Building a Logistic Regression

Create a logistic regression based on the bank data provided.

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).

Note that the first column of the dataset is the index.

Import the relevant libraries

import pandas as pd

import numpy as np

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 ‘Example\_bank\_data.csv’ dataset.

from google.colab import files

uploaded = files.upload()

raw\_data = pd.read\_csv('Example\_bank\_data.csv')

raw\_data

**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.**

# We make sure to create a copy of the data before we start altering itNote that we don't change the original data we loaded.

data = raw\_data.copy()

# Removes the index column that came with the data

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

# Check the descriptive statistics

data.describe()

### Declare the dependent and independent variables

y = data['y']

x1 = data['duration']

### Simple Logistic Regression

x = sm.add\_constant(x1)

reg\_log = sm.Logit(y,x)

results\_log = reg\_log.fit()

# Get the regression summary

results\_log.summary()

# Create a scatter plot of x1 (Duration, no constant) and y (Subscribed)

plt.scatter(x1,y,color = 'C0')

# Don't forget to label your axes!

plt.xlabel('Duration', fontsize = 20)

plt.ylabel('Subscription', fontsize = 20)

plt.show()

np.set\_printoptions(formatter={'float': lambda x: "{0:0.2f}".format(x)})

#np.set\_printoptions(formatter=None)

results\_log.predict()

np.array(data['y'])

results\_log.pred\_table()

cm\_df = pd.DataFrame(results\_log.pred\_table())

cm\_df.columns = ['Predicted 0','Predicted 1']

cm\_df = cm\_df.rename(index={0: 'Actual 0',1:'Actual 1'})

cm\_df

cm = np.array(cm\_df)

accuracy\_train = (cm[0,0]+cm[1,1])/cm.sum()

accuracy\_train