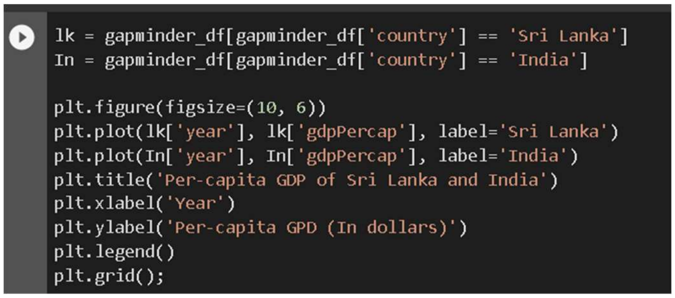
TODO 1:

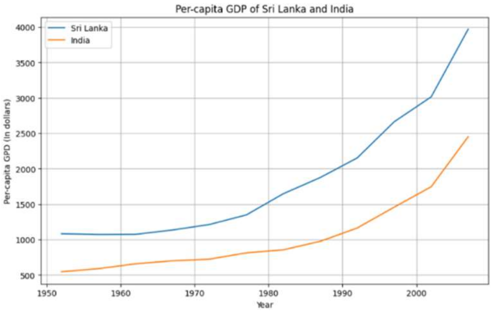
The more bins we specify, the narrower the bins will be. And the graph is more precise. But if we have too many bins, then the data distribution will look rough, and the details of the graph will be fewer. Therefore, the bin size will be determined according to the requirement. We can determine the bin size by looking at the data set or with Sturge's Rule which is another way to calculate bin size using a formula. ($K = 1 + 3.\ 322\log_{N}$)

TODO 2:



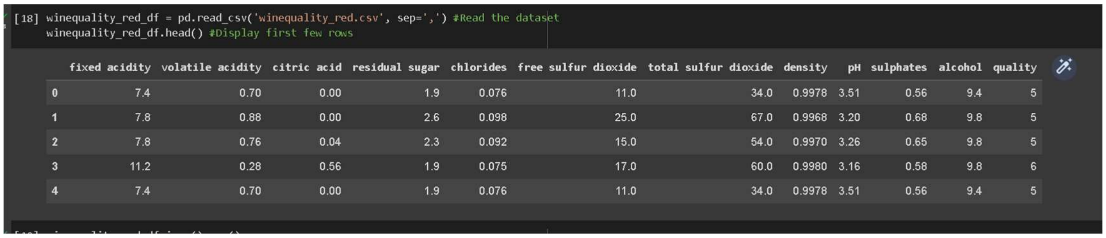
```
lk = gapminder_df[gapminder_df['country'] == 'Sri Lanka']
In = gapminder_df[gapminder_df['country'] == 'India']

plt.figure(figsize=(10, 6))
plt.plot(lk['year'], lk['gdpPercap'], label='Sri Lanka')
plt.plot(In['year'], In['gdpPercap'], label='India')
plt.title('Per-capita GDP of Sri Lanka and India')
plt.xlabel('Year')
plt.ylabel('Per-capita GPD (In dollars)')
plt.legend()
plt.grid();
```



TODO 3: Exploratory data analysis (EDA)

1.



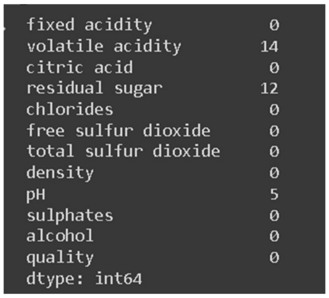
```
[18] winequality_red_df = pd.read_csv('winequality_red.csv', sep=',') #Read the dataset
     winequality_red_df.head() #Display first few rows
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

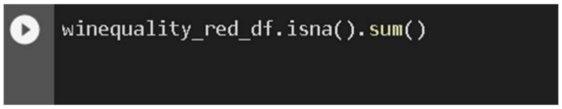
2. **Independent**– fixed acidity, Volatile, acidity, Citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol

Dependent – quality

3. (a)



