# Comparison of SVC, Logistic Regression, Random Forest and ANN Result on Portuguese Bank Direct Marketing Campaign Report

By Ghazali Akmal Rabbani

Student of CDS 4 Jan Datamites

April 1st - 20th, 2021.

# 1. INTRODUCTION

## a. Background

There are two main approaches for enterprises to promote products and/or services: through mass campaigns, targeting general indiscriminate public or directed marketing, targeting a specific set of contacts (Ling and Li 1998). Nowadays, in a global competitive world, positive responses to mass campaigns are typically very low, less than 1%, according to the same study. Alternatively, directed marketing focus on targets that assumable will be keener to that specific product/service, making this kind of campaigns more attractive due to its efficiency (Ou et al. 2003).

## b. Problem

Nevertheless, directed marketing has some drawbacks, for instance it may trigger a negative attitude towards banks due to the intrusion of privacy (Page and Luding 2003). It should be stressed that due to internal competition and current financial crisis, there are huge pressures for European banks to increase a financial asset. To solve this issue, one adopted strategy is offer attractive long-term deposit applications with good interest rates, in particular by using directed marketing campaigns. Also, the same drivers are pressing for a reduction in costs and time. Thus, there is a need for an improvement in efficiency: lesser contacts should be done, but an approximately number of successes (clients subscribing the deposit) should be kept.

## c. Interest

The business goal is to find a model that can explain success of a contact, i.e. if the client subscribes the deposit. Such model can increase campaign efficiency by identifying the main characteristics that affect success, helping in a better management of the available resources (e.g. human effort, phone calls, time) and selection of a high quality and affordable set of potential buying customers.

# 2. DATA ACQUISITION AND EXPLANATION

Real-world data were collected from a Portuguese marketing campaign related with bank deposit subscription. We collected data from a Portuguese bank that used its own contact-center to do directed marketing campaigns (Moro et al. 2011). The dataset contains the following columns:

*Bank client data*

- Age : age of person (numeric)
- Job : type of job (categorical : “admin”, “unknown”, “unemployed”, “management”, “housemaid”, “entrepreneur”, “student”, “blue-collar”, “self-employed”, “retired”, “technician”, “services”)
- Marital : marital status (categorical : “married”, “divorced”, “single”)

- Education : education of person (categorical : “unknown”, “secondary”, “primary”, “tertiary”)
- Default: has credit in default? (binary: “yes”, “no”)
- Balance : average yearly balance, in euros (numeric)
- Housing: has housing loan? (binary: “yes”, ”no”)
- Loan: has personal loan? (binary: “yes”, “no”)

*Related with the last contact of the current campaign*

- Contact: contact communication type (categorical : “unknown”, “telephone”, “cellular”)
- Day: last contact day of the month (numeric)
- Month: last contact month of year (categorical : “Jan”, ”Feb”, ”Mar”,…,”Nov”, ”Dec”)
- Duration: last contact duration, in seconds (numeric)

*Other attributes*

- Campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
- Pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted)
- Previous: number of contacts performed before this campaign and for this client (numeric)
- Poutcome: outcome of the previous marketing campaign (categorical: “unknown”, “other”, “failure”, “success”)

*Output Variables*

- Y: has the client subscribed a term deposit? (binary: “yes”, “no”)

