Team 8 - Python Machine Learning Continuous Assessment

|  |  |
| --- | --- |
| Antonio | E0384801 |
| Wee Hui Ching | E0384796 |
| Kyaw Sithu | E0395894 |
| Neo Wei Sheng | E0388060 |
| Udaya Bhaskar Reddy Malkannagari | E0384968 |
| Koh Zhen Xiang | E0385211 |
| Suramya Surendran | E0397329 |
| Ge Zhong Bo | E0390025 |

[**Supervised Learning**](#_3yw3jpnmpiz5) **5**

[**Introduction**](#_uh3nd79zchmf) **5**

[**Dataset Information**](#_16vjr3fne43t) **5**

[**Problem Statement**](#_m9b9l23dmukj) **5**

[**Simple Data Dictionary of training models (columns involved in training)**](#_nnnn4f2rh0ev) **5**

[**No of records and columns in dataset**](#_xas1w4k0xzky) **6**

[**No of records and columns in the training/test dataset**](#_1vd2ggqn1qh) **6**

[**Data Engineering**](#_jmjlwx1i7zck) **7**

[**Undersampling data/Data Balancing**](#_f5pz2myj3zfe) **9**

[**Supervised Learning (Logistic Regression Report)**](#_6yg3g37l3r9r) **12**

[**Using Data Set Without Normalization**](#_7affw6dxyvev) **12**

[**How data/feature/model engineering was performed to achieve a better outcome**](#_j7eu772j4iyg) **15**

[**Using Data Set to Test Another Attribute**](#_j3akijd5bf1z) **17**

[**Using Data Set to Test Multiple High Correlation Columns**](#_b186hpaz0ml2) **19**

[**Duration of training for Logistic Regression**](#_894ayp4oxyl1) **21**

[**Logistic Regression Conclusion**](#_u9yrd4i9c6x9) **21**

[**Supervised Learning (K-N Neighbors Classification Report)**](#_mwsh44s43hmu) **22**

[**Comparing the Attributes - Loan**](#_b64npdlik995) **22**

[**Comparing the Attributes - Education**](#_y1o1drkoq20c) **23**

[**Using Data Set Without Normalization**](#_hfjvfuwds4u6) **24**

[**Using Data Set to Test Another Attribute**](#_3p9o5elyb7r) **29**

[**Using Data Set to Test Multiple High Correlation Columns**](#_e1c8okymzqv2) **32**

[**Using Housing, Loan with 75:25 training and testing dataset**](#_vchfjxnskpvb) **35**

[**Duration of training for K-NN**](#_rgwl9yv89uzy) **38**

[**K-NN Conclusion**](#_5hrrrjaia4uq) **38**

[**Supervised Learning (Support Vector Machine)**](#_xt94qisxua1c) **39**

[**Using Data Set Without Normalization**](#_z1vz4y27ithw) **39**

[**How data/feature/model engineering was performed to achieve a better outcome**](#_jzodi93pka77) **41**

[**Duration of training for Support Vector Machines**](#_2os1iowlejj1) **42**

[**Support Vector Machine Conclusion**](#_us329fn9anmz) **43**

[**Supervised Learning (Decision Tree/Random Forest Report)**](#_y7quijexjd88) **43**

[**Using dataset without Normalization**](#_z7khd08jrpv7) **43**

[**How data/feature/model engineering was performed to achieve a better outcome**](#_it3ot6jko02y) **45**

[**Optimizing the decision tree performance**](#_c80t6ni7dusn) **47**

[**Duration of training for Tree Diagram**](#_qlilnnzcd6sr) **48**

[**Decision Tree Conclusion**](#_ox5cie1ezx63) **48**

[**Overall Conclusion**](#_hn31qj2l76oy) **49**

[**Unsupervised Learning**](#_sb1l1csemrko) **50**

[**Introduction**](#_sqdtpe6uuoim) **50**

[**Problem Statement**](#_5cvvtcr1gtz4) **50**

[**Simple Data Dictionary of training models (columns involved in training), incl source of dataset**](#_haibu89rruh5) **50**

[**No of records and columns in dataset**](#_9tatn7meknk1) **50**

[**Feature Engineering (Data Balancing and Data Cleansing)**](#_9au2hjbt94bl) **51**

[**Balancing the dataset**](#_wqb68z8w0d9j) **53**

[**PCA and LDA**](#_c2dxpolqdw60) **55**

[**Duration of training for PCA and LDA**](#_jkzjq7s861dn) **60**

[**PCA and LDA Conclusion**](#_kkjpj39v3acf) **60**

[**DBSCAN**](#_7obrz0ipnk99) **61**

[**Using Data Set Without Normalization**](#_fipxgw5fx0jy) **61**

[**Duration of training for Tree Diagram**](#_qn93fc4jja0y) **63**

[**DB SCAN Conclusion**](#_6h0dilxdb7lw) **64**

[**Agglomerative Clustering**](#_3z94c0n6op5r) **65**

[**Using Data Set Without Normalization**](#_yrt9b7fczfui) **65**

[**How data/feature/model engineering was performed to achieve a better outcome**](#_wimyna59jv2y) **69**

[**Duration of training for each model**](#_ouxx3pm62dc0) **71**

[**Agglomerative Clustering Conclusion**](#_hcnzf6xqq28w) **72**

[**K Means Clustering Analysis**](#_lx011ho36s8l) **72**

[**Visualizing the pair plots**](#_k2rvrd3qwzrg) **73**

[**Correlation Matrix**](#_lkihbe65kn3i) **74**

[**Correlation Heat Map**](#_8q2djs49vdxi) **74**

[**Checking positive correlation**](#_q5y9o980omrg) **74**

[**Selecting variables which we think will help us form good clusters**](#_zecbrwy4jspt) **75**

[**Box plotting**](#_a6p17e6gyg7u) **75**

[**1. Alcohol percent vs Wine Quality**](#_ul0bofngea7b) **75**

[**2. Free Sulphur Dioxide percent vs Wine Quality**](#_dsut3p3l6255) **76**

[**3. pH vs Wine Quality**](#_o8m6opqb8mxh) **76**

[**4. Citric Acid vs Wine Quality**](#_r356kcc0xokw) **77**

[**Deduction from the box plots**](#_ld9m5konytxd) **77**

[**K-Means Clustering Algorithm**](#_hwwb9ppgqaij) **78**

[**Finding the ideal number of Clusters**](#_mca7u3oi4yxn) **78**

[**Visualizing the clusters**](#_z81ago4dzhv5) **79**

[**K Means Clustering using the original unbalanced dataset**](#_1eqhzlzd2y00) **80**

[**K Means Clustering using negatively correlated Attributes**](#_ejfjtyd18m15) **81**

[**Visualizing the Clusters**](#_purjegt8py28) **82**

[**Kmeans clustering using the standardized values (x\_sdr) with all features included**](#_g7pfr1rq5io0) **82**

[**Duration of training for k-Means Analysis**](#_ibi721lt83ce) **84**

[**K-means Analysis and Conclusion**](#_8p2frgg7isc1) **85**

[**Overall Conclusion**](#_4ne3gm7wp4e4) **86**

# 

# Supervised Learning

## Introduction

Dataset Source: <https://archive.ics.uci.edu/ml/datasets/bank+marketing>

## Dataset Information

The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

## Problem Statement

What are the factors that influence an individual to subscribe to a term deposit? (Yes : subscribe, No: Don't’ subscribe)

## Simple Data Dictionary of training models (columns involved in training)

|  |  |
| --- | --- |
| Input Variables | |
| Age (Numeric) | Job (Type of job in categories) |
| Marital (Marital Status) | Education (Type of education in categories) |
| Default (Has credit in default? (Binary)) | Balance (average yearly balance, in euros) |
| Housing (Has housing loan? (Binary)) | Loan (Has personal loan? (Binary)) |
| Contact (Contact Communication type) | Day (Last contact day of the month) |
| Month (Last contact of year) | Duration (Last contact duration in seconds) |
| Campaign (Number of contacts performed during this campaign) | pdays(Number of days that passed by after the client was last contacted from a previous campaign) |
| Previous (Number of contacts performed before this campaign and for this client) | Poutcome (Outcome of the previous marketing campaign ) |

Output variable (desired target): Y - Has the client subscribed to a term deposit? (binary: "yes","no")

## No of records and columns in dataset

No. of records in dataset: 45211, no of columns = 17.

## No of records and columns in the training/test dataset

10024 after undersampling. For comparison across supervised learning methods, we use the columns ‘housing’ and ‘loan’ for the input variables. Reasons are explained in the Data engineering section.

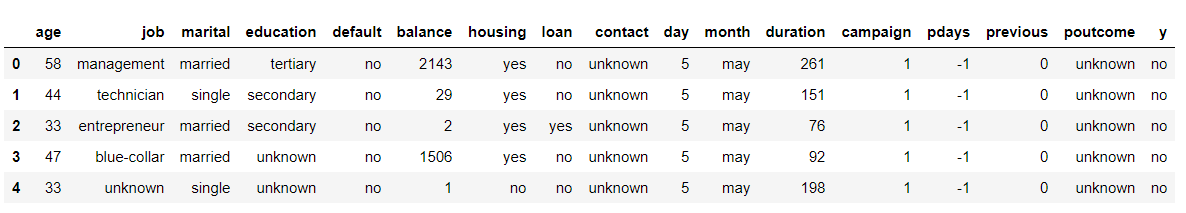
# Data Engineering

First, we import the Python Libraries:

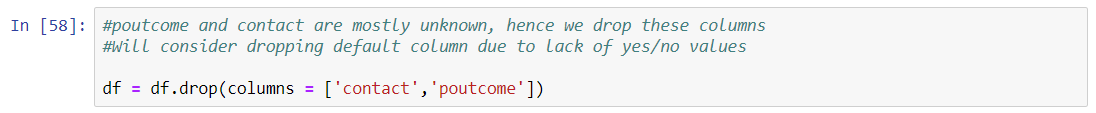


Our team has selected the following dataset and Python will read the CSV file for processing :

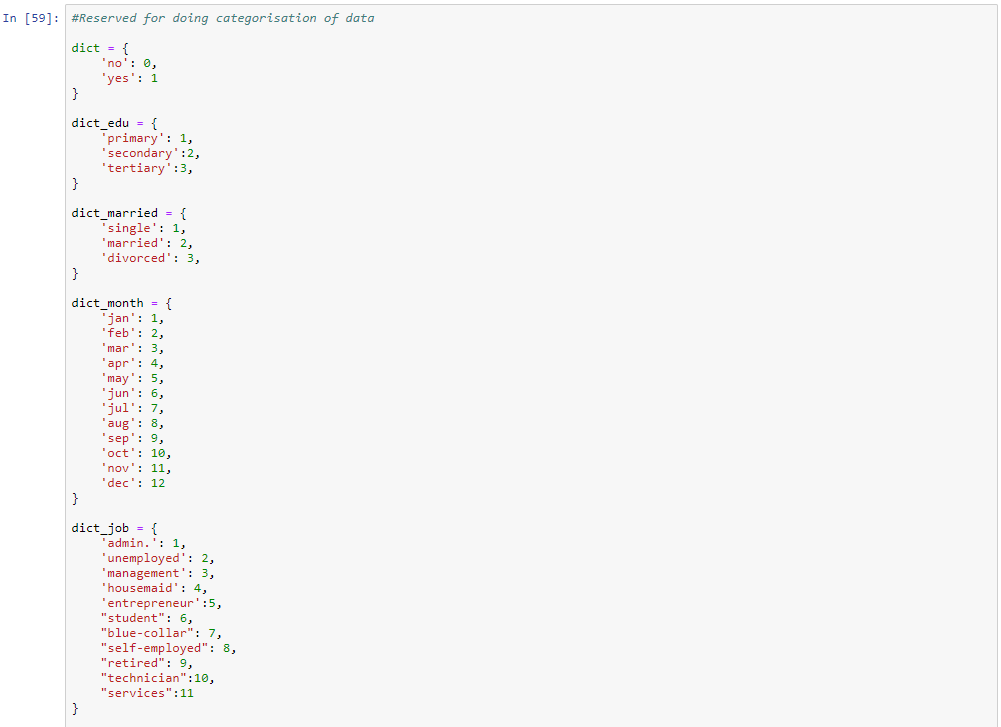


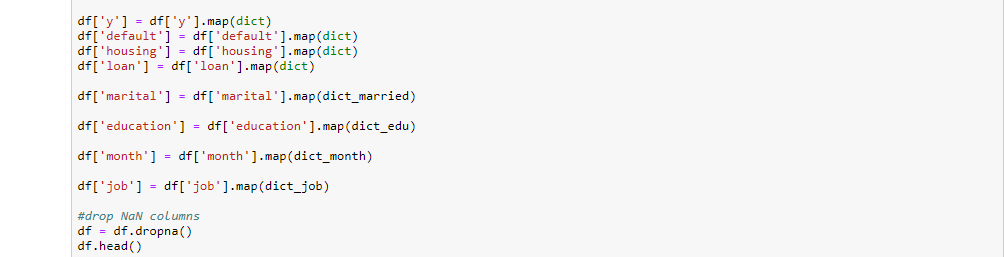


For the first round of data engineering, we dropped the columns **‘contact’** and **‘poutcome’** as there are too many ‘unknown’ inputs in the dataset:



Next, we mapped the string values of the data to our dictionary as follows:





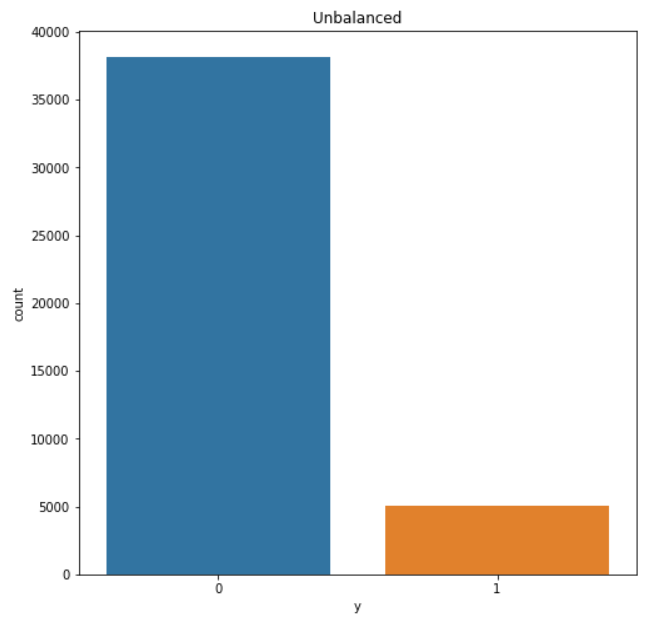
Result after mapping:



## Undersampling data/Data Balancing

Using the newly mapped data, we plotted a chart to further examine the data before applying learning models to it. The first histogram plot of data is shown below:

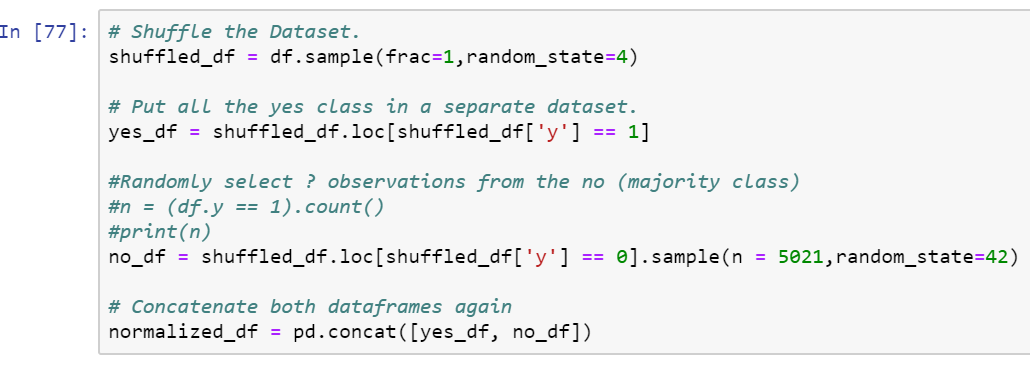




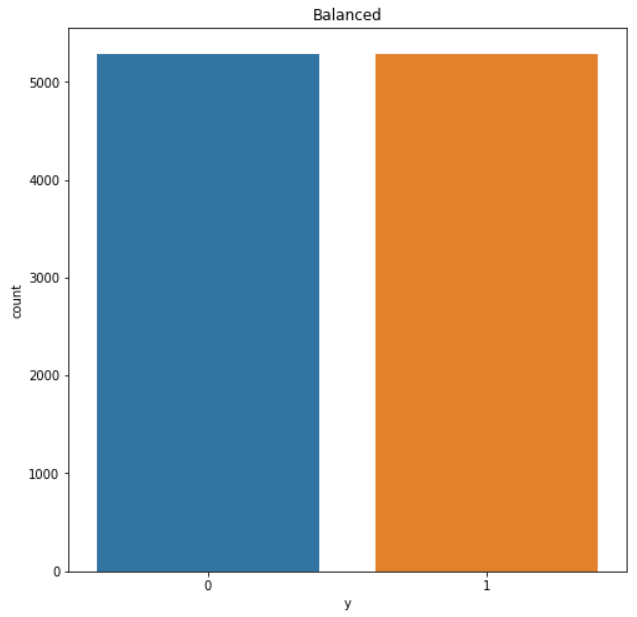
As shown, there is a skew of data on one category. By performing a count on the data, we can know the number of the minority class. (Category 0 or ‘no’ is overwhelmingly large [total = 38172 observations] compared to Category ‘1’ or yes[total = 5021 observations])



As a first strategy, undersampling was applied in order to balance the y-values.





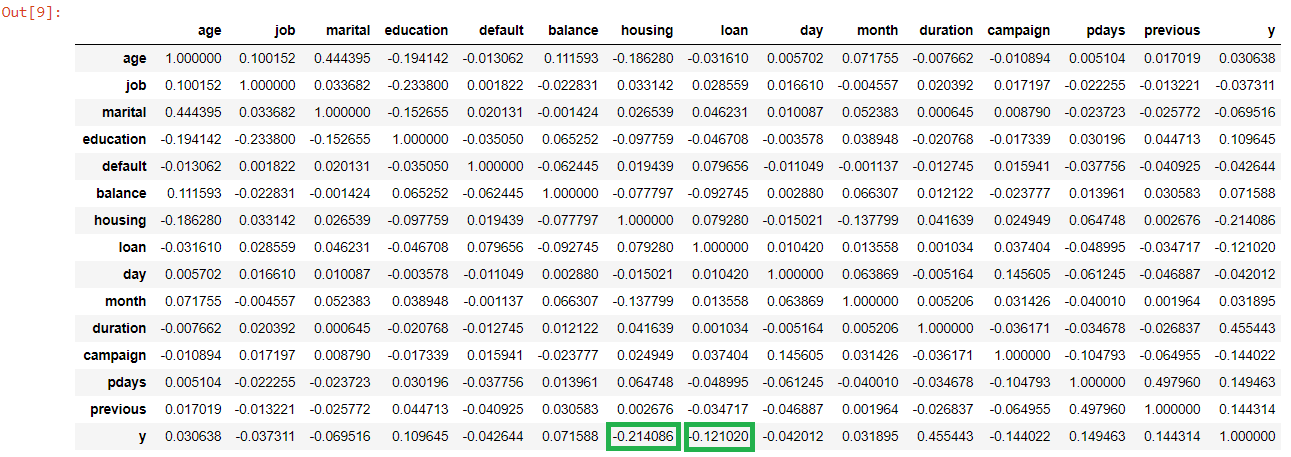


We replotted the histogram after balancing the data and the results are shown in the histogram above.

Next, we did a correlation analysis to select the attributes:



Results:



No of records and columns in dataset (after balancing and normalising) :

Columns : 15 columns (14 features and 1 outcome)

Records : 10042 records

The attributes that we have selected are ‘housing’ and ‘loan’. Despite “duration” attribute having the highest correlation, it was not selected as it was explicitly mentioned in the data source not to use it.

