# Predicting Marketing Outcomes with Logistic Regression

Douglas Ehlert

Predicting Marketing Outcomes with Logistic Regression

## Research Question

The research question that this analysis will answer is if, by using the available data, a financial institution can predict the outcome of a new marketing effort on a particular prospect. (A prospect is the person to which the marketing is directed.) This is a very pertinent question to modern organizations. Large quantities of resources are spent on marketing, and organizations want to ensure that their dollars are spent intelligently and efficiently. The alternative hypothesis being tested is stated as follows. By utilizing age, education level, marital status, the number of previous marketing contacts, and current account balance as input variables, a financial institution can predict, with a level of statistical significance, whether the prospect will enroll in a new service.

## Data Collection

The data for this analysis was collected from UC-Irvine's Machine Learning Repository. (*UCI Machine Learning Repository: Bank Marketing Data Set*, 2012). It is publicly available. The advantage to this type of data collection is that it is readily accessible. One disadvantage is that publicly available data can need a great deal of data cleaning before it is usable. One challenge that was encountered during the data collection process was the necessity of learning the meaning of various input variables. This was overcome with exploratory data analysis.

## Data Extraction and Preparation

The data was initially imported from a CSV file with the Python library Pandas. One advantage of the Pandas library is that is has a robust set of features with which to work with

data. One disadvantage is that certain parts of Panda's syntax are difficult to easily recall without referring to the Pandas documentation. This tool was used because the author has a great deal of experience with Pandas, and it is widely used in the data community.

```
import pandas as pd
df=pd.read_csv(r'/Users/dougehlert/Library/CloudStorage/OneDrive-Personal/WGU/D214-Data/bank-D214.csv')
```

```
df.head(10)
```

| | age | marital | education | balance | previouscontacts | outcome |
|---|---|---|---|---|---|---|
| **0** | 58 | married | tertiary | 2143 | 0 | no |
| **1** | 44 | single | secondary | 29 | 0 | no |
| **2** | 33 | married | secondary | 2 | 0 | no |
| **3** | 47 | married | unknown | 1506 | 0 | no |
| **4** | 33 | single | unknown | 1 | 0 | no |
| **5** | 35 | married | tertiary | 231 | 0 | no |
| **6** | 28 | single | tertiary | 447 | 0 | no |
| **7** | 42 | divorced | tertiary | 2 | 0 | no |
| **8** | 58 | married | primary | 121 | 0 | no |
| **9** | 43 | single | secondary | 593 | 0 | no |

The data was then checked for null values, again utilizing the Pandas library.

```
In [5]: df.isnull().sum()
```

```
Out[5]: age               0
        marital           0
        education         0
        balance           0
        previouscontacts  0
        outcome           0
        dtype: int64
```

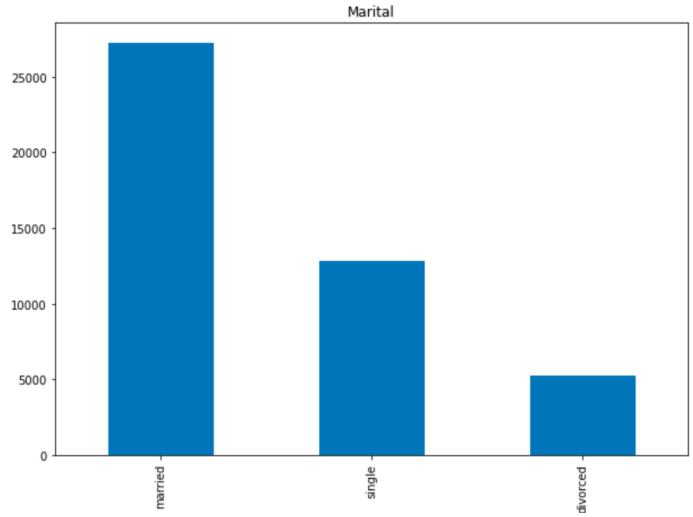
The categorical input variables were then visualized. 'PyPlot', from the 'MatPlotLib' library, was chosen to be used in conjunction with Pandas for visualization because it has robust documentation and produces insightful, simple visualizations. A disadvantage of PyPlot is that customizations of visualizations is time consuming. For exploratory data analysis, PyPlot and Pandas will be sufficient. Marital status will be reduced to two options in later data

