Classification Prediction Model: Client Term Bank Deposit

By: Katrina James, Christian Urday, Justin Kerry

Introductory Statement

Banks offer a wide variety of financial services to the population and have long found themselves as a critical element in the foundation of our civilization. As a society, we rely on banks as a major pillar for the overall national economic health and stability. As individuals, we rely on banks as institutions that allow us to hold, borrow, and deposit money in a highly organized and archived manner. Due to this, banking is widely considered a service and an ally for many people. Although that is true, banking should also be considered a business. And like any other type of business, there are limitations and restrictions held in place to protect and safeguard the business. Some of these limitations and restrictions can include charging clients a penalty for early withdrawals and maintaining interest rates against rising inflation. What we are studying here is called a term deposit.

Background

A term deposit is a fixed term investment made by a client when they deposit money into an account. This investment is then taken by the bank and loaned to other clients or invested in other finacial products with a higher rate of return. Since banks can be considered businesses, they ideally want to pay back to the investor the lowest possible rate of interest for the term deposit and generate the highest possible rate of interest through the loan or product of investment. Ultimately term deposits can be attractive for low risk investors since they are risk free however certain restrictions can lead to clients opting out of a term deposit. A term deposit that is made must be held by the bank for a specified time period and not withdrawn earlier or there will be a penalty charged. Interest rates paid to investors do not keep up with the rising level of inflation over time. Many other fixed-rate investments pay higher interest rates than term deposits. So even though term deposits can be attractive to investors because they are a low risk investment, these restrictions can lead to clients choosing not to invest in term deposits.

Objective

We will be using a machine learning model to predict whether or not a bank client will choose to invest in the low risk yet limited investment known as a term deposit. This can be achieved by utilizing some dependent variables that have a great deal of predictive power while at the same time avoiding multi-collinearity that will hinder the coefficients of our independent variables creating bias in our models.

Analysis

For our research, we will use a dataset that we sourced from Kaggle called Bank Marketing (Binary Classification).

This dataset was created by: Paulo Cortez (Univ. Minho) and Sérgio Moro (ISCTE-IUL) @ 2012. It was fully described and analysed in S. Moro, R. Laureano and P. Cortez. Using Data Mining for Bank Direct Marketing: An Application of the CRISP-DM Methodology.

For the purposes of the assignment we have extracted the full-dataset which includes over 40,000 records. The number of attributes that this dataset contains is 16 outputs.

The following code will generate the dataset as a viewable table. This will allow us to view all variables including the 'deposit' column which as mentioned previously is our target variable:

In [2]:
```
import numpy as np
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, auc, classification_report
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.pipeline import Pipeline
import statsmodels.api as sm

pd.set_option('display.max_columns', None)

bank_df = pd.read_csv(os.path.join('data', 'Bank.csv'), index_col=0)
bank_df.head()
```

Out[2]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duratior
0	44	blue-collar	married	basic.4y	unknown	yes	no	cellular	aug	thu	21(
1	53	technician	married	unknown	no	no	no	cellular	nov	fri	13ɛ
2	28	management	single	university.degree	no	yes	no	cellular	jun	thu	33ç
3	39	services	married	high.school	no	no	no	cellular	apr	fri	18!
4	55	retired	married	basic.4y	no	yes	no	cellular	aug	fri	13;

Purpose

The dataset was collected with the intention of predicting whether or not the client of a bank will subscribe to a term deposit. We decided on this dataset because we found that it was clear and concise. It also has sufficient records of data as well as sufficient independent variables that can be considered relatable to our target variable.

Representation

For the purposes of our classification prediction models, we can define and measure the outcomes from the dataset using just a couple of different categorical values. These values include people who did subscribe for term deposit encoded as '1' and people who did not subscribe for the deposit encoded as '0'. We will encode these target variables as binary values to the dataset using the .map function in python.

In [3]:
```
bank_df['deposit'] = bank_df['deposit'].map({'yes': 1, 'no': 0})
bank_df.head()
```

Out[3]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duratior
0	44	blue-collar	married	basic.4y	unknown	yes	no	cellular	aug	thu	21(
1	53	technician	married	unknown	no	no	no	cellular	nov	fri	13ɛ
2	28	management	single	university.degree	no	yes	no	cellular	jun	thu	33ç
3	39	services	married	high.school	no	no	no	cellular	apr	fri	18!
4	55	retired	married	basic.4y	no	yes	no	cellular	aug	fri	13;

We had to represent 'was not previously contacted' (pdays = 999) as a boolean value which is now shown in the following table under 'client_was_contacted' (0 for no and 1 for yes). This feature engineering is important since it allows our models to recognize whether the client was contacted or not by the bank.

In [4]:
```
bank_df['client_was_contacted'] = np.where(bank_df['pdays'] == 999, 0, 1)
bank_df['pdays'] = np.where(bank_df['pdays'] == 999, 0, bank_df['pdays'])
bank_df.head()
```

Out[4]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duratior
0	44	blue-collar	married	basic.4y	unknown	yes	no	cellular	aug	thu	21(
1	53	technician	married	unknown	no	no	no	cellular	nov	fri	13{
2	28	management	single	university.degree	no	yes	no	cellular	jun	thu	33{
3	39	services	married	high.school	no	no	no	cellular	apr	fri	18{
4	55	retired	married	basic.4y	no	yes	no	cellular	aug	fri	13

We will also measure the effectiveness of our algorithms using a Test Confusion Matrix which will show us a test accuracy score likely displayed as a decimal number to provide us with a percentage. Confusion Matrix for both models will be displayed later in this report at appropriate times.

Constructing a Final Dataset and Building our Logistic Regression Model

In order for our classification prediction models to be more accurate, we had to modify some variables in our dataset. First we had to use the info_value_calc function with an if statement in python to model text data as categorical and numerical data as continous. Continuous data has been binned into decimals for normalization:

In [5]:
```
def info_value_calc(df, column, is_categorical):
    if is_categorical:
        info_val_df = df.groupby([column])['deposit'].agg(['count', 'sum'])
    else:
        df['variable_bin'] = pd.qcut(df[column].rank(method='first'), 10)
        info_val_df = df.groupby(['variable_bin'])['deposit'].agg(['count', 'sum'])

    info_val_df = info_val_df.rename(columns={'sum': 'bad'})
    info_val_df["good"] = info_val_df["count"] - info_val_df["bad"]
    info_val_df["bad_percentage"] = info_val_df["bad"] / info_val_df["bad"].sum()
    info_val_df["good_percentage"] = info_val_df["good"] / info_val_df["good"].sum()
    info_val_df["information_value"] = info_val_df.apply(lambda row: (row['good_percentage
    return info_val_df
```

In [6]:
```
print(info_value_calc(bank_df, 'education', True))
print('Total information value: ' + str(info_value_calc(bank_df, 'education', True)['infor
```

```
                     count   bad   good  bad_percentage  good_percentage  \
education
basic.4y              4176   428   3748        0.092241         0.102550
basic.6y              2292   188   2104        0.040517         0.057568
basic.9y              6045   473   5572        0.101940         0.152457
high.school           9515  1031   8484        0.222198         0.232133
illiterate              18     4     14        0.000862         0.000383
professional.course   5243   595   4648        0.128233         0.127175
university.degree    12168  1670  10498        0.359914         0.287239
unknown               1731   251   1480        0.054095         0.040495

                     information_value
education
basic.4y                      0.001092
basic.6y                      0.005989
basic.9y                      0.020333
high.school                   0.000435
illiterate                    0.000389
professional.course           0.000009
university.degree             0.016392
unknown                       0.003938
```

```
Total information value: 0.048576408429404055
```

Predictive Power

Another process we had to consider was removing unnecessary variables that didn't have any predictive power for our models. Before we could do that however we had to address other considerations prior. We first performed an information value calculation to determine predictive power. When we printed out the information value we ended up with the following results:

information_value column 13 1.961060 duration 19 1.092445 nr_employed 18 1.059292 euribor3m 15 0.777944 emp_var_rate 17 0.622916 cons_conf_idx 11 0.551306 client_was_contacted 10 0.547671 poutcome 7 0.485117 month 16 0.449511 cons_price_idx 20 0.261490 previous 6 0.251663 contact 14 0.244849 pdays 0 0.188713 job 3 0.127851 default 12 0.127037 age 9 0.063208 campaign 2 0.048576 education 1 0.028215 marital 8 0.006493 day_of_week 4 0.001383 housing 5 0.000269 loan

Now that we have an idea of predictive power, we have a better understanding on which variables can be removed. However before we start removing them, we should manipulate the data further to achieve more specific information regarding our data.

Dummy Variables

In order for our models to handle categorical data types, we will need to transpose them into dummy variables:

In [7]:
```
client_contacted_dummies = pd.get_dummies(bank_df['client_was_contacted'], prefix='client_
poutcome_dummies = pd.get_dummies(bank_df['poutcome'], prefix='poutcome')
month_dummies = pd.get_dummies(bank_df['month'], prefix='month')
contact_dummies = pd.get_dummies(bank_df['contact'], prefix='contact')
job_dummies = pd.get_dummies(bank_df['job'], prefix='job')
default_dummies = pd.get_dummies(bank_df['default'], prefix='default')

bank_model_df = pd.concat([bank_df[['duration', 'nr_employed', 'euribor3m', 'emp_var_rate'
                                   'pdays', 'age', 'previous']],
                          client_contacted_dummies, poutcome_dummies, month_dummies,
                          contact_dummies, job_dummies, default_dummies], axis=1)
bank_model_df.head()
```

Out[7]:

	duration	nr_employed	euribor3m	emp_var_rate	cons_conf_idx	cons_price_idx	deposit	pdays	age	previ
0	210	5228.1	4.963	1.4	-36.1	93.444	0	0	44	
1	138	5195.8	4.021	-0.1	-42.0	93.200	0	0	53	
2	339	4991.6	0.729	-1.7	-39.8	94.055	1	6	28	
3	185	5099.1	1.405	-1.8	-47.1	93.075	0	0	39	
4	137	5076.2	0.869	-2.9	-31.4	92.201	1	3	55	

Removing Variables and Feature Selection

As previously mentioned, we had to get rid of certain variables that didn't have any predictive power for our machine learning models. We can start by removing excess dummy variables created by pandas when we initially called for them.

Let's split our data into the train set and the test set. Then we will remove the excess dummy variables from the train set. That way they won't be used during testing process.

In [8]:
```
x_train, x_test, y_train, y_test = train_test_split(bank_model_df.drop(['deposit'] ,axis=1
                                                    bank_model_df['deposit'],
                                                    train_size=0.7, # 70-30 split
                                                    random_state=42) # constant seed allow
y_train = pd.DataFrame(y_train)
y_test = pd.DataFrame(y_test)
```

In [9]:
```
columns_to_drop = ['client_was_contacted_1', 'poutcome_nonexistent', 'month_apr', 'contact
logistic_model = sm.Logit(y_train, sm.add_constant(x_train.drop(columns_to_drop, axis=1)))
print(logistic_model.summary())
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.202419
         Iterations: 1000
                           Logit Regression Results
==============================================================================
Dep. Variable:                deposit   No. Observations:                28831
Model:                          Logit   Df Residuals:                    28795
Method:                           MLE   Df Model:                           35
Date:                Sat, 13 Feb 2021   Pseudo R-squ.:                  0.4210
Time:                        18:02:02   Log-Likelihood:                -5836.0
converged:                      False   LL-Null:                       -10079.
Covariance Type:            nonrobust   LLR p-value:                     0.000
========================================================================================
                           coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------------
const                 -288.6847     46.823     -6.165      0.000    -380.455    -196.914
duration                 0.0049   9.21e-05     53.610      0.000       0.005       0.005
nr_employed              0.0083      0.004      2.200      0.028       0.001       0.016
euribor3m                0.3740      0.156      2.404      0.016       0.069       0.679
emp_var_rate            -2.0565      0.175    -11.729      0.000      -2.400      -1.713
cons_conf_idx            0.0257      0.009      2.745      0.006       0.007       0.044
cons_price_idx           2.5887      0.310      8.359      0.000       1.982       3.196
pdays                   -0.0164      0.022     -0.751      0.453      -0.059       0.026
age                     -0.0032      0.003     -1.200      0.230      -0.008       0.002
previous                -0.0557      0.074     -0.751      0.453      -0.201       0.090
client_was_contacted_0  -1.0519      0.358     -2.935      0.003      -1.754      -0.350
poutcome_failure        -0.5093      0.117     -4.367      0.000      -0.738      -0.281
poutcome_success         0.5638      0.283      1.994      0.046       0.010       1.118
month_aug                0.9459      0.147      6.441      0.000       0.658       1.234
month_dec                0.3735      0.249      1.499      0.134      -0.115       0.862
month_jul                0.1779      0.116      1.536      0.124      -0.049       0.405
month_jun               -0.6102      0.153     -3.980      0.000      -0.911      -0.310
month_mar                2.1335      0.174     12.268      0.000       1.793       2.474
month_may               -0.3623      0.100     -3.630      0.000      -0.558      -0.167
month_nov               -0.3852      0.146     -2.633      0.008      -0.672      -0.098
month_oct                0.1542      0.188      0.822      0.411      -0.214       0.522
month_sep                0.5423      0.219      2.475      0.013       0.113       0.972
contact_telephone       -0.7579      0.095     -8.014      0.000      -0.943      -0.573
job_admin.               0.2625      0.146      1.799      0.072      -0.023       0.548
job_blue-collar         -0.0288      0.152     -0.190      0.849      -0.326       0.269
job_entrepreneur         0.1851      0.197      0.941      0.347      -0.201       0.571
job_housemaid            0.2303      0.217      1.059      0.290      -0.196       0.657
job_management           0.3112      0.164      1.898      0.058      -0.010       0.633
job_retired              0.3646      0.182      2.007      0.045       0.009       0.721
job_services             0.0861      0.164      0.526      0.599      -0.235       0.407
job_student              0.3539      0.184      1.927      0.054      -0.006       0.714
job_technician           0.2360      0.151      1.558      0.119      -0.061       0.533
job_unemployed           0.0618      0.203      0.304      0.761      -0.337       0.460
job_unknown              0.3260      0.320      1.018      0.309      -0.302       0.954
default_no               0.2927      0.081      3.620      0.000       0.134       0.451
default_yes            -16.2398   2.64e+04     -0.001      1.000   -5.18e+04    5.18e+04
========================================================================================
/Applications/anaconda3/lib/python3.8/site-packages/statsmodels/base/model.py:566: Converg
enceWarning: Maximum Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

--

The excess dummy variables have now been removed from the dataset and we can begin to narrow down which variables are unneccesary or detrimental to our predictive power.

As we eliminate the undesirable variables, we can also simultaneously build our Logistic Regression model.

First let us check for variables that hinder our independent coefficients through multi-collinearity. We can achieve this through the use of the variance inflation factor:

In [10]:
```
variance_inflation_list = []
variables = pd.Series(x_train.drop(columns_to_drop, axis=1).columns)
completed_cols = []
variance_inflation_factors = []

