Predictive Analytics for a Bank Marketing Campaign

Data Cleaning, Exploration and Preparation Report

Robert Maheux

February 18 2019

Contents

[1. Introduction 3](#_Toc3900160)

[2. Import Python Libraries and Data 4](#_Toc3900161)

[3. Look for NULL Values 5](#_Toc3900162)

[4. Look for Invalid Data 5](#_Toc3900163)

[5. Explore the Outcome Variable 9](#_Toc3900164)

[6. Calculate Interquartile Range (IQR) to Determine Outliers 10](#_Toc3900165)

[7. Explore the Numerical Features 11](#_Toc3900166)

[7.1 Explore the Values for the AGE Feature 11](#_Toc3900167)

[7.2 Explore the Values for the DURATION Feature 14](#_Toc3900168)

[7.3 Explore the Values for the CAMPAIGN Feature 17](#_Toc3900169)

[7.4 Explore the Values for the PDAYS Feature 20](#_Toc3900170)

[7.5 Explore the Values for the PREVIOUS Feature 22](#_Toc3900171)

[7.6 Explore the Values for the EMP\_VAR\_RATE Feature 24](#_Toc3900172)

[7.7 Explore the Values for the CONS\_PRICE\_IDX Feature 25](#_Toc3900173)

[7.8 Explore the Values for the CONS\_CONF\_IDX Feature 26](#_Toc3900174)

[7.9 Explore the Values for the EURIBOR3M Feature 27](#_Toc3900175)

[7.10 Explore the Values for the NR\_EMPLOYED Feature 28](#_Toc3900176)

[8. Explore the Categorical Features 29](#_Toc3900177)

[8.1 Explore the Values for the JOB Feature 29](#_Toc3900178)

[8.2 Explore the Values for the MARITAL Feature 30](#_Toc3900179)

[8.3 Explore the Values for the EDUCATION Feature 31](#_Toc3900180)

[8.4 Explore the Values for the DEFAULT Feature 32](#_Toc3900181)

[8.5 Explore the Values for the HOUSING Feature 33](#_Toc3900182)

[8.6 Explore the Values for the LOAN Feature 34](#_Toc3900183)

[8.7 Explore the Values for the CONTACT Feature 35](#_Toc3900184)

[8.8 Explore the Values for the MONTH Feature 36](#_Toc3900185)

[8.9 Explore the Values for the DAY\_OF\_WEEK Feature 37](#_Toc3900186)

[8.10 Explore the Values for the POUTCOME Feature 38](#_Toc3900187)

[9. Explore the Correlation between the Numerical Features 39](#_Toc3900188)

[10. Consider the Imputation of Values 40](#_Toc3900189)

[11. Drop Some Features from the Dataset 40](#_Toc3900190)

[12. Transform Categorical Features into Numerical Dummy Variables 41](#_Toc3900191)

[13. Standardize the Features 44](#_Toc3900192)

[14. Handle the Unbalanced Data Using Random Under-Sampling 45](#_Toc3900193)

[15. Summary 47](#_Toc3900194)

[Appendix A – Listing of Python Commands Used for this Report 48](#_Toc3900195)

# 1. Introduction

This report describes the steps to clean, explore and prepare the data within the bank marketing dataset. The data will be explored in order to identify how each feature is distributed and to detect missing values, invalid values and outliers. The correlation between each features and the outcome variable will be examined. Finally, the categorical features will be converted to numeric features and the data will be standardized and the target variable will be balanced.

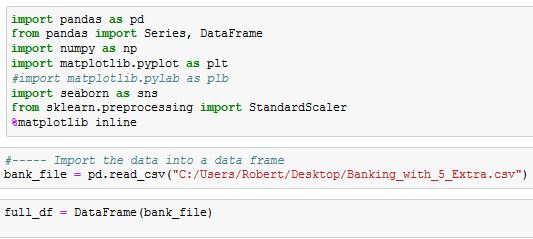
After the steps have been executed, the dataset will be ready to be used by the *feature selection* process.

The bank marketing dataset contains the following features:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Feature** | **Description** | **Type** | **Values** |
| 1 | age | Age of client | Numeric | 17 to 98 |
| 2 | job | Type of job | Categorical | admin, blue-collar, technician… |
| 3 | marital | Marital status | Categorical | divorced, married, single … |
| 4 | education | Education level | Categorical | high.school, university.degree… |
| 5 | default | Does the client have credit in default? | Categorical | yes, no, unknown |
| 6 | housing | Does the client have a housing loan | Categorical | yes, no, unknown |
| 7 | loan | Does the client have a personal loan? | Categorical | yes, no, unknown |
| 8 | contact | Contact communication type | Categorical | cellular, telephone |
| 9 | month | Last contact month of the year | Categorical | jan, feb, … dec |
| 10 | day\_of\_week | Last contact day of the week | Categorical | mon, tue, … fri |
| 11 | duration | Last contact duration, measured in seconds | Numeric | 0 to 4918 |
| 12 | campaign | Number of contacts during this campaign for this client | Numeric | 1 to 56 |
| 13 | pdays | Number of days since client was last contacted for a previous campaign | Numeric | 0 to 27, 999 |
| 14 | previous | Number of contacts before this campaign for this client | Numeric | 0 to 7 |
| 15 | poutcome | Outcome of the previous marketing campaign | Categorical | failure, nonexistent, success |
| 16 | emp.var.rate | Employment variation rate - quarterly indicator | Numeric | -3.4 to 1.4 |
| 17 | cons.price.idx | Consumer price index - monthly indicator | Numeric | 92.201 to 94.767 |
| 18 | cons.conf.idx | Consumer confidence index - monthly indicator | Numeric | -50.8 to -26.9 |
| 19 | euribor3m | Euribor 3 month rate - daily indicator | Numeric | 0.634 to 5.045 |
| 20 | nr.employed | Number of employees - quarterly indicator | Numeric | 4963.6 to 5228.1 |
| 21 | outcome | Does client want a term deposit account? | Categorical | yes, no |

# 2. Import Python Libraries and Data

Import the Python libraries and load the data into a data frame:



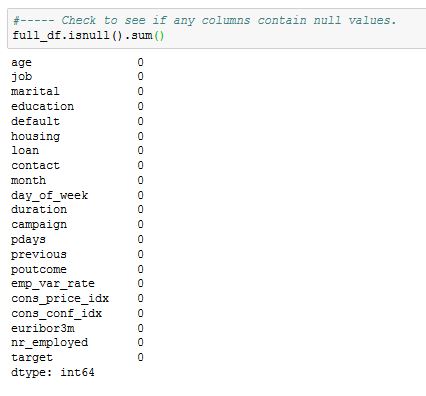
Display the shape of the data frame and peek at the data to ensure that it was loaded correctly.

We can see that **41,188 observations** and **21 features** were loaded. This corresponds to the data within the Excel spreadsheet.

#### 

# 3. Look for NULL Values

Check to see if any columns contain null values. The results indicate that there are no null values in any column.



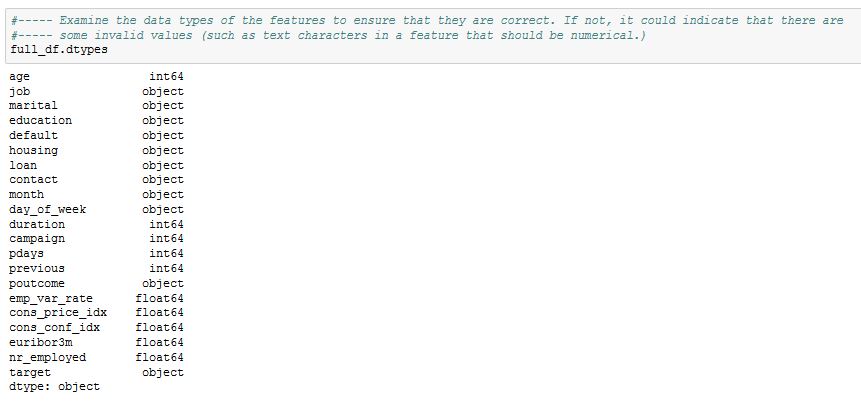
# 4. Look for Invalid Data

In this section, the data will be explored to see if there are any invalid values.

**4.1 Explore the Data types**

Explore the data types of the features to ensure that they are correct. If not, it could indicate that there are some invalid values such as text characters in a feature that should be numerical. In those cases, the column would have been created with a data type of *object* instead of *int64* or *float64*.

The results shown below indicate that the data types are correct for each feature. This indicates that the numerical features only contain numerical values and that the categorical features are not numerical.

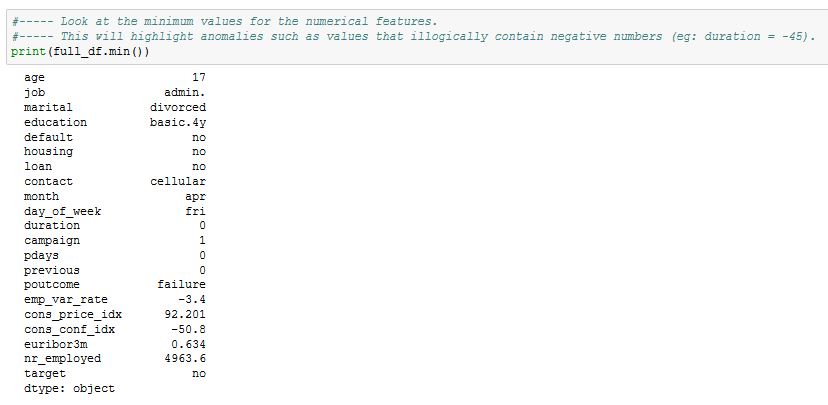


**4.2 Explore the Minimum and Maximum Values**

Look at the minimum values for the numerical features. This will highlight anomalies such as values that illogically contain negative numbers (eg. duration = -45) or values that are contextually too small

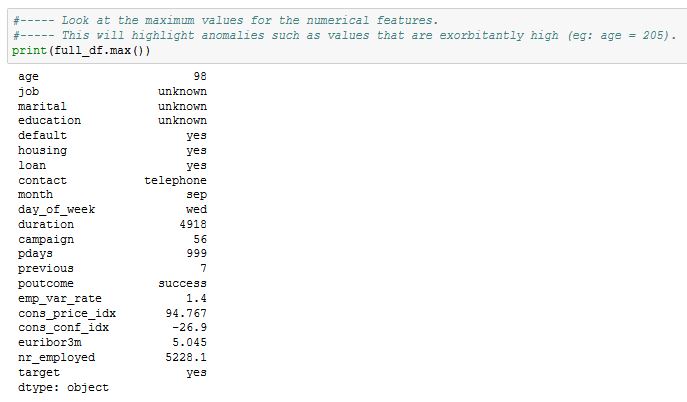
(eg. age = 5).

The results shown below display no anomalies.



Look at the maximum values for the numerical features. This will highlight anomalies such as values that are exorbitantly high, such as age = 205 or campaign = 10000.

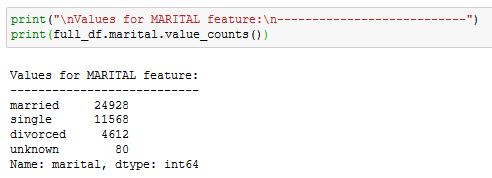
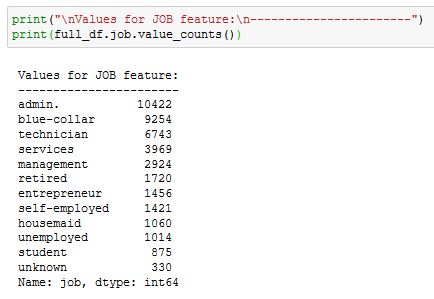
The results shown below display no anomalies.

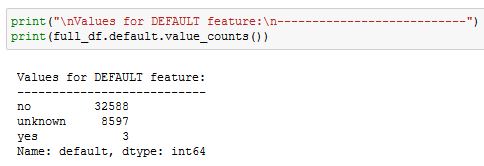
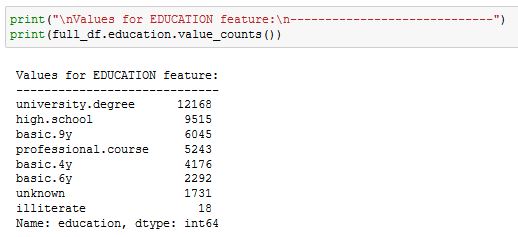


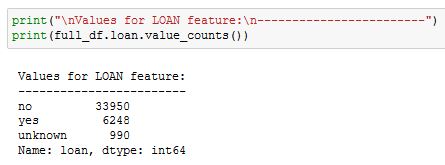
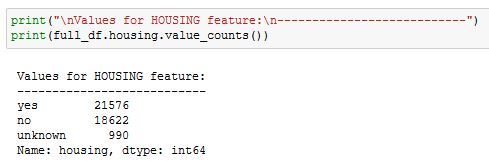
**4.3 Explore the Distinct Values within Each Categorical Feature**

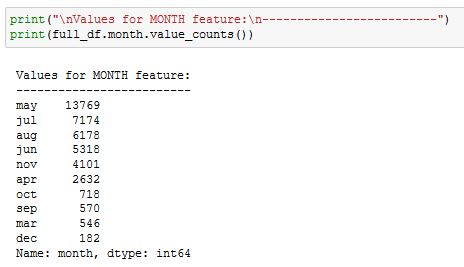
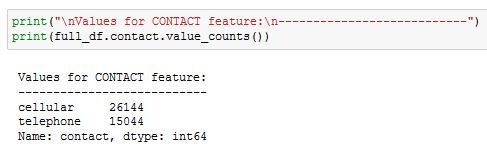
List the distinct values for each categorical feature. This will highlight invalid values.

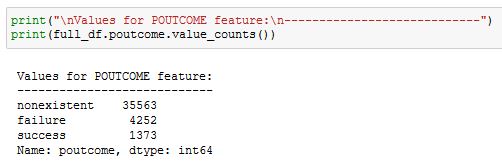
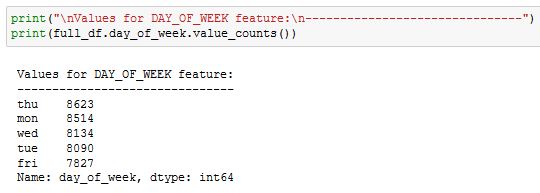
The screen snapshots below display a listing of the values for each categorical feature. The results display no anomalies or inconsistencies.

