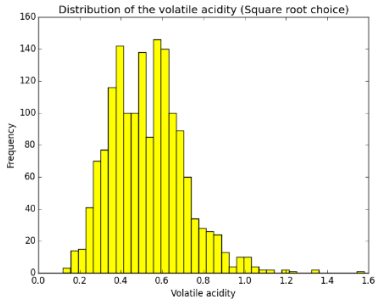


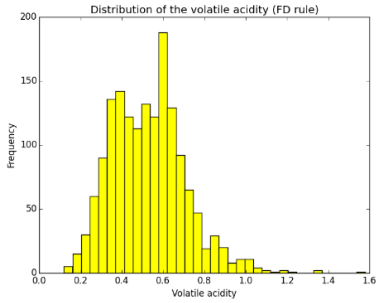


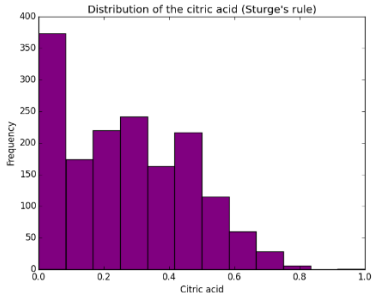


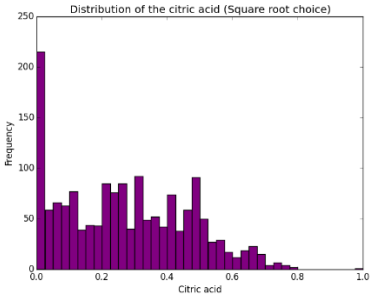


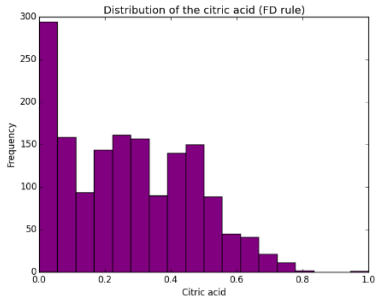


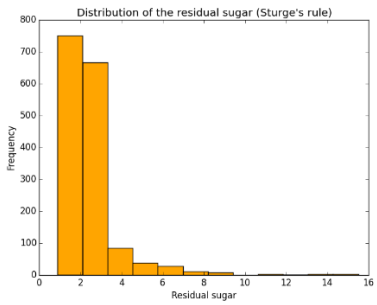


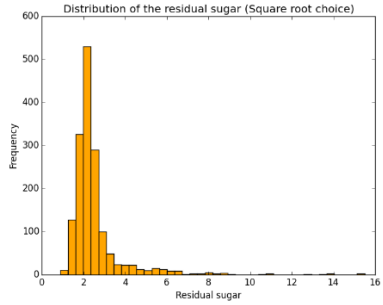


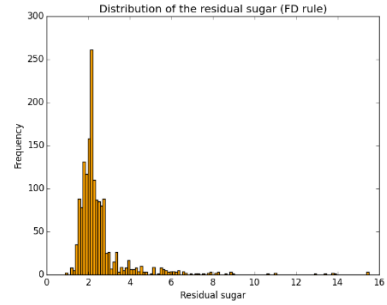






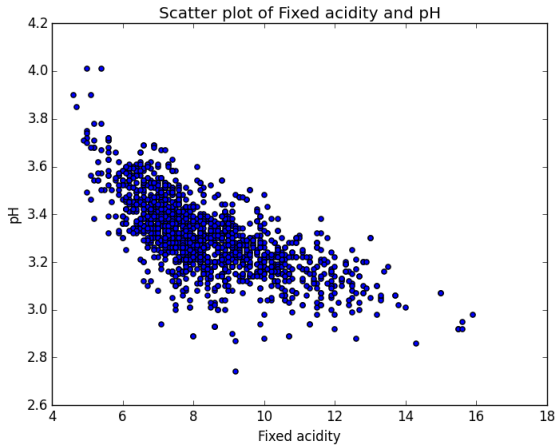


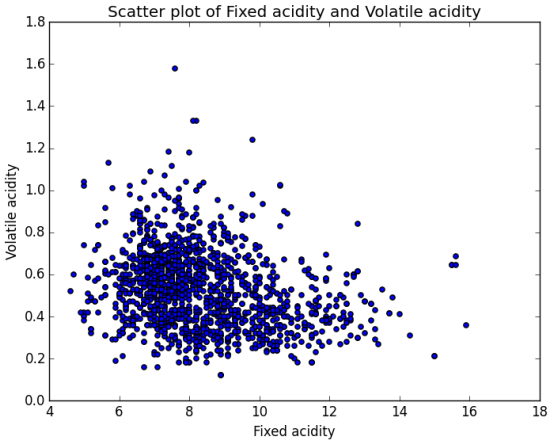


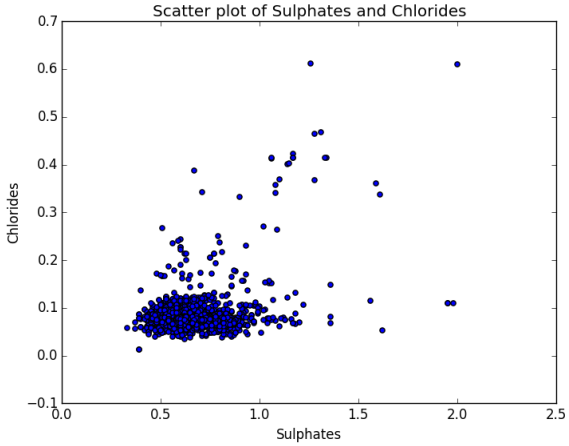


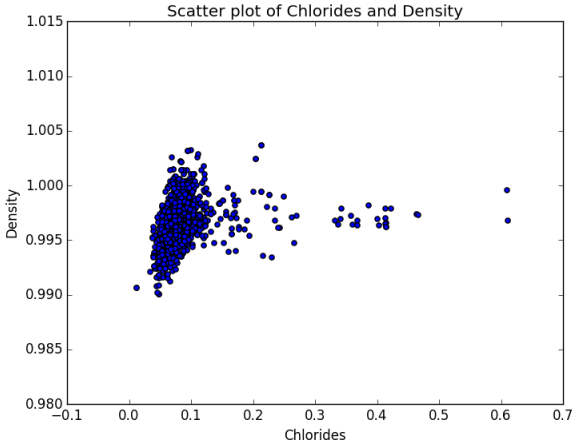