for variable in variables:
    completed_cols.append(variable)
    dependent_variable = variable
    independent_variables = variables[~variables.isin(completed_cols)]
    mod = sm.OLS(x_train.drop(columns_to_drop, axis=1)[dependent_variable], sm.add_constar
    residuals = mod.fit()
    variance_inflation_factor = 1 / (1 - residuals.rsquared)
    variance_inflation_factors.append({'dependent_variable': dependent_variable, 'variance

print(pd.DataFrame(variance_inflation_factors))
```

```
        dependent_variable  variance_inflation_factor
0                 duration                   1.011601
1              nr_employed                 199.199988
2                euribor3m                 104.090973
3              emp_var_rate                   7.440831
4             cons_conf_idx                   2.220549
5            cons_price_idx                   2.621768
6                    pdays                   3.963402
7                      age                   1.405326
8                 previous                   5.568172
9     client_was_contacted_0                  11.033087
10         poutcome_failure                   1.157539
11         poutcome_success                   1.073545
12                month_aug                   2.925225
13                month_dec                   1.024404
14                month_jul                   1.522485
15                month_jun                   1.601930
16                month_mar                   1.021689
17                month_may                   1.214999
18                month_nov                   1.059931
19                month_oct                   1.016458
20                month_sep                   1.014909
21         contact_telephone                   1.030093
22               job_admin.                   6.248120
23          job_blue-collar                   1.427232
24         job_entrepreneur                   1.030387
25            job_housemaid                   1.021212
26           job_management                   1.046097
27              job_retired                   1.020034
28             job_services                   1.029785
29              job_student                   1.006696
30           job_technician                   1.011123
31           job_unemployed                   1.000913
32              job_unknown                   1.003515
33               default_no                   1.000268
34              default_yes                   1.000000
```

--

Now that we have access to the variance inflation factors for each variable, we should determine the C-Statistic (AUC) which will tell us how accurate the classifier is at predicting the target variable.

--

In [11]:
```
def df_crossjoin(df1, df2, **kwargs):
    df1['temp_key'] = 1
    df2['temp_key'] = 1
    return_df = pd.merge(df1, df2, on='temp_key', **kwargs).drop('temp_key', axis=1)
    return_df.index = pd.MultiIndex.from_product((df1.index, df2.index))
    return return_df

y_prediction = pd.DataFrame(logistic_model.predict(sm.add_constant(x_train.drop(columns_tc
y_prediction.columns = ["probabilities"]
both_df = pd.concat([y_train, y_prediction], axis=1)
zeros_df = both_df[['deposit', 'probabilities']][both_df['deposit'] == 0]
ones_df = both_df[['deposit', 'probabilities']][both_df['deposit'] == 1]
joined_df = df_crossjoin(ones_df, zeros_df)
joined_df['concordant_pair'] = 0
joined_df.loc[joined_df['probabilities_x'] > joined_df['probabilities_y'], 'concordant_pai
joined_df['discordant_pair'] = 0
joined_df.loc[joined_df['probabilities_x'] < joined_df['probabilities_y'], 'discordant_pai
joined_df['tied_pair'] = 0
joined_df.loc[joined_df['probabilities_x'] == joined_df['probabilities_y'], 'tied_pair'] =
p_concordant = (sum(joined_df['concordant_pair']) * 1.0 ) / (joined_df.shape[0])
p_discordant = (sum(joined_df['discordant_pair']) * 1.0 ) / (joined_df.shape[0])
c_statistic = 0.5 + (p_concordant - p_discordant) / 2.0
print("C-statistic: " + str(c_statistic))
```