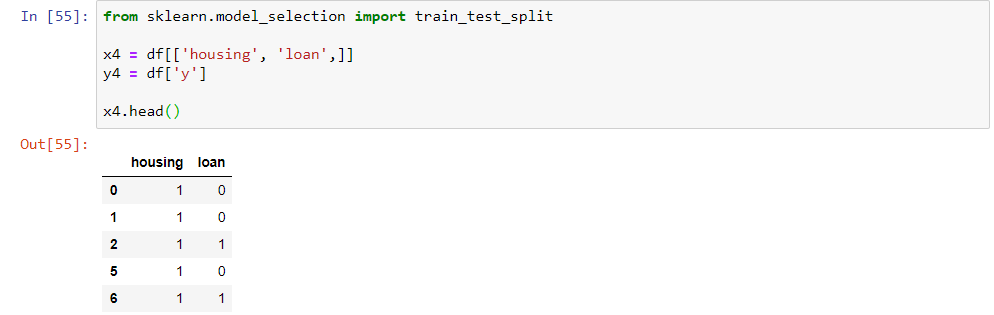
Fewer factors actually give better performance than choosing more factors. As a result, we stopped based on these two attributes, which give the higher (but still not very good) correlations

# 

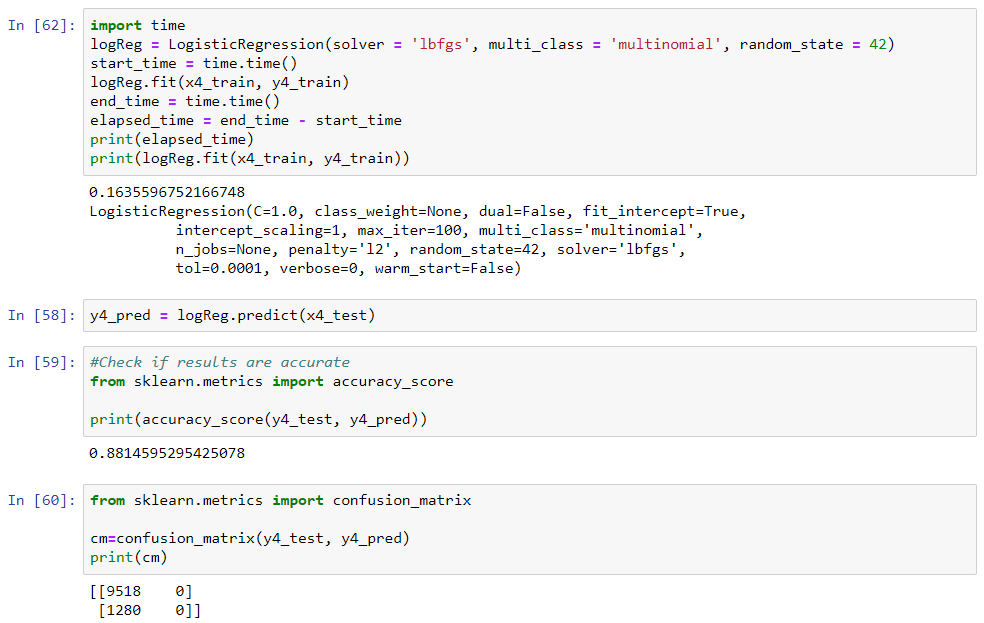
# **Supervised Learning (Logistic Regression Report)**

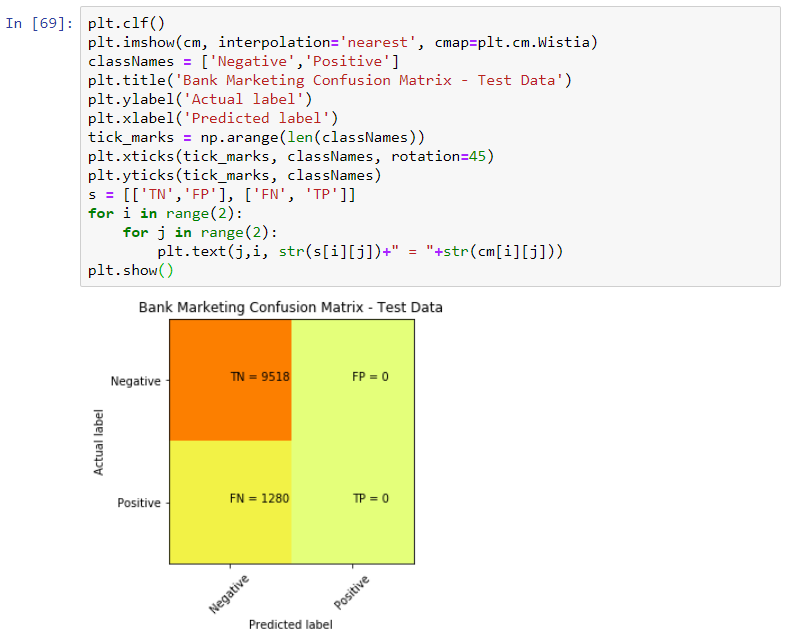
## Using Data Set Without Normalization

First, we would like to test the results if the data set is not balanced. Instead of using normalized\_df, we will use just df. We will take the same attributes (Housing and Loan).









Accuracy = (0+9518)/10,798 = 0.88145

Accuracy score matches sklearn.metrics which confirms the scoring.

Recall = 0/(0+1280) = 0

Out of all the positive classes, none was predicted correctly.

Precision = 0/(0+0) = 0

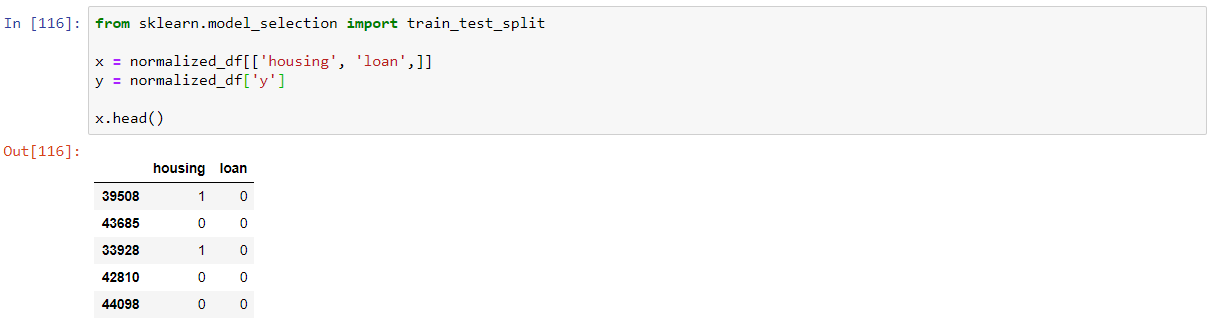
Out of all the classes (Both positive and negative), none was predicted correctly.

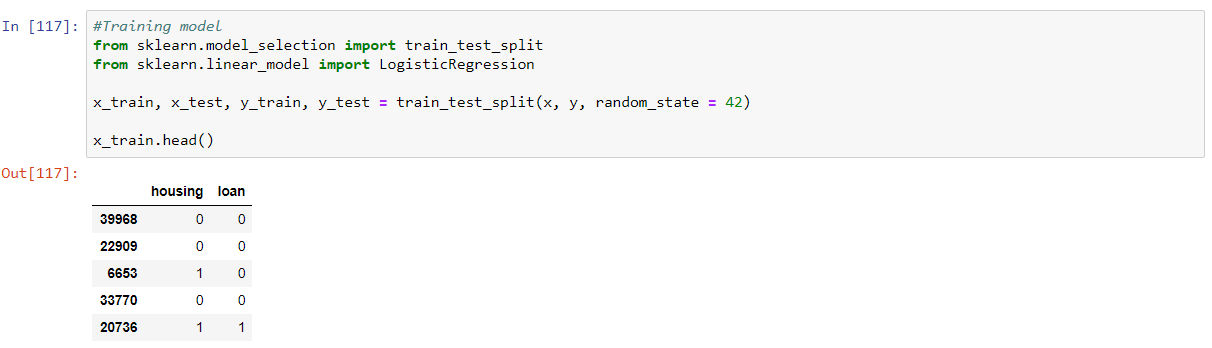
As seen in the above results, the accuracy score is very high at 88%, however, we do not think that this is accurate and thus we ran the confusion matrix. True enough, the results can be seen as very skewed as this is due to the high occurrence of ‘0’ in the dataset. Both recall and precision scores was 0 which does not make the unnormalized dataset feasible to be used.

As you can see from the confusion matrix, the True Negatives and is very high as the dataset consists of approximately 30,000 negative values (“No”) is to 5,000 (“Yes”).

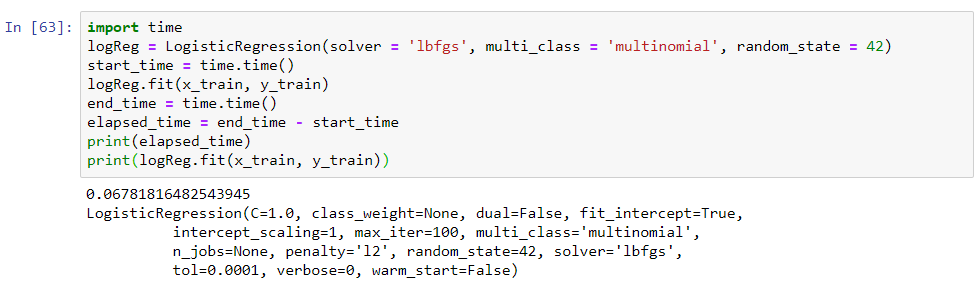
## How data/feature/model engineering was performed to achieve a better outcome

Based on the above observation, we started with the normalized dataset ‘normalized\_df”.



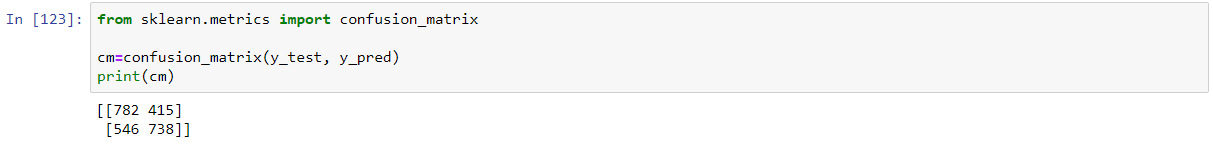


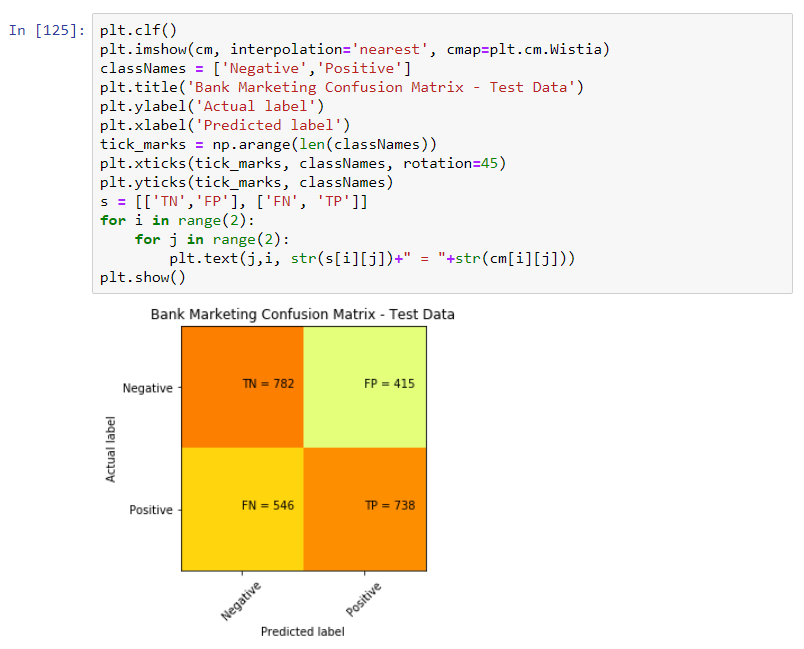
We added an elapsed time to see the duration the computer takes to run the dataset:











From the above Confusion Matrix, n = 2,481

Accuracy = (738+782)/2481 = 0.612656

Accuracy score matches sklearn.metrics which confirms the scoring.

Recall = 738/(738+546) = 738/1284 = 0.574766

Out of all the positive classes, 57% was predicted correctly.

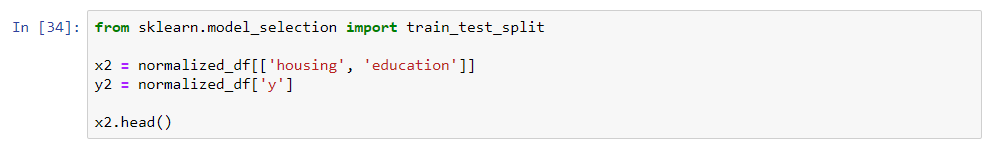
Precision = 738 /(738+415) = 738/1153 = 0.64006

Out of all the classes (Both positive and negative), 64% was predicted correctly.

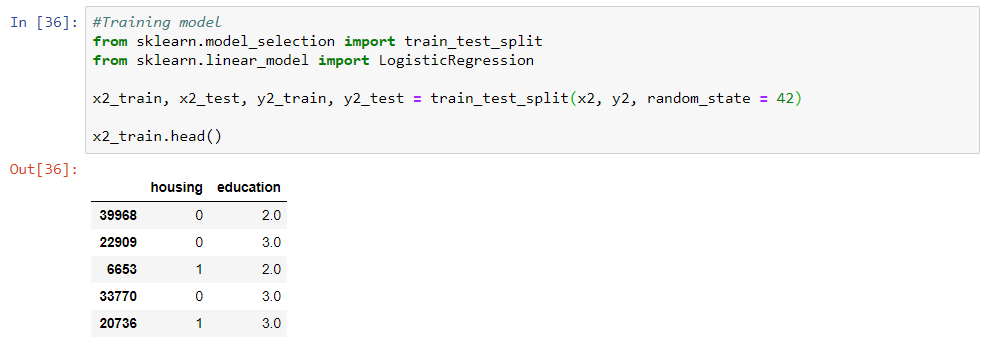
As seen in the above results, the accuracy score drops to 61%. To test if this is accurate, we ran the confusion matrix. True enough, the result reverted is a more accurate and precise. With this, we can conclude that using the balanced dataset produces a far more reliable and fair result even if the accuracy score drops.

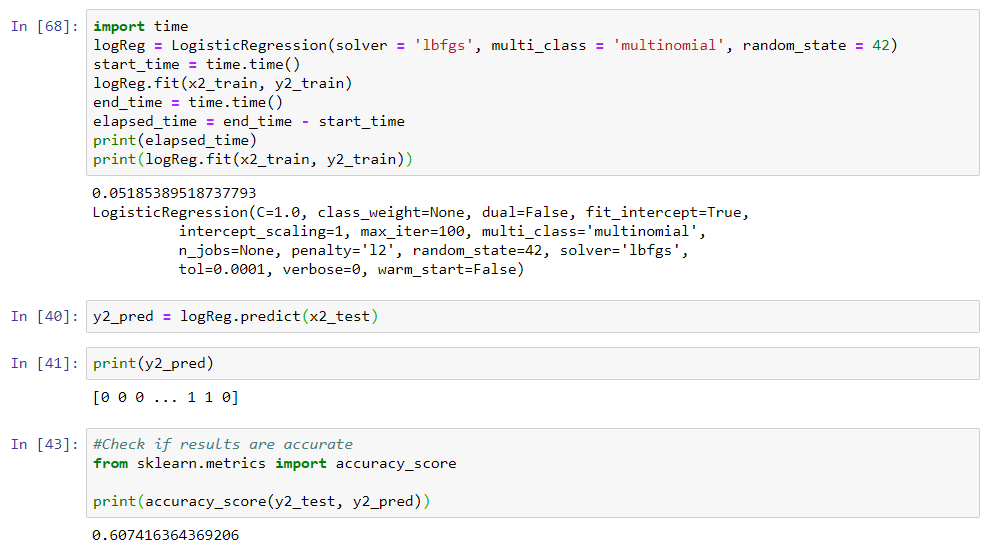
## Using Data Set to Test Another Attribute

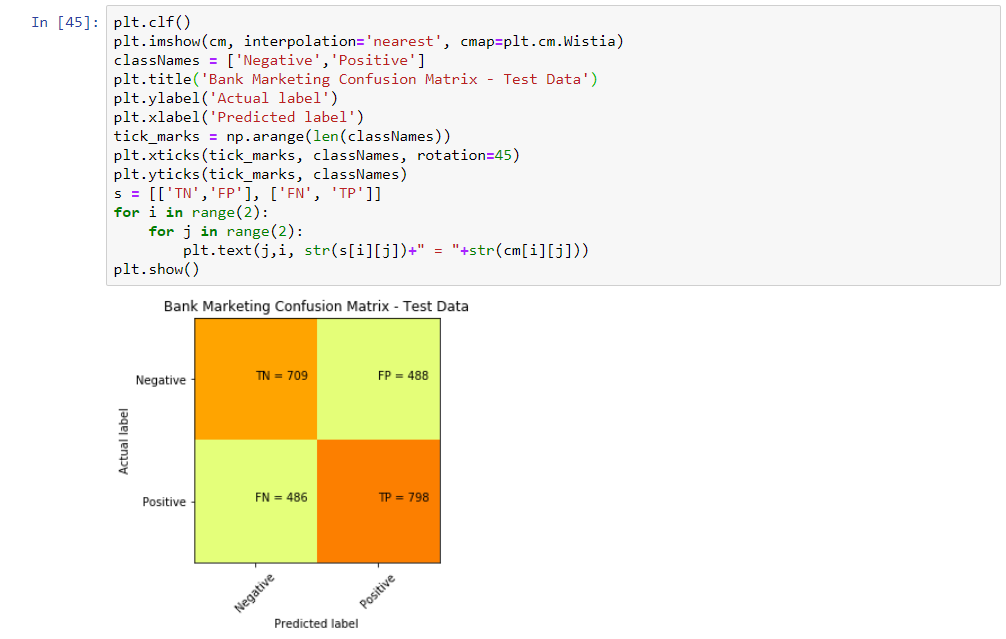
Here, we include another attribute with high correlation to test if the accuracy increases. We have chosen “education”.











Accuracy = (798+709) / 2481 = 0.6074

Accuracy score matches sklearn.metrics which confirms the scoring.

Recall = 798/(798+486) = 798/1284 = 0.6214

Out of all the positive classes, 62% was predicted correctly which was higher than the 57% using loan.

Precision = 798 /(798+488) = 798/1286 = 0.6205

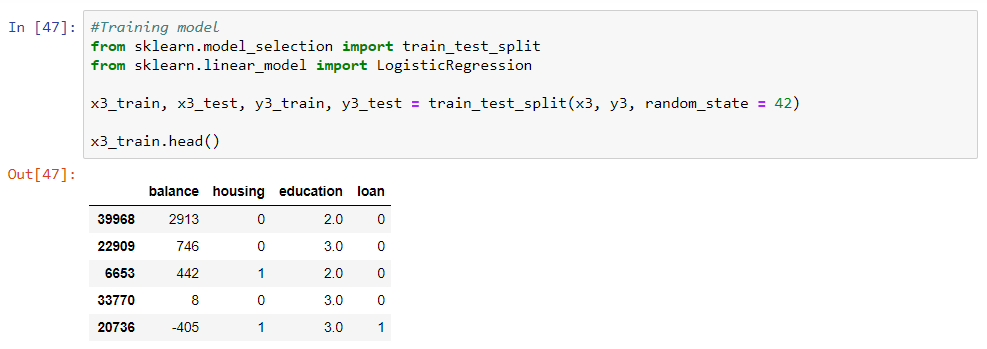
Out of all the classes (Both positive and negative), 62% was predicted correctly which is lower than the 64% using loan.

