skema BUSINESS SCHOOL

Academic Project-Data Science

Detection of Automatic Flush System

GROUP 7 – MSC DBDAM – 09/12/2020

HIMPENS DORIAN –RAMESH REBBA– FELDERHOFF NOÉ – WANG SHIHUA

Table of Content

I/ Our Methodology 1

1. Description of the DataSet 1
2. Our assumption about the independent variables 2

II/Our prediction models 7

1. Method PCA 7
2. Method K-Means 10
3. Method OLS 12
4. Method GLM 15

III/ Discussion 18

Python Code 19

I/ Our Methodology

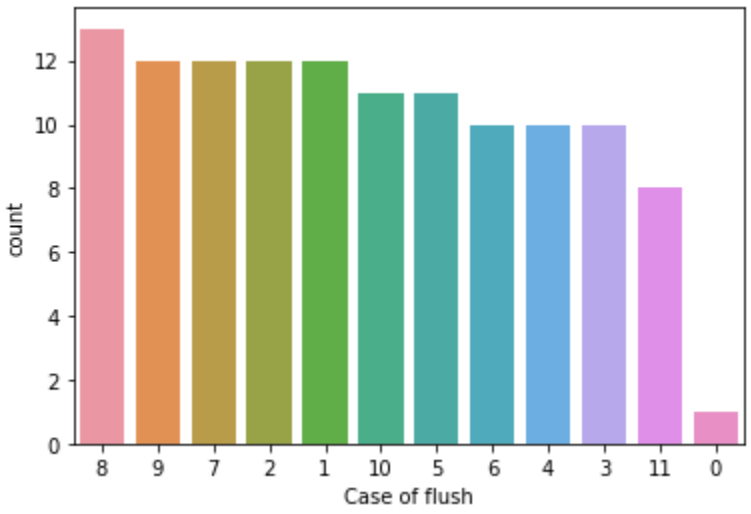
First, we have decided to work on correlations of the different variables of the data set. We pass different test of the data set. We want to understand more the correlation between the variables to have an overview of the system.
In a second part we apply the different Methods for predictions to see which one would be the best for the prediction and the improvement of the model already developed by the company.
In the last part, we discuss on the different points that could have make our work and predictions more accurate.

1. Description of the DataSet

This dataset contains 122 observations which represents the test the company made with different level of waste inside the toilet. This dataset also contains 13 important columns that we will use in order to build a predictive model. These columns contain the information regarding the led and photodiodes (12 columns) and the level of water needed in order to clean the toilet (case of flush).

What is our dependent variable?

The goal of the company Z is to create a contactless flush solution which is automated. In order to automate it, we need to predict the flush volume needed (Y) according to the information collected by the sensors (X) which represents the degree of waste inside the toilet.
As the company needs to implement the model inside the toilet in a little chip, the simpler the model, the better it is. After calculating the numbers of each flush volume, we found that the flush volume (our dependent variable Y) is a discrete variable. value ranging from 0 to 11. This value is available in the dataset under the column “Case of flush”.



In this dataset, we can see that there is only one experiment in which the outcome is 0. In this case, it will be hard to predicted well in our model since we do not have a lot of information regarding the independent variables of this outcome. In the other hand, all the others dependent variables have 10 (or almost 10 for the Case of flush 11) different experimentations. It means that we have enough information in this dataset to predict the dependent variables Case of flush.

What are our independent variables?

As mentioned earlier, our X is composed by the information collected by the two sensors. In the dataset, this information is categorized in 12 different columns (6 different columns per led composed by two pictures recaptioning three colours).

These 12 columns are our independent variables. In order to describe them, we have decided to merge these columns into 3 new ones which represent all the information available for each of the three colours (red, blue, green).

2. Our assumption about the independent variables

Regrouping the information by colours will also allow us to confirm or infirm our assumption. Indeed, our assumption is that the higher the intensity of a colour will be, the less volume of water would be needed. As the intensity is high, it would mean that the light can reflects in the water, thus be captured by the sensors. In the other hand, if the intensities of colours are low, it would mean that the light could not reflects on the water, the waste obstructing it. Making the sensors enable to catch them.

Description of each colour:

	Red	Blue	Green	Sum
Mean	1247.1	403.4	404.1	2054.7
Std	861.6	487.8	284.4	1493
Median	1302	237	414.5	2017
[Min-Max]	[1-2920]	[0-2364]	[0-1181]	[0-6394]

Description of each colour grouped by case of flush:

Mean:

Case of flush	Red	Blue	Green	Sum
0	2240	1795	928	4963
1	2275.6	770.8	754.2	3800.6
2	2230.2	632.5	655.5	3518.2
3	2198	650.5	654.9	3503.4
4	1823	456.8	533.7	2813.6
5	1298.4	456.27	447.6	2202.3
6	1173.2	404	396.4	1973.6
7	970.5	344.75	349.3	1664.6
8	645.9	172.69	210	1028.6
9	514.7	203	193.6	911.3
10	288.4	136.72	122.5	547.6
11	28.2	16.5	11.1	55.9

Median:

Case of flush	Red	Blue	Green	Sum
0	2240	1795	928	4963
1	2182.5	526	725	3593.5
2	2231.5	337	594	3094.5
3	2337	301	600.5	3192.5
4	2002	281.5	547.5	2802
5	1360	315	442	2279
6	1270	277.5	401.5	1884.5
7	1093.5	288	380.5	1680
8	454	119	200	1052
9	537	93.5	187	943
10	83	49	31	163
11	13.5	4.5	3.5	21

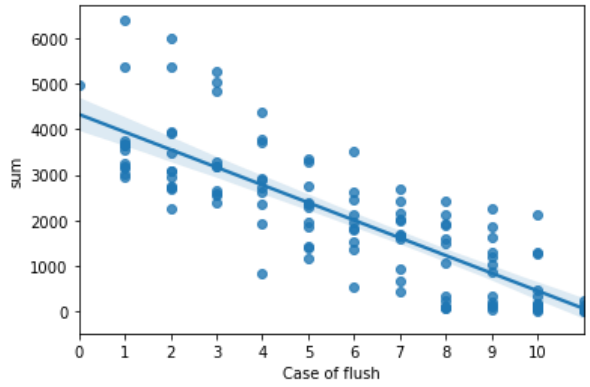
Minimum/Maximum:

Case of flush	Red	Blue	Green	Sum
0	[2240]	[1795]	[928]	[4963]
1	[2109-2849]	[279-2364]	[571-1181]	[2959-6394]
2	[1617-2750]	[215-2139]	[393-1101]	[2256-5990]
3	[1660-2920]	[155-1893]	[428-1001]	[2396-5259]
4	[398-2587]	[154-1542]	[144-794]	[828-4365]
5	[771-1589]	[120-1183]	[227-626]	[1164-3339]
6	[366-1594]	[62-1287]	[90-680]	[518-3521]
7	[254-1483]	[103-981]	[90-535]	[447-2692]
8	[40-1595]	[7-885]	[9-478]	[56-2423]
9	[24-1262]	[7-685]	[9-498]	[40-2249]
10	[7-922]	[3-778]	[3-427]	[14-2127]
11	[1-115]	[0-57]	[0-45]	[1-216]

Regression plots:

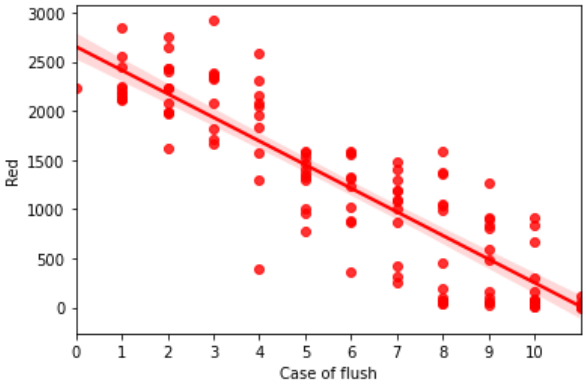
Regplot of all the diodes

Pearson's correlation coefficient: **-0.82**



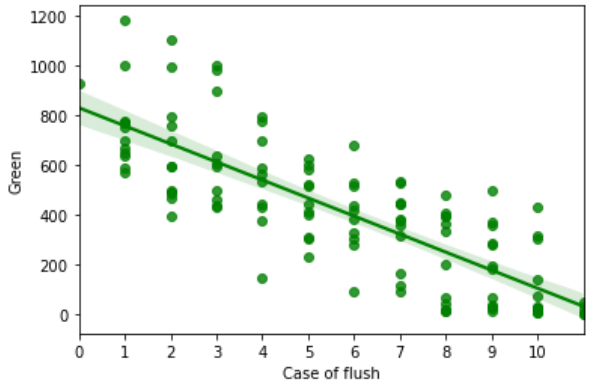
Regplot Red diodes

Pearson's correlation coefficient: **-0.88**



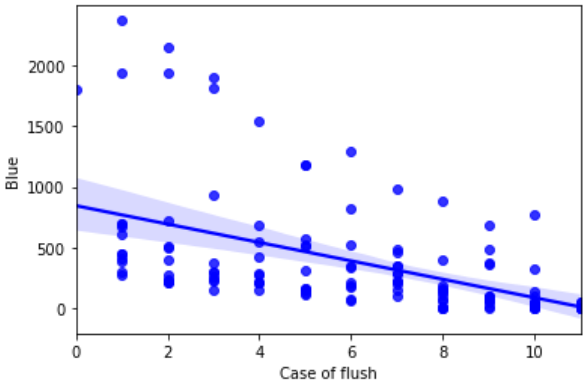
Regplot Green diodes

Pearson's correlation coefficient: **-0.81**



Regplot Blue diodes

Pearson's correlation coefficient: **-0.49**

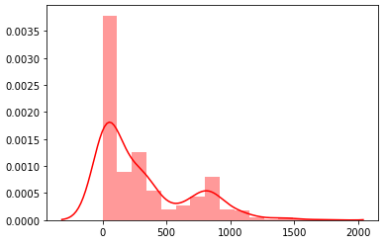


The regression plot as well as the Pearson's coefficient show a strong negative relationship between the colours and the case of flush value. It means that the higher the value of the diode is, the lower will be the case of flush. It confirms of assumptions that the more you have waster in the toilet, the more you will need water.

