In [1]:
```python
import pandas as pd
import numpy as np
from sklearn import preprocessing
import matplotlib.pyplot as plt
plt.rc("font", size=14)
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import seaborn as sns
sns.set(style="white")
sns.set(style="whitegrid", color_codes=True)
```

The Data

The data is related with direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe (1/0) a term deposit (variable y).

This dataset provides the customer information. It includes 41188 records and 21 fields.

In [2]:
```python
data = pd.read_csv('bank-full.csv', header=0)
data = data.dropna()
print(data.shape)
print(list(data.columns))
```

```
(41188, 21)
['age', 'job', 'marital', 'education', 'default', 'housing', 'loan', 'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays', 'previous', 'poutcome', 'emp_var_rate', 'cons_price_idx', 'cons_con
f_idx', 'euribor3m', 'nr_employed', 'y']
```

In [3]:
```python
data
```

Out[3]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp_var_rate	cons_price_idx	cons_conf_idx	euribor3m	nr_employed	y
0	44	blue-collar	married	basic.4y	unknown	yes	no	cellular	aug	thu	...	1	999	0	nonexistent	1.4	93.444	-36.1	4.963	5228.1	0
1	53	technician	married	unknown	no	no	no	cellular	nov	fri	...	1	999	0	nonexistent	-0.1	93.200	-42.0	4.021	5195.8	0
2	28	management	single	university.degree	no	yes	no	cellular	jun	thu	...	3	6	2	success	-1.7	94.055	-39.8	0.729	4991.6	1
3	39	services	married	high.school	no	no	no	cellular	apr	fri	...	2	999	0	nonexistent	-1.8	93.075	-47.1	1.405	5099.1	0
4	55	retired	married	basic.4y	no	yes	no	cellular	aug	fri	...	1	3	1	success	-2.9	92.201	-31.4	0.869	5076.2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
41183	59	retired	married	high.school	unknown	no	yes	telephone	jun	thu	...	1	999	0	nonexistent	1.4	94.465	-41.8	4.866	5228.1	0
41184	31	housemaid	married	basic.4y	unknown	no	no	telephone	may	thu	...	2	999	0	nonexistent	1.1	93.994	-36.4	4.860	5191.0	0
41185	42	admin.	single	university.degree	unknown	yes	yes	telephone	may	wed	...	3	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	0
41186	48	technician	married	professional.course	no	no	yes	telephone	oct	tue	...	2	999	0	nonexistent	-3.4	92.431	-26.9	0.742	5017.5	0
41187	25	student	single	high.school	no	no	no	telephone	may	fri	...	4	999	0	nonexistent	1.1	93.994	-36.4	4.859	5191.0	0

41188 rows × 21 columns

Input variables 1 - age (numeric)

2 - job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')

3 - marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)

4 - education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')

5 - default: has credit in default? (categorical: 'no','yes','unknown')

6 - housing: has housing loan? (categorical: 'no','yes','unknown')

7 - loan: has personal loan? (categorical: 'no','yes','unknown')

8 - contact: contact communication type (categorical: 'cellular','telephone')

9 - month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')

10 - day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')

11 - duration: last contact duration, in seconds (numeric). Important note: this attribute 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 and should be discarded if the intention is to have a realistic predictive model.

12 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)

13 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)

14 - previous: number of contacts performed before this campaign and for this client (numeric)

15 - poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')

16 - emp.var.rate: employment variation rate - (numeric)

17 - cons.price.idx: consumer price index - (numeric)

18 - cons.conf.idx: consumer confidence index - (numeric)

19 - euribor3m: euribor 3 month rate - (numeric)

20 - nr.employed: number of employees - (numeric)

Predict variable (desired target):

y - has the client subscribed a term deposit? (binary: '1','0')

The education column of the dataset has many categories and we need to reduce the categories for a better modelling. The education column has the following categories:

In [4]:
```python
data['education'].unique()
```

Out[4]:
```
array(['basic.4y', 'unknown', 'university.degree', 'high.school',
       'basic.9y', 'professional.course', 'basic.6y', 'illiterate'],
      dtype=object)
```

In [5]:
```python
data['education']=np.where(data['education'] =='basic.9y', 'Basic', data['education'])
data['education']=np.where(data['education'] =='basic.6y', 'Basic', data['education'])
data['education']=np.where(data['education'] =='basic.4y', 'Basic', data['education'])
```