```
C-statistic: 0.9385172268269756
```

--

Our C-statistic is 0.9 and since that statistic is >0.7 our Logistic Regression model can be considered very strong at predicting the target variable. This is great news and now we can begin the process of removing the unneccessary variables that are multi-colinear. We can determine which variables can be removed by analysing the data to see which variables have a high p-value and a high variance inflation factor.

The following code contains the columns that we have already dropped when we removed the excess dummy variables. Only now we will include the column 'default_yes' which will be the first variable dropped due to multi-collinearity.

--

In [12]:
```
columns_to_drop = ['client_was_contacted_1', 'poutcome_nonexistent', 'month_apr', 'contact
logistic_model = sm.Logit(y_train, sm.add_constant(x_train.drop(columns_to_drop, axis=1)))
print(logistic_model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.202420
         Iterations 10
                           Logit Regression Results
==============================================================================
Dep. Variable:                deposit   No. Observations:                28831
Model:                          Logit   Df Residuals:                    28796
Method:                           MLE   Df Model:                           34
Date:                Sat, 13 Feb 2021   Pseudo R-squ.:                  0.4210
Time:                        18:04:44   Log-Likelihood:                -5836.0
converged:                       True   LL-Null:                       -10079.
Covariance Type:            nonrobust   LLR p-value:                     0.000
========================================================================================
                           coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------------
const                 -288.6970     46.823     -6.166      0.000    -380.468    -196.926
duration                 0.0049   9.21e-05     53.611      0.000       0.005       0.005
nr_employed              0.0083      0.004      2.200      0.028       0.001       0.016
euribor3m                0.3740      0.156      2.403      0.016       0.069       0.679
emp_var_rate            -2.0566      0.175    -11.730      0.000      -2.400      -1.713
cons_conf_idx            0.0257      0.009      2.745      0.006       0.007       0.044
cons_price_idx           2.5888      0.310      8.359      0.000       1.982       3.196
pdays                   -0.0164      0.022     -0.751      0.453      -0.059       0.026
age                     -0.0032      0.003     -1.200      0.230      -0.008       0.002
```

previous	-0.0557	0.074	-0.751	0.453	-0.201	0.090
client_was_contacted_0	-1.0519	0.358	-2.935	0.003	-1.754	-0.350
poutcome_failure	-0.5094	0.117	-4.368	0.000	-0.738	-0.281
poutcome_success	0.5638	0.283	1.994	0.046	0.010	1.118
month_aug	0.9459	0.147	6.441	0.000	0.658	1.234
month_dec	0.3736	0.249	1.499	0.134	-0.115	0.862
month_jul	0.1779	0.116	1.537	0.124	-0.049	0.405
month_jun	-0.6103	0.153	-3.980	0.000	-0.911	-0.310
month_mar	2.1336	0.174	12.268	0.000	1.793	2.474
month_may	-0.3623	0.100	-3.630	0.000	-0.558	-0.167
month_nov	-0.3852	0.146	-2.633	0.008	-0.672	-0.098
month_oct	0.1542	0.188	0.822	0.411	-0.214	0.522
month_sep	0.5423	0.219	2.475	0.013	0.113	0.972
contact_telephone	-0.7579	0.095	-8.014	0.000	-0.943	-0.573
job_admin.	0.2625	0.146	1.799	0.072	-0.023	0.548
job_blue-collar	-0.0288	0.152	-0.190	0.850	-0.326	0.269
job_entrepreneur	0.1851	0.197	0.941	0.347	-0.201	0.571
job_housemaid	0.2303	0.217	1.059	0.290	-0.196	0.657
job_management	0.3112	0.164	1.898	0.058	-0.010	0.633
job_retired	0.3646	0.182	2.007	0.045	0.009	0.721
job_services	0.0862	0.164	0.526	0.599	-0.235	0.407
job_student	0.3539	0.184	1.927	0.054	-0.006	0.714
job_technician	0.2360	0.151	1.558	0.119	-0.061	0.533
job_unemployed	0.0616	0.203	0.303	0.762	-0.337	0.460
job_unknown	0.3260	0.320	1.018	0.309	-0.301	0.954
default_no	0.2928	0.081	3.621	0.000	0.134	0.451

--

Now instead of showing the code for each elimination, we will simply show each Logit Regression Results table to demonstrate how the statistics for each attribute will change everytime an attribute is eliminated.

--

Optimization terminated successfully. Current function value: 0.202422 Iterations 10 Logit Regression Results == Dep. Variable: deposit No. Observations: 28831 Model: Logit Df Residuals: 28797 Method: MLE Df Model: 33 Date: Thu, 11 Feb 2021 Pseudo R-squ.: 0.4210 Time: 19:14:26 Log-Likelihood: -5836.0 converged: True LL-Null: -10079. Covariance Type: nonrobust LLR p-value: 0.000 == coef std err z P>|z| [0.025 0.975] -- const -288.5897 46.819 -6.164 0.000 -380.354 -196.826 duration 0.0049 9.21e-05 53.611 0.000 0.005 0.005 nr_employed 0.0083 0.004 2.198 0.028 0.001 0.016 euribor3m 0.3740 0.156 2.403 0.016 0.069 0.679 emp_var_rate -2.0563 0.175 -11.729 0.000 -2.400 -1.713 cons_conf_idx 0.0257 0.009 2.747 0.006 0.007 0.044 cons_price_idx 2.5884 0.310 8.358 0.000 1.981 3.195 pdays -0.0164 0.022 -0.751 0.453 -0.059 0.026 age -0.0032 0.003 -1.198 0.231 -0.008 0.002 previous -0.0556 0.074 -0.750 0.453 -0.201 0.090 client_was_contacted_0 -1.0512 0.358 -2.933 0.003 -1.754 -0.349 poutcome_failure -0.5097 0.117 -4.371 0.000 -0.738 -0.281 poutcome_success 0.5649 0.283 1.998 0.046 0.011 1.119 month_aug 0.9463 0.147 6.444 0.000 0.658 1.234 month_dec 0.3746 0.249 1.503 0.133 -0.114 0.863 month_jul 0.1785 0.116 1.542 0.123 -0.048 0.405 month_jun -0.6094 0.153 -3.975 0.000 -0.910 -0.309 month_mar 2.1337 0.174 12.269 0.000 1.793 2.475 month_may -0.3619 0.100 -3.627 0.000 -0.558 -0.166 month_nov -0.3848 0.146 -2.630 0.009 -0.671 -0.098 month_oct 0.1547 0.188 0.825 0.410 -0.213 0.522 month_sep 0.5425 0.219 2.476 0.013 0.113 0.972 contact_telephone -0.7577 0.095 -8.011 0.000 -0.943 -0.572 job_admin. 0.2342 0.111 2.101 0.036 0.016 0.453 job_blue-collar -0.0569 0.120 -0.476 0.634 -0.291 0.177 job_entrepreneur 0.1570 0.173 0.907 0.365 -0.182 0.496 job_housemaid 0.2019 0.196 1.031 0.303 -0.182 0.586 job_management 0.2829 0.134 2.106 0.035 0.020 0.546 job_retired 0.3358 0.154 2.175 0.030 0.033 0.638 job_services 0.0580 0.134 0.431 0.666 -0.206 0.322 job_student 0.3253 0.157 2.069 0.039 0.017 0.634 job_technician 0.2078 0.119 1.748 0.080 -0.025 0.441 job_unknown 0.2974 0.306 0.973 0.331 -0.302 0.897 default_no 0.2926 0.081 3.619 0.000 0.134 0.451 ==

'job_unemployed' was removed.

Optimization terminated successfully. Current function value: 0.202425 Iterations 10 Logit Regression Results == Dep. Variable: deposit No. Observations: 28831 Model: Logit Df Residuals: 28798 Method: MLE Df Model: 32 Date: Thu, 11 Feb 2021 Pseudo R-squ.: 0.4210 Time: 19:14:26 Log-Likelihood: -5836.1 converged: True LL-Null: -10079. Covariance Type: nonrobust LLR p-value: 0.000

==
coef std err z P>|z| [0.025 0.975]
-- const -288.6046 46.815 -6.165 0.000 -380.359 -196.850 duration 0.0049 9.2e-05 53.610 0.000 0.005 0.005 nr_employed 0.0083 0.004 2.202 0.028 0.001 0.016 euribor3m 0.3727 0.156 2.396 0.017 0.068 0.678 emp_var_rate -2.0551 0.175 -11.726 0.000 -2.399 -1.712 cons_conf_idx 0.0257 0.009 2.748 0.006 0.007 0.044 cons_price_idx 2.5883 0.310 8.359 0.000 1.981 3.195 pdays -0.0163 0.022 -0.748 0.454 -0.059 0.026 age -0.0033 0.003 -1.215 0.224 -0.009 0.002 previous -0.0556 0.074 -0.749 0.454 -0.201 0.090 client_was_contacted_0 -1.0510 0.358 -2.933 0.003 -1.753 -0.349 poutcome_failure -0.5088 0.117 -4.364 0.000 -0.737 -0.280 poutcome_success 0.5645 0.283 1.997 0.046 0.010 1.119 month_aug 0.9450 0.147 6.437 0.000 0.657 1.233 month_dec 0.3738 0.249 1.500 0.134 -0.115 0.862 month_jul 0.1778 0.116 1.536 0.124 -0.049 0.405 month_jun -0.6105 0.153 -3.983 0.000 -0.911 -0.310 month_mar 2.1324 0.174 12.265 0.000 1.792 2.473 month_may -0.3615 0.100 -3.622 0.000 -0.557 -0.166 month_nov -0.3853 0.146 -2.634 0.008 -0.672 -0.099 month_oct 0.1541 0.188 0.822 0.411 -0.214 0.522 month_sep 0.5408 0.219 2.469 0.014 0.111 0.970 contact_telephone -0.7574 0.095 -8.010 0.000 -0.943 -0.572 job_admin. 0.2014 0.081 2.485 0.013 0.043 0.360 job_blue-collar -0.0904 0.091 -0.998 0.318 -0.268 0.087 job_entrepreneur 0.1240 0.155 0.799 0.424 -0.180 0.428 job_housemaid 0.1695 0.181 0.938 0.348 -0.185 0.524 job_management 0.2502 0.111 2.262 0.024 0.033 0.467 job_retired 0.3048 0.136 2.234 0.025 0.037 0.572 job_student 0.2923 0.137 2.131 0.033 0.023 0.561 job_technician 0.1748 0.091 1.927 0.054 -0.003 0.353 job_unknown 0.2653 0.296 0.895 0.371 -0.316 0.846 default_no 0.2924 0.081 3.616 0.000 0.134 0.451
==

'job_services' was removed.

Optimization terminated successfully. Current function value: 0.202435 Iterations 10 Logit Regression Results
== Dep. Variable: deposit No. Observations: 28831 Model: Logit Df Residuals: 28799 Method: MLE Df Model: 31 Date: Thu, 11 Feb 2021 Pseudo R-squ.: 0.4209 Time: 19:14:26 Log-Likelihood: -5836.4 converged: True LL-Null: -10079. Covariance Type: nonrobust LLR p-value: 0.000
==
coef std err z P>|z| [0.025 0.975]
-- const -287.9096 46.823 -6.149 0.000 -379.680 -196.139 duration 0.0049 9.2e-05 53.609 0.000 0.005 0.005 nr_employed 0.0084 0.004 2.211 0.027 0.001 0.016 euribor3m 0.3727 0.156 2.395 0.017 0.068 0.678 emp_var_rate -2.0540 0.175 -11.715 0.000 -2.398 -1.710 cons_conf_idx 0.0258 0.009 2.752 0.006 0.007 0.044 cons_price_idx 2.5778 0.309 8.331 0.000 1.971 3.184 pdays -0.0134 0.021 -0.626 0.532 -0.055 0.029 age -0.0033 0.003 -1.219 0.223 -0.009 0.002 client_was_contacted_0 -0.9496 0.332 -2.863 0.004 -1.600 -0.300 poutcome_failure -0.5737 0.078 -7.321 0.000 -0.727 -0.420 poutcome_success 0.5673 0.283 2.007 0.045 0.013 1.121 month_aug 0.9368 0.146 6.396 0.000 0.650 1.224 month_dec 0.3716 0.249 1.492 0.136 -0.117 0.860 month_jul 0.1764 0.116 1.525 0.127 -0.050 0.403 month_jun -0.6100 0.153 -3.979 0.000 -0.910 -0.309 month_mar 2.1288 0.174 12.249 0.000 1.788 2.469 month_may -0.3643 0.100 -3.653 0.000 -0.560 -0.169 month_nov -0.3905 0.146 -2.672 0.008 -0.677 -0.104 month_oct 0.1535 0.188 0.819 0.413 -0.214 0.521 month_sep 0.5364 0.219 2.450 0.014 0.107 0.966 contact_telephone -0.7545 0.094 -7.987 0.000 -0.940 -0.569 job_admin. 0.2013 0.081 2.485 0.013 0.043 0.360 job_blue-collar -0.0906 0.091 -1.000 0.317 -0.268 0.087 job_entrepreneur 0.1243 0.155 0.801 0.423 -0.180 0.428 job_housemaid 0.1710 0.181 0.946 0.344 -0.183 0.525 job_management 0.2490 0.111 2.250 0.024 0.032 0.466 job_retired 0.3054 0.136 2.238 0.025 0.038 0.573 job_student 0.2886 0.137 2.104 0.035 0.020 0.557 job_technician 0.1753 0.091 1.933 0.053 -0.002 0.353 job_unknown 0.2664 0.297 0.898 0.369 -0.315 0.848 default_no 0.2926 0.081 3.618 0.000 0.134 0.451
==

'previous' was removed.

Optimization terminated successfully. Current function value: 0.202441 Iterations 10 Logit Regression Results
== Dep. Variable: deposit No. Observations: 28831 Model: Logit Df Residuals: 28800 Method: MLE Df Model: 30 Date: Thu, 11 Feb 2021 Pseudo R-squ.: 0.4209 Time: 19:14:26 Log-Likelihood: -5836.6 converged: True LL-Null: -10079. Covariance Type: nonrobust LLR p-value: 0.000
==
coef std err z P>|z| [0.025 0.975]
-- const -288.6002 46.814 -6.165 0.000 -380.354 -196.847 duration 0.0049 9.2e-05 53.609 0.000 0.005 0.005 nr_employed 0.0084 0.004 2.221 0.026 0.001 0.016 euribor3m 0.3753 0.156 2.413 0.016 0.070 0.680 emp_var_rate -2.0593 0.175 -11.758 0.000 -2.403 -1.716 cons_conf_idx 0.0257 0.009 2.747 0.006 0.007 0.044 cons_price_idx 2.5816 0.309 8.343 0.000 1.975 3.188 age -0.0033 0.003 -1.217 0.224 -0.009 0.002 client_was_contacted_0 -0.8130 0.250 -3.254 0.001 -1.303 -0.323 poutcome_failure -0.5732 0.078 -7.316 0.000 -0.727 -0.420 poutcome_success 0.6306 0.264 2.387 0.017 0.113 1.148 month_aug 0.9386 0.146 6.409 0.000 0.652 1.226 month_dec 0.3703 0.249 1.486 0.137 -0.118 0.859 month_jul 0.1754 0.116 1.515 0.130

-0.051 0.402 month_jun -0.6118 0.153 -3.991 0.000 -0.912 -0.311 month_mar 2.1294 0.174 12.251 0.000 1.789 2.470 month_may -0.3636 0.100 -3.646 0.000 -0.559 -0.168 month_nov -0.3902 0.146 -2.671 0.008 -0.677 -0.104 month_oct 0.1537 0.188 0.819 0.413 -0.214 0.521 month_sep 0.5380 0.219 2.457 0.014 0.109 0.967 contact_telephone -0.7548 0.094 -7.990 0.000 -0.940 -0.570 job_admin. 0.2004 0.081 2.475 0.013 0.042 0.359 job_blue-collar -0.0911 0.091 -1.006 0.314 -0.269 0.086 job_entrepreneur 0.1232 0.155 0.795 0.427 -0.181 0.427 job_housemaid 0.1713 0.181 0.949 0.343 -0.183 0.525 job_management 0.2488 0.111 2.250 0.024 0.032 0.466 job_retired 0.3060 0.136 2.243 0.025 0.039 0.573 job_student 0.2874 0.137 2.096 0.036 0.019 0.556 job_technician 0.1744 0.091 1.922 0.055 -0.003 0.352 job_unknown 0.2689 0.296 0.908 0.364 -0.312 0.849 default_no 0.2927 0.081 3.621 0.000 0.134 0.451
==

'pdays' was removed.