manipulation.  Logically, if a person is neither married, nor single, they are divorced.

```
#One of these categories will be eliminated during one-hot encoding
marital_status = df['marital'].value_counts()
marital_status

plt.figure(figsize = (10, 7))
#.plot uses pandas
marital_status.plot(kind = "bar")
plt.title("Marital")
```

```
Out[6]: Text(0.5, 1.0, 'Marital')
```

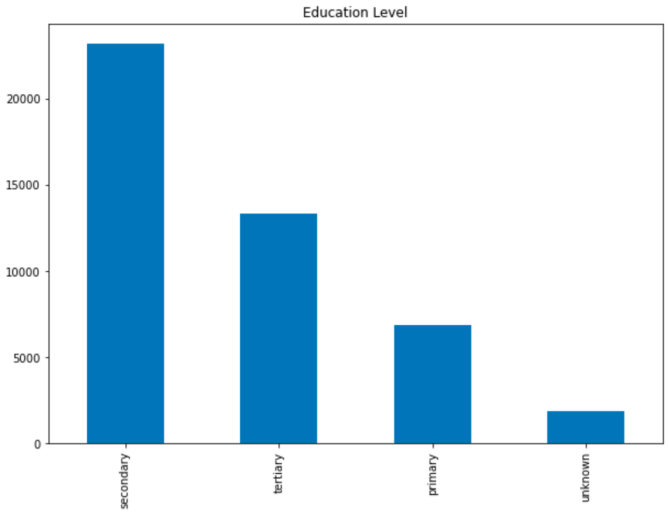



The same logic will apply to education level.  If a label of secondary, tertiary, or primary does not exist, it must be “unknown”.  Therefore, unknown will be dropped in future data manipulation.

In [7]:
```
education_level = df['education'].value_counts()
education_level

plt.figure(figsize = (10, 7))
#.plot uses pandas
education_level.plot(kind = "bar")
plt.title("Education Level")
```

Out[7]:
```
Text(0.5, 1.0, 'Education Level')
```




Next, descriptions of the continuous variables are printed.

In [8]:
```
account_balance = df['balance']
account_balance

account_balance.describe()
```

Out[8]:
```
count     45211.000000
mean       1362.272058
std        3044.765829
min       -8019.000000
25%          72.000000
50%         448.000000
75%        1428.000000
max      102127.000000
Name: balance, dtype: float64
```

In [9]:
```
previous_contacts = df['previouscontacts']
previous_contacts

previous_contacts.describe()
```

Out[9]:
```
count    45211.000000
mean         0.580323
std          2.303441
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max        275.000000
Name: previouscontacts, dtype: float64
```

In [10]:
```
age = df['age']
age

age.describe()
```

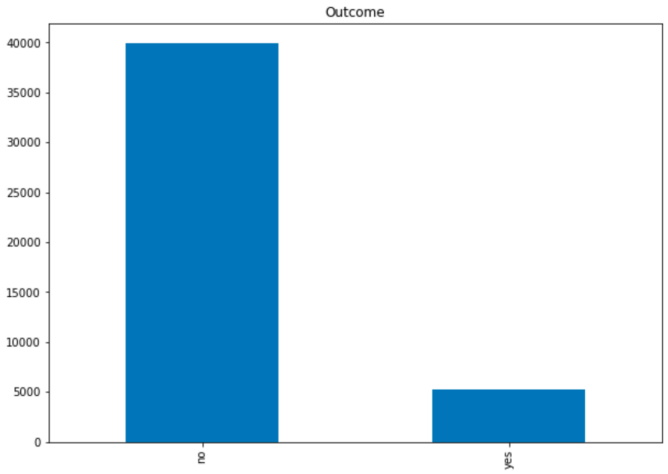
Out[10]:
```
count    45211.000000
mean        40.936210
std         10.618762
min         18.000000
25%         33.000000
50%         39.000000
75%         48.000000
max         95.000000
Name: age, dtype: float64
```