After grouping, this is the columns

In [6]:
```python
data['education'].unique()
```

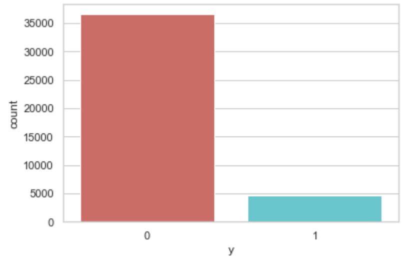
Out[6]:
```
array(['Basic', 'unknown', 'university.degree', 'high.school',
       'professional.course', 'illiterate'], dtype=object)
```

Data exploration

In [7]:
```python
data['y'].value_counts()
```

Out[7]:
```
0    36548
1     4640
Name: y, dtype: int64
```

In [8]:
```
sns.countplot(x='y',data=data, palette='hls')
plt.show()
```



In [9]:
```
count_no_sub = len(data[data['y']==0])
count_sub = len(data[data['y']==1])
pct_of_no_sub = count_no_sub/(count_no_sub+count_sub)
print("percentage of no subscription is", pct_of_no_sub*100)
pct_of_sub = count_sub/(count_no_sub+count_sub)
print("percentage of subscription", pct_of_sub*100)
```

```
percentage of no subscription is 88.73458288821988
percentage of subscription 11.265417111780131
```

In [10]:
```
data.groupby('y').mean()
```

Out[10]:

y	age	duration	campaign	pdays	previous	emp_var_rate	cons_price_idx	cons_conf_idx	euribor3m	nr_employed
0	39.911185	220.844807	2.633085	984.113878	0.132374	0.248875	93.603757	-40.593097	3.811491	5176.166600
1	40.913147	553.191164	2.051724	792.035560	0.492672	-1.233448	93.354386	-39.789784	2.123135	5095.115991

In [11]:
```
data.groupby('job').mean()
```

Out[11]:

job	age	duration	campaign	pdays	previous	emp_var_rate	cons_price_idx	cons_conf_idx	euribor3m	nr_employed	y
admin.	38.187296	254.312128	2.623489	954.319229	0.189023	0.015563	93.534054	-40.245433	3.550274	5164.125350	0.129726
blue-collar	39.555760	264.542360	2.558461	985.160363	0.122542	0.248995	93.656656	-41.375816	3.771996	5175.615150	0.068943
entrepreneur	41.723214	263.267857	2.535714	981.267170	0.138736	0.158723	93.605372	-41.283654	3.791120	5176.313530	0.085165
housemaid	45.500000	250.454717	2.639623	960.579245	0.137736	0.433396	93.676576	-39.495283	4.009645	5179.529623	0.100000
management	42.362859	257.058140	2.476060	962.647059	0.185021	-0.012688	93.522755	-40.489466	3.611316	5166.650513	0.112175
retired	62.027326	273.712209	2.476744	897.936047	0.327326	-0.698314	93.430786	-38.573081	2.770066	5122.262151	0.252326
self-employed	39.949331	264.142153	2.660802	976.621393	0.143561	0.094159	93.559982	-40.488107	3.689376	5170.674384	0.104856
services	37.926430	258.398085	2.587805	979.974049	0.154951	0.175359	93.634659	-41.290048	3.699187	5171.600126	0.081381
student	25.894857	283.683429	2.104000	840.217143	0.524571	-1.408000	93.331613	-40.187543	1.884224	5085.939086	0.314286
technician	38.507638	250.232241	2.577339	964.408127	0.153789	0.274566	93.561471	-39.927569	3.820401	5175.648391	0.108260
unemployed	39.733728	249.451677	2.564103	935.316568	0.199211	-0.111736	93.563781	-40.007594	3.466583	5157.156509	0.142012
unknown	45.563636	239.675758	2.648485	938.727273	0.154545	0.357879	93.718942	-38.797879	3.949033	5172.931818	0.112121

In [12]:
```
data.groupby('marital').mean()
```

Out[12]:

marital	age	duration	campaign	pdays	previous	emp_var_rate	cons_price_idx	cons_conf_idx	euribor3m	nr_employed	y
divorced	44.899393	253.790330	2.61340	968.639853	0.168690	0.163985	93.606563	-40.707069	3.715603	5170.878643	0.103209
married	42.307165	257.438623	2.57281	967.247673	0.155608	0.183625	93.597367	-40.270659	3.745832	5171.848772	0.101573
single	33.158714	261.524378	2.53380	949.909578	0.211359	-0.167989	93.517300	-40.918698	3.317447	5155.199265	0.140041
unknown	40.275000	312.725000	3.18750	937.100000	0.275000	-0.221250	93.471250	-40.820000	3.313038	5157.393750	0.150000

