Practical Data Science

Assignment 1

Chi Ting, Low

S3611774

Table of Contents

Data Preparation

The aim of the current report is to conduct data cleaning and data exploration from the given dataset using Python 2.7 (Python Software Foundation, 2018). The dataset used is related to direct marketing campaigns of a Portuguese banking institution. This campaign was conducted based on phone calls in order to access a client will subscribed to their product or not.

Attributes information:

- age – client age (numeric)
- job – type of job (categorical)
- marital – marital status (categorical)
- education – education status (categorical)
- default – credit default history (categorical)
- housing – has housing loan (categorical)
- contact – communication type (categorical)
- month – month of last contact (categorical)
- day_of_week -day of the week of last contact (categorical)
- duration – contact duration (numeric)
- campaign – number of contact performed (numeric)
- pdays – number of day passed after last contact (numeric)
- previous – number of contact before the campaign (numeric)
- poutcome – outcome of previous campaign (categorical)
- emp.var.rate – employment variation rate, quarterly indicator (numeric)
- cons.price.idx – consumer price index, monthly indicator (numeric)
- cons.conf.idx – consumer confidence index, monthly indicator (numeric)

- euribor3m – euribor 3 month rate, daily indicator (numeric)
- nr.employed – number of employees, quarterly indicator (numeric)
- y – has the client subscribed a term deposit (categorical)

Task 1.1

To read the data, pandas packages (McKinney, 2010) is called and pd.read_csv function is used along with arguments (sep, decimal, header, names and na_values) to load the data into python environment. The argument *sep* is used to separate stacked values in the data set. The argument *decimal* is used to recognize the values into decimal values. The argument *header* is used to set the column names. In current report, the *header* is set to 0 which allows the column names to be replaced (references). The *name* argument is used to rename the columns names. Lastly, the *na_values* argument is used to set the missing values to string NaN.

Task 1.2

After loading dataset into python environment, *.dtypes* method is used to access the types of variables assigned to each variable (Boschetti & Massaron, 2015). As observed, some variables are assigned as object. This indicated that the variables do not assigned properly into the data frame except age, duration, campaign, cons_price_idx and cons_conf_idx which are assigned into float64 (numeric with double decimal point). To convert the variables to the data types presented in the attribute information above two methods is called (*.to_numeric* and *.astype*). The *.to_numeric* is used to convert the object variable into numerical data type. The *.astype* is used to convert the variable into categorical data types.

Task 1.3

To examine the typos in the data due to imputation error method *.value_counts*() is used. The types are found in the marital, education, default, housing, loan, contact, month, day_of_week, pdays, previous, poutcome, emp_var_rate, cons_price_idx, cons_conf_idx, euribor3m, and nr_employed variables. These variables contain strings that is not assigned or contain no meaningful information for the variables. By using masks method, these invalid strings are set into NaN.

Task 1.4

To examine whether there are instances of extra whitespaces in the data, method *str.strip* is called on the categorical variables. For example, *data_copy["job"].str.strip()* which is used to omit the whitespace in the job column in the data frame.

Task 1.5

To set the text data into lower case method *.str.lower()* is used to set the text data into lower-case. For example in *data_copy["marital"].str.lower(),* this allows the method to be called and search the marital variables columns and set all the row into lower case.

Task 1.6

To check whether there are impossible values for each variable in the data set, *mean* method is used to perform a series of sanity checks to filter out impossible values. In the variable age, the mean age is 40.112. However, in the row 385 (age = 143) and 394 (age = 250) the age is above mean age by large margin and the chance of an individual to live longer than 143 and 250 is slim.

In addition, in the campaign variables which indicated the number of contact to client during campaign have the mean of 3.042. This indicated that on average each client will receive 3 calls. But, in row 569, 650 and 678 shows that a person has received 999 and 93.994 call which indicated this is impossible. Similar approach is conducted to the variables; previous (mean = 0.191), emp.var.rate (mean = 0.085), cons.price.idx (mean = 93.558), cons.conf.idx (mean = -

40.499), euribor3m (mean = 4.859) and nr.employed (mean = 5166.517) which shows that there are values beyond the mean or lower than mean. All these values is replaced as missing values.

Task 1.7

Lastly, to examine the missing values in the data set *.isnull().sum()* method is called to examine the total missing values in each variable. It shows that marital, education, default, and housing have one missing values; pdays has two missing values; loan, contact, month, day_of_week and campaign have 3 missing values; previous, poutcome, emp.var.rate, cons.conf.idx and y have 4 missing values; euribor3m and age have 5 missing values; cons.price.idx and nr_employed have 6 missing values after the process of data cleaning and replace all the typos and impossible values to missing values. To replace all the missing values method, *.fillna* and *.mean* to fill the missing value by using mean of each correspondent variables.

Extension

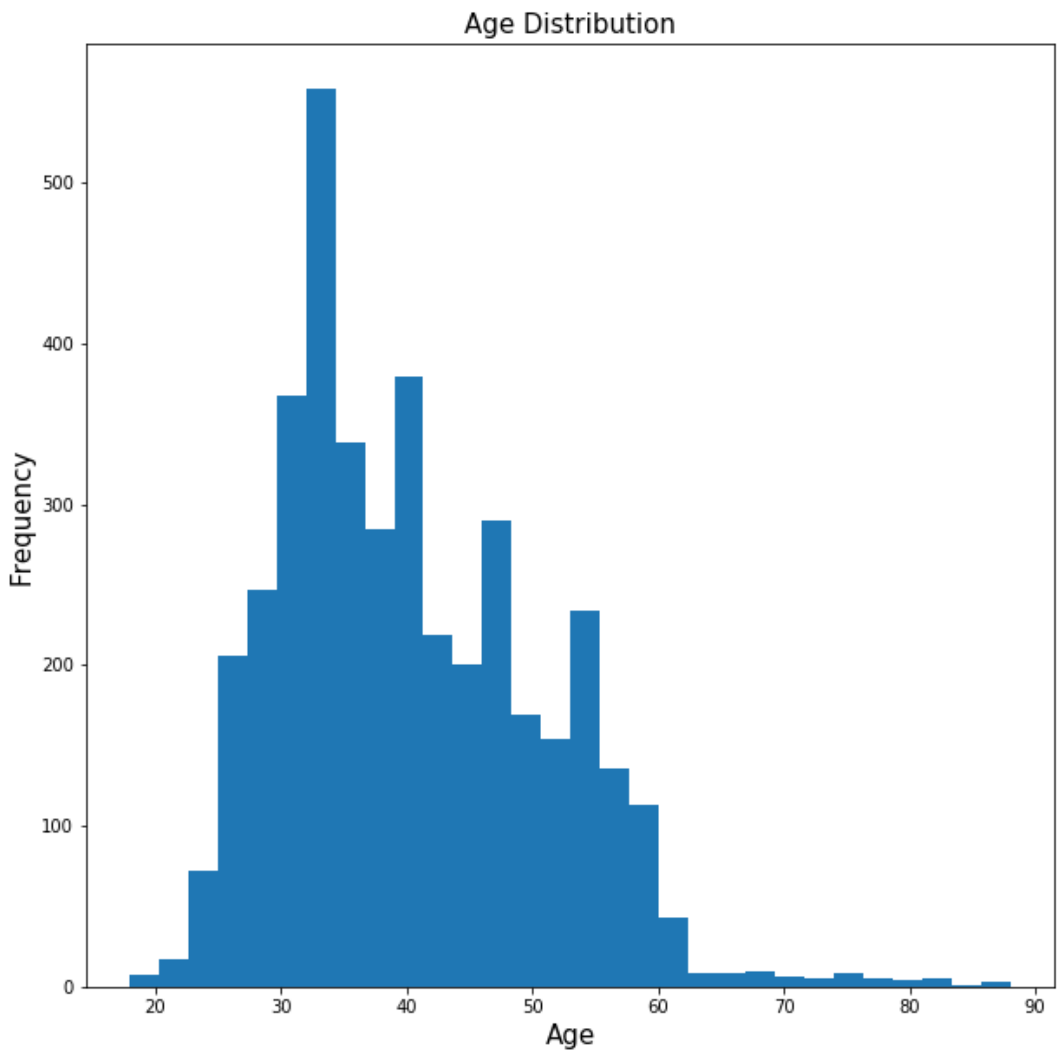
To replace all the missing values, three other approaches are used; median, using fix-value and dropping the missing values. Same approach is used to fill the missing values by using median of the correspondent variables. Lastly, using the method *.fillna* to fill all the missing values using fix value 9999 which is demonstrated in Task1.ipynb in line 96, 98, and 100.

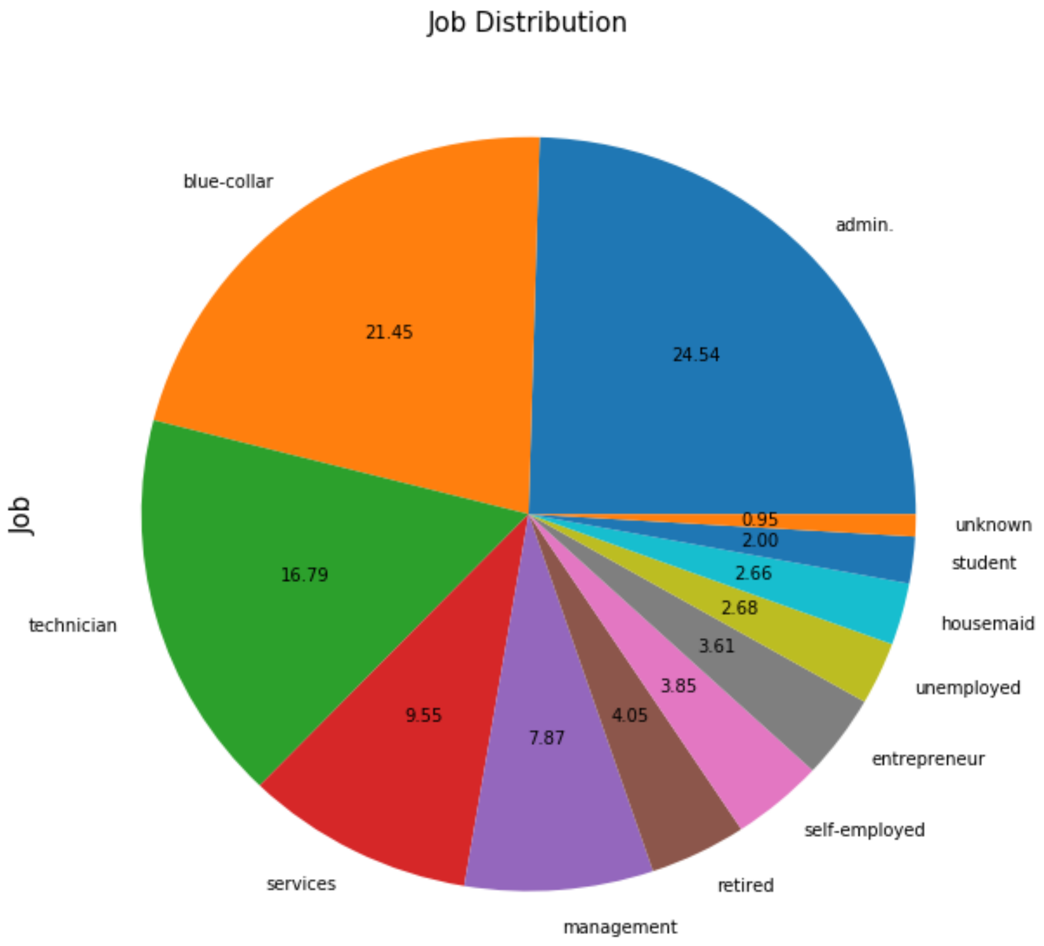
The last approach to deal with missing values is used *.dropna* method. This method is used to remove the entire rows that contain missing values from data frame. As this approach applied, the number of observation in the data set is drop from 4119 to 4103 as demonstrated in Task1.ipynb in line 119.