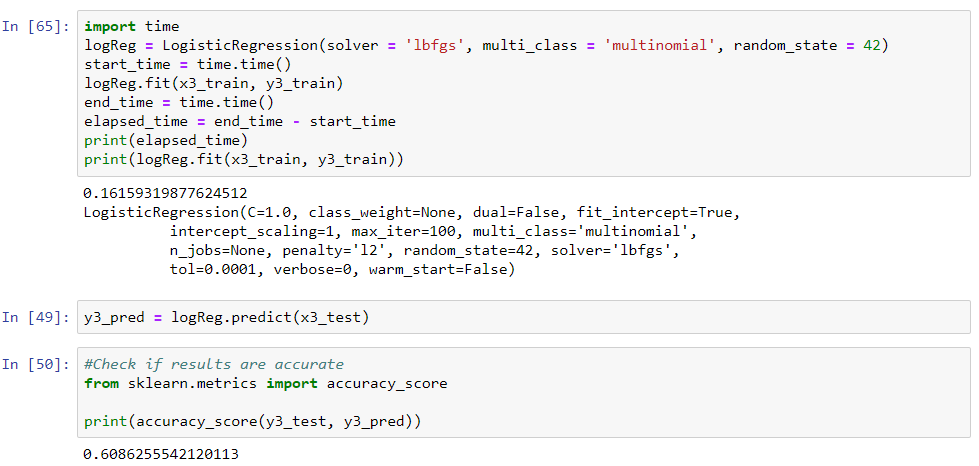
From the above, there are some differences in the Recall and Precision scores between “Education” and “Loan”.

## Using Data Set to Test Multiple High Correlation Columns

From the above test, we then tried to use the highest 4 correlation attributes to check if there is any difference in accuracy and precision; as well as the time taken to run the dataset.







From this, you can see that there is only a marginal change in the accuracy. We then move on to check confusion matrix.



Accuracy = (753+757) / 2481 = 0.6086

Accuracy score matches sklearn.metrics which confirms the scoring.

Recall = 753/(753+531) = 753/1284 = 0.5864

Out of all the positive classes, 58% was predicted correctly which was lower than both “Loan” and “Education”. This also shows that taking more attributes might not necessarily increase the prediction.

Precision = 753 /(753+440) = 753/1193 = 0.6311

Out of all the classes (Both positive and negative), 63% was predicted correctly.

## Duration of training for Logistic Regression

Here is a table of the duration, accuracy, precision and recall taken to run all the above.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy | Recall | Precision | Elapsed Time (Seconds) |
| Unnormalized Dataset | 0.8814 | 0 | 0 | 0.1635 |
| Normalized Dataset with “Housing” and “Loan” | 0.6126 | 0.5747 | 0.6400 | 0.0678 |
| Normalized Dataset with “Housing” and “Education” | 0.6074 | 0.6214 | 0.6205 | 0.0518 |
| Normalized Dataset with 4 Attributes (“Balance”, “Housing”, “Loan”, “Education”) | 0.6086 | 0.5864 | 0.6311 | 0.1615 |

## Logistic Regression Conclusion

As deduced earlier, using the unnormalized dataset will not serve to provide us with accurate and precise results.

As for using other highly correlated attributes, it does not improve the accuracy, neither does it significantly increase the recall and precision values. However, using 4 attributes significantly contributes to and increase in the elapsed time that the computer takes to run the data set. As seen in the above table, there is a 37% and 32% increase in duration with “Housing”/”Loan” and “Housing”/”Education” respectively. We will need to keep this in mind as the elapsed time will scale if much bigger data sets are used.

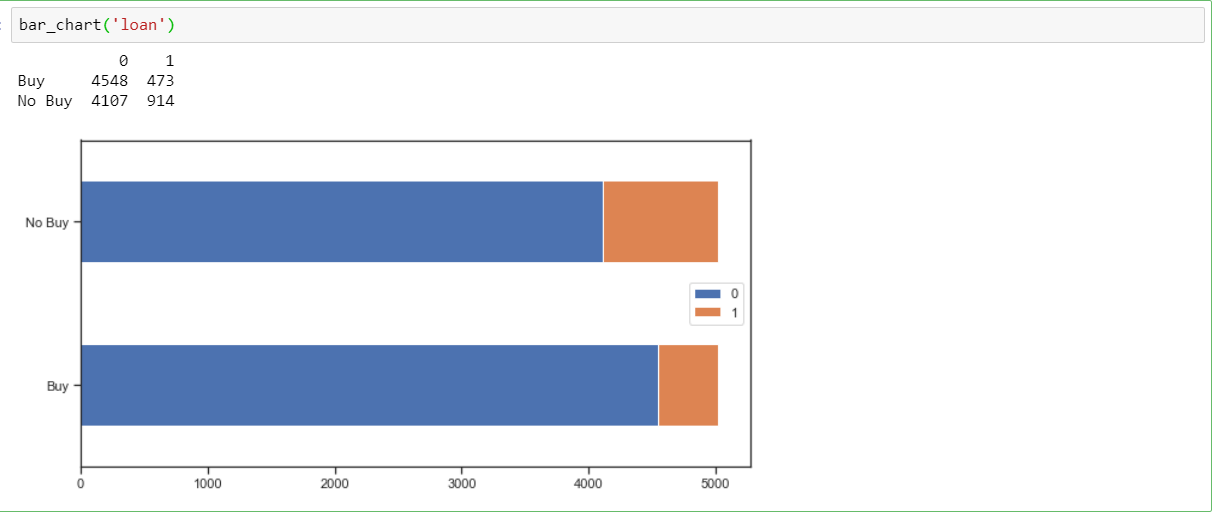
# 

# **Supervised Learning (K-N Neighbors Classification Report)**

For further exploration, let’s plot a bar chart to see the result of the outcome of people buying the bank term deposit based on several features : **“loan”, “housing”, “education”.**

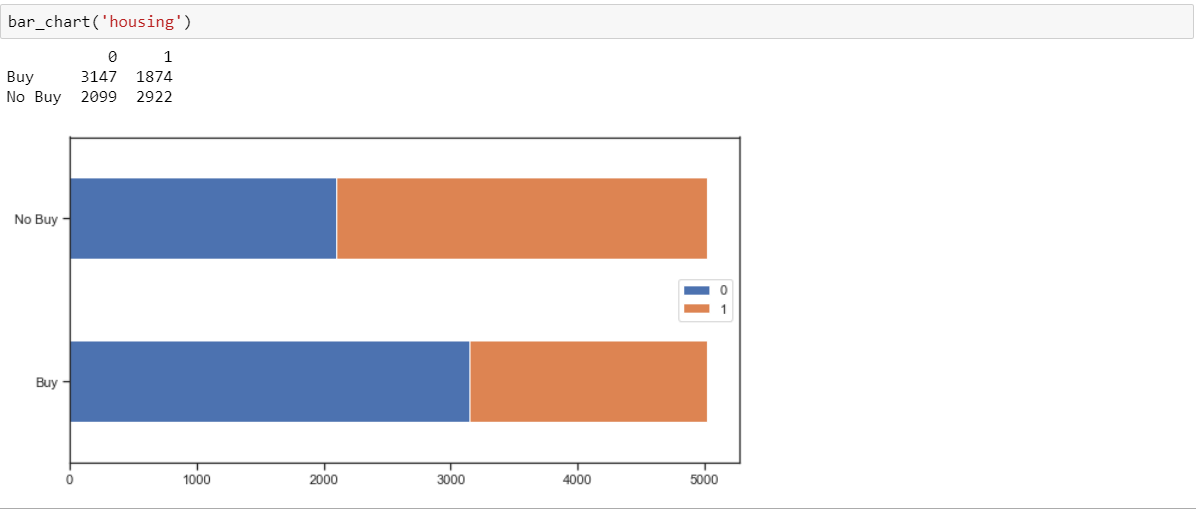


## Comparing the Attributes - Loan



*Category 0 : no personal loan, Category 1 : personal loan*

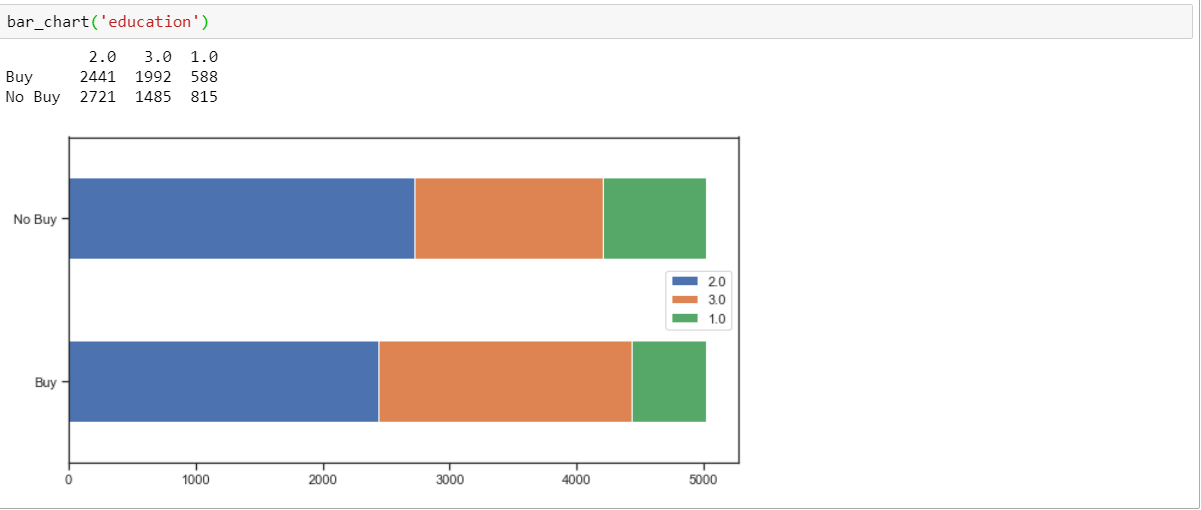
From the chart above, we can see there are more people with no loan(category : 0) buy bank term deposit compared to those who has loan(category : 1). The reason could be people with no loan have more financial freedom to buy other financial instruments.

Comparing the Attributes - Housing

*Category 0 : no housing loan, Category 1 : housing loan*

From the chart above, we can see there are more people with no housing loan(category : 0) buy bank term deposit compared to those who have housing loan(category : 1). The reason is similar to the personal loan aforementioned above : people who have housing loan has more financial obligations, hence less financial freedom for them to purchase another financial instruments.

## Comparing the Attributes - Education

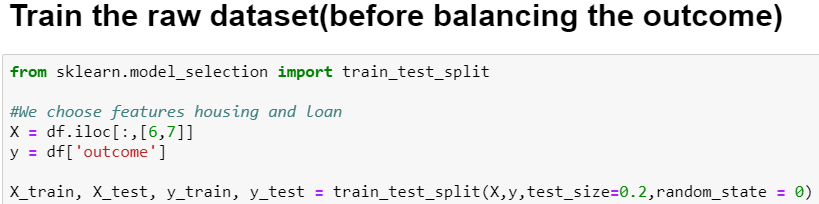
****

*Category 1 : primary, Category 2 : secondary, Category 3 : tertiary*

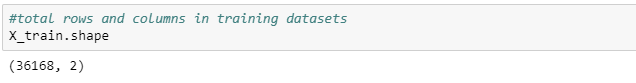
From the chart above, we can see there are more people with secondary and tertiary background that bought bank term deposit compared to people with primary background. The reason could be due to people with secondary and tertiary background has earned more salary than people with only primary background, hence they could have better financial position in buying fixed term deposit.

## Using Data Set Without Normalization

Let’s use the unbalanced dataset to do the training/testing and check the accuracy score using the K-N Neighbors classification for housing and loan

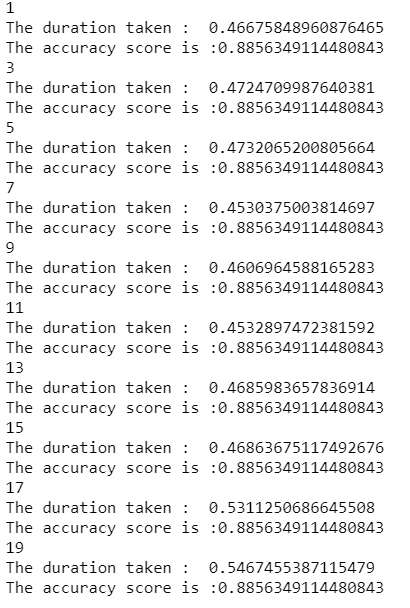




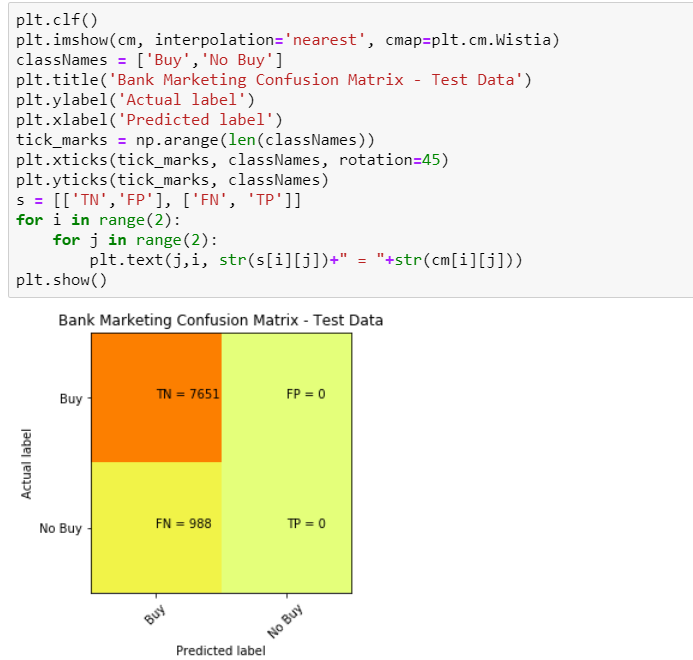










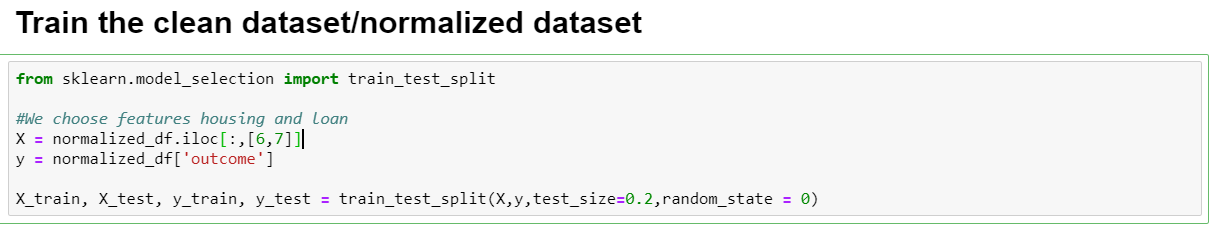
****

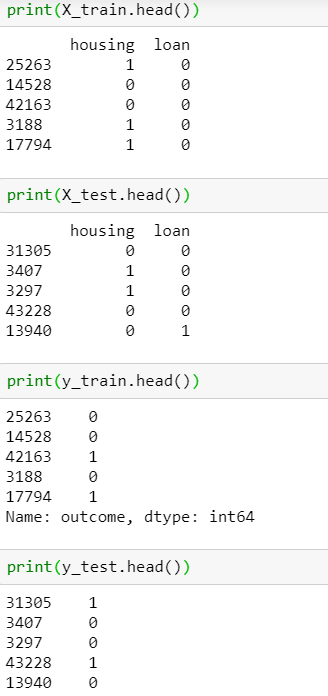
The result above shows if we use the unbalanced dataset, the accuracy score will increase to 88.5% . However, this does not provide a good picture/ does not mean the model we have trained is a good predictor to predict the testing data. The high accuracy is due to the unbalanced data of category ‘0’ in our dataset. Furthermore, every time we have some data we want to predict, it will most likely show the category ‘0’.

How data/feature/model engineering was performed to achieve a better outcome

Let’s take our normalized dataset and split the data for training and testing for K-N Neighbors Classification using different features. We are using the attributes “housing” and “loan”.

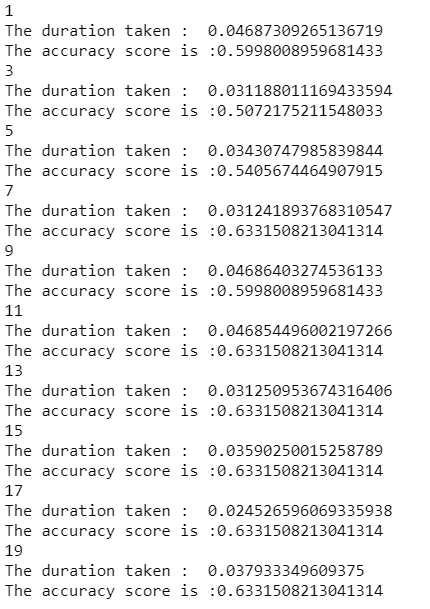
*\*Notice the dataset being used is different than the above (we use ‘normalized\_df‘ instead of ‘df’)*