The outcome variable, whether or not the prospect signs up for the new service, is visualized.

```
In [11]: outcome = df['outcome'].value_counts()
         outcome

         plt.figure(figsize = (10, 7))
         #.plot uses pandas
         outcome.plot(kind = "bar")
         plt.title("Outcome")
```

```
Out[11]: Text(0.5, 1.0, 'Outcome')
```




Next the ‘sklearn’ module LabelEncoder will be used to convert the ‘Yes’ or ‘No’ of the outcome variable to a binary 1 or 0, 1 for yes, 0 for no. The advantage to using LabelEncoder is that it is an efficient way to turn the yes/no into 1/0. The disadvantage is that if the data is not simply a binary choice there is not a way to assign a ranking to the data (Nixon, 2020). In this

case, there is no concern about ranking because the data only contains yes or no.

In [13]:
```
#Label encoder object
le = LabelEncoder()
# Columns with one or two values can have label encoder run on them
le_count=0
for col in df.columns[1:]:
    if df[col].dtype == 'object':
        if len(list(df[col].unique())) <= 2:
            le.fit(df[col])
            df[col] = le.transform(df[col])
            le_count +=1
print('{} columns were label encoded.' .format(le_count))

df
```

```
1 columns were label encoded.
```

Out[13]:

| | age | marital | education | balance | previouscontacts | outcome |
|---|---|---|---|---|---|---|
| **0** | 58 | married | tertiary | 2143 | 0 | 0 |
| **1** | 44 | single | secondary | 29 | 0 | 0 |
| **2** | 33 | married | secondary | 2 | 0 | 0 |
| **3** | 47 | married | unknown | 1506 | 0 | 0 |
| **4** | 33 | single | unknown | 1 | 0 | 0 |
| **...** | ... | ... | ... | ... | ... | ... |
| **45206** | 51 | married | tertiary | 825 | 0 | 1 |
| **45207** | 71 | divorced | primary | 1729 | 0 | 1 |
| **45208** | 72 | married | secondary | 5715 | 3 | 1 |
| **45209** | 57 | married | secondary | 668 | 0 | 0 |
| **45210** | 37 | married | secondary | 2971 | 11 | 0 |

45211 rows × 6 columns

The next step is to convert the categorical variables of marital status and education into numerical values. This will be done with Panda's "getdummies". The advantage to "getdummies" is that it is easy to see the result; the result will be laid out in new columns. One disaadvantage is that the removal of the old columns must be completed before analysis. In the case of the marital variable, three new columns are created. Each record will have a 1 (true) or a 0 (false) for each new column. A logistic regression could not be run with the original data as it

contained text.

In [14]:
```
#getdummies will be used to assign values and create new columns to the categorical variables
#creating a dummy variable for marital
marital_cat= pd.get_dummies(df['marital'])

marital_cat
```

Out[14]:

| | divorced | married | single |
|---|---|---|---|
| **0** | 0 | 1 | 0 |
| **1** | 0 | 0 | 1 |
| **2** | 0 | 1 | 0 |
| **3** | 0 | 1 | 0 |
| **4** | 0 | 0 | 1 |
| **...** | ... | ... | ... |
| **45206** | 0 | 1 | 0 |
| **45207** | 1 | 0 | 0 |
| **45208** | 0 | 1 | 0 |
| **45209** | 0 | 1 | 0 |
| **45210** | 0 | 1 | 0 |

45211 rows × 3 columns

The same process is completed for education.