Distributions of the diodes :

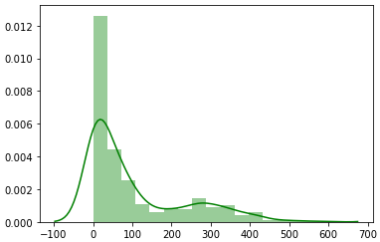
Red Distribution

P-value Shapiro: **1.83e-22**



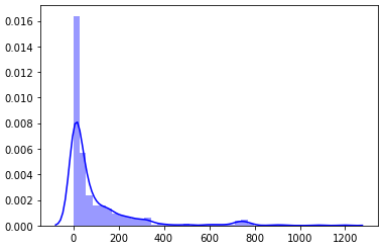
Green Distribution

P-value Shapiro: **8.23e-25**



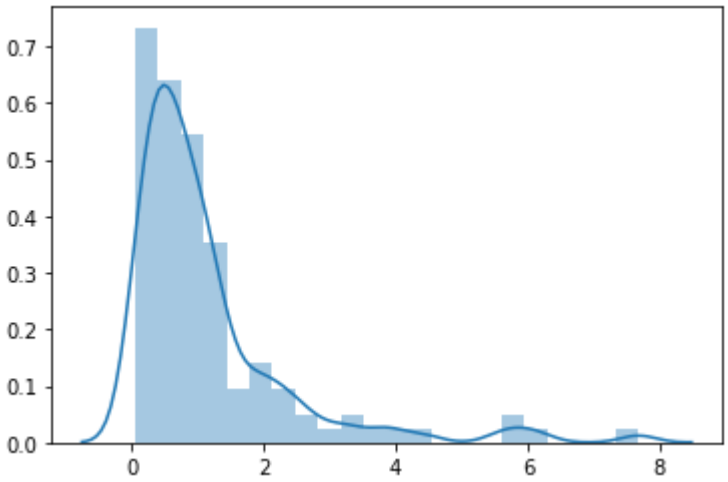
Blue Distribution

P-value Shapiro: **3.66e-32**

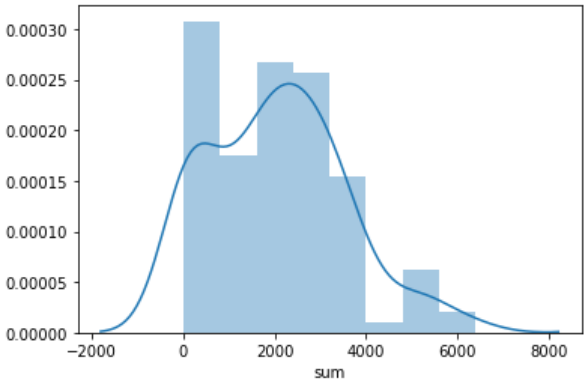


The distributions of the colours look like they are following the same distribution. All the P-values of the Shapiro test are close to 0. We thus reject H0 and can say that these distributions are not following a normal distribution. These distributions look like they are following a lognormal distribution:

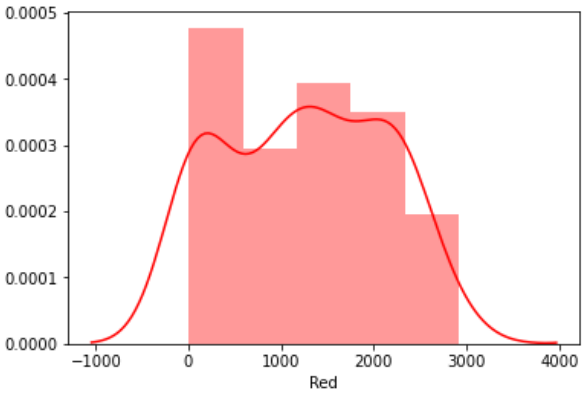
Lognormal Distribution (size=122)



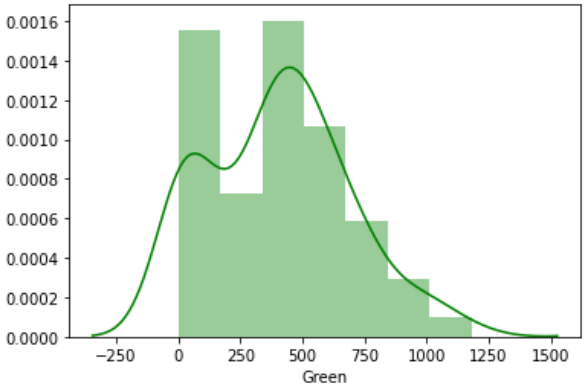
Distribution of all the diodes aggregated



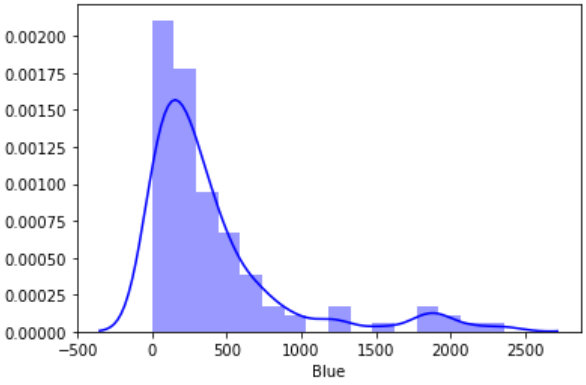
Distribution of all Red diodes aggregated



Distribution of Green diodes aggregated



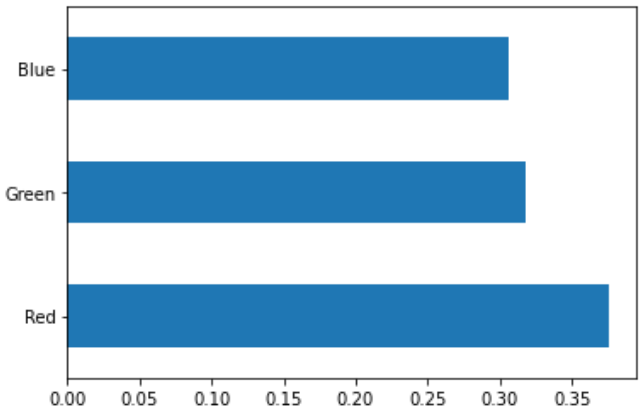
Distribution of Blue diodes aggregated



When each diode is aggregated per colour, we can see that the red colour has the highest intensity. Indeed, its intensity range from 0 to almost 3000 and its mode is between 0 and 1000 whereas

green range from 0 to almost 1250 with a mode around 500. Finally, blue range from 0 to almost 2500 but it has the lowest mode which is around 250. The findings coming from these distributions are confirmed by the earlier tables. Considering these findings, we can assume that red is the most important colour, followed by green and blue. This assumption can be confirmed by a python formula which can rank the most important features of a dataset.

Importance of feature:

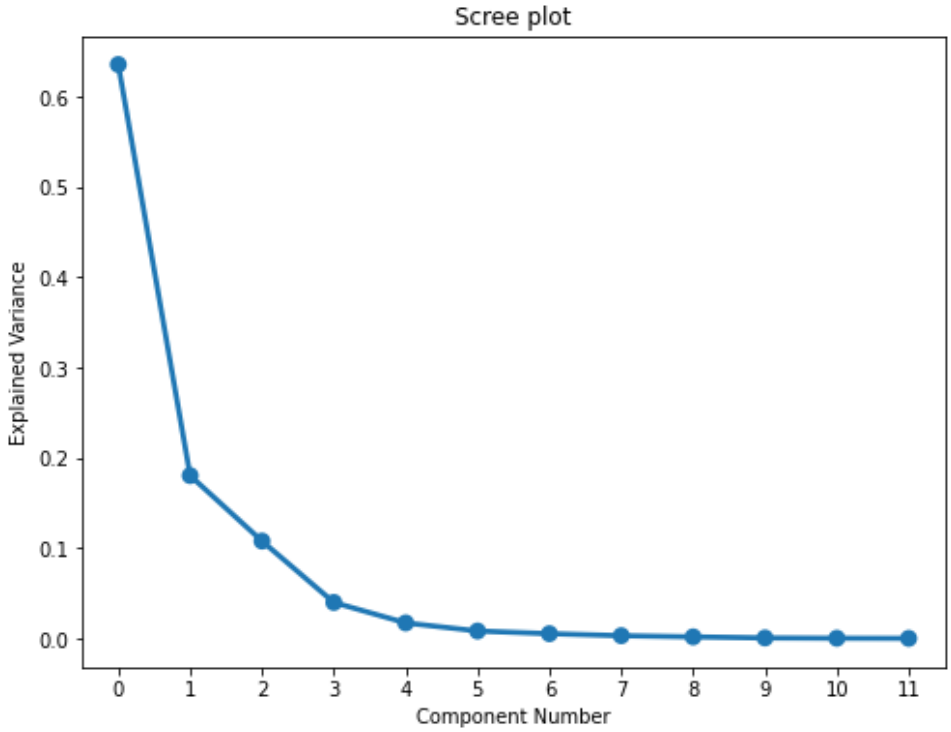


The python ExtraTreesClassifier model ranked the importance of the colour regarding the dependent variable "Case of flush". The model confirmed our assumption saying that red is the most important colour, followed by green and blue.
Thanks to the description made ahead, we can deduct that the colours (and thus all the 12 features) contain redundant information. The next part will be dedicated to our predictions and all the preparations steps before in order to create the most accurate model possible.

II/Our prediction models

1. Method PCA

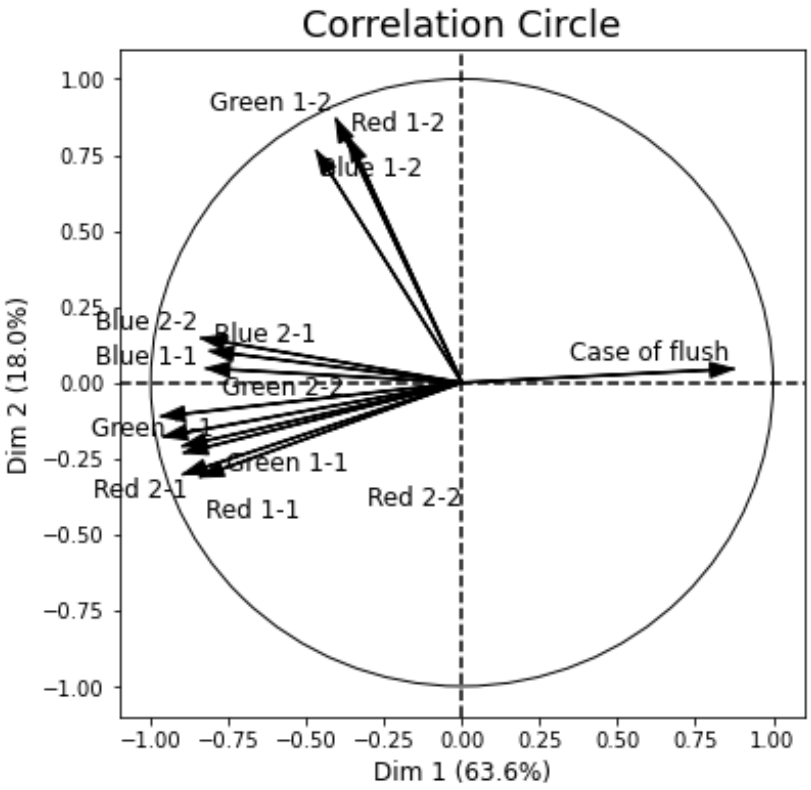
PCA:



Explained Variance per Component Number (rounded)											
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
0.64	0.18	0.10	0.04	0.017	0.008	0.005	0.003	0.002	0.0005	0.0002	0.00004
Cumulative Sum of Explained Variance (rounded)											
0.64	**0.82**	**0.92**	**0.96**	0.977	0.985	0.99	0.993	0.995	0.9955	0.9957	100 *rounded*

The Principal Component Analysis shows that with the first 4 components (out of 12), explain 96% of the variance of our dataset. Thus, it confirms our assumption of redundant information between the 12 different independent variables. In the end, this analysis shows that the 2 first components explain 82% of the dataset.