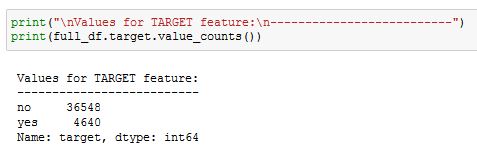








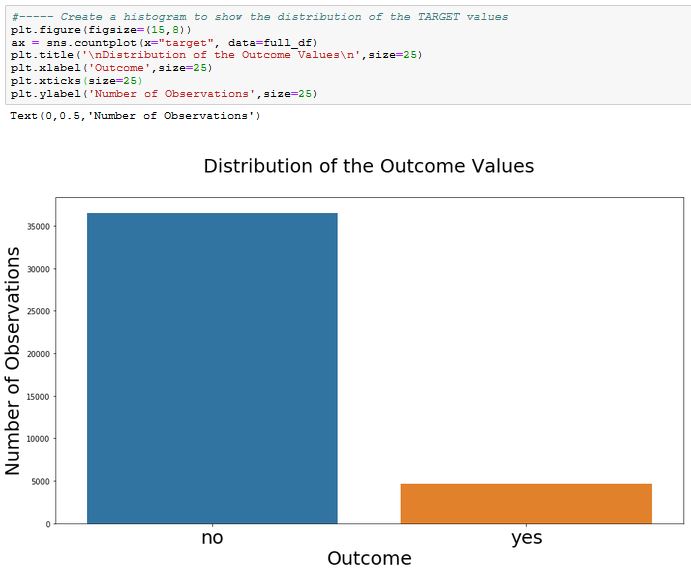


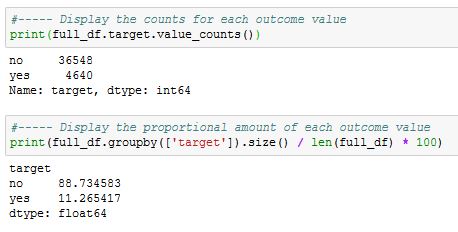


# 5. Explore the Outcome Variable

The outcome values are very imbalanced. There are 36,548 “no” values (88.7%) and only 4,640 “yes” values (11.3%).

Many classification algorithms struggle with imbalanced data so this will need to be addressed when the dataset is sampled and split into training and testing datasets.



****

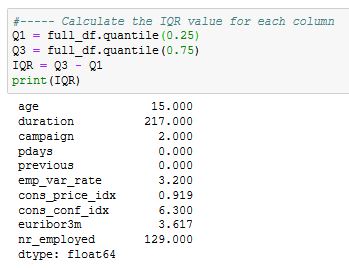
# 6. Calculate Interquartile Range (IQR) to Determine Outliers

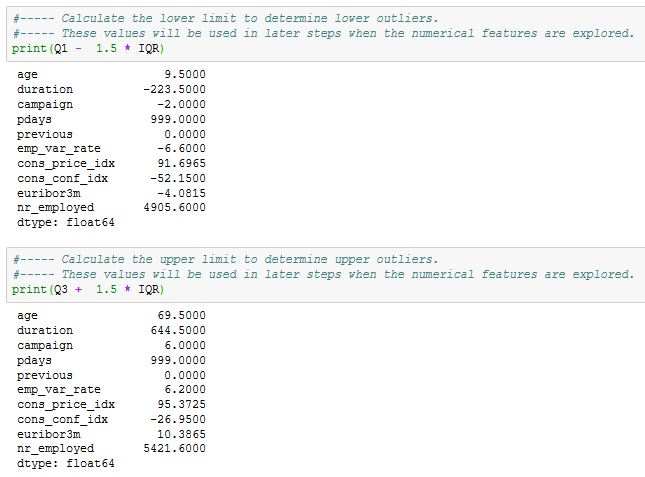
Calculate the Interquartile Range (IQR) values for each column. These values will be used later in this report to find the outliers for the numerical features.

Outliers are considered to be the values that are outside the range:

**[ (Q1 – 1.5\*(Q3 – Q1)) , (Q3 + 1.5\*(Q3 – Q1)) ]**

Calculate the IQR values and then use it to calculate the upper and lower limits of the range for each feature:





# 7. Explore the Numerical Features

The numerical features will be explored in order to answer the following questions:

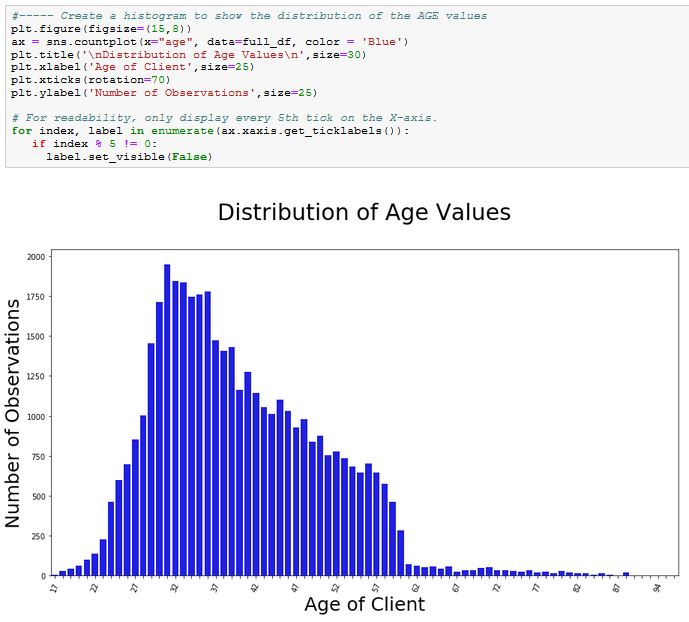
* What does the distribution of the data look like?
* How many outliers are there and where are they located?
* What is the proportion of the data that consists of outliers?
* What are the outcome values for the outliers?
* What is the correlation between the feature and the outcome variable?

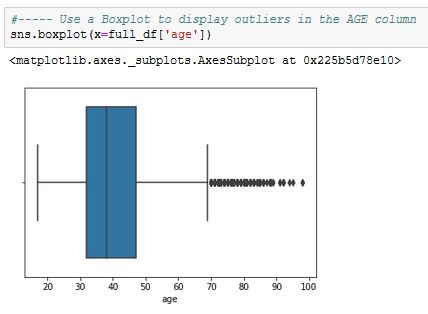
## 7.1 Explore the Values for the AGE Feature

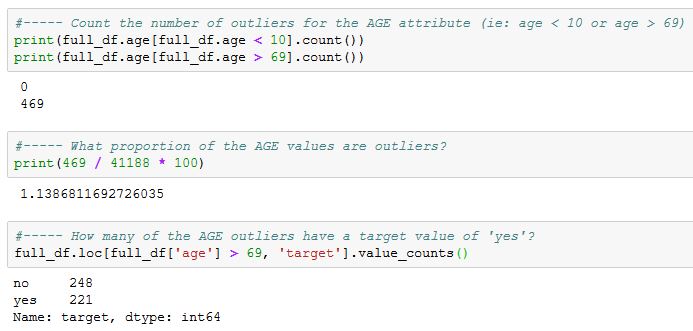
The results shown below indicate that the values for the *age* feature are almost normally distributed. There are some outliers in the upper region but these only represent 1.14% of the data.

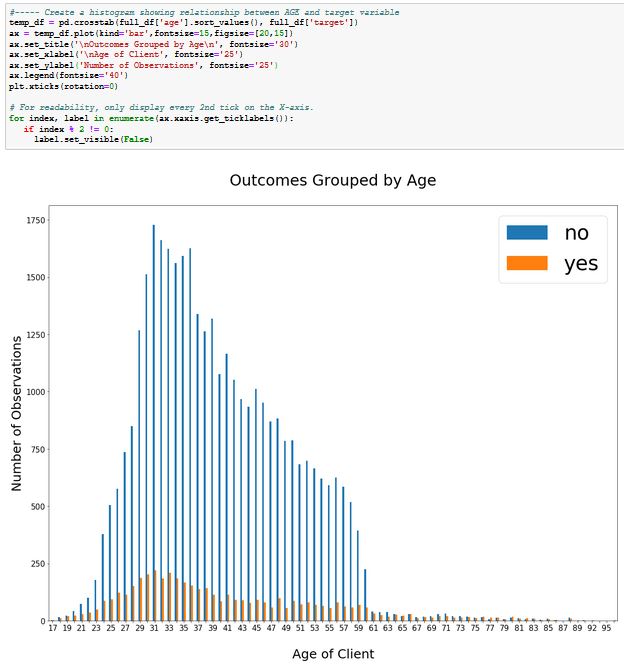
The graphs below indicate that the people who are most likely to agree to open a term savings account are the youngest or the oldest people. The outcomes (‘yes’ or ‘no’) are almost evenly balanced for the outliers.

It will be beneficial to keep the outliers since they are legitimate age values and they contain valuable information.







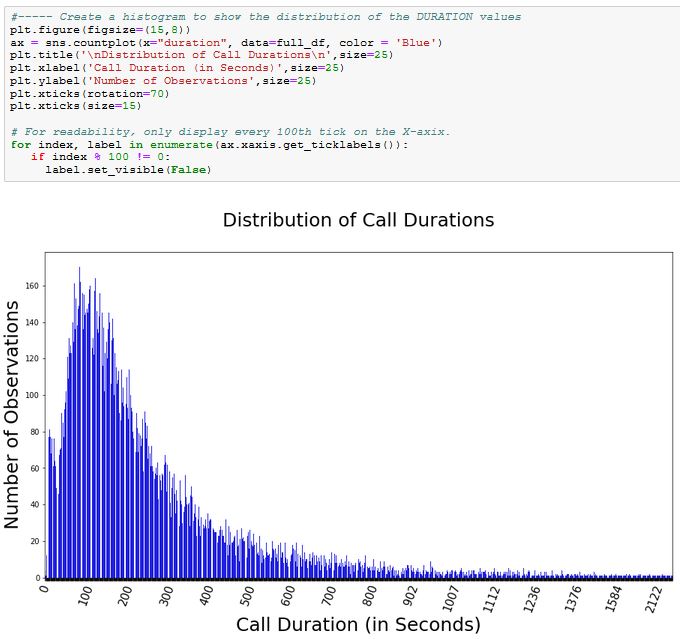


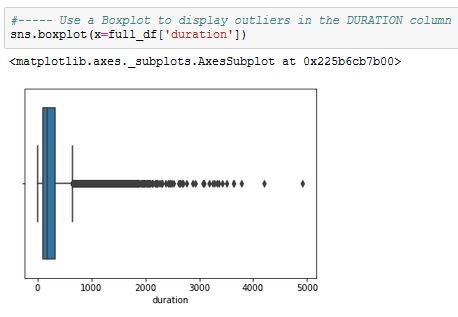
## 7.2 Explore the Values for the DURATION Feature

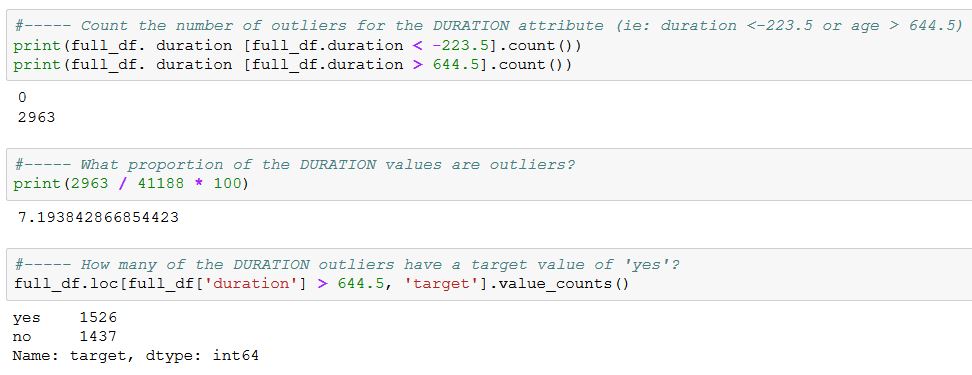
The results shown below indicate that the values for the *duration* feature are skewed to the right. All of the outliers are in the upper region and they represent 7.2% of the data. The values for the outliers are valid durations and their outcomes (‘yes’ or ‘no’) are almost evenly balanced.

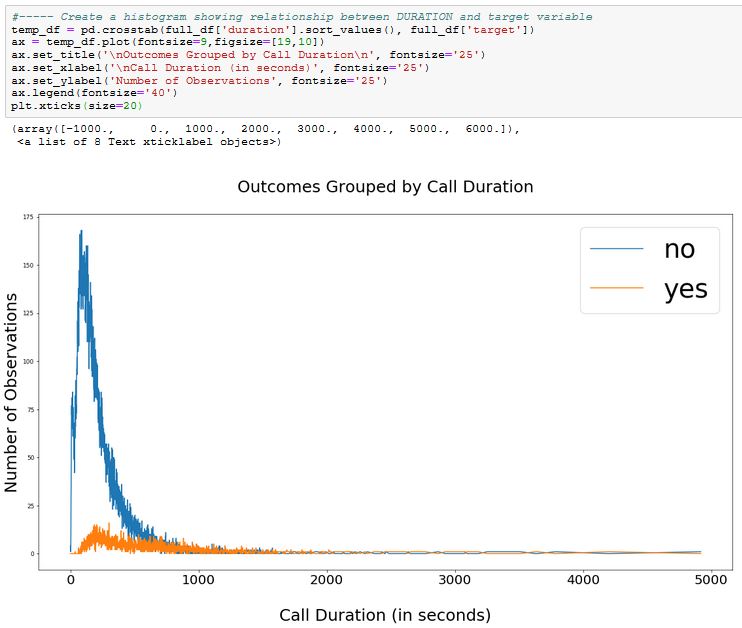
This indicates that the likelihood of obtaining a ‘yes’ outcome increases as the length of the call increases. This may be caused by the fact the fact that longer calls provide the bank agent with more time to build a convincing case for the client to open an account. On the other hand, it might merely indicate that when a client is interested, the call lasts longer.

In any case, it will be beneficial to keep these outliers since they are valid values and they contain valuable information.