```
fixed acidity            0
volatile acidity        14
citric acid              0
residual sugar          12
chlorides                0
free sulfur dioxide      0
total sulfur dioxide     0
density                  0
pH                       5
sulphates                0
alcohol                  0
quality                  0
dtype: int64
```



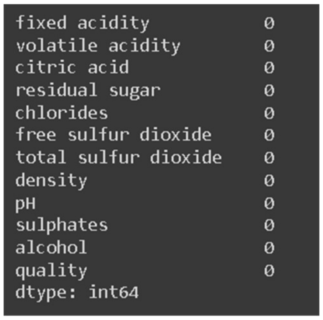
```
winequality_red_df.isna().sum()
```

There are some missing values.

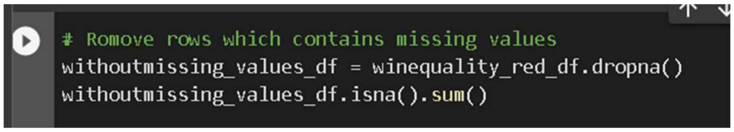
(b)

- Remove rows with empty cells: dropna()
- Replace Empty values: fillna(), replace(), interpolate()
- Using a separate category (treat them as a separate category altogether)

(c)



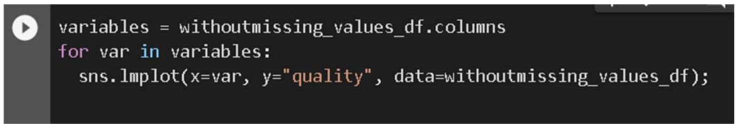
```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```



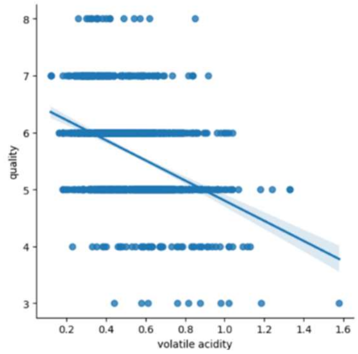
```
# Romove rows which contains missing values
withoutmissing_values_df = winequality_red_df.dropna()
withoutmissing_values_df.isna().sum()
```

Used dropna() to remove empty cells

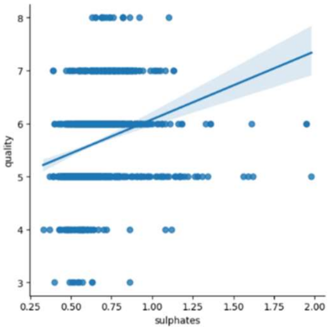
4.



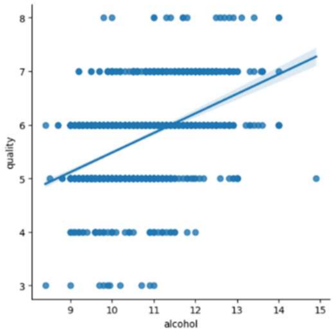
```
variables = withoutmissing_values_df.columns
for var in variables:
  sns.lmplot(x=var, y="quality", data=withoutmissing_values_df);
```