Correlation Circle:



Dim 1 Analysis: The intensity of Led 1 Photodiode 1, Led 2 Photodiode 1 and Led 2 Photodiode 2 of all three colours (Red 1-1, Red 2-1, Red 2-2, Green 1-1, Green 2-1, Green 2-2, Blue 1-1, Blue 2-1, Blue 2-2) highly influence DIM 1 through a negative correlation with Case of flush. It means that the tests which are on the left side of DIM 1 have low intensities from their sensors but a high Case of flush. In the other hand, if the tests on the right side of DIM 1 have high intensities from their sensors but a low Case of flush.

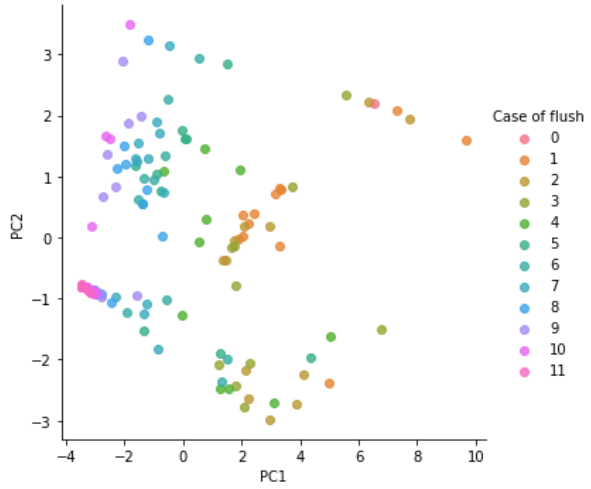
Dim 2 Analysis: The intensity of Led 1 Photodiode 2 of all three colours (Red 1-2, Green 1-2, Blue 1-2) has a strong influence on DIM 2. It means that the tests which are at the top of DIM 2 have strong intensities from Led 1 and Photodiodes 2. In the other hand, if the tests are at the bottom of Dim 2 have low intensities coming from Led 1 and Photodiodes 2.
This analysis can be reformulated into a 2-D Matrix which confirm the discoveries we made in the description part of the dataset (strong negative correlation between the intensities and Case of flush).

Matrix interpreting the position of the tests:

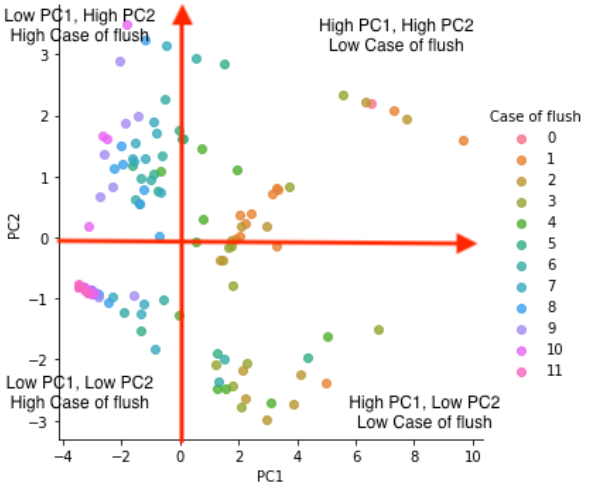
DIM 2	**Top Left:** Test with low intensities of 1-1, 2-1, 2-2 Led-Photodiodes but high intensities of 1-2 Led-Photodiodes and high case of flush	**Top Right:** Test with high intensities of 1-1, 2-1, 2-2 Led-Photodiodes and high intensities of 1-2 Led-Photodiodes and low case of flush
	Bottom Left: Test with low intensities of 1-1, 2-1, 2-2 Led-Photodiodes and low intensities of 1-2 Led-Photodiodes and high case of flush	**Bottom Right:** Test with high intensities of 1-1, 2-1, 2-2 Led-Photodiodes but low intensities of 1-2 Led-Photodiodes and low case of flush
	DIM 1	

Data Recasting along the PCA:

Raw recast along the 2 PCA



Recast with the interpretation from the matrix



The recast of the test on the two principal components shows consistency with the interpretation of the matrix. The test on the left (with an overall low intensity) trigger a high value of Case of flush (from 5 to 11) whereas the tests on the right (with an overall high intensity) trigger a low value of Case of flush (from 0 to 4).

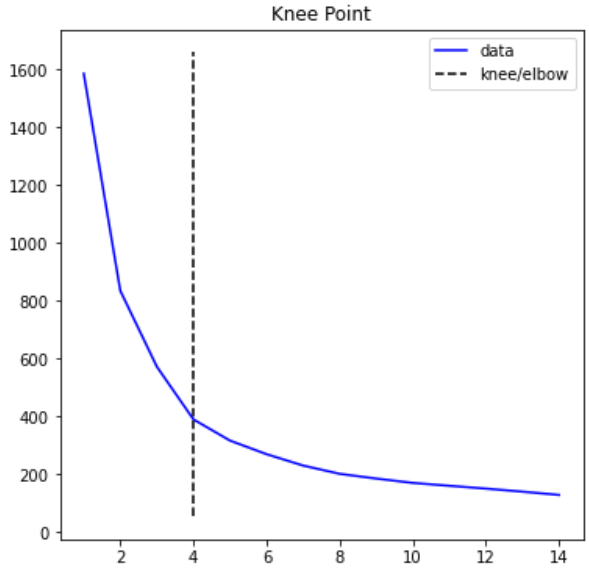
Though, this recasting on only two principal components does not look like an optimum foundation for predictions. That is why we decided to use unsupervised learning as our next step in order to find the optimum number of clusters following our PCA.

2. Method K-Means

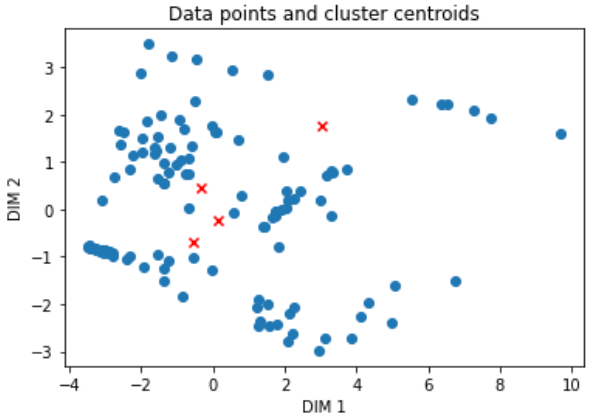
Elbow Method:

Minimization of the SSE (Sum Squared Error):

4 Clusters (SSE: 390)



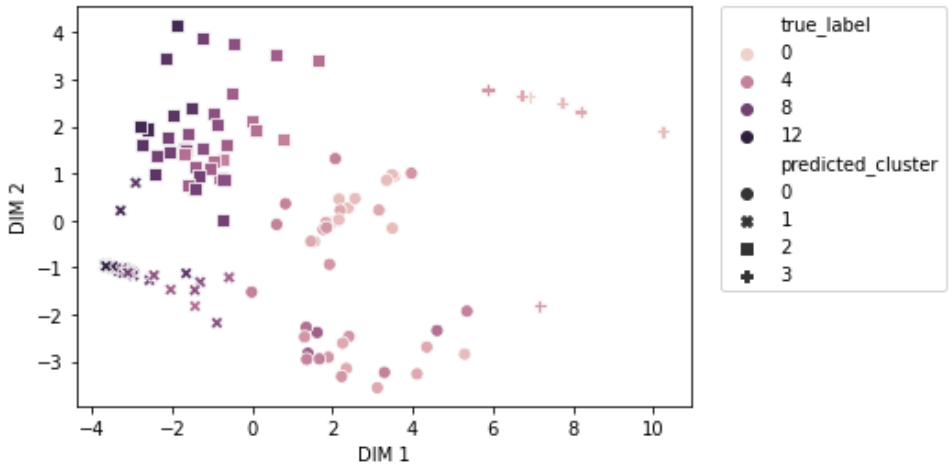
Centroids of the 4 Clusters:

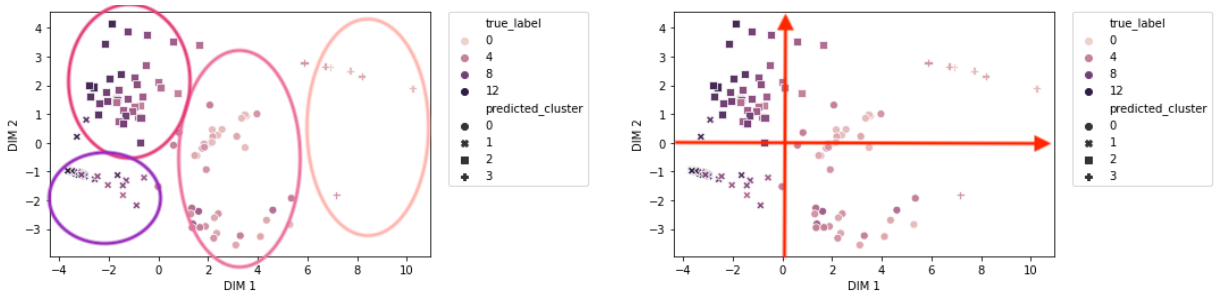


Now that we have our K and our centroids, let's plot the 4 clusters with a scatter plot.