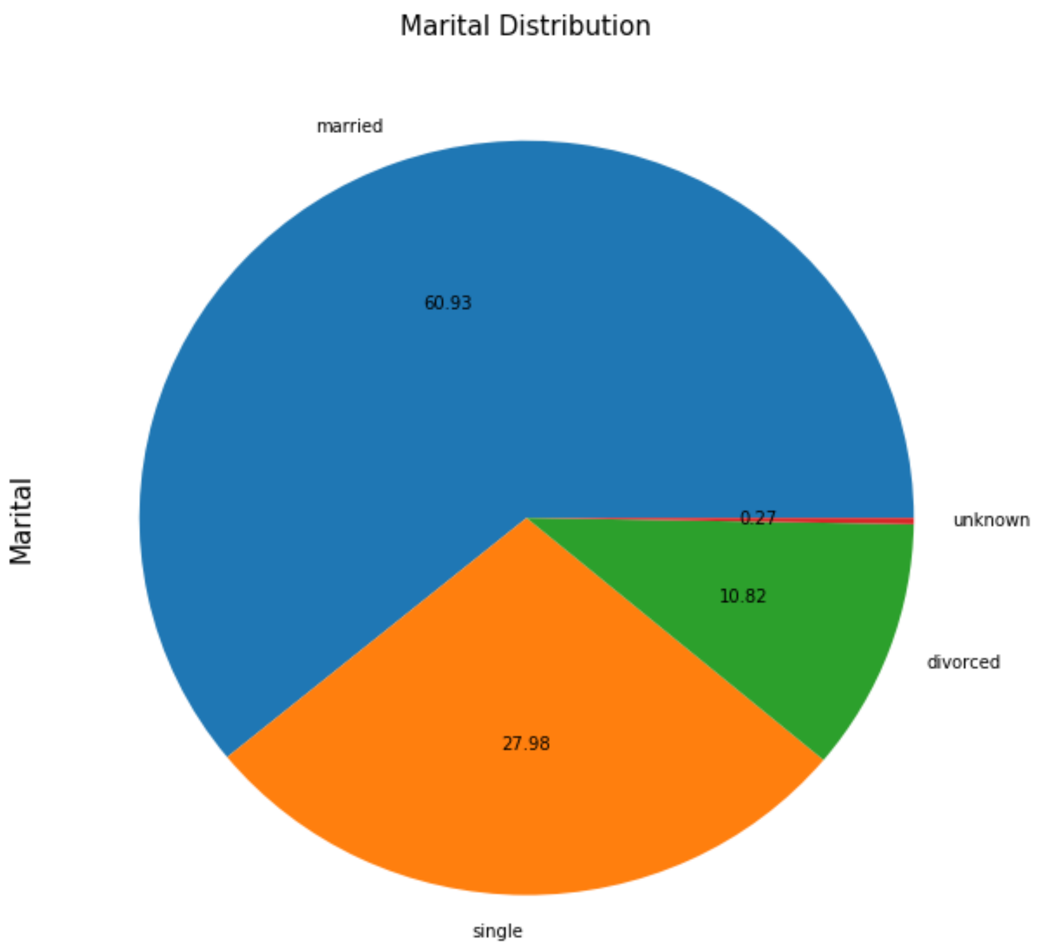
Data Exploration

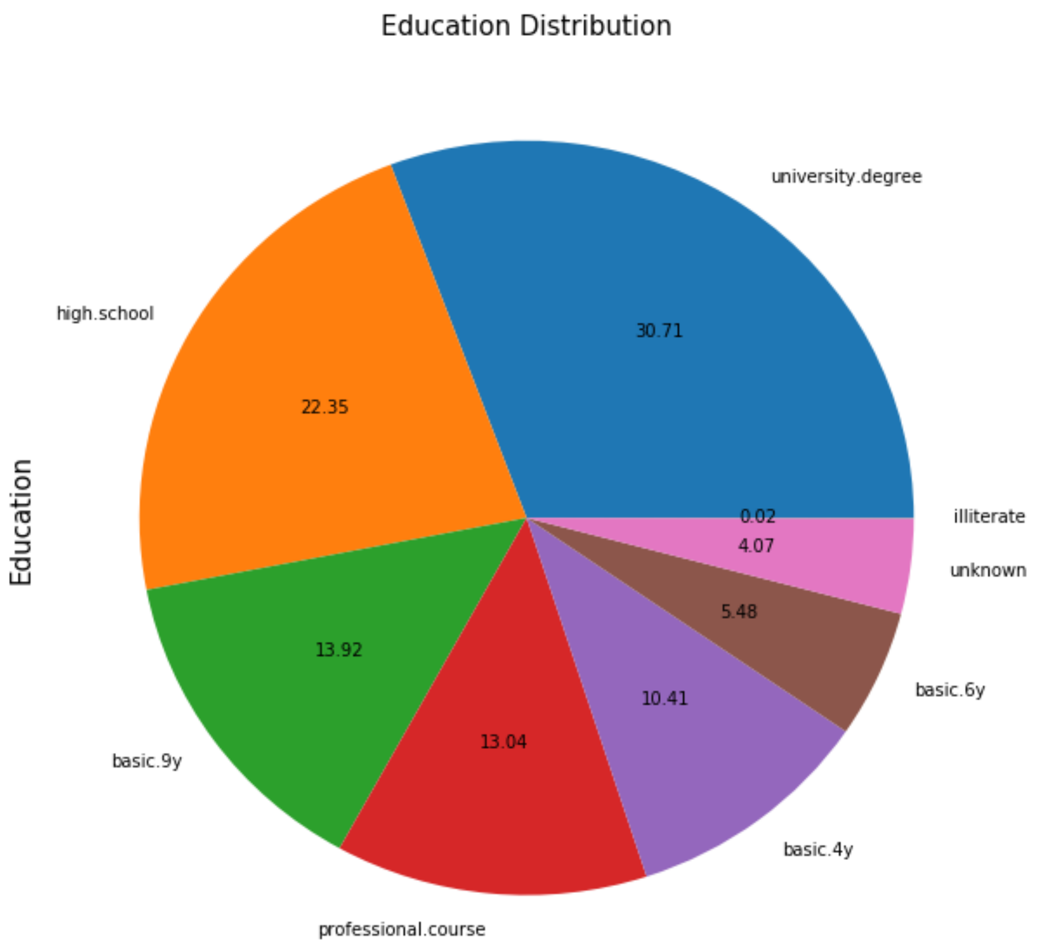
Task 2.1

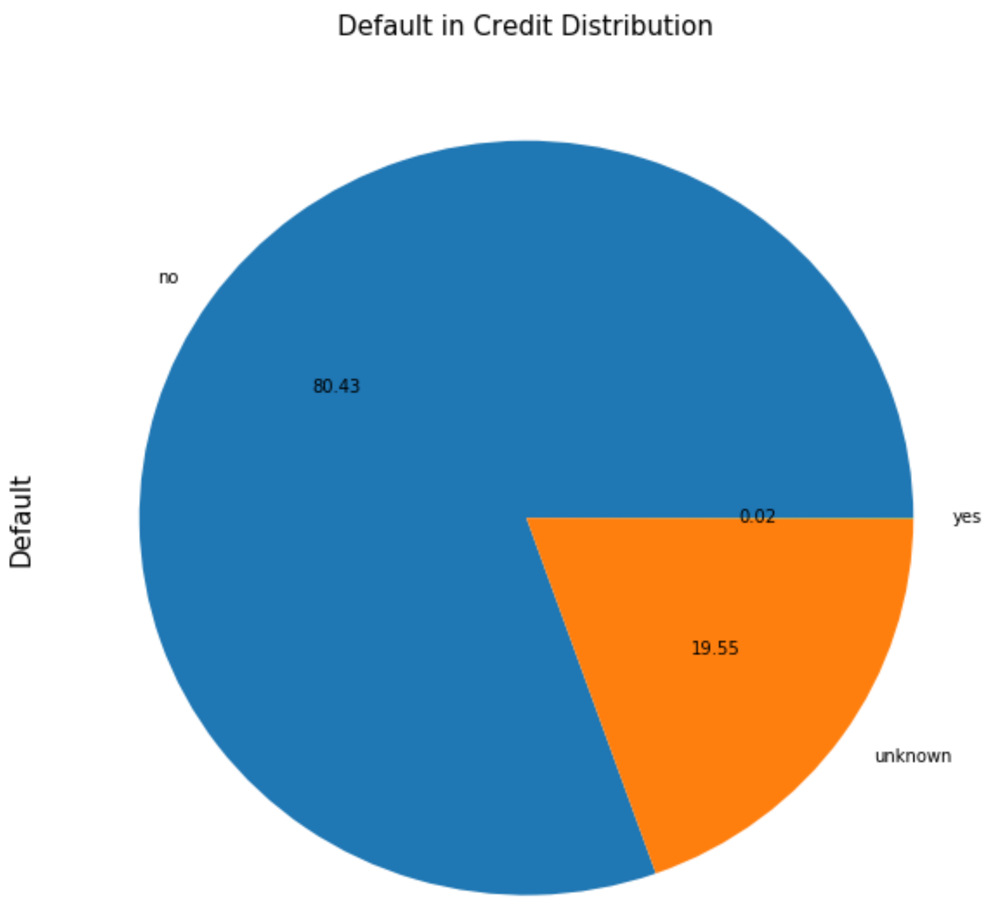
To conduct simple exploration of the data set, a basic visualization is performed for each variable using histogram, pie charts, bar charts and density plot for the data set.