In [15]:
```
#dummy variable for education
education_cat = pd.get_dummies(df['education'])
education_cat
```

Out[15]:

| | primary | secondary | tertiary | unknown |
|---|---|---|---|---|
| **0** | 0 | 0 | 1 | 0 |
| **1** | 0 | 1 | 0 | 0 |
| **2** | 0 | 1 | 0 | 0 |
| **3** | 0 | 0 | 0 | 1 |
| **4** | 0 | 0 | 0 | 1 |
| **...** | ... | ... | ... | ... |
| **45206** | 0 | 0 | 1 | 0 |
| **45207** | 1 | 0 | 0 | 0 |
| **45208** | 0 | 1 | 0 | 0 |
| **45209** | 0 | 1 | 0 | 0 |
| **45210** | 0 | 1 | 0 | 0 |

45211 rows × 4 columns

Using Pandas, the new columns are concatanted into the existing dataframe and the original education and marital variables are dropped from the dataframe.

In [16]:
```
#Combining the dummy variables into the dataset
df= pd.concat([df, education_cat, marital_cat], axis =1)
df
```

Out[16]:

| | age | marital | education | balance | previouscontacts | outcome | primary | secondary | tertiary | unknown | divorced | married | single |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 58 | married | tertiary | 2143 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| **1** | 44 | single | secondary | 29 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| **2** | 33 | married | secondary | 2 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **3** | 47 | married | unknown | 1506 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| **4** | 33 | single | unknown | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **45206** | 51 | married | tertiary | 825 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| **45207** | 71 | divorced | primary | 1729 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| **45208** | 72 | married | secondary | 5715 | 3 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **45209** | 57 | married | secondary | 668 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **45210** | 37 | married | secondary | 2971 | 11 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

45211 rows × 13 columns

In [17]:
```
#drop marital and education
df=df.drop(['marital', 'education'], axis =1)
df
```

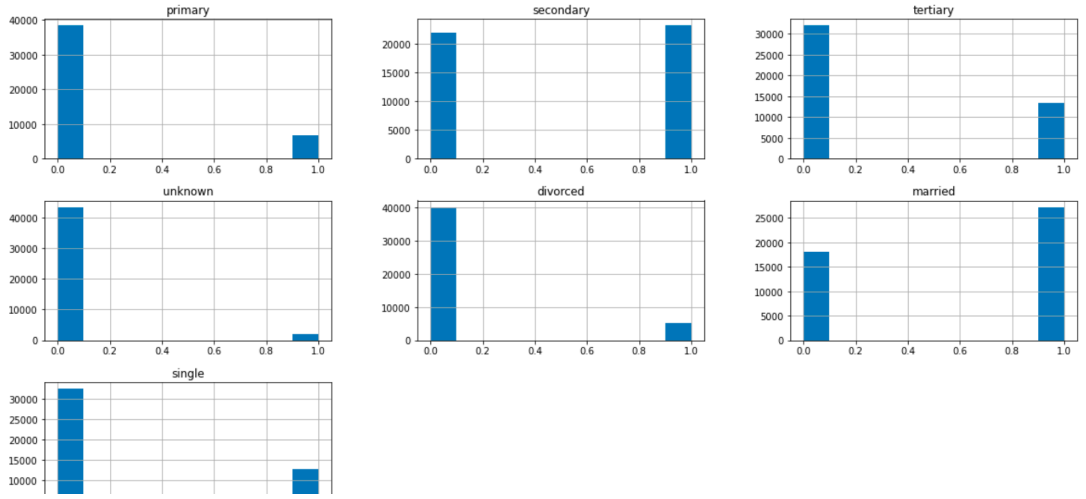
Out[17]:

| | age | balance | previouscontacts | outcome | primary | secondary | tertiary | unknown | divorced | married | single |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 58 | 2143 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| **1** | 44 | 29 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |

The categorical variables are then checked using Pandas ".hist" functionality. They only contain a 1 (true) and 0 (false).