Scatter Plot of the clusters:

Plot of the 4 clusters regarding the Case of flush





The 4 clusters made by our model are plotted through forms whereas the colours are the level of case of flush of the tests of our sample (the darker the higher):

Cluster 0 (o): located on the middle of the plot and encompass the tests with low case of flush (light colour)

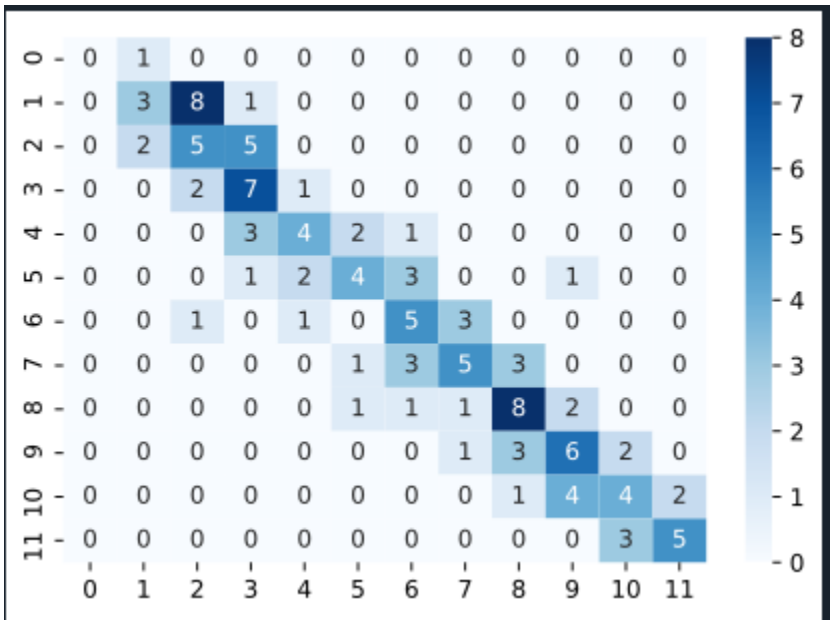
Cluster 1 (x): located on the bottom left of the plot and encompass the tests with a very high case of flush (darkest colour)

Cluster 2 (■): located on the upper left of the plot and encompass the tests with a high case of flush (dark colour)

Cluster 3 (+): located on the upper right of the plot and encompass the tests with a very low case of flush (lightest colour)
As we used our PCA to create the K-Means model, our 4 clusters are consistent with the interpretations we have made from the matrix earlier.

3. Method OLS

Since we found that the dependent variable Y is different cases, we can treat it like qualitative data (although it is discrete variables) to use Chi-square test to compare the difference between the predicted result and the true flush volume result in excel like below. Chi square test can be used to compare the coincidence between the theoretical frequency and the actual frequency or to test the fitting optimization. We suppose the null hypothesis is that the predicted value and true value is independent, when the P value is small, we reject null hypothesis, thus the smaller the P value, the better our model fitting the dataset. When we find the lowest P-value of Chi-square test, we find the most accurate parameters and power to make the predicted value closer to the true value of flush volume.
Finally, we draw the heatmap to analyse the fitting result like this.



An example of the overlapping picture of the predicted flush volume and true flush volume

At the beginning we used OLS to try different kind of equation. We assume that:

1. Y=aX1^(½) +bX2^(½)+cX3^(½)… +lX12^(½)
2. Y=aX1+bX2+cX3+dX4+eX5.... +lX12
3. $Y=aX1^2+bX2^2+cX3^2... +lX12^2$
4. Y= a*log(X1+1)+b*log(X2+1)+c*log(X3+1)+d*log(X4+1)+e*log(X5+1).... +l*log(X12+1)

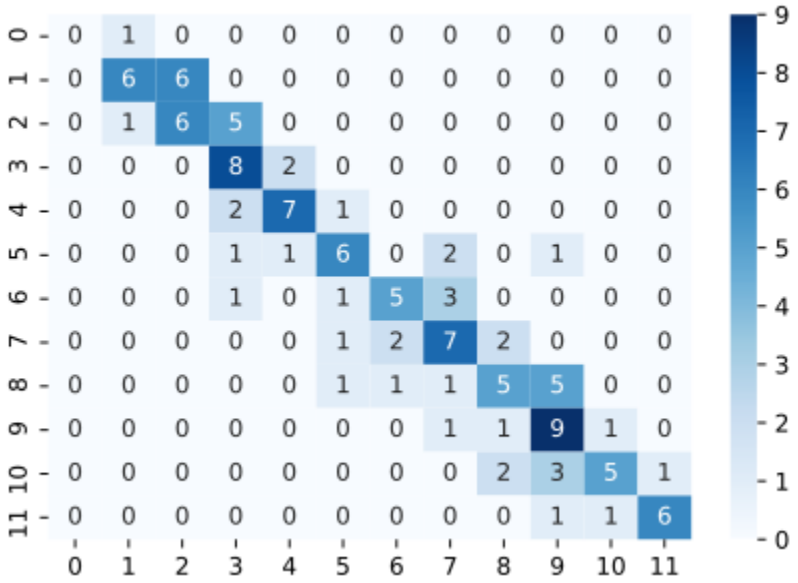
Assumption	R-square	P-value
^(½)	0.895	4.203382217526121e-25
^1	0.871	4.62526748964767e-22
^2	0.813	2.549091996368603e-13
log	0.870	5.629189891557796e-21

We used OLS to get the 12 parameters in each assumption, we got the R-square. Then we compared the predicted value and true value using Chi-square test to get the result of P-value. We found the p-value in the first and second assumption are smaller, at the same time the R-square in first and second assumptions are bigger, and the more the power is close to 0 the R-square and P-value is better. So, we predicted that the power should be between 0 to 1.
After predicting that, we randomly set a small p-value like e^-30 and loop the code on python to find a p-value as small as possible, finally we found a very small p-value 3.326956159301724e-42 after looping 15 minutes, and the R-square is 0.904. This means 90.4% of variables are expressed by this model and the predicted flush volume and true flush volume are similar.

Parameter	Independent variable	Power
0.0137	Blue LED 1 Photodiode 1	0.6049757
0.0044	Blue LED 1 Photodiode 2	0.80858354
-0.2363	Blue LED 2 Photodiode 1	0.257002
-0072	Blue LED 2 Photodiode 2	0.63890724
0.0061	Green LED 1 Photodiode 1	0.5909667
-0.0909	Green LED 1 Photodiode 2	0.59050965
-0.0542	Green LED 2 Photodiode 1	0.61059362
-0.1216	Green LED 2 Photodiode 2	0.10776922
0.0840	Red LED 1 Photodiode 1	0.33897334
0.2966	Red LED 1 Photodiode 2	0.31753335
0.0058	Red LED 2 Photodiode 1	0.67646697
-0.1574	Red LED 2 Photodiode 2	0.49213317

Finally, the formula is:

Y= 0.0137*X1^ 0.6049757+0.0044*X2* 0.80858354-0.2363*X3^ 0.257002 -0072*X4^ 0.63890724+0.0061*X5^ 0.5909667-0.0909*X6^ 0.59050965 -0.0542*X7* 0.61059362 -0.1216*X8^ 0.10776922 + 0.0840*X9^ 0.33897334 +0.2966*X10^ 0.31753335 +0.0058*X11^ 0.67646697 -0.1574*X12^ 0.49213317



Overlapping picture of predicted flush volume and true flush volume in this model

These predictions on diagonal are accurate, the further the prediction to the diagonal, the lower the accuracy is. And the numbers in the picture shows how many right predictions we have. The picture shows we have a good result. While the accuracy rate we calculated for this model is 57.37%, so we try to make another model with higher accuracy rate.

4. Method GLM

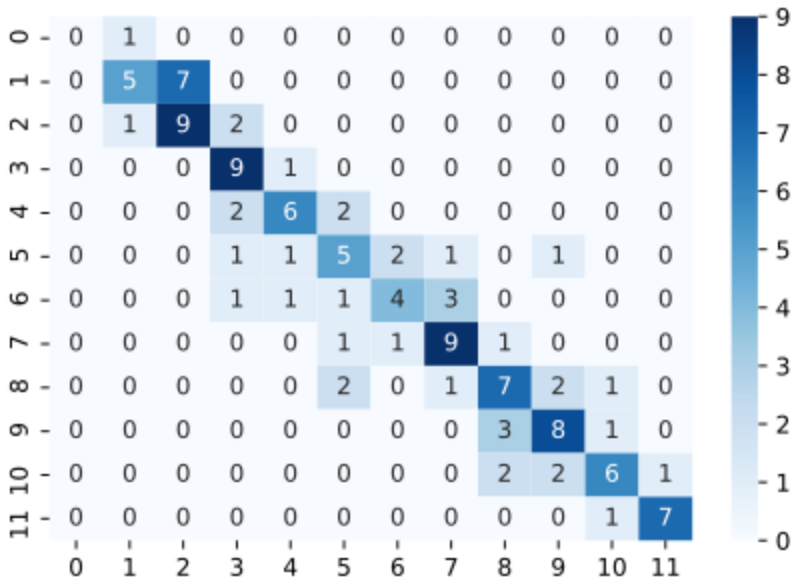
In this part, we tried Gaussian, Poisson, Gamma and Negative Binomial model in exponential family. In these 4 models, using loop function in python we get the highest accuracy score in Gaussian model, which is 61.475%, and P value is 1.0278368867117633e-48, which means the predicted flush volume are much more similar than before with the true value. The relatively parameters and power are as below.

Parameter	Independent variable	Power
0.2307	F (Blue LED 1 Photodiode 1)	0.17182156
-0.0552	F (Blue LED 1 Photodiode 2)	0.42594121
0.0037	F (Blue LED 2 Photodiode 1)	0.8522537
0.1546	F (Blue LED 2 Photodiode 2)	0.513989
0.2494	F (Green LED 1 Photodiode 1)	0.53666289
0.9801	F (Green LED 1 Photodiode 2)	0.17705916
-0.4866	F (Green LED 2 Photodiode 1)	0.655312
-0.8017	F (Green LED 2 Photodiode 2)	0.12910251
-0.3429	F (Red LED 1 Photodiode 1)	0.42398372
-0.2425	F (Red LED 1 Photodiode 2)	0.34662343
0.0280	F (Red LED 2 Photodiode 1)	0.94068168
-0.0553	F (Red LED 2 Photodiode 2)	0.81186728

Note: F(x) is the function of Gaussian model.