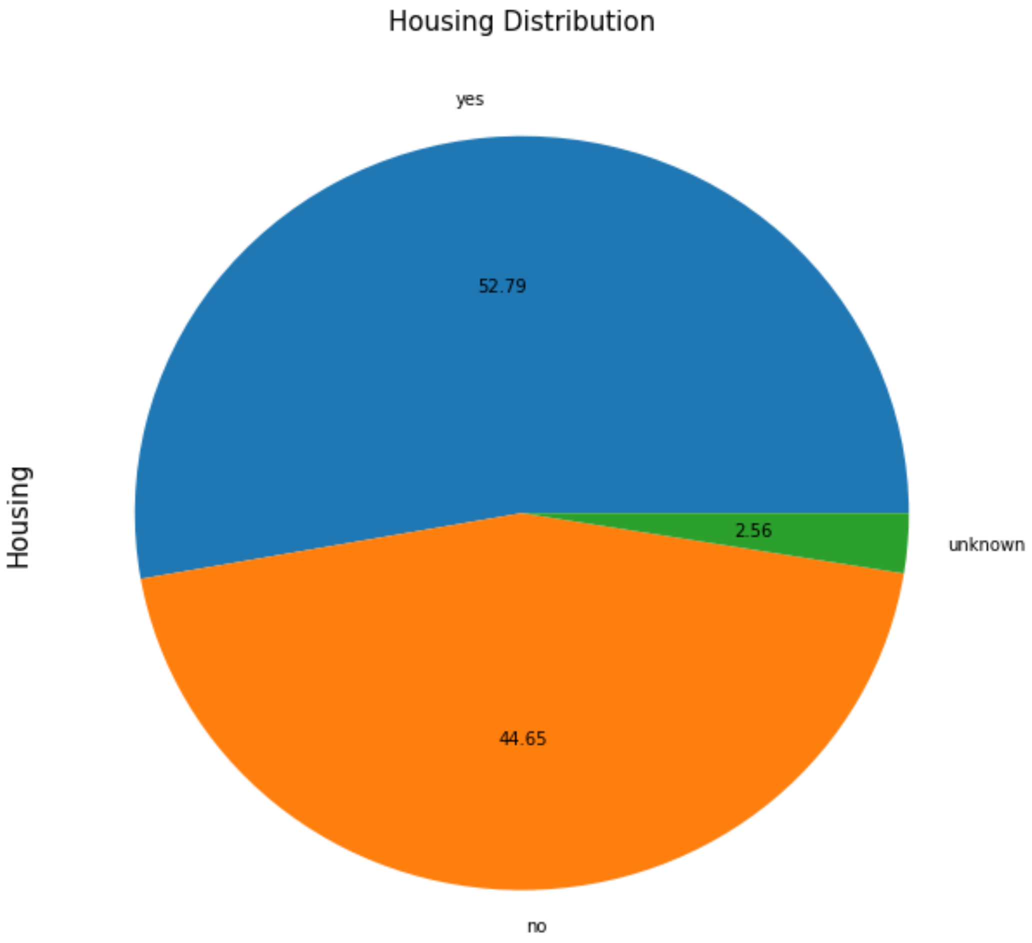


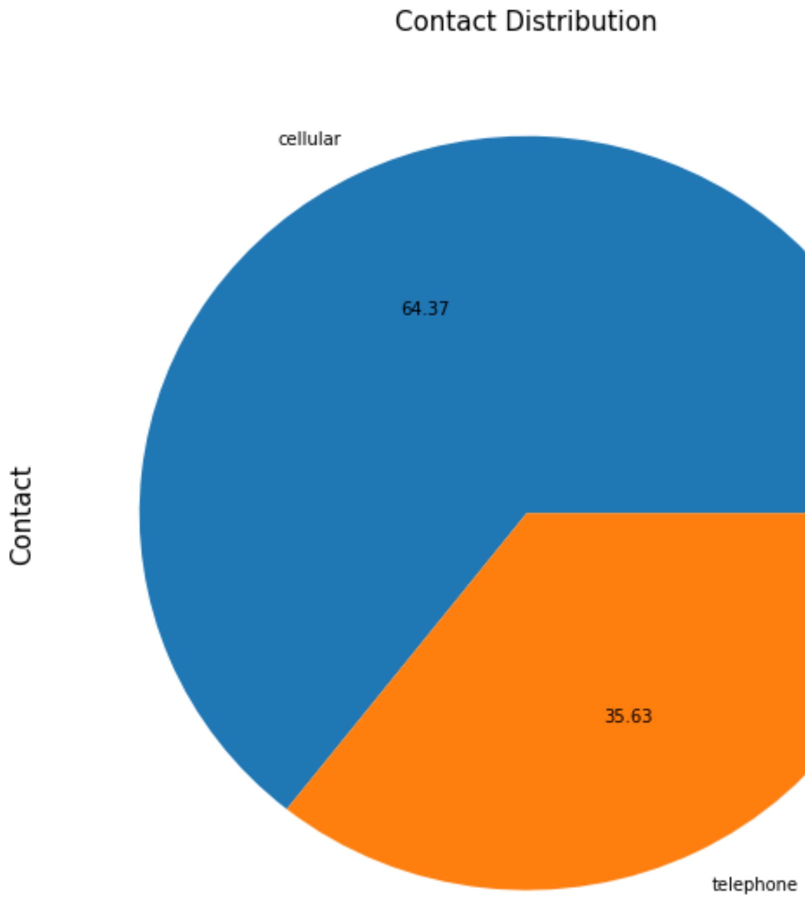

Job Distribution
Job
admin.
24.54
blue-collar
21.45
technician
16.79
services
9.55
management
7.87
retired
4.05
self-employed
3.85
entrepreneur
3.61
unemployed
2.68
housemaid
2.66
student
2.00
unknown
0.95

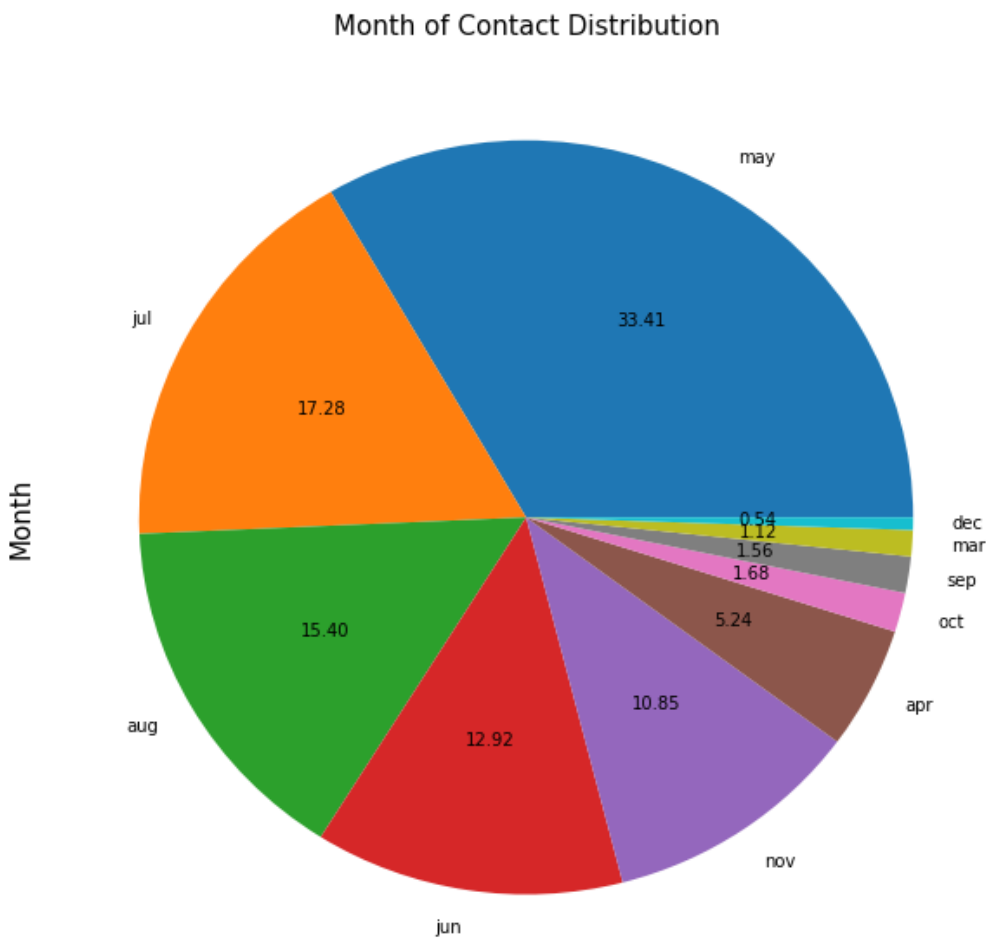

Marital Distribution
married
60.93
0.27
unknown
10.82
divorced
27.98
single
Marital

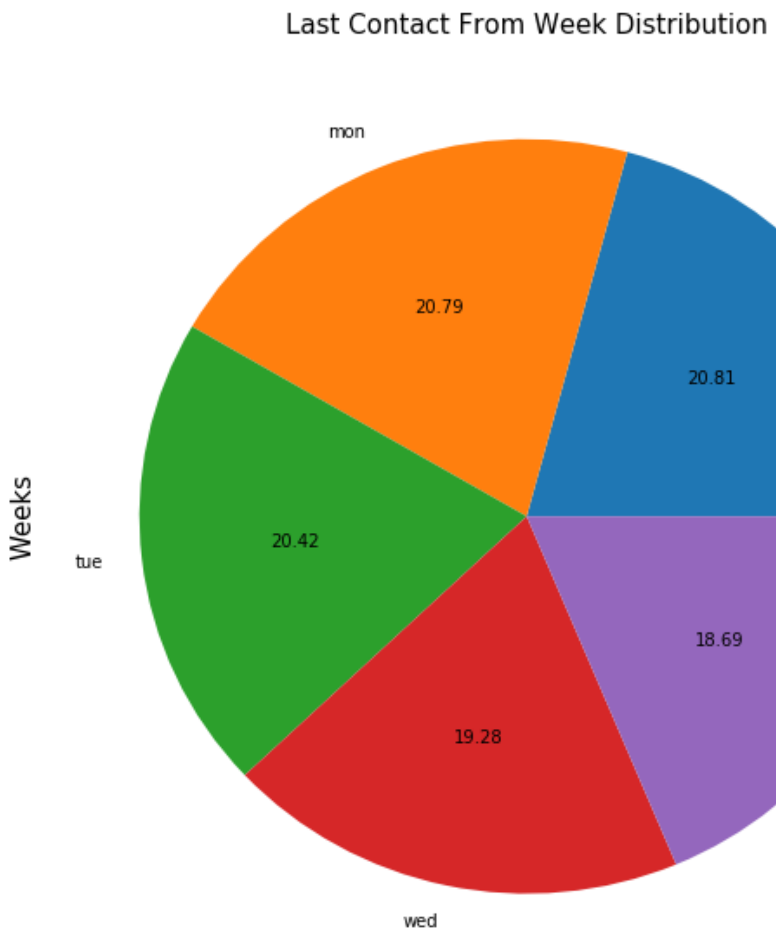

Education Distribution
Education
university.degree
30.71
high.school
22.35
basic.9y
13.92
professional.course
13.04
basic.4y
10.41
basic.6y
5.48
unknown
4.07
illiterate
0.02

