姓名：林成鍇 Julius Ling

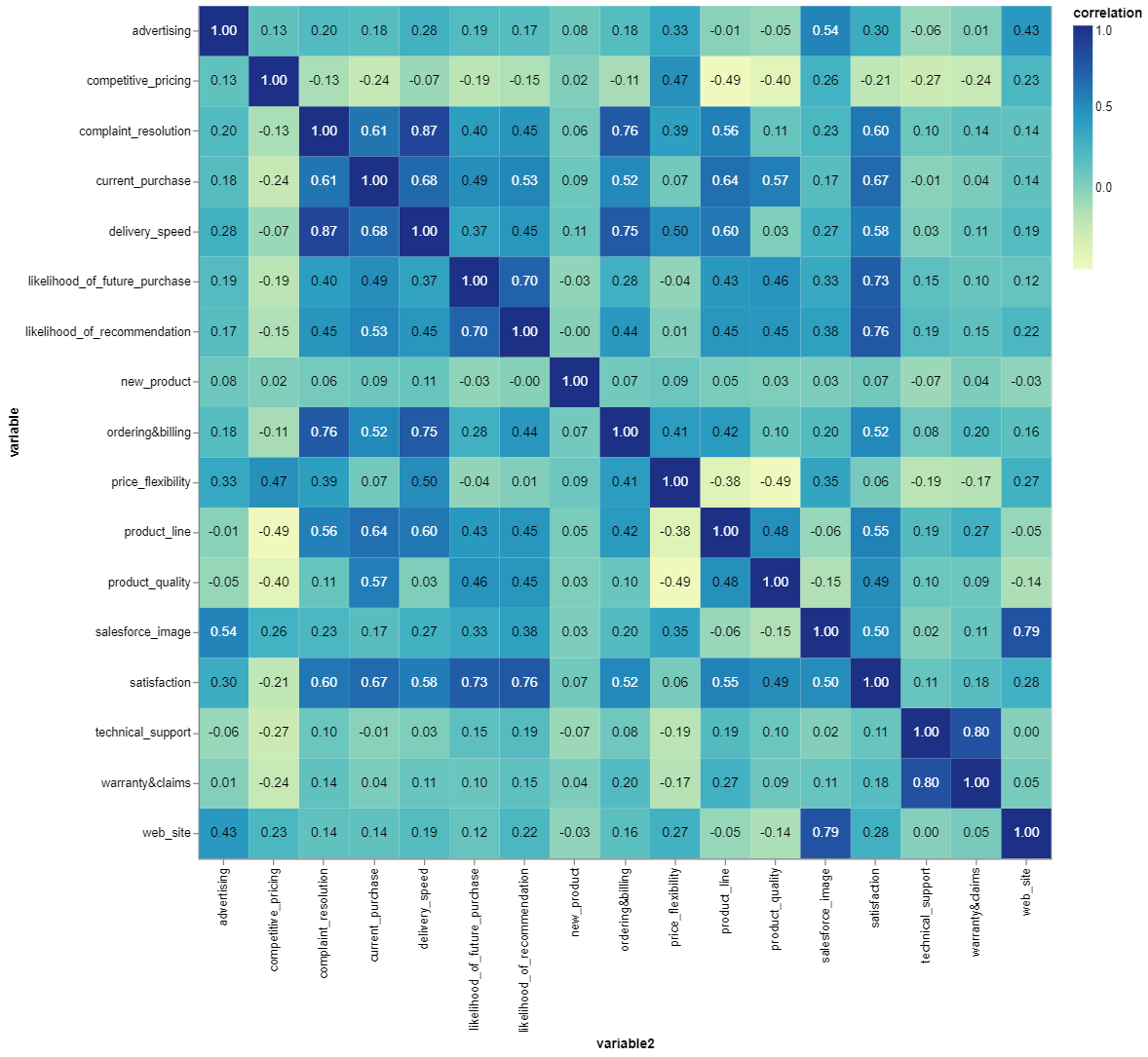
學號：D84085360

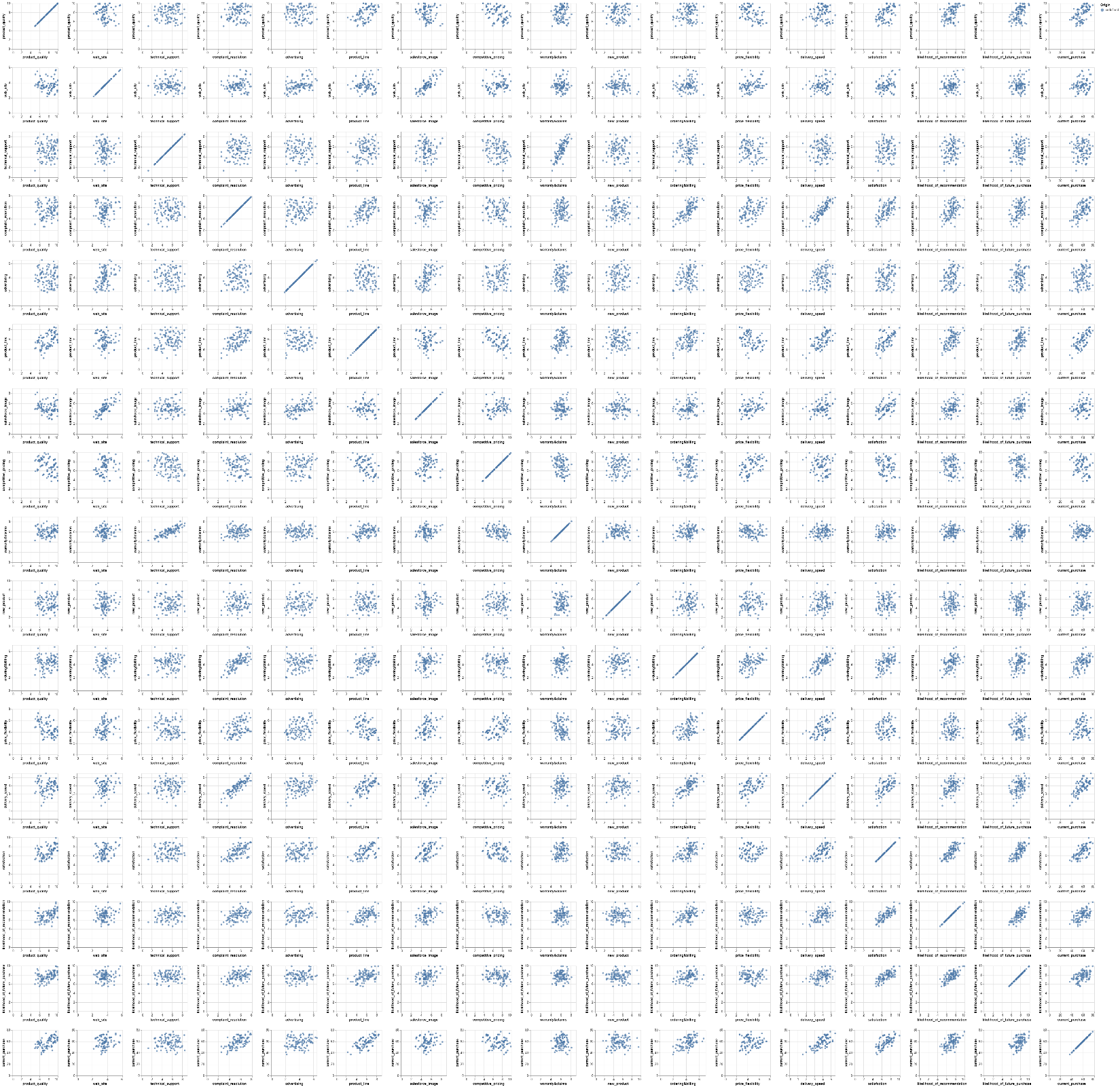
系級：心理112

\*\*\* This assignment is performed with *Python* coding. For detailed display in IPython Notebook, click the link below.