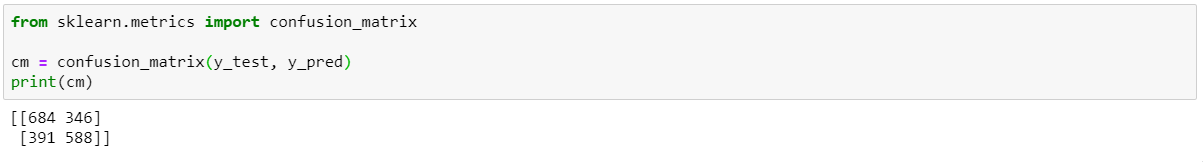


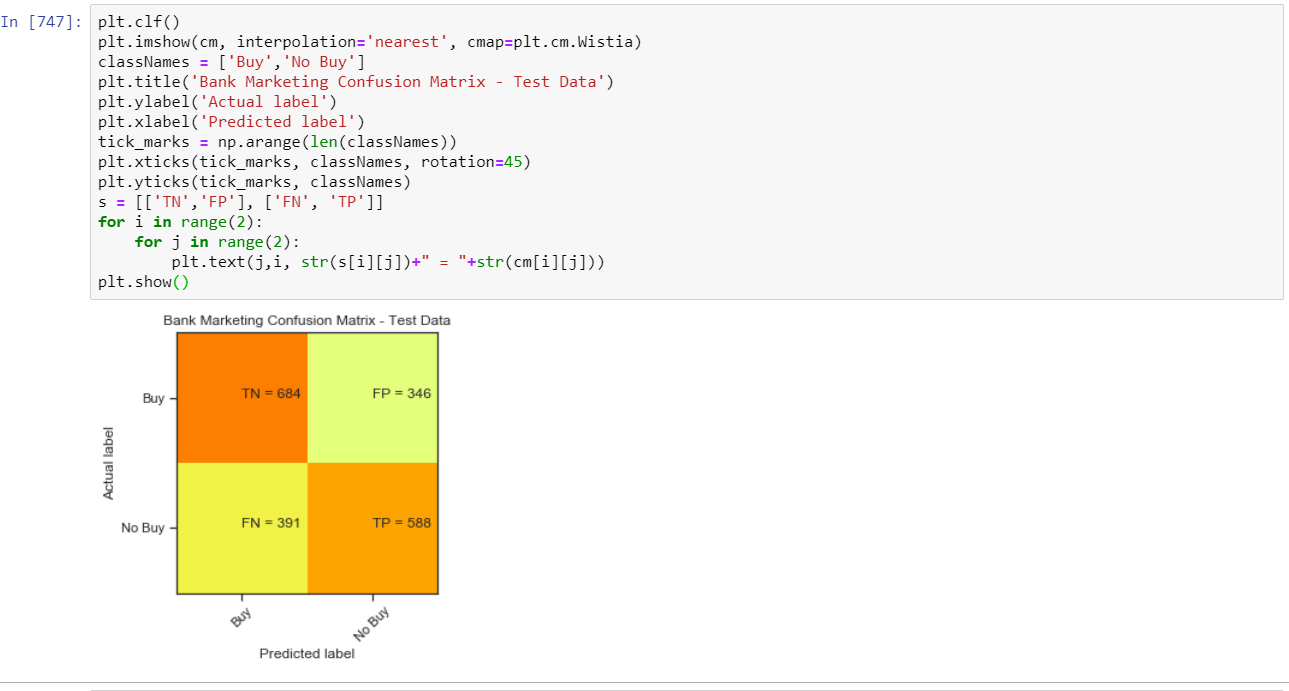






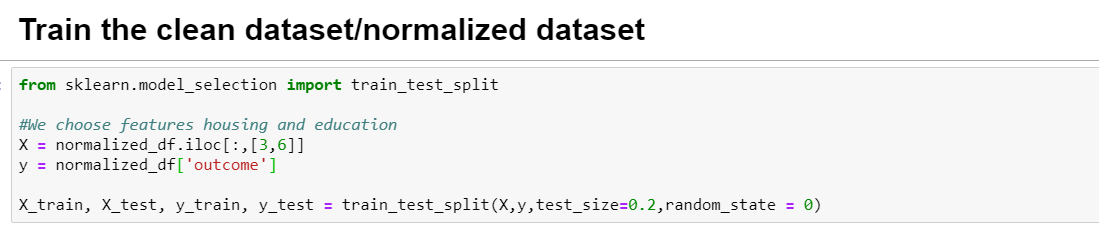


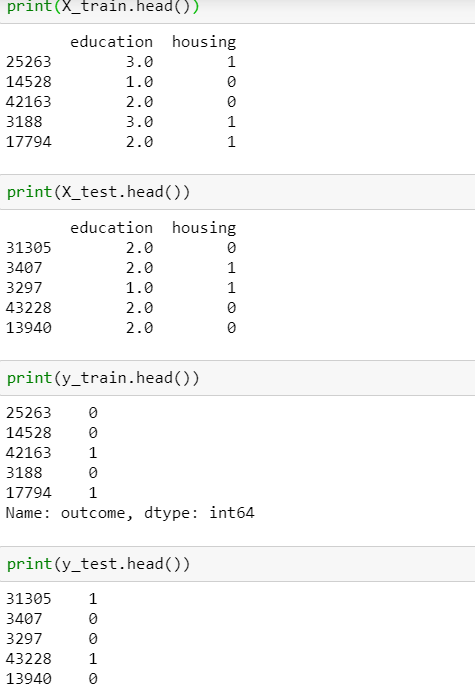




From the above testing, we can see that the best accuracy score will be achieved when the k is 7 or 11 to 19 (with an increment of 2). (we only define until 19 as the higher the k value , it will result in the inaccuracy of the training set, meaning it makes boundaries between classes less distinct )

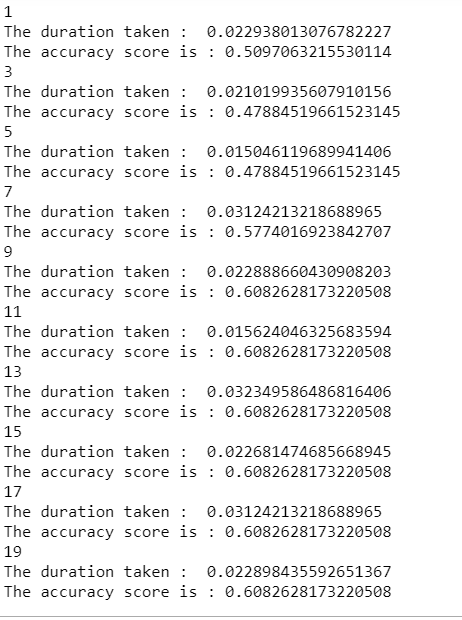
## Using Data Set to Test Another Attribute

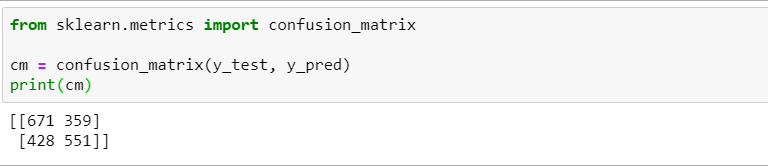


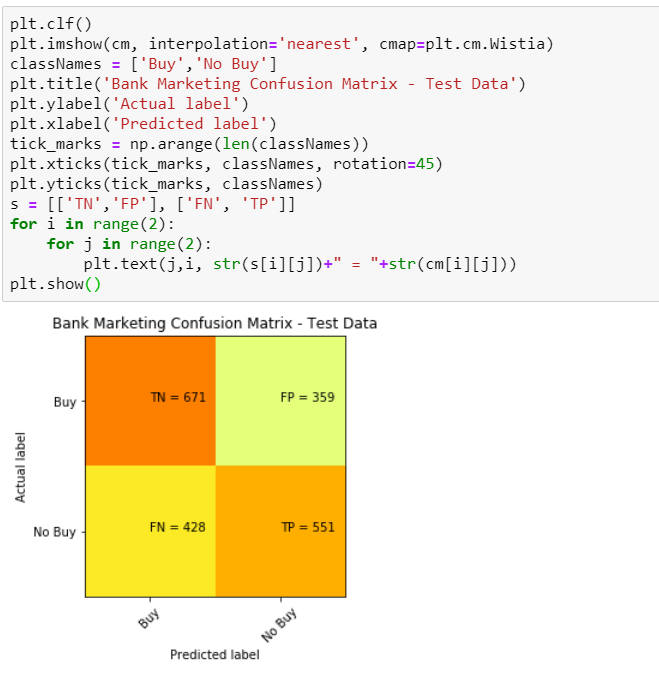






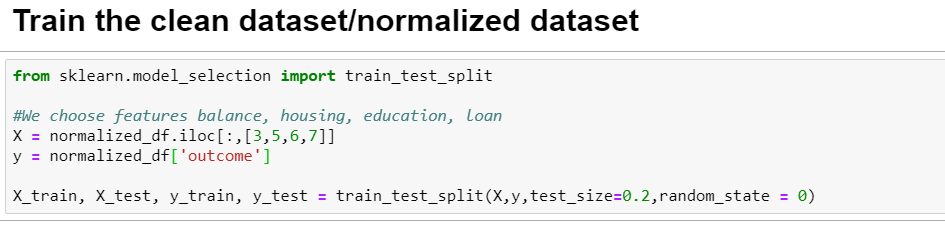


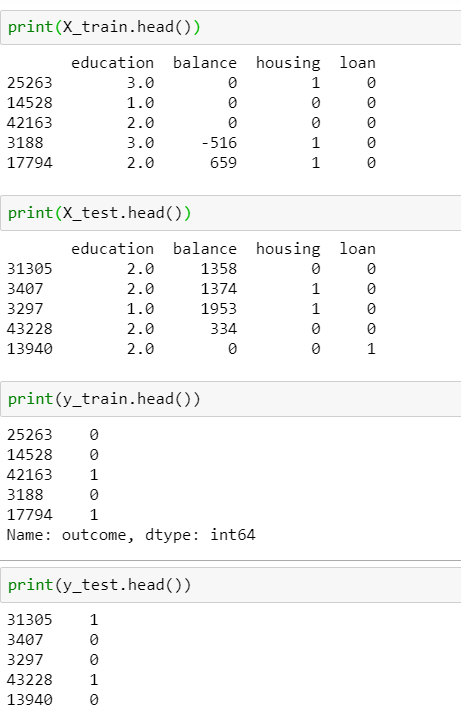
****



From the above testing, we can see that the best accuracy score (using ‘housing’ and ‘education’ will be achieved when the **k is 9 to 19 with increment of 2.** (we only define until 19 as the higher the k value , it will result in the inaccuracy of the training set, meaning it makes boundaries between classes less distinct.)

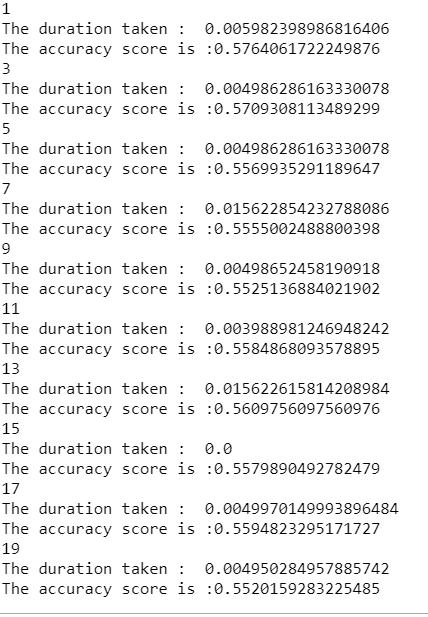
## Using Data Set to Test Multiple High Correlation Columns

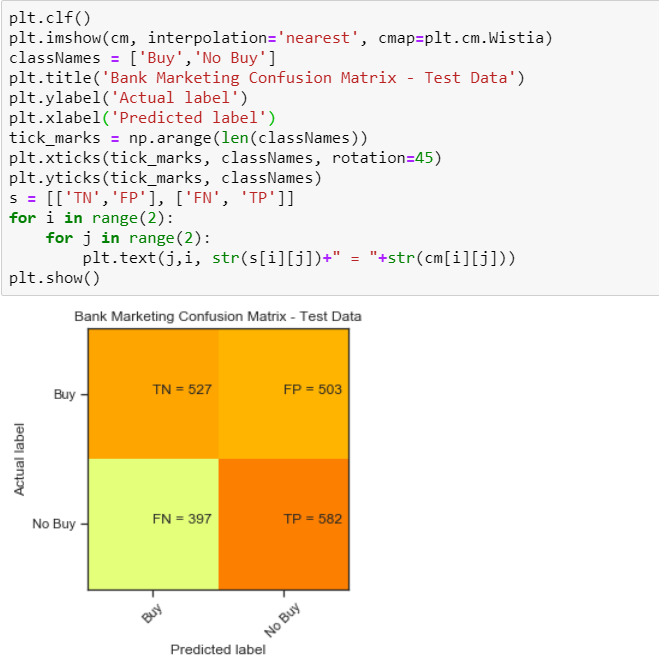


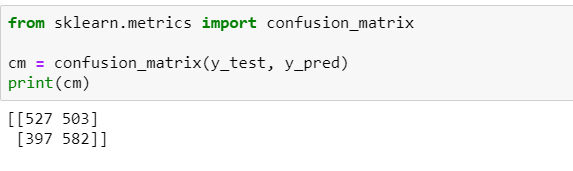










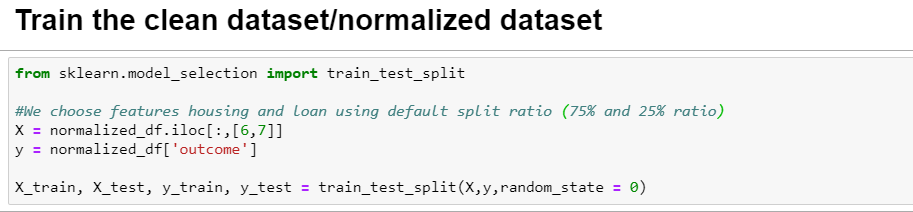


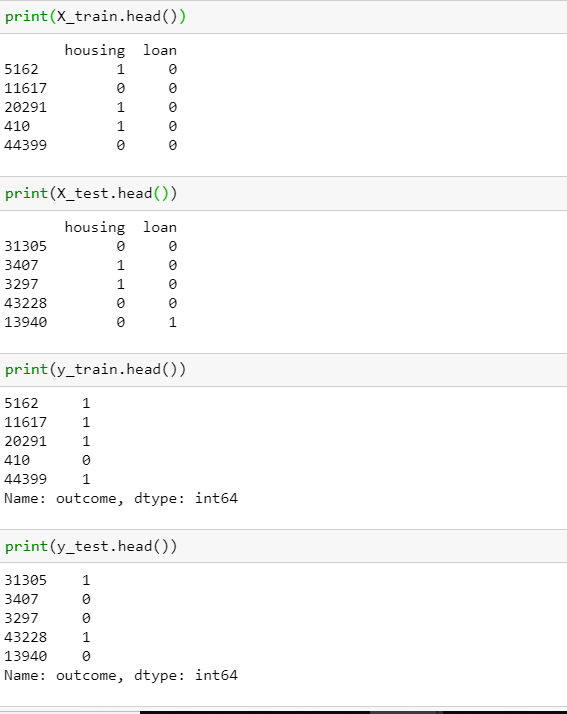
From the above testing, we can see that the best accuracy score (using balance, education, housing, loan) will be achieved when the k is 1 (we only define until 19 as the higher the k value , it will result in the inaccuracy of the training set, meaning it makes boundaries between classes less distinct.)

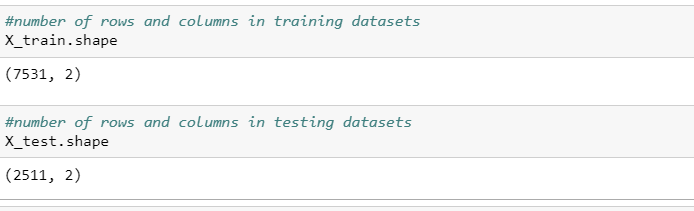
## 

## Using Housing, Loan with 75:25 training and testing dataset

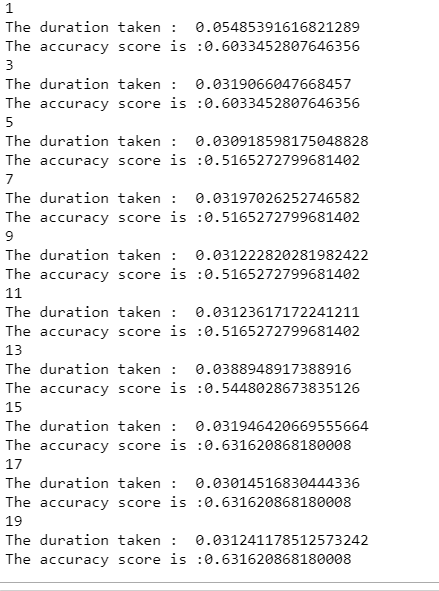
Let’s see how the accuracy score affected by splitting the training and testing dataset with 75% : 25% ratio



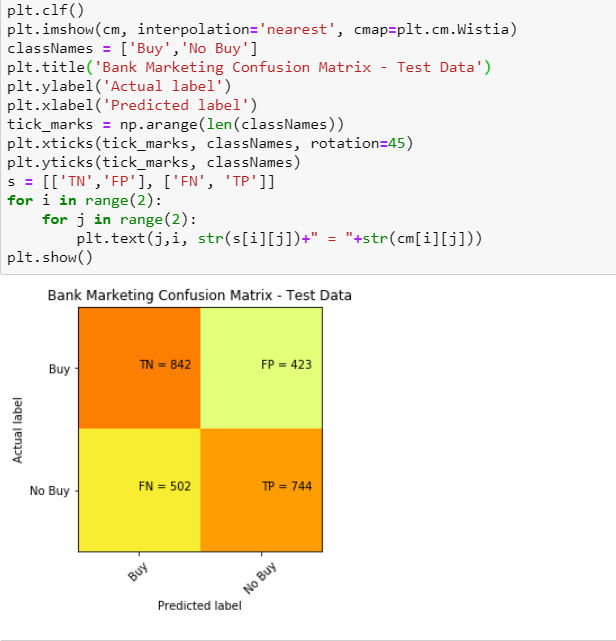












From the above testing, we can see that the accuracy score will change slightly different after using a different split ratio between training data and splitting data.

This implies that changing the splitting ratio will also affect the accuracy score.

## 

## Duration of training for K-NN

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy | Recall (TP/TP+FN) | Precision  (TP/TP+FP) | Elapsed Time (seconds) |
| Unnormalized Dataset with “Housing” and “Loan” | 0.885 | 0 | 0 | 0.453 |
| Normalized Dataset with “Housing” and “Loan” | 0.633 | 0.6006 | 0.6295 | 0.046 |
| Normalized Dataset with “Housing” and “Loan” (75:25 split ratio training and testing) | 0.6316 | 0.5971 | 0.6375 | 0.038 |
| Normalized Dataset with “Housing” and “Education” | 0.608 | 0.5628 | 0.6055 | 0.015 |
| Normalized Dataset with 4 Attributes (“Balance”, “Housing”, “Loan”, “Education”) | 0.558 | 0.5944 | 0.5364 | 0.004 (May be due to computer scheduling tasks differently, hence it is faster) |

## K-NN Conclusion

Depends on what our objective is, it is encouraged to use a balanced dataset when we train our model data, in order to come up with a fairer and reliable accuracy score and a shorter duration in training the model.

# 

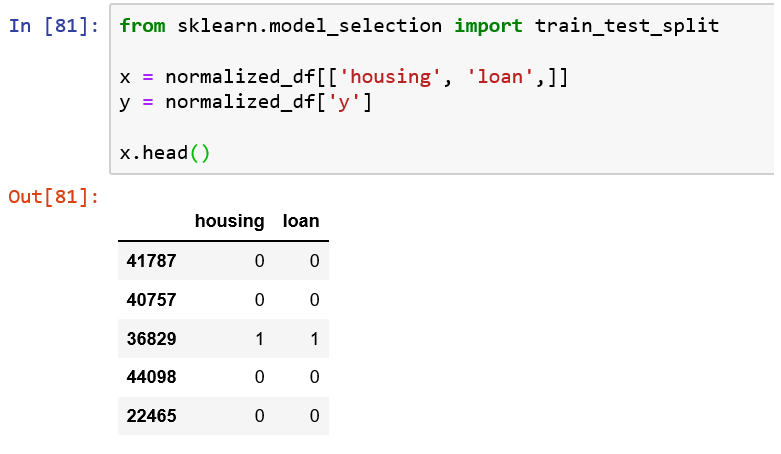
# **Supervised Learning (Support Vector Machine)**

## Using Data Set Without Normalization

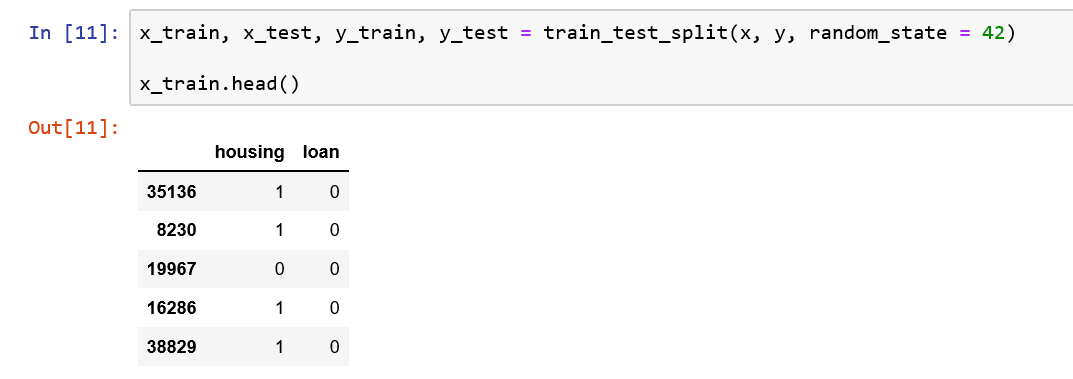
First, we would like to check the impact of unbalanced data on the predictive ability of the SVM module. The attributes used are “Housing” and “Loan”.