Volatile acidity

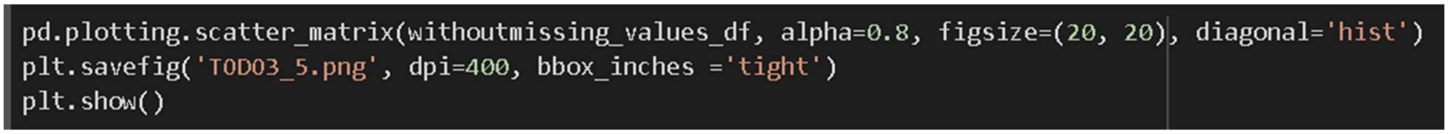


Sulphates

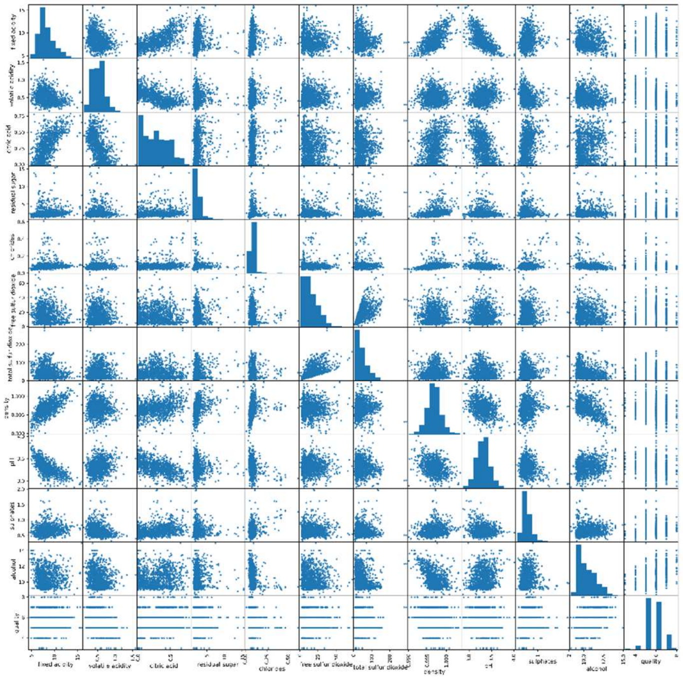


Alcohol

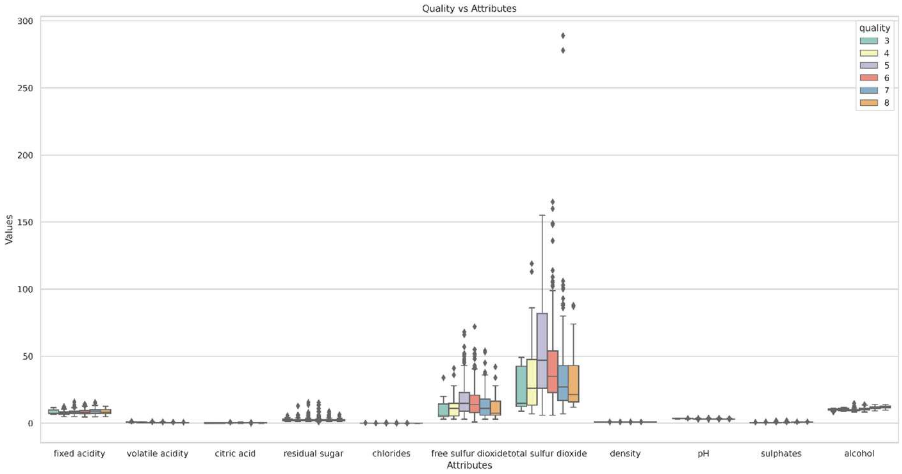
5.



```
pd.plotting.scatter_matrix(withoutmissing_values_df, alpha=0.8, figsize=(20, 20), diagonal='hist')
plt.savefig('TODO3_5.png', dpi=400, bbox_inches ='tight')
plt.show()
```



6.



7.

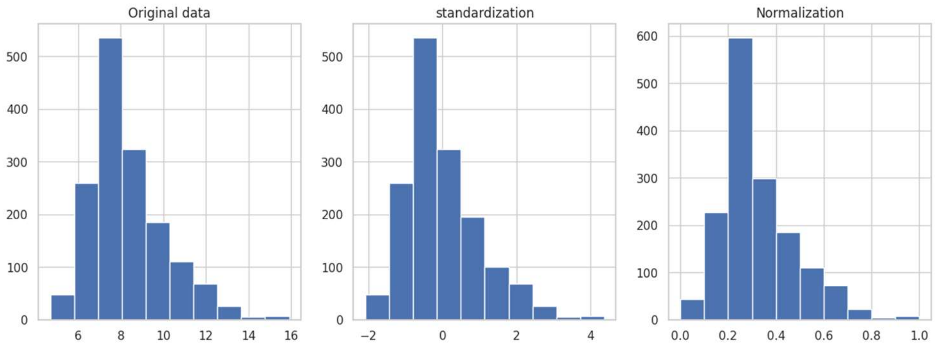
Yes, we can see some points away from the whiskers. They are outliers.