In [13]:
```
data.groupby('education').mean()
```

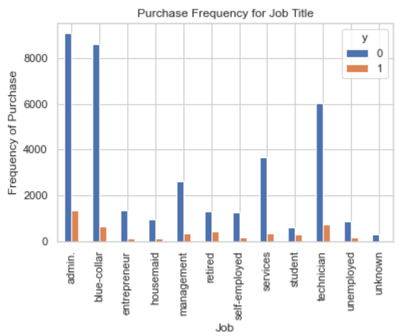
Out[13]:

education	age	duration	campaign	pdays	previous	emp_var_rate	cons_price_idx	cons_conf_idx	euribor3m	nr_employed	y
Basic	42.163910	263.043874	2.559498	974.877967	0.141053	0.191329	93.639933	-40.927595	3.729654	5172.014113	0.087029
high.school	37.998213	260.886810	2.568576	964.358382	0.185917	0.032937	93.584857	-40.940641	3.556157	5164.994735	0.108355
illiterate	48.500000	276.777778	2.277778	943.833333	0.111111	-0.133333	93.317333	-39.950000	3.516556	5171.777778	0.222222
professional.course	40.080107	252.533855	2.586115	960.765974	0.163075	0.173012	93.569864	-40.124108	3.710457	5170.155979	0.113485
university.degree	38.879191	253.223373	2.563527	951.807692	0.192390	-0.028090	93.493466	-39.975805	3.529663	5163.226298	0.137245
unknown	43.481225	262.390526	2.596187	942.830734	0.226459	0.059099	93.658615	-39.877816	3.571098	5159.549509	0.145003

Visualizations

In [14]:
```
%matplotlib inline
pd.crosstab(data.job,data.y).plot(kind='bar')
plt.title('Purchase Frequency for Job Title')
plt.xlabel('Job')
plt.ylabel('Frequency of Purchase')
```

Out[14]:
```
Text(0, 0.5, 'Frequency of Purchase')
```

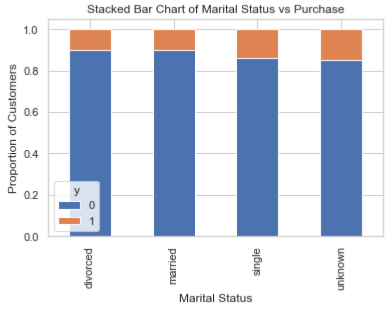


In [15]:
```
table=pd.crosstab(data.marital,data.y)
```

```
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True)
plt.title('Stacked Bar Chart of Marital Status vs Purchase')
plt.xlabel('Marital Status')
plt.ylabel('Proportion of Customers')
```

Out[15]:

```
Text(0, 0.5, 'Proportion of Customers')
```

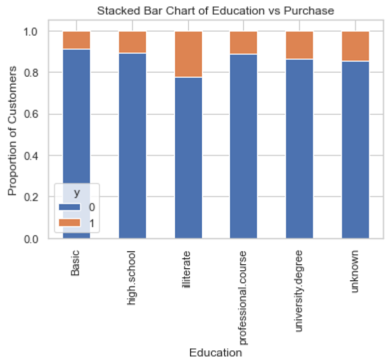


In [16]:

```
table=pd.crosstab(data.education,data.y)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True)
plt.title('Stacked Bar Chart of Education vs Purchase')
plt.xlabel('Education')
plt.ylabel('Proportion of Customers')
```

Out[16]:

```
Text(0, 0.5, 'Proportion of Customers')
```



In [17]:

```
pd.crosstab(data.day_of_week,data.y).plot(kind='bar')
plt.title('Purchase Frequency for Day of Week')
plt.xlabel('Day of Week')
plt.ylabel('Frequency of Purchase')
```

Out[17]:

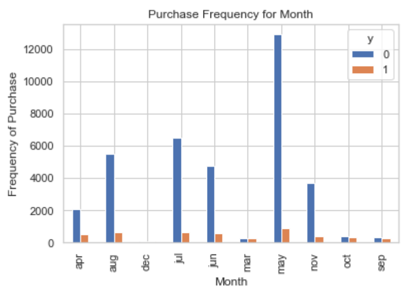
```
Text(0, 0.5, 'Frequency of Purchase')
```

In [18]:

```
pd.crosstab(data.month,data.y).plot(kind='bar')
plt.title('Purchase Frequency for Month')
plt.xlabel('Month')
plt.ylabel('Frequency of Purchase')
```

Out[18]:

```
Text(0, 0.5, 'Frequency of Purchase')
```

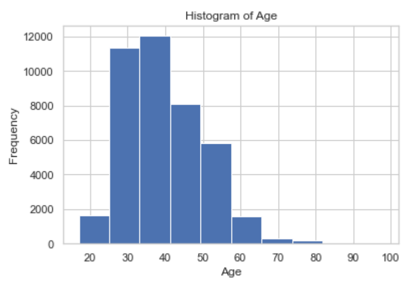


In [19]:

```
data.age.hist()
plt.title('Histogram of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
```