Using *Sturge's rule* the number of bins (12) is relatively low, so the histograms don't portray the distributions very accurately. The histograms seem more informative and we can get a better sense of the distribution when the number of bins is a bit higher like it is with the *square root choice* (40) and the *Freedman-Diaconis' rule*. Having too many bins, however, creates a kind of a "comb effect", which means that the values are divided into bins that are too small. What we can see from most of the histograms is that the distribution is weighted towards the left side of the plot.

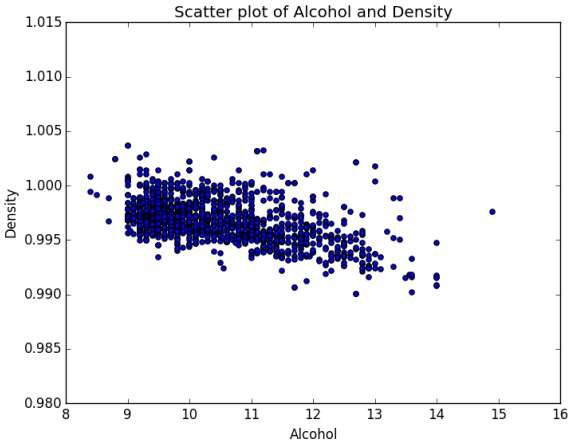
2. Scatter plots and parallel coordinates