The formula is as below:

Y=0.2307*f(x1)^0.17182156-0.0552*f(x2)^ 0.42594121+ 0.0037*f(x3)^ 0.8522537+ 0.1546*f(x4)^ 0.513989+ 0.2494*f(x5)^ 0.53666289+ 0.9801*f(x6)^ 0.17705916 - 0.4866*f(x7)^ 0.655312-0.8017*f(x8)^ 0.12910251-0.3429*f(x9)^ 0.42398372- 0.2425*f(x10)^ 0.34662343+ 0.0280*f(x11)^ 0.94068168- 0.0553*f(x12)^ 0.81186728



We get the over lapping picture of predicted value and true value through Chi-square test like in OLS. After comparing the overlapping pictures with OLS's, we can find that the figures on diagonal are higher and better than before, and the figures beside the diagonal are smaller, which means the wrong prediction decreased. The more the figures are close to the diagonal, the bigger they are.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	1
1	0.71	0.42	0.53	12
2	0.56	0.75	0.64	12
3	0.60	0.90	0.72	10
4	0.67	0.60	0.63	10
5	0.45	0.45	0.45	11
6	0.57	0.40	0.47	10
7	0.64	0.75	0.69	12
8	0.54	0.54	0.54	13
9	0.62	0.67	0.64	12
10	0.67	0.55	0.60	11
11	0.88	0.88	0.88	8
accuracy			0.61	122

Recall means the number of good predictions for each case. For instance, for case of flush 3, we predicted well 9 times out of 10. So, recall for case of flush 3 is 9/10 = 0.90. Basically, recall allow to see how well we predicted a certain case of flush.

Precision means the number of times we predicted for this class. For instance, for case of flush 3, we predicted well 9 times. But overall, we predicted a case of flush of 3 15 times. So the precision for case of flush 3 is 9/15 =0.60

Basically, precision allow to see how MANY times we predicted a certain class (too many or not enough times) and f1-score is just a balanced of the two metrics. So high recall but low precision means we predict well a certain class but too much (maybe bias) and low recall, but high prevision means we predict not enough of this class.
Since case of flush 0, which is a special case, only show up for one time, we got accuracy of 0 in this case. This significantly deduced the overall accuracy rates. If we put this case out of the model, the accuracy rate will be greatly improved.

III/ Discussion

In this discussion, we will highlight two different part in which we found difficulties and need to be cautious about to improve our predictions.

The first point is linked to the size of our dataset. Indeed, our dataset is a sample composed of 122 observations. Two concerns arise from this fact:

- Trade-off in terms of train/test split: since our dataset contains 122 observations, if we put 20% as our test size, only 25 predictions will be made. It creates a trade-off between the accuracy of our predictions versus the quantity of predictions made
- Only one test where the outcome of Case of flush is 0: very difficult for our model to predict this outcome (at least 10 tests of the same outcome is needed to make accurate predictions)

All in all, we think that testing our model on a dataset which as more iteration of observations with a case of flush of 0 will allow us to reduce these points of concerns. It'll also increase the accuracy of our model.

The second point of concern is linked to our predictions. Indeed, we do not have a perfect model which always predict the perfect value. In this case, when we do not make the correct prediction, we will always prefer to predict the closest value of the true one.
For instance, for Case of flush 0, we have an accuracy of 0%, but we predicted 1, it is the closest value to 0.

As the principal need of this case is to clean efficiently the toilet, all predictions which put way too much, or not enough water must be avoided and/or monitored. A special attention must be putted when our model is predicting such error. That is why we need to be cautious when the predicted value is distant of more than 1 unity of Case of flush. For instance, for Case of flush 8, two predictions are 5. These predictions are too distant from the true value, an improved model will need to work on this case.

Finally, even if our model does not predict all true values each try, we still think that it's a good base for improvement. A larger dataset and special attention to large prediction errors will increase the quality of our predictions.

Python Code

Description code

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import ExtraTreesClassifier
import scipy.stats as stats

data = pd.read_excel('/Users/dorian/Desktop/Data Science/Group 7/File 7 - Group7 - Dorian.xlsx')
ml_data=data[['Blue LED 1 Photodiode 1','Blue LED 1 Photodiode 2','Green LED 1 Photodiode 1','Green LED 1 Photodiode 2','Red LED 1 Photodiode 1','Red LED 1 Photodiode 2','Blue LED 2 Photodiode 1','Blue LED 2 Photodiode 2','Green LED 2 Photodiode 1','Green LED 2 Photodiode 2','Red LED 2 Photodiode 1','Red LED 2 Photodiode 2','Case of flush']]
features=ml_data[['Blue LED 1 Photodiode 1','Blue LED 1 Photodiode 2','Green LED 1 Photodiode 1','Green LED 1 Photodiode 2','Red LED 1 Photodiode 1','Red LED 1 Photodiode 2','Blue LED 2 Photodiode 1','Blue LED 2 Photodiode 2','Green LED 2 Photodiode 1','Green LED 2 Photodiode 2','Red LED 2 Photodiode 1','Red LED 2 Photodiode 2']]
target=ml_data['Case of flush']

#Analysis of Case of flush

cof0=ml_data[ml_data['Case of flush']==0]
f0=cof0[['Blue LED 1 Photodiode 1','Blue LED 1 Photodiode 2','Green LED 1 Photodiode 1','Green LED 1 Photodiode 2','Red LED 1 Photodiode 1','Red LED 1 Photodiode 2','Blue LED 2 Photodiode 1','Blue LED 2 Photodiode 2','Green LED 2 Photodiode 1','Green LED 2 Photodiode 2','Red LED 2 Photodiode 1','Red LED 2 Photodiode 2']]
sns.distplot(f0)
plt.show()

cof1=ml_data[ml_data['Case of flush']==1]
f1=cof1[['Blue LED 1 Photodiode 1','Blue LED 1 Photodiode 2','Green LED 1 Photodiode 1','Green LED 1 Photodiode 2','Red LED 1 Photodiode 1','Red LED 1 Photodiode 2','Blue LED 2 Photodiode 1','Blue LED 2 Photodiode 2','Green LED 2 Photodiode 1','Green LED 2 Photodiode 2','Red LED 2 Photodiode 1','Red LED 2 Photodiode 2']]
sns.distplot(f1)
plt.show()

cof2=ml_data[ml_data['Case of flush']==2]
f2=cof2[['Blue LED 1 Photodiode 1','Blue LED 1 Photodiode 2','Green LED 1 Photodiode 1','Green LED 1 Photodiode 2','Red LED 1 Photodiode 1','Red LED 1 Photodiode 2','Blue LED 2 Photodiode 1','Blue LED 2 Photodiode 2','Green LED 2 Photodiode 1','Green LED 2 Photodiode 2','Red LED 2 Photodiode 1','Red LED 2 Photodiode 2']]
sns.distplot(f2)
plt.show()

cof3=ml_data[ml_data['Case of flush']==3]
f3=cof3[['Blue LED 1 Photodiode 1','Blue LED 1 Photodiode 2','Green LED 1 Photodiode 1','Green LED 1 Photodiode 2','Red LED 1 Photodiode 1','Red LED 1 Photodiode 2','Blue LED 2 Photodiode 1','Blue LED 2 Photodiode 2','Green LED 2 Photodiode 1','Green LED 2 Photodiode 2','Red LED 2 Photodiode 1','Red LED 2 Photodiode 2']]
sns.distplot(f3)
plt.show()

cof4=ml_data[ml_data['Case of flush']==4]
f4=cof4[['Blue LED 1 Photodiode 1','Blue LED 1 Photodiode 2','Green LED 1 Photodiode 1','Green LED 1 Photodiode 2','Red LED 1 Photodiode 1','Red LED 1 Photodiode 2','Blue LED 2 Photodiode 1','Blue LED 2 Photodiode 2','Green LED 2 Photodiode 1','Green LED 2 Photodiode 2','Red LED 2 Photodiode 1','Red LED 2 Photodiode 2']]
sns.distplot(f4)
```

```
plt.show()

cof5=ml_data[ml_data['Case of flush']==5]
f5=cof5[['Blue LED 1 Photodiode 1','Blue LED 1 Photodiode 2','Green LED 1 Photodiode 1','Green LED 1 Photodiode 2','Red LED 1 Photodiode 1','Red LED 1 Photodiode 2','Blue LED 2 Photodiode 1','Blue LED 2 Photodiode 2','Green LED 2 Photodiode 1','Green LED 2 Photodiode 2','Red LED 2 Photodiode 1','Red LED 2 Photodiode 2']]
sns.distplot(f5)
plt.show()

cof6=ml_data[ml_data['Case of flush']==6]
f6=cof6[['Blue LED 1 Photodiode 1','Blue LED 1 Photodiode 2','Green LED 1 Photodiode 1','Green LED 1 Photodiode 2','Red LED 1 Photodiode 1','Red LED 1 Photodiode 2','Blue LED 2 Photodiode 1','Blue LED 2 Photodiode 2','Green LED 2 Photodiode 1','Green LED 2 Photodiode 2','Red LED 2 Photodiode 1','Red LED 2 Photodiode 2']]
sns.distplot(f6)
plt.show()

cof7=ml_data[ml_data['Case of flush']==7]
f7=cof7[['Blue LED 1 Photodiode 1','Blue LED 1 Photodiode 2','Green LED 1 Photodiode 1','Green LED 1 Photodiode 2','Red LED 1 Photodiode 1','Red LED 1 Photodiode 2','Blue LED 2 Photodiode 1','Blue LED 2 Photodiode 2','Green LED 2 Photodiode 1','Green LED 2 Photodiode 2','Red LED 2 Photodiode 1','Red LED 2 Photodiode 2']]
sns.distplot(f7)
plt.show()

cof8=ml_data[ml_data['Case of flush']==8]
f8=cof8[['Blue LED 1 Photodiode 1','Blue LED 1 Photodiode 2','Green LED 1 Photodiode 1','Green LED 1 Photodiode 2','Red LED 1 Photodiode 1','Red LED 1 Photodiode 2','Blue LED 2 Photodiode 1','Blue LED 2 Photodiode 2','Green LED 2 Photodiode 1','Green LED 2 Photodiode 2','Red LED 2 Photodiode 1','Red LED 2 Photodiode 2']]
sns.distplot(f8)
plt.show()

cof9=ml_data[ml_data['Case of flush']==9]
f9=cof9[['Blue LED 1 Photodiode 1','Blue LED 1 Photodiode 2','Green LED 1 Photodiode 1','Green LED 1 Photodiode 2','Red LED 1 Photodiode 1','Red LED 1 Photodiode 2','Blue LED 2 Photodiode 1','Blue LED 2 Photodiode 2','Green LED 2 Photodiode 1','Green LED 2 Photodiode 2','Red LED 2 Photodiode 1','Red LED 2 Photodiode 2']]
sns.distplot(f9)
plt.show()

cof10=ml_data[ml_data['Case of flush']==10]
f10=cof10[['Blue LED 1 Photodiode 1','Blue LED 1 Photodiode 2','Green LED 1 Photodiode 1','Green LED 1 Photodiode 2','Red LED 1 Photodiode 1','Red LED 1 Photodiode 2','Blue LED 2 Photodiode 1','Blue LED 2 Photodiode 2','Green LED 2 Photodiode 1','Green LED 2 Photodiode 2','Red LED 2 Photodiode 1','Red LED 2 Photodiode 2']]
sns.distplot(f10)
plt.show()

cof11=ml_data[ml_data['Case of flush']==11]
f11=cof11[['Blue LED 1 Photodiode 1','Blue LED 1 Photodiode 2','Green LED 1 Photodiode 1','Green LED 1 Photodiode 2','Red LED 1 Photodiode 1','Red LED 1 Photodiode 2','Blue LED 2 Photodiode 1','Blue LED 2 Photodiode 2','Green LED 2 Photodiode 1','Green LED 2 Photodiode 2','Red LED 2 Photodiode 1','Red LED 2 Photodiode 2']]
sns.distplot(f1)
plt.show()