Optimization terminated successfully. Current function value: 0.202452 Iterations 10 Logit Regression Results == Dep. Variable: deposit No. Observations: 28831 Model: Logit Df Residuals: 28801 Method: MLE Df Model: 29 Date: Thu, 11 Feb 2021 Pseudo R-squ.: 0.4209 Time: 19:14:27 Log-Likelihood: -5836.9 converged: True LL-Null: -10079. Covariance Type: nonrobust LLR p-value: 0.000
==
coef std err z P>|z| [0.025 0.975]
-- const -288.7404 46.811 -6.168 0.000 -380.488 -196.993 duration 0.0049 9.2e-05 53.619 0.000 0.005 0.005 nr_employed 0.0084 0.004 2.228 0.026 0.001 0.016 euribor3m 0.3745 0.155 2.409 0.016 0.070 0.679 emp_var_rate -2.0594 0.175 -11.758 0.000 -2.403 -1.716 cons_conf_idx 0.0257 0.009 2.745 0.006 0.007 0.044 cons_price_idx 2.5819 0.309 8.345 0.000 1.975 3.188 age -0.0031 0.003 -1.169 0.242 -0.008 0.002 client_was_contacted_0 -0.8149 0.250 -3.261 0.001 -1.305 -0.325 poutcome_failure -0.5742 0.078 -7.330 0.000 -0.728 -0.421 poutcome_success 0.6284 0.264 2.379 0.017 0.111 1.146 month_aug 0.9375 0.146 6.402 0.000 0.650 1.225 month_dec 0.3684 0.249 1.478 0.139 -0.120 0.857 month_jul 0.1743 0.116 1.506 0.132 -0.052 0.401 month_jun -0.6123 0.153 -3.995 0.000 -0.913 -0.312 month_mar 2.1274 0.174 12.241 0.000 1.787 2.468 month_may -0.3645 0.100 -3.656 0.000 -0.560 -0.169 month_nov -0.3885 0.146 -2.660 0.008 -0.675 -0.102 month_oct 0.1532 0.188 0.817 0.414 -0.214 0.521 month_sep 0.5389 0.219 2.461 0.014 0.110 0.968 contact_telephone -0.7545 0.094 -7.988 0.000 -0.940 -0.569 job_admin. 0.1788 0.076 2.352 0.019 0.030 0.328 job_blue-collar -0.1134 0.086 -1.319 0.187 -0.282 0.055 job_housemaid 0.1482 0.178 0.832 0.405 -0.201 0.497 job_management 0.2263 0.107 2.120 0.034 0.017 0.436 job_retired 0.2811 0.133 2.120 0.034 0.021 0.541 job_student 0.2681 0.135 1.989 0.047 0.004 0.532 job_technician 0.1527 0.086 1.770 0.077 -0.016 0.322 job_unknown 0.2464 0.295 0.836 0.403 -0.331 0.824 default_no 0.2930 0.081 3.625 0.000 0.135 0.451
==

'job_entrepreneur' was removed.

Optimization terminated successfully. Current function value: 0.202464 Iterations 10 Logit Regression Results == Dep. Variable: deposit No. Observations: 28831 Model: Logit Df Residuals: 28802 Method: MLE Df Model: 28 Date: Thu, 11 Feb 2021 Pseudo R-squ.: 0.4209 Time: 19:14:27 Log-Likelihood: -5837.2 converged: True LL-Null: -10079. Covariance Type: nonrobust LLR p-value: 0.000
==
coef std err z P>|z| [0.025 0.975]
-- const -288.0357 46.805 -6.154 0.000 -379.771 -196.300 duration 0.0049 9.2e-05 53.620 0.000 0.005 0.005 nr_employed 0.0083 0.004 2.211 0.027 0.001 0.016 euribor3m 0.3765 0.155 2.422 0.015 0.072 0.681 emp_var_rate -2.0577 0.175 -11.748 0.000 -2.401 -1.714 cons_conf_idx 0.0258 0.009 2.753 0.006 0.007 0.044 cons_price_idx 2.5781 0.309 8.333 0.000 1.972 3.185 age -0.0028 0.003 -1.063 0.288 -0.008 0.002 client_was_contacted_0 -0.8158 0.250 -3.265 0.001 -1.306 -0.326 poutcome_failure -0.5747 0.078 -7.336 0.000 -0.728 -0.421 poutcome_success 0.6270 0.264 2.373 0.018 0.109 1.145 month_aug 0.9368 0.146 6.396 0.000 0.650 1.224 month_dec 0.3664 0.249 1.471 0.141 -0.122 0.855 month_jul 0.1752 0.116 1.515 0.130 -0.052 0.402 month_jun -0.6101 0.153 -3.981 0.000 -0.911 -0.310 month_mar 2.1256 0.174 12.231 0.000 1.785 2.466 month_may -0.3654 0.100 -3.665 0.000 -0.561 -0.170 month_nov -0.3904 0.146 -2.673 0.008 -0.677 -0.104 month_oct 0.1499 0.187 0.800 0.424 -0.218 0.517 month_sep 0.5347 0.219 2.443 0.015 0.106 0.964 contact_telephone -0.7550 0.094 -7.994 0.000 -0.940 -0.570 job_admin. 0.1627 0.073 2.218 0.027 0.019 0.306 job_blue-collar -0.1295 0.084 -1.549 0.121 -0.293 0.034 job_management 0.2090 0.105 1.998 0.046 0.004 0.414 job_retired 0.2555 0.129 1.982 0.047 0.003 0.508 job_student 0.2557 0.134 1.910 0.056 -0.007 0.518 job_technician 0.1365 0.084 1.627 0.104 -0.028 0.301 job_unknown 0.2266 0.294 0.772 0.440 -0.349 0.802 default_no 0.2923 0.081 3.616 0.000 0.134 0.451
==

'job_housemaid' was removed.

Optimization terminated successfully. Current function value: 0.202474 Iterations 10 Logit Regression Results

== Dep. Variable: deposit No. Observations: 28831 Model: Logit Df Residuals: 28803 Method: MLE Df Model: 27 Date: Thu, 11 Feb 2021 Pseudo R-squ.: 0.4208 Time: 19:14:27 Log-Likelihood: -5837.5 converged: True LL-Null: -10079. Covariance Type: nonrobust LLR p-value: 0.000
==
coef std err z P>|z| [0.025 0.975]
-- const -288.0765 46.809 -6.154 0.000 -379.820 -196.333 duration 0.0049 9.2e-05 53.623 0.000 0.005 0.005 nr_employed 0.0083 0.004 2.206 0.027 0.001 0.016 euribor3m 0.3796 0.155 2.442 0.015 0.075 0.684 emp_var_rate -2.0605 0.175 -11.765 0.000 -2.404 -1.717 cons_conf_idx 0.0258 0.009 2.754 0.006 0.007 0.044 cons_price_idx 2.5795 0.309 8.337 0.000 1.973 3.186 age -0.0027 0.003 -1.020 0.308 -0.008 0.002 client_was_contacted_0 -0.8171 0.250 -3.270 0.001 -1.307 -0.327 poutcome_failure -0.5740 0.078 -7.330 0.000 -0.728 -0.421 poutcome_success 0.6264 0.264 2.371 0.018 0.109 1.144 month_aug 0.9382 0.146 6.406 0.000 0.651 1.225 month_dec 0.3658 0.249 1.468 0.142 -0.123 0.854 month_jul 0.1763 0.116 1.524 0.128 -0.050 0.403 month_jun -0.6092 0.153 -3.975 0.000 -0.910 -0.309 month_mar 2.1267 0.174 12.237 0.000 1.786 2.467 month_may -0.3638 0.100 -3.650 0.000 -0.559 -0.168 month_nov -0.3922 0.146 -2.686 0.007 -0.678 -0.106 month_oct 0.1498 0.187 0.799 0.424 -0.218 0.517 month_sep 0.5373 0.219 2.455 0.014 0.108 0.966 contact_telephone -0.7564 0.094 -8.010 0.000 -0.941 -0.571 job_admin. 0.1550 0.073 2.134 0.033 0.013 0.297 job_blue-collar -0.1371 0.083 -1.653 0.098 -0.300 0.025 job_management 0.2008 0.104 1.931 0.054 -0.003 0.405 job_retired 0.2437 0.128 1.904 0.057 -0.007 0.495 job_student 0.2491 0.134 1.865 0.062 -0.013 0.511 job_technician 0.1288 0.083 1.547 0.122 -0.034 0.292 default_no 0.2908 0.081 3.599 0.000 0.132 0.449
==

'job_unknown' was removed.

Optimization terminated successfully. Current function value: 0.202485 Iterations 9 Logit Regression Results
== Dep. Variable: deposit No. Observations: 28831 Model: Logit Df Residuals: 28804 Method: MLE Df Model: 26 Date: Thu, 11 Feb 2021 Pseudo R-squ.: 0.4208 Time: 19:14:27 Log-Likelihood: -5837.8 converged: True LL-Null: -10079. Covariance Type: nonrobust LLR p-value: 0.000
==
coef std err z P>|z| [0.025 0.975]
-- const -269.0830 40.395 -6.661 0.000 -348.256 -189.910 duration 0.0049 9.2e-05 53.624 0.000 0.005 0.005 nr_employed 0.0066 0.003 2.144 0.032 0.001 0.013 euribor3m 0.4501 0.128 3.516 0.000 0.199 0.701 emp_var_rate -2.0317 0.172 -11.831 0.000 -2.368 -1.695 cons_conf_idx 0.0239 0.009 2.636 0.008 0.006 0.042 cons_price_idx 2.4717 0.279 8.861 0.000 1.925 3.018 age -0.0027 0.003 -1.019 0.308 -0.008 0.002 client_was_contacted_0 -0.8221 0.250 -3.288 0.001 -1.312 -0.332 poutcome_failure -0.5728 0.078 -7.315 0.000 -0.726 -0.419 poutcome_success 0.6229 0.264 2.356 0.018 0.105 1.141 month_aug 0.8748 0.123 7.105 0.000 0.633 1.116 month_dec 0.2819 0.226 1.247 0.212 -0.161 0.725 month_jul 0.1417 0.107 1.323 0.186 -0.068 0.352 month_jun -0.6100 0.153 -3.982 0.000 -0.910 -0.310 month_mar 2.0572 0.150 13.676 0.000 1.762 2.352 month_may -0.4043 0.086 -4.720 0.000 -0.572 -0.236 month_nov -0.4634 0.116 -4.010 0.000 -0.690 -0.237 month_sep 0.4258 0.169 2.524 0.012 0.095 0.756 contact_telephone -0.7546 0.094 -7.995 0.000 -0.940 -0.570 job_admin. 0.1544 0.073 2.126 0.033 0.012 0.297 job_blue-collar -0.1361 0.083 -1.642 0.101 -0.299 0.026 job_management 0.2007 0.104 1.931 0.053 -0.003 0.404 job_retired 0.2465 0.128 1.927 0.054 -0.004 0.497 job_student 0.2495 0.134 1.868 0.062 -0.012 0.511 job_technician 0.1287 0.083 1.545 0.122 -0.035 0.292 default_no 0.2923 0.081 3.619 0.000 0.134 0.451
==

'month_oct' was removed.

Optimization terminated successfully. Current function value: 0.202503 Iterations 9 Logit Regression Results
== Dep. Variable: deposit No. Observations: 28831 Model: Logit Df Residuals: 28805 Method: MLE Df Model: 25 Date: Thu, 11 Feb 2021 Pseudo R-squ.: 0.4207 Time: 19:14:28 Log-Likelihood: -5838.4 converged: True LL-Null: -10079. Covariance Type: nonrobust LLR p-value: 0.000
==
coef std err z P>|z| [0.025 0.975]
-- const -267.0590 40.346 -6.619 0.000 -346.136 -187.982 duration 0.0049 9.2e-05 53.622 0.000 0.005 0.005 nr_employed 0.0064 0.003 2.102 0.036 0.000 0.012 euribor3m 0.4526 0.128 3.536 0.000 0.202 0.703 emp_var_rate -2.0245 0.172 -11.798 0.000 -2.361 -1.688 cons_conf_idx 0.0232 0.009 2.563 0.010 0.005 0.041 cons_price_idx 2.4560 0.279 8.818 0.000 1.910 3.002 client_was_contacted_0 -0.8234 0.250 -3.295 0.001 -1.313 -0.334 poutcome_failure -0.5735 0.078 -7.323 0.000 -0.727 -0.420 poutcome_success 0.6198 0.264 2.346 0.019 0.102 1.138 month_aug 0.8735 0.123 7.095 0.000 0.632 1.115 month_dec 0.2717 0.226 1.204 0.229 -0.171 0.714 month_jul 0.1488 0.107 1.392 0.164 -0.061 0.358 month_jun -0.5977 0.153 -3.914 0.000 -0.897 -0.298 month_mar 2.0562 0.150 13.668 0.000 1.761 2.351 month_may -0.3999 0.086 -4.674

0.000 -0.568 -0.232 month_nov -0.4630 0.116 -4.007 0.000 -0.689 -0.237 month_sep 0.4236 0.169 2.512 0.012 0.093 0.754 contact_telephone -0.7529 0.094 -7.980 0.000 -0.938 -0.568 job_admin. 0.1608 0.072 2.224 0.026 0.019 0.303 job_blue-collar -0.1336 0.083 -1.611 0.107 -0.296 0.029 job_management 0.1951 0.104 1.879 0.060 -0.008 0.399 job_retired 0.1786 0.109 1.634 0.102 -0.036 0.393 job_student 0.2916 0.127 2.295 0.022 0.043 0.541 job_technician 0.1344 0.083 1.619 0.106 -0.028 0.297 default_no 0.3044 0.080 3.809 0.000 0.148 0.461
==

'age' was removed.

Optimization terminated successfully. Current function value: 0.202528 Iterations 9 Logit Regression Results == Dep. Variable: deposit No. Observations: 28831 Model: Logit Df Residuals: 28806 Method: MLE Df Model: 24 Date: Thu, 11 Feb 2021 Pseudo R-squ.: 0.4207 Time: 19:14:28 Log-Likelihood: -5839.1 converged: True LL-Null: -10079. Covariance Type: nonrobust LLR p-value: 0.000
==
coef std err z P>|z| [0.025 0.975]
-- const -260.2541 39.921 -6.519 0.000 -338.498 -182.010 duration 0.0049 9.2e-05 53.636 0.000 0.005 0.005 nr_employed 0.0059 0.003 1.951 0.051 -2.59e-05 0.012 euribor3m 0.4665 0.127 3.661 0.000 0.217 0.716 emp_var_rate -2.0062 0.171 -11.748 0.000 -2.341 -1.672 cons_conf_idx 0.0233 0.009 2.571 0.010 0.006 0.041 cons_price_idx 2.4116 0.276 8.742 0.000 1.871 2.952 client_was_contacted_0 -0.8184 0.250 -3.280 0.001 -1.308 -0.329 poutcome_failure -0.5714 0.078 -7.300 0.000 -0.725 -0.418 poutcome_success 0.6282 0.264 2.381 0.017 0.111 1.145 month_aug 0.8399 0.120 7.012 0.000 0.605 1.075 month_jul 0.1340 0.106 1.263 0.207 -0.074 0.342 month_jun -0.5984 0.153 -3.922 0.000 -0.897 -0.299 month_mar 2.0274 0.148 13.654 0.000 1.736 2.318 month_may -0.4195 0.084 -5.000 0.000 -0.584 -0.255 month_nov -0.4886 0.114 -4.304 0.000 -0.711 -0.266 month_sep 0.3802 0.165 2.309 0.021 0.058 0.703 contact_telephone -0.7508 0.094 -7.963 0.000 -0.936 -0.566 job_admin. 0.1622 0.072 2.243 0.025 0.020 0.304 job_blue-collar -0.1325 0.083 -1.598 0.110 -0.295 0.030 job_management 0.1943 0.104 1.871 0.061 -0.009 0.398 job_retired 0.1817 0.109 1.664 0.096 -0.032 0.396 job_student 0.2995 0.127 2.362 0.018 0.051 0.548 job_technician 0.1351 0.083 1.628 0.104 -0.028 0.298 default_no 0.3042 0.080 3.807 0.000 0.148 0.461
==

'month_dec' was removed.

Optimization terminated successfully. Current function value: 0.202556 Iterations 9 Logit Regression Results == Dep. Variable: deposit No. Observations: 28831 Model: Logit Df Residuals: 28807 Method: MLE Df Model: 23 Date: Thu, 11 Feb 2021 Pseudo R-squ.: 0.4206 Time: 19:14:28 Log-Likelihood: -5839.9 converged: True LL-Null: -10079. Covariance Type: nonrobust LLR p-value: 0.000
==
coef std err z P>|z| [0.025 0.975]
-- const -275.6509 38.033 -7.248 0.000 -350.195 -201.107 duration 0.0049 9.2e-05 53.635 0.000 0.005 0.005 nr_employed 0.0070 0.003 2.410 0.016 0.001 0.013 euribor3m 0.4545 0.127 3.583 0.000 0.206 0.703 emp_var_rate -2.0528 0.167 -12.312 0.000 -2.380 -1.726 cons_conf_idx 0.0266 0.009 3.072 0.002 0.010 0.044 cons_price_idx 2.5191 0.263 9.592 0.000 2.004 3.034 client_was_contacted_0 -0.8215 0.249 -3.293 0.001 -1.310 -0.333 poutcome_failure -0.5733 0.078 -7.326 0.000 -0.727 -0.420 poutcome_success 0.6267 0.264 2.376 0.017 0.110 1.144 month_aug 0.7909 0.113 6.975 0.000 0.569 1.013 month_jun -0.6886 0.135 -5.114 0.000 -0.952 -0.425 month_mar 2.0171 0.148 13.592 0.000 1.726 2.308 month_may -0.4565 0.079 -5.811 0.000 -0.610 -0.302 month_nov -0.5430 0.105 -5.180 0.000 -0.748 -0.338 month_sep 0.3635 0.164 2.215 0.027 0.042 0.685 contact_telephone -0.7945 0.088 -8.989 0.000 -0.968 -0.621 job_admin. 0.1635 0.072 2.262 0.024 0.022 0.305 job_blue-collar -0.1327 0.083 -1.601 0.109 -0.295 0.030 job_management 0.1917 0.104 1.847 0.065 -0.012 0.395 job_retired 0.1815 0.109 1.663 0.096 -0.032 0.395 job_student 0.3032 0.127 2.391 0.017 0.055 0.552 job_technician 0.1334 0.083 1.607 0.108 -0.029 0.296 default_no 0.3079 0.080 3.855 0.000 0.151 0.464
==

'month_jul' was removed.