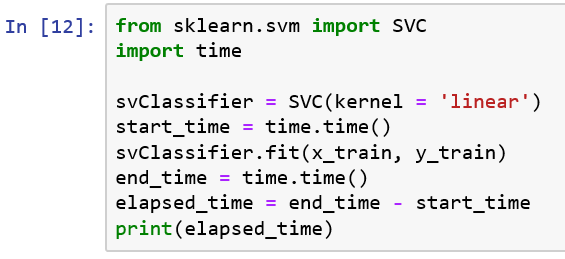
This section describes how a typical SVM run is conducted

We select the following attributes this way:

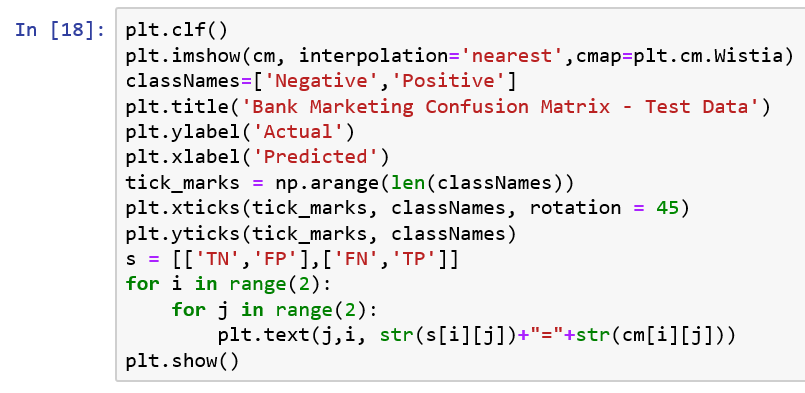


Also, we split test-data by their default percentages (test\_size set to 0.25)

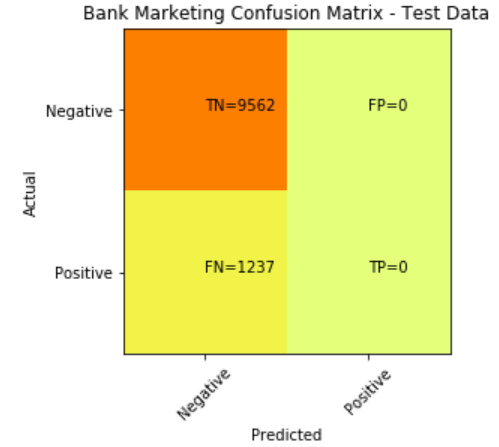


Then we run the following code to train the module. The time functions are included to give a rough measure of much time the algorithm takes.

The accuracy, true positive and true negative rates are then obtained through the confusion matrix

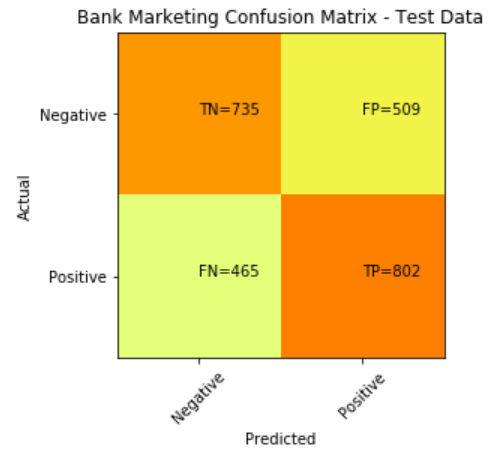


Henceforth, unless otherwise stated, the methodology carried out will be as described above.

First, we look at what happens if the data is used in its raw, unbalanced form. This is the confusion matrix that results from running the SVM :

From the result, while the accuracy is high (about 89%), we see that the model’s does not detect any positives. Hence, using the unbalanced data set is not useful at all for prediction, as it is simply predicting “Not buy” all the time. (See attachment Bank-SVM-Unbalanced)

## How data/feature/model engineering was performed to achieve a better outcome

Next, we compare this to the confusion matrix generated from the model, using a balanced, undersampled dataset (Methodology for undersampling mentioned earlier):

The accuracy has fallen to about 61%, but the recall rate has increased somewhat. Here, we can see that having a housing loan affects whether or not the person buys a time deposit. (See attachment Bank-SVM-Balanced)

SVM takes significantly more time when dealing with numeric values. When the attribute “campaign” is added to replace “Loan”(a binary categorical value), elapsed time increased. Even when using just “campaign”, the time taken is longer. (See attachment Bank-SVM-Balanced, section for campaign)

Using a linear kernel, it appears that once housing is added as an attribute, the accuracy and sensitivity (true positive and false positive rate) seems completely dependant on it. Changing the random state of the data does not change the problem (i.e. the accuracy does not change much once housing is added). (See attachment Bank-SVM-Balanced, followed by Bank-SVM-Balanced-DifferentRandomState for the comparison)

Changing the kernel for the chosen factors had no impact on the accuracy and other related factors when using the same random seed. It is speculated that since we use very little attributes, the difference does not make an impact. (Compare Bank-SVM-Balanced and Bank-SVM-Balanced-SigmoidKernel)

## Duration of training for Support Vector Machines

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy | Recall | Precision | Elapsed Time (Seconds) |
| Unbalanced Dataset using “Housing” and “Loan”, linear kernel | 0.88545 | 0 | 0 | 4.414 |
| Balanced Dataset with “Housing” and “Loan” (via undersampling), linear kernel | 0.61210 | 0.63299 | 0.61174 | 1.358 |
| Balanced Dataset with “Housing” and “Loan” (via undersampling), sigmoid kernel | 0.61210 | 0.63299 | 0.61174 | 0.693 (Might be some issue with warning generated.) |
| Balanced Dataset with “Housing”, linear kernel | 0.61210 | 0.63299 | 0.61174 | 1.385 |
| Balanced Dataset with “Housing”, “Campaign”, linear kernel | 0.61210 | 0.63299 | 0.61174 | 5.448 |
| Balanced Dataset with “Campaign”, linear kernel | 0.55077 | 0.87766 | 0.5333 | 3.745 |

## Support Vector Machine Conclusion

SVM does not appear to be a good model to be used with this classification method. Higher accuracies are found using other methods

SVM is slowed down considerably when numerical(instead of categorical) data in involved. It may not perform well with large datasets

Changing the kernel type (in this case, sigmoid) does not have much impact on the accuracy.

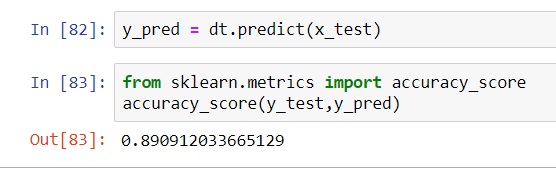
While running SVM in a column with both positive and negative numbers, the algorithm seems to hang. This was because SVM could not complete operations when the attribute pdays was used. Could be due to the fact that since pdays has negative numbers, it could also prevent the SVM from running properly (speculated reason: lack of convergence).

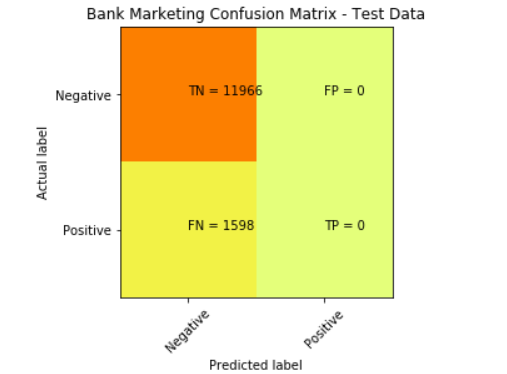
# **Supervised Learning (Decision Tree/Random Forest Report)**

## Using dataset without Normalization

We would like to check the raw data which is not normalized yet.



We got accuracy of 89% because of unbalanced data and we can’t rely on the results.



Accuracy = (0+11966)/ 13564= 0.8821

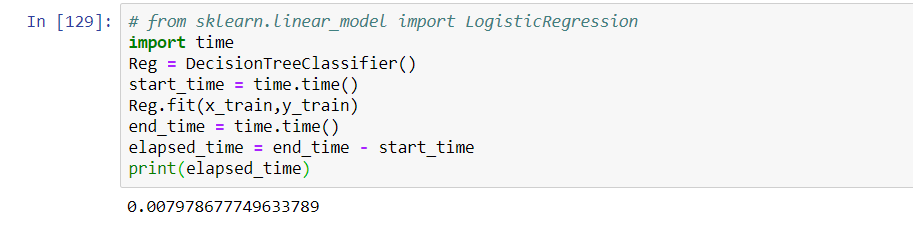
Recall = 0/(0+1598) = 0

Out of all the positive classes, none was predicted correctly.

Precision = 0/(0+0) = 0

Out of all the classes (Both positive and negative), none was predicted correctly.

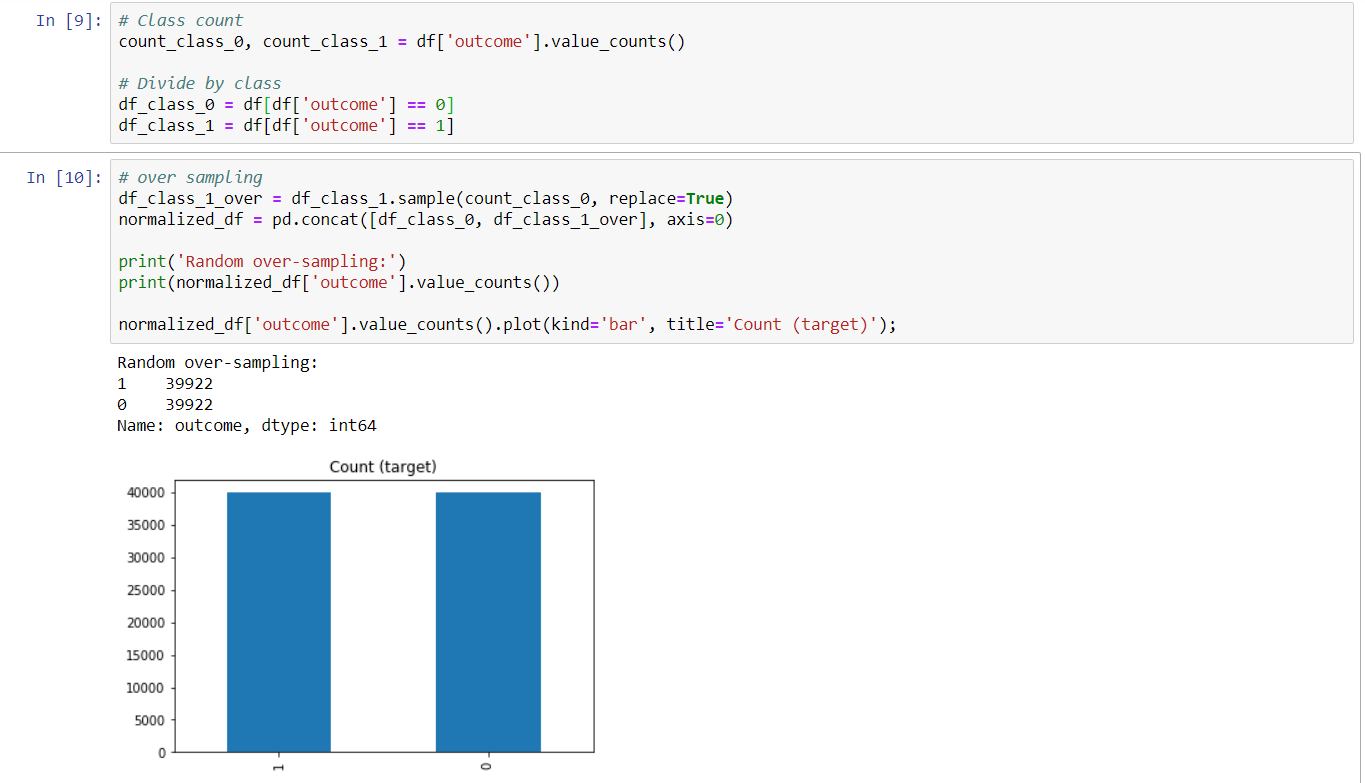
As seen in the above results, the accuracy score is very high at 89%, however, we do not think that this is accurate and thus we ran the confusion matrix. True enough, None of both the positive and negative classes were predicted correctly which does not make it reliable for usage.



## 

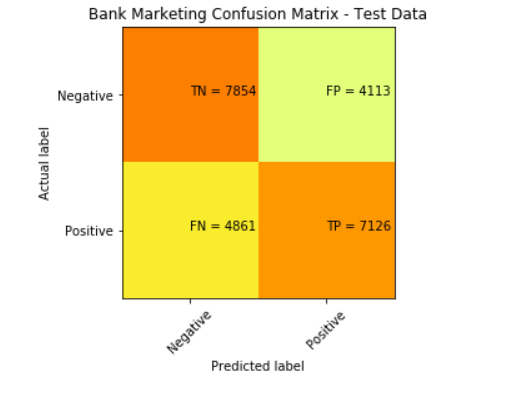
## How data/feature/model engineering was performed to achieve a better outcome

Over Sampling:



We use the same features as the last one. We got accuracy of 62%.





Accuracy = (7854+7126)/ 23954= 0.625

Accuracy score matches sklearn.metrics which confirms the scoring.

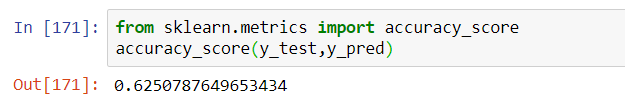
Recall = 7126/11987 = 0.594

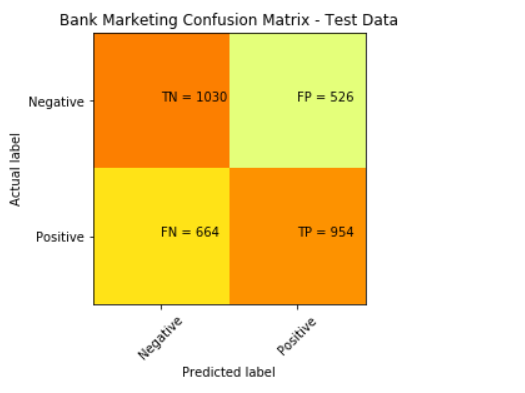
Out of all the positive classes, 59% was predicted correctly.

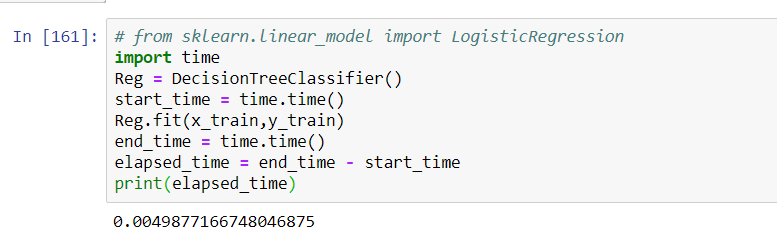
Precision = 7126/(7854+7126)= 0.476

Out of all the classes (Both positive and negative), 47% was predicted correctly.

Undersampling:



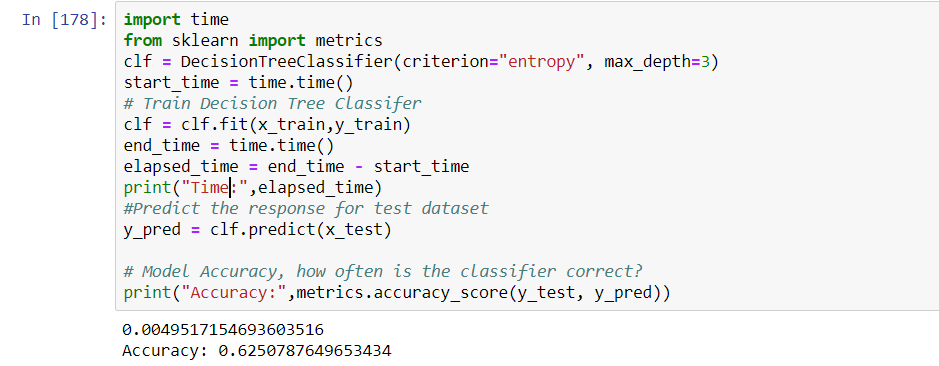


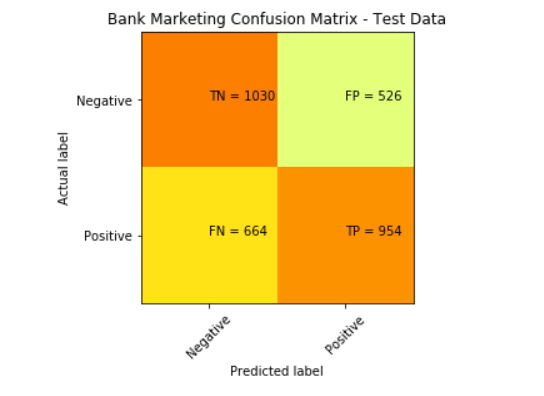


Decision tree is easy to interpret compared to other methods. Decision trees are biased to unbalanced data, that’s why we balance out the dataset before creating decision tree.

## Optimizing the decision tree performance

The last resultant tree is unpruned. In Scikit-learn, optimization of decision tree classifier performed by only pre-pruning. We will use entropy to select attributes.





Accuracy = (1954)/ 3174= 0.616

Accuracy score matches sklearn.metrics which confirms the scoring.

Recall = 954/(664+954) = 0.590

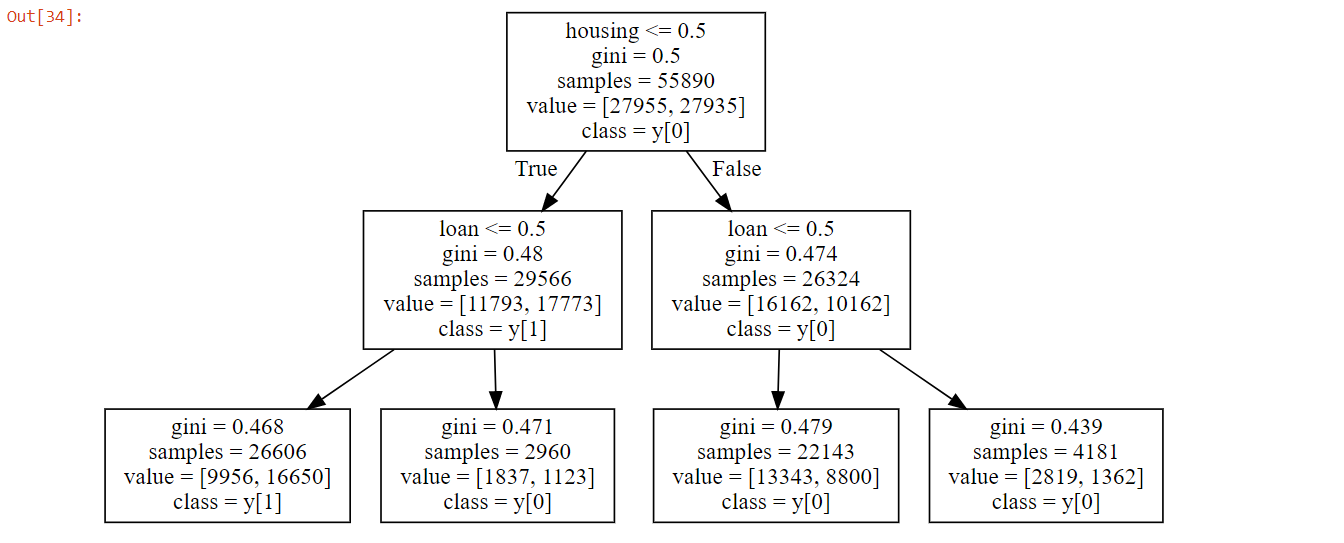
Out of all the positive classes, 59% was predicted correctly.

Precision = 954/(954+526)= 0.645

Out of all the classes (Both positive and negative), 64% was predicted accurately.

As shown, the optimization of the decision tree has improved our precision results quite a bit. Before optimization, the score was at 47% vs 64% now.

The image below is the visualization of decision tree.