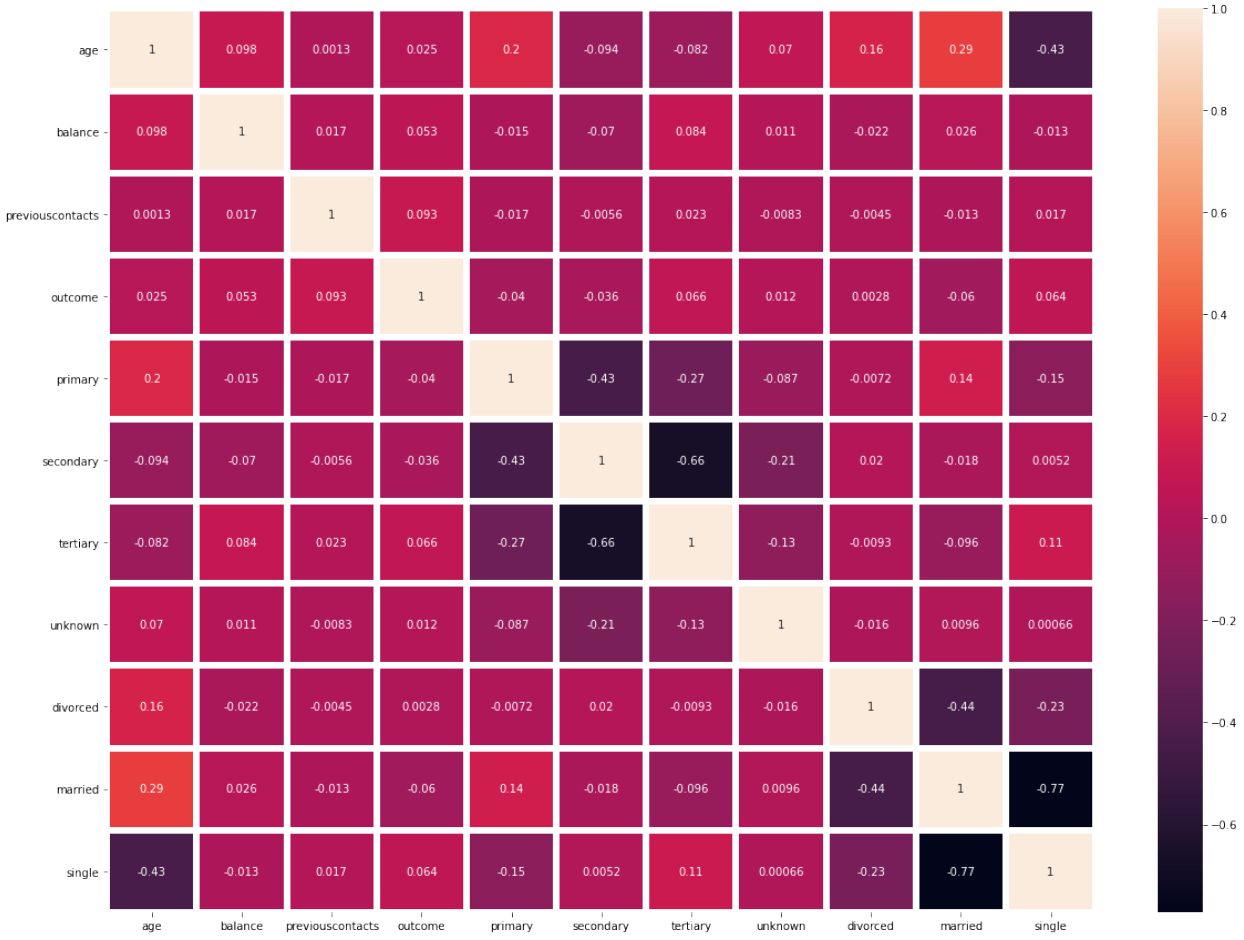
```
In [24]: cat_var.hist(figsize = (20,10))

Out[24]: array([[<AxesSubplot:title={'center':'primary'}>,
                 <AxesSubplot:title={'center':'secondary'}>,
                 <AxesSubplot:title={'center':'tertiary'}>],
                [<AxesSubplot:title={'center':'unknown'}>,
                 <AxesSubplot:title={'center':'divorced'}>,
                 <AxesSubplot:title={'center':'married'}>],
                [<AxesSubplot:title={'center':'single'}>, <AxesSubplot:>,
                 <AxesSubplot:>]], dtype=object)
```




A correlation heatmap is then produced in order to check for correlation. Logistic regression is difficult to run when there is high correlation among the input variables. This is

called ‘multicollinearity’.