#Sum of all values by color

#Blue

blue=ml_data[['Blue LED 1 Photodiode 1','Blue LED 1 Photodiode 2','Blue LED 2 Photodiode 1','Blue LED 2 Photodiode 2']]
sns.distplot(blue,color='b')
plt.show()
print(stats.shapiro(blue))

```

```
blue['agg']=blue[list(blue.columns)].sum(axis=1)
blue_agg=blue['agg']
ml_data['Blue']=blue_agg

print(ml_data['Blue'].describe())

mean_blue=ml_data.groupby('Case of flush')['Blue'].mean()
median_blue=ml_data.groupby('Case of flush')['Blue'].median()
min_blue=ml_data.groupby('Case of flush')['Blue'].min()
max_blue=ml_data.groupby('Case of flush')['Blue'].max()
print(mean_blue)
print(median_blue)
print(min_blue)
print(max_blue)

fig, ax = plt.subplots()
sns.regplot(target,ml_data['Blue'],color='b')
ax.set_xticks(range(0,11))
plt.show()

sns.distplot(ml_data['Blue'],color='b')
plt.show()

#Red

red=ml_data[['Red LED 1 Photodiode 1','Red LED 1 Photodiode 2','Red LED 2 Photodiode 1','Red LED 2 Photodiode 2']]
sns.distplot(red,color='r')
plt.show()
print(stats.shapiro(red))

red['agg']=red[list(red.columns)].sum(axis=1)
red_agg=red['agg']
ml_data['Red']=red_agg

print(ml_data['Red'].describe())

mean_red=ml_data.groupby('Case of flush')['Red'].mean()
median_red=ml_data.groupby('Case of flush')['Red'].median()
min_red=ml_data.groupby('Case of flush')['Red'].min()
max_red=ml_data.groupby('Case of flush')['Red'].max()
print(mean_red)
print(median_red)
print(min_red)
print(max_red)

fig, ax = plt.subplots()
sns.regplot(target,ml_data['Red'],color='red')
ax.set_xticks(range(0,11))
plt.show()

sns.distplot(ml_data['Red'],color='r')
plt.show()

#Green

green=ml_data[['Green LED 1 Photodiode 1','Green LED 1 Photodiode 2','Green LED 2 Photodiode 1','Green LED 2 Photodiode 2']]
sns.distplot(green,color='g')
plt.show()
print(stats.shapiro(green))
```

```
143.
144. green['agg']=green[list(green.columns)].sum(axis=1)
145. green_agg=green['agg']
146. ml_data['Green']=green_agg
147.
148. lognormal=np.random.lognormal(size=122)
149. sns.distplot(lognormal)
150. plt.show()
151.
152. print(ml_data['Green'].describe())
153.
154. mean_green=ml_data.groupby('Case of flush')['Green'].mean()
155. median_green=ml_data.groupby('Case of flush')['Green'].median()
156. min_green=ml_data.groupby('Case of flush')['Green'].min()
157. max_green=ml_data.groupby('Case of flush')['Green'].max()
158. print(mean_green)
159. print(median_green)
160. print(min_green)
161. print(max_green)
162.
163. fig, ax = plt.subplots()
164. sns.regplot(target,ml_data['Green'],color='green')
165. ax.set_xticks(range(0,11))
166. plt.show()
167.
168. sns.distplot(ml_data['Green'],color='g')
169. plt.show()
170.
171. #Displot of the sum of colors
172.
173. ml_data['sum']=ml_data.apply(lambda row: row.Blue + row.Green + row.Red, axis=1)
174. print(ml_data['sum'].describe())
175. sns.distplot(ml_data['sum'])
176.
177. #Relation plot color - Case of flush
178.
179. fig, ax = plt.subplots()
180. sns.regplot(target,ml_data['sum'])
181. ax.set_xticks(range(0,11))
182. plt.show()
183.
184. #Correlation
185.
186. ml=ml_data[["Case of flush", "sum"]].dropna()
187. corrMatrix = ml.corr()
188. print(corrMatrix)
189. print(stats.pearsonr(ml['Case of flush'], ml['sum']))
190. #We reject H0 -> strong negative correlation
191.
192. print(stats.pearsonr(ml_data['Case of flush'], ml_data['Red']))
193. #We reject H0 -> strong negative correlation
194. print(stats.pearsonr(ml_data['Case of flush'], ml_data['Blue']))
195. #We reject H0 -> strong negative correlation
196. print(stats.pearsonr(ml_data['Case of flush'], ml_data['Green']))
197. #We reject H0 -> strong negative correlation
198.
199. #Number of trials per case of flush
200.
201. count=ml_data['Case of flush'].value_counts()
202. print(count)
203. sns.countplot(data=ml_data,x='Case of flush',order=ml_data['Case of flush'].value_counts().index)
```

```
plt.show()

#Description group by Case of flush

mean_cof=ml_data.groupby('Case of flush')['sum'].mean()
median_cof=ml_data.groupby('Case of flush')['sum'].median()
min  cof=ml  data.groupby('Case of flush')['sum'].min()
max_cof=ml_data.groupby('Case of flush')['sum'].max()
print(mean_cof)
print(median_cof)
print(min  cof)
print(max_cof)


#Importance of Feature

color=ml_data[['Blue','Red','Green']]
X=color
y=target
model = ExtraTreesClassifier()
model.fit(X,y)
print(model.feature_importances_)
feat  importances = pd.Series(model.feature  importances  , index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.show()
```

PCA & K-means Code

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import classification_report
from sklearn.cluster import KMeans
from kneed import KneeLocator
from mlxtend.plotting import plot pca correlation graph
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from statsmodels.stats.outliers influence import variance inflation factor
import statsmodels.api as sm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score



data = pd.read_excel('/Users/dorian/Desktop/Data Science/Group 7/File 7 - Group7 - Dorian.xlsx')
ml data=data[['Blue LED 1 Photodiode 1','Blue LED 1 Photodiode 2','Green LED 1 Photodiode 1','Green LED 1 Photo
diode 2','Red LED 1 Photodiode 1','Red LED 1 Photodiode 2','Blue LED 2 Photodiode 1','Blue LED 2 Photodiode 2','G
reen LED 2 Photodiode 1','Green LED 2 Photodiode 2','Red LED 2 Photodiode 1','Red LED 2 Photodiode 2','Case of fl
ush']]
features=ml data[['Blue LED 1 Photodiode 1','Blue LED 1 Photodiode 2','Green LED 1 Photodiode 1','Green LED 1 P
hotodiode 2','Red LED 1 Photodiode 1','Red LED 1 Photodiode 2','Blue LED 2 Photodiode 1','Blue LED 2 Photodiode
2','Green LED 2 Photodiode 1','Green LED 2 Photodiode 2','Red LED 2 Photodiode 1','Red LED 2 Photodiode 2']]
target=ml_data['Case of flush']


numeric_df=features.apply(lambda x: np.log(x+1) if np.issubdtype(x.dtype,np.number)else x)
print(np.log(ml_data['Blue LED 1 Photodiode 1']))
#PCA

#Standardize features

x norm = StandardScaler().fit transform(ml data)
np.mean(x norm),np.std(x norm)
feat_cols = ['feature'+str(i) for i in range(x_norm.shape[1])]
normalised_features = pd.DataFrame(x_norm,columns=feat_cols)


#Creation of the Components

n components=12
pca cof = PCA(n components)
principalComponents_cof = pca_cof.fit_transform(x_norm)
print('Explained variation per principal component: {}'.format(pca_cof.explained_variance_ratio_))

#Optimal number of components: 4 (95% of dataset explained)



reduced = principalComponents cof