## Duration of training for Tree Diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy | Recall | Precision | Elapsed Time (Seconds) |
| Unbalanced Dataset using “Housing” and “Loan” | 89% | 0 | 0 | 0.008 |
| Normalized Dataset with “Housing” and “Loan” (Undersampling) | 62% | 0.590 | 0.645 | 0.005 |
| Normalized Dataset with “Housing” and “Loan” (Oversampling) | 62% | 0.594 | 0.476 | 0.005 |
| Optimizing the tree with entropy | 61% | 0.590 | 0.645 | 0.005 |

## Decision Tree Conclusion

As decision tree is sensitive to noisy data, it overfit the noisy data. Although decision tree is easy to interpret and visualize as compared to other methods, small variance in data might result in different decision tree. Decision trees are biased to unbalanced data, that’s why we balance out the dataset before creating decision tree.

# Overall Conclusion

From the above applied supervised learning methods, a key takeaway is that we will need to ensure our data set is to be normalized/balanced before usage. This allows the generated result to be less skewed towards the majority of negatives in our case. The rate of precision and recall also drastically improves, from 0% to approximately 60%.

This also shows us that an accuracy score alone does not necessarily contribute to whether the generated result can be used straight off the bat. We will also need to take into account other factors such as if the model predicted True Positives, True Negatives, False Positives and False Negatives correctly.

Below table shows a summary of the same data set across the supervised learning models used (Using a normalized data set with “Housing” and “Loan”):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Logistic Regression | K-N Neighbor  (k = 13) | Support Vector Machines | Decision Tree |
| Accuracy | 0.6126 | 0.6316 | 0.61210 | 0.616 |
| Recall | 0.5747 | 0.5971 | 0.63299 | 0.590 |
| Precision | 0.6400 | 0.6375 | 0.61174 | 0.645 |

*The highest value in each category of cell is highlighted for easy viewing.*

The overall conclusion is that the different supervised learning models have output values that are relatively close in range to one another. In addition to this, the observation that all accuracy values are relatively low shows that the input variables in the dataset have at best, a weak predictive value.

# 

# 

# **Unsupervised Learning**

# Introduction

Dataset Source: <https://archive.ics.uci.edu/ml/datasets/Wine+Quality>

## Problem Statement

Wine quality is dependent on a lot of factors which include the alcohol content in the wine, the pH values, acidity, presence of sulphates to name a few. The several chemical ingredients define the taste, smell and potency of the wine.We try group together different categories of wine based on the percentage of chemical ingredients present in it.

## Simple Data Dictionary of training models (columns involved in training), incl source of dataset

|  |  |
| --- | --- |
| Input Variables | |
| Fixed Acidity | Volatile Acidity |
| Citric Acid | Residual Sugar |
| Chlorides | Free Sulfur Dioxide |
| Total Sulfur Dioxide | Density |
| pH | Sulphates |
| Alcohol |  |

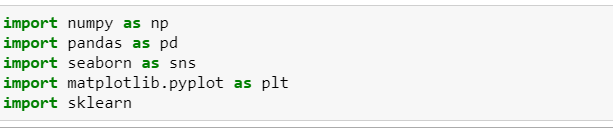
Output variable: Quality (score between 0 and 10)

## No of records and columns in dataset

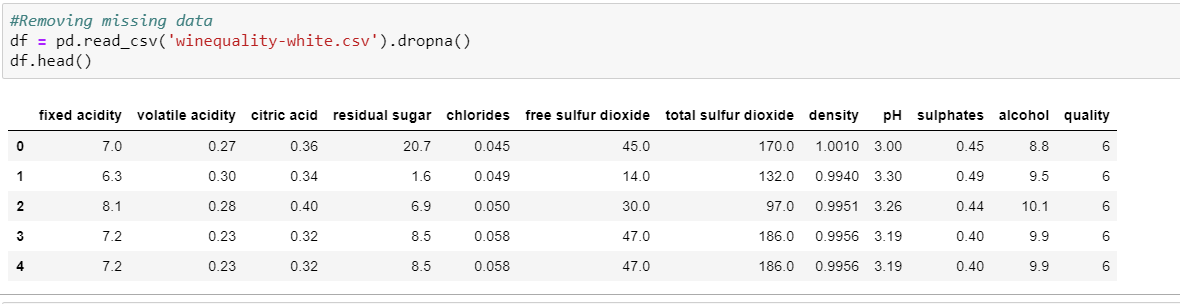
No. of columns in dataset: 11 attributes and 1 outcome

No. of records in dataset: 4899 rows (raw data), 815 rows (after undersampling)

# Feature Engineering (Data Balancing and Data Cleansing)



We have chosen our dataset and read it using the panda library







From the above data, we can see that the quality of ‘6’ is proportionally larger as compared to the rest.

Hence, we are doing undersampling in order to prevent overfitting. We can do it by balancing all the quality outcome to be around the same.

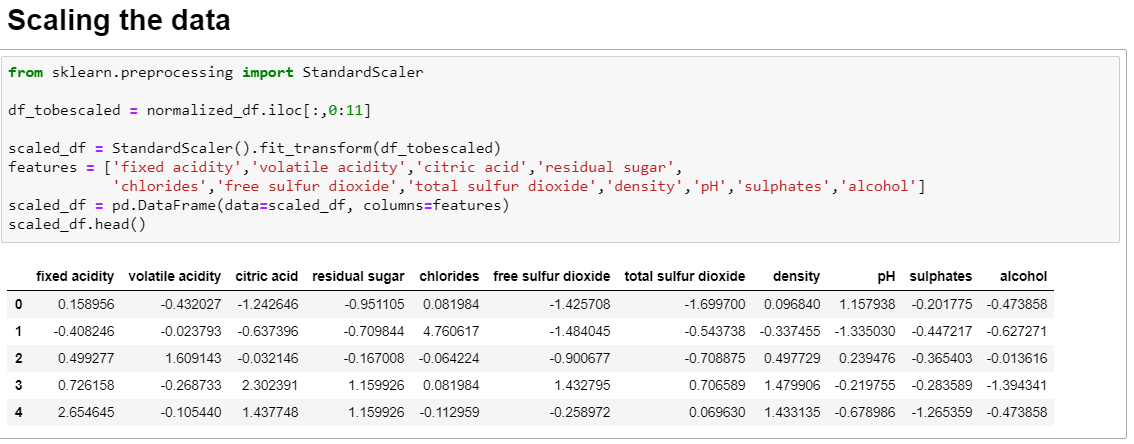
Before that, we will do some mapping to remove the quality of ‘3’ and ‘9’ as it only generates small observations in the dataset which we can ignore the both of them..



After the removal, we proceed to do the balancing of the outcome by balancing according to the total number of quality ‘4’, which is 163.

## Balancing the dataset







For clustering, it is better to do standardization of all the datasets so they will have the same distribution. It is important to scale the features to a range which is centered around zero. This is done so that the variance of the features are in the same range. If a feature’s variance is orders of magnitude more than the variance of other features, that particular feature might dominate other features in the dataset, which is not something we want happening in our model.

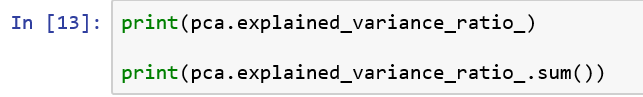
To sum up, we have done some data engineering by :

1. Firstly removing the null or empty dataset.
2. Map the outcome for quality ‘4’,’5’,’6’,’7’,’8’, in order to remove quality ‘3’ and ‘9’ as it was deemed not significant to the dataset.
3. Last but not least, we undersample the data by looking at quality ‘4’ and balance the rest of the quality(‘5’,’6’,’7’,’8’) to have the same total number as quality ’4’ (163).
4. Scaling/standardizing the dataset.

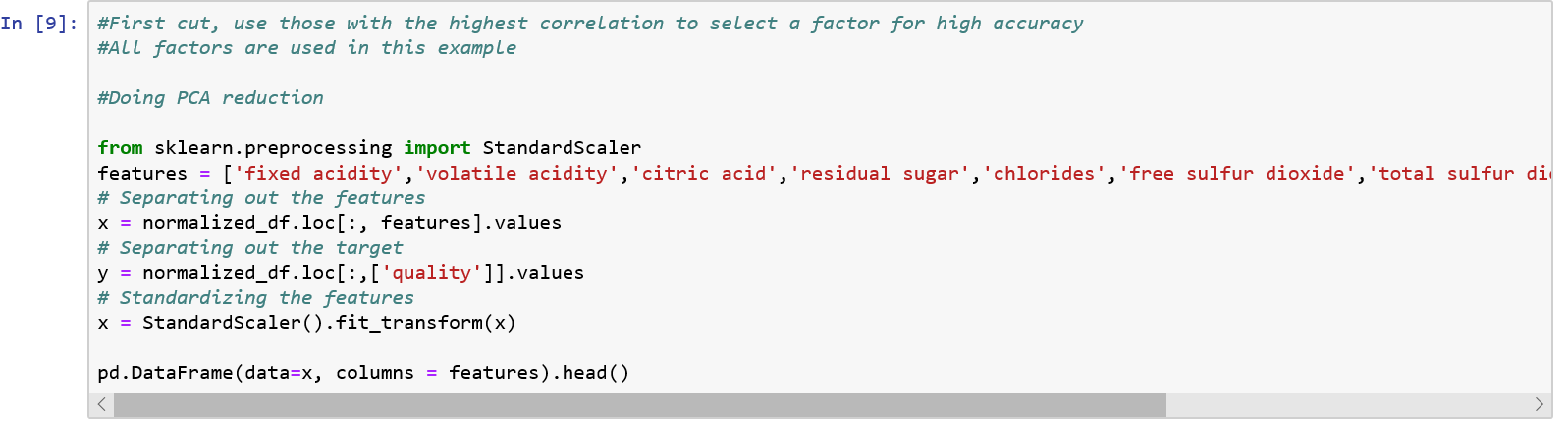
# **PCA and LDA**

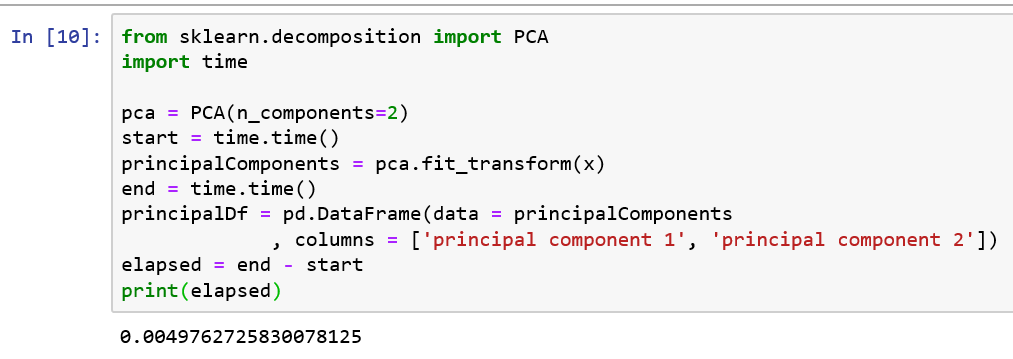
Before running the PCA and LDA, the data has been appropriately balanced and scaled (We have seen earlier that unbalanced outputs do not allow us to make good estimations)

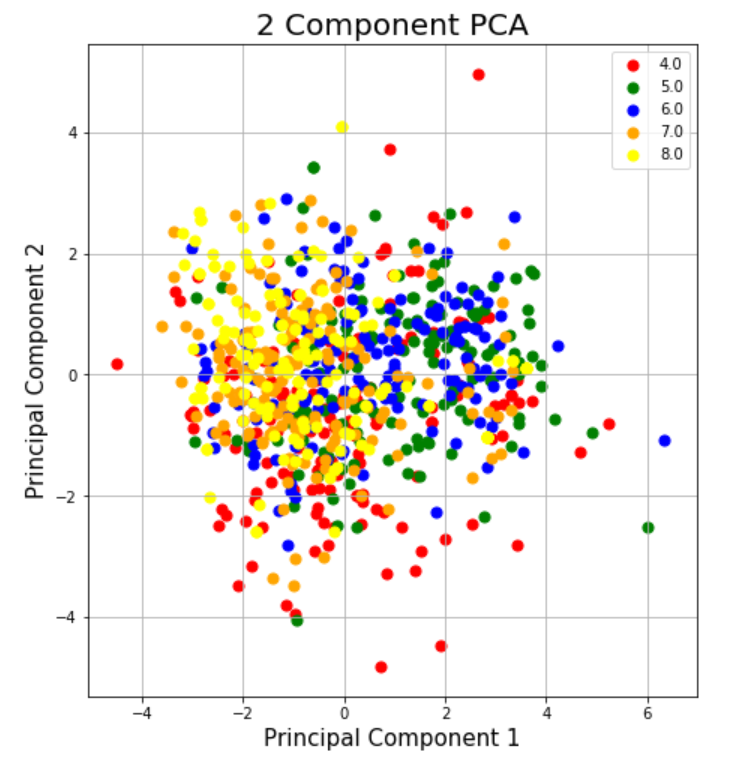
For all cases, estimated variance is calculated with the following code:



First we do a PCA using all factors except the y-variable.





This is the visualisation:

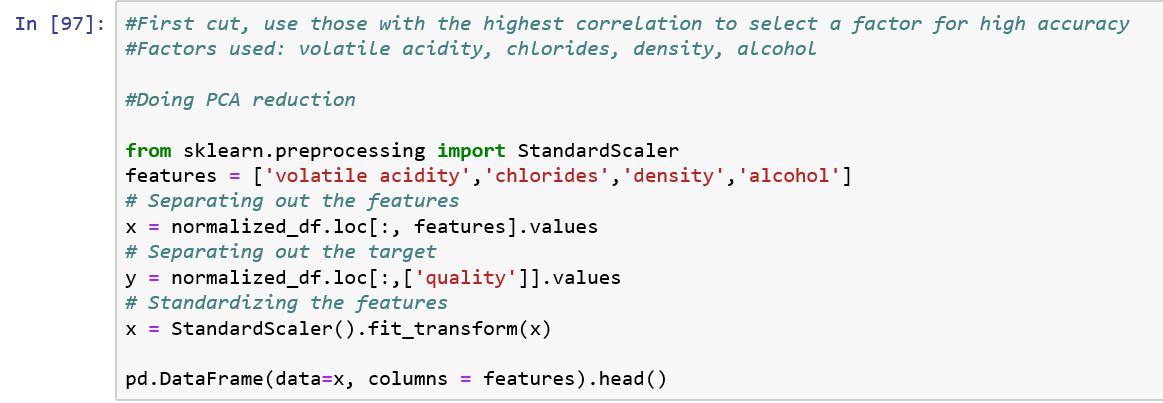
The explained variance ratio is very low (about 0.44), suggesting that there is no good way of projecting the data with just two components.

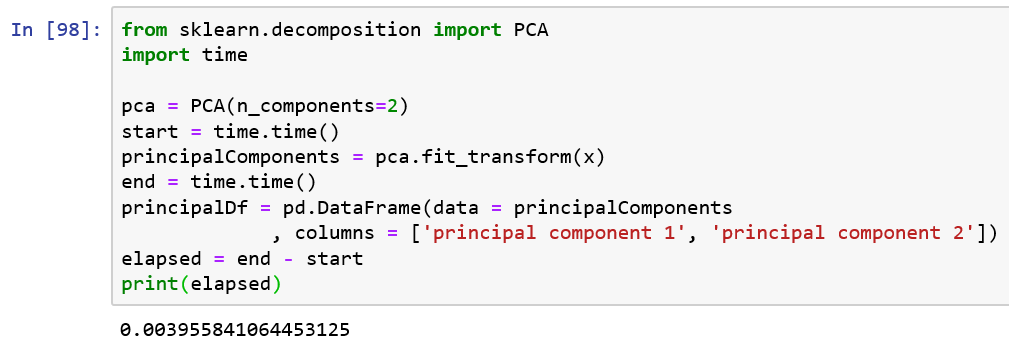
Hence we try projecting it 5 components (about half the number of input variables) instead.

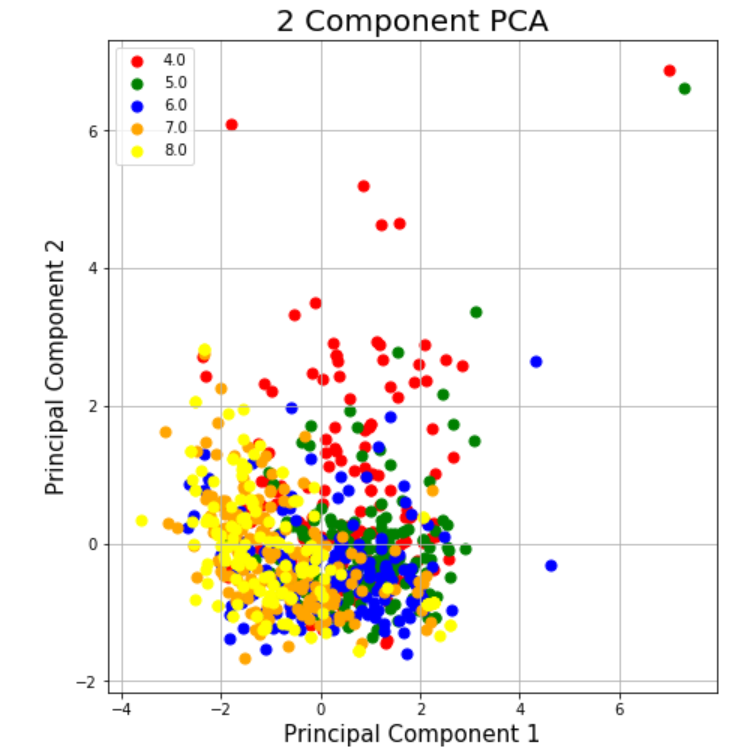


The truncated text means principal component 5

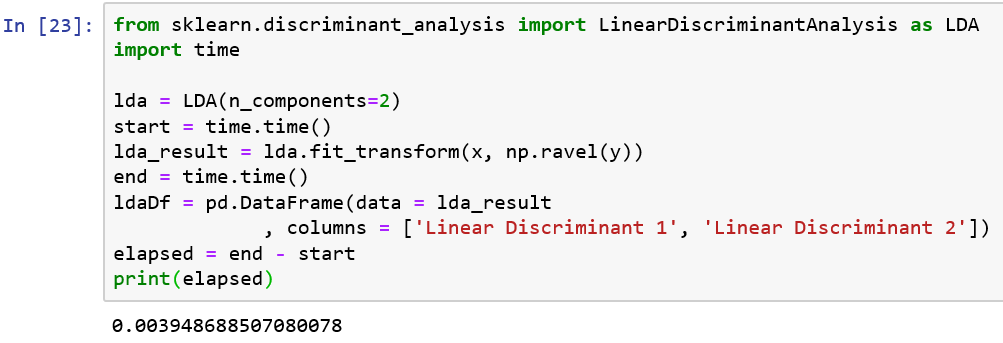
The explained variance ratio has now improved by adding more components (about 0.75). However, with more components, it is likely that further work using the PCA model will require more time.

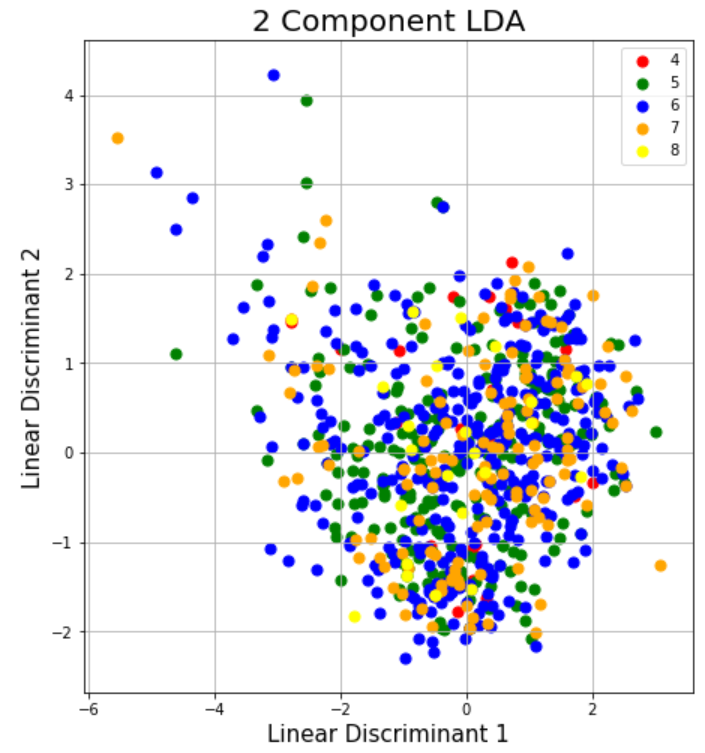
Also tried using PCA on a reduced feature set. The reduced feature set is a set of four factors, that are not correlated with each other.