# 3. METHODOLOGY

## a. Exploratory Data Analysis

### i. Summary Statistical Analysis

#### 1. Information Analysis

Check for Missing Value

```
df.notnull().all().all()
```

```
True
```

*There are no missing value in this dataset*

Summary information of each column in dataframe

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        45211 non-null  int64
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64
 12  campaign   45211 non-null  int64
 13  pdays      45211 non-null  int64
 14  previous   45211 non-null  int64
 15  poutcome   45211 non-null  object
 16  y          45211 non-null  object
dtypes: int64(7), object(10)
```

Based on the summary information in the above, there are **7 columns with integer type (int64)** which denotes **Numerical Column** and **10 columns with object type** which denotes **Categorical Column**. And also there are **45211 data.**

## 2. Statistical Numerical Analysis

| | age | balance | day | duration | campaign | pdays | previous |
|---|---|---|---|---|---|---|---|
| count | 45211.000000 | 45211.000000 | 45211.000000 | 45211.000000 | 45211.000000 | 45211.000000 | 45211.000000 |
| mean | 40.936210 | 1362.272058 | 15.806419 | 258.163080 | 2.763841 | 40.197828 | 0.580323 |
| std | 10.618762 | 3044.765829 | 8.322476 | 257.527812 | 3.098021 | 100.128746 | 2.303441 |
| min | 18.000000 | -8019.000000 | 1.000000 | 0.000000 | 1.000000 | -1.000000 | 0.000000 |
| 25% | 33.000000 | 72.000000 | 8.000000 | 103.000000 | 1.000000 | -1.000000 | 0.000000 |
| 50% | 39.000000 | 448.000000 | 16.000000 | 180.000000 | 2.000000 | -1.000000 | 0.000000 |
| 75% | 48.000000 | 1428.000000 | 21.000000 | 319.000000 | 3.000000 | -1.000000 | 0.000000 |
| max | 95.000000 | 102127.000000 | 31.000000 | 4918.000000 | 63.000000 | 871.000000 | 275.000000 |

- **Count** is a total data of each column
- **Mean** is an average of all the numbers
- **std** or Standard Deviation is measurement to tell how a set of values spread out from their mean. A low Standard Deviation whows that the values are close to the mean and a high Standard Deviation shows a high diversion from the mean

### 3. Statistical Categorical Analysis

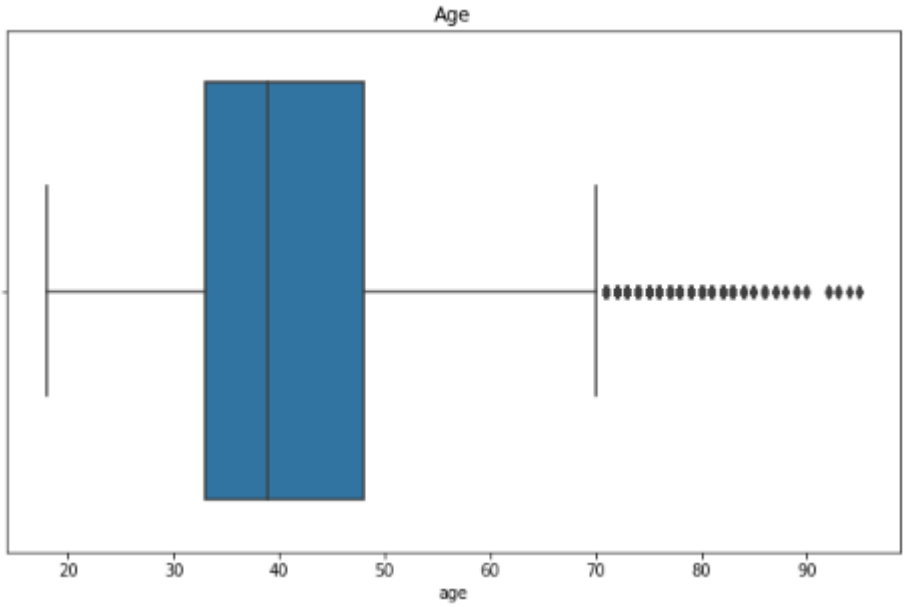
| | job | marital | education | default | housing | loan | contact | month | poutcome | y |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 45211 | 45211 | 45211 | 45211 | 45211 | 45211 | 45211 | 45211 | 45211 | 45211 |