<https://colab.research.google.com/drive/16JEnOLj5Nm0bispJ-u5qfgqjGRB6CTm0?usp=sharing>

Dataset I：HBAT.sav (Data Profiling, Normality Check & Outlier Detection)

 I am to know the Linear Correlation between the variables with Correlation Heatmap

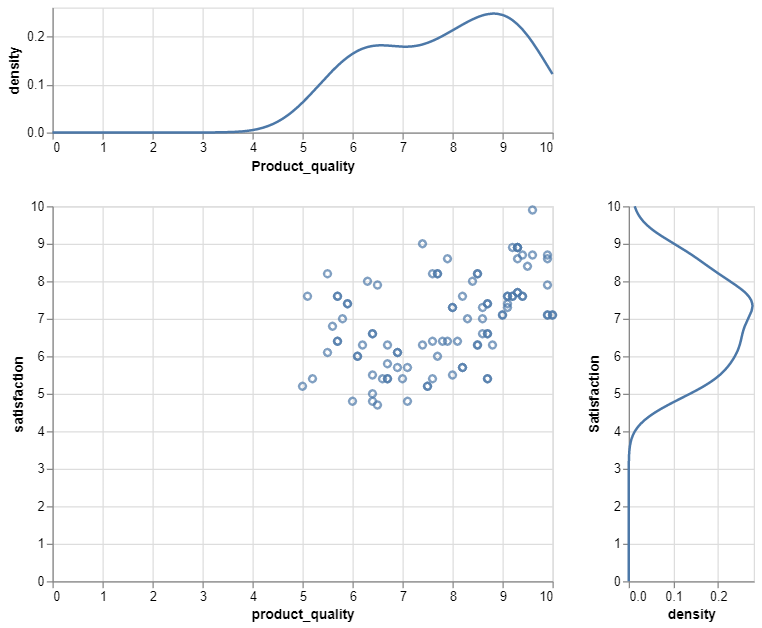
Grid Scatter Plot works to visually check the trend lies within the variable, particularly fits well with the Correlation Heatmap in terms of gaining insights about the multivariate dataset.

Certain Variables of interest are focused:

* Product\_quality
* Satisfaction

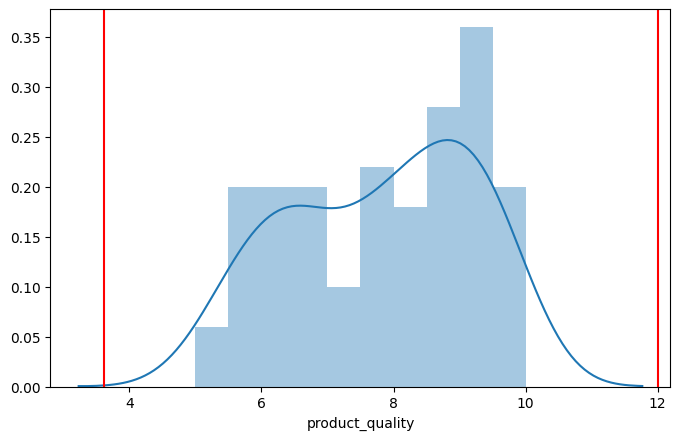
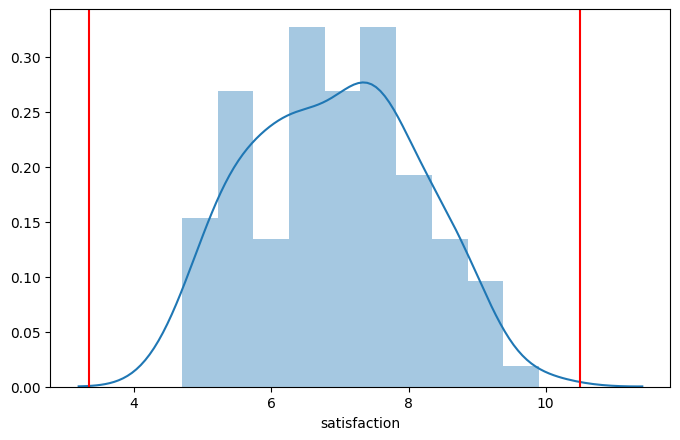
The correlation between these two variables is a mere 0.49, it’s quite intriguing to me.