Visualisation of the reduced feature PCA

The explained variance ratio has improved, probably because we have removed information from the sample (about 0.78)

2 component LDA has also been carried out on the data set with all test features

The visualisation of the LDA

The explained variance ratio is very high (about 0.97). More work needs to be done why the LDA captures more explained variance than PCA

## Duration of training for PCA and LDA

|  |  |  |
| --- | --- | --- |
|  | Explained Variance ratio | Time Elapsed (seconds) |
| PCA with all factors, 2 components | 0.44504 | 0.0049762 |
| PCA with all factors, 5 components | 0.75206 | 0.0029907 |
| PCA with reduced feature set, 2 components | 0.78219 | 0.0039558 |
| LDA with all factors, 2 components | 0.97868 | 0.0039486 |

\*All timings were very brief, so big differences in timing may occur

## PCA and LDA Conclusion

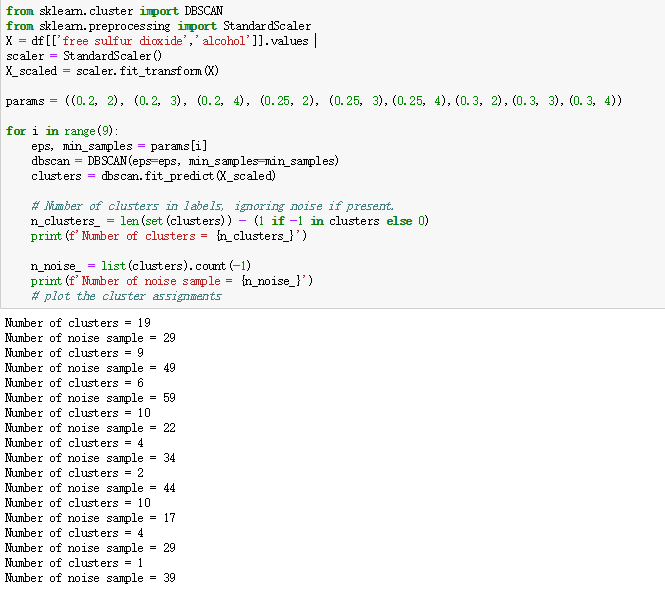
In conclusion, a PCA transformation with 2 components on all features does not capture all the information present in the data sample. Increasing the number of components used increases the explained variance ratio, but this would impact performance times.

A possibly interesting finding is that LDA gives a much better variance ratio the PCA when selecting for the same components. Given more time, this can be investigated further.

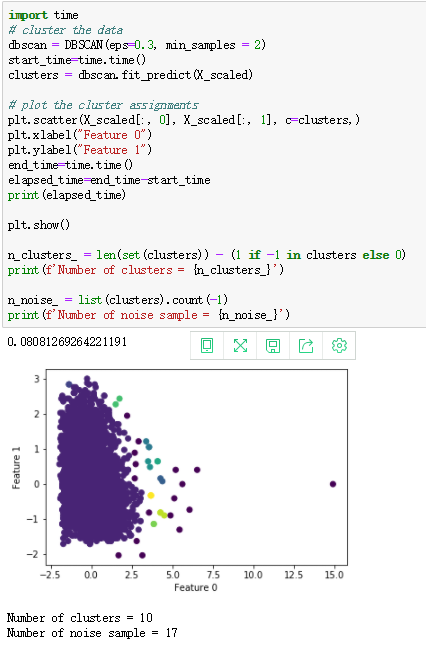
# DBSCAN

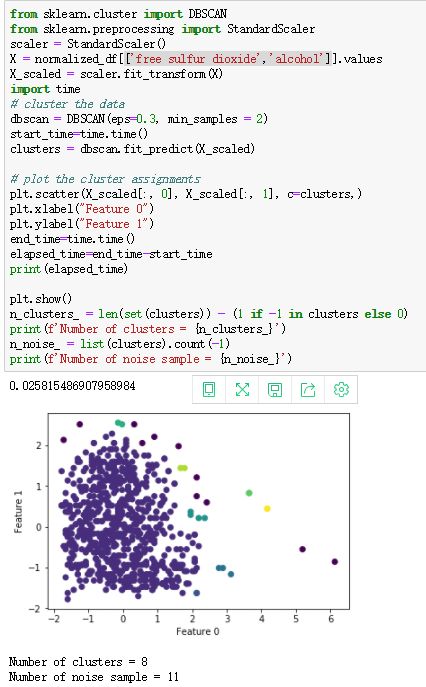
First, let's pick two features at random ‘free sulfur dioxide' and 'alcohol’.To get the appropriate eps and min\_samples we write a for loop.

## Using Data Set Without Normalization



As we can see eps=0.3 and min\_samples=2 has more clusters(10) and less noise(17)

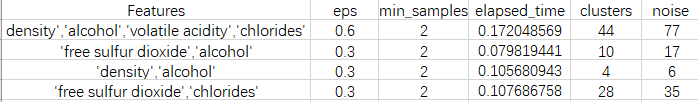




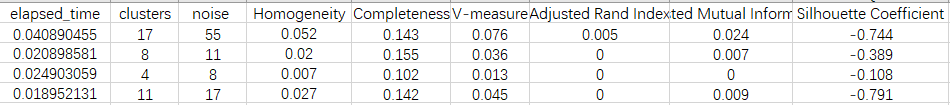
After we get normalized data and run DBSCAN again (with same eps and min\_samples) We can find that clusters and noise reduced and there is a large decentralized cluster. Clusters are easier to identify than before data balancing, indicating that data balancing has had some effect.

## Duration of training for Tree Diagram

Before data balancing:



After data balancing:



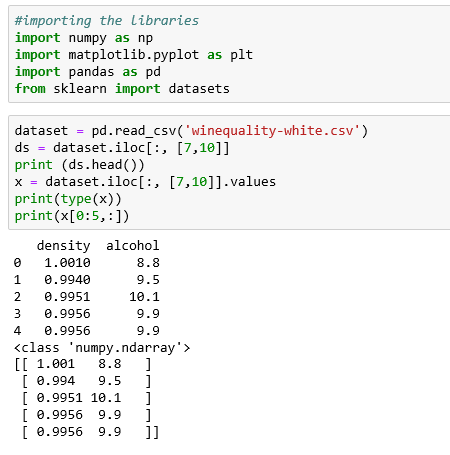
## DB SCAN Conclusion

From this table, we can see that, if there are more features, the appropriate EPS will be larger, the computing time will be increased, clusters and noises will be larger, but appropriate min\_samples are generally the same. Data Balancing can reduce noise most of the time, but sometimes noise will increase a little. According to the data of EVALUATION METRICS, we can see that It is considered that DBSCAN is not suitable for this dataset.

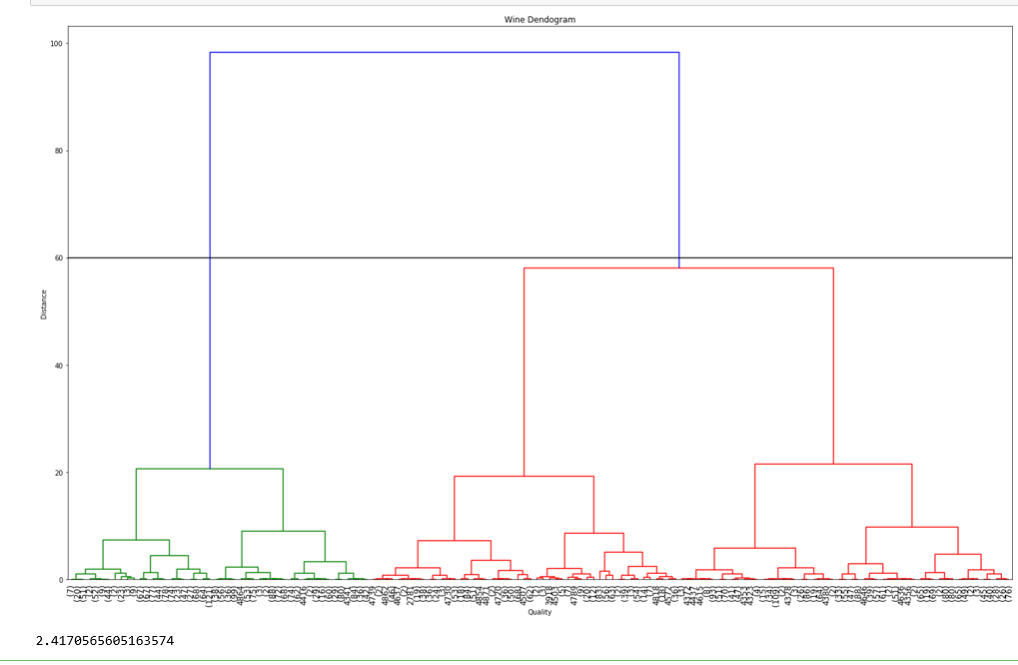
# Agglomerative Clustering

Our dataset has eleven attributes. However, to view the result in two-dimensional feature space, we will choose two out of the eleven attributes. We have decided to use column 7 and column 10 which are ‘density’ and ‘alcohol’ respectively. The reason we chose these 2 attributes is because they have the highest correlation to quality of wine which is shown in the correlation table above.

## Using Data Set Without Normalization



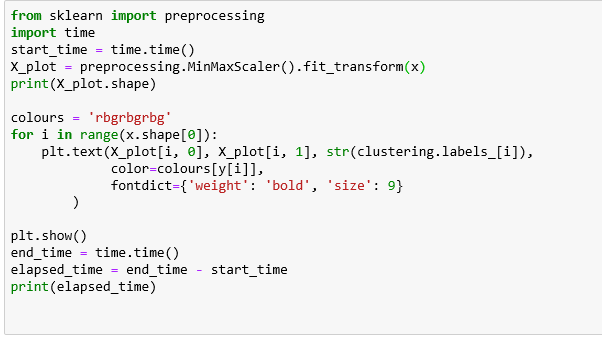


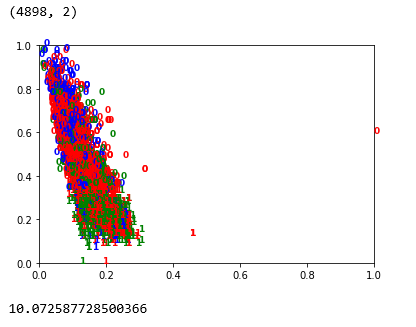


We drew a horizontal line that passes through the longest distance without a horizontal line therefore we got 2 clusters.

Knowing the number of clusters from out dataset, we can group the data points into these 2 clusters.







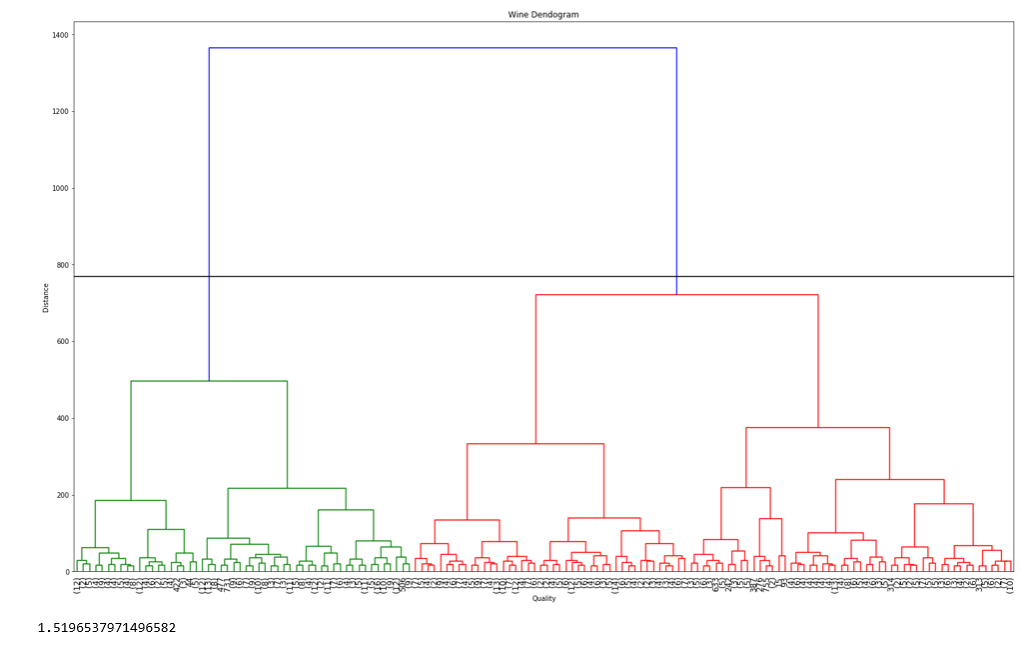
Since we have 2 clusters, we have 2 labels in the output of 0 to 2. The diagram above also show how our data has been clustered. From the looks of it the data are all clustered together and makes it quite difficult to be able to extract significant information out of it. This is one of the major issues with Hierarchical Clustering, it does not work very well with huge data.

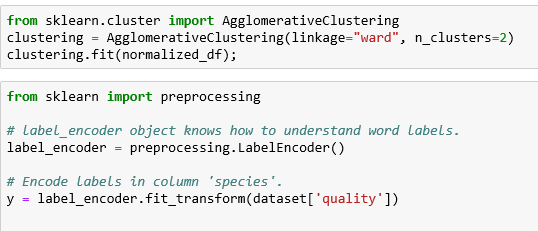
## 

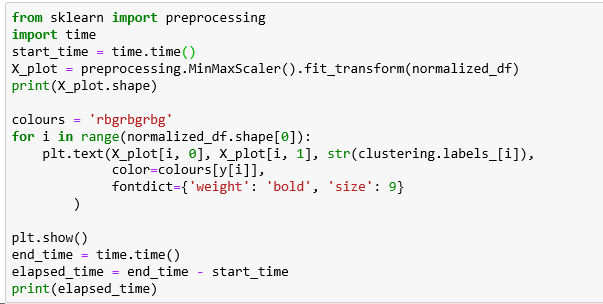
## How data/feature/model engineering was performed to achieve a better outcome

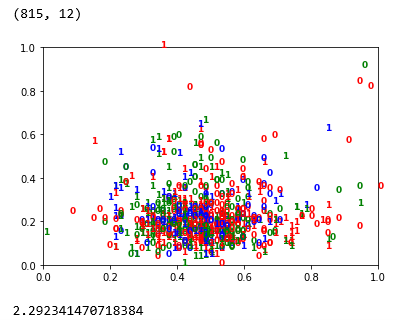
We then balance and cleanse the data in order to reduce the amount of data in order to hopefully get a slightly more accurate cluster. Steps applied after balancing and cleansing the data are the same as what we had done before balancing and cleansing of data.

****

****

****

****

****

## **Duration of training for each model**

|  |  |  |
| --- | --- | --- |
|  | Time taken before | Time taken after |
| Dendrogram | 2.417 seconds | 1.5196 seconds |
| Cluster diagram | 10.0725 seconds | 2.29234 seconds |

## Agglomerative Clustering Conclusion

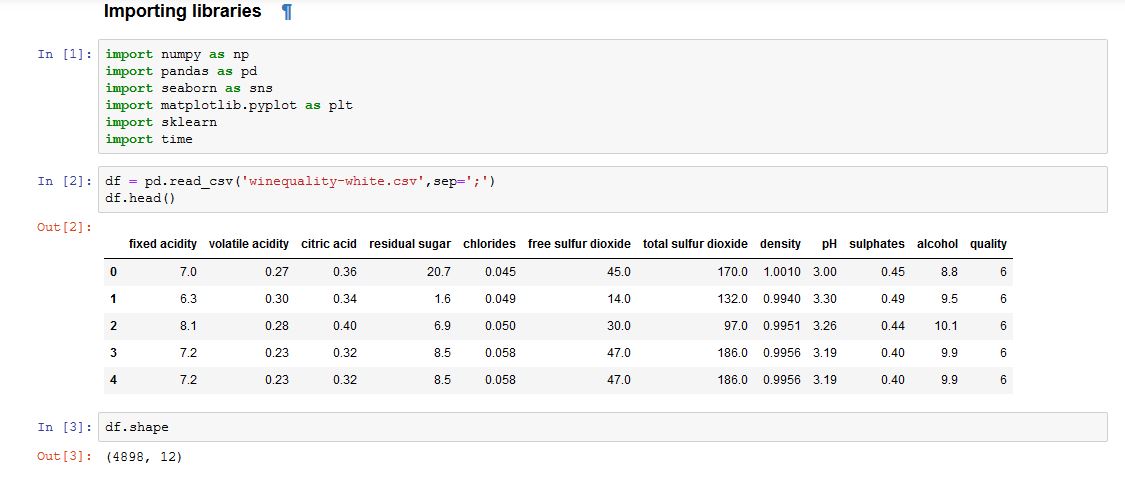
The output after balancing and cleansing the data seemed to have improved slightly as there is a wider spread of spread as compared to before balancing and cleansing the data. The time taken to build the models also has been reduced. However, due to the data still being quite huge, there is still a big chunk of data being clustered together.

# K Means Clustering Analysis

Introduction

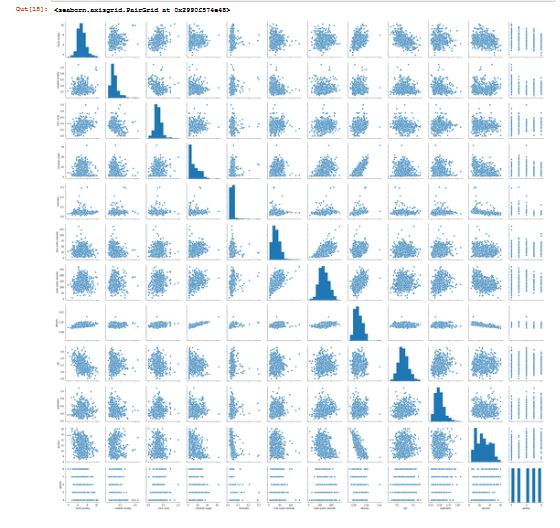
Wine quality is dependent on a lot of factors which include the alcohol content in white wine, the pH values, acidity, presence of sulphates to name a few. The several chemical ingredients define the taste, smell and potency of the wine.We try group together different categories of wine based on the percentage of chemical ingredients present in it.

Importing libraries





## Visualizing the pair plots

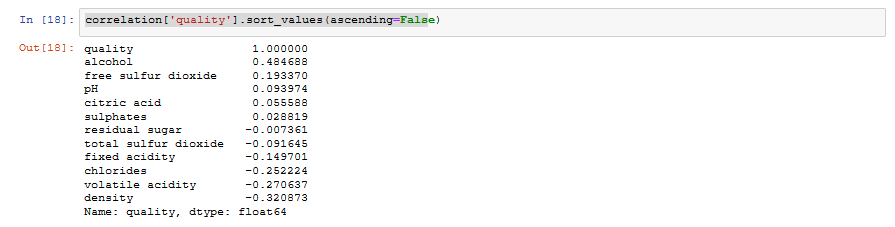


## Correlation Matrix

## Correlation Heat Map

## Checking positive correlation

We now check the relation of quality of the wine with other features in the dataset. We try to understand what all factors affect the quality of the wine in a positive way and by how much.

****

In the above fragment of code, a conclusion can be made that the attributes 'alcohol', 'free sulphur dioxide', 'pH' and 'citric acid' have maximum correlation with the 'quality' attribute.