Optimization terminated successfully. Current function value: 0.202600 Iterations 9 Logit Regression Results == Dep. Variable: deposit No. Observations: 28831 Model: Logit Df Residuals: 28808 Method: MLE Df Model: 22 Date: Thu, 11 Feb 2021 Pseudo R-squ.: 0.4205 Time: 19:14:28 Log-Likelihood: -5841.2 converged: True LL-Null: -10079. Covariance Type: nonrobust LLR p-value: 0.000
==
coef std err z P>|z| [0.025 0.975]
-- const -275.8813 38.042 -7.252

0.000 -350.442 -201.320 duration 0.0049 9.19e-05 53.637 0.000 0.005 0.005 nr_employed 0.0069 0.003 2.389 0.017 0.001 0.013 euribor3m 0.4625 0.127 3.649 0.000 0.214 0.711 emp_var_rate -2.0604 0.167 -12.359 0.000 -2.387 -1.734 cons_conf_idx 0.0266 0.009 3.078 0.002 0.010 0.044 cons_price_idx 2.5239 0.263 9.608 0.000 2.009 3.039 client_was_contacted_0 -0.8156 0.249 -3.269 0.001 -1.304 -0.327 poutcome_failure -0.5726 0.078 -7.317 0.000 -0.726 -0.419 poutcome_success 0.6325 0.264 2.399 0.016 0.116 1.149 month_aug 0.7967 0.113 7.028 0.000 0.575 1.019 month_jun -0.6899 0.135 -5.122 0.000 -0.954 -0.426 month_mar 2.0192 0.148 13.608 0.000 1.728 2.310 month_may -0.4618 0.078 -5.884 0.000 -0.616 -0.308 month_nov -0.5433 0.105 -5.182 0.000 -0.749 -0.338 month_sep 0.3681 0.164 2.243 0.025 0.046 0.690 contact_telephone -0.7962 0.088 -9.004 0.000 -0.969 -0.623 job_admin. 0.2216 0.063 3.520 0.000 0.098 0.345 job_management 0.2502 0.097 2.567 0.010 0.059 0.441 job_retired 0.2369 0.104 2.282 0.022 0.033 0.440 job_student 0.3595 0.122 2.946 0.003 0.120 0.599 job_technician 0.1920 0.075 2.565 0.010 0.045 0.339 default_no 0.3176 0.080 3.989 0.000 0.162 0.474
==

'job_blue-collar' was removed

Optimization terminated successfully. Current function value: 0.202700 Iterations 8 Logit Regression Results == Dep. Variable: deposit No. Observations: 28831 Model: Logit Df Residuals: 28809 Method: MLE Df Model: 21 Date: Thu, 11 Feb 2021 Pseudo R-squ.: 0.4202 Time: 19:14:29 Log-Likelihood: -5844.0 converged: True LL-Null: -10079. Covariance Type: nonrobust LLR p-value: 0.000
==
coef std err z P>|z| [0.025 0.975]
-- const -189.7745 11.935 -15.901 0.000 -213.166 -166.383 duration 0.0049 9.19e-05 53.658 0.000 0.005 0.005 euribor3m 0.6488 0.100 6.477 0.000 0.452 0.845 emp_var_rate -1.8221 0.133 -13.723 0.000 -2.082 -1.562 cons_conf_idx 0.0122 0.006 1.968 0.049 4.72e-05 0.024 cons_price_idx 1.9719 0.123 15.986 0.000 1.730 2.214 client_was_contacted_0 -0.8097 0.249 -3.257 0.001 -1.297 -0.322 poutcome_failure -0.5672 0.078 -7.257 0.000 -0.720 -0.414 poutcome_success 0.6342 0.263 2.413 0.016 0.119 1.149 month_aug 0.7146 0.108 6.640 0.000 0.504 0.925 month_jun -0.4810 0.102 -4.701 0.000 -0.681 -0.280 month_mar 1.8548 0.131 14.179 0.000 1.598 2.111 month_may -0.5072 0.076 -6.679 0.000 -0.656 -0.358 month_nov -0.6121 0.101 -6.071 0.000 -0.810 -0.414 month_sep 0.1637 0.140 1.171 0.242 -0.110 0.438 contact_telephone -0.7449 0.085 -8.741 0.000 -0.912 -0.578 job_admin. 0.2191 0.063 3.481 0.000 0.096 0.342 job_management 0.2441 0.097 2.505 0.012 0.053 0.435 job_retired 0.2339 0.104 2.256 0.024 0.031 0.437 job_student 0.3540 0.122 2.901 0.004 0.115 0.593 job_technician 0.1806 0.075 2.416 0.016 0.034 0.327 default_no 0.3148 0.080 3.952 0.000 0.159 0.471
==

'nr_employed' was dropped.

Optimization terminated successfully. Current function value: 0.203417 Iterations 8 Logit Regression Results == Dep. Variable: deposit No. Observations: 28831 Model: Logit Df Residuals: 28810 Method: MLE Df Model: 20 Date: Thu, 11 Feb 2021 Pseudo R-squ.: 0.4181 Time: 19:14:58 Log-Likelihood: -5864.7 converged: True LL-Null: -10079. Covariance Type: nonrobust LLR p-value: 0.000
==
coef std err z P>|z| [0.025 0.975]
-- const -124.8097 6.365 -19.610 0.000 -137.284 -112.335 duration 0.0049 9.18e-05 53.563 0.000 0.005 0.005 emp_var_rate -0.9822 0.028 -35.296 0.000 -1.037 -0.928 cons_conf_idx 0.0329 0.005 6.186 0.000 0.023 0.043 cons_price_idx 1.3111 0.068 19.212 0.000 1.177 1.445 client_was_contacted_0 -0.8277 0.247 -3.345 0.001 -1.313 -0.343 poutcome_failure -0.6032 0.078 -7.762 0.000 -0.756 -0.451 poutcome_success 0.5724 0.261 2.189 0.029 0.060 1.085 month_aug 0.3666 0.092 3.971 0.000 0.186 0.548 month_jun -0.1439 0.088 -1.629 0.103 -0.317 0.029 month_mar 1.6006 0.128 12.513 0.000 1.350 1.851 month_may -0.6678 0.071 -9.348 0.000 -0.808 -0.528 month_nov -0.3568 0.093 -3.847 0.000 -0.539 -0.175 month_sep -0.0188 0.137 -0.137 0.891 -0.287 0.250 contact_telephone -0.6077 0.081 -7.472 0.000 -0.767 -0.448 job_admin. 0.2236 0.063 3.560 0.000 0.100 0.347 job_management 0.2564 0.097 2.637 0.008 0.066 0.447 job_retired 0.2506 0.103 2.423 0.015 0.048 0.453 job_student 0.3464 0.122 2.842 0.004 0.107 0.585 job_technician 0.1806 0.075 2.420 0.016 0.034 0.327 default_no 0.3188 0.080 4.001 0.000 0.163 0.475
==

'euribor3m' was dropped.

Optimization terminated successfully. Current function value: 0.203606 Iterations 8 Logit Regression Results == Dep. Variable: deposit No. Observations: 28831 Model: Logit Df Residuals: 28811 Method: MLE Df Model: 19 Date: Thu, 11 Feb 2021 Pseudo R-squ.: 0.4176 Time: 19:15:34 Log-Likelihood: -5870.2 converged: True LL-Null: -10079. Covariance Type: nonrobust LLR p-value: 0.000

== coef std err z P>|z| [0.025 0.975] -- const -127.1946 6.316 -20.139 0.000 -139.573 -114.816 duration 0.0049 9.18e-05 53.599 0.000 0.005 0.005 emp_var_rate -0.9872 0.028 -35.562 0.000 -1.042 -0.933 cons_conf_idx 0.0338 0.005 6.355 0.000 0.023 0.044 cons_price_idx 1.3281 0.068 19.544 0.000 1.195 1.461 poutcome_failure -0.5484 0.075 -7.290 0.000 -0.696 -0.401 poutcome_success 1.3905 0.093 14.987 0.000 1.209 1.572 month_aug 0.3767 0.092 4.085 0.000 0.196 0.557 month_jun -0.1470 0.088 -1.667 0.096 -0.320 0.026 month_mar 1.6157 0.128 12.647 0.000 1.365 1.866 month_may -0.6676 0.071 -9.349 0.000 -0.808 -0.528 month_nov -0.3500 0.093 -3.779 0.000 -0.532 -0.169 month_sep -0.0191 0.137 -0.139 0.889 -0.287 0.249 contact_telephone -0.6118 0.081 -7.526 0.000 -0.771 -0.452 job_admin. 0.2247 0.063 3.580 0.000 0.102 0.348 job_management 0.2594 0.097 2.671 0.008 0.069 0.450 job_retired 0.2561 0.103 2.480 0.013 0.054 0.459 job_student 0.3596 0.122 2.955 0.003 0.121 0.598 job_technician 0.1831 0.075 2.455 0.014 0.037 0.329 default_no 0.3172 0.080 3.982 0.000 0.161 0.473 ==

'client_was_contacted_0' was dropped.

Optimization terminated successfully. Current function value: 0.203607 Iterations 8 Logit Regression Results == Dep. Variable: deposit No. Observations: 28831 Model: Logit Df Residuals: 28812 Method: MLE Df Model: 18 Date: Thu, 11 Feb 2021 Pseudo R-squ.: 0.4176 Time: 19:16:04 Log-Likelihood: -5870.2 converged: True LL-Null: -10079. Covariance Type: nonrobust LLR p-value: 0.000 == coef std err z P>|z| [0.025 0.975] -- const -127.0892 6.270 -20.269 0.000 -139.378 -114.800 duration 0.0049 9.18e-05 53.604 0.000 0.005 0.005 emp_var_rate -0.9868 0.028 -35.738 0.000 -1.041 -0.933 cons_conf_idx 0.0336 0.005 6.546 0.000 0.024 0.044 cons_price_idx 1.3269 0.067 19.694 0.000 1.195 1.459 poutcome_failure -0.5486 0.075 -7.294 0.000 -0.696 -0.401 poutcome_success 1.3900 0.093 14.993 0.000 1.208 1.572 month_aug 0.3793 0.090 4.207 0.000 0.203 0.556 month_jun -0.1450 0.087 -1.667 0.096 -0.316 0.026 month_mar 1.6172 0.127 12.705 0.000 1.368 1.867 month_may -0.6664 0.071 -9.403 0.000 -0.805 -0.527 month_nov -0.3479 0.091 -3.808 0.000 -0.527 -0.169 contact_telephone -0.6111 0.081 -7.531 0.000 -0.770 -0.452 job_admin. 0.2246 0.063 3.578 0.000 0.102 0.348 job_management 0.2593 0.097 2.670 0.008 0.069 0.450 job_retired 0.2560 0.103 2.479 0.013 0.054 0.458 job_student 0.3595 0.122 2.954 0.003 0.121 0.598 job_technician 0.1829 0.075 2.453 0.014 0.037 0.329 default_no 0.3171 0.080 3.981 0.000 0.161 0.473 ==

'month_sep' was dropped.

Optimization terminated successfully. Current function value: 0.226093 Iterations 8 Logit Regression Results == Dep. Variable: deposit No. Observations: 28831 Model: Logit Df Residuals: 28813 Method: MLE Df Model: 17 Date: Thu, 11 Feb 2021 Pseudo R-squ.: 0.3533 Time: 19:16:36 Log-Likelihood: -6518.5 converged: True LL-Null: -10079. Covariance Type: nonrobust LLR p-value: 0.000 == coef std err z P>|z| [0.025 0.975] -- const 56.8294 4.565 12.449 0.000 47.882 65.777 duration 0.0045 8.43e-05 53.537 0.000 0.004 0.005 cons_conf_idx 0.0526 0.005 9.778 0.000 0.042 0.063 cons_price_idx -0.6291 0.050 -12.705 0.000 -0.726 -0.532 poutcome_failure 0.1966 0.071 2.755 0.006 0.057 0.337 poutcome_success 2.7501 0.087 31.774 0.000 2.580 2.920 month_aug -0.9828 0.080 -12.282 0.000 -1.140 -0.826 month_jun 0.5965 0.086 6.901 0.000 0.427 0.766 month_mar 1.7394 0.127 13.700 0.000 1.491 1.988 month_may -0.7230 0.071 -10.118 0.000 -0.863 -0.583 month_nov -0.8658 0.086 -10.027 0.000 -1.035 -0.697 contact_telephone -0.8971 0.084 -10.658 0.000 -1.062 -0.732 job_admin. 0.3156 0.060 5.267 0.000 0.198 0.433 job_management 0.3004 0.093 3.229 0.001 0.118 0.483 job_retired 0.7690 0.099 7.802 0.000 0.576 0.962 job_student 1.0178 0.118 8.612 0.000 0.786 1.249 job_technician 0.1971 0.071 2.775 0.006 0.058 0.336 default_no 0.6742 0.076 8.878 0.000 0.525 0.823 ==

'emp_var_rate' was dropped.

--

As shown above, this process was used to drop unneccesary variables until all variables had a p-value < 0.05 and all variance inflation factors were < 5 with an overall C-Statistic greater than 0.7 as originally desired.

These unneccesary variables included:

- job_unemployed
- job_services
- previous
- pdays
- job_entrepreneur
- job_housemaid
- job_unknown
- month_oct
- age
- month_dec
- month_jul
- job_blue_collar

Some explanitory variables were also removed due to multi-collinearity:

- nr-employed
- euribor3m
- client_was_contacted_0
- month_sep

--

Now let us take a look at the final dataset that we will use for the construction of our models.

--

In [13]:
```
columns_to_drop = ['client_was_contacted_1', 'poutcome_nonexistent', 'month_apr', 'contact
logistic_model = sm.Logit(y_train, sm.add_constant(x_train.drop(columns_to_drop, axis=1)))
print(logistic_model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.226093
         Iterations 8
                           Logit Regression Results
==============================================================================
Dep. Variable:                deposit   No. Observations:                28831
Model:                          Logit   Df Residuals:                    28813
Method:                           MLE   Df Model:                           17
Date:                Sat, 13 Feb 2021   Pseudo R-squ.:                  0.3533
Time:                        18:05:54   Log-Likelihood:                -6518.5
converged:                       True   LL-Null:                       -10079.
Covariance Type:            nonrobust   LLR p-value:                     0.000
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                56.8294      4.565     12.449      0.000      47.882      65.777
duration              0.0045   8.43e-05     53.537      0.000       0.004       0.005
cons_conf_idx         0.0526      0.005      9.778      0.000       0.042       0.063
cons_price_idx       -0.6291      0.050    -12.705      0.000      -0.726      -0.532
poutcome_failure      0.1966      0.071      2.755      0.006       0.057       0.337
poutcome_success      2.7501      0.087     31.774      0.000       2.580       2.920
month_aug            -0.9828      0.080    -12.282      0.000      -1.140      -0.826
month_jun             0.5965      0.086      6.901      0.000       0.427       0.766
month_mar             1.7394      0.127     13.700      0.000       1.491       1.988
month_may            -0.7230      0.071    -10.118      0.000      -0.863      -0.583
month_nov            -0.8658      0.086    -10.027      0.000      -1.035      -0.697
contact_telephone    -0.8971      0.084    -10.658      0.000      -1.062      -0.732
job_admin.            0.3156      0.060      5.267      0.000       0.198       0.433
job_management        0.3004      0.093      3.229      0.001       0.118       0.483
job_retired           0.7690      0.099      7.802      0.000       0.576       0.962
job_student           1.0178      0.118      8.612      0.000       0.786       1.249
job_technician        0.1971      0.071      2.775      0.006       0.058       0.336
default_no            0.6742      0.076      8.878      0.000       0.525       0.823
=====================================================================================
```

--

As mentioned, the above table shows us all of our remaining variables in our final dataset that will be used to predict whether or not a bank client will subscribe to a term deposit.

Before we build a model, we should check the Variance Inflation Factors once more to ensure that they are all still < 5 after dropping the unnecessary columns.

--

In [14]:
```
variance_inflation_list = []
variables = pd.Series(x_train.drop(columns_to_drop, axis=1).columns)
completed_cols = []
variance_inflation_factors = []