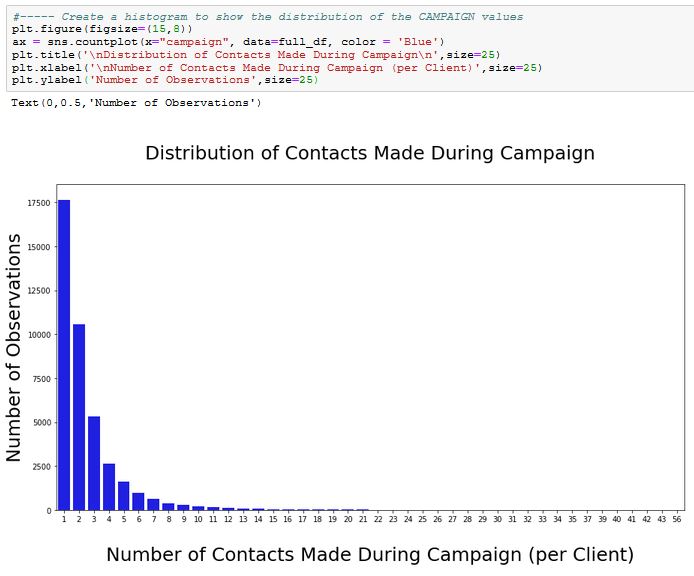


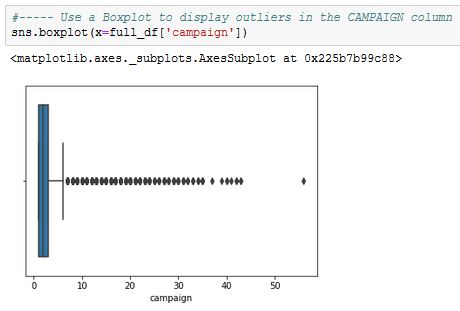


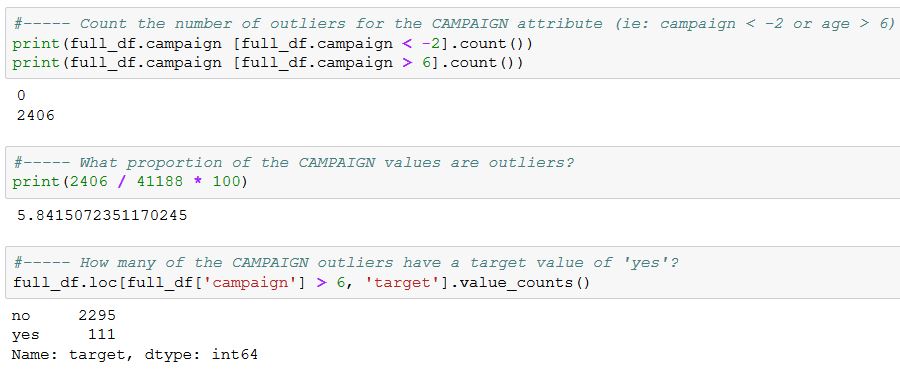
## 7.3 Explore the Values for the CAMPAIGN Feature

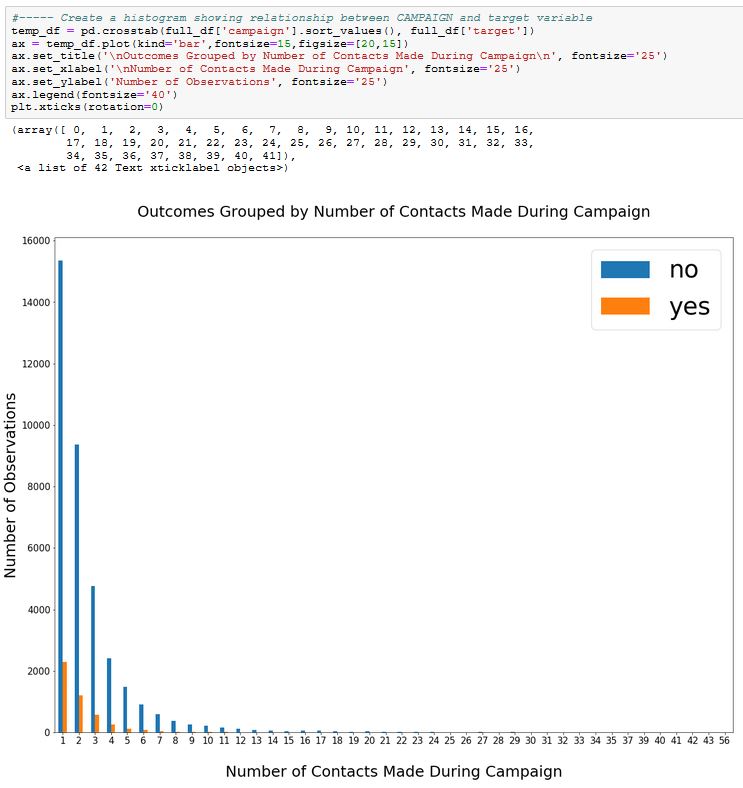
The results shown below indicate that the values for the *campaign* feature are skewed to the right. All of the outliers are in the upper region and they represent 5.8% of the data. The values for the outliers are valid durations. Most the outlier outcomes are ‘no’ but there is still a fair number of ‘yes’ outcomes.

It will be beneficial to keep these outliers since they are valid values and they contain valuable information.





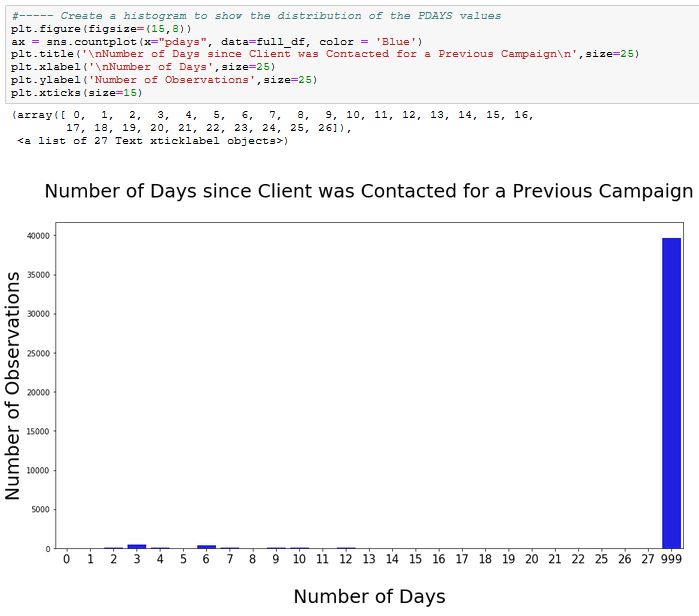


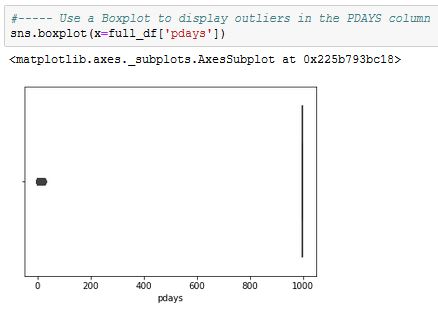


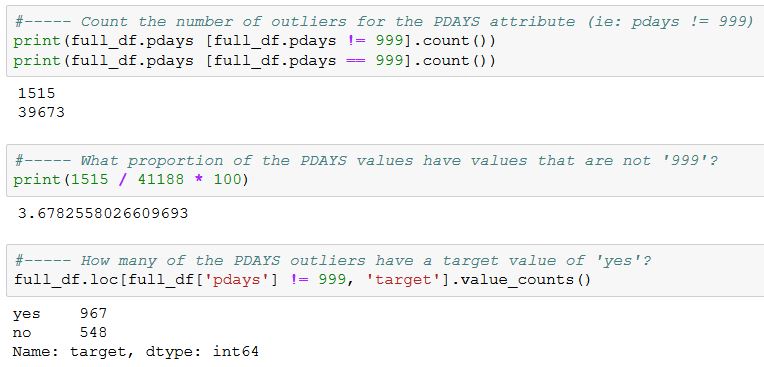
## 7.4 Explore the Values for the PDAYS Feature

For the *pdays* feature, the value ‘999’ indicates that the value is unknown. The results shown below indicate that only 3.68% of the values are known.

Since 96.3% of the values are designated as being unknown, it won’t be beneficial to include this feature in the model. It will be removed from the dataset prior to the data preparation steps.



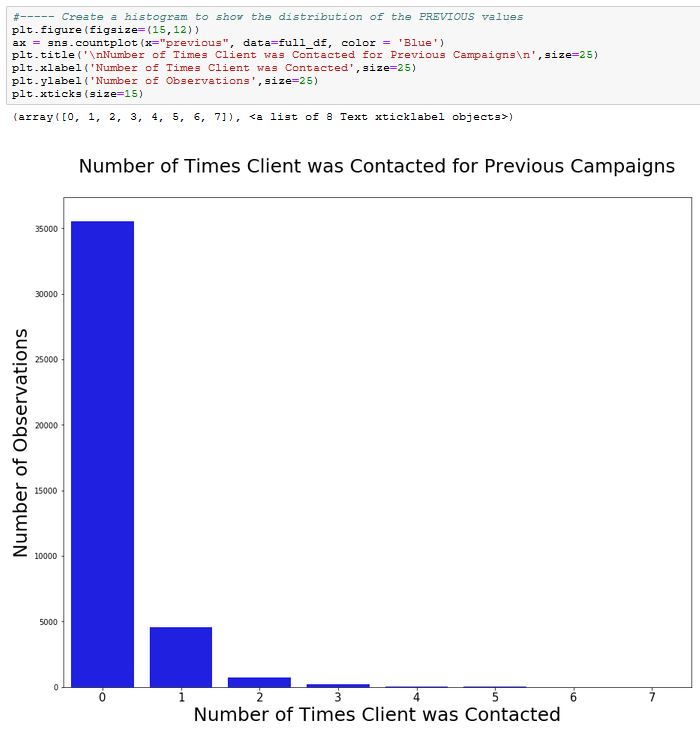


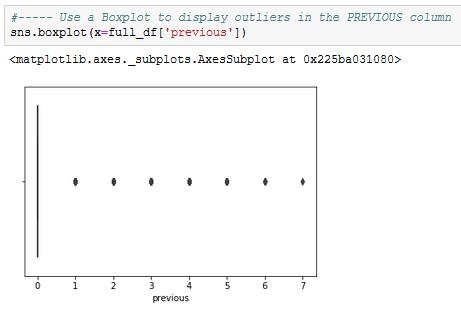


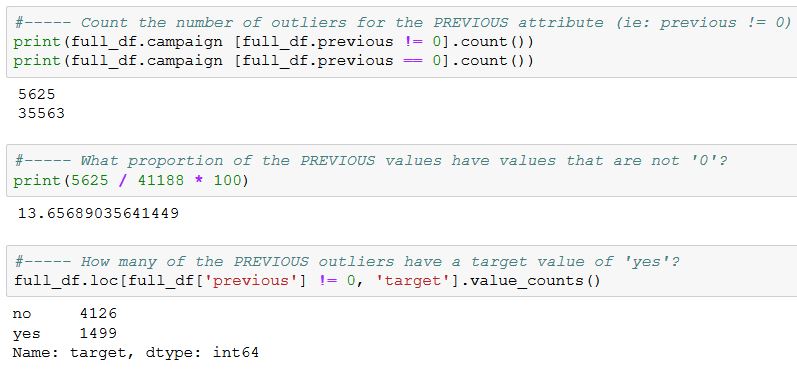
## 7.5 Explore the Values for the PREVIOUS Feature

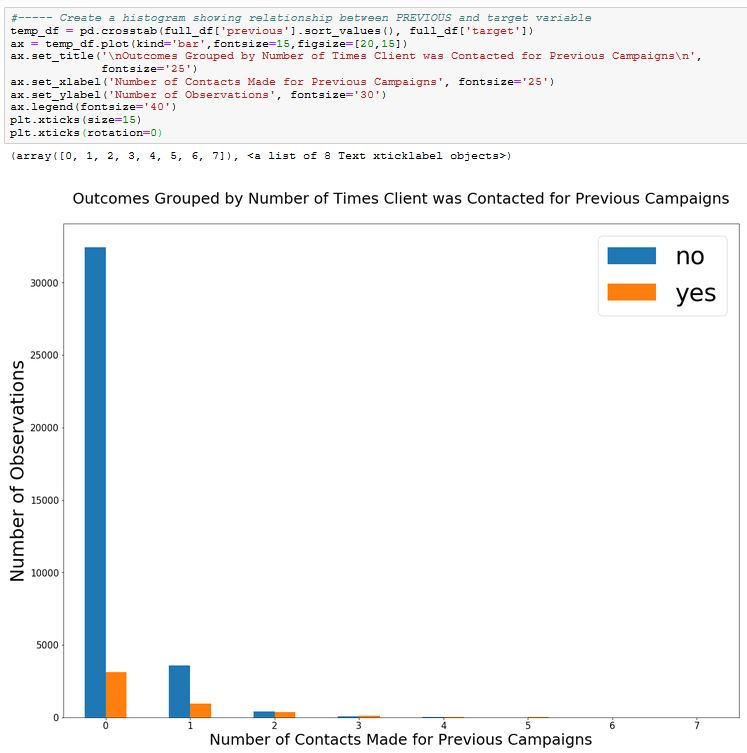
The ‘previous’ feature denotes the number of contacts with a client prior to the current campaign.

The results shown below indicate that only 13.65% observations have a value that isn’t ‘0’. However, when the values are compared with the outcomes, it’s obvious that the proportion of the ‘yes’ responses increases as the number of contacts increases. Although this variable is very imbalanced, it has a strong impact on the outcome variable and it should be included in the model.