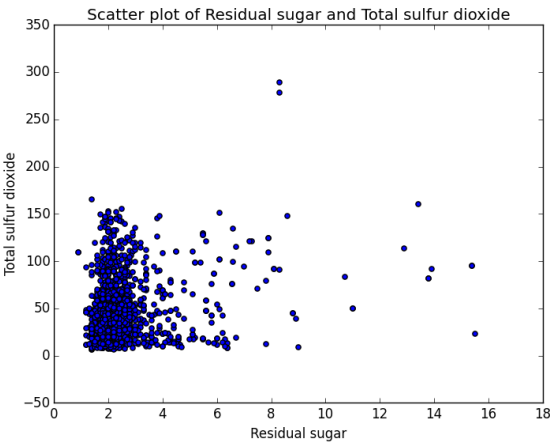








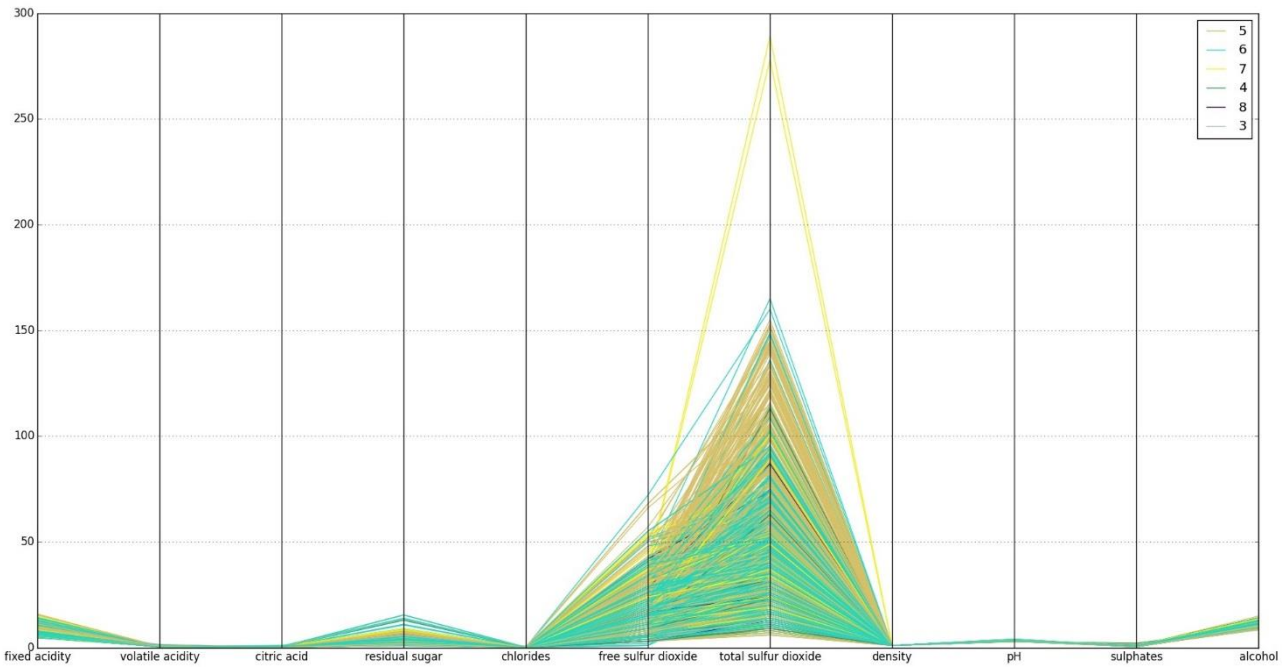




Above: some scatter plots for the attributes

Here I've included some of the scatter plots for the attribute pairs. For the scatter plots I created a method that takes two of the aforementioned attribute data lists as parameters and plots a scatter plot for those