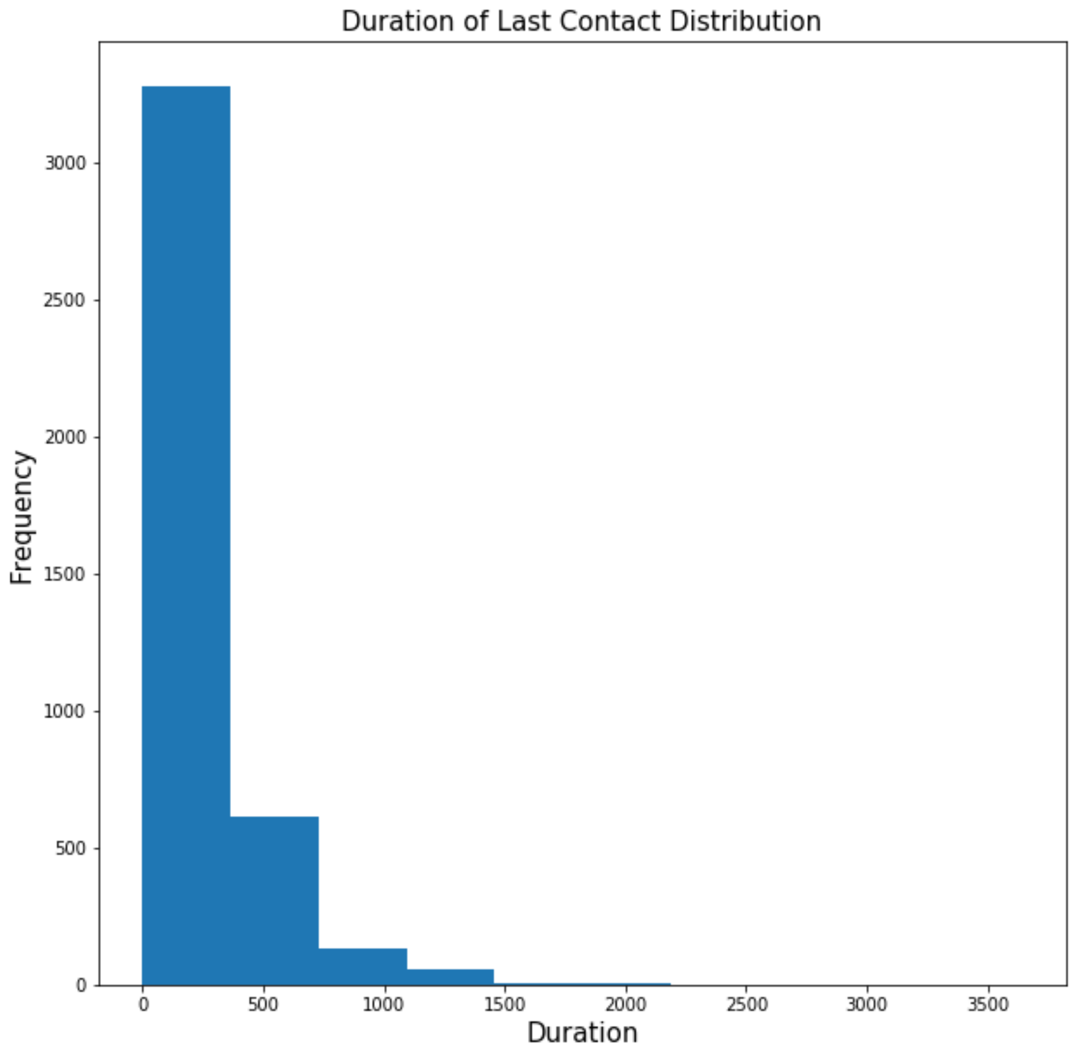

Default in Credit Distribution
Default
no
80.43
0.02
yes
19.55
unknown

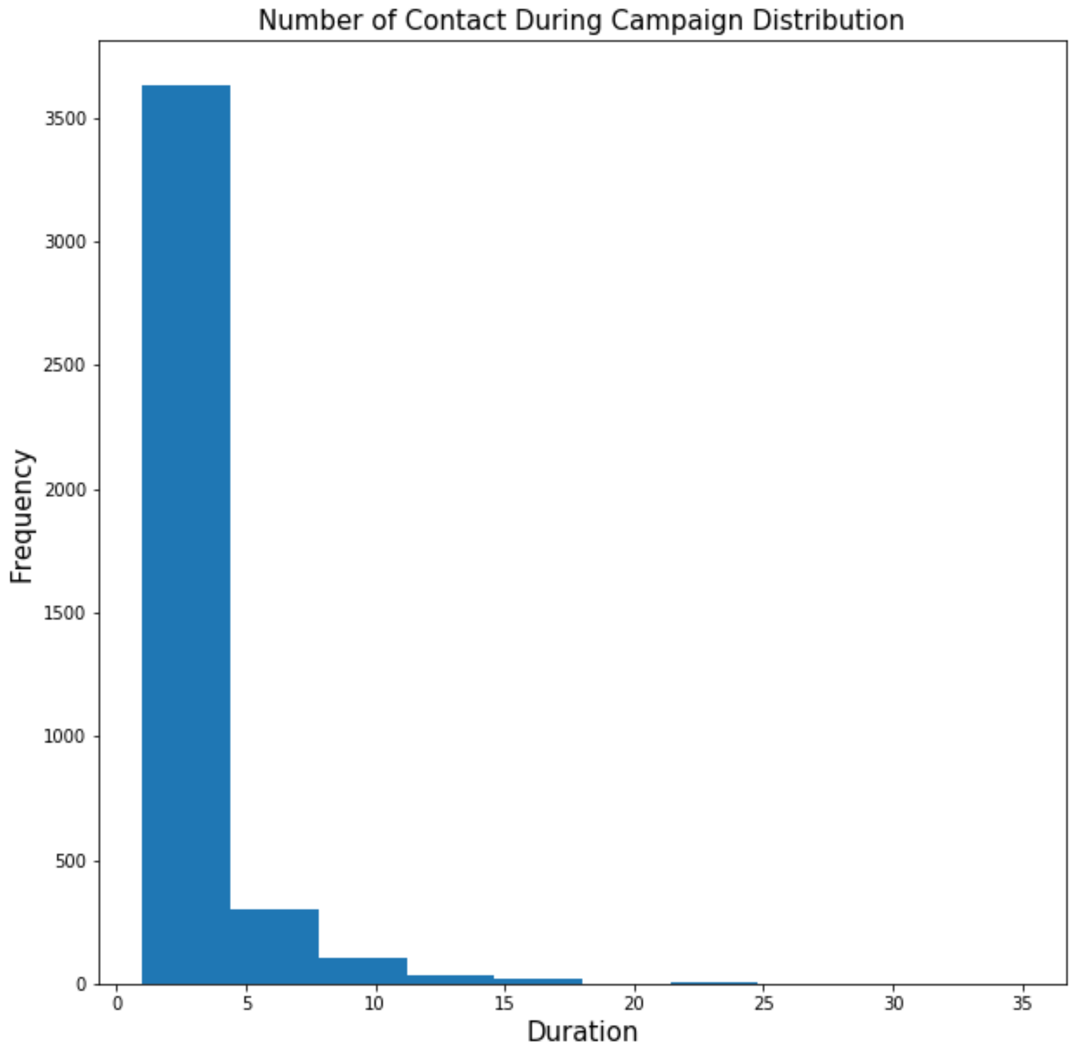

Housing Distribution
yes
52.79
Housing
2.56
unknown
44.65
no

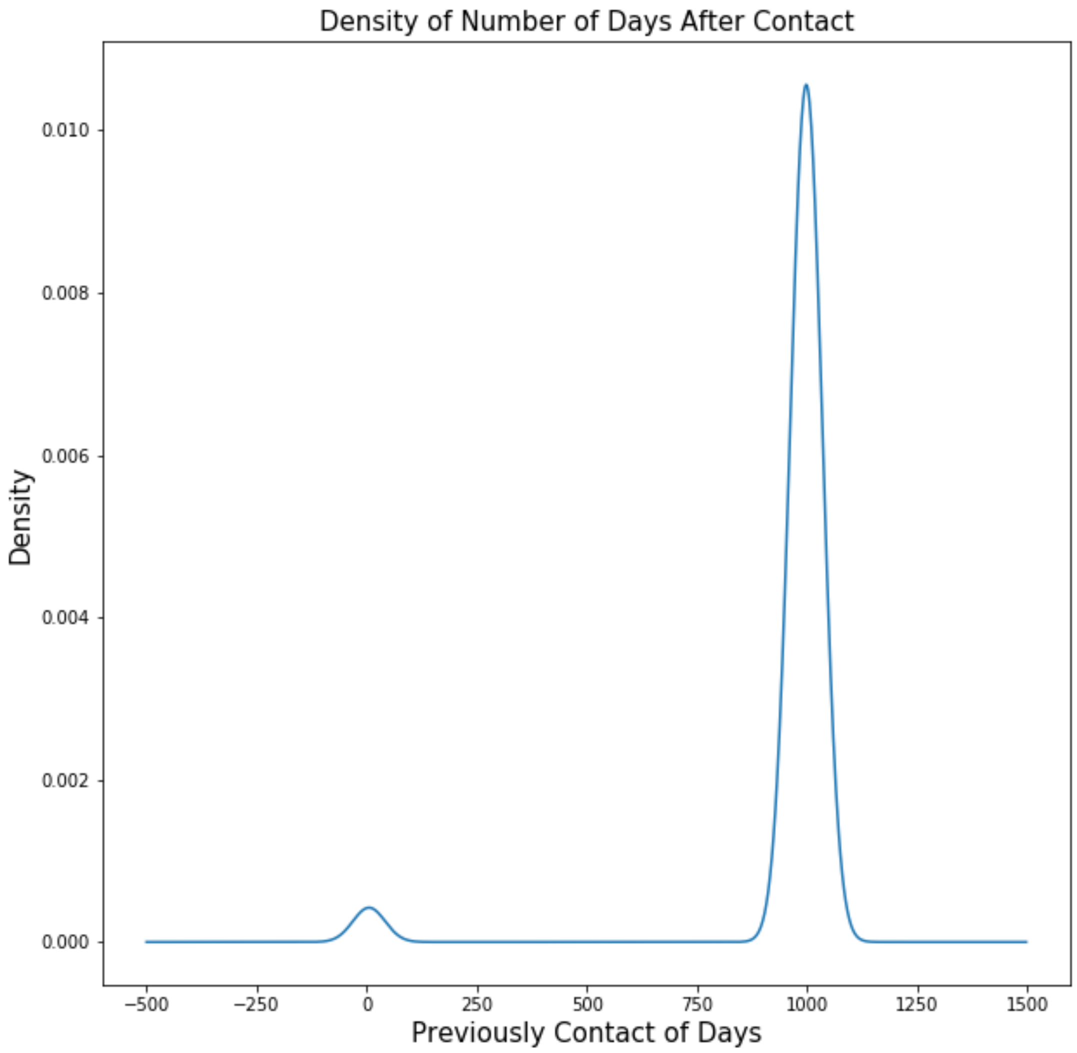

Contact Distribution
Contact
cellular
64.37
telephone