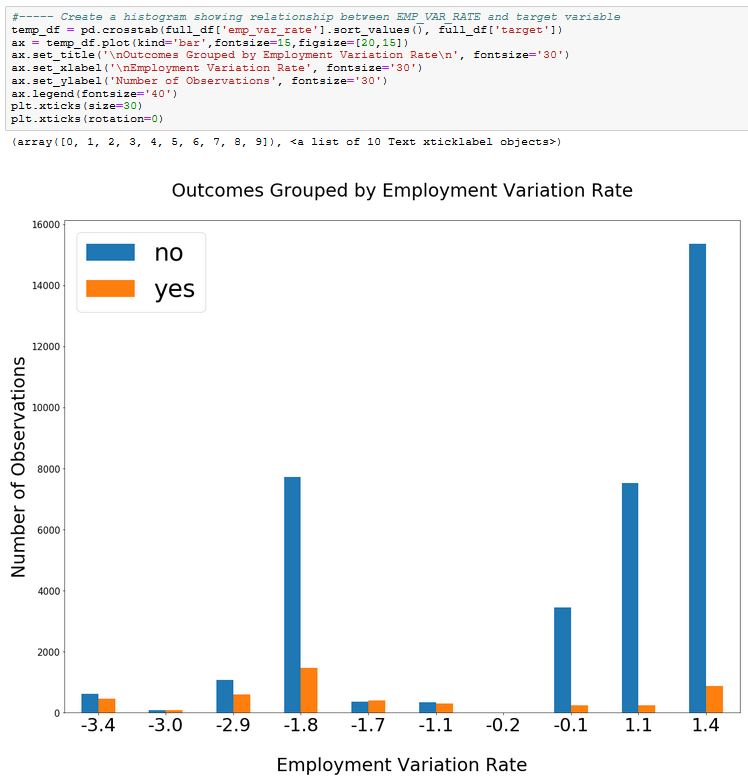






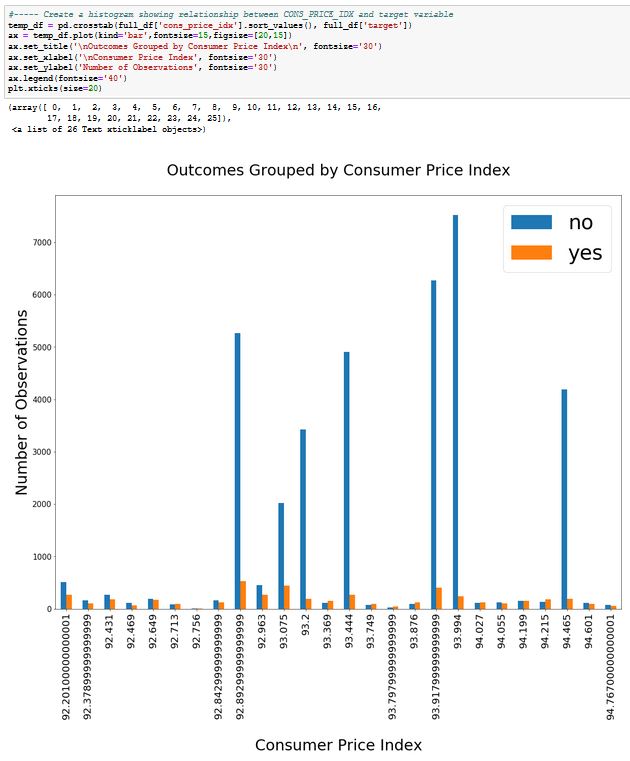
## 7.6 Explore the Values for the EMP\_VAR\_RATE Feature

This feature is a quarterly indicator for the employment variation rate. The bar chart below shows that when the rates are low, the proportion of “yes” outcomes tends to be higher. There’s an insufficient number of indicators to confirm whether or not there’s a direct relationship, but this feature seems to have an impact on the outcome variable.



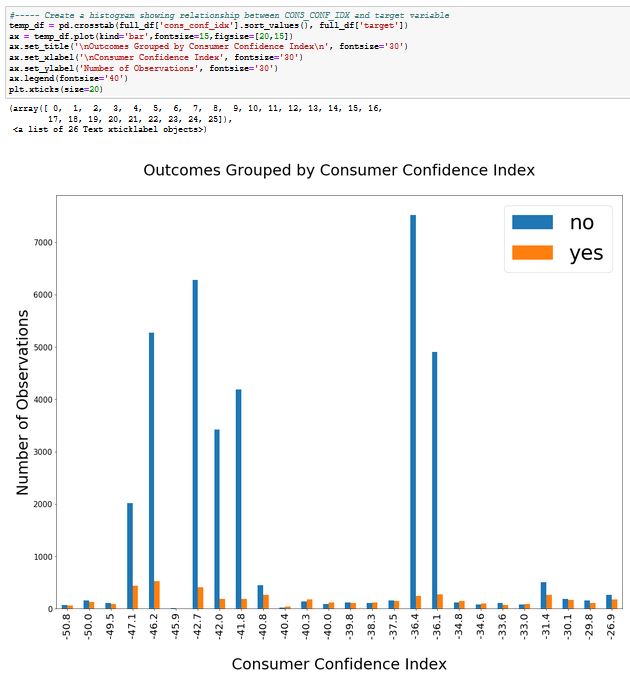
## 7.7 Explore the Values for the CONS\_PRICE\_IDX Feature

This feature is a monthly indicator for the consumer price index. From the bar chart below, it’s difficult to see whether or not there’s a relationship between this feature and the outcome variable. The Feature Selection phase of this project will be able to establish this.

****

## 7.8 Explore the Values for the CONS\_CONF\_IDX Feature

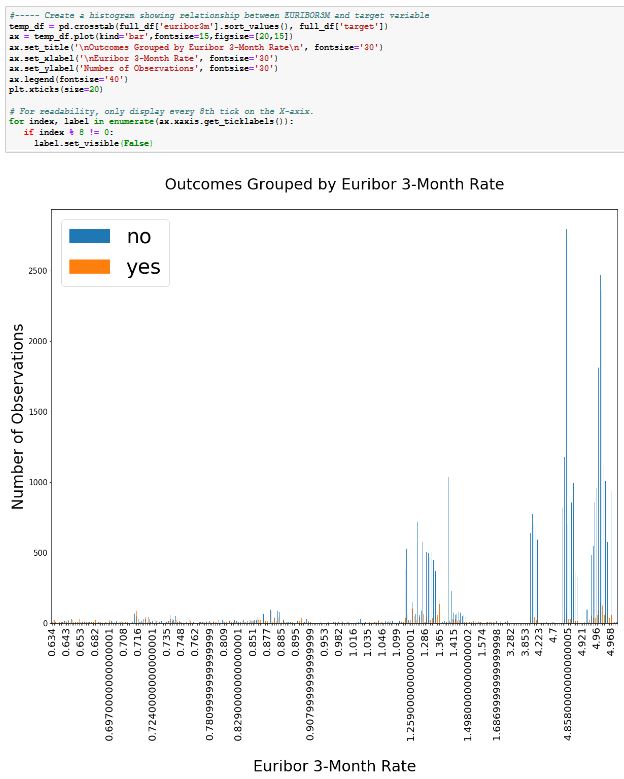
This feature is a monthly indicator for the consumer confidence index. From the bar chart below, it’s difficult to see whether or not there’s a relationship between this feature and the outcome variable. The Feature Selection phase of this project will be able to establish this.



## 7.9 Explore the Values for the EURIBOR3M Feature

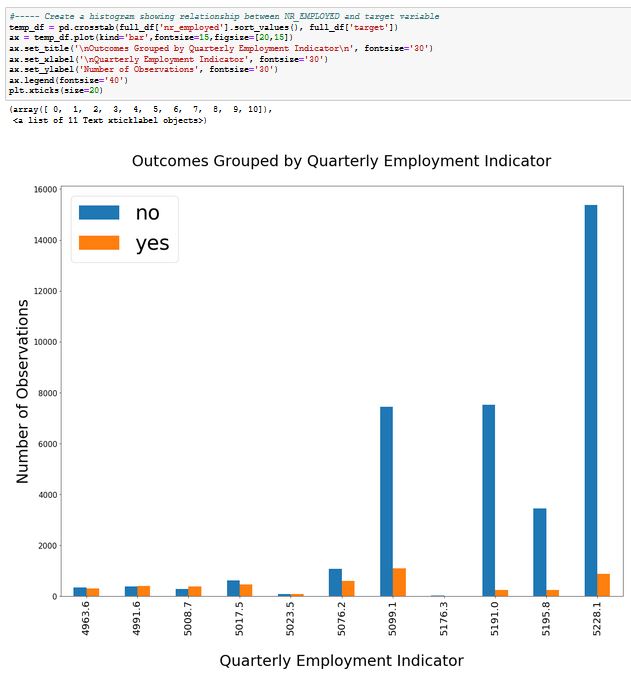
This feature is a daily indicator that represents the average interest rate that Eurozone banks charge each other for 3-month uncollateralized loans. It’s an important benchmark for a range of European financial products, including mortgages, savings accounts and car loans.

From the bar chart below, it seems that when the rates are low, the proportion of “yes” outcomes tends to be higher. The Feature Selection phase of this project will be able to analyze this more thoroughly.



## 7.10 Explore the Values for the NR\_EMPLOYED Feature

This variable is a quarterly average for the total number of employed citizens. It seems to have a strong impact on the outcome variable since the proportion of the ‘yes’ outcome clearly decreases as the indicator increases. This is illogical so it’s likely that the outcomes for the last five quarters were influenced by something else. The Feature Selection phase of this project will be able to establish the strength of the relationship between this feature and the outcome variable.



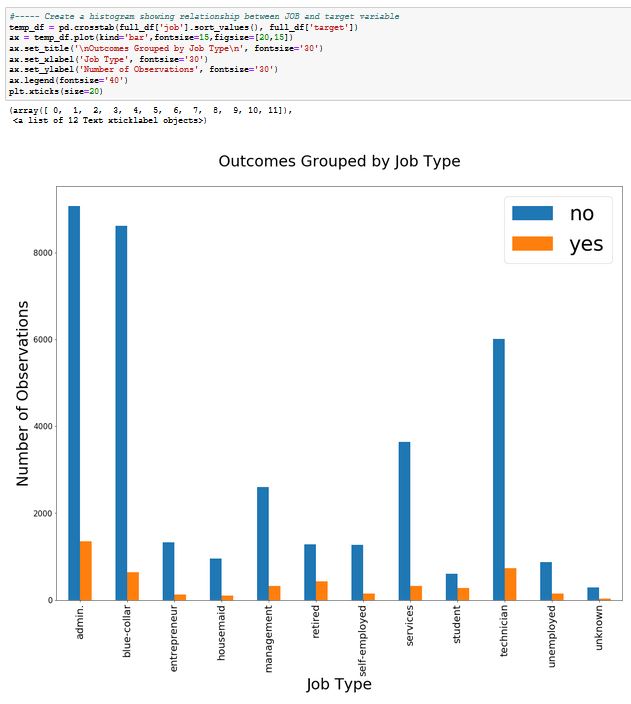
# 8. Explore the Categorical Features

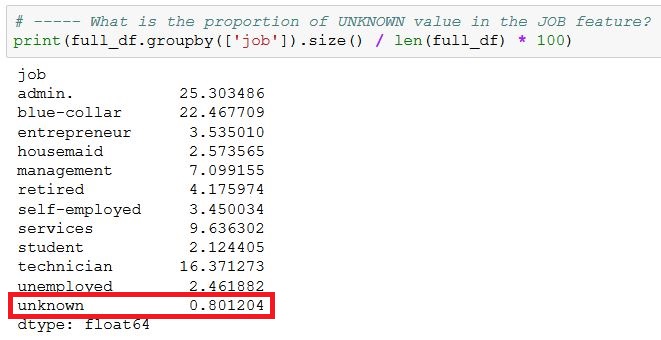
The categorical features will be explored in order to answer the following questions:

* What does the overall distribution of the data look like?
* What is the proportion of *yes/no* outcomes for each value?
* What is the proportion of the values entered as *unknown*?

## 8.1 Explore the Values for the JOB Feature

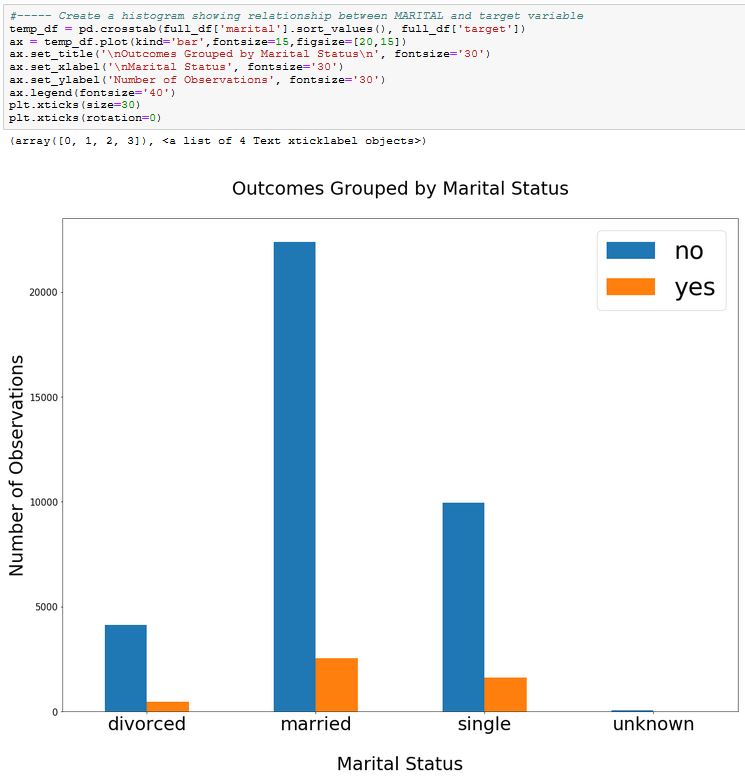
The bar chart below shows the distribution of yes/no outcomes according to each job type. Only 0.8% of the values have been marked as “unknown”. These values will be treated as a job type.