| unique | 12 | 3 | 4 | 2 | 2 | 2 | 3 | 12 | 4 | 2 |
| top | blue-collar | married | secondary | no | yes | no | cellular | may | unknown | no |
| freq | 9732 | 27214 | 23202 | 44396 | 25130 | 37967 | 29285 | 13766 | 36959 | 39922 |

- **count** is total data of each column
- **unique** is total of unique data in each column. For example, in Job column there are 12 difference kind of data
- **top** is the most frequency or mode data in each column
- **freq** is total of top data or mode data in each column. For example, in Job column, the mode data is *blue-collar* and total of data with value of *blue-collar* is 9732 data in Job column

## ii. Univariate Analysis

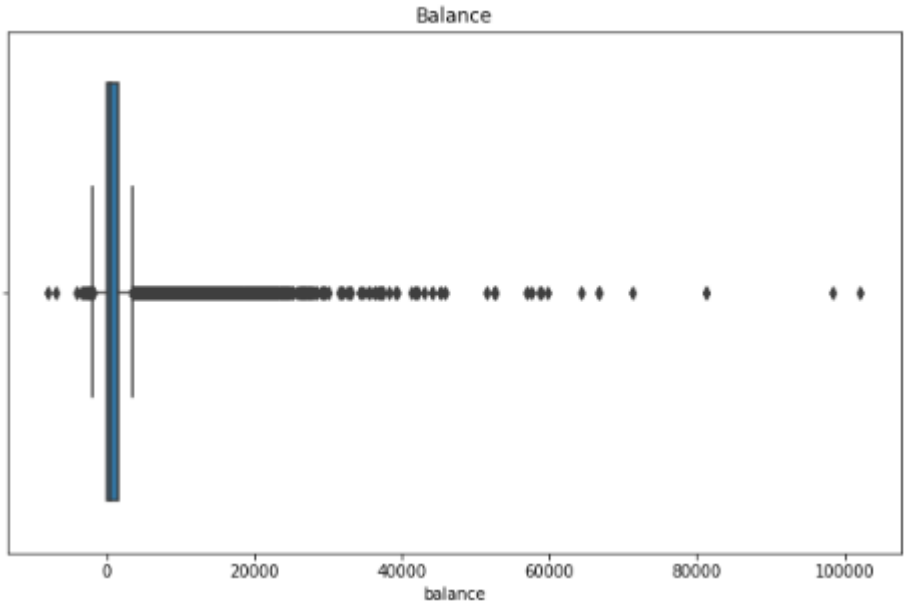
### 1. Boxplot

#### a. Age Boxplot



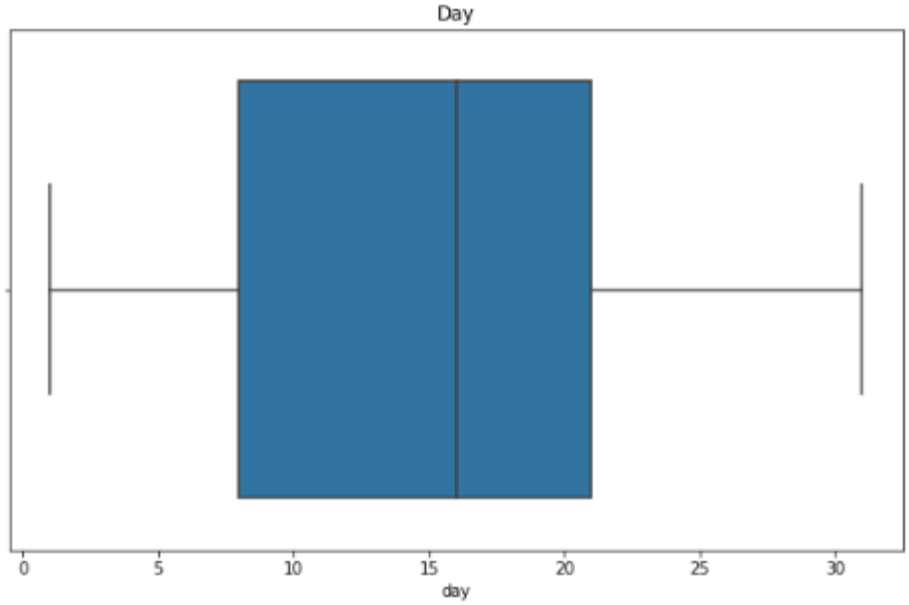

| Measurement | | Value |
|---|---|---|
| Median | : | 39.0 |
| 1st Quartile | : | 33.0 |
| 3rd Quartile | : | 48.0 |
| Interquartile | : | 15.0 |
| Outlier | : | Yes |

## b. Balance Boxplot




| Measurement | | Value |
|---|---|---|
| Median | : | 448.0 |
| 1st Quartile | : | 72.0 |
| 3rd Quartile | : | 1428.0 |
| Interquartile | : | 1356.0 |
| Outlier | : | Yes |

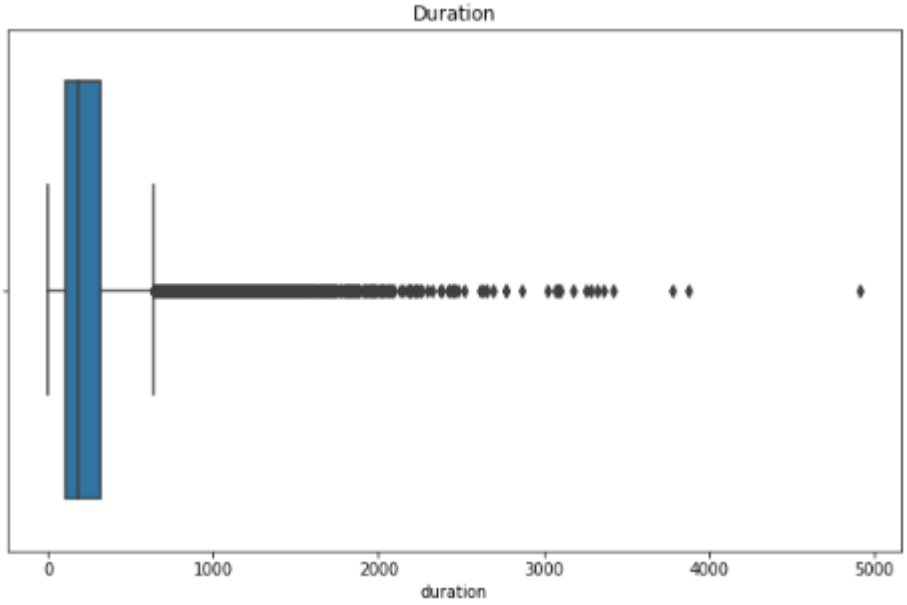
## c. Day Boxplot




| Measurement | | Value |
|---|---|---|
| Median | : | 16.0 |

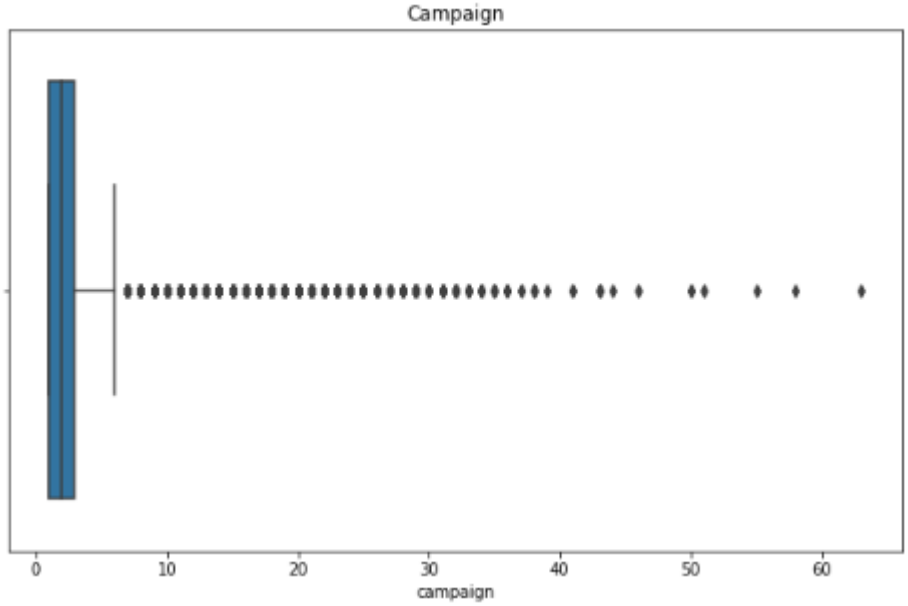
| | | |
|---|---|---|
| 1st Quartile | : | 8.0 |
| 3rd Quartile | : | 21.0 |
| Interquartile | : | 13.0 |
| Outlier | : | Yes |

## d. Duration Boxplot



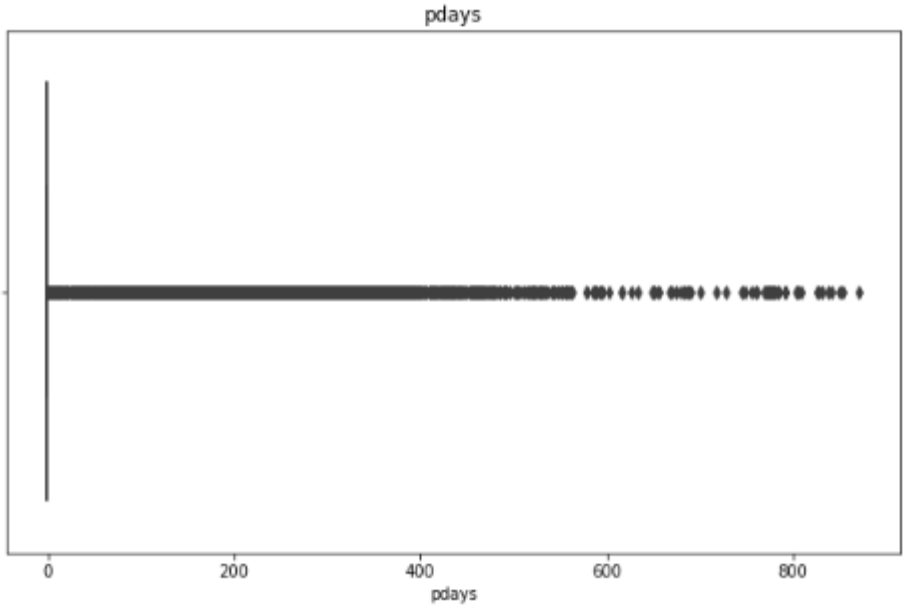

| Measurement | | Value |
|---|---|---|
| Median | : | 180.0 |
| 1st Quartile | : | 103.0 |
| 3rd Quartile | : | 319.0 |
| Interquartile | : | 216.0 |
| Outlier | : | Yes |

## e. Campaign Boxplot




| Measurement | | Value |
|---|---|---|
| Median | : | 2.0 |
| 1st Quartile | : | 1.0 |
| 3rd Quartile | : | 3.0 |
| Interquartile | : | 2.0 |
| Outlier | : | Yes |

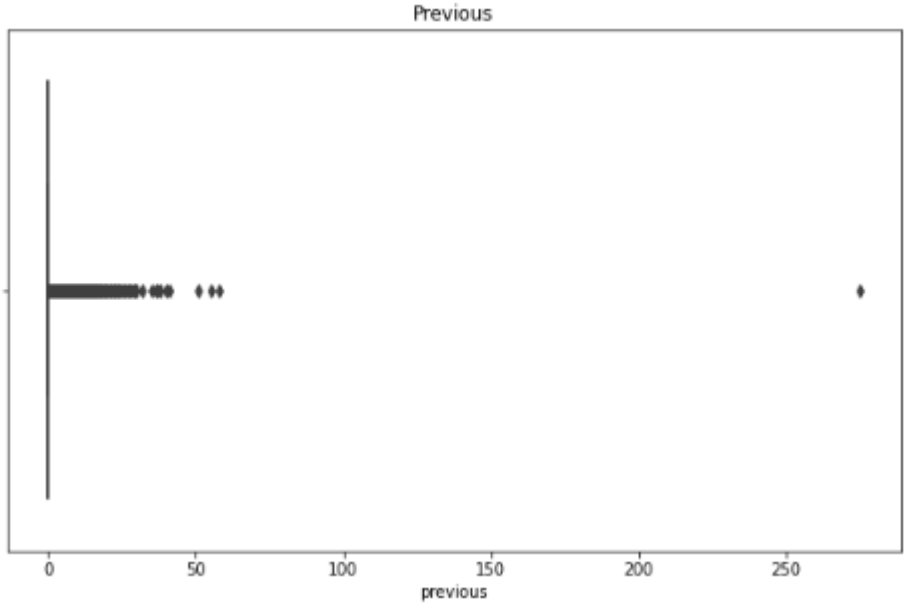
## f. Pdays Boxplot




| Measurement | | Value |
|---|---|---|
| Median | : | -1.0 |

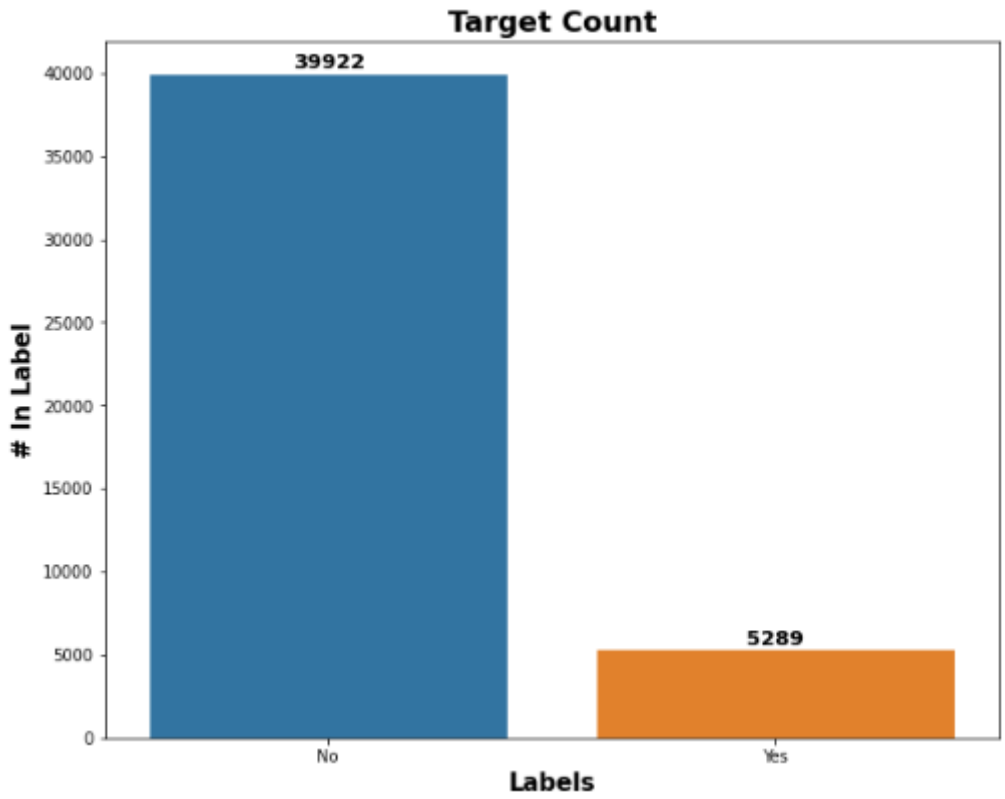
| | | |
|---|---|---|
| 1st Quartile | : | -1.0 |
| 3rd Quartile | : | -1.0 |
| Interquartile | : | 0.0 |
| Outlier | : | Yes |

## g. Previous Boxplot




| Measurement | | Value |
|---|---|---|
| Median | : | 0.0 |
| 1st Quartile | : | 0.0 |
| 3rd Quartile | : | 0.0 |
| Interquartile | : | 0.0 |
| Outlier | : | Yes |

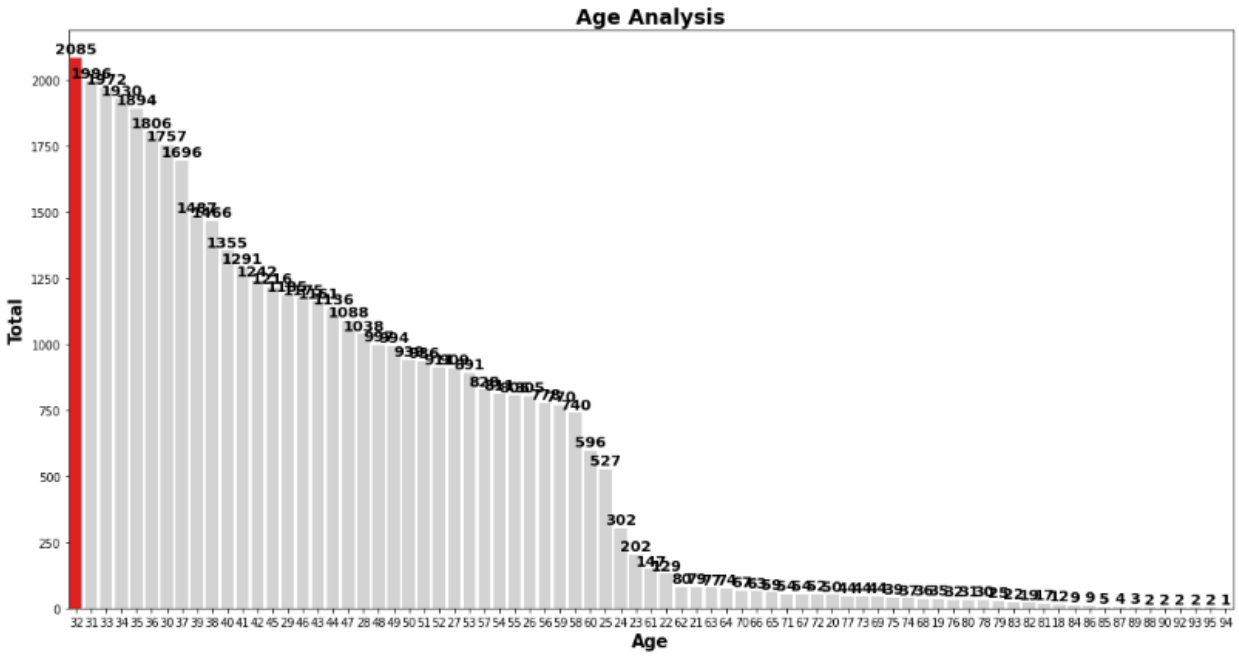