for variable in variables:
    completed_cols.append(variable)
    dependent_variable = variable
    independent_variables = variables[~variables.isin(completed_cols)]
    mod = sm.OLS(x_train.drop(columns_to_drop, axis=1)[dependent_variable], sm.add_constar
    residuals = mod.fit()
    variance_inflation_factor = 1 / (1 - residuals.rsquared)
    variance_inflation_factors.append({'dependent_variable': dependent_variable, 'variance

print(pd.DataFrame(variance_inflation_factors))
```

```
  dependent_variable  variance_inflation_factor
0           duration                   1.008880
1      cons_conf_idx                   1.744903
2     cons_price_idx                   2.088726
3   poutcome_failure                   1.110921
```

```
4    poutcome_success                   1.032184
5           month_aug                   1.360583
6           month_jun                   1.571653
7           month_mar                   1.018829
8           month_may                   1.190489
9           month_nov                   1.047445
10  contact_telephone                   1.029174
11         job_admin.                   1.201458
12     job_management                   1.025359
13        job_retired                   1.009973
14        job_student                   1.005703
15     job_technician                   1.004838
16         default_no                   1.000000
```

--

As we can see, all of the remaining variables in our final datset still have variance inflation factors that are valued at less than 5.

Now that we have the final dataset, let's evaluate our dataset and verify the data in the dataset before we begin constructing our first classification model; Logistic Regression.

--

Evaluation of Data

Here is a description of the meaning and type of data for each attribute in our final dataset:

- **duration:** Time elapsed for last client contact in seconds *(numeric)*.
- **cons_conf_idx:** Monthly indicator for consumer confidence index *(numeric)*.
- **cons_price_idx:** Monthly indicator for consumer price index *(numeric)*.
- **poutcome_failure:** Previous marketing campaign was unsuccessful based on campaign results as defined by objective *(categorical)*.
- **poutcome_success:** Previous marketing campaign was successful based on campaign results as defined by objective *(categorical)*.
- **month_aug:** Last contact for client was during the month of August during specified year *(categorical)*.
- **month_jun:** Last contact for client was during the month of June during specified year *(categorical)*.
- **month_mar:** Last contact for client was during the month of March during specified year *(categorical)*.
- **month_may:** Last contact for client was during the month of May during specified year *(categorical)*.
- **month_nov:** Last contact for client was during the month of November during specified year *(categorical)*.
- **contact_telephone:** Method of contacting specified client was by telephone *(categorical)*.
- **job_admin.:** Client career position was administrative *(categorical)*.
- **job_management:** Client career position was in management *(categorical)*.
- **job_retired:** Client was retired during collection of data *(catergorical)*.
- **job_student:** Client was a student during collection of data *(categorical)*.
- **job_technician:** Client career position was as a technician *(categorical)*.
- **default_no:** Client did not have any credit in default during collection of data *(categorical)*.

Data Verification

In order to effectively verify the quality of our data, we will need to examine the dataset for missing values, duplicates, and outliers. If any are found, we will need to come to a concensus on whether those discrepancies are errors or part of a quality dataset.

We will also briefly analyze some statistics regarding our dataset. This will help us to better understand the correlations between our predictive variables.

--

First let us check to see if there is any missing data in our dataset. We will use the original dataset from our bank_model_df dataframe.

In [15]:
```
bank_model_df.isnull()
```

Out[15]:

	duration	nr_employed	euribor3m	emp_var_rate	cons_conf_idx	cons_price_idx	deposit	pdays	age
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
41183	False	False	False	False	False	False	False	False	False
41184	False	False	False	False	False	False	False	False	False
41185	False	False	False	False	False	False	False	False	False
41186	False	False	False	False	False	False	False	False	False
41187	False	False	False	False	False	False	False	False	False

41188 rows × 42 columns

--

The .isnull() function allows us to see if our dataset has any missing values. Since the boolean value False is returned for each column of each row, we can see that our dataset has no missing values. The .get_dummies function generated some duplicate variables, however they have already been previously dropped when eliminating unnecessary variables.

--

Next let us create some boxplots in order to visualize any outliers in our dataset. We will run our target variable (deposit) against our numeric data types, each contained within their own boxplot. This will give us our outliers for analysis.

--

First let us create a boxplot featuring **deposit** against **duration**.

In [16]:
```
sns.boxplot(x = 'deposit', y = 'duration', data = bank_model_df)
```

Out[16]:
```
<AxesSubplot:xlabel='deposit', ylabel='duration'>
```

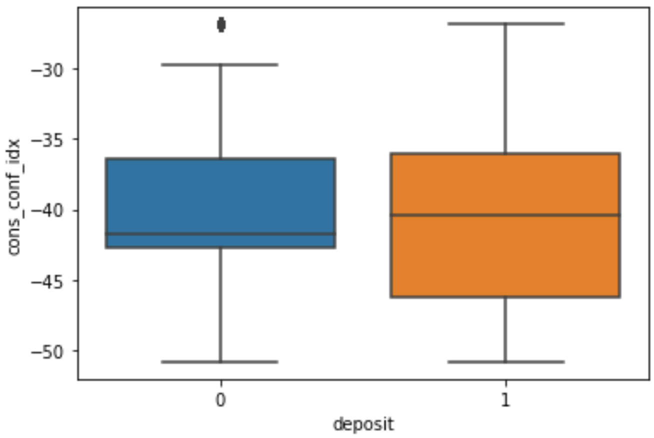
Duration relates to the length of time in seconds that the specified client was called. We felt it was necessary to include these outliers since the range of time for calls is usually broad.

--

Now let us create another boxplot this time featuring **deposit** against the **consumer confidence index**.

In [17]:
```
sns.boxplot(x = 'deposit', y = 'cons_conf_idx', data = bank_model_df)
```

Out[17]: <AxesSubplot:xlabel='deposit', ylabel='cons_conf_idx'>



Consumer Confidence Index measures the degree of optimism that the bank client has with the country's overall economic state in proportion to the client's own financial situations.

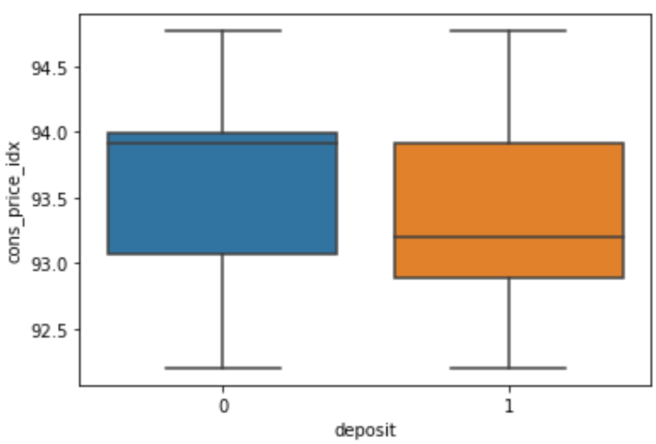
As shown in the plot, the outliers for *consumer confidence* express more optimism compared to the client's who fell within the central tendency. Due to this, we felt that it was best to keep such individuals in our dataset since they generally reflect the optimism of those individuals who help influence the economy.

--

Lastly, let us create a final boxplot featuring **deposit** against **consumer price index**.

In [19]:
```
sns.boxplot(x = 'deposit', y = 'cons_price_idx', data = bank_model_df)
```

Out[19]: <AxesSubplot:xlabel='deposit', ylabel='cons_price_idx'>



As we can see based on the boxplot, there are no outliers for the **consumer price index** when deposit is 'yes' and when deposit is 'no'.

--

Data Statistics

In order to define key statistics, we will need to use the .describe function.

In [19]:
```
bank_model_df.describe()
```

Out[19]:

	duration	nr_employed	euribor3m	emp_var_rate	cons_conf_idx	cons_price_idx	deposit
count	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000
mean	258.285010	5167.035911	3.621291	0.081886	-40.502600	93.575664	0.112654
std	259.279249	72.251528	1.734447	1.570960	4.628198	0.578840	0.316173
min	0.000000	4963.600000	0.634000	-3.400000	-50.800000	92.201000	0.000000
25%	102.000000	5099.100000	1.344000	-1.800000	-42.700000	93.075000	0.000000
50%	180.000000	5191.000000	4.857000	1.100000	-41.800000	93.749000	0.000000
75%	319.000000	5228.100000	4.961000	1.400000	-36.400000	93.994000	0.000000
max	4918.000000	5228.100000	5.045000	1.400000	-26.900000	94.767000	1.000000

Immediately we can see that the longest call duration was 4918 seconds and the shortest was 0 seconds. This shows a strong correlation with our target variable since 0 second calls can be considered deposit = 0.

Another interesting statistic is that the mean of client's age is roughly 40 years old. This illustrates a stage for the common target variable.

Lastly, the standard deviation for client_was_contacted_0 (no) and client_was_contacted_1 (yes) is the exact same. That tells us that the average amount of variability of whether or not the client was contacted is the same. Each score lies the exact same distance from the mean. Since both of the standard deviation values are low (0.188230), that tells us that both values for whether or not the client was contacted are clustered around the mean.

--

Now that we have evaluated our final dataset, we can begin the construction of our first classification model, Logistic Regression!

Logistic Regression Model

To build our Logistic Regression model, the following code will be used to determine the probability of a client securing a term deposit using the y_prediction variable in python.

At the same time we will also print the C-Statistic as a string to show the predictive accuracy of this Logistic Regression model.

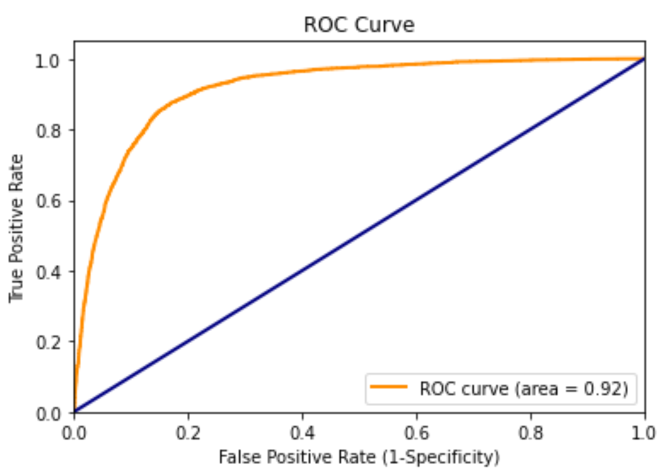
In [28]:
```
y_prediction = pd.DataFrame(logistic_model.predict(sm.add_constant(x_train.drop(columns_tc
y_prediction.columns = ["probabilities"]
both_df = pd.concat([y_train, y_prediction], axis=1)
zeros_df = both_df[['deposit', 'probabilities']][both_df['deposit'] == 0]
ones_df = both_df[['deposit', 'probabilities']][both_df['deposit'] == 1]
joined_df = df_crossjoin(ones_df, zeros_df)
joined_df['concordant_pair'] = 0
joined_df.loc[joined_df['probabilities_x'] > joined_df['probabilities_y'], 'concordant_pai
joined_df['discordant_pair'] = 0
joined_df.loc[joined_df['probabilities_x'] < joined_df['probabilities_y'], 'discordant_pai
joined_df['tied_pair'] = 0
joined_df.loc[joined_df['probabilities_x'] == joined_df['probabilities_y'], 'tied_pair'] =
p_concordant = (sum(joined_df['concordant_pair']) * 1.0 ) / (joined_df.shape[0])
p_discordant = (sum(joined_df['discordant_pair']) * 1.0 ) / (joined_df.shape[0])
c_statistic = 0.5 + (p_concordant - p_discordant) / 2.0
print("C-statistic: " + str(c_statistic))
```