Out[19]:

```
Text(0, 0.5, 'Frequency')
```



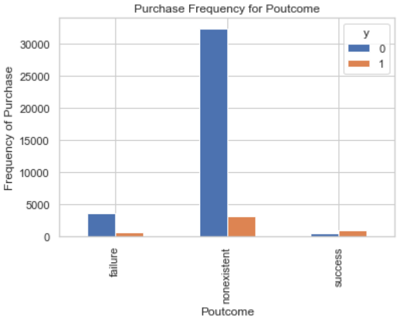
In [20]:

```
pd.crosstab(data.poutcome,data.y).plot(kind='bar')
plt.title('Purchase Frequency for Poutcome')
```

```
plt.xlabel('Poutcome')
plt.ylabel('Frequency of Purchase')
```

Out[20]:

```
Text(0, 0.5, 'Frequency of Purchase')
```



Create dummy variables

In [21]:

```
cat_vars=['job','marital','education','default','housing','loan','contact','month','day_of_week','poutcome']
for var in cat_vars:
    cat_list='var'+'_'+var
    cat_list = pd.get_dummies(data[var], prefix=var)
    data1=data.join(cat_list)
    data=data1

cat_vars=['job','marital','education','default','housing','loan','contact','month','day_of_week','poutcome']
data_vars=data.columns.values.tolist()
to_keep=[i for i in data_vars if i not in cat_vars]
data_final=data[to_keep]
data_final.columns.values
```

Out[21]:

```
array(['age', 'duration', 'campaign', 'pdays', 'previous', 'emp_var_rate',
       'cons_price_idx', 'cons_conf_idx', 'euribor3m', 'nr_employed', 'y',
       'job_admin.', 'job_blue-collar', 'job_entrepreneur',
       'job_housemaid', 'job_management', 'job_retired',
       'job_self-employed', 'job_services', 'job_student',
       'job_technician', 'job_unemployed', 'job_unknown',
       'marital_divorced', 'marital_married', 'marital_single',
       'marital_unknown', 'education_Basic', 'education_high.school',
       'education_illiterate', 'education_professional.course',
       'education_university.degree', 'education_unknown', 'default_no',
       'default_unknown', 'default_yes', 'housing_no', 'housing_unknown',
       'housing_yes', 'loan_no', 'loan_unknown', 'loan_yes',
       'contact_cellular', 'contact_telephone', 'month_apr', 'month_aug',
       'month_dec', 'month_jul', 'month_jun', 'month_mar', 'month_may',
       'month_nov', 'month_oct', 'month_sep', 'day_of_week_fri',
       'day_of_week_mon', 'day_of_week_thu', 'day_of_week_tue',
       'day_of_week_wed', 'poutcome_failure', 'poutcome_nonexistent',
       'poutcome_success'], dtype=object)
```

Over-sampling using SMOTE

In [22]:

```
X = data_final.loc[:, data_final.columns != 'y']
y = data_final.loc[:, data_final.columns == 'y']
```

In [27]:

```
from imblearn.over_sampling import SMOTE

os = SMOTE(random_state=0)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
columns = X_train.columns

os_data_X,os_data_y=os.fit_resample(X_train, y_train)
os_data_X = pd.DataFrame(data=os_data_X,columns=columns )
os_data_y= pd.DataFrame(data=os_data_y,columns=['y'])
# we can Check the numbers of our data
print("length of oversampled data is ",len(os_data_X))
print("Number of no subscription in oversampled data",len(os_data_y[os_data_y['y']==0]))
print("Number of subscription",len(os_data_y[os_data_y['y']==1]))
print("Proportion of no subscription data in oversampled data is ",len(os_data_y[os_data_y['y']==0])/len(os_data_X))
print("Proportion of subscription data in oversampled data is ",len(os_data_y[os_data_y['y']==1])/len(os_data_X))
```

```
length of oversampled data is  51134
Number of no subscription in oversampled data 25567
Number of subscription 25567
Proportion of no subscription data in oversampled data is  0.5
Proportion of subscription data in oversampled data is  0.5
```

Recursive feature elimination

In [28]:

```
data_final_vars=data_final.columns.values.tolist()
y=['y']
X=[i for i in data_final_vars if i not in y]
```

In [31]:

```
from sklearn import datasets
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression()

rfe = RFE(logreg, step = 20)
rfe = rfe.fit(os