## 2. Target Analysis



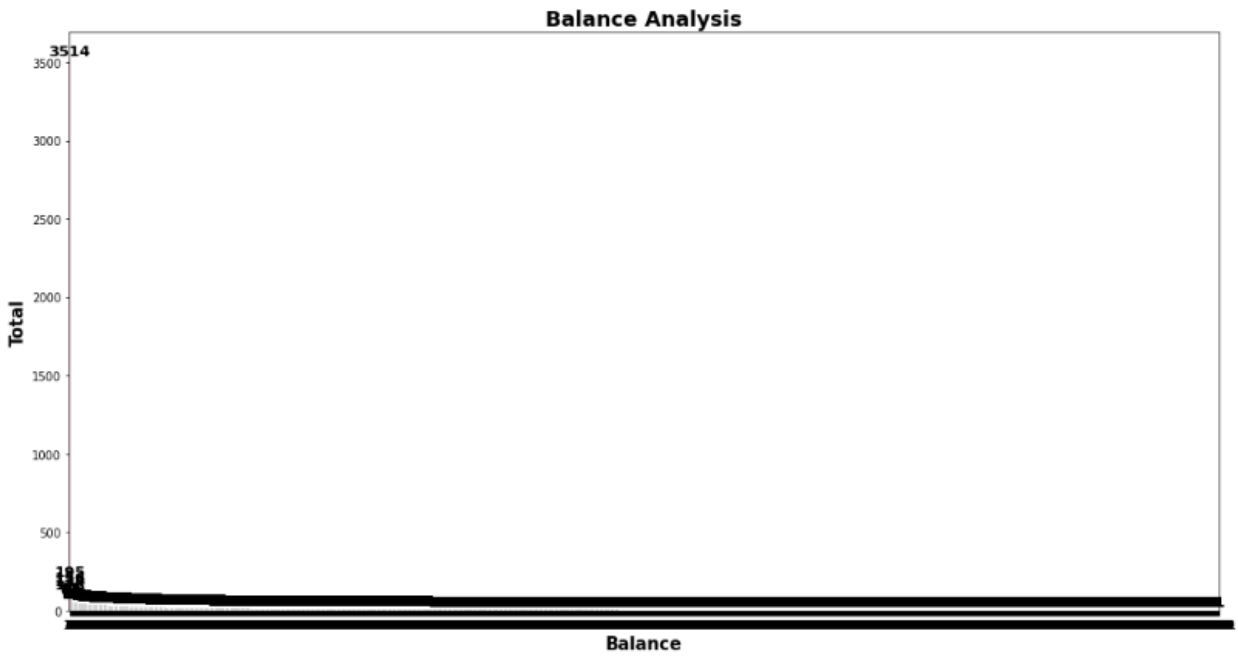

From visualization in above, there are a lot of “No” data with ratio comparison to “Yes” data is 15:2 of No:Yes data.

## 3. Visualization Numerical Analysis

### a. Age Analysis



## b. Balance Analysis

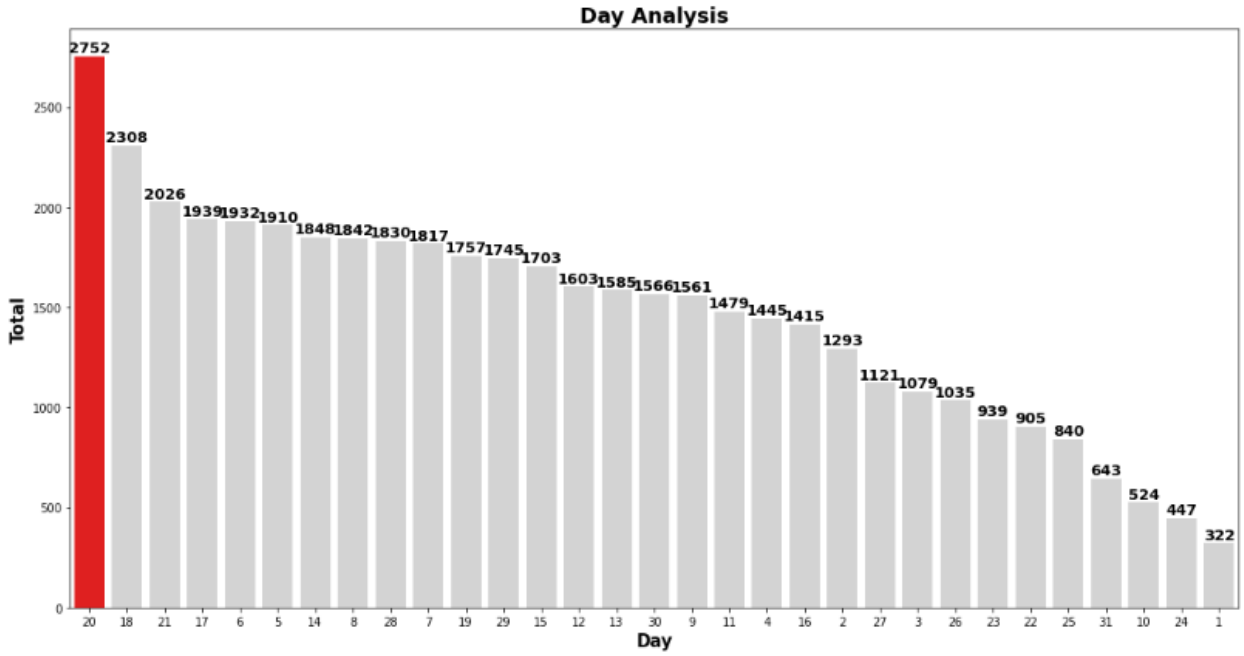



| | unique_values | counts |
|---|---|---|
| 0 | 0 | 3514 |
| 1 | 1 | 195 |
| 2 | 2 | 156 |
| 3 | 4 | 139 |
| 4 | 3 | 134 |
| ... | ... | ... |
| 7163 | 4305 | 1 |
| 7164 | 6352 | 1 |
| 7165 | 18881 | 1 |
| 7166 | 14889 | 1 |
| 7167 | 7218 | 1 |

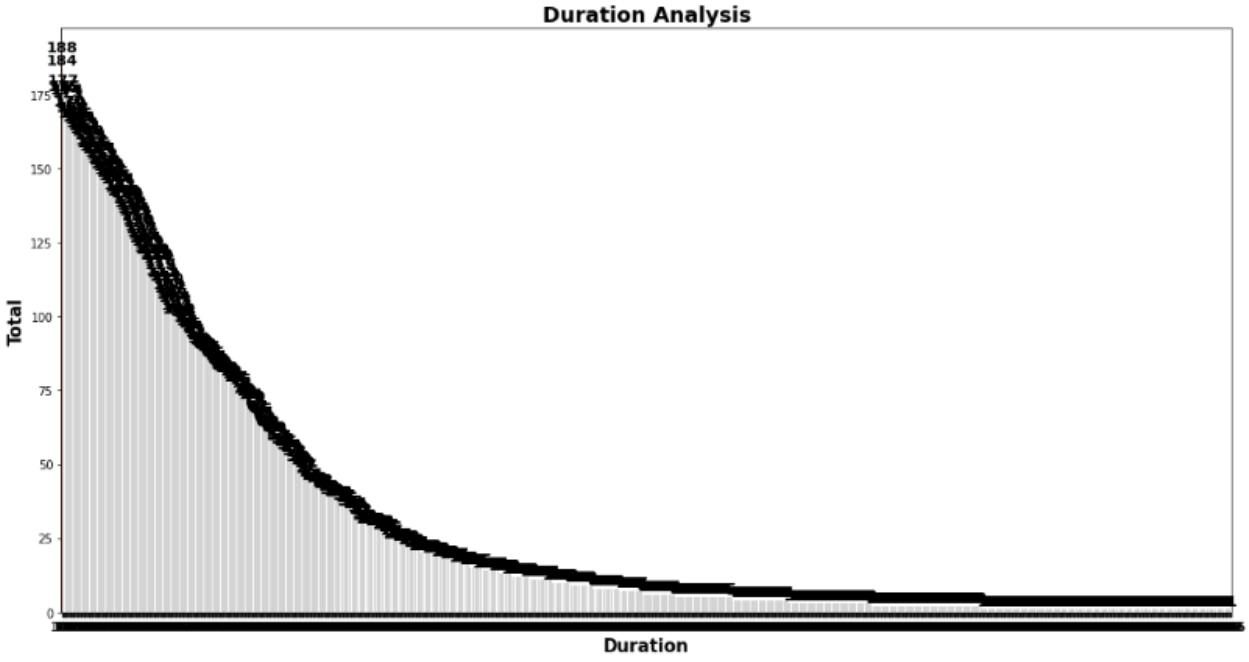
7168 rows × 2 columns

Table of Visualization of Top 5 and Least 5 Total Data Balance

## c. Day Analysis




## d. Duration Analysis



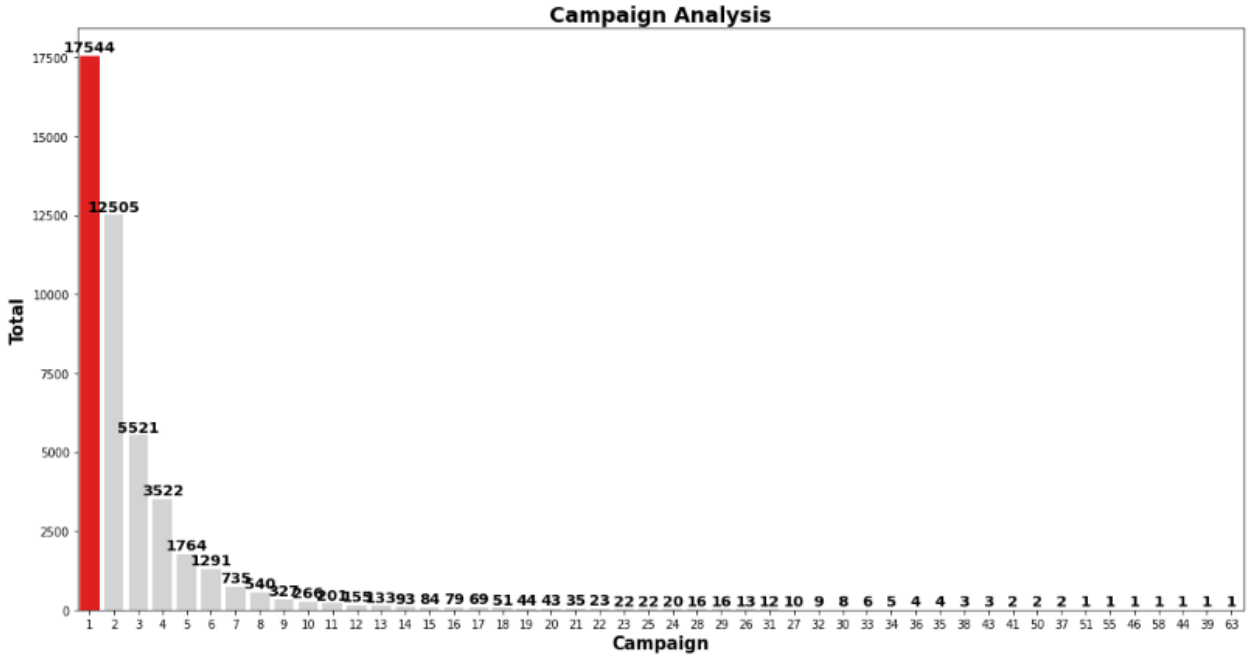

| | unique_values | counts |
|---|---|---|
| 0 | 124 | 188 |
| 1 | 90 | 184 |
| 2 | 89 | 177 |
| 3 | 122 | 175 |
| 4 | 104 | 175 |

| | | |
|---|---|---|
| 1568 | 2150 | 1 |
| 1569 | 1970 | 1 |
| 1570 | 1906 | 1 |
| 1571 | 1842 | 1 |
| 1572 | 2015 | 1 |

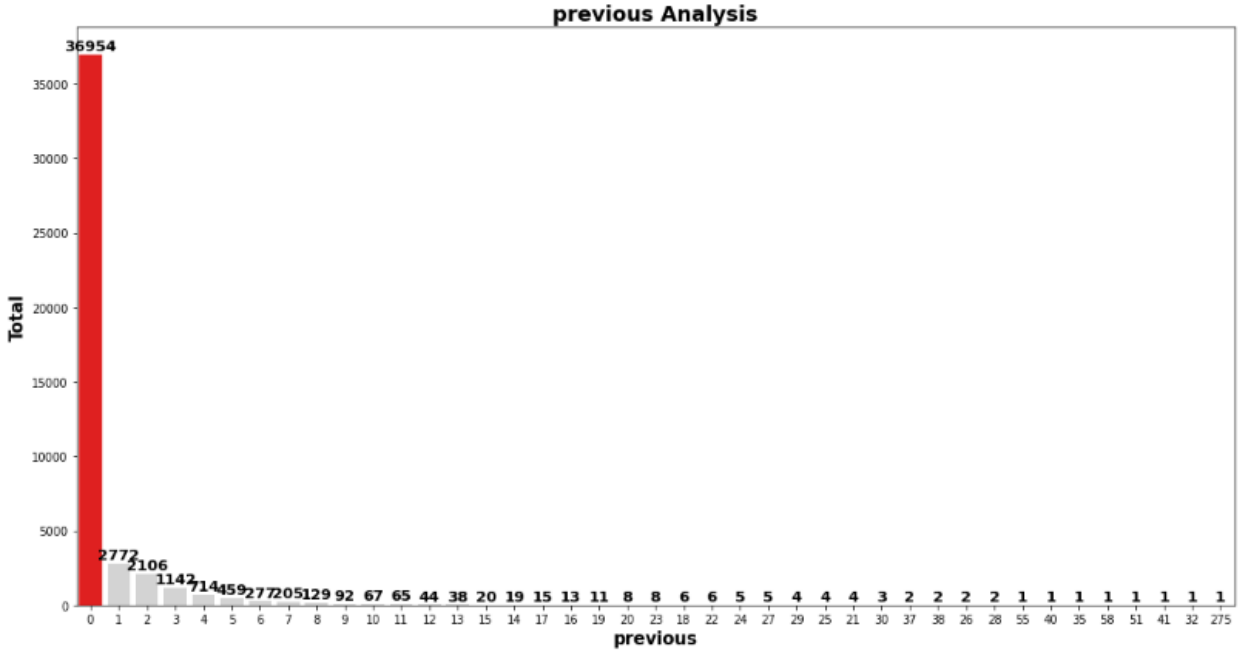
1573 rows × 2 columns

Table of Visualization of Top 5 and Least 5 Total Data Duration

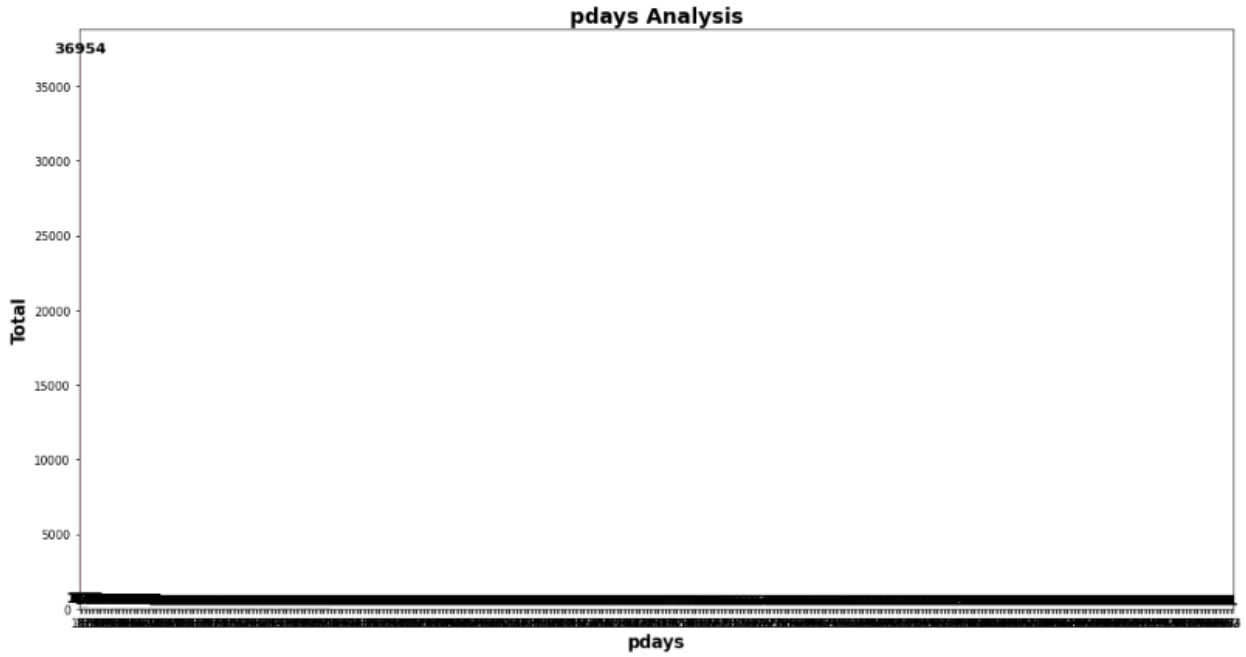
## e. Campaign Analysis




## f. Previous Analysis



## g. Pdays Analysis




| | unique_values | counts |
|---|---|---|
| 0 | -1 | 36954 |
| 1 | 182 | 167 |
| 2 | 92 | 147 |
| 3 | 183 | 126 |
| 4 | 91 | 126 |

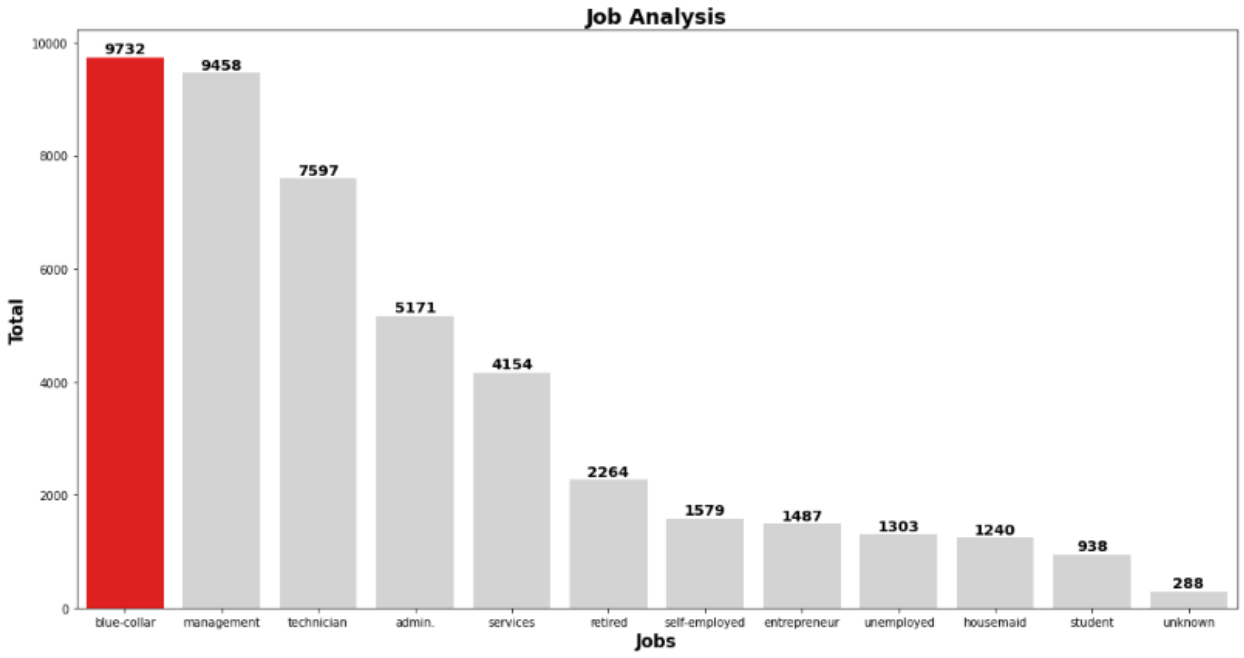
| | | |
|---|---|---|
| 554 | 749 | 1 |
| 555 | 717 | 1 |
| 556 | 589 | 1 |
| 557 | 493 | 1 |
| 558 | 32 | 1 |

559 rows × 2 columns

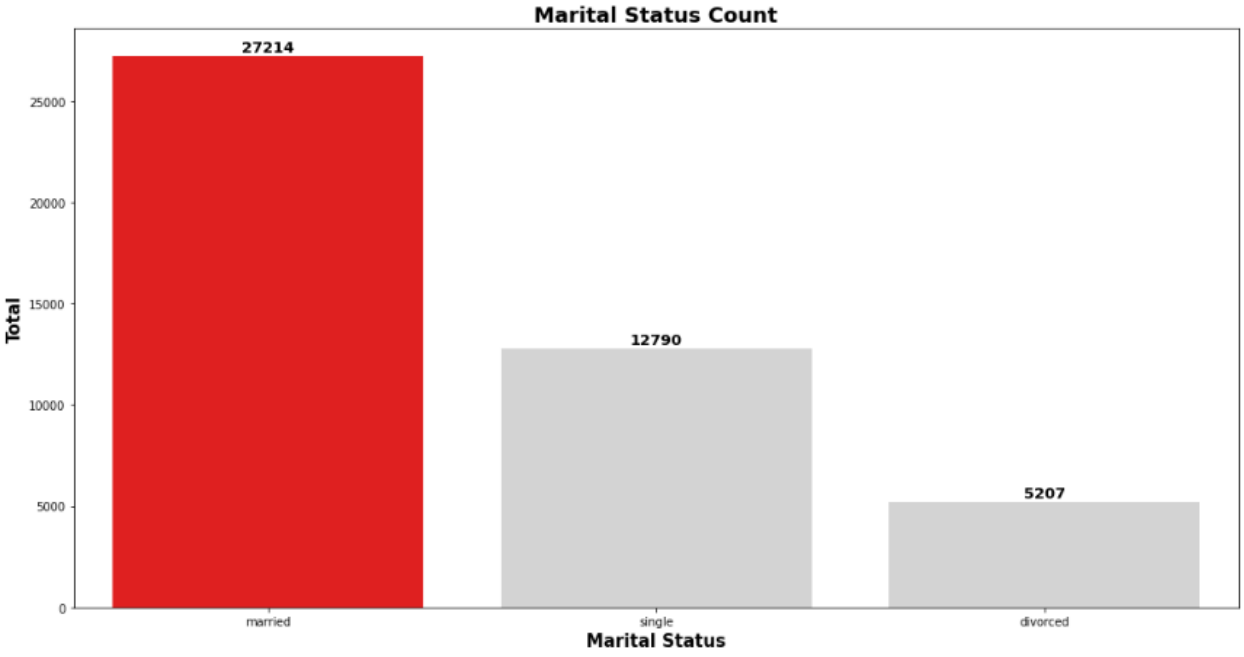
Table of Visualization of Top 5 and Least 5 Total Data pdays

## 4. Visualization Categorical Analysis

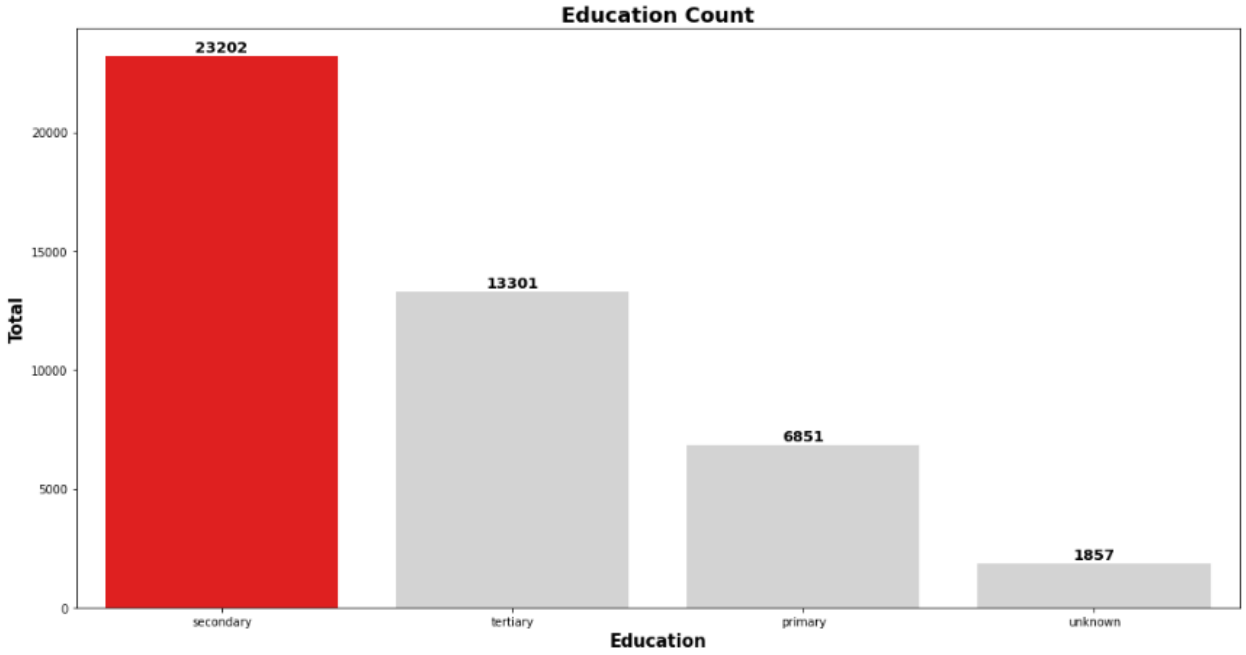
### a. Job Analysis



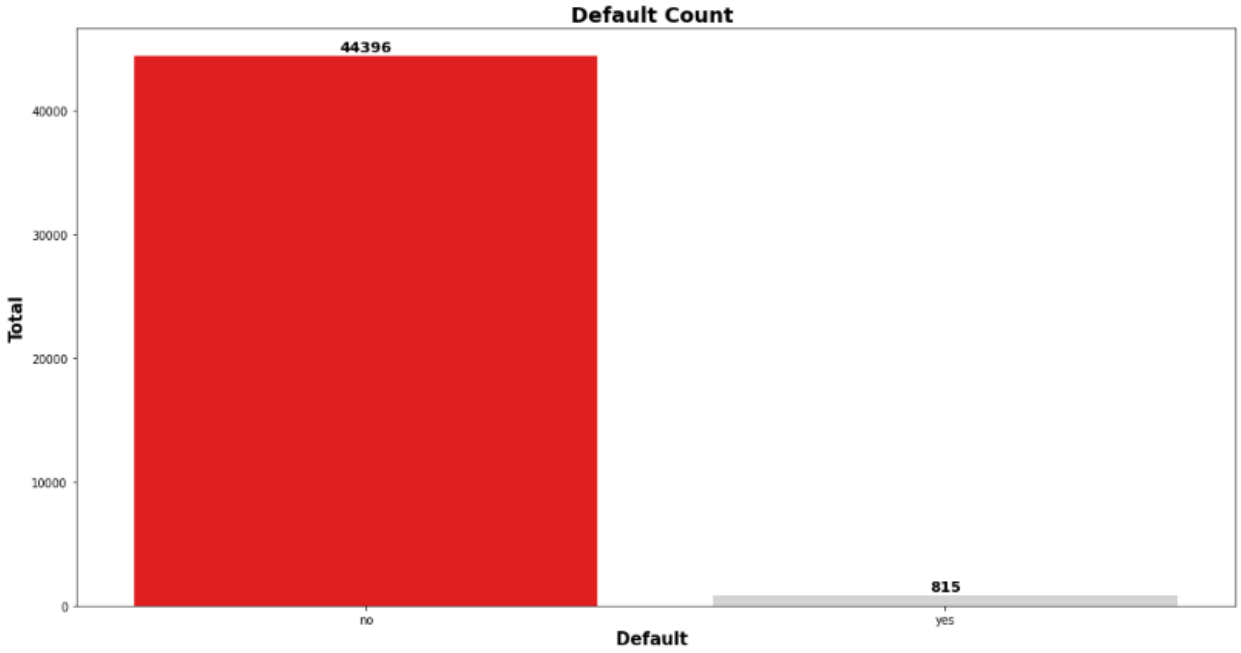

### b. Marital Status Analysis



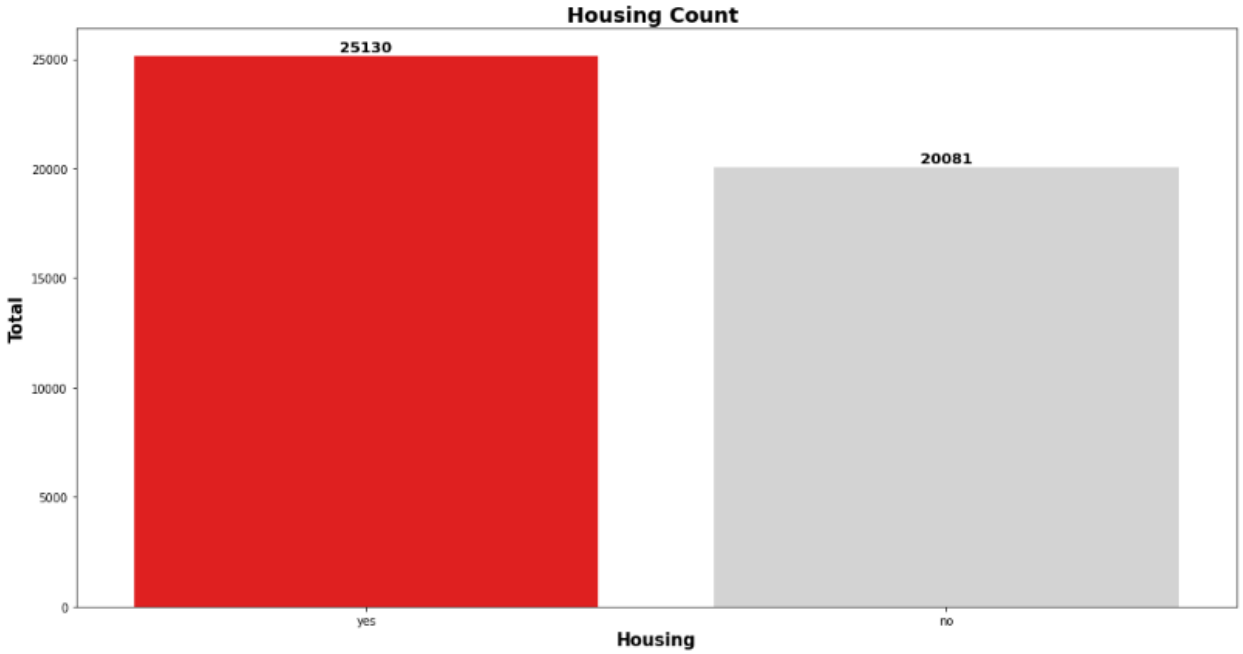
## c. Education Analysis



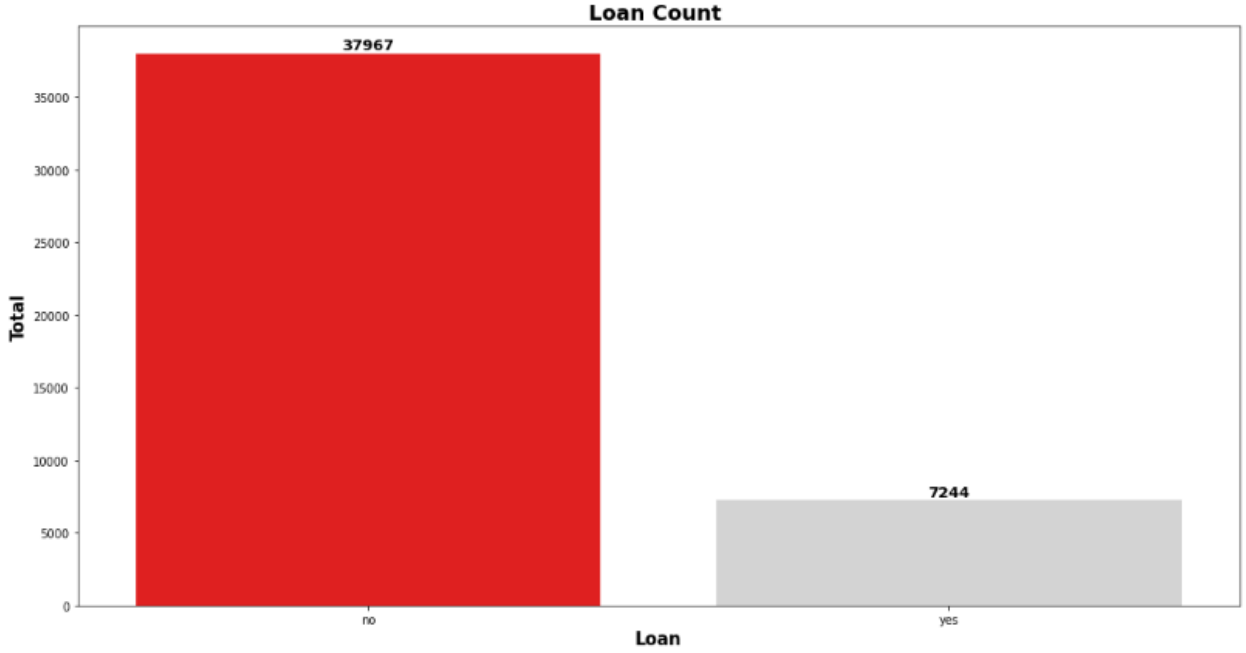

## d. Default Analysis



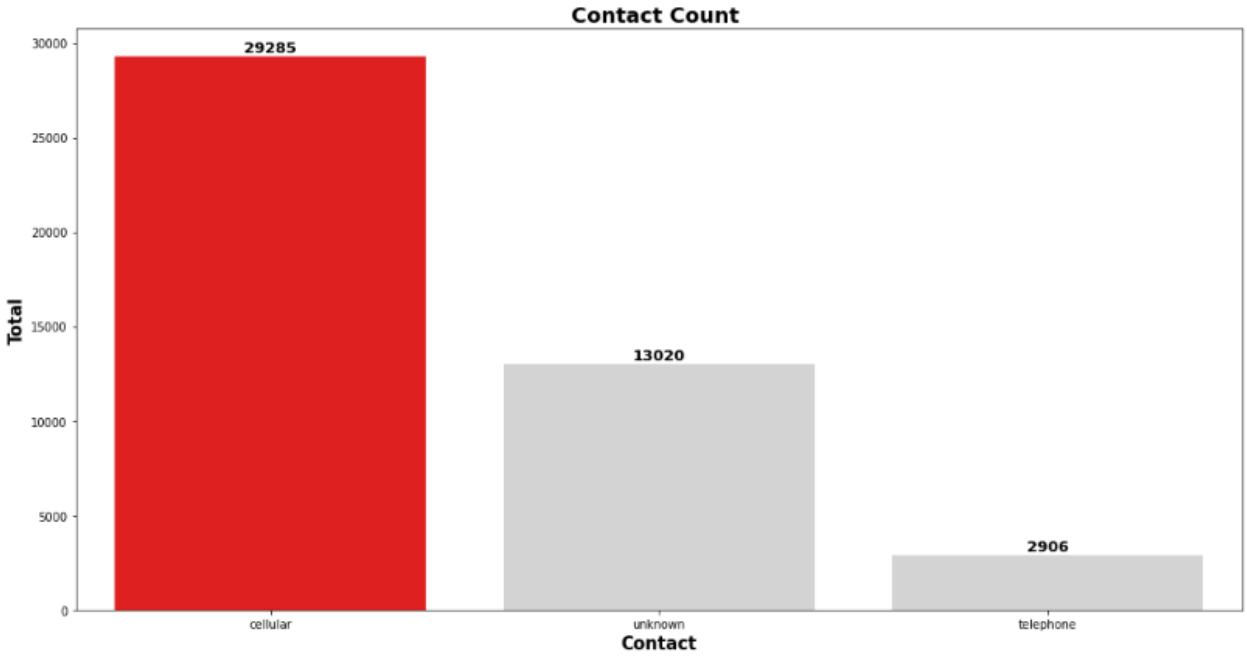
## e. Housing Analysis



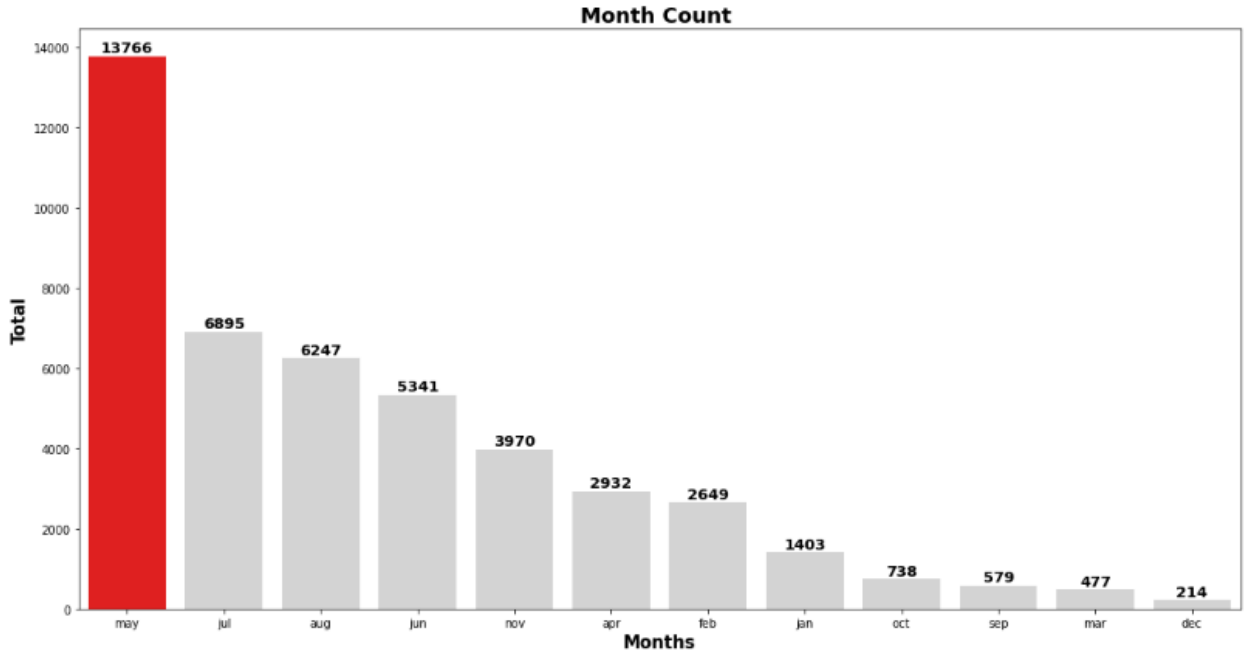

## f. Loan Analysis



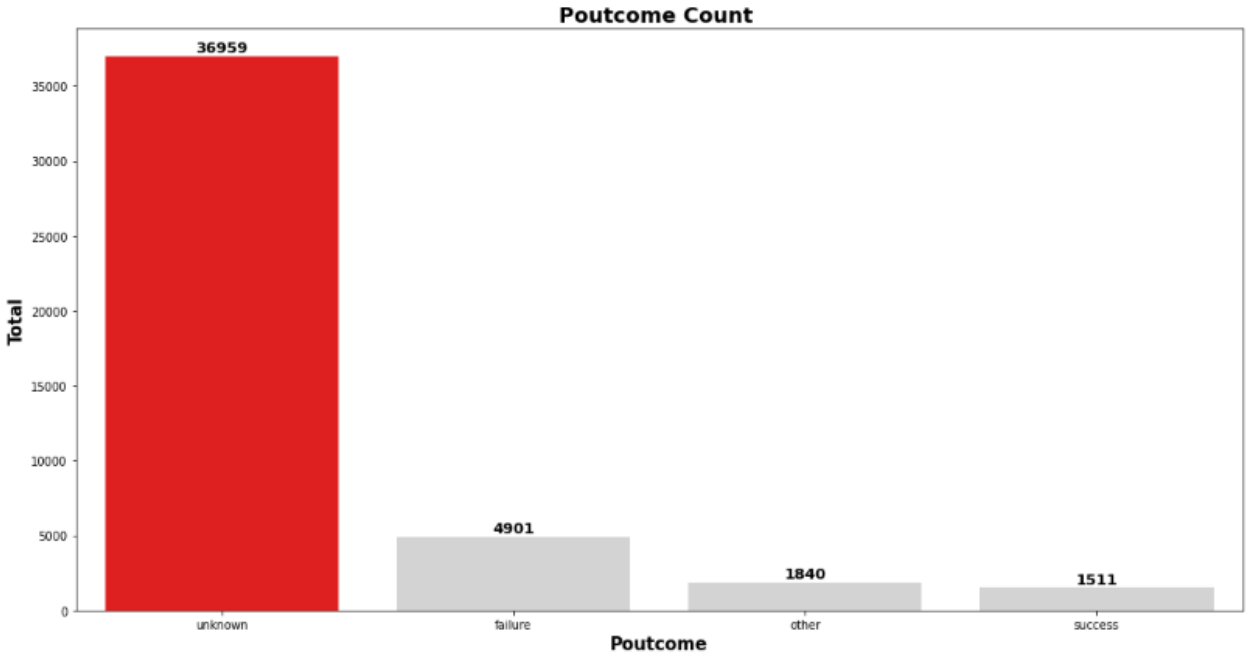
## g. Contact Analysis



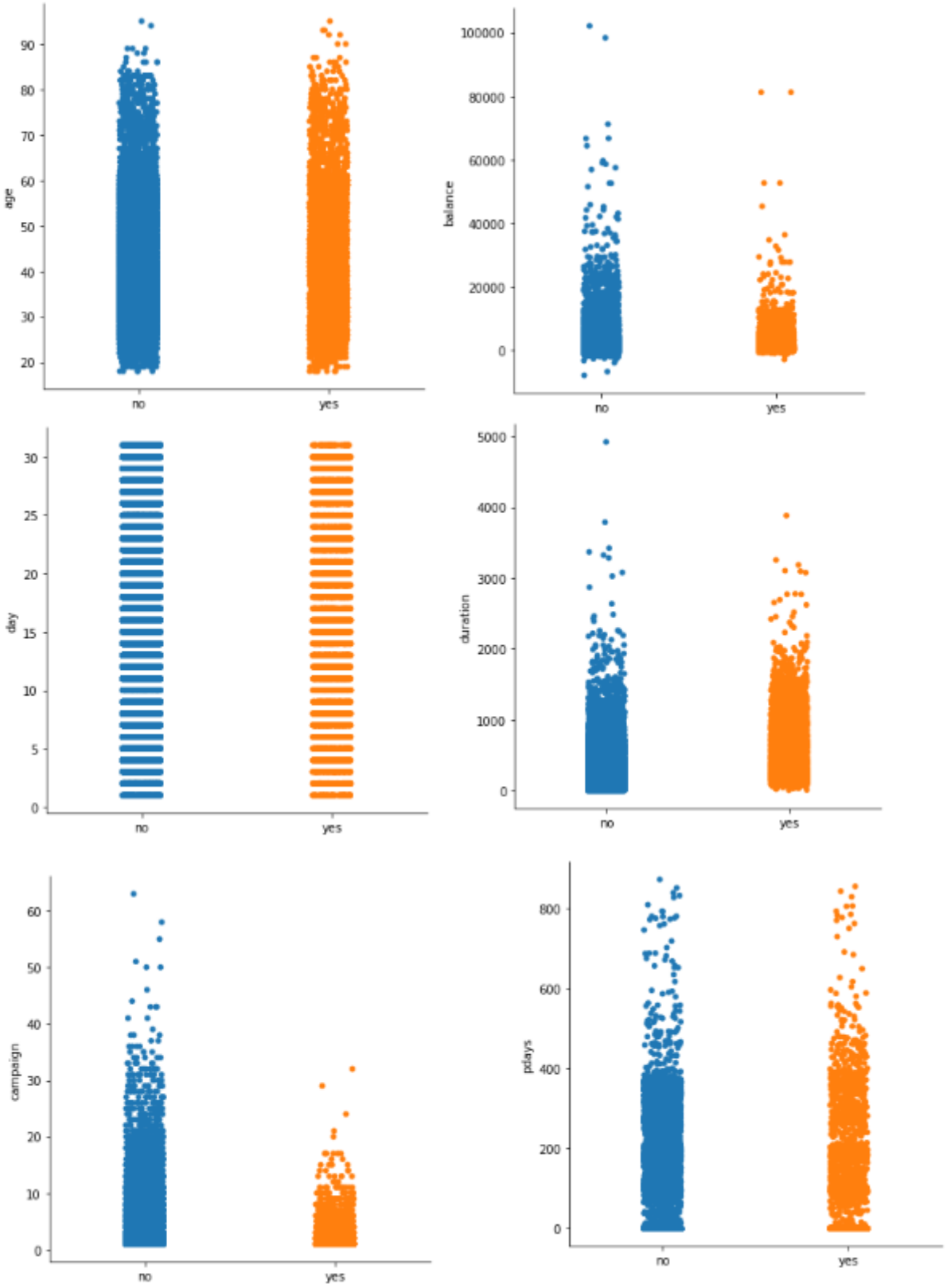

## h. Month Analysis



## i. Poutcome Analysis

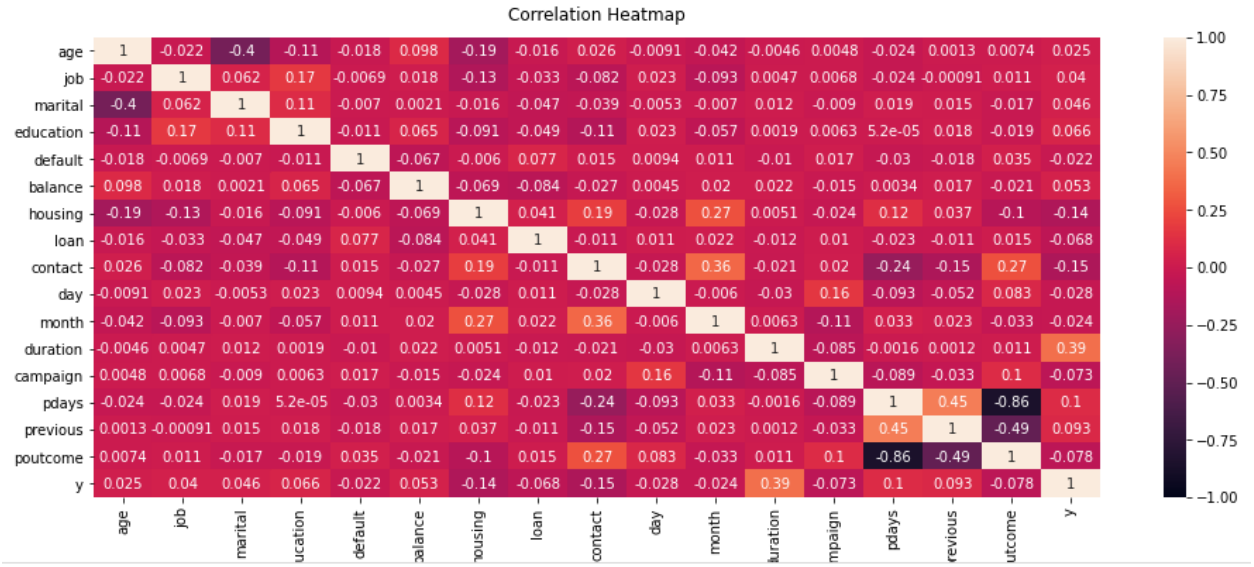


## iii. Bivariate Analysis




*campaign No > campaign Yes. balance No > balance Yes. pdays Yes > pdays No*

### iv. Multivariate Analysis




- *Duration has the strongest Linear Correlation between Feature and Target with 39% positive linear correlation*
- *No features seems to be redundant*

## b. Modelling

- **Label Encoding**

  Label Encoding are used to transform non-numerical labels (as long as they are hash able and comparable) to numerical labels. Label that are being transform to numerical are categorical data column which is job, marital, education, default, housing, loan, contact, month and poutcome.

| | age | job | marital | education | default | balance | housing | loan | contact | day | month | duration | campaign | pdays | previous | poutcome |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 58 | 4 | 1 | 2 | 0 | 2143 | 1 | 0 | 2 | 5 | 8 | 261 | 1 | -1 | 0 | 3 |
| **1** | 44 | 9 | 2 | 1 | 0 | 29 | 1 | 0 | 2 | 5 | 8 | 151 | 1 | -1 | 0 | 3 |
| **2** | 33 | 2 | 1 | 1 | 0 | 2 | 1 | 1 | 2 | 5 | 8 | 76 | 1 | -1 | 0 | 3 |
| **3** | 47 | 1 | 1 | 3 | 0 | 1506 | 1 | 0 | 2 | 5 | 8 | 92 | 1 | -1 | 0 | 3 |
| **4** | 33 | 11 | 2 | 3 | 0 | 1 | 0 | 0 | 2 | 5 | 8 | 198 | 1 | -1 | 0 | 3 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **45206** | 51 | 9 | 1 | 2 | 0 | 825 | 0 | 0 | 0 | 17 | 9 | 977 | 3 | -1 | 0 | 3 |
| **45207** | 71 | 5 | 0 | 0 | 0 | 1729 | 0 | 0 | 0 | 17 | 9 | 456 | 2 | -1 | 0 | 3 |
| **45208** | 72 | 5 | 1 | 1 | 0 | 5715 | 0 | 0 | 0 | 17 | 9 | 1127 | 5 | 184 | 3 | 2 |
| **45209** | 57 | 1 | 1 | 1 | 0 | 668 | 0 | 0 | 1 | 17 | 9 | 508 | 4 | -1 | 0 | 3 |
| **45210** | 37 | 2 | 1 | 1 | 0 | 2971 | 0 | 0 | 0 | 17 | 9 | 361 | 2 | 188 | 11 | 1 |

45211 rows × 16 columns

- **Split Data**

  Split data are used to split the original data into 2 different parts which is training data and testing data. Training data will also be divided into 2 parts training data and validation training data. This process to ensure when the model being trained

and we have a data to validate the predicted output of a model. In this step, we used train_test_split method with ratio 70% for training data and 30% for testing data.

| | | |
|---|---|---|
| Amount of X_train Data | : | no 27916<br>yes 3731<br>dtype: int64 |
| Amount of X_test Data | : | no 12006<br>yes 1558<br>Name: y, dtype: int64 |

- **Scale Data**

  Here we transform the data to fit within a specific scale using these algorithms a change of "1" in any numeric feature will give the same importance to each data. We used StandardScaler() method for SVC, RandomForest, and Logistic Regression algorithm and used MinMaxScaler() method for ANN algorithm from Sklearn library. Define the transformation for train and test data:

  - X_train will be scale using fit_transform()

X_train_scaled

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -1.026305 | 1.423354 | -0.280363 | -1.640840 | -0.13531 | -0.410155 | 0.893093 | -0.436666 | -0.710219 | -0.094313 | 0.827351 | -0.148404 | -0.250493 | 2.575874 | 0.232301 | -2.600080 |
| 1 | -0.367918 | -1.325856 | 1.364851 | -0.298415 | -0.13531 | -0.440750 | -1.119705 | -0.436666 | 1.521344 | 1.470066 | 0.827351 | -0.938292 | 3.303510 | -0.411514 | -0.304903 | 0.443520 |
| 2 | 0.102359 | -1.325856 | -0.280363 | -0.298415 | -0.13531 | -0.440750 | 0.893093 | -0.436666 | -0.710219 | -0.936671 | -0.169407 | 2.929436 | -0.573585 | -0.411514 | -0.304903 | 0.443520 |
| 3 | -0.838194 | -1.020388 | -0.280363 | -0.298415 | -0.13531 | -0.434309 | 0.893093 | 2.290082 | 1.521344 | -0.936671 | 0.827351 | -0.549185 | 0.072598 | -0.411514 | -0.304903 | 0.443520 |
| 4 | 1.513190 | 0.201483 | -0.280363 | -0.298415 | -0.13531 | -0.483583 | -1.119705 | -0.436666 | -0.710219 | 0.387035 | -1.498418 | -0.237899 | -0.250493 | -0.411514 | -0.304903 | 0.443520 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 31642 | -0.744139 | -0.103985 | 1.364851 | 1.044009 | -0.13531 | -0.277792 | -1.119705 | -0.436666 | -0.710219 | -1.418018 | 0.162846 | -0.319611 | -0.573585 | 0.507682 | 0.769506 | -0.571014 |
| 31643 | 0.102359 | 0.812419 | -1.925577 | -0.298415 | -0.13531 | -0.290675 | -1.119705 | -0.436666 | -0.710219 | 1.470066 | -0.501660 | -0.817669 | -0.573585 | -0.411514 | -0.304903 | 0.443520 |
| 31644 | -0.932250 | 1.423354 | 1.364851 | 1.044009 | -0.13531 | 0.082582 | -1.119705 | -0.436666 | -0.710219 | -1.658692 | -0.833913 | -0.389651 | -0.573585 | -0.411514 | -0.304903 | 0.443520 |
| 31645 | -1.308471 | 0.812419 | 1.364851 | -0.298415 | -0.13531 | -0.331575 | -1.119705 | -0.436666 | -0.710219 | -1.418018 | 0.162846 | -0.296265 | -0.573585 | -0.411514 | -0.304903 | 0.443520 |
| 31646 | 0.760747 | -0.103985 | -0.280363 | -0.298415 | -0.13531 | 0.074853 | -1.119705 | -0.436666 | -0.710219 | 1.590403 | -0.169407 | -0.821560 | 0.718780 | -0.411514 | -0.304903 | 0.443520 |

31647 rows × 16 columns

  - X_test will be scale using transform()

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1.325079 | 1.423354 | -1.925577 | -0.298415 | -0.13531 | -0.440750 | 0.893093 | -0.436666 | 1.521344 | 0.507372 | 0.827351 | 0.415802 | 0.395689 | -0.411514 | -0.304903 | 0.44352 |
| 1 | -0.932250 | 0.506951 | 1.364851 | 1.044009 | -0.13531 | 0.145382 | 0.893093 | -0.436666 | 1.521344 | 0.507372 | 0.827351 | -0.739847 | -0.573585 | -0.411514 | -0.304903 | 0.44352 |
| 2 | -1.214416 | -0.103985 | 1.364851 | 1.044009 | -0.13531 | 0.284829 | 0.893093 | -0.436666 | 1.521344 | 0.266698 | 0.162846 | 2.520873 | -0.250493 | -0.411514 | -0.304903 | 0.44352 |
| 3 | -1.214416 | 1.423354 | -0.280363 | -0.298415 | -0.13531 | 1.958202 | 0.893093 | -0.436666 | -0.710219 | -1.057007 | 0.827351 | -0.253463 | -0.573585 | -0.411514 | -0.304903 | 0.44352 |
| 4 | 0.196415 | -0.714920 | -0.280363 | -0.298415 | -0.13531 | -0.421749 | -1.119705 | -0.436666 | 1.521344 | -0.334986 | 0.827351 | 0.695959 | -0.250493 | -0.411514 | -0.304903 | 0.44352 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 13559 | -1.684693 | 0.812419 | -0.280363 | -0.298415 | -0.13531 | -0.436241 | 0.893093 | 2.290082 | 0.405563 | -0.334986 | 0.827351 | -0.930510 | 1.688054 | 3.175350 | 0.232301 | -2.60008 |
| 13560 | -0.932250 | 1.423354 | -0.280363 | -0.298415 | -0.13531 | -0.278759 | -1.119705 | -0.436666 | -0.710219 | 0.507372 | 1.159604 | -0.311829 | -0.573585 | -0.411514 | -0.304903 | 0.44352 |
| 13561 | -0.650084 | 1.423354 | 1.364851 | -0.298415 | -0.13531 | 1.404920 | 0.893093 | -0.436666 | -0.710219 | 0.627708 | -0.169407 | -0.611442 | 0.072598 | -0.411514 | -0.304903 | 0.44352 |
| 13562 | -0.179807 | 0.812419 | -0.280363 | -0.298415 | -0.13531 | -0.424003 | -1.119705 | -0.436666 | 0.405563 | -1.658692 | -0.833913 | 4.520886 | 0.072598 | -0.411514 | -0.304903 | 0.44352 |
| 13563 | 1.231023 | 1.423354 | -0.280363 | -0.298415 | -0.13531 | -0.427224 | -1.119705 | 2.290082 | 1.521344 | -1.538355 | -0.169407 | 0.509188 | -0.250493 | -0.411514 | -0.304903 | 0.44352 |

13564 rows × 16 columns

- **Mix Sampling**

  Here we distributed the data using SMOTE and RandomUnderSampler to make the classification output data balance. We used SMOTE() to duplicates and variance the 'yes' data for the model to learn more variance of yes data. We used RandomUnderSampler() to delete random sample of the 'no' data to make the model not to have low bias towards 'yes' data.

  Total Amount of Data after Mix Sampled :

```
no      8374
yes     4187
dtype: int64
```

- **Create Default Model**

  In this step we create model with default parameters of each algorithm itself. These are default parameters and each algorithm :

| | |
|---|---|
| SVC | `SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0, decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf', max_iter=-1, probability=False, random_state=None, shrinking=True, tol=0.001, verbose=False)` |
| Logistic Regression | `LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True, intercept_scaling=1, l1_ratio=None, max_iter=100, multi_class='auto', n_jobs=None, penalty='l2', random_state=None, solver='lbfgs', tol=0.0001, verbose=0, warm_start=False)` |
| Random Forest | `RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None, criterion='gini', max_depth=None, max_features='auto', max_leaf_nodes=None, max_samples=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None, oob_score=False, random_state=None, verbose=0, warm_start=False)` |
| ANN | `MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08, hidden_layer_sizes=(100,), learning_rate='constant', learning_rate_init=0.001, max_fun=15000, max_iter=200, momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5, random_state=None, shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1, verbose=False, warm_start=False)` |

- **Find Best Parameter Using RandomizedSearch CV**

  After we train the data with default parameter model in above. In this step, we try to find best parameter for each algorithm with focus scoring on Recall because we would like to know if the result of the predicted output made by the model already predict most of the correct data inside the predicted output. To process this step, we should define parameters for each algorithm and run process the RandomizedSeach CV.

| DEFINED PARAMETERS | |
|---|---|
| SVC | `from sklearn.model_selection import RandomizedSearchCV`<br><br>`parameters = {`<br>`'kernel': ['rbf','linear','poly','sigmoid'],`<br>`'C': [1,10,20,50,100],`<br>`'gamma': ['scale','auto'],`<br>`'degree' : [3,5,7]`<br>`}` |
| Logistic Regression | `parameters = {`<br>`'tol': [0.00001, 0.0001, 0.001],`<br>`'C': [0.001,0.01,0.1,1,10,100,1000],`<br>`'class_weight': ['balanced', None],`<br>`'random_state' : [None,10,20,30,42],`<br>`'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],`<br>`'max_iter' : [100],`<br>`'verbose':[0,1,2],`<br>`}` |
| Random Forest | `parameters = {`<br>`'criterion':['gini','entropy'],`<br>`'bootstrap':[True,False],`<br>`'oob_score':[True,False],`<br>`'warm_start':[True,False],`<br>`'class_weight' : ['balanced', 'balanced_subsample']`<br>`}` |
| ANN | `parameters = {`<br>`'hidden_layer_sizes': [(50,50,50), (50,100,50), (100,)],`<br>`'activation': ['tanh', 'relu','logistic'],`<br>`'solver': ['sgd', 'adam'],`<br>`'alpha': [0.0001, 0.05],`<br>`'learning_rate': ['constant','adaptive','invscaling'],`<br>`}` |

| BEST PARAMETERS | |
|---|---|
| SVC | `{'C': 20, 'degree': 7, 'gamma': 'auto', 'kernel': 'rbf'}` |
| Logistic Regression | `{'C': 0.01,`<br>`'class_weight': 'balanced',`<br>`'max_iter': 100,`<br>`'random_state': 20,`<br>`'solver': 'liblinear',`<br>`'tol': 0.001,`<br>`'verbose': 1}` |

| | |
|---|---|
| Random Forest | {'bootstrap': True,<br>'class_weight': 'balanced_subsample',<br>'criterion': 'entropy',<br>'oob_score': True,<br>'warm_start': True} |
| ANN | {'activation': 'tanh',<br>'alpha': 0.0001,<br>'hidden_layer_sizes': (50, 50, 50),<br>'learning_rate': 'adaptive',<br>'solver': 'adam'} |

- **Create Hyper Parameterized Model**

| HYPER PARAMETERIZED MODEL | |
|---|---|
| SVC | `model = SVC(kernel = 'rbf', C=20, gamma='auto',degree=7)` |
| Logistic Regression | `model = LogisticRegression(C= 0.01,`<br>`class_weight= 'balanced',`<br>`max_iter= 100,`<br>`random_state= 20,`<br>`solver= 'liblinear',`<br>`tol= 0.001,`<br>`verbose= 1)` |
| Random Forest | `model = RandomForestClassifier(class_weight='balanced_subsample',`<br>`bootstrap=True,`<br>`criterion='entropy',oob_score=True,`<br>`warm_start=True)` |
| ANN | `model = MLPClassifier(hidden_layer_sizes=(50,50,50),activation='tanh',`<br>`alpha=0.0001,learning_rate='adaptive',`<br>`solver='adam')` |

# 4. RESULT

## a. CONFUSION MATRIX

Here we used confusion matrix which is crosstab() method from pandas There are 4 terms as a representation of the result of the classification process confusion matrix. The four terms:

- True Positive (TP) : Represents positive data that is predicted to be correct
- True Negative (TN) : Represent negative data that is predicted to be correct
- False Negative (FN) : Type I Error - Represent negative data but predicted as positive data
- False Positive (FP) : Type II Error - Represents positive data but predicted as negative data

| | SVC | | | |
|---|---|---|---|---|
| | **TP** | **TN** | **FN** | **FP** |
| **Default Parameter** | 1129 | 10689 | 429 | 1317 |
| **Hyper Parameterized** | 1082 | 10603 | 476 | 1403 |
| | | | | |
| | **Logistic Regression** | | | |
| | **TP** | **TN** | **FN** | **FP** |
| **Default Parameter** | 952 | 10843 | 606 | 1163 |
| **Hyper Parameterized** | 1256 | 9646 | 302 | 2360 |
| | | | | |
| | **Random Forest** | | | |
| | **TP** | **TN** | **FN** | **FP** |
| **Default Parameter** | 1199 | 10732 | 359 | 1274 |
| **Hyper Parameterized** | 1200 | 10767 | 358 | 1239 |
| | | | | |
| | **ANN** | | | |
| | **TP** | **TN** | **FN** | **FP** |
| **Default Parameter** | 1178 | 10480 | 380 | 1526 |
| **Hyper Parameterized** | 1194 | 10470 | 364 | 1536 |
| | | | | |

## b. CLASSIFICATION REPORT

Here we used the classification report from sklearn library in the classification report function we have precision, recall, f1-score and support for the evaluation metrics.

| | | SVC | | | | |
|---|---|---|---|---|---|---|
| | | **Precision** | **Recall** | **F1 Score** | **Support** | **Accuracy** |
| **Default Parameter** | No | 0.96 | 0.89 | 0.92 | 12006 | 0.87 |
| | Yes | 0.46 | 0.72 | 0.56 | 1558 | |
| **Hyper Parameterized** | No | 0.96 | 0.88 | 0.92 | 12006 | 0.86 |
| | Yes | 0.44 | 0.69 | 0.54 | 1558 | |
| | | **Logistic Regression** | | | | |
| | | **Precision** | **Recall** | **F1 Score** | **Support** | **Accuracy** |
| **Default Parameter** | No | 0.95 | 0.90 | 0.92 | 12006 | 0.87 |
| | Yes | 0.45 | 0.61 | 0.52 | 1558 | |
| **Hyper Parameterized** | No | 0.97 | 0.80 | 0.88 | 12006 | 0.80 |
| | Yes | 0.35 | 0.81 | 0.49 | 1558 | |
| | | **Random Forest** | | | | |
| | | **Precision** | **Recall** | **F1 Score** | **Support** | **Accuracy** |
| **Default Parameter** | No | 0.97 | 0.89 | 0.93 | 12006 | 0.88 |
| | Yes | 0.48 | 0.77 | 0.59 | 1558 | |
| **Hyper Parameterized** | No | 0.97 | 0.90 | 0.93 | 12006 | 0.88 |
| | Yes | 0.49 | 0.77 | 0.60 | 1558 | |
| | | **ANN** | | | | |
| | | **Precision** | **Recall** | **F1 Score** | **Support** | **Accuracy** |
| **Default Parameter** | No | 0.97 | 0.87 | 0.92 | 12006 | 0.86 |
| | Yes | 0.44 | 0.76 | 0.55 | 1558 | |
| **Hyper Parameterized** | No | 0.97 | 0.87 | 0.92 | 12006 | 0.86 |
| | Yes | 0.44 | 0.77 | 0.56 | 1558 | |

# 5. DISCUSSION

In this project dataset there are a lot of outliers that affect to the model result. The original dataset also imbalanced between “No” data and “Yes” data, but however we already fix the imbalanced data with Mix Sampled.

For this project, we use SVC, Logistic Regression, Random Forest and ANN for the machine learning algorithm the data that we want to predict in the future. For the result of the model, since we are focusing on the Recall score of the model, the best model for Portugese Bank Project is Logistic Regression in Hyper Parameterized Model. Because based on the classification report in above, Logistic Regression in Hyper Parameterized Model has a good result in Recall Score with 80% for “No” data and 81% for “Yes” data

# 6. CONCLUSION

In this project, if Portugese Bank User or Director of Data want to use a Machine Learning for predicting data in the future. He can use :

- Random Forest : If he want to have good model in Precision, F1 Score and Accuracy
- Logistic Regression : If he want to have good model in Recall Score