35.63

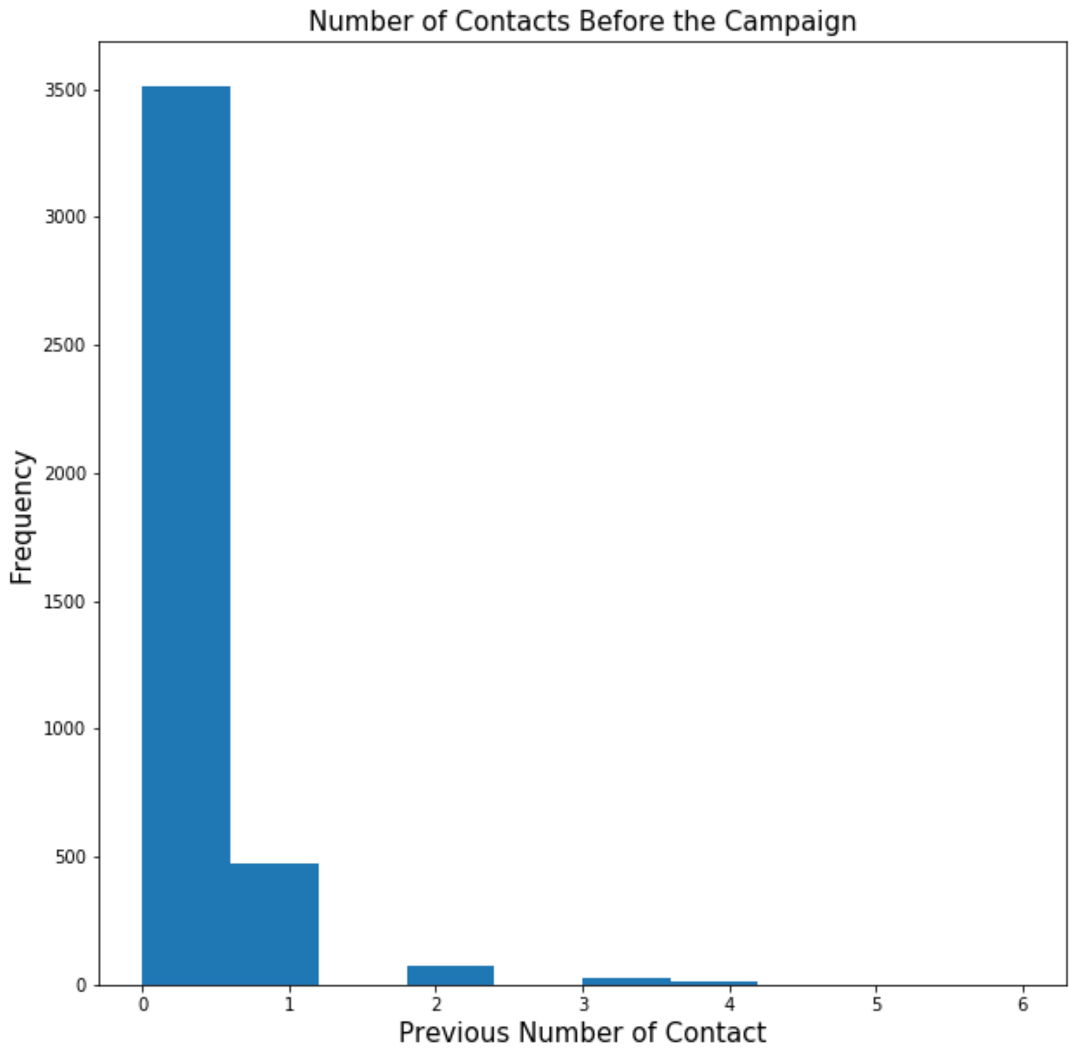

Month of Contact Distribution
Month
may
33.41
jul
17.28
aug
15.40
jun
12.92
nov
10.85
apr
5.24
oct
1.68
sep
1.56
mar
1.12
dec
0.54

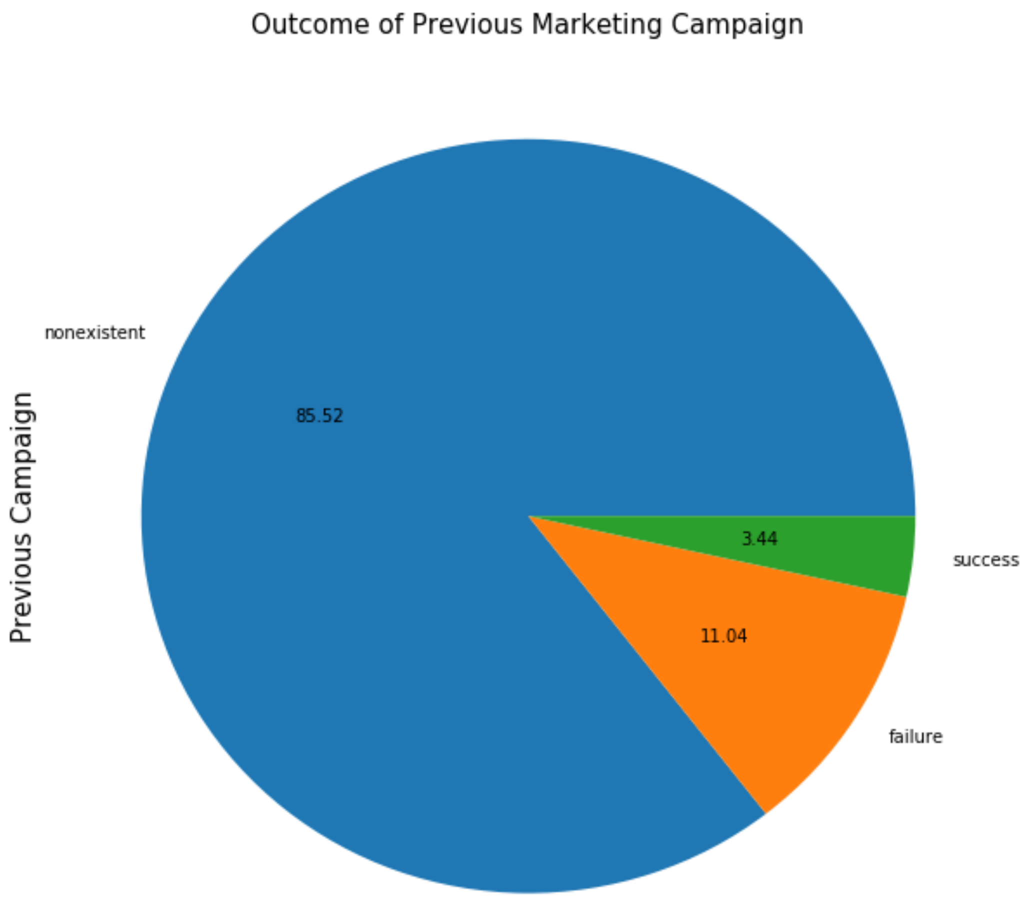

Last Contact From Week Distribution
Weeks
mon
20.79
thu
20.81
tue
20.42
18.69
fri
19.28
wed