```

```
#Append the principle components for each entry to the dataframe

for i in range(0, n  components):
    ml_data['PC' + str(i + 1)] = reduced[:, i]

print(ml_data.head())

#Scree plot

ind = np.arange(0, n_components)
(fig, ax) = plt.subplots(figsize=(8, 6))
sns.pointplot(x=ind, y=pca_cof.explained_variance_ratio_)
ax.set_title('Scree plot')
ax.set_xticks(ind)
ax.set  xticklabels(ind)
ax.set_xlabel('Component Number')
ax.set_ylabel('Explained Variance')
plt.show()

#Show the Case of Flush in terms of the first two PCs

g = sns.lmplot('PC1','PC2',hue='Case of flush',data=ml_data,fit_reg=False,scatter=True)
plt.show()

#Correlation Circle
features_name=['Blue 1-1','Blue 1-2','Green 1-1','Green 1-2','Red 1-1','Red 1-2','Blue 2-1','Blue 2-2','Green 2-
1','Green 2-2','Red 2-1','Red 2-2','Case of flush']
fig, correlation_matrix=plot_pca_correlation_graph(x_norm,features_name,dimensions=(1,2,3,4))
plt.show()


#K-means

#Elbow Method - SSE = Sum Squared Error

sse=[]
for k in range(1,15):
    kmeans=KMeans(n_clusters=k,random_state=42)
    kmeans.fit(x  norm)
    sse.append(kmeans.inertia_)

kl=KneeLocator(range(1,15),sse,curve='convex',direction='decreasing')
nbr  cluster=kl.elbow
KneeLocator.plot_knee(kl)
plt.show()
print(nbr_cluster)

#Print the minimal SSE

km=KMeans(n_clusters=4)
km.fit(x  norm)
print(km.inertia_)

# Plotting the cluster centers and the data points on a 2D plane

plt.scatter(ml_data['PC1'],ml_data['PC2'])
plt.scatter(km.cluster_centers_[:, 0], km.cluster_centers_[:, 1], c='red', marker='x')
plt.xlabel('DIM 1')
plt.ylabel('DIM 2')
plt.title('Data points and cluster centroids')
plt.show()
```

```
112.
113. #Creation of Pipelines
114.
115. preprocessor=Pipeline([("scaler",StandardScaler()),("pca",PCA(n_components=4,random_state=42))])
116. clusterer=Pipeline([("kmeans",KMeans(n_clusters=nbr_cluster,random_state=42))])
117. pipe=Pipeline([("preprocessor",preprocessor),("clusterer",clusterer)])
118. pipe.fit(ml_data)
119.
120. pcadf=pd.DataFrame(pipe['preprocessor'].transform(ml_data),columns=['DIM 1','DIM 2','DIM 3','DIM 4'])
121. pcadf["predicted_cluster"]=pipe['clusterer']['kmeans'].labels_
122. pcadf['true_label']=ml_data['Case of flush']
123.
124. #Scatter Plot of the clusters
125.
126. scat=sns.scatterplot("DIM 1","DIM 2",data=pcadf,hue='true_label',style='predicted_cluster',s=50)
127. plt.legend(bbox_to_anchor=(1.05,1),loc=2,borderaxespad=0.0)
128. plt.show()
129.
130. #Predictions
131.
132. #GLM - All Dataset
133.
134. X0=ml_data[['PC1','PC2','PC3','PC4','PC5','PC6','PC7','PC8','PC9','PC10','PC11','PC12']]
135. X0 = sm.add_constant(X0)
136.
137. Y=ml_data['Case of flush']
138.
139. model0 = sm.GLM(Y, X0)
140. results0 = model0.fit()
141. print(results0.summary())
142.
143. ypred0=results0.predict(X0)
144. print(ypred0)
145. print(ypred0.round())
146. predicted_class=ypred0.round()
147.
148. ml_data["Bpred"]=predicted_class
149. Z=ml_data["Bpred"]
150. results = confusion_matrix(Y,Z)
151.
152. print(results)
153. print(sns.heatmap(results, annot=True,  fmt='', cmap='Blues'))
154. print(classification_report(Y, Z))
155.
156. #Logistic - Part of the Dataset
157.
158. X_train, X_test, y_train, y_test = train_test_split(X0, Y, test_size = 0.4,random_state=42)
159. model = LogisticRegression(multi_class='multinomial', penalty='none').fit(X_train, y_train)
160. preds = model.predict(X_test)
161. print(preds)
162.
163. print('Accuracy Score:', accuracy_score(y_test, preds))
164. print(classification_report(y_test, preds))
165.
166.
167.
168. #VIF
169.
170. vif_data=pd.DataFrame()
171. vif_data['feature']=pcadf.columns
172.
```

```
vif_data['VIF']=[variance_inflation_factor(pcadf.values,i)
                 for i in range(len(pcadf.columns))]
print(vif  data)

#Correlation Matrix

corrMatrix=normalised  features.corr()
print(corrMatrix)
```

Initial Assumptions

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
from sklearn.metrics import confusion  matrix
import scipy.stats as stats
import seaborn as sns
from sklearn.metrics import accuracy_score

toilet_data = pd.read_excel("File 7 - Group7.xlsx", header = 0, sep = " ")
toilet=toilet_data

print(toilet.columns)

print(toilet["Flush volume"].value_counts())









#4.203382217526121e-25   Assumption 1
# ============================================================================
toilet["Blue11"]=np.sqrt(toilet['Blue LED 1\nPhotodiode 1'])
toilet["Blue12"]=np.sqrt(toilet['Blue LED 1\nPhotodiode 2'])
toilet["Blue21"]=np.sqrt(toilet['Blue LED 2\nPhotodiode 1'])
toilet["Blue22"]=np.sqrt(toilet['Blue LED 2\nPhotodiode 2'])
toilet["Green11"]=np.sqrt(toilet['Green LED 1\nPhotodiode 1'])
toilet["Green12"]=np.sqrt(toilet['Green LED 1\nPhotodiode 2'])
toilet["Green21"]=np.sqrt(toilet['Green LED 2\nPhotodiode 1'])
toilet["Green22"]=np.sqrt(toilet['Green LED 2\nPhotodiode 2'])
toilet["Red11"]=np.sqrt(toilet['Red LED 1\nPhotodiode 1'])
toilet["Red12"]=np.sqrt(toilet['Red LED 1\nPhotodiode 2'])
toilet["Red21"]=np.sqrt(toilet['Red LED 2\nPhotodiode 1'])
toilet["Red22"]=np.sqrt(toilet['Red LED 2\nPhotodiode 2'])
X=toilet[["Blue11", "Blue12", "Blue21", "Blue22", "Green11", "Green12", "Green21", "Green22", "Red11", "Red12",
"Red21", "Red22"]]
X = sm.add_constant(X)
Y=toilet[["Flush volume"]]
model = sm.OLS(Y, X)
results = model.fit()
print(results.summary())
bins = [-100,0.75,1.7,2.15,2.6,3.05,3.55,4,4.45,4.95,5.4,5.85,100]

toilet.loc[:,'FlushV  int']=pd.cut(toilet['Flush volume'],bins,
                                   labels=[0,1,2,3,4,5,6,7,8,9,10,11])
Y2=toilet.loc[:,'FlushV_int']
ypred=results.predict(X)
toilet.loc[:,"Ypred"]=ypred

toilet.loc[:,'Flushpr_int']=pd.cut(ypred,bins,labels=[0,1,2,3,4,5,6,7,8,9,10,11])
X2=toilet['Flushpr_int']
results = confusion  matrix(Y2,X2)
y_pred = pd.Series(toilet['Flushpr_int'])
y_true = pd.Series(toilet["Flush volume"])
fchi=pd.crosstab(y_true, y_pred, rownames=['True'], colnames=['Predicted'], margins=True)
fchi=stats.chi2  contingency(fchi)
print(fchi)
# ============================================================================
```

```
58.
59.  #5.629189891557796e-21  Assumption 2
60.  # ==========================================================================
61.  toilet["Blue11"]=np.log(toilet['Blue LED 1\nPhotodiode 1']+1)
62.  toilet["Blue12"]=np.log(toilet['Blue LED 1\nPhotodiode 2']+1)
63.  toilet["Blue21"]=np.log(toilet['Blue LED 2\nPhotodiode 1']+1)
64.  toilet["Blue22"]=np.log(toilet['Blue LED 2\nPhotodiode 2']+1)
65.  toilet["Green11"]=np.log(toilet['Green LED 1\nPhotodiode 1']+1)
66.  toilet["Green12"]=np.log(toilet['Green LED 1\nPhotodiode 2']+1)
67.  toilet["Green21"]=np.log(toilet['Green LED 2\nPhotodiode 1']+1)
68.  toilet["Green22"]=np.log(toilet['Green LED 2\nPhotodiode 2']+1)
69.  toilet["Red11"]=np.log(toilet['Red LED 1\nPhotodiode 1']+1)
70.  toilet["Red12"]=np.log(toilet['Red LED 1\nPhotodiode 2']+1)
71.  toilet["Red21"]=np.log(toilet['Red LED 2\nPhotodiode 1']+1)
72.  toilet["Red22"]=np.log(toilet['Red LED 2\nPhotodiode 2']+1)
73.  X=toilet[["Blue11", "Blue12", "Blue21", "Blue22", "Green11", "Green12", "Green21", "Green22", "Red11", "Red12",
     "Red21", "Red22"]]
74.  X = sm.add_constant(X)
75.  Y=toilet[["Flush volume"]]
76.  model = sm.OLS(Y, X)
77.  results = model.fit()
78.  print(results.summary())
79.  bins = [-100,0.75,1.7,2.15,2.6,3.05,3.55,4,4.45,4.95,5.4,5.85,100]
80.
81.  toilet.loc[:,'FlushV_int']=pd.cut(toilet['Flush volume'],bins,
82.                                   labels=[0,1,2,3,4,5,6,7,8,9,10,11])
83.  Y2=toilet.loc[:,'FlushV  int']
84.  ypred=results.predict(X)
85.  toilet.loc[:,"Ypred"]=ypred
86.
87.  toilet.loc[:,'Flushpr  int']=pd.cut(ypred,bins,labels=[0,1,2,3,4,5,6,7,8,9,10,11])
88.  X2=toilet['Flushpr_int']
89.  results = confusion_matrix(Y2,X2)
90.
91.  y  pred = pd.Series(toilet['Flushpr  int'])
92.  y_true = pd.Series(toilet["Flush volume"])
93.  fchi=pd.crosstab(y_true, y_pred, rownames=['True'], colnames=['Predicted'], margins=True)
94.  fchi=stats.chi2_contingency(fchi)
95.  print(fchi)
96.  # ==========================================================================
97.
98.  #4.62526748964767e-22   Assumption 3
99.  # ==========================================================================
100. toilet["Blue11"]=toilet['Blue LED 1\nPhotodiode 1']
101. toilet["Blue12"]=toilet['Blue LED 1\nPhotodiode 2']
102. toilet["Blue21"]=toilet['Blue LED 2\nPhotodiode 1']
103. toilet["Blue22"]=toilet['Blue LED 2\nPhotodiode 2']
104. toilet["Green11"]=toilet['Green LED 1\nPhotodiode 1']
105. toilet["Green12"]=toilet['Green LED 1\nPhotodiode 2']
106. toilet["Green21"]=toilet['Green LED 2\nPhotodiode 1']
107. toilet["Green22"]=toilet['Green LED 2\nPhotodiode 2']
108. toilet["Red11"]=toilet['Red LED 1\nPhotodiode 1']
109. toilet["Red12"]=toilet['Red LED 1\nPhotodiode 2']
110. toilet["Red21"]=toilet['Red LED 2\nPhotodiode 1']
111. toilet["Red22"]=toilet['Red LED 2\nPhotodiode 2']
112. X=toilet[["Blue11", "Blue12", "Blue21", "Blue22", "Green11", "Green12", "Green21", "Green22", "Red11", "Red12",
     "Red21", "Red22"]]
113. X = sm.add_constant(X)
114. Y=toilet[["Flush volume"]]
115. model = sm.OLS(Y, X)
116. results = model.fit()
```

```
print(results.summary())
bins = [-100,0.75,1.7,2.15,2.6,3.05,3.55,4,4.45,4.95,5.4,5.85,100]

toilet.loc[:,'FlushV_int']=pd.cut(toilet['Flush volume'],bins,
                                  labels=[0,1,2,3,4,5,6,7,8,9,10,11])
Y2=toilet.loc[:,'FlushV_int']
ypred=results.predict(X)
toilet.loc[:,"Ypred"]=ypred

toilet.loc[:,'Flushpr_int']=pd.cut(ypred,bins,labels=[0,1,2,3,4,5,6,7,8,9,10,11])
X2=toilet['Flushpr  int']
results = confusion_matrix(Y2,X2)
y_pred = pd.Series(toilet['Flushpr_int'])
y_true = pd.Series(toilet["Flush volume"])
fchi=pd.crosstab(y  true, y  pred, rownames=['True'], colnames=['Predicted'], margins=True)
fchi=stats.chi2_contingency(fchi)
print(fchi)
# ================================================================================