It is evident that correlation exists between the variables.

Next, extraneous variables will be removed to reduce the correlation amongst input variables. The two that were previously discussed were ‘unknown’ and ‘single’. They are

removed because the new variables created with ‘getdummies’ are sufficient.

```
In [27]: df = df.drop(['unknown', 'single'], axis =1)

In [28]: df
```

Out[28]:

| | age | balance | previouscontacts | outcome | primary | secondary | tertiary | divorced | married |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 58 | 2143 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| **1** | 44 | 29 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| **2** | 33 | 2 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| **3** | 47 | 1506 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| **4** | 33 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **45206** | 51 | 825 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| **45207** | 71 | 1729 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| **45208** | 72 | 5715 | 3 | 1 | 0 | 1 | 0 | 0 | 1 |
| **45209** | 57 | 668 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| **45210** | 37 | 2971 | 11 | 0 | 0 | 1 | 0 | 0 | 1 |

45211 rows × 9 columns

To prepare the data for modelling, variance inflation scores (VIF) scores will be checked on the input variables. VIF scores are an indicator of multicollinearity. Multicollinearity among the input variables can result in a weak logistic regression model. VIF score’s advantage to the analysis is that they are readily understood. A disadvantage to using VIF scores to check input variables is that it can be time consuming with a large amount of input variables; after each input variable is removed, the VIF function must be run again on the remaining variables. If an input variable has a VIF score above 10, it will be removed from the model (*10.7 - Detecting Multicollinearity Using Variance Inflation Factors | STAT 462*, 2018). In this case, ‘age’ will be

removed from the final model.

In [29]:
```
#We can also remove variables with a VIF score above 10
from statsmodels.stats.outliers_influence import variance_inflation_factor

def calc_vif(X):
    vif=pd.DataFrame()
    vif["variables"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values, i)
                  for i in range(X.shape[1])]
    return(vif)
df_x=df[['previouscontacts', 'balance', 'age', 'primary', 'secondary', 'tertiary', 'divorced', 'married']]
calc_vif(df_x)
```

Out[29]:

| | variables | VIF |
|---|---|---|
| 0 | previouscontacts | 1.063950 |
| 1 | balance | 1.225165 |
| 2 | age | 12.916094 |
| 3 | primary | 2.709451 |
| 4 | secondary | 5.364295 |
| 5 | tertiary | 3.465449 |
| 6 | divorced | 1.592834 |
| 7 | married | 3.794651 |

Then the VIF function will be run again. No input variables have a VIF score above 10.

In [31]:
```
#rerun and repeat in order to avoid multicollinearity
#We can also remove variables with a VIF score above 10
from statsmodels.stats.outliers_influence import variance_inflation_factor

def calc_vif(X):
    vif=pd.DataFrame()
    vif["variables"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values, i)
                  for i in range(X.shape[1])]
    return(vif)
df_x=df[['previouscontacts', 'balance', 'primary', 'secondary', 'tertiary', 'divorced', 'married']]
calc_vif(df_x)
```

Out[31]:

| | variables | VIF |
|---|---|---|
| 0 | previouscontacts | 1.062642 |
| 1 | balance | 1.201573 |
| 2 | primary | 1.554595 |
| 3 | secondary | 2.277066 |
| 4 | tertiary | 1.649916 |
| 5 | divorced | 1.366611 |
| 6 | married | 2.929249 |

## Analysis

Logistic regression has been chosen for this analysis because it excels at binary classification and is easily interpreted by the human mind. A disadvantage is that the preceding steps must be performed to prepare data and check for multicollinearity. The next step is to check the p-values of the input variables by running logistic regression. If an input variable has a p-value greater that 0.05, it can be deemed statistically insignificant and removed from the final model. The advantage to using p-values to manually remove the input variables is that it is evident to all why the input variable is removed. A disadvantage is that the manual process can require a larger time investment than an automated process.