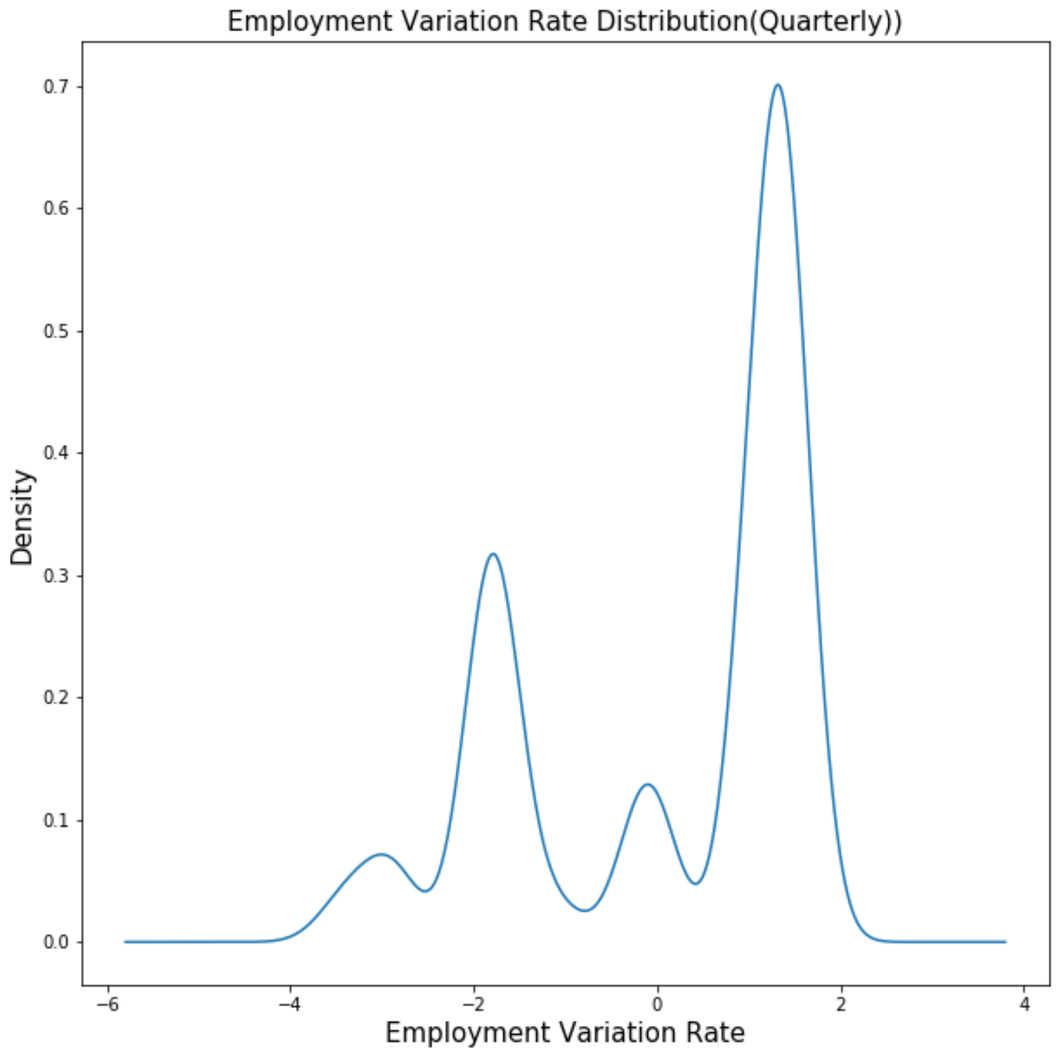

Duration of Last Contact Distribution
Frequency
Duration
0
500
1000
1500
2000
2500
3000
3500

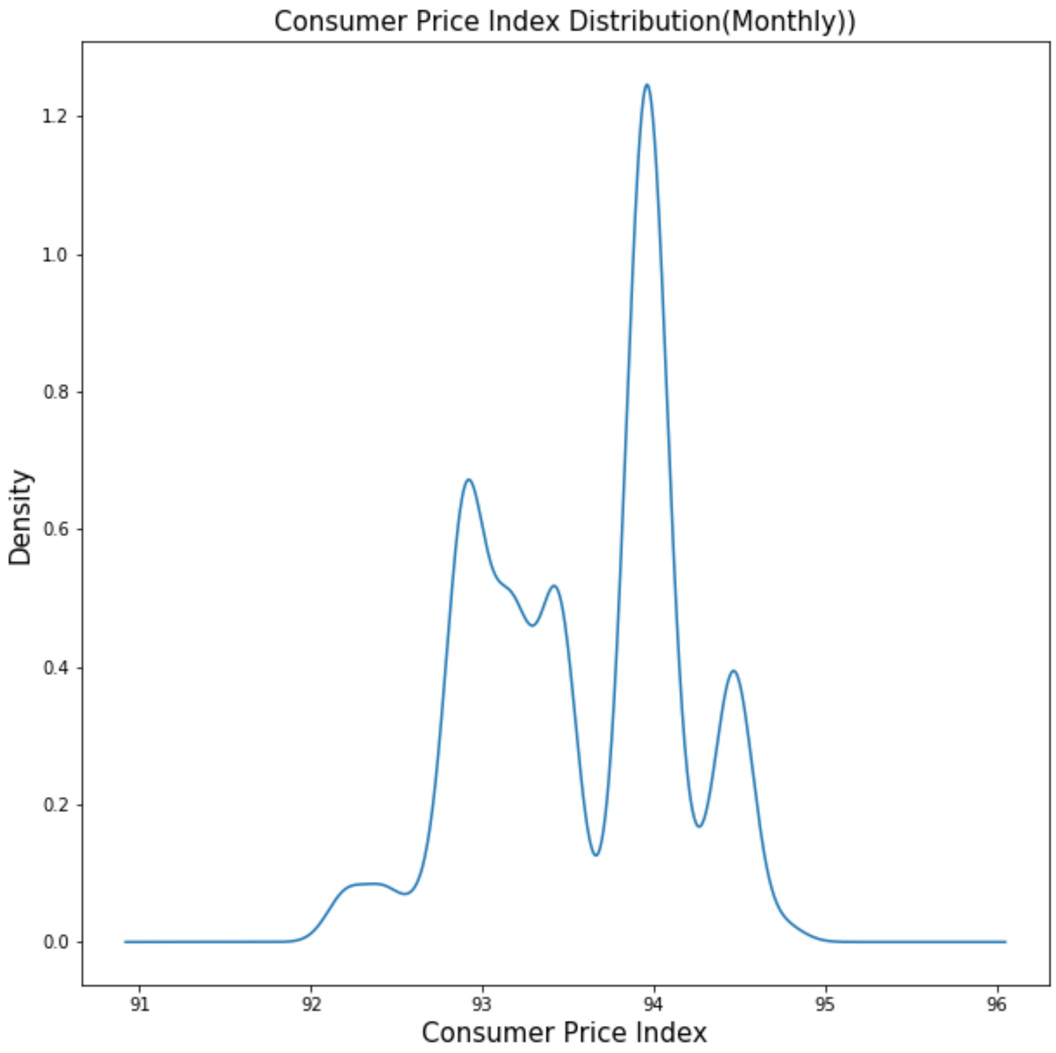

Number of Contact During Campaign Distribution
Frequency
Duration
0
500
1000
1500
2000
2500
3000
3500
0
5
10
15
20
25
30
35


Density of Number of Days After Contact
Density
Previously Contact of Days
0.010
0.008
0.006
0.004
0.002
0.000
−500
−250
0
250
500
750
1000
1250
1500

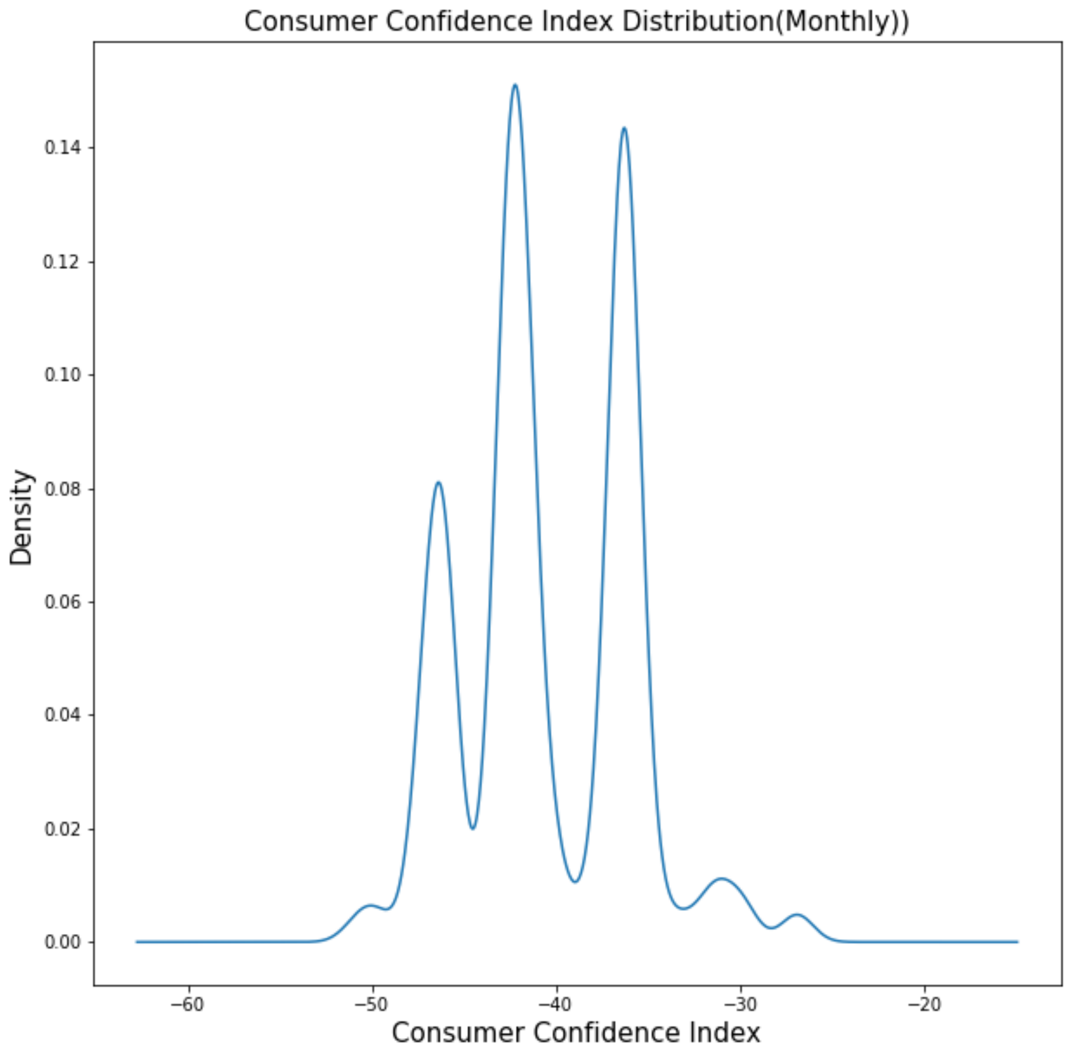

Number of Contacts Before the Campaign
Frequency
0
500
1000
1500
2000
2500
3000
3500
0
1
2
3
4
5
6
Previous Number of Contact