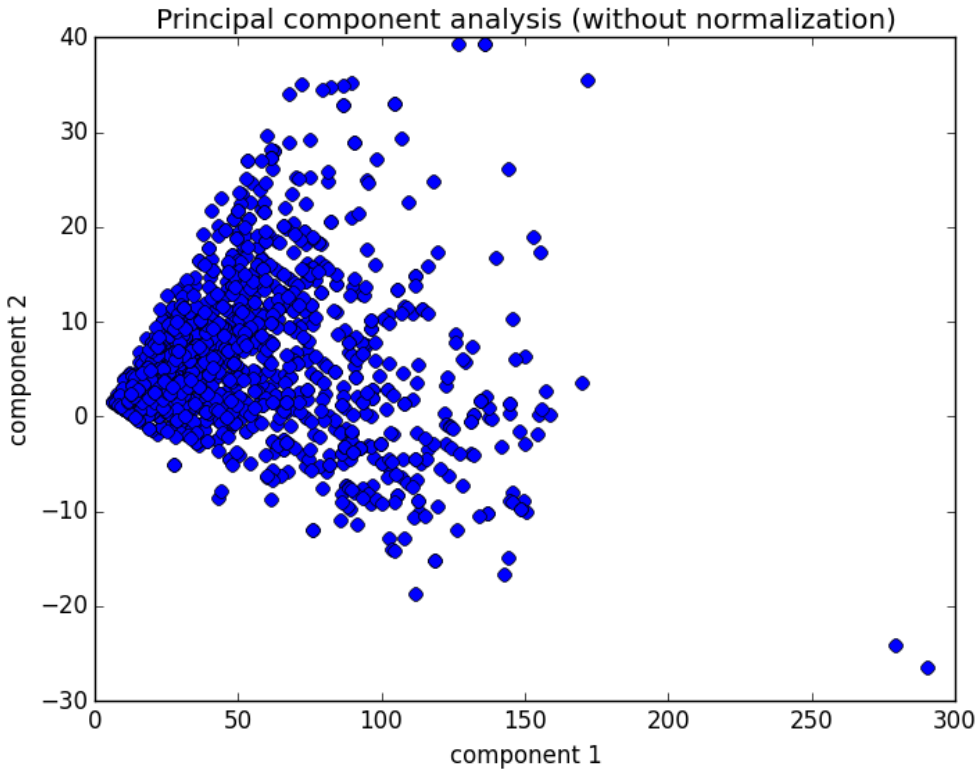
attributes' data using *matplotlib's scatter()* method. For the parallel coordinates presentation I used *pandas' parallel_coordinates* and chose to present the data in relation to the wine "quality" attribute.

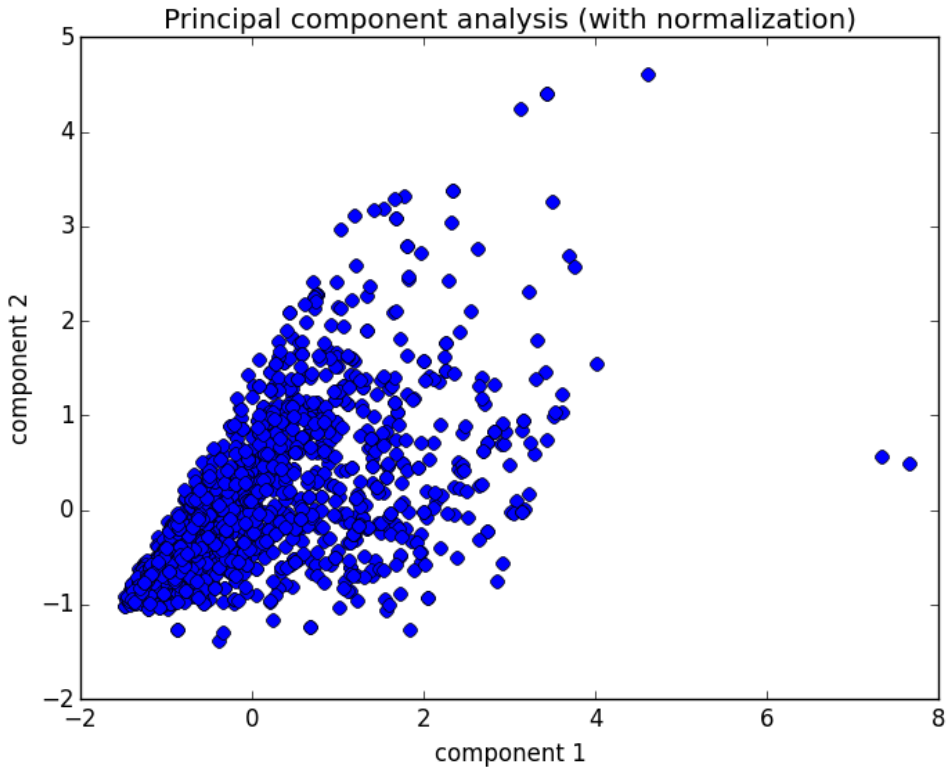


Above: parallel coordinates presentation

From the parallel coordinates presentation we can see that the "total sulfur dioxide" attribute has significantly larger values than the other attributes. Therefore it dominates the plot over the other attributes. We could get better information by zooming and scaling the plot.