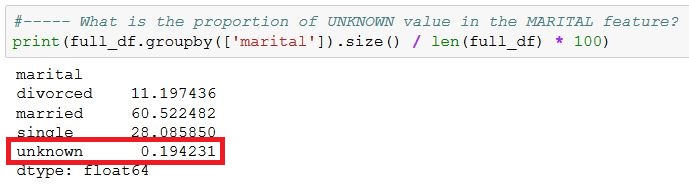




## 8.2 Explore the Values for the MARITAL Feature

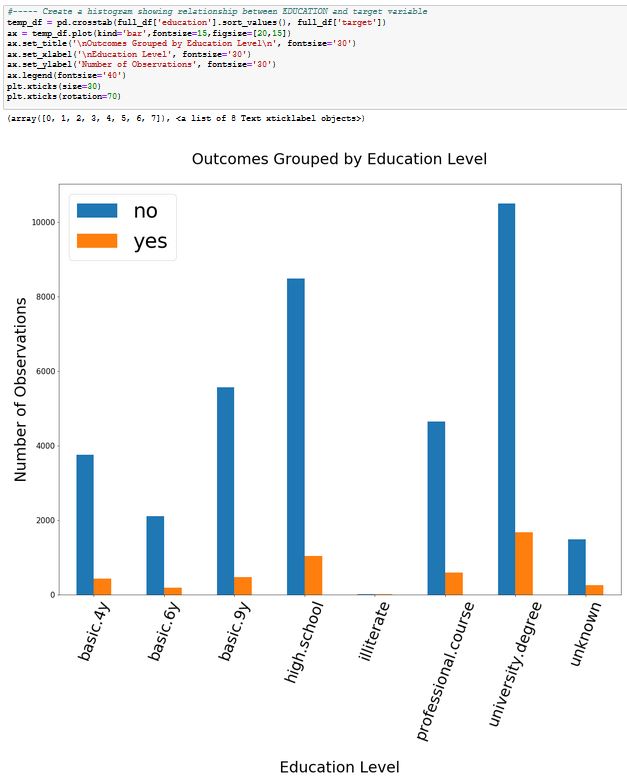
The bar chart below shows the distribution of yes/no outcomes according to each marital status. Only 0.2% of the values have been marked as “unknown”. These values will be treated as a marital status.

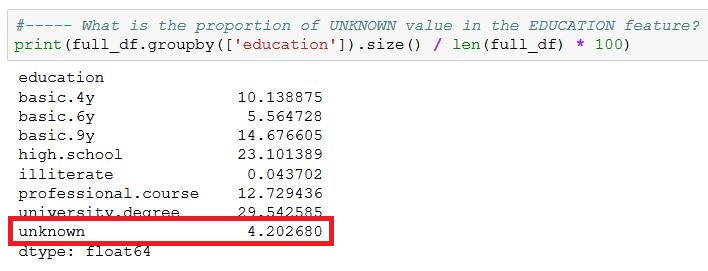




## 8.3 Explore the Values for the EDUCATION Feature

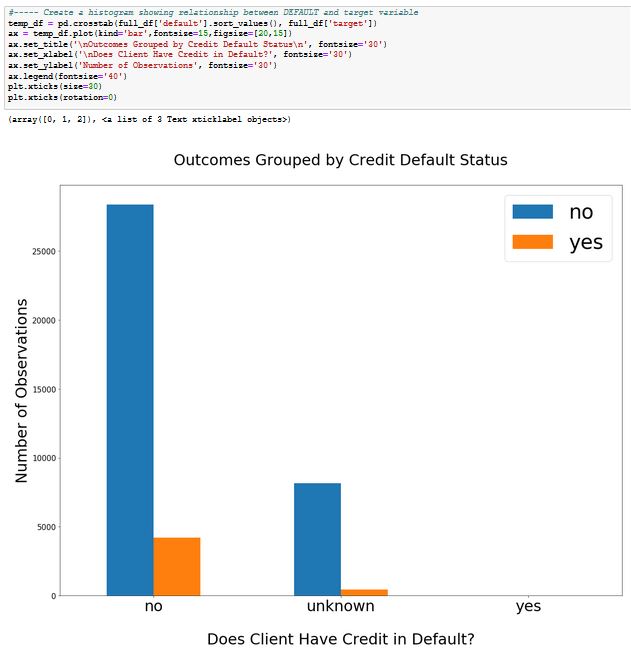
The bar chart below shows the distribution of yes/no outcomes according to each education level. Only 4.2% of the values have been marked as “unknown”. These values will be treated as an education level.

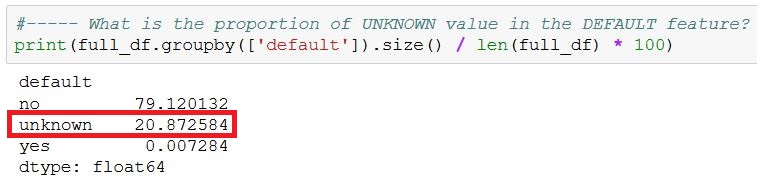




## 8.4 Explore the Values for the DEFAULT Feature

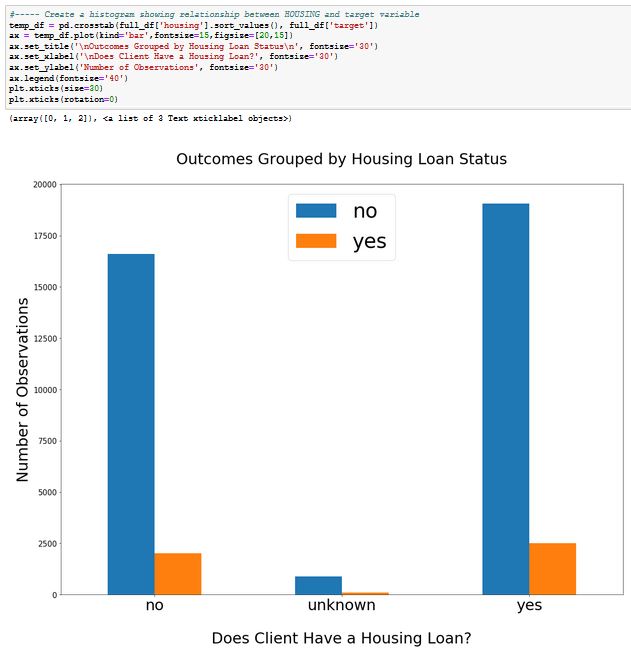
The bar chart below shows the distribution of yes/no outcomes according to whether or not the client has credit in default. This feature only contains 0.007% of “yes” outcomes which makes it meaningless. It will be removed from the dataset prior to the data preparation steps.

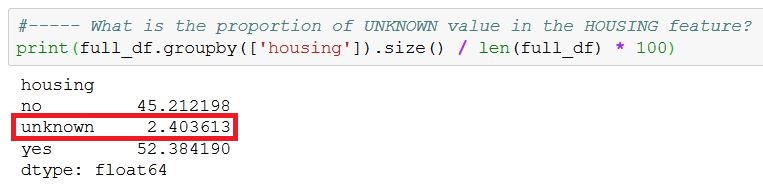




## 8.5 Explore the Values for the HOUSING Feature

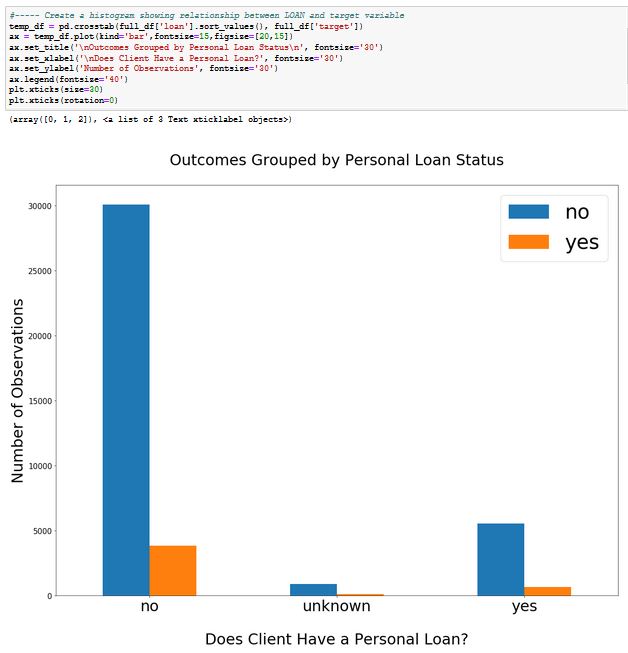
The bar chart below shows the distribution of yes/no outcomes according to whether or not the client has a housing loan. Only 2.4% of the values have been marked as “unknown”. These “unknown” values will be treated as a legitimate *housing* value.

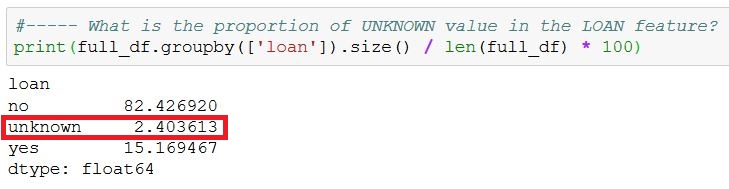




## 8.6 Explore the Values for the LOAN Feature

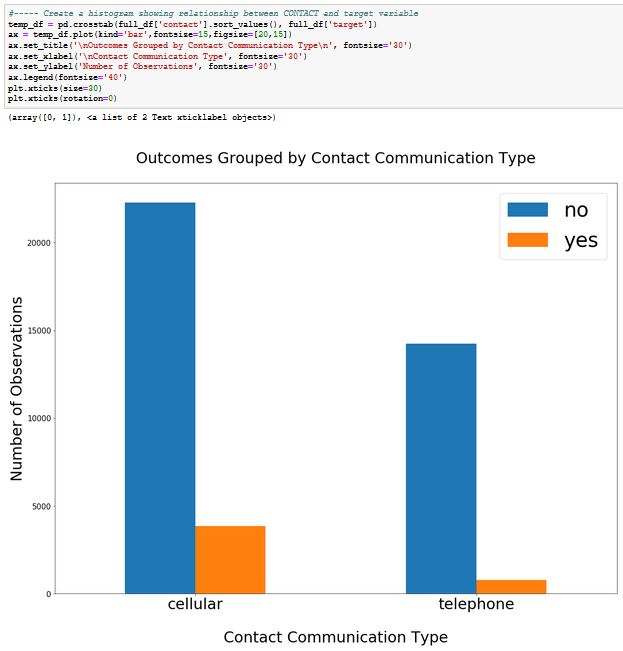
The bar chart below shows the distribution of yes/no outcomes according to whether or not the client has a housing loan. Only 2.4% of the values have been marked as “unknown”. These values will be treated as a value for the *loan* feature.





## 8.7 Explore the Values for the CONTACT Feature

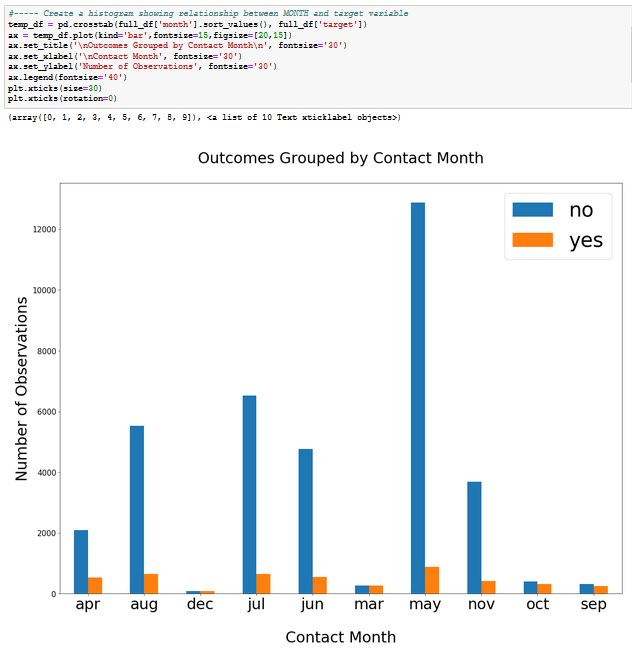
The bar chart below shows the distribution of yes/no outcomes according to whether the client was speaking to the bank agent via a cellular or wired phone. There are no unknown values in this feature. It is difficult to tell from the bar chart whether this variable has any impact on the outcome variable. The Feature Selection phase of this project will be able to analyse this more thoroughly.



## 8.8 Explore the Values for the MONTH Feature

The bar chart below shows the distribution of yes/no outcomes according to the month when the client was contacted. There are no unknown values in this feature.

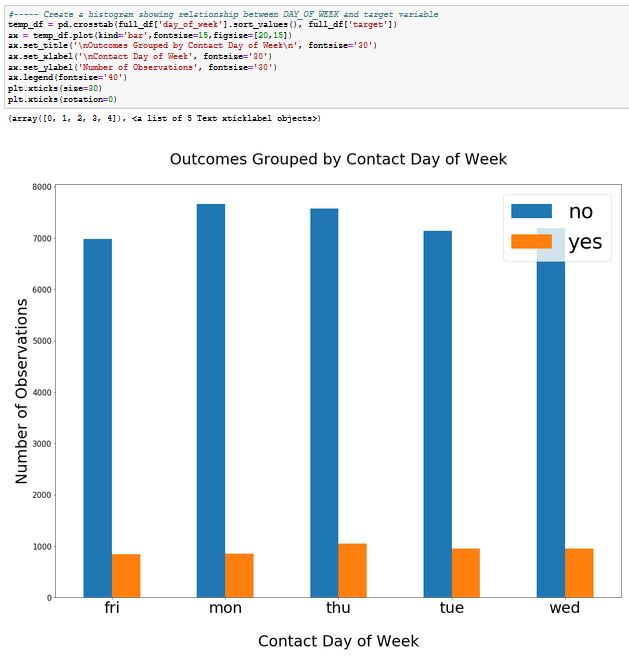
Interestingly, there are no values for January or February and the number of calls is exceptionally high between the months of May and August. This seems to indicate that there are high and low seasons for the bank’s telemarking campaigns.



## 8.9 Explore the Values for the DAY\_OF\_WEEK Feature

The bar chart below shows the distribution of yes/no outcomes according to the day of the week when the client was contacted. There are no unknown values in this feature.