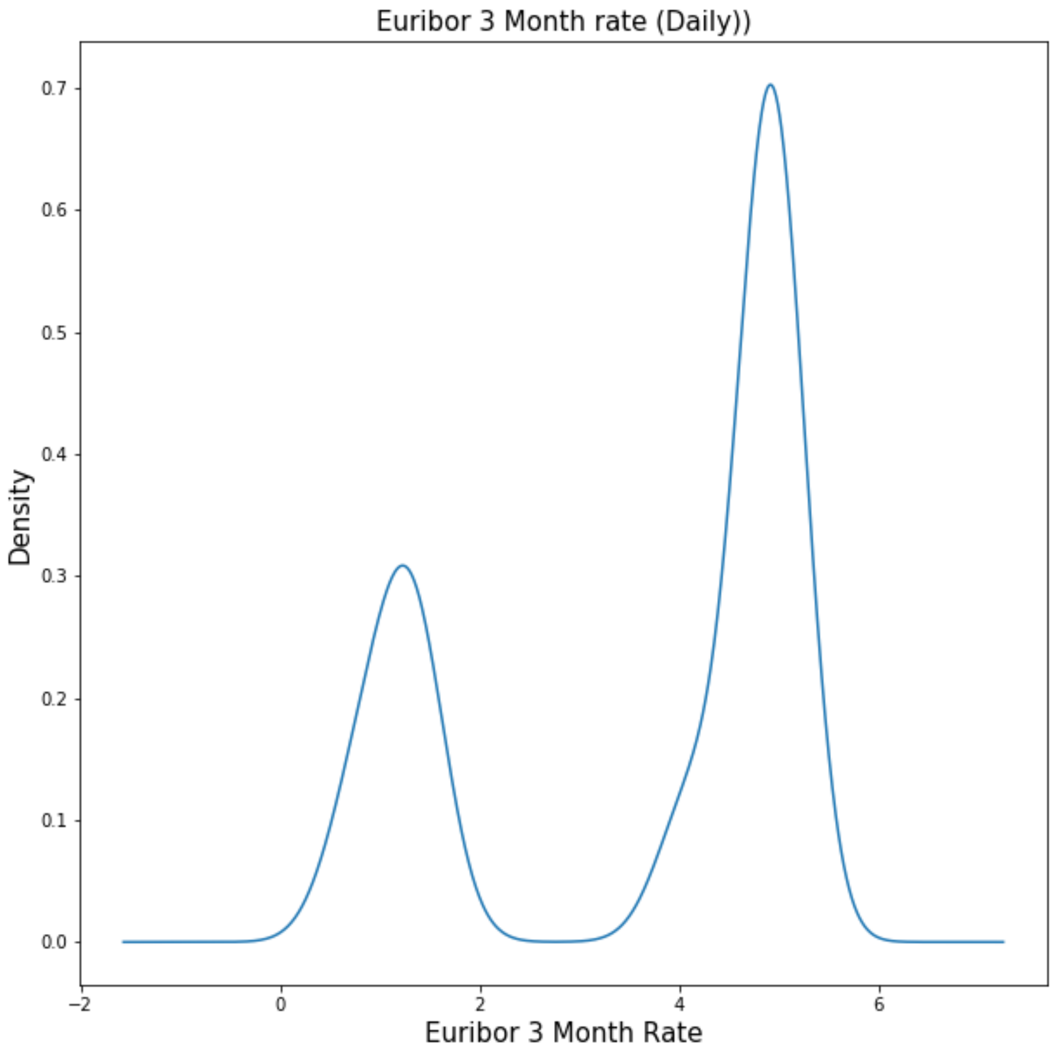

Outcome of Previous Marketing Campaign
Previous Campaign
nonexistent
85.52
success
3.44
failure
11.04

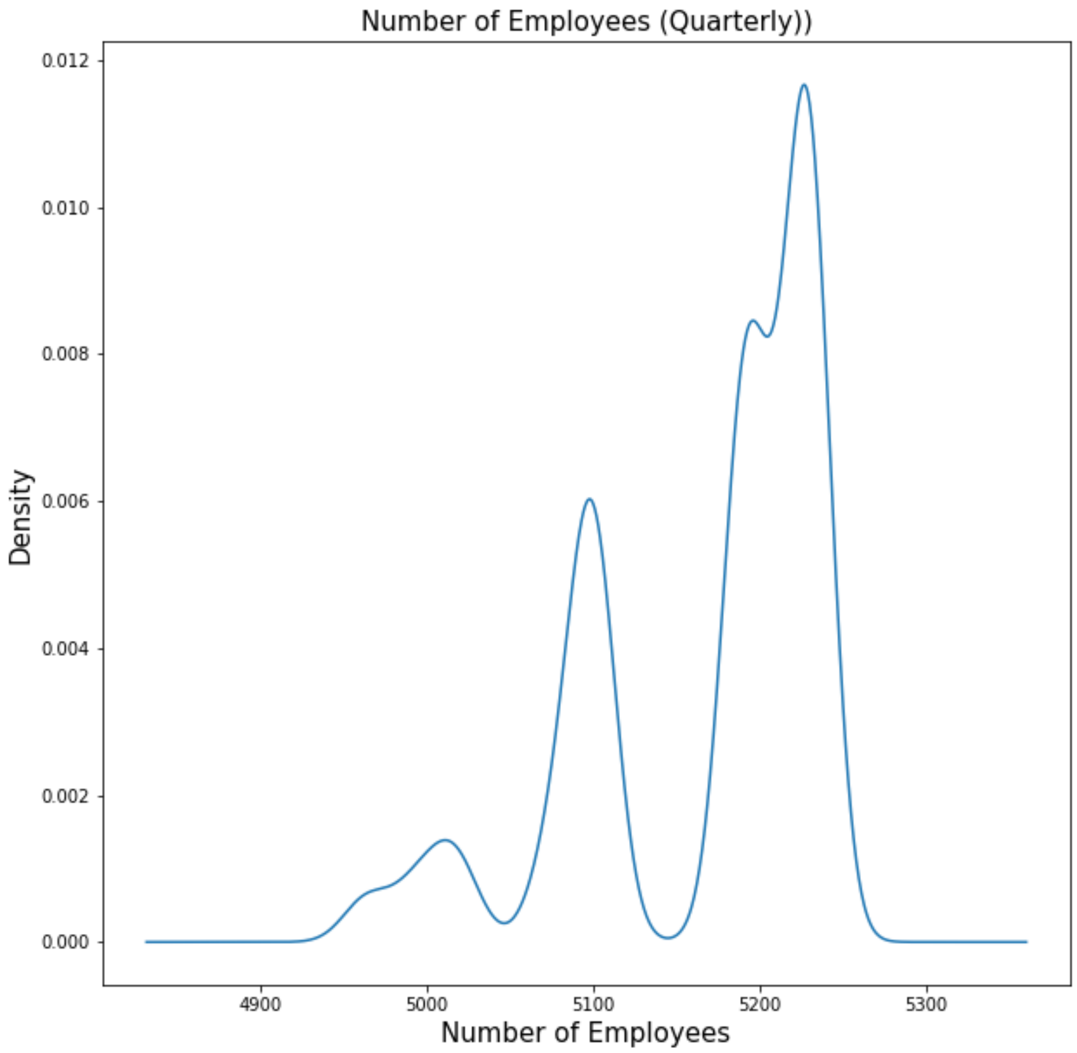

Employment Variation Rate Distribution(Quarterly))
Density
Employment Variation Rate
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
−6
−4
−2
0
2
4

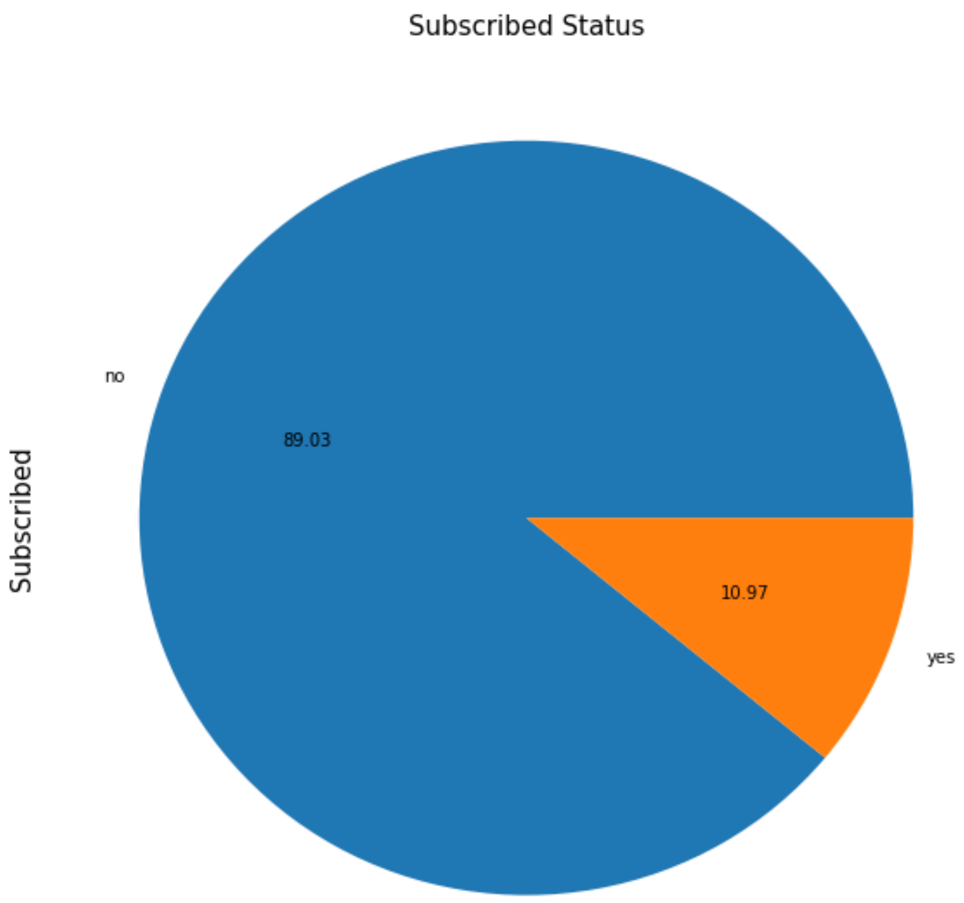


Consumer Price Index Distribution(Monthly))
Density
Consumer Price Index
0.0
0.2
0.4
0.6
0.8
1.0
1.2
91
92
93
94
95
96