The input variables ('balance', 'previouscontacts', 'primary', 'secondary', 'tertiary', 'divorced', and 'married' are assigned to the variable 'X' and the output variable ('outcome') is assigned to 'y'.

```
y=df['outcome']
X = df.drop(columns ='outcome')

y
```

Out[33]:
```
0         0
1         0
2         0
3         0
4         0
         ..
45206     1
45207     1
45208     1
45209     0
45210     0
Name: outcome, Length: 45211, dtype: int64
```

In [34]:
```
X
```

Out[34]:

| | balance | previouscontacts | primary | secondary | tertiary | divorced | married |
|---|---|---|---|---|---|---|---|
| **0** | 2143 | 0 | 0 | 0 | 1 | 0 | 1 |
| **1** | 29 | 0 | 0 | 1 | 0 | 0 | 0 |
| **2** | 2 | 0 | 0 | 1 | 0 | 0 | 1 |
| **3** | 1506 | 0 | 0 | 0 | 0 | 0 | 1 |
| **4** | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... |

A logistic regression model is run on the data and input variables with a p-value over 0.05 will be removed.

```
Xc=sm.add_constant(X)
log_reg = sm.Logit(y, Xc)
model_fitted = log_reg.fit()
model_fitted.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.352135
         Iterations 6
```

Out[42]:

Logit Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | outcome | **No. Observations:** | 45211 |
| **Model:** | Logit | **Df Residuals:** | 45203 |
| **Method:** | MLE | **Df Model:** | 7 |
| **Date:** | Sat, 26 Nov 2022 | **Pseudo R-squ.:** | 0.02422 |
| **Time:** | 10:33:38 | **Log-Likelihood:** | -15920. |
| **converged:** | True | **LL-Null:** | -16315. |
| **Covariance Type:** | nonrobust | **LLR p-value:** | 2.387e-166 |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **const** | -1.7289 | 0.072 | -24.107 | 0.000 | -1.869 | -1.588 |
| **balance** | 3.587e-05 | 3.87e-06 | 9.260 | 0.000 | 2.83e-05 | 4.35e-05 |
| **previouscontacts** | 0.1111 | 0.006 | 18.254 | 0.000 | 0.099 | 0.123 |
| **primary** | -0.4448 | 0.081 | -5.489 | 0.000 | -0.604 | -0.286 |
| **secondary** | -0.2911 | 0.072 | -4.068 | 0.000 | -0.431 | -0.151 |
| **tertiary** | 0.0528 | 0.073 | 0.728 | 0.467 | -0.089 | 0.195 |
| **divorced** | -0.2008 | 0.050 | -4.017 | 0.000 | -0.299 | -0.103 |
| **married** | -0.3858 | 0.033 | -11.800 | 0.000 | -0.450 | -0.322 |

In this case, ‘tertiary’ will be removed and the logistic regression model will be rerun.

In [44]:
```
#tertiary will be dropped, its p-value is over .467
X = X.drop(['tertiary'], axis =1)
```

In [45]:
```
X
```

Out[45]:

| | balance | previouscontacts | primary | secondary | divorced | married |
|---|---|---|---|---|---|---|
| **0** | 2143 | 0 | 0 | 0 | 0 | 1 |
| **1** | 29 | 0 | 0 | 1 | 0 | 0 |
| **2** | 2 | 0 | 0 | 1 | 0 | 1 |
| **3** | 1506 | 0 | 0 | 0 | 0 | 1 |
| **4** | 1 | 0 | 0 | 0 | 0 | 0 |
| **...** | ... | ... | ... | ... | ... | ... |
| **45206** | 825 | 0 | 0 | 0 | 0 | 1 |
| **45207** | 1729 | 0 | 1 | 0 | 1 | 0 |
| **45208** | 5715 | 3 | 0 | 1 | 0 | 1 |
| **45209** | 668 | 0 | 0 | 1 | 0 | 1 |
| **45210** | 2971 | 11 | 0 | 1 | 0 | 1 |

45211 rows × 6 columns

In [46]:
```
#independent variables with p-values over .05 will be removed
Xc=sm.add_constant(X)
log_reg = sm.Logit(y, Xc)
model_fitted = log_reg.fit()
model_fitted.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.352141
         Iterations 6
```