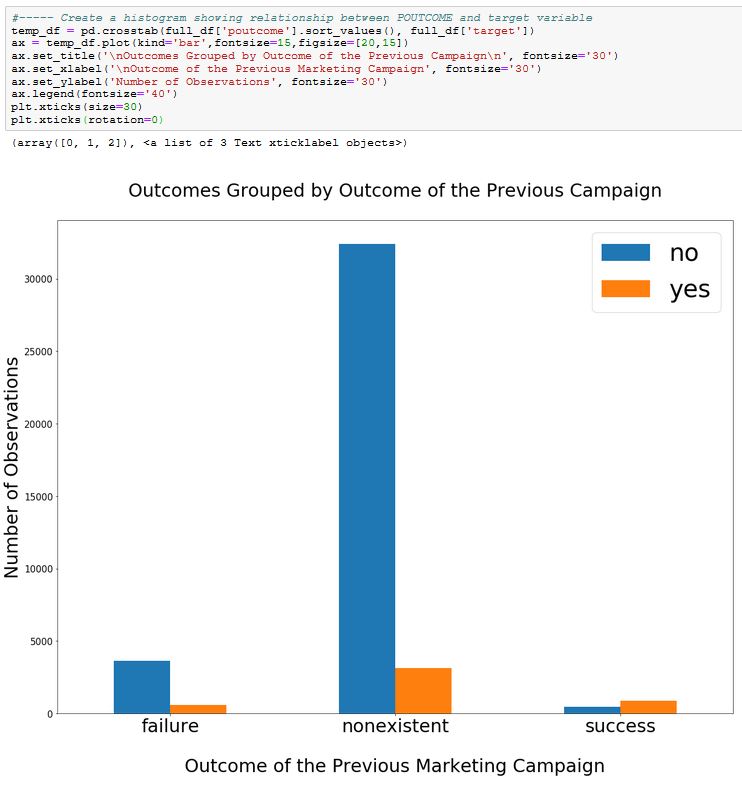
There seems to be a slight difference in the proportions, but it isn’t very strong. For instance, the number of “no” outcomes is very high for Mondays, while the number of “yes” outcomes is smaller than for any other day. The Feature Selection phase of this project will be able to analyse this more thoroughly.

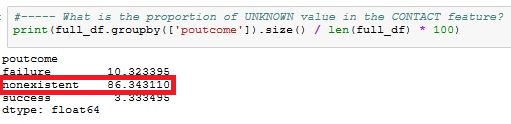


## 8.10 Explore the Values for the POUTCOME Feature

The bar chart below shows the distribution of yes/no outcomes according to the outcome of the previous marketing campaign for the client. It’s interesting to note that the majority of clients where the previous outcome was “success” have “yes” as the outcomes for the current campaign. This variable seems to have a high impact on the outcome variable.

Also, 86.3% of the calls were made to clients that had not been contacted for a previous campaign. The “nonexistent” value isn’t a missing value. It will be treated as a legitimate value for this feature.





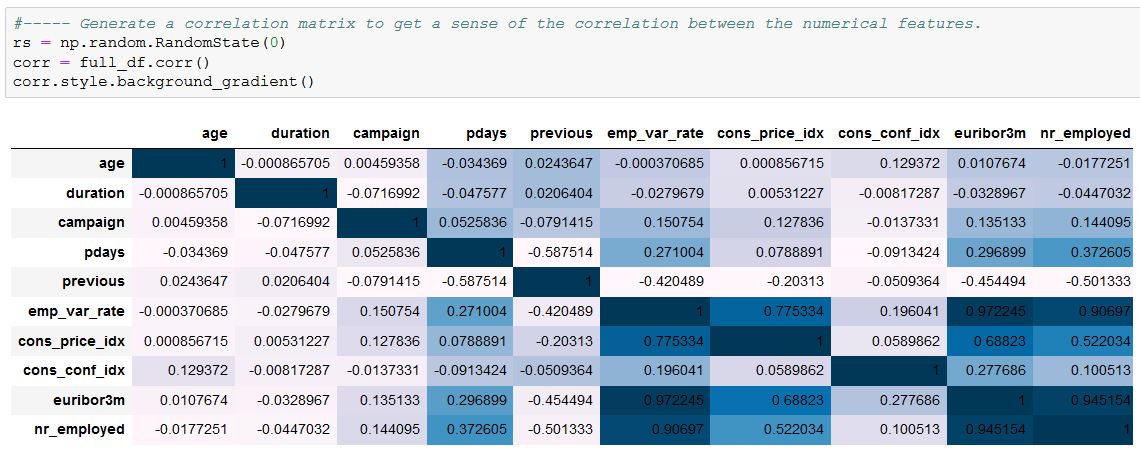
# 9. Explore the Correlation between the Numerical Features

The correlation matrix indicates that there is a high level of correlation between some of the “economic indicator” features, such as:

* 97.2% between *euribor3m* and *emp\_var\_rate*
* 94.5% between *euribor3m* and *nr\_employed*
* 90.7% between *nr\_employed* and *emp\_var\_rate*

One or more of these features will need to be eliminated. This will be explored further during the *feature selection* phase.

There seems to be a low level of multicollinearity between the remaining features.



# 10. Consider the Imputation of Values

The dataset doesn’t contain any missing values. However, several values have been entered with the indicator ‘*unknown’* or ‘*999’*. The reason for the values being unspecified is unknown so imputing them may damage the validity of the data.

There is a famous anecdote concerning a churn project for a large retailer, based on loyalty card data. The best predictor of churn turned out to be whether or not the customer had filled in his/her email address when registering for the card. Sometimes, a lack of information is valuable information itself.

The bank marketing values that have been marked as unknown will not be imputed. The fact that they are unspecified will be considered to be just another value for the categorical variable.

# 11. Drop Some Features from the Dataset

The following features have been judged to have minimal or no impact on the outcome variable. They are being removed from the dataset prior to the data preparation steps:

1. pdays
2. default



The *feature selection* process may identify more features that have minimal impact on the outcome variable. Once identified, those features will be removed as well.

# 

# 12. Transform Categorical Features into Numerical Dummy Variables

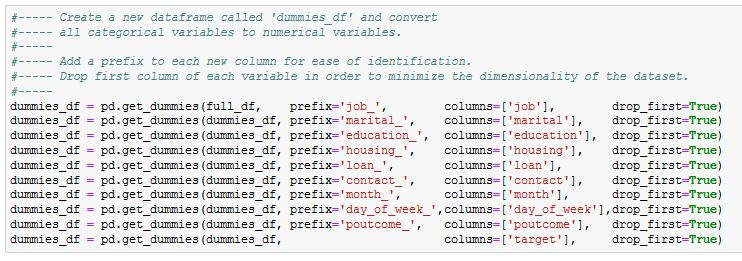
The categorical values need to be converted into numerical values so that they can be used by machine learning algorithms. The *get\_dummies* function will use the *one-hot* encoding method to separate the values into individual columns and convert them into 1 or 0. This is preferable to giving each value an individual number because it means that the features will not be weighted differently.

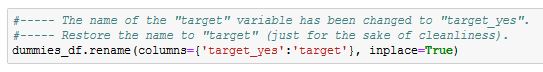
A prefix will be added to each dummy variable for ease of identification.

The first column for each feature will be dropped in order to minimize the dimensionality of the dataset.

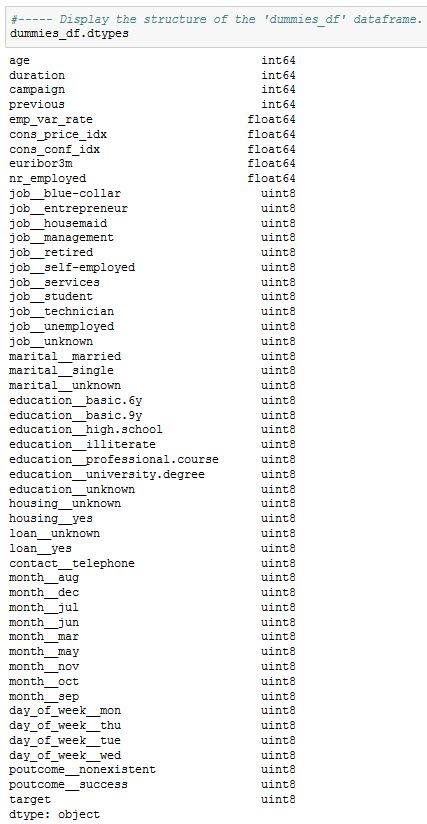
The values in the outcome feature will also be transformed. As a side-effect, the feature will get renamed to “target\_yes”. For the sake of cleanliness, its’ name will be restored to be “target”.

The final result will be placed in a dataset called *dummies\_df*.



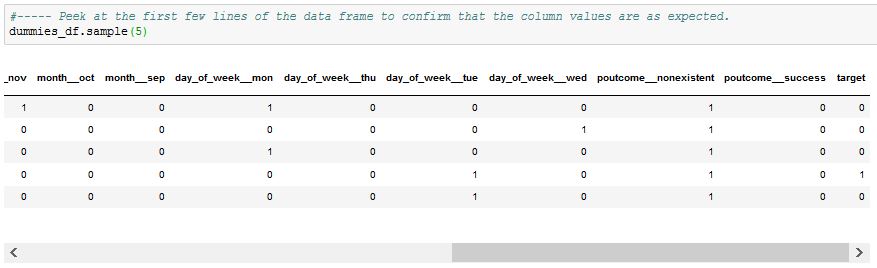


Display the structure of the *dummies\_df* data frame to confirm that the transformation was successful.



Peek at the first few lines of the data frame to confirm that the column values have been transformed as expected.

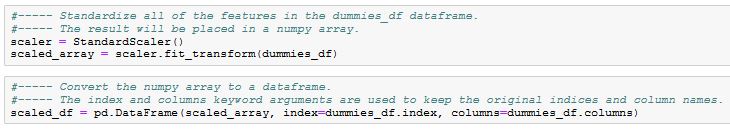




# 13. Standardize the Features

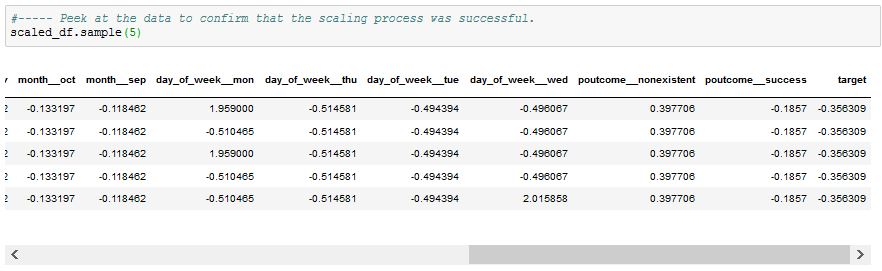
The features in the dataset need to be rescaled in order to prevent some features being weighted more than others. The data is standardized so that each feature has its’ mean = 0 and its’ standard deviation = 1.

The *StandardScaler* function is used to perform the data standardization. The final result is placed in a dataset called *scaled\_df*.



Peek at the data to confirm that the scaling process was successful.





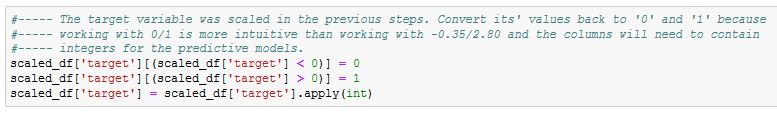
# 14. Handle the Unbalanced Data Using Random Under-Sampling

The outcome classes for the bank marketing dataset are very imbalanced. If this isn’t addressed, the predictive models will be biased and inaccurate because machine learning algorithms don’t take into account the distribution of the classes.

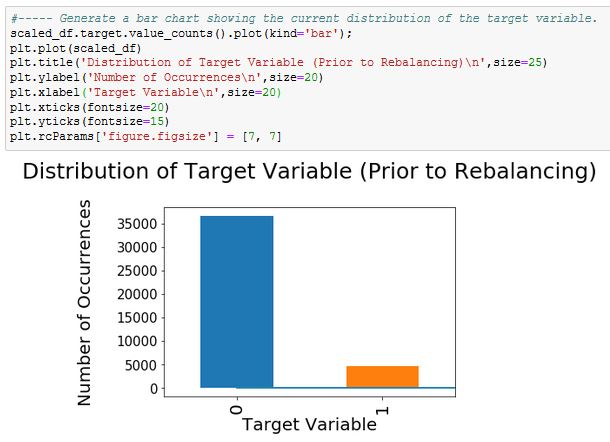
In this instance, 89% of the values are ‘0’ and 11% are ‘1’. An accuracy of 89% can be reached by simply predicting ‘0’ every time but the accuracy would only be a reflection of the class distribution.

This imbalance can be resolved with a technique called *under-sampling*. It involves the random removal of records from the majority class until the number of records for both classes is roughly the same.

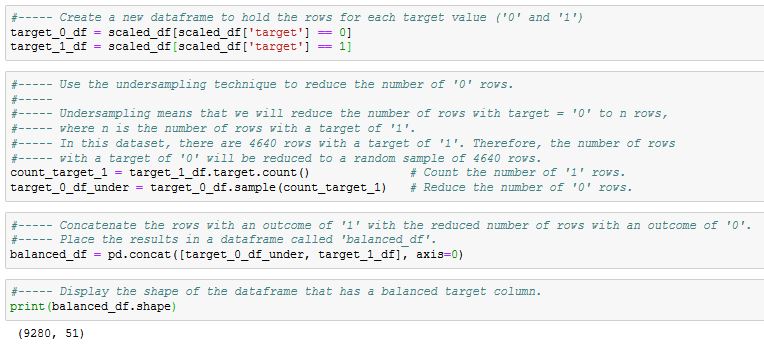
Before proceeding, the target values should be modified. During the scaling process, the 0/1 values were changed to be -0.35/2.80. Besides the fact that it’s more intuitive to work with 0/1 values, the target values will need to be integer values for the predictive models. The following snapshot displays the steps taken to convert the outcome values to 0/1 and to convert the column to an integer-type:

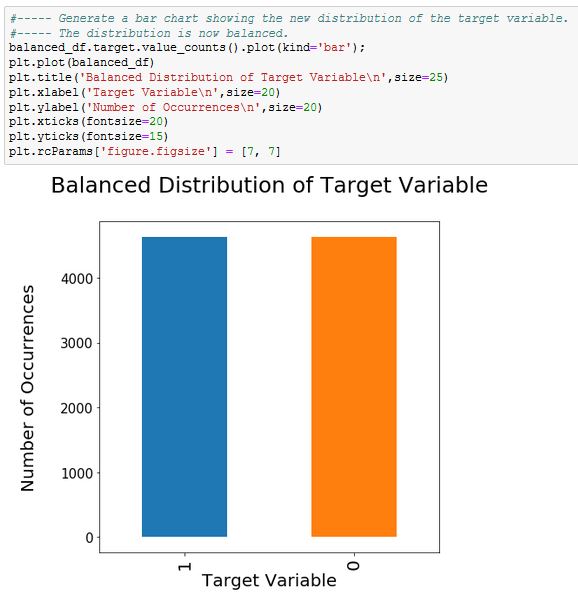


Generate a bar chart showing the original unbalanced distribution of the target variable:



The following snapshot displays the steps to apply the *under-sampling* technique to the target values. Two temporary data frames are created to hold the observations with outcomes of either ‘0’ or ‘1’. The number of observations with an outcome of ‘0’ are reduced so that their number is equal to the number of observations with an outcome of ‘1’. The two data frames are then merged into a single data frame to create a dataset with a balanced distribution of outcomes.