#2.549091996368603e-13    Assumption 4
# ================================================================================
toilet["Blue11"]=toilet['Blue LED 1\nPhotodiode 1']**2
toilet["Blue12"]=toilet['Blue LED 1\nPhotodiode 2']**2
toilet["Blue21"]=toilet['Blue LED 2\nPhotodiode 1']**2
toilet["Blue22"]=toilet['Blue LED 2\nPhotodiode 2']**2
toilet["Green11"]=toilet['Green LED 1\nPhotodiode 1']**2
toilet["Green12"]=toilet['Green LED 1\nPhotodiode 2']**2
toilet["Green21"]=toilet['Green LED 2\nPhotodiode 1']**2
toilet["Green22"]=toilet['Green LED 2\nPhotodiode 2']**2
toilet["Red11"]=toilet['Red LED 1\nPhotodiode 1']**2
toilet["Red12"]=toilet['Red LED 1\nPhotodiode 2']**2
toilet["Red21"]=toilet['Red LED 2\nPhotodiode 1']**2
toilet["Red22"]=toilet['Red LED 2\nPhotodiode 2']**2
X=toilet[["Blue11", "Blue12", "Blue21", "Blue22", "Green11", "Green12", "Green21", "Green22", "Red11", "Red12",
    "Red21", "Red22"]]
X = sm.add_constant(X)
Y=toilet[["Flush volume"]]
model = sm.OLS(Y, X)
results = model.fit()
print(results.summary())
bins = [-100,0.75,1.7,2.15,2.6,3.05,3.55,4,4.45,4.95,5.4,5.85,100]

toilet.loc[:,'FlushV  int']=pd.cut(toilet['Flush volume'],bins,
                                  labels=[0,1,2,3,4,5,6,7,8,9,10,11])
Y2=toilet.loc[:,'FlushV_int']
ypred=results.predict(X)
toilet.loc[:,"Ypred"]=ypred

toilet.loc[:,'Flushpr_int']=pd.cut(ypred,bins,labels=[0,1,2,3,4,5,6,7,8,9,10,11])
X2=toilet['Flushpr_int']
results = confusion  matrix(Y2,X2)
y_pred = pd.Series(toilet['Flushpr_int'])
y_true = pd.Series(toilet["Flush volume"])
fchi=pd.crosstab(y_true, y_pred, rownames=['True'], colnames=['Predicted'], margins=True)
fchi=stats.chi2  contingency(fchi)
print(fchi)
```

OLS Code

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
from sklearn.metrics import confusion_matrix
import scipy.stats as stats
import seaborn as sns
from sklearn.metrics import accuracy_score

toilet_data = pd.read_excel("File 7 - Group7.xlsx", header = 0, sep = " ")
toilet=toilet_data

print(toilet.columns)

print(toilet["Flush volume"].value_counts())

while (1):

    #xn=[0.6049757,0.80858354,0.257002,0.63890724,0.5909667,0.59050965,
    #0.61059362,0.10776922,0.33897334,0.31753335,0.67646697,0.49213317]#
    xn=np.random.rand(12)
    # 0.4095061179987186 8.742664186153026e-31
    # 0.3704170514493971 5.861962592836711e-33
    #[0.6049757  0.80858354 0.257002   0.63890724 0.5909667  0.59050965
    #0.61059362 0.10776922 0.33897334 0.31753335 0.67646697 0.49213317] 3.326956159301724e-42

    #your model
    toilet["Blue11"]=toilet['Blue LED 1\nPhotodiode 1']**xn[0]
    toilet["Blue12"]=toilet['Blue LED 1\nPhotodiode 2']**xn[1]
    toilet["Blue21"]=toilet['Blue LED 2\nPhotodiode 1']**xn[2]
    toilet["Blue22"]=toilet['Blue LED 2\nPhotodiode 2']**xn[3]
    toilet["Green11"]=toilet['Green LED 1\nPhotodiode 1']**xn[4]
    toilet["Green12"]=toilet['Green LED 1\nPhotodiode 2']**xn[5]
    toilet["Green21"]=toilet['Green LED 2\nPhotodiode 1']**xn[6]
    toilet["Green22"]=toilet['Green LED 2\nPhotodiode 2']**xn[7]
    toilet["Red11"]=toilet['Red LED 1\nPhotodiode 1']**xn[8]
    toilet["Red12"]=toilet['Red LED 1\nPhotodiode 2']**xn[9]
    toilet["Red21"]=toilet['Red LED 2\nPhotodiode 1']**xn[10]
    toilet["Red22"]=toilet['Red LED 2\nPhotodiode 2']**xn[11]


    X=toilet[[ 'Blue11', 'Blue12','Blue21', 'Blue22', 'Green11', 'Green12',
         'Green21', 'Green22', 'Red11','Red12', 'Red21', 'Red22']]
    X = sm.add_constant(X)
    Y=toilet[['Flush volume']]
    model = sm.OLS(Y, X)
    results = model.fit()
    print(results.summary())

    bins = [-100,0.75,1.7,2.15,2.6,3.05,3.55,4,4.45,4.95,5.4,5.85,100]

    toilet.loc[:,'FlushV_int']=pd.cut(toilet['Flush volume'],bins,
                                labels=[0,1,2,3,4,5,6,7,8,9,10,11])
    Y2=toilet.loc[:,'FlushV_int']
    ypred=results.predict(X)
    toilet.loc[:,"Ypred"]=ypred

    toilet.loc[:,'Flushpr_int']=pd.cut(ypred,bins,labels=[0,1,2,3,4,5,6,7,8,9,10,11])
```

```
    X2=toilet['Flushpr_int']
    results = confusion_matrix(Y2,X2)

    y_pred = pd.Series(toilet['Flushpr_int'])
    y_true = pd.Series(toilet["Flush volume"])
    fchi=pd.crosstab(y_true, y_pred, rownames=['True'], colnames=['Predicted'], margins=True)
    fchi=stats.chi2  contingency(fchi)
    if fchi[1]<3.326956159301724e-42:
        print(xn, fchi[1])
        break

sns.heatmap(results, annot=True,  fmt='', cmap='Blues')
```

GLM Code

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
from sklearn.metrics import confusion_matrix
import scipy.stats as stats
import seaborn as sns
from sklearn.metrics import accuracy_score
import warnings
from sklearn.metrics import classification  report
warnings.filterwarnings("ignore")

toilet_data = pd.read_excel("File 7 - Group7.xlsx", header = 0, sep = " ")
toilet=toilet  data

print(toilet.columns)

print(toilet["Flush volume"].value  counts())
xn=[0.17182156,0.42594121,0.8522537,0.513989,0.53666289,0.17705916
      ,0.655312,0.12910251,0.42398372,0.34662343,0.94068168,0.81186728]
#while (1):
#xn=np.random.rand(12)
#xn=[0.06199986,0.32110619,0.83769856,0.41505129,0.43535525,0.13027712,
#0.59053889,0.07540637,0.35578637,0.21652152,0.82629245,0.74468811]
#xn=[0.6049757,0.80858354,0.257002,0.63890724,0.5909667,0.59050965,
#0.61059362,0.10776922,0.33897334,0.31753335,0.67646697,0.49213317]#np.random.rand(12)
# 0.4095061179987186 8.742664186153026e-31
# 0.3704170514493971 5.861962592836711e-33
#[0.6049757  0.80858354 0.257002   0.63890724 0.5909667  0.59050965
#0.61059362 0.10776922 0.33897334 0.31753335 0.67646697 0.49213317] 3.326956159301724e-42
toilet["Blue11"]=toilet['Blue LED 1\nPhotodiode 1']**xn[0]
toilet["Blue12"]=toilet['Blue LED 1\nPhotodiode 2']**xn[1]
toilet["Blue21"]=toilet['Blue LED 2\nPhotodiode 1']**xn[2]
toilet["Blue22"]=toilet['Blue LED 2\nPhotodiode 2']**xn[3]
toilet["Green11"]=toilet['Green LED 1\nPhotodiode 1']**xn[4]
toilet["Green12"]=toilet['Green LED 1\nPhotodiode 2']**xn[5]
toilet["Green21"]=toilet['Green LED 2\nPhotodiode 1']**xn[6]
toilet["Green22"]=toilet['Green LED 2\nPhotodiode 2']**xn[7]
toilet["Red11"]=toilet['Red LED 1\nPhotodiode 1']**xn[8]
toilet["Red12"]=toilet['Red LED 1\nPhotodiode 2']**xn[9]
toilet["Red21"]=toilet['Red LED 2\nPhotodiode 1']**xn[10]
toilet["Red22"]=toilet['Red LED 2\nPhotodiode 2']**xn[11]



X=toilet[[ 'Blue11', 'Blue12','Blue21', 'Blue22', 'Green11', 'Green12',
        'Green21', 'Green22', 'Red11','Red12', 'Red21', 'Red22']]
X = sm.add_constant(X)
Y=toilet[['Flush volume']]

bins = [-100,0.75,1.7,2.15,2.6,3.05,3.55,4,4.45,4.95,5.4,5.85,100]

toilet.loc[:,'FlushV_int']=pd.cut(toilet['Flush volume'],bins,
labels=[0,1,2,3,4,5,6,7,8,9,10,11]).astype("int")
Y2=toilet.loc[:,'FlushV_int']

model = sm.GLM(Y2, X, family=sm.families.Gaussian())
results = model.fit()
#print(results.summary())
```

```
59.
60. ypred=results.predict(X)
61. np.where(ypred<0,0,ypred)
62. Ypred=np.where(ypred<0,0,ypred).round()
63.
64. #toilet.loc[:,'Flushpr_int']=pd.cut(ypred,bins,labels=[0,1,2,3,4,5,6,7,8,9,10,11])
65. #X2=toilet['Flushpr  int']
66. resultsns = confusion_matrix(Y2,Ypred)
67.
68. #y_pred = pd.Series(toilet['Flushpr_int'])
69. #y  true = pd.Series(toilet["Flush volume"])
70. fchi=pd.crosstab(Y2,Ypred, rownames=['True'], colnames=['Predicted'], margins=True)
71. fchi=stats.chi2_contingency(fchi)
72. print(fchi[1])
73.
74. print(sns.heatmap(resultsns, annot=True,  fmt='', cmap='Blues'))
75. print(accuracy_score(Y2,Ypred))
76. print(classification_report(Y2,Ypred))
```