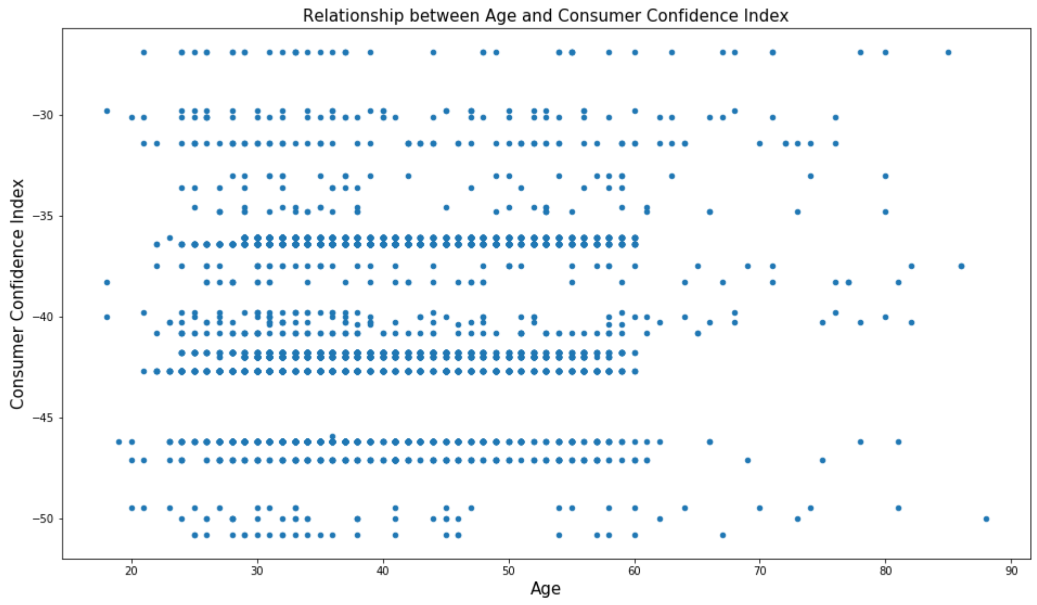

Consumer Confidence Index Distribution(Monthly))
Density
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
−60
−50
−40
−30
−20
Consumer Confidence Index


Euribor 3 Month rate (Daily))
Density
Euribor 3 Month Rate
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
−2
0
2
4
6

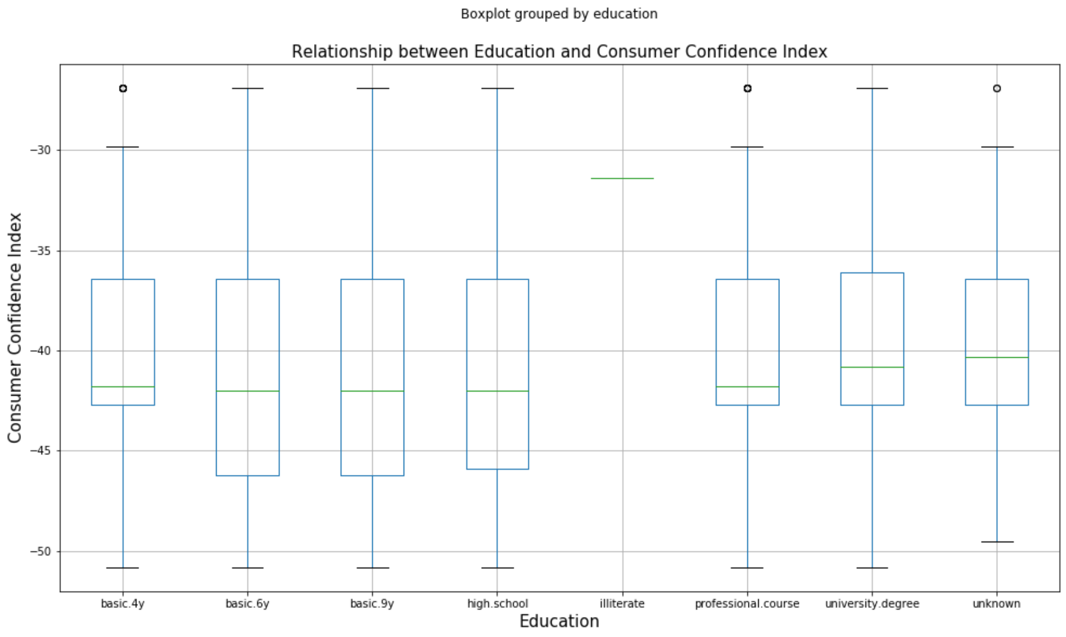

Number of Employees (Quarterly))
Density
0.012
0.010
0.008
0.006
0.004
0.002
0.000
4900
5000
5100
5200
5300
Number of Employees

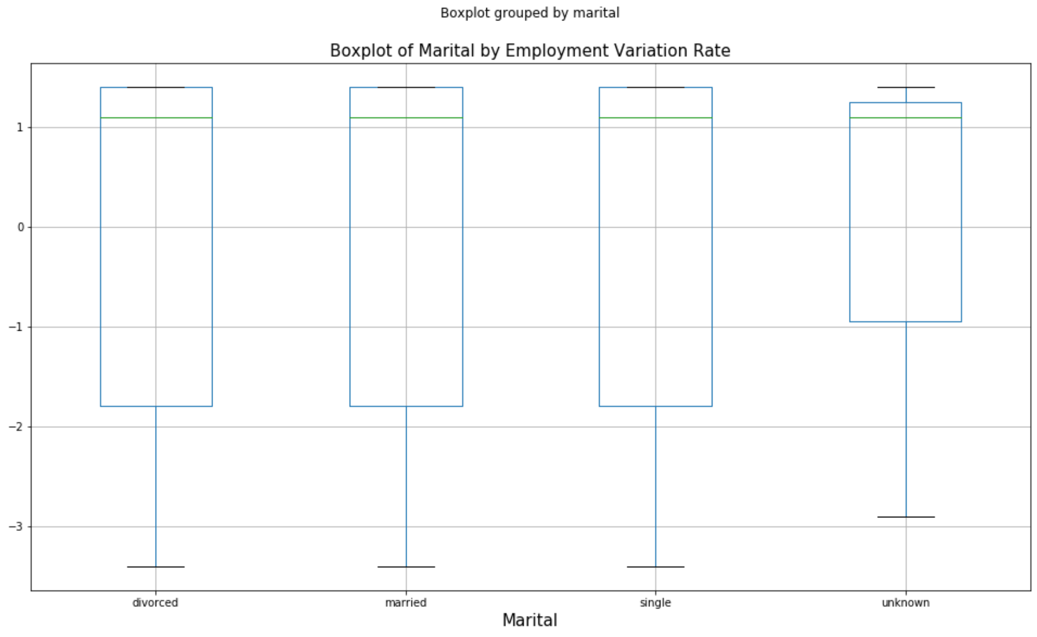

Subscribed Status
Subscribed
no
89.03
10.97
yes

Task 2.2

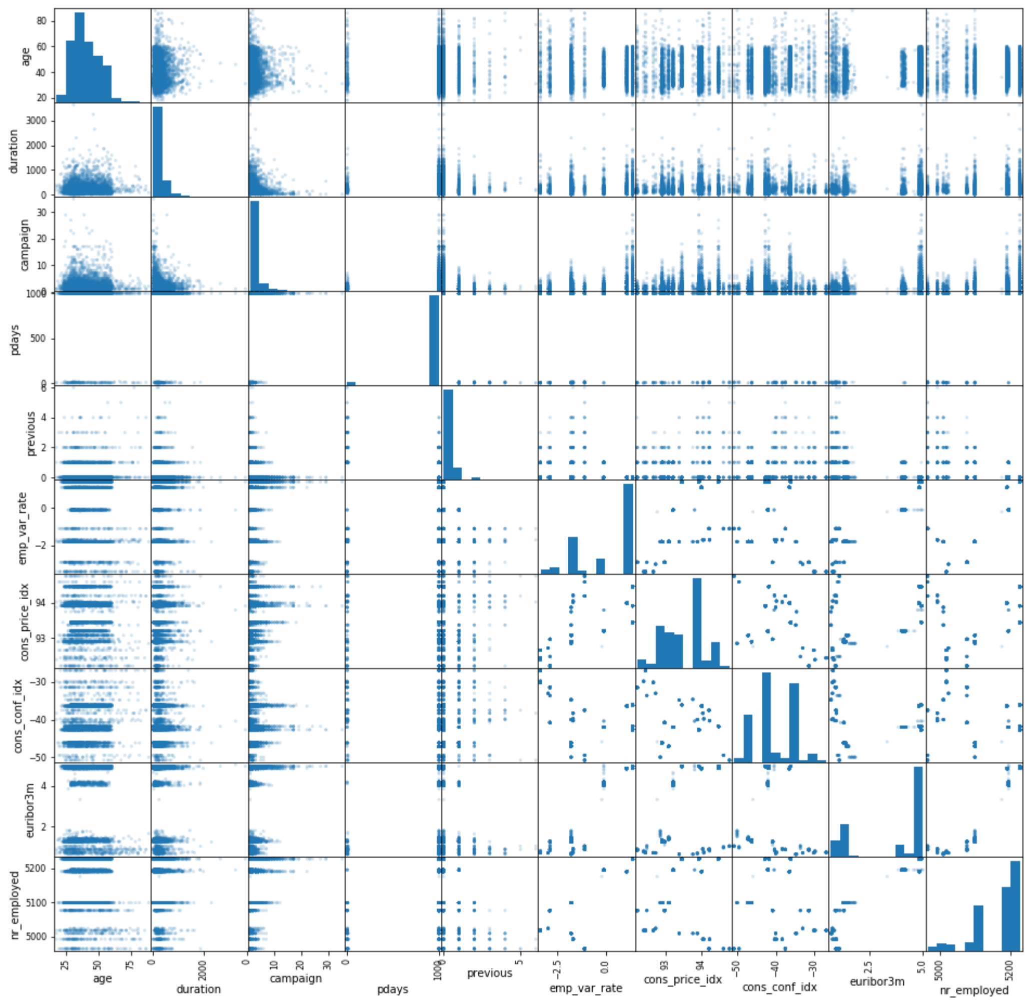


To explore the relationship, matplotlib packages is used (Hunter, 2007). The scatter above shows the relationship between consumer confidence index and ages. It shows that between age 30 and 60 age group, there is highest confidence index.


Boxplot grouped by education
Relationship between Education and Consumer Confidence Index
Consumer Confidence Index
−30
−35
−40
−45
−50
basic.4y
basic.6y
basic.9y
high.school
illiterate
professional.course
university.degree
unknown
Education


Boxplot grouped by marital
Boxplot of Marital by Employment Variation Rate
1
0
−1
−2
−3
divorced
married
single
unknown
Marital

Task 2.3


age
duration
campaign
pdays
previous
emp_var_rate
cons_price_idx
cons_conf_idx
euribor3m
nr_employed

References

Boschetti, A., & Massaron, L. (2015). *Python data science essentials: Become an efficient data science practitioner by thoroughly understanding the key concepts of Python*. Birmingham, England: Packt Publishing.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in science & engineering*, *9*(3), 90-95.

McKinney, W. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference* (Vol. 445, pp. 51-56).

Python Software Foundation. Python Language Reference, version 2.7. Available at http://www.python.org