Generate a bar chart showing the new balanced distribution of the target variable:

# 15. Summary

Several features contained outliers but they haven’t been removed. It was considered to be beneficial to retain them since they are all legitimate values.

The dataset did not contain any missing or invalid values. However, some features contained values that were entered with the indicator ‘*unknown’* or ‘*999’*. These values have not been imputed. Instead, they’ll be considered to be just another value for the categorical variable.

Two features were dropped from the dataset because they were deemed to be too heavily imbalanced or to have no impact on the outcome variable. The *feature selection* process may identify more features that should be removed.

The categorical values needed to be converted into numerical values so that they can be used by machine learning algorithms. The *one-hot* encoding method was used to separate the values into individual columns and convert them into 1 or 0.

The features in the dataset had differing scales. To compensate for this, the data was standardized so that each feature has its’ mean = 0 and its’ standard deviation = 1.

The outcome classes for the bank marketing dataset were very imbalanced. This was resolved by using a technique called *under-sampling*.

The dataset is now ready to be used by the feature selection process.

# Appendix A – Listing of Python Commands Used for this Report

**Load the Libraries and Import the data**

#----- Import all the libraries and functions that we will need

import pandas as pd

from pandas import Series, DataFrame

import numpy as np

import matplotlib.pyplot as plt

#import matplotlib.pylab as plb

import seaborn as sns

from sklearn.preprocessing import StandardScaler

%matplotlib inline

#----- Import the data into a data frame

bank\_file = pd.read\_csv("C:/Users/Robert/Desktop/Banking\_with\_5\_Extra.csv")

full\_df = DataFrame(bank\_file)

#----- Look at the shape of the data

print(full\_df.shape)

#----- Peek at the first few lines of the data frame to ensure that all columns are loaded

full\_df.head(10)

**Look for NULL Values**

#----- Check to see if any columns contain null values.

full\_df.isnull().sum()

**Look for Invalid Data**

#----- Explore the data types of the features to ensure that they are correct. If not, it could indicate that there are

#----- some invalid values (such as text characters in a feature that should be numerical.)

full\_df.dtypes

#----- Look at the minimum values for the numerical features.

#----- This will highlight anomalies such as values that illogically contain negative numbers (eg: duration = -45).

print(full\_df.min())

#----- Look at the maximum values for the numerical features.

#----- This will highlight anomalies such as values that are exorbitantly high (eg: age = 205).

print(full\_df.max())

#----- List the distinct values for each categorical feature.

#----- This will highlight invalid values.

print("\nValues for JOB feature:\n-----------------------")

print(full\_df.job.value\_counts())

print("\nValues for MARITAL feature:\n---------------------------")

print(full\_df.marital.value\_counts())

print("\nValues for EDUCATION feature:\n-----------------------------")

print(full\_df.education.value\_counts())

print("\nValues for DEFAULT feature:\n---------------------------")

print(full\_df.default.value\_counts())

print("\nValues for HOUSING feature:\n---------------------------")

print(full\_df.housing.value\_counts())

print("\nValues for LOAN feature:\n------------------------")

print(full\_df.loan.value\_counts())

print("\nValues for CONTACT feature:\n---------------------------")

print(full\_df.contact.value\_counts())

print("\nValues for MONTH feature:\n-------------------------")

print(full\_df.month.value\_counts())

print("\nValues for DAY\_OF\_WEEK feature:\n-------------------------------")

print(full\_df.day\_of\_week.value\_counts())

print("\nValues for POUTCOME feature:\n----------------------------")

print(full\_df.poutcome.value\_counts())

print("\nValues for TARGET feature:\n--------------------------")

print(full\_df.target.value\_counts())

**Explore the Values for the TARGET Variable**

#----- Create a histogram to show the distribution of the TARGET values

plt.figure(figsize=(15,8))

ax = sns.countplot(x="target", data=full\_df)

plt.title('\nDistribution of the Outcome Values\n',size=25)

plt.xlabel('Outcome',size=25)

plt.xticks(size=25)

plt.ylabel('Number of Observations',size=25)

#----- Display the counts for each outcome value

print(full\_df.target.value\_counts())

#----- Display the proportional amount of each outcome value

print(full\_df.groupby(['target']).size() / len(full\_df) \* 100)

**Calculate Interquartile Range (IQR) to Determine Outliers**

#----- Calculate the IQR value for each column

Q1 = full\_df.quantile(0.25)

Q3 = full\_df.quantile(0.75)

IQR = Q3 - Q1

print(IQR)

#----- Calculate the lower limit to determine lower outliers.

#----- These values will be used in later steps when the numerical features are explored.

print(Q1 - 1.5 \* IQR)

#----- Calculate the upper limit to determine upper outliers.

#----- These values will be used in later steps when the numerical features are explored.

print(Q3 + 1.5 \* IQR)

**Explore the Numerical Variables**

**Explore the Values for the AGE Feature**

#----- Create a histogram to show the distribution of the AGE values

plt.figure(figsize=(15,8))

ax = sns.countplot(x="age", data=full\_df, color = 'Blue')

plt.title('\nDistribution of Age Values\n',size=30)

plt.xlabel('Age of Client',size=25)

plt.xticks(rotation=70)

plt.ylabel('Number of Observations',size=25)

# For readability, only display every 5th tick on the X-axis.

for index, label in enumerate(ax.xaxis.get\_ticklabels()):

if index % 5 != 0:

label.set\_visible(False)

#----- Use a Boxplot to display outliers in the AGE column

sns.boxplot(x=full\_df['age'])

#----- Count the number of outliers for the AGE attribute (ie: age < 10 or age > 69)

print(full\_df.age[full\_df.age < 10].count())

print(full\_df.age[full\_df.age > 69].count())

#----- What proportion of the AGE values are outliers?

print(469 / 41188 \* 100)

#----- How many of the AGE outliers have a target value of 'yes'?

full\_df.loc[full\_df['age'] > 69, 'target'].value\_counts()

#----- Create a histogram showing relationship between AGE and target variable

temp\_df = pd.crosstab(full\_df['age'].sort\_values(), full\_df['target'])

ax = temp\_df.plot(kind='bar',fontsize=15,figsize=[20,15])

ax.set\_title('\nOutcomes Grouped by Age\n', fontsize='30')

ax.set\_xlabel('\nAge of Client', fontsize='25')

ax.set\_ylabel('Number of Observations', fontsize='25')

ax.legend(fontsize='40')

plt.xticks(rotation=0)

# For readability, only display every 2nd tick on the X-axis.

for index, label in enumerate(ax.xaxis.get\_ticklabels()):

if index % 2 != 0:

label.set\_visible(False)

**Explore the Values for the DURATION Feature**

#----- Create a histogram to show the distribution of the DURATION values

plt.figure(figsize=(15,8))

ax = sns.countplot(x="duration", data=full\_df, color = 'Blue')

plt.title('\nDistribution of Call Durations\n',size=25)

plt.xlabel('Call Duration (in Seconds)',size=25)

plt.ylabel('Number of Observations',size=25)

plt.xticks(rotation=70)

plt.xticks(size=15)

# For readability, only display every 100th tick on the X-axix.

for index, label in enumerate(ax.xaxis.get\_ticklabels()):

if index % 100 != 0:

label.set\_visible(False)

#----- Use a Boxplot to display outliers in the DURATION column

sns.boxplot(x=full\_df['duration'])

#----- Count the number of outliers for the DURATION attribute (ie: duration <-223.5 or age > 644.5)

print(full\_df. duration [full\_df.duration < -223.5].count())

print(full\_df. duration [full\_df.duration > 644.5].count())

#----- What proportion of the DURATION values are outliers?

print(2963 / 41188 \* 100)

#----- How many of the DURATION outliers have a target value of 'yes'?

full\_df.loc[full\_df['duration'] > 644.5, 'target'].value\_counts()

#----- Create a histogram showing relationship between DURATION and target variable

temp\_df = pd.crosstab(full\_df['duration'].sort\_values(), full\_df['target'])

ax = temp\_df.plot(fontsize=9,figsize=[19,10])

ax.set\_title('\nOutcomes Grouped by Call Duration\n', fontsize='25')

ax.set\_xlabel('\nCall Duration (in seconds)', fontsize='25')

ax.set\_ylabel('Number of Observations', fontsize='25')

ax.legend(fontsize='40')

plt.xticks(size=20)

**Explore the Values for the CAMPAIGN Feature**

#----- Create a histogram to show the distribution of the CAMPAIGN values

plt.figure(figsize=(15,8))

ax = sns.countplot(x="campaign", data=full\_df, color = 'Blue')

plt.title('\nDistribution of Contacts Made During Campaign\n',size=25)

plt.xlabel('\nNumber of Contacts Made During Campaign (per Client)',size=25)

plt.ylabel('Number of Observations',size=25)

#----- Use a Boxplot to display outliers in the CAMPAIGN column

sns.boxplot(x=full\_df['campaign'])

#----- Count the number of outliers for the CAMPAIGN attribute (ie: campaign < -2 or age > 6)

print(full\_df.campaign [full\_df.campaign < -2].count())

print(full\_df.campaign [full\_df.campaign > 6].count())

#----- What proportion of the CAMPAIGN values are outliers?

print(2406 / 41188 \* 100)

#----- How many of the CAMPAIGN outliers have a target value of 'yes'?

full\_df.loc[full\_df['campaign'] > 6, 'target'].value\_counts()

#----- Create a histogram showing relationship between CAMPAIGN and target variable

temp\_df = pd.crosstab(full\_df['campaign'].sort\_values(), full\_df['target'])

ax = temp\_df.plot(kind='bar',fontsize=15,figsize=[20,15])

ax.set\_title('\nOutcomes Grouped by Number of Contacts Made During Campaign\n', fontsize='25')

ax.set\_xlabel('\nNumber of Contacts Made During Campaign', fontsize='25')

ax.set\_ylabel('Number of Observations', fontsize='25')

ax.legend(fontsize='40')

plt.xticks(rotation=0)

**Explore the Values for the PDAYS Feature**

#----- Create a histogram to show the distribution of the PDAYS values

plt.figure(figsize=(15,8))

ax = sns.countplot(x="pdays", data=full\_df, color = 'Blue')

plt.title('\nNumber of Days since Client was Contacted for a Previous Campaign\n',size=25)

plt.xlabel('\nNumber of Days',size=25)

plt.ylabel('Number of Observations',size=25)

plt.xticks(size=15)

#----- Use a Boxplot to display outliers in the PDAYS column

sns.boxplot(x=full\_df['pdays'])

#----- Count the number of outliers for the PDAYS attribute (ie: pdays != 999)

print(full\_df.pdays [full\_df.pdays != 999].count())

print(full\_df.pdays [full\_df.pdays == 999].count())

#----- What proportion of the PDAYS values have values that are not '999'?

print(1515 / 41188 \* 100)

#----- How many of the PDAYS outliers have a target value of 'yes'?

full\_df.loc[full\_df['pdays'] != 999, 'target'].value\_counts()

**Explore the Values for the PREVIOUS Feature**

#----- Create a histogram to show the distribution of the PREVIOUS values

plt.figure(figsize=(15,12))

ax = sns.countplot(x="pdays", data=full\_df, color = 'Blue')

plt.title('\nNumber of Times Client was Contacted for Previous Campaigns\n',size=25)

plt.xlabel('Number of Times Client was Contacted',size=25)

plt.ylabel('Number of Observations',size=25)

plt.xticks(size=15)

#----- Use a Boxplot to display outliers in the PREVIOUS column

sns.boxplot(x=full\_df['previous'])

#----- Count the number of outliers for the PREVIOUS attribute (ie: previous != 0)

print(full\_df.campaign [full\_df.previous != 0].count())

print(full\_df.campaign [full\_df.previous == 0].count())

#----- What proportion of the PREVIOUS values have values that are not '0'?

print(5625 / 41188 \* 100)

#----- How many of the PREVIOUS outliers have a target value of 'yes'?

full\_df.loc[full\_df['previous'] != 0, 'target'].value\_counts()

#----- Create a histogram showing relationship between PREVIOUS and target variable

temp\_df = pd.crosstab(full\_df['previous'].sort\_values(), full\_df['target'])

ax = temp\_df.plot(kind='bar',fontsize=15,figsize=[20,15])

ax.set\_title('\nOutcomes Grouped by Number of Times Client was Contacted for Previous Campaigns\n',

fontsize='25')

ax.set\_xlabel('Number of Contacts Made for Previous Campaigns', fontsize='25')

ax.set\_ylabel('Number of Observations', fontsize='30')

ax.legend(fontsize='40')

plt.xticks(size=15)

plt.xticks(rotation=0)

#----- Create a histogram showing relationship between EMP\_VAR\_RATE and target variable

temp\_df = pd.crosstab(full\_df['emp\_var\_rate'].sort\_values(), full\_df['target'])

ax = temp\_df.plot(kind='bar',fontsize=15,figsize=[20,15])

ax.set\_title('\nOutcomes Grouped by Employment Variation Rate\n', fontsize='30')

ax.set\_xlabel('\nEmployment Variation Rate', fontsize='30')

ax.set\_ylabel('Number of Observations', fontsize='30')

ax.legend(fontsize='40')

plt.xticks(size=30)

plt.xticks(rotation=0)

#----- Create a histogram showing relationship between CONS\_PRICE\_IDX and target variable

temp\_df = pd.crosstab(full\_df['cons\_price\_idx'].sort\_values(), full\_df['target'])

ax = temp\_df.plot(kind='bar',fontsize=15,figsize=[20,15])

ax.set\_title('\nOutcomes Grouped by Consumer Price Index\n', fontsize='30')

ax.set\_xlabel('\nConsumer Price Index', fontsize='30')

ax.set\_ylabel('Number of Observations', fontsize='30')

ax.legend(fontsize='40')

plt.xticks(size=20)