```
C-statistic: 0.9155895049081026
```

Now that the Linear Regression model has been built, we will plot out the ROC curve which will also include the Area Under Curve to provide us with an accuracy inside of a visual representation.

In [29]:
```
fpr, tpr, thresholds = metrics.roc_curve(both_df['deposit'], both_df['probabilities'], pos
roc_auc = auc(fpr, tpr)

%matplotlib inline
plt.figure()
lw = 2
plt.plot(fpr, tpr, color='darkorange', lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate (1-Specificity)')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc="lower right")
plt.show()
```



In order for us to optimize the classifier of our test set, we will need to determine the threshold cut-off point for the probability of our training set.

We can use the grid search to determine the threshold.

In [30]:
```
for i in list(np.arange(0, 1, 0.1)):
    both_df["y_predictor"] = 0
    both_df.loc[both_df["probabilities"] > i, 'y_predictor'] = 1
    print ("Threshold", round(i, 2), "Train Accuracy:", round(accuracy_score(both_df['depo
```

```
Threshold 0.0 Train Accuracy: 0.1115
```

```
Threshold 0.1 Train Accuracy: 0.8393
Threshold 0.2 Train Accuracy: 0.894
Threshold 0.3 Train Accuracy: 0.9066
Threshold 0.4 Train Accuracy: 0.9096
Threshold 0.5 Train Accuracy: 0.9076
Threshold 0.6 Train Accuracy: 0.9057
Threshold 0.7 Train Accuracy: 0.9019
Threshold 0.8 Train Accuracy: 0.898
Threshold 0.9 Train Accuracy: 0.8949
```

--

Based on this data, we can see that the optimal threshold cut-off point can be found at 0.4 with an accuracy of 0.9096

Now we can set the threshold to 0.4 when classifying our data.

In order to visualize the test set accuracy of our Logistic Regression model, we can print a confusion matrix that will allow us to compare this model with our other model.

In [31]:
```
y_prediction_test_df = pd.DataFrame(logistic_model.predict(sm.add_constant(x_test.drop(col
y_prediction_test_df.columns = ["probabilities"]
both_test_df = pd.concat([y_test, y_prediction_test_df], axis=1)
both_test_df["y_predictor"] = 0
both_test_df.loc[both_test_df["probabilities"] > 0.5, 'y_predictor'] = 1
print ("Test Confusion Matrix\n", pd.crosstab(both_test_df['deposit'], both_test_df['y_pre
print ("Test Accuracy:", round(accuracy_score(both_test_df['deposit'], both_test_df['y_pre
```

```
Test Confusion Matrix
 Predicted      0    1
Actual
0           10605  326
1             902  524
Test Accuracy: 0.9006
```

--

As we can see, our Test Accuracy shows 0.9 for our Logistic Regression model. Let's build another Machine Learning model for Classification and compare the Test Accuracy results. Our other model for classification will be the Random Forest model.

Random Forest Model

We can reuse the same dummy variables that we created from our dataset to use in the Random Forest model. Independent variables that we excluded from our Logistic Regression model can be represented by new dummy variables that we will now create.

In [32]:
```
marital_dummies = pd.get_dummies(bank_df['marital'], prefix='marital')
education_dummies = pd.get_dummies(bank_df['education'], prefix='education')
housing_dummies = pd.get_dummies(bank_df['housing'], prefix='housing')
loan_dummies = pd.get_dummies(bank_df['loan'], prefix='loan')
day_of_week_dummies = pd.get_dummies(bank_df['day_of_week'], prefix='day_of_week')
# dummies not constructed here were already constructed as part of the logistic regression
bank_rand_forest_df = pd.concat([bank_df[['age', 'duration', 'campaign', 'pdays', 'previou
                                          'cons_conf_idx', 'euribor3m', 'nr_employed', 'dep
                                 job_dummies, marital_dummies, education_dummies, default_c
                                 contact_dummies, month_dummies, day_of_week_dummies, poutc
bank_rand_forest_df.head()
```

Out[32]:

	age	duration	campaign	pdays	previous	emp_var_rate	cons_price_idx	cons_conf_idx	euribor3m	nr_emp
0	44	210	1	0	0	1.4	93.444	-36.1	4.963	5
1	53	138	1	0	0	-0.1	93.200	-42.0	4.021	5
2	28	339	3	6	2	-1.7	94.055	-39.8	0.729	4

	age	duration	campaign	pdays	previous	emp_var_rate	cons_price_idx	cons_conf_idx	euribor3m	nr_emp
3	39	185	2	0	0	-1.8	93.075	-47.1	1.405	5

--

We will now split the dataset including the dummy variables into the train and test set. The train size will be the exact same as our Logistic Regression model at 70%.

Using the classifier, we will specify the hyperparameters of the training set in order to optimize our results for the test set.

In order to fully represent the accuracy of the Random Forest model effectively, we will print a confusion matrix for both the train set and the test set.

--

In [33]:

```
x_train, x_test, y_train, y_test = train_test_split(bank_rand_forest_df.drop(['deposit'],
                                                    bank_rand_forest_df['deposit'],
                                                    train_size=0.7,
                                                    random_state=42) # random state set fo
rand_forest_fit = RandomForestClassifier(n_estimators=1000,
                                         criterion="gini",
                                         max_depth=100,
                                         min_samples_split=3,
                                         min_samples_leaf=2) # these hyperparameters will
rand_forest_fit.fit(x_train, y_train)
print("Random Forest - Train Confusion Matrix\n", pd.crosstab(y_train,
                                                              rand_forest_fit.predict(x_tr
                                                              rownames=["Actual"],
                                                              colnames=["Predicted"]))
print("Random Forest - Train accuracy", round(accuracy_score(y_train, rand_forest_fit.pred
print("Random Forest - Test Confusion Matrix", pd.crosstab(y_test,
                                                           rand_forest_fit.predict(x_test)
                                                           rownames=["Actual"],
                                                           colnames=["Predicted"]))
print("Random Forest - Test accuracy", round(accuracy_score(y_test, rand_forest_fit.predic
```

```
Random Forest - Train Confusion Matrix
 Predicted      0     1
Actual
0          25553    64
1            624  2590
Random Forest - Train accuracy 0.976
Random Forest - Test Confusion Matrix Predicted      0    1
Actual
0          10627  304
1            777  649
Random Forest - Test accuracy 0.913
```

--

With the inception of both the train and test confusion matrix for the train and test set, we can see that the train set has an accuracy of 0.97 which can be expected given its function.

However when we look at the confusion matrix for the test set, we can see that the accuracy of the test set is 0.91 which is actually very good.

This demonstrates that our Random Forest model is highly capable of making accurate predictions on whether a bank client will subscribe to a term deposit given our final dataset.

We will now perform a grid search to tune the hyperparameters of the random forest model allowing us to witness if there will be a potential for improvement to the model.

In [34]:
```
pipeline = Pipeline([('clf', RandomForestClassifier(criterion='gini'))])
parameters = {
    'clf__n_estimators': (200, 300, 500),
    'clf__max_depth': (20, 30, 50),
    'clf__min_samples_split': (2, 3),
    'clf__min_samples_leaf': (1, 2)}
grid_search = GridSearchCV(pipeline, parameters, n_jobs=-1, cv=5, verbose=1, scoring='accu
grid_search.fit(x_train, y_train)
print('Best Training score: ' + str(grid_search.best_score_))
print('Best parameters set:')
best_parameters = grid_search.best_estimator_.get_params()

for param_name in sorted(parameters.keys()):
    print(str(param_name) + ': ' + str(best_parameters[param_name]))

predictions = grid_search.predict(x_test)
print("Testing accuracy: " + str(accuracy_score(y_test, predictions)))
print("Complete report of Testing data", classification_report(y_test, predictions))
print("Random Forest Grid Search - Test Confusion Matrix", pd.crosstab(y_test, predictions
```

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  5.0min
[Parallel(n_jobs=-1)]: Done 180 out of 180 | elapsed: 29.2min finished
Best Training score: 0.9152301109672223
Best parameters set:
clf__max_depth: 50
clf__min_samples_leaf: 2
clf__min_samples_split: 3
clf__n_estimators: 300
Testing accuracy: 0.9128429230395727
Complete report of Testing data               precision    recall  f1-score   support

           0       0.93      0.97      0.95     10931
           1       0.68      0.46      0.55      1426

    accuracy                           0.91     12357
   macro avg       0.81      0.71      0.75     12357
weighted avg       0.90      0.91      0.91     12357

Random Forest Grid Search - Test Confusion Matrix Predicted      0    1
Actual
0           10631  300
1             777  649
```

--

As we can see, the hyperparameter tuning did not noticeably improve the Random Forest model since the Testing Accuracy remains about the same at 0.91

--

Before we compare our models, we should plot the order of importance for the variables featured in our Random Forest model so we can represent their predictive power of our target variable. This is helpful to see since we have not eliminated the low predictive power variables in our Random Forest since it is not required like it is in Logistic Regression.