3. Principal component analysis (PCA)





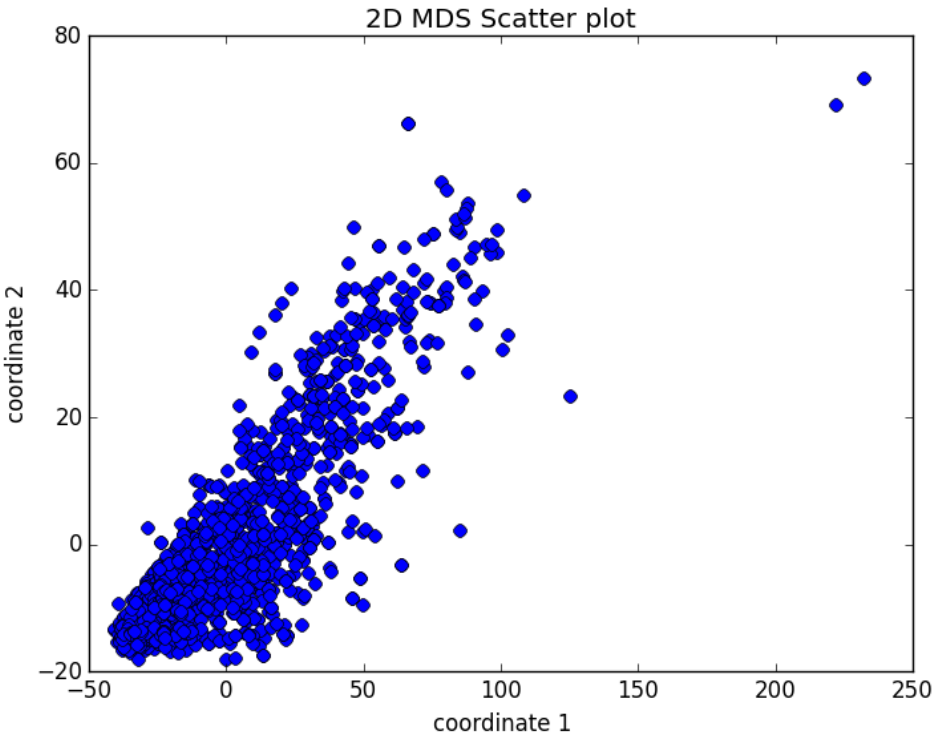
For the principal component analysis I needed to calculate the covariance matrix of the data. For this I first turned the data array I had into a feature matrix, and then used *numpy's cov()* function to get the covariance matrix of the data.

Now I could get the eigenvectors and eigenvalues from the covariance matrix, and stored them to corresponding lists (eigenValues, eigenVectors). Next I created a list of eigenvalue, eigenvector tuples, and sorted this list into decreasing order by the eigenvalues. Since for this task I needed the first two principal components, I chose and stored the two eigenvectors with the largest eigenvalues.

Next, I formed a two dimensional matrix by stacking those two eigenvectors (by using *numpy's hstack()* method), and then used this new matrix to transform the samples onto the new subspace by transposing and multiplying it with the original data matrix. After this I created a scatter plot of the resulting matrix.

For data normalization I used the z-score standardization (*scipy's zscore()* method). By normalizing the data, the principal components have a similar scale, which makes the visualization easier to read. With normalization, we can see that there's a positive correlation between the principal components. We can also see that, by reducing dimensions, we have managed to simplify the data and the outliers are easy to spot.

4. Multidimensional scaling (2D)



For the multidimensional scaling I created a distance matrix using Euclidean distance. From the Euclidean distance matrix, I did the 2D multidimensional scaling by using *sklearn's manifold* library's *MDS()* method. After this, I needed to calculate the coordinates for the new 2D space and create the scatter plot.

What's visible from the visualization is that the shape looks very similar to the PCA plot with normalization, with the scale similar to the PCA plot without normalization. The data is very much clustered towards the left side of the plot, like in PCA.