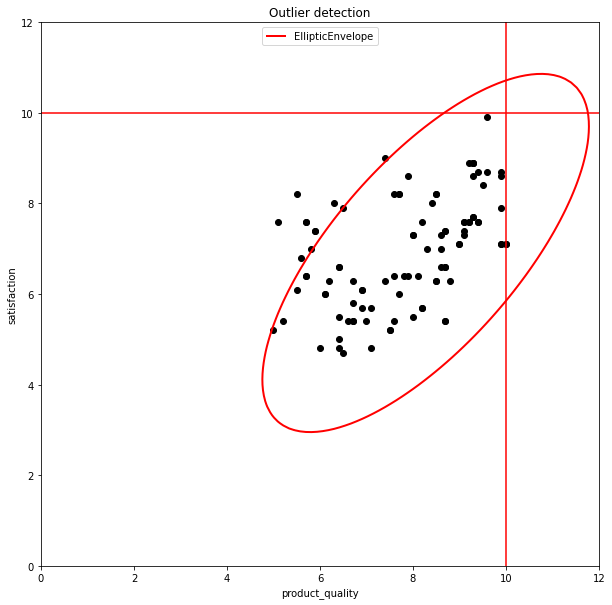
And here’s the Scatter Plot in detail, accompanying with Marginal Density Plot:



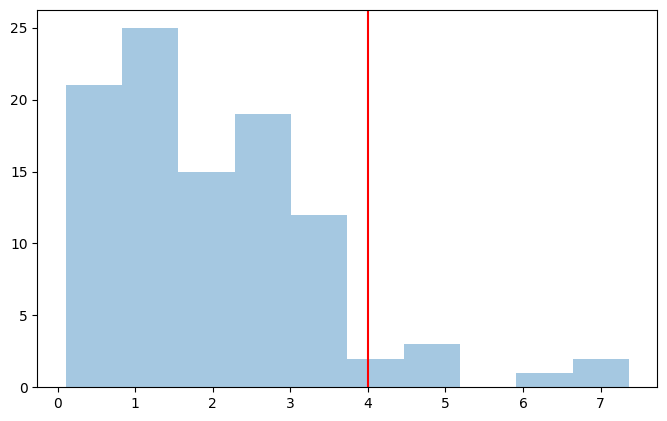
Note that the Density Plots do show hump-like centralized trend.

And their respective Histograms are plotted (& no outlier detected in terms of lying beyond 3\*standard deviation)



However the Elliptic Envelope does point out some of the possible outliers under the bivariate setting as the raw *Minimum Covariance Determinant*algorithm applied under the default setting.

Mahalanobis is calculated its respective Histogram is plotted, assume those > 4 are the possible outliers .



Result of the Kurtosis Test on the Satisfaction:

KurtosistestResult(statistic=-2.479922184687465, pvalue=0.013141105779122239)

Result of the Skewness Test on the Satisfaction:

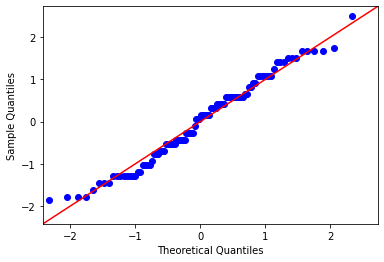
SkewtestResult(statistic=0.33420807842092776, pvalue=0.7382225484444592)

Result of the Kolmogorov-Smirnov Test on the Satisfaction:

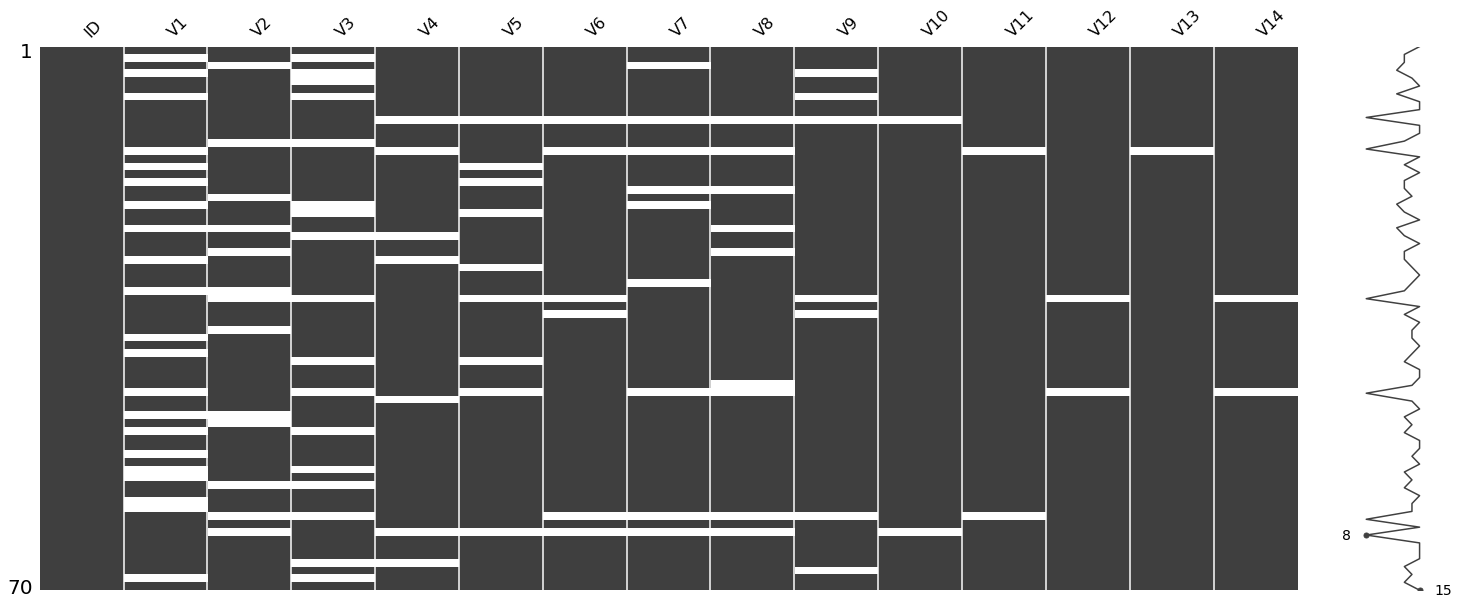
KstestResult(statistic=0.9999986991925461, pvalue=0.0)

Statistical Insignificance justifies the assumption of Normality.

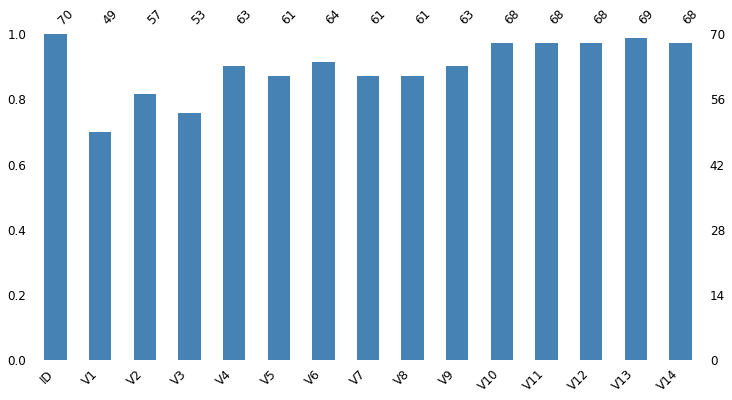
Picture below are the Quantile-quantile Plot of the sampling distribution of “Satisfaction” in comparison with normal distribution, further supporting the assumption of Normality.