Hence, these four attributes will be used for pattern and correlation exploration. It can be noted clearly that the alcohol percentage has the highest correlation with the wine quality.

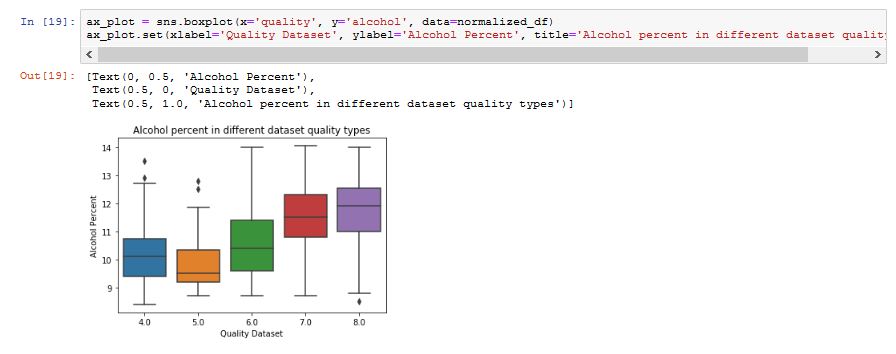
## Selecting variables which we think will help us form good clusters

As a first step, we have decided on using 4 attributes: pH, citric acid, alcohol and free sulfur dioxide.

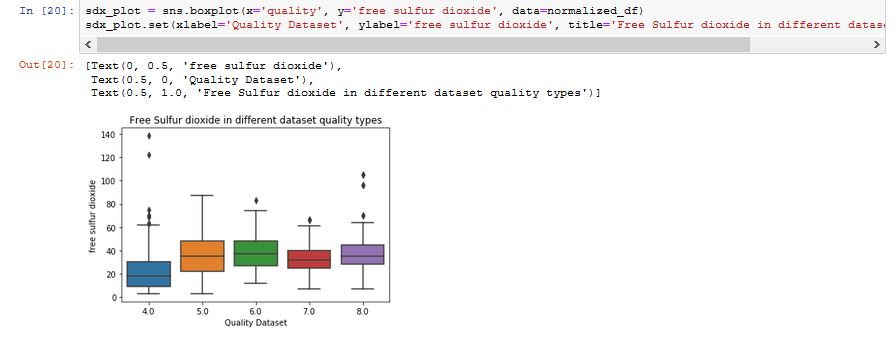
## Box plotting

Using box-plots will help us understand the relative-ness of the quality of wine with respect to each of the chosen attributes.

### 1. Alcohol percent vs Wine Quality

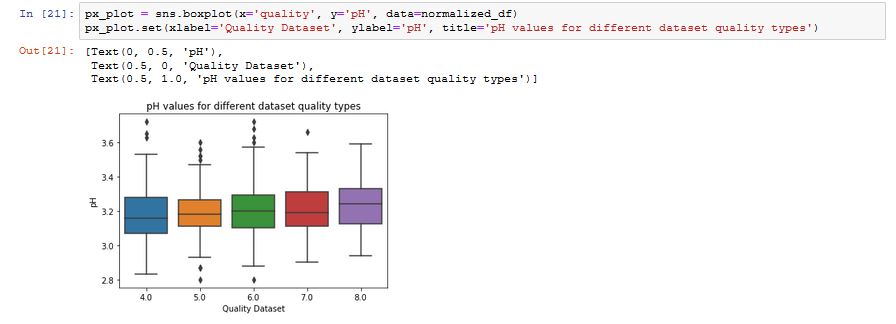


### 2. Free Sulphur Dioxide percent vs Wine Quality



### 

### 3. pH vs Wine Quality



### 4. Citric Acid vs Wine Quality

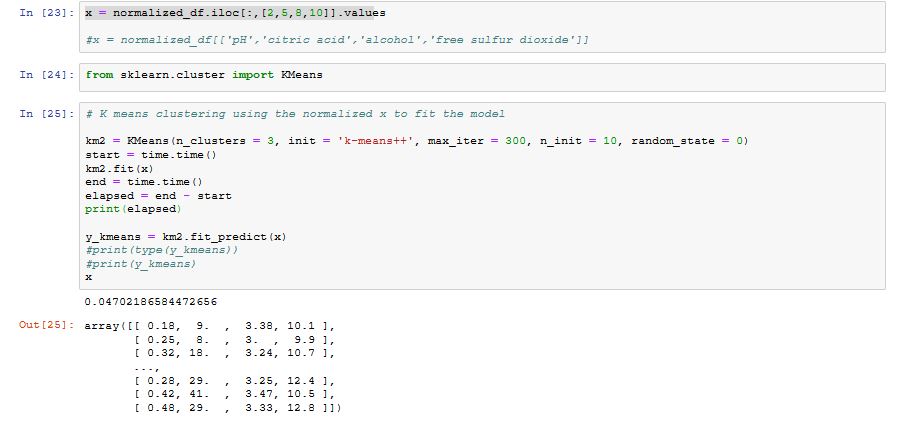
### **Deduction from the box plots**

By observing the box plots we can clearly observe the effect of the attribute in the y-axis relative the attribute in the x-axis, which is the wine quality. We can notice the variation in the box plot is decreasing.

The variation in the box plot can be seen as high when we compare wine quality with alcohol content in it.

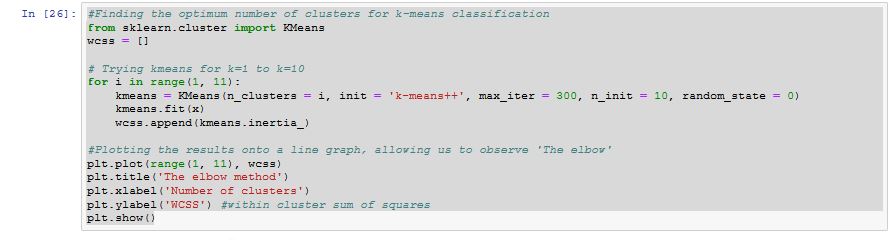
The variation can be seen as gradually decreasing as we move on down to the attributes which are less strongly correlated to the quality of wine.

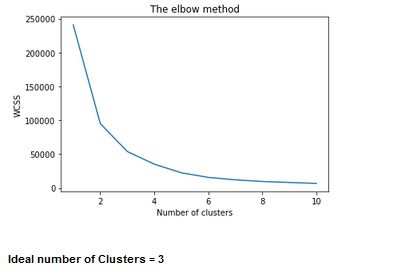
## K-Means Clustering Algorithm



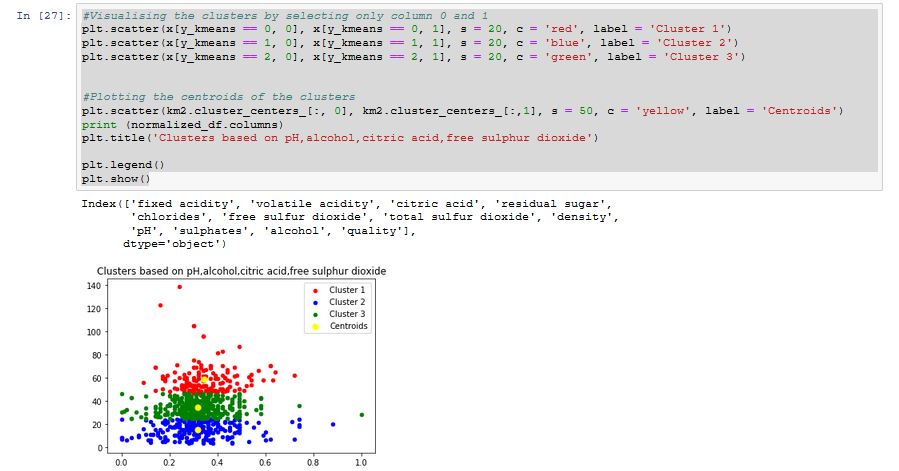
## Finding the ideal number of Clusters

WCSS indicates how far are the points from its centroids. With more centroids we expect that WCSS will be lower.





## Visualizing the clusters

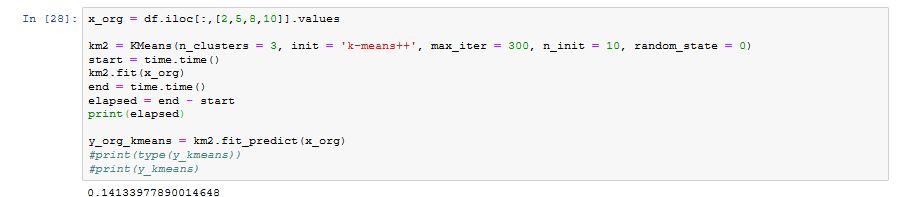


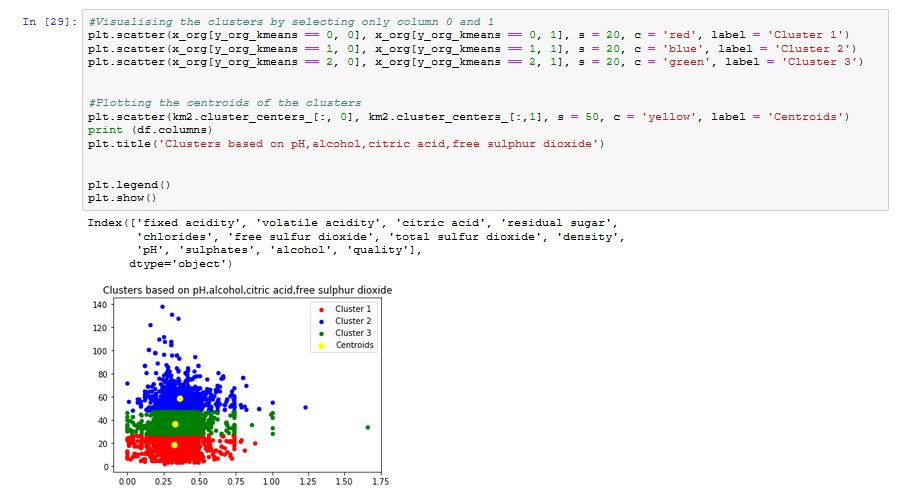
We can observe a clear clustering pattern between the different groups based on the attributes chosen. It can be deduced from the above the figure that quality of wine can be categorised into separate groups by affecting the prominent features which govern it. In this case, we observe factors like alcohol, pH, citric acid, free sulphur dioxide can be used to adjust the quality of wine produced.

**Time taken to build the model = 0.059465885162353516 seconds**

### 

## K Means Clustering using the original unbalanced dataset

****



We observe the same pattern in clustering using the original unbalanced dataset as the balanced one with a clear categorization of the dataset based on the chosen features

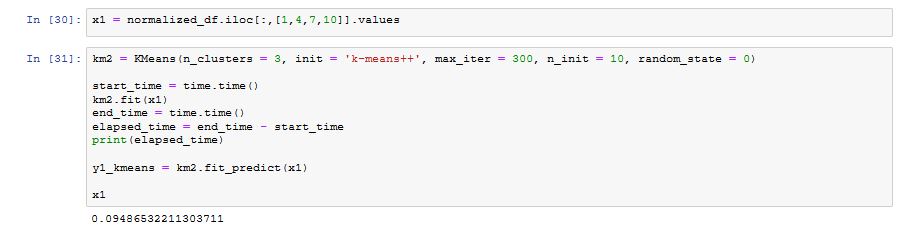
**Time taken to build the model = 0.10280847549438477 seconds**

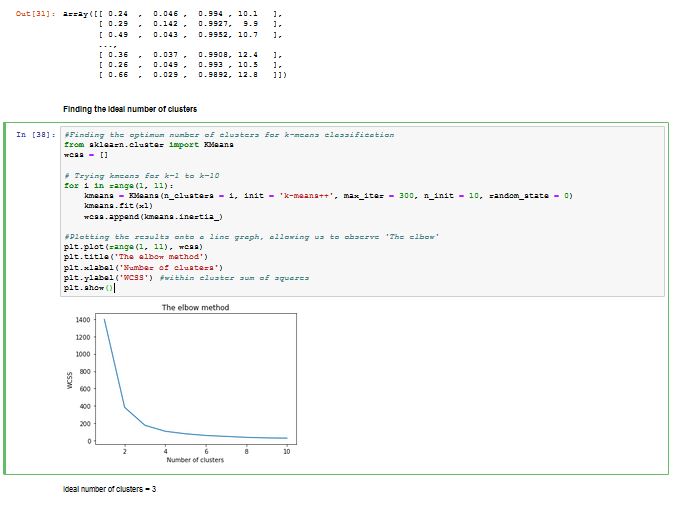
## 

## K Means Clustering using negatively correlated Attributes

Attributes chosen - Volatile Acidity, Chlorides, Density, Alcohol

Here we chose three strongly negatively correlated attributes 'Volatile Acidity', 'Chlorides', 'Density'. We also use 'Alcohol' even though its positively correlated as we feel it as an essential feature in determining the quality of the wine





By using the elbow method from the figure above, we can clearly identify the optimal number of clusters as 3 for our data analysis.

## Visualizing the Clusters

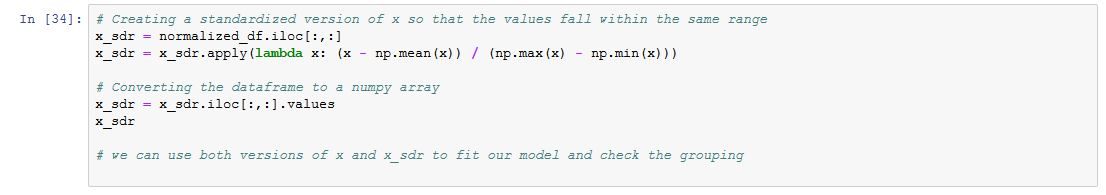
****

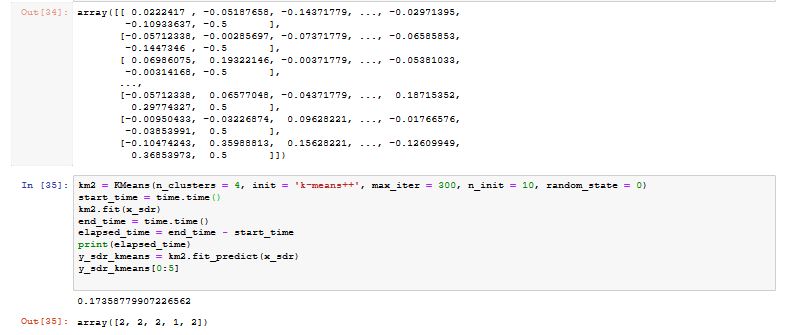
**Observations**

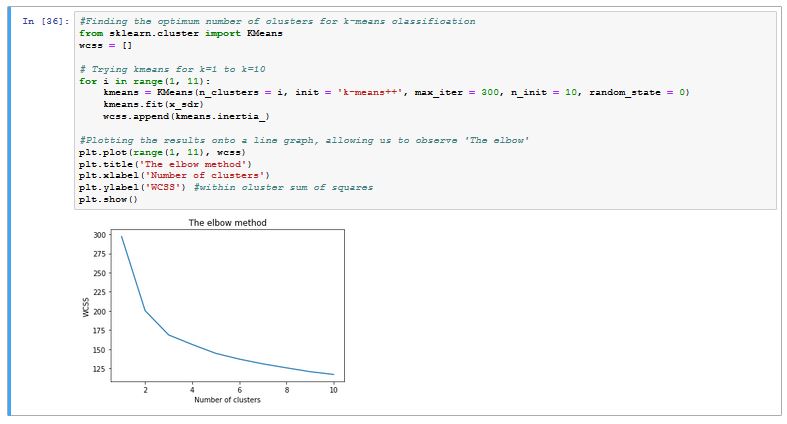
The clusters formed clearly indicate the grouping based on the features chosen. We also notice the presence of a few outliers in the various cluster groups.

**Time taken to build the model = 0.05110764503479004 seconds**

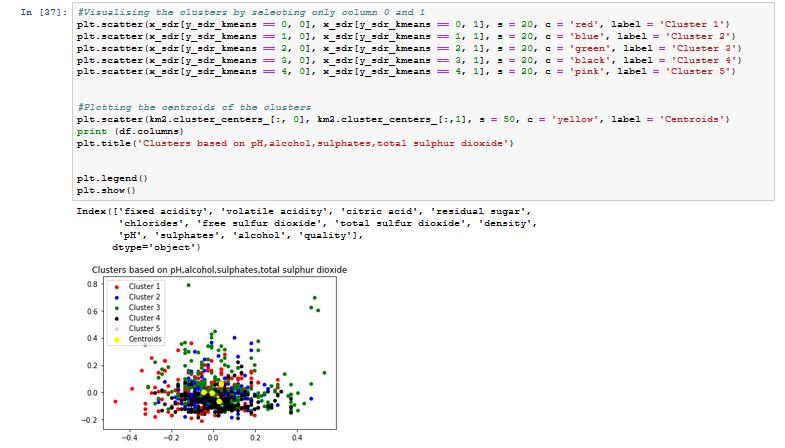
## Kmeans clustering using the standardized values (x\_sdr) with all features included







We observe there is a marginal shift in the elbow when we use a standardized version of x. The optimal number of clusters can be identified as 5 now.



By using the standardized dataset with all the features we observe the clustering pattern is overlapping and dense and there is no clear demarcation between the clusters

**Time taken to build the model = 0.10366106033325195 seconds**

## Duration of training for k-Means Analysis

|  |  |  |
| --- | --- | --- |
| **Features Used** | **Time taken** | **Dataset** |
| Chlorides,Density,Volatile Acidity,Alcohol | **0.0594** | Balanced |
| Chlorides,Density,Volatile Acidity,Alcohol | **0.1028** | Unbalanced |
| Citric Acid,pH,Free Sulfur Dioxide,Alcohol | **0.0511** | Balanced |
| All Features | **0.1036** | Balanced and Standardized |

**Balanced Dataset:** All attributes in the dataset normalized to contain the same number of observations

**Unbalanced Dataset:** Original Dataset sample

**Standardized Dataset:** Data values standardized to the same scale.

## K-means Analysis and Conclusion

The usage of this analysis will help us understand whether by modifying a certain set of features, we would be able to increase/decrease the quality of the wine. The correlation matrix has effectively guided us to chose the most relevant attributes (alcohol, citric acid, pH, free sulphur dioxide ) that can affect the wine quality positively. We have also taken some negatively correlated factors into consideration to analyse the clustering patterns based on the features - Chlorides, Density, Volatile Acidity along with Alcohol for our analysis.

Plotting graphs to show the relationship between different variables can be considered basic step to determine the factors that have a larger impact and to work upon those factors. As seen,this clustering method was clearly able to group the wine into separate clusters based on the chosen attributes. This will help us to identify the quality of wine based on the grouping done.

Any further changes in increasing or decreasing the quality of the wine can be done changing the values of a select few attributes rather than altering the values of each and every attribute that impact the wine quality.

# 

# Overall Conclusion

Out of the three algorithms used for clustering, K-Means seems to be the most suitable method for our current dataset. We were able to identify clear clustering patterns using the K Means to categorize wine quality into distinct and separate groups. The optimal number of clusters is found to be 3. We can cluster the groups as low, medium and high quality to perform future analysis of data.

DBSCAN algorithm could not produce any significant results when dealing with the noise associated with the dataset. Decreasing the number of clusters, the noise reduction was only minimal and not as much as we expected. Hence we assume that this algorithm is not suitable for the current dataset in comparison to the K-Means where clear patterns were observed.

The Agglomerative method was also inefficient as we observed very unclear clustering patterns in the dataset. Even after reducing the size of the dataset to 20%, it could not yield any distinct patterns. The only thing that changed was that the cluster were slightly more spread out. This shows that huge database is not suitable for Agglomerative method.

From the above observations, we can conclude that the K Means is the most suitable performing clustering algorithm for the current dataset. DBSCAN and hierarchical clustering are technically inefficient as they suffer from indistinct clusters.