5. Tau correlation tables

Pearson's tau correlation table

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1.000000	-0.256131	0.671703	0.114777	0.093705	-0.153794	-0.113181	0.668047	-0.682978	0.183006	-0.061668	0.124052
volatile acidity	-0.256131	1.000000	-0.552496	0.001918	0.061298	-0.010504	0.076470	0.022026	0.234937	-0.260987	-0.202288	-0.390558
citric acid	0.671703	-0.552496	1.000000	0.143577	0.203823	-0.060978	0.035533	0.364947	-0.541904	0.312770	0.109903	0.226373
residual sugar	0.114777	0.001918	0.143577	1.000000	0.055610	0.187049	0.203028	0.355283	-0.085652	0.005527	0.042075	0.013732
chlorides	0.093705	0.061298	0.203823	0.055610	1.000000	0.005562	0.047400	0.200632	-0.265026	0.371260	-0.221141	-0.128907
free sulfur dioxide	-0.153794	-0.010504	-0.060978	0.187049	0.005562	1.000000	0.667666	-0.021946	0.070377	0.051658	-0.069408	-0.050656
total sulfur dioxide	-0.113181	0.076470	0.035533	0.203028	0.047400	0.667666	1.000000	0.071269	-0.066495	0.042947	-0.205654	-0.185100
density	0.668047	0.022026	0.364947	0.355283	0.200632	-0.021946	0.071269	1.000000	-0.341699	0.148506	-0.496180	-0.174919
pH	-0.682978	0.234937	-0.541904	-0.085652	-0.265026	0.070377	-0.066495	-0.341699	1.000000	-0.196648	0.205633	-0.057731
sulphates	0.183006	-0.260987	0.312770	0.005527	0.371260	0.051658	0.042947	0.148506	-0.196648	1.000000	0.093595	0.251397
alcohol	-0.061668	-0.202288	0.109903	0.042075	-0.221141	-0.069408	-0.205654	-0.496180	0.205633	0.093595	1.000000	0.476166
quality	0.124052	-0.390558	0.226373	0.013732	-0.128907	-0.050656	-0.185100	-0.174919	-0.057731	0.251397	0.476166	1.000000

Kendall's tau correlation table

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1.000000	-0.185197	0.484271	0.155029	0.176043	-0.119301	-0.056879	0.457461	-0.527832	0.141343	-0.048870	0.087966
volatile acidity	-0.185197	1.000000	-0.428354	0.022407	0.109608	0.012573	0.063701	0.015913	0.158746	-0.228888	-0.151839	-0.300779
citric acid	0.484271	-0.428354	1.000000	0.123007	0.076729	-0.049804	0.011645	0.245729	-0.389752	0.226669	0.064004	0.167318
residual sugar	0.155029	0.022407	0.123007	1.000000	0.152415	0.052682	0.102265	0.295986	-0.063127	0.026959	0.081206	0.025744
chlorides	0.176043	0.109608	0.076729	0.152415	1.000000	0.000439	0.091610	0.287866	-0.162706	0.014227	-0.197176	-0.148919
free sulfur dioxide	-0.119301	0.012573	-0.049804	0.052682	0.000439	1.000000	0.606908	-0.028972	0.079300	0.031706	-0.056019	-0.045646
total sulfur dioxide	-0.056879	0.063701	0.011645	0.102265	0.091610	0.606908	1.000000	0.087719	-0.006798	-0.000194	-0.179212	-0.156612
density	0.457461	0.015913	0.245729	0.295986	0.287866	-0.028972	0.087719	1.000000	-0.217228	0.110191	-0.329754	-0.136611
pH	-0.527832	0.158746	-0.389752	-0.063127	-0.162706	0.079300	-0.006798	-0.217228	1.000000	-0.053568	0.125311	-0.034235
sulphates	0.141343	-0.228888	0.226669	0.026959	0.014227	0.031706	-0.000194	0.110191	-0.053568	1.000000	0.143745	0.299270
alcohol	-0.048870	-0.151839	0.064004	0.081206	-0.197176	-0.056019	-0.179212	-0.329754	0.125311	0.143745	1.000000	0.380367
quality	0.087966	-0.300779	0.167318	0.025744	-0.148919	-0.045646	-0.156612	-0.136611	-0.034235	0.299270	0.380367	1.000000

In this task I created two methods (One for Pearson's tau table and one for Kendall's tau table) that calculated the correlation coefficients between each pair of attributes, created a table out of those coeffients, and then wrote those tables into HTML files for easy output.

There are no interesting correlation factors (lower than -0.7 or higher than 0.7) to be spotted from either table. What can be spotted, however, (in addition to the correlations between the different acidity related attributes), there's also a slight correlation factor between the "density" and "fixed acidity" attributes (0.668047 Pearson, 0.457461 Kendall), and between the "alcohol" and "quality" attributes (0.476166 Pearson, 0.380367 Kendall).