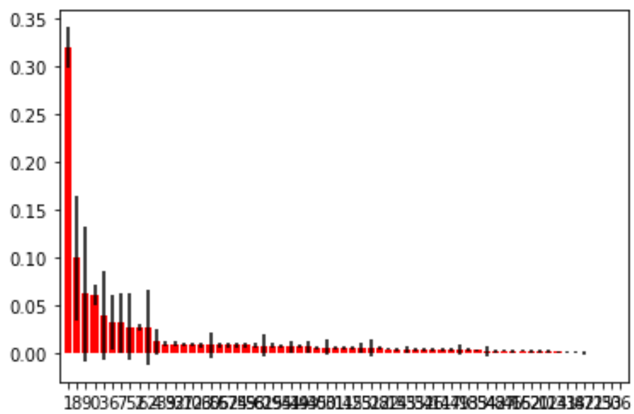
In [42]:
```
random_forest_fit = RandomForestClassifier(n_estimators=500, criterion="gini", max_depth=3
random_forest_fit.fit(x_train, y_train)
importances = random_forest_fit.feature_importances_
standard_deviations = np.std([tree.feature_importances_ for tree in random_forest_fit.esti
indices = np.argsort(importances)[::-1]
column_names = list(x_train.columns)

print("Feature ranking:")
for feature in range(x_train.shape[1]):
    print ("Feature", indices[feature], ",", column_names[indices[feature]], importances[i

plt.figure()
plt.bar(range(x_train.shape[1]), importances[indices], color="r", yerr=standard_deviations
plt.xticks(range(x_train.shape[1]), indices)
plt.xlim([-1, x_train.shape[1]])
plt.show()
```

```
Feature ranking:
Feature 1 , duration 0.3193237858842872
Feature 8 , euribor3m 0.09967824750087868
Feature 9 , nr_employed 0.06195920621340059
Feature 0 , age 0.06118213730390594
Feature 3 , pdays 0.03898235515698117
Feature 6 , cons_price_idx 0.031983423746595095
Feature 7 , cons_conf_idx 0.03170000439723065
Feature 5 , emp_var_rate 0.02795280448657108
Feature 2 , campaign 0.02724909861281814
Feature 62 , poutcome_success 0.0267395351091288
Feature 4 , previous 0.012016064634860772
Feature 39 , housing_yes 0.009822996769720867
Feature 32 , education_university.degree 0.009794192350282765
Feature 37 , housing_no 0.009435401102366073
Feature 10 , job_admin. 0.009334541548162368
Feature 23 , marital_married 0.009013222130410987
Feature 60 , poutcome_failure 0.00899232496624738
Feature 56 , day_of_week_mon 0.008857586485628455
Feature 57 , day_of_week_thu 0.008661499660287566
Feature 24 , marital_single 0.008489511313230514
Feature 59 , day_of_week_wed 0.008402814702665679
Feature 58 , day_of_week_tue 0.00832578779578923
Feature 61 , poutcome_nonexistent 0.00802301943749594
Feature 29 , education_high.school 0.008016275812583825
Feature 55 , day_of_week_fri 0.0079695001369347
Feature 44 , contact_telephone 0.007236634054520694
Feature 19 , job_technician 0.00722106939533741
Feature 43 , contact_cellular 0.006884189359821496
Feature 40 , loan_no 0.006400354920268354
Feature 50 , month_mar 0.006377263893327725
Feature 31 , education_professional.course 0.006085562251922908
Feature 11 , job_blue-collar 0.006072000560944257
Feature 42 , loan_yes 0.005991053016720622
Feature 53 , month_oct 0.005974933343865747
Feature 51 , month_may 0.005813016789564767
Feature 28 , education_basic.9y 0.005268625878076265
Feature 22 , marital_divorced 0.004644875467281685
Feature 15 , job_retired 0.004356103581351798
Feature 45 , month_apr 0.004181706556000872
Feature 35 , default_unknown 0.004114582060735629
Feature 34 , default_no 0.004113298844713186
Feature 26 , education_basic.4y 0.004113115477780585
Feature 14 , job_management 0.004062907977422505
Feature 17 , job_services 0.0038916816232930826
Feature 49 , month_jun 0.003585184662777236
Feature 18 , job_student 0.003451762660735613
Feature 33 , education_unknown 0.0033176573595715977
Feature 54 , month_sep 0.002810997618215888
Feature 48 , month_jul 0.002716209103132174
Feature 27 , education_basic.6y 0.002651341933343562
Feature 46 , month_aug 0.0025858085853596017
Feature 16 , job_self-employed 0.0024762575751631805
Feature 52 , month_nov 0.002281980790073924
Feature 20 , job_unemployed 0.002247736464718669
```

```
Feature 12 , job_entrepreneur 0.0020497280414438932
Feature 13 , job_housemaid 0.0016612247881777668
Feature 41 , loan_unknown 0.0009212165522398988
Feature 38 , housing_unknown 0.0009169715804454155
Feature 47 , month_dec 0.0007735134083212998
Feature 21 , job_unknown 0.0005641208727264714
Feature 25 , marital_unknown 0.0002161572861805036
Feature 30 , education_illiterate 6.381840595909982e-05
```



As you can see, our top five variables for predictive power in our Random Forest model are:

1. duration
2. euribor3m
3. nr_employed
4. age
5. pdays

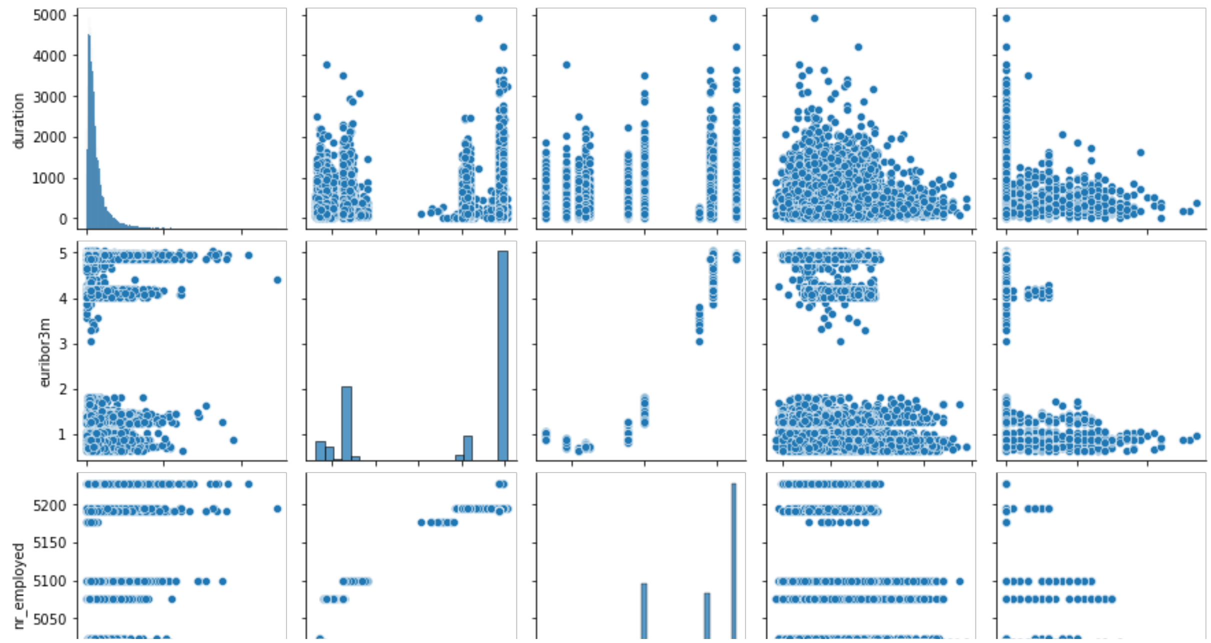
--

Let us take these top five variables with predictive power in our dataset and visualize it in a matrix using the pairplot function in the seaborn library. This will display all of our top five predictive variables on both axis allowing us to cross-examine each of them and check for interdependency.

In [16]:
```
fig = plt.figure(figsize = (20,20))

sns.pairplot(bank_model_df[['duration','euribor3m','nr_employed','age','pdays']])
```

Out[16]:
```
<seaborn.axisgrid.PairGrid at 0x119beb880>
<Figure size 1440x1440 with 0 Axes>
```



This pairplot demonstrates the independency between future predictors. This is a good indicator that each of these features are interdependent with each other since each graph in the matrix is unique in both density and design.

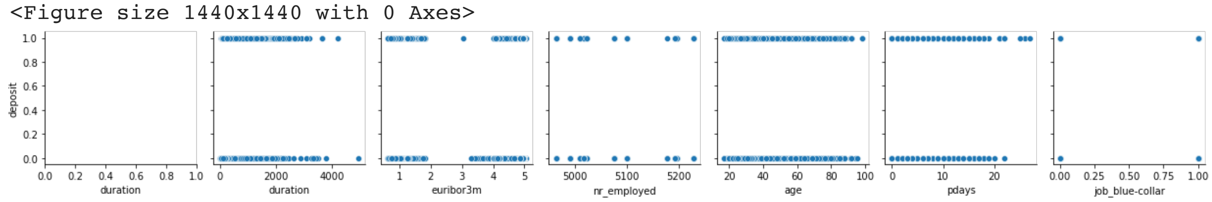
--

We can also plot our top five variables in another pairplot, this time visualizing them directly against our target variable. This will allow us a more simple and direct approach for comparing these predictive variables with the target variable.

In [29]:
```
fig = plt.figure(figsize = (20,20))

sns.pairplot(data = bank_model_df, x_vars = ['duration', 'duration', 'euribor3m', 'nr_empl
```

Out[29]:
```
<seaborn.axisgrid.PairGrid at 0x124289850>
<Figure size 1440x1440 with 0 Axes>
```



We included job_blue-collar in this pairplot since it has low predictive power and we wanted to compare it to other features with high predictive power.

This pairplot shows us that all other features are strong predictors for the label target. We can verify this since all of the features in this plot are quite dense with the exception of job_blue-collar which is sparse.

Side note: We had to include duration twice because for some unknown reason to us, the first x-axis variable in this plot would always show up blank no matter which variable it was.

Comparison of Models

Now we are at the point where we can compare our models. This will greatly help us with the selection of our model for the construction of our app. Let's get to it!

These are the results of the **Logistic Regression** model:

Test Confusion Matrix Predicted 0 1 Actual 0 10605 326 1 902 524 Test Accuracy: 0.9006

This tells us that for our **Logistic Regression** model, our test results are the following:

- **True Positives** = 524
- **True Negatives** = 10605
- **False Positives** = 326
- **False Negatives** = 902
- **Test Accuracy** = 0.90

--

These are the results of the **Random Forest** model:

Random Forest - Train Confusion Matrix Predicted 0 1 Actual 0 25553 64 1 624 2590 Random Forest - Train accuracy 0.976 Random Forest - Test Confusion Matrix Predicted 0 1 Actual 0 10627 304 1 777 649 Random Forest - Test accuracy 0.913

This tells us that for our **Random Forest** model, our test results are the following:

- **True Positives** = 2590
- **True Negatives** = 25553
- **False Positives** = 64
- **False Negatives** = 624
- **Test Accuracy** =0.91

The component we would like to analyze for model comparison would be the testing accuracy of the models, which can be found in the corresponding confusion matrices. They are the following (rounded to second decimal place):

Logistic Regression Testing Accuracy: **0.90**

Random Forest Testing Accuracy: **0.91**

--

Based on these results, the Random Forest model has a higher testing accuracy than the Logistic Regression model. We also do not need to worry about multi-collinearity with the Random Forest model.

It is for these reasons that we have decided to use our Random Forest model for our Supervised Classification Machine Learning solution!

--

Performance Evaluation

Upon completion of this assignment, there are both elements that we have done well with as well as elements that we could have done better with.

*Elements that we have done well with:

- Statistical analysis on the predictors.
- Feature engineering on data.

*Elements that we could have done better with:

- We could have discarded duration since it highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes.

- We should have used SMOTE for oversampling to balance our dataset

Deployment

Our Bank Analysis app would have real life application which will help financial institutions make decisions regarding which bank clients should be targeted for term deposit subscriptions.

Our app will definitely require updates to train the data every year to reflect the current behaviour based on different economic situations. We can upgrade this app with different classification models based on current trends.

Bibliography

Book

{Statistics for Machine Learning - Builds supervised, unsupervised, and reinforcement learning models using both Python and R} by Pratap Dangeti

Web Page

{Kaggle - For Bank Marketing Dataset} - https://www.kaggle.com/henriqueyamahata/bank-marketing

{UCI - Center for Machine Learning and Intelligent Systems} - http://archive.ics.uci.edu/ml/datasets/Bank+Marketing#