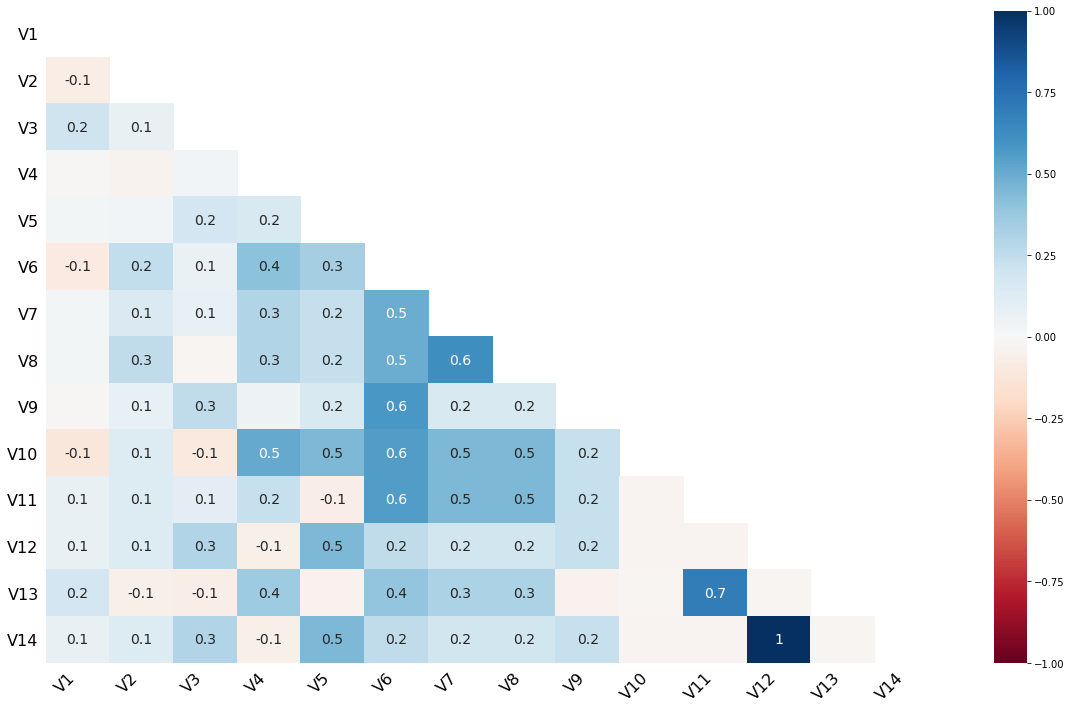
Dataset II: HBAT\_MISSING.sav (Missing Data Handling)

\*\*\* The visualization is performed with trendy missing data visualizing package: *missingno.*

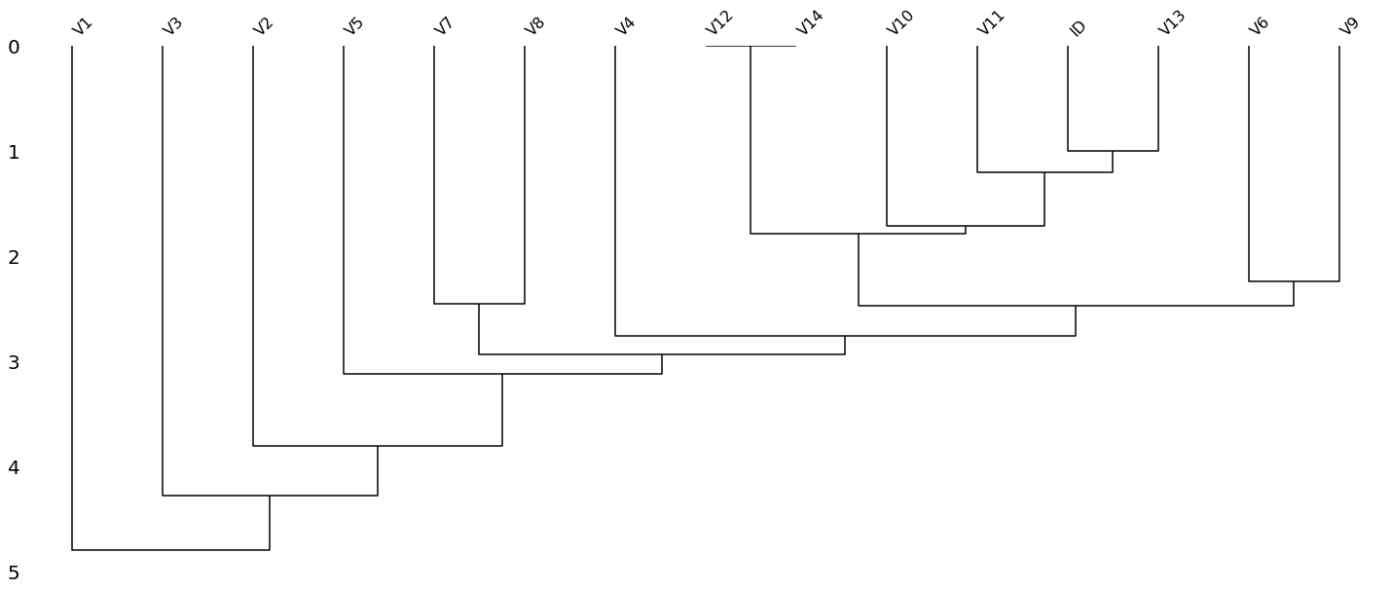
Speed overview of the missing data within the dataset (the white tick = missing value) and it is notable that some of the subjects have continuous pattern of missing value worth investigating.