#----- Create a histogram showing relationship between CONS\_CONF\_IDX and target variable

temp\_df = pd.crosstab(full\_df['cons\_conf\_idx'].sort\_values(), full\_df['target'])

ax = temp\_df.plot(kind='bar',fontsize=15,figsize=[20,15])

ax.set\_title('\nOutcomes Grouped by Consumer Confidence Index\n', fontsize='30')

ax.set\_xlabel('\nConsumer Confidence Index', fontsize='30')

ax.set\_ylabel('Number of Observations', fontsize='30')

ax.legend(fontsize='40')

plt.xticks(size=20)

#----- Create a histogram showing relationship between EURIBOR3M and target variable

temp\_df = pd.crosstab(full\_df['euribor3m'].sort\_values(), full\_df['target'])

ax = temp\_df.plot(kind='bar',fontsize=15,figsize=[20,15])

ax.set\_title('\nOutcomes Grouped by Euribor 3-Month Rate\n', fontsize='30')

ax.set\_xlabel('\nEuribor 3-Month Rate', fontsize='30')

ax.set\_ylabel('Number of Observations', fontsize='30')

ax.legend(fontsize='40')

plt.xticks(size=20)

# For readability, only display every 100th tick on the X-axix.

for index, label in enumerate(ax.xaxis.get\_ticklabels()):

if index % 8 != 0:

label.set\_visible(False)

#----- Create a histogram showing relationship between NR\_EMPLOYED and target variable

temp\_df = pd.crosstab(full\_df['nr\_employed'].sort\_values(), full\_df['target'])

ax = temp\_df.plot(kind='bar',fontsize=15,figsize=[20,15])

ax.set\_title('\nOutcomes Grouped by Quarterly Employment Indicator\n', fontsize='30')

ax.set\_xlabel('\nQuarterly Employment Indicator', fontsize='30')

ax.set\_ylabel('Number of Observations', fontsize='30')

ax.legend(fontsize='40')

plt.xticks(size=20)

**Explore the Categorical Features**

#----- Create a histogram showing relationship between JOB and target variable

temp\_df = pd.crosstab(full\_df['job'].sort\_values(), full\_df['target'])

ax = temp\_df.plot(kind='bar',fontsize=15,figsize=[20,15])

ax.set\_title('\nOutcomes Grouped by Job Type\n', fontsize='30')

ax.set\_xlabel('Job Type', fontsize='30')

ax.set\_ylabel('Number of Observations', fontsize='30')

ax.legend(fontsize='40')

plt.xticks(size=20)

# ----- What is the proportion of UNKNOWN value in the JOB feature?

print(full\_df.groupby(['job']).size() / len(full\_df) \* 100)

#----- Create a histogram showing relationship between MARITAL and target variable

temp\_df = pd.crosstab(full\_df['marital'].sort\_values(), full\_df['target'])

ax = temp\_df.plot(kind='bar',fontsize=15,figsize=[20,15])

ax.set\_title('\nOutcomes Grouped by Marital Status\n', fontsize='30')

ax.set\_xlabel('\nMarital Status', fontsize='30')

ax.set\_ylabel('Number of Observations', fontsize='30')

ax.legend(fontsize='40')

plt.xticks(size=30)

plt.xticks(rotation=0)

#----- What is the proportion of UNKNOWN value in the MARITAL feature?

print(full\_df.groupby(['marital']).size() / len(full\_df) \* 100)

#----- Create a histogram showing relationship between EDUCATION and target variable

temp\_df = pd.crosstab(full\_df['education'].sort\_values(), full\_df['target'])

ax = temp\_df.plot(kind='bar',fontsize=15,figsize=[20,15])

ax.set\_title('\nOutcomes Grouped by Education Level\n', fontsize='30')

ax.set\_xlabel('\nEducation Level', fontsize='30')

ax.set\_ylabel('Number of Observations', fontsize='30')

ax.legend(fontsize='40')

plt.xticks(size=30)

plt.xticks(rotation=70)

#----- What is the proportion of UNKNOWN value in the EDUCATION feature?

print(full\_df.groupby(['education']).size() / len(full\_df) \* 100)

#----- Create a histogram showing relationship between DEFAULT and target variable

temp\_df = pd.crosstab(full\_df['default'].sort\_values(), full\_df['target'])

ax = temp\_df.plot(kind='bar',fontsize=15,figsize=[20,15])

ax.set\_title('\nOutcomes Grouped by Credit Default Status\n', fontsize='30')

ax.set\_xlabel('\nDoes Client Have Credit in Default?', fontsize='30')

ax.set\_ylabel('Number of Observations', fontsize='30')

ax.legend(fontsize='40')

plt.xticks(size=30)

plt.xticks(rotation=0)

#----- What is the proportion of UNKNOWN value in the DEFAULT feature?

print(full\_df.groupby(['default']).size() / len(full\_df) \* 100)

#----- Create a histogram showing relationship between HOUSING and target variable

temp\_df = pd.crosstab(full\_df['housing'].sort\_values(), full\_df['target'])

ax = temp\_df.plot(kind='bar',fontsize=15,figsize=[20,15])

ax.set\_title('\nOutcomes Grouped by Housing Loan Status\n', fontsize='30')

ax.set\_xlabel('\nDoes Client Have a Housing Loan?', fontsize='30')

ax.set\_ylabel('Number of Observations', fontsize='30')

ax.legend(fontsize='40')

plt.xticks(size=30)

plt.xticks(rotation=0)

#----- What is the proportion of UNKNOWN value in the HOUSING feature?

print(full\_df.groupby(['housing']).size() / len(full\_df) \* 100)

#----- Create a histogram showing relationship between LOAN and target variable

temp\_df = pd.crosstab(full\_df['loan'].sort\_values(), full\_df['target'])

ax = temp\_df.plot(kind='bar',fontsize=15,figsize=[20,15])

ax.set\_title('\nOutcomes Grouped by Personal Loan Status\n', fontsize='30')

ax.set\_xlabel('\nDoes Client Have a Personal Loan?', fontsize='30')

ax.set\_ylabel('Number of Observations', fontsize='30')

ax.legend(fontsize='40')

plt.xticks(size=30)

plt.xticks(rotation=0)

#----- What is the proportion of UNKNOWN value in the LOAN feature?

print(full\_df.groupby(['loan']).size() / len(full\_df) \* 100)

#----- Create a histogram showing relationship between CONTACT and target variable

temp\_df = pd.crosstab(full\_df['contact'].sort\_values(), full\_df['target'])

ax = temp\_df.plot(kind='bar',fontsize=15,figsize=[20,15])

ax.set\_title('\nOutcomes Grouped by Contact Communication Type\n', fontsize='30')

ax.set\_xlabel('\nContact Communication Type', fontsize='30')

ax.set\_ylabel('Number of Observations', fontsize='30')

ax.legend(fontsize='40')

plt.xticks(size=30)

plt.xticks(rotation=0)

#----- Create a histogram showing relationship between MONTH and target variable

temp\_df = pd.crosstab(full\_df['month'].sort\_values(), full\_df['target'])

ax = temp\_df.plot(kind='bar',fontsize=15,figsize=[20,15])

ax.set\_title('\nOutcomes Grouped by Contact Month\n', fontsize='30')

ax.set\_xlabel('\nContact Month', fontsize='30')

ax.set\_ylabel('Number of Observations', fontsize='30')

ax.legend(fontsize='40')

plt.xticks(size=30)

plt.xticks(rotation=0)

#----- Create a histogram showing relationship between DAY\_OF\_WEEK and target variable

temp\_df = pd.crosstab(full\_df['day\_of\_week'].sort\_values(), full\_df['target'])

ax = temp\_df.plot(kind='bar',fontsize=15,figsize=[20,15])

ax.set\_title('\nOutcomes Grouped by Contact Day of Week\n', fontsize='30')

ax.set\_xlabel('\nContact Day of Week', fontsize='30')

ax.set\_ylabel('Number of Observations', fontsize='30')

ax.legend(fontsize='40')

plt.xticks(size=30)

plt.xticks(rotation=0)

#----- Create a histogram showing relationship between POUTCOME and target variable

temp\_df = pd.crosstab(full\_df['poutcome'].sort\_values(), full\_df['target'])

ax = temp\_df.plot(kind='bar',fontsize=15,figsize=[20,15])

ax.set\_title('\nOutcomes Grouped by Outcome of the Previous Campaign\n', fontsize='30')

ax.set\_xlabel('\nOutcome of the Previous Marketing Campaign', fontsize='30')

ax.set\_ylabel('Number of Observations', fontsize='30')

ax.legend(fontsize='40')

plt.xticks(size=30)

plt.xticks(rotation=0)

#----- What is the proportion of NONEXISTANT value in the POUTCOME feature?

print(full\_df.groupby(['poutcome']).size() / len(full\_df) \* 100)

**Explore Correlation between the Features**

#----- Generate a correlation matrix to get a sense of the correlation between the numerical features.

rs = np.random.RandomState(0)

corr = full\_df.corr()

corr.style.background\_gradient()

**Drop Some Features from the Dataset**

#----- Drop some features from the dataset because they were deemed to have no value.

del full\_df['pdays']

del full\_df['default']

**Transform Categorical Features into Numerical Dummy Variables**

#----- Create a new dataframe called 'dummies\_df' and convert

#----- all categorical variables to numerical variables.

#-----

#----- Add a prefix to each new column for ease of identification.

#----- Drop first column of each variable in order to minimize the dimensionality of the dataset.

#-----

dummies\_df = pd.get\_dummies(full\_df, prefix='job\_', columns=['job'], drop\_first=True)

dummies\_df = pd.get\_dummies(dummies\_df, prefix='marital\_', columns=['marital'], drop\_first=True)

dummies\_df = pd.get\_dummies(dummies\_df, prefix='education\_', columns=['education'], drop\_first=True)

dummies\_df = pd.get\_dummies(dummies\_df, prefix='housing\_', columns=['housing'], drop\_first=True)

dummies\_df = pd.get\_dummies(dummies\_df, prefix='loan\_', columns=['loan'], drop\_first=True)

dummies\_df = pd.get\_dummies(dummies\_df, prefix='contact\_', columns=['contact'], drop\_first=True)

dummies\_df = pd.get\_dummies(dummies\_df, prefix='month\_', columns=['month'], drop\_first=True)

dummies\_df = pd.get\_dummies(dummies\_df, prefix='day\_of\_week\_',columns=['day\_of\_week'],drop\_first=True)

dummies\_df = pd.get\_dummies(dummies\_df, prefix='poutcome\_', columns=['poutcome'], drop\_first=True)

dummies\_df = pd.get\_dummies(dummies\_df, columns=['target'], drop\_first=True)

#----- The name of the "target" variable has been changed to "target\_yes".

#----- Restore the name to "target" (just for the sake of cleanliness).

dummies\_df.rename(columns={'target\_yes':'target'}, inplace=True)

#----- Display the structure of the 'dummies\_df' dataframe.

dummies\_df.dtypes

#----- Peek at the first few lines of the data frame to confirm that the column values are as expected.

dummies\_df.sample(5)

**Standardize the Features**

#----- Standardize all of the features in the dummies\_df dataframe.

#----- The result will be placed in a numpy array.

scaler = StandardScaler()

scaled\_array = scaler.fit\_transform(dummies\_df)

#----- Convert the numpy array to a dataframe.

#----- The index and columns keyword arguments are used to keep the original indices and column names.

scaled\_df = pd.DataFrame(scaled\_array, index=dummies\_df.index, columns=dummies\_df.columns)

#----- Peek at the data to confirm that the scaling process was successful.

scaled\_df.sample(5)

**Handle the Unbalanced Data Using Random Under-Sampling**

#----- The target variable was scaled in the previous steps. Convert its' values back to '0' and '1' because

#----- working with 0/1 is more intuitive than working with -0.35/2.80 and the columns will need to contain

#----- integers for the predictive models.

scaled\_df['target'][(scaled\_df['target'] < 0)] = 0

scaled\_df['target'][(scaled\_df['target'] > 0)] = 1

scaled\_df['target'] = scaled\_df['target'].apply(int)

#----- Generate a bar chart showing the current distribution of the target variable.

scaled\_df.target.value\_counts().plot(kind='bar');

#plt.plot(scaled\_df)

plt.title('Distribution of Target Variable (Prior to Rebalancing)\n',size=25)

plt.ylabel('Number of Occurrences\n',size=20)

plt.xlabel('Target Variable\n',size=20)

plt.xticks(fontsize=20)

plt.yticks(fontsize=15)

plt.rcParams['figure.figsize'] = [7, 7]

#----- Create a new dataframe to hold the rows for each target value ('0' and '1')

target\_0\_df = scaled\_df[scaled\_df['target'] == 0]

target\_1\_df = scaled\_df[scaled\_df['target'] == 1]

#----- Use the undersampling technique to reduce the number of '0' rows.

#-----

#----- Undersampling means that we will reduce the number of rows with target = '0' to n rows,

#----- where n is the number of rows with a target of '1'.

#----- In this dataset, there are 4640 rows with a target of '1'. Therefore, the number of rows

#----- with a target of '0' will be reduced to a random sample of 4640 rows.

count\_target\_1 = target\_1\_df.target.count() # Count the number of '1' rows.

target\_0\_df\_under = target\_0\_df.sample(count\_target\_1) # Reduce the number of '0' rows.

#----- Concatenate the rows with an outcome of '1' with the reduced number of rows with an outcome of '0'.

#----- Place the results in a dataframe called 'balanced\_df'.

balanced\_df = pd.concat([target\_0\_df\_under, target\_1\_df], axis=0)

#----- Display the shape of the dataframe that has a balanced target column.

print(balanced\_df.shape)

#----- Generate a bar chart showing the new distribution of the target variable.

#----- The distribution is now balanced.

balanced\_df.target.value\_counts().plot(kind='bar');

plt.plot(balanced\_df)

plt.title('Balanced Distribution of Target Variable\n',size=25)

plt.ylabel('Number of Occurrences\n',size=20)

plt.xlabel('Target Variable\n',size=20)

plt.xticks(fontsize=20)

plt.yticks(fontsize=15)

plt.rcParams['figure.figsize'] = [7, 7]