Out[46]:

Logit Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | outcome | **No. Observations:** | 45211 |
| **Model:** | Logit | **Df Residuals:** | 45204 |
| **Method:** | MLE | **Df Model:** | 6 |
| **Date:** | Sat, 26 Nov 2022 | **Pseudo R-squ.:** | 0.02420 |
| **Time:** | 10:35:17 | **Log-Likelihood:** | -15921. |
| **converged:** | True | **LL-Null:** | -16315. |
| **Covariance Type:** | nonrobust | **LLR p-value:** | 2.600e-167 |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **const** | -1.6819 | 0.031 | -54.249 | 0.000 | -1.743 | -1.621 |
| **balance** | 3.592e-05 | 3.87e-06 | 9.274 | 0.000 | 2.83e-05 | 4.35e-05 |
| **previouscontacts** | 0.1112 | 0.006 | 18.268 | 0.000 | 0.099 | 0.123 |
| **primary** | -0.4912 | 0.050 | -9.861 | 0.000 | -0.589 | -0.394 |
| **secondary** | -0.3377 | 0.032 | -10.635 | 0.000 | -0.400 | -0.275 |
| **divorced** | -0.2010 | 0.050 | -4.021 | 0.000 | -0.299 | -0.103 |
| **married** | -0.3867 | 0.033 | -11.838 | 0.000 | -0.451 | -0.323 |

All p-values are under 0.05 and all input variables are therefore statistically significant.

The data will now be split into train and test sets using ‘sklearn’s train_test_split’. The advantage to the ‘train_test_split’ method is that is a simple process to break data into training and test sets. Splitting the data is necessary because a model cannot be accurately evaluated on data that it has already processed in training. The test data will be used to evaluate model accuracy. A disadvantage to ‘train_test_split’ is that the code must specify a random state. If that is not specified, the experiment will not be reproducible. The data is split into 70% train and 30% test.

In [47]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.30, random_state=42)
```

A logistic regression is then instantiated and fit to the data. This regression is used to predict the outcome from the testing input variables.

In [54]:
```
from sklearn.linear_model import LogisticRegression
logistic_regression=LogisticRegression()
logistic_regression.fit(X_train, y_train)
y_pred=logistic_regression.predict(X_test)
```

An accuracy score is calculated from the model. The model created can accurately predict the outcome 88% of the time and is statistically significant.

```
In [52]: from sklearn import metrics
         print('Accuracy: ' ,metrics.accuracy_score(y_test, y_pred))
         plt.show()

         Accuracy:  0.8800501327042171
```

## Data Summary and Implications

A logistic regression model was chosen due to the binary nature of the output variable (outcome). The final model had an accuracy score of 88% and was statistically significant because all input variables in the final model had a p-value of less than 0.05. P-values less than 0.05 are deemed to be statistically significant. The research question has successfully been answered in the positive. A model that predicts (with 88% accuracy) the outcome of a marketing campaign and is statistically significant can be created with the initial dataset.

It is recommended that the organization use this model to predict the prospect's response to the marketing campaign and use the results to target efforts to prospects that are likely to have a positive outcome.

One limitation of this analysis is that outside factors (i.e. factors that are not included in the input variables) could be having a large effect on the outcome (output variable). The outside factors would not be accounted for in this model.

Two directions for further analysis include utilizing a larger set of input values to increase model accuracy. Additionally, advanced machine learning techniques, such as K-Nearest Neighbor, could be implemented to increase model accuracy and business value.

References

Nixon, A. E. (2020, December 16). *A common mistake to avoid when encoding categorical features*. Medium. https://towardsdatascience.com/a-common-mistake-to-avoid-when-encoding-ordinal-features-79e402796ab4

*UCI Machine Learning Repository: Bank Marketing Data Set*. (2012). Uci.edu. https://archive.ics.uci.edu/ml/datasets/bank+marketing

*10.7 - Detecting Multicollinearity Using Variance Inflation Factors | STAT 462*. (2018). Online.stat.psu.edu. https://online.stat.psu.edu/stat462/node/180/