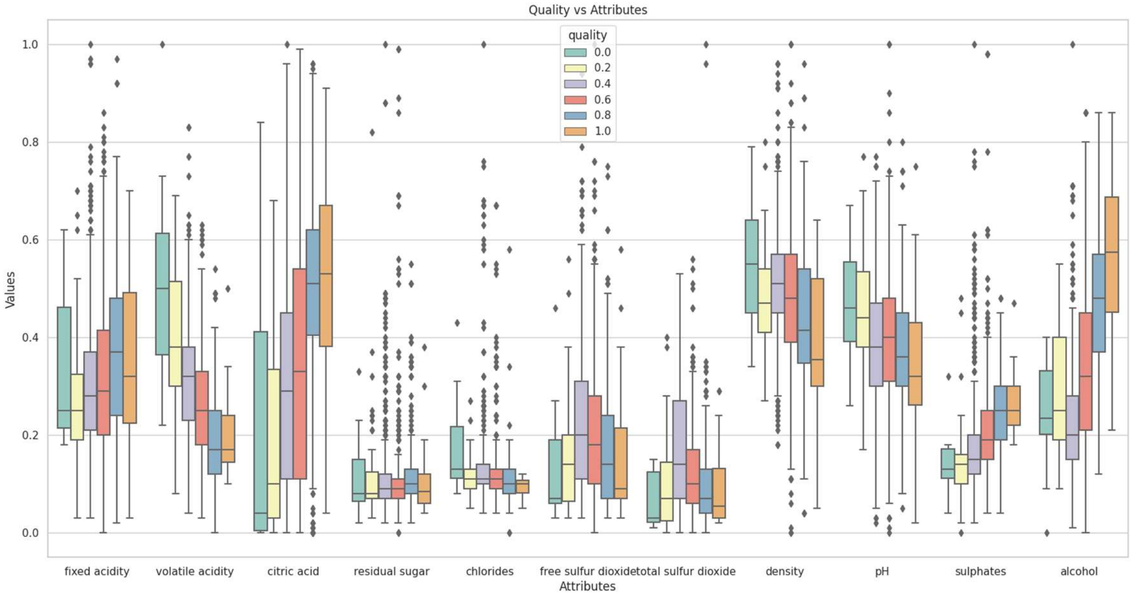
8. (a)

It transformed data into a structure where all the features are on a similar scale. It reduces the effect of outliers on the dataset and makes the effect of different attributes on the result the same. By standardizing the data, machine learning algorithms can work more effectively and efficiently, and the results will be more accurate and reliable.

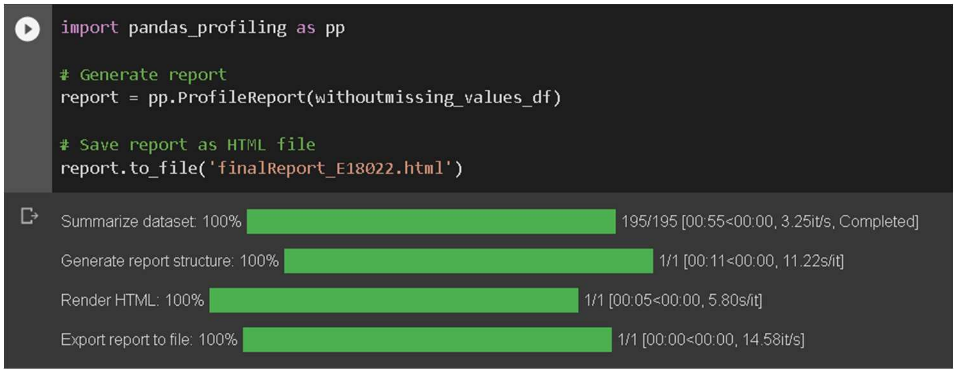
(b)

Normalization scales the data between 0 and 1, while standardization transforms the data to have a mean of 0 and a standard deviation of 1.





9.



```
import pandas_profiling as pp

# Generate report
report = pp.ProfileReport(withoutmissing_values_df)

# Save report as HTML file
report.to_file('finalReport_E18022.html')
```

```
Summarize dataset: 100% 195/195 [00:55<00:00, 3.25it/s, Completed]
Generate report structure: 100% 1/1 [00:11<00:00, 11.22s/it]
Render HTML: 100% 1/1 [00:05<00:00, 5.80s/it]
Export report to file: 100% 1/1 [00:00<00:00, 14.58it/s]
```

Google Colab Notebook –

https://colab.research.google.com/drive/13c2Nnix6pAx5Bwp-yMINQlmrwMB-Xxyx#scrollTo=0NL97uYTlQlH

Final Report –

https://drive.google.com/file/d/1wV-z2gpRqEBOU7Qan2bPFoiCXKvG2L0w/view?usp=share_link