Sample size = 70. The numbers above the bar tell the number of complete cases within each variable. Note that V1 has the most missing values.



The Correlation Heatmap measures Nullity Correlation: how strongly the presence or absence of one variable affects the presence of another. Nullity correlation ranges from -1 (if one variable appears the other definitely does not) to 0 (variables appearing or not appearing have no effect on one another) to 1 (if one variable appears the other definitely also does). Certain variables worth checking out in detail are framed in red.



To interpret this *Dendrogram*, read it from a top-down perspective. Cluster leaves which linked together at a distance of zero fully predict one another's presence—one variable might always be empty when another is filled, or they might always both be filled or both empty, and so on. In this specific example the dendrogram glues together the variables which are required and therefore present in every record. The frames are to show the respective relationship framed in the Nullity Correlation Heatmap.

Imputation of the Data:

\*\*\*The following section is not easy to be visualize so I show the codes instead.

Imputation with Mean / Mode / Median:

df.fillna({"V2":df["V2"].mean()})

Imputation with Linear Regression:

df\_rg = df[['V1','V4']]

df\_rg.dropna(inplace = True)

x = df\_rg[['V1']]

y = df\_rg["V4"]

from sklearn import linear\_model

lr\_sklearn = linear\_model.LinearRegression(fit\_intercept = True, normalize = False)

lr\_fitted\_sklearn = lr\_sklearn.fit(x.values, y.values)

y\_embed = df["V4"].isnull()

x\_tobe\_pred = pd.DataFrame(df["V1"][y\_embed])

x\_tobe\_pred.dropna(inplace=True)

y\_pred = lr\_fitted\_sklearn.predict(x\_tobe\_pred)

x\_tobe\_pred["y\_pred"] = 0

for i in x\_tobe\_pred.index:

  j = 0

  x\_tobe\_pred.loc[i, "y\_pred"] = y\_pred[j]

  j += 1

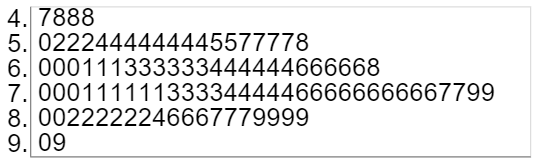
for i in x\_tobe\_pred.index:

  df.loc[i,"V4"] = x\_tobe\_pred.loc[i,"y\_pred"]

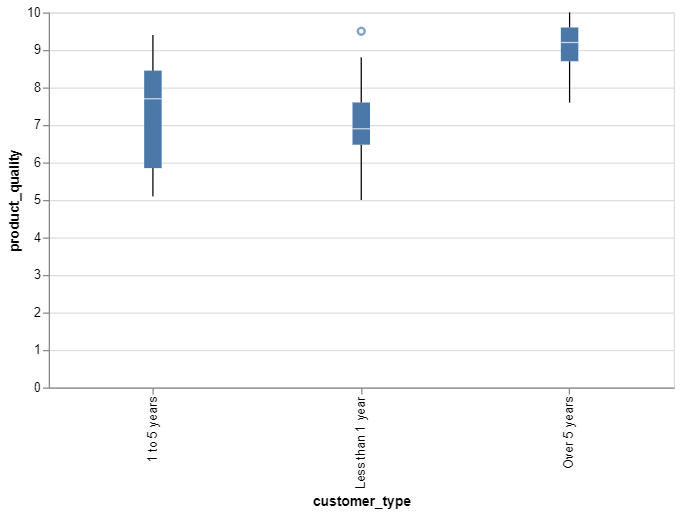
Appendix:

\*\*\* Here’s the Visualization I made for practice. The detailed codes display is in the IPython Notebook I have shared at the very beginning.

Stem-Leaf Plot (“Satisfaction”):



Box Plot (Categorical Variable + Numerical Variable):



Line Chart:

