# Logistic Regression

dataset: Example_bank_data.csv

The data is based on the marketing campaign efforts of a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit (variable y).

Source: [Moro et al., 2014] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014

## Import the relevant libraries

In [8]:

```
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

# this part not be needed after the latests updates of the library
from scipy import stats
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)
```

## Load the data

Load the 'Example_bank_data.csv' dataset.

In [9]:

```
raw_data = pd.read_csv('Example_bank_data.csv')
raw_data
```

Out[9]:

| | Unnamed: 0 | duration | y |
|---|---|---|---|
| **0** | 0 | 117 | no |
| **1** | 1 | 274 | yes |
| **2** | 2 | 167 | no |
| **3** | 3 | 686 | yes |
| **4** | 4 | 157 | no |
| **...** | ... | ... | ... |
| **513** | 513 | 204 | no |
| **514** | 514 | 806 | yes |
| **515** | 515 | 290 | no |
| **516** | 516 | 473 | yes |
| **517** | 517 | 142 | no |

518 rows × 3 columns

We want to know whether the bank marketing strategy was successful, so we need to transform the outcome variable into 0s and 1s in order to perform a logistic regression.

In [10]:

```
data = raw_data.copy()
data = data.drop(['Unnamed: 0'], axis = 1)

# We use the map function to change any 'yes' values to 1 and 'no' values to 0.
data['y'] = data['y'].map({'yes':1, 'no':0})
data
```

Out[10]:

| | duration | y |
|---|---|---|
| **0** | 117 | 0 |
| **1** | 274 | 1 |
| **2** | 167 | 0 |
| **3** | 686 | 1 |
| **4** | 157 | 0 |
| **...** | ... | ... |
| **513** | 204 | 0 |
| **514** | 806 | 1 |
| **515** | 290 | 0 |
| **516** | 473 | 1 |
| **517** | 142 | 0 |

518 rows × 2 columns

In [11]:

```
# descriptive statistics
data.describe()
```

Out[11]:

| | duration | y |
|---|---|---|
| **count** | 518.000000 | 518.000000 |
| **mean** | 382.177606 | 0.500000 |
| **std** | 344.295990 | 0.500483 |
| **min** | 9.000000 | 0.000000 |
| **25%** | 155.000000 | 0.000000 |
| **50%** | 266.500000 | 0.500000 |
| **75%** | 482.750000 | 1.000000 |
| **max** | 2653.000000 | 1.000000 |

## Declare the dependent and independent variables

In [12]:

```
y = data['y']
x1 = data['duration']
```

## Simple Logistic Regression

In [13]:

```
x = sm.add_constant(x1)
reg_log = sm.Logit(y,x)
results_log = reg_log.fit()

# Get the regression summary
results_log.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.546118
         Iterations 7
```

Out[13]:

Logit Regression Results

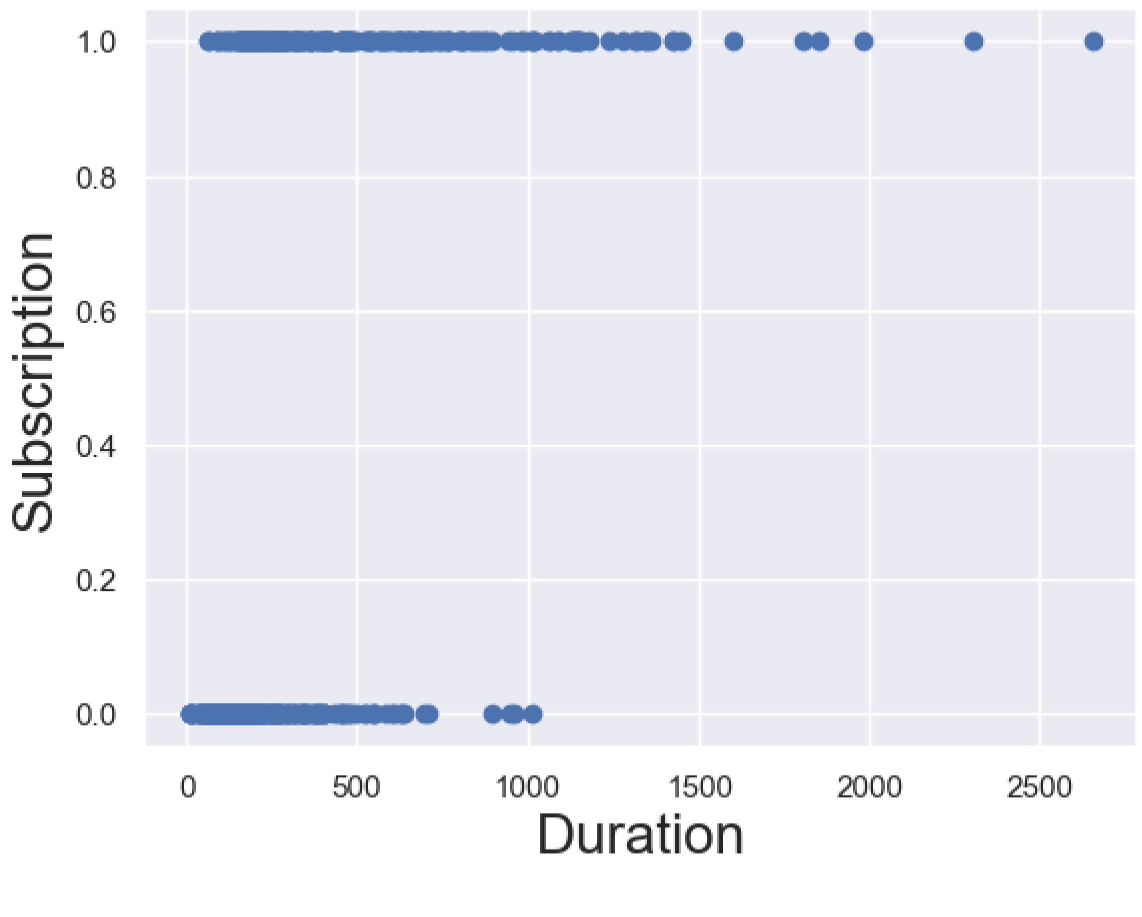
| | | | |
|---|---|---|---|
| **Dep. Variable:** | y | **No. Observations:** | 518 |
| **Model:** | Logit | **Df Residuals:** | 516 |
| **Method:** | MLE | **Df Model:** | 1 |
| **Date:** | Wed, 26 Oct 2022 | **Pseudo R-squ.:** | 0.2121 |
| **Time:** | 11:24:02 | **Log-Likelihood:** | -282.89 |
| **converged:** | True | **LL-Null:** | -359.05 |
| **Covariance Type:** | nonrobust | **LLR p-value:** | 5.387e-35 |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **const** | -1.7001 | 0.192 | -8.863 | 0.000 | -2.076 | -1.324 |
| **duration** | 0.0051 | 0.001 | 9.159 | 0.000 | 0.004 | 0.006 |

In [14]:

```
# Create a scatter plot of x1 (Duration, no constant) and y (Subscribed)
plt.scatter(x1,y,color = 'C0')

# labels
plt.xlabel('Duration', fontsize = 20)
plt.ylabel('Subscription', fontsize = 20)
plt.show()
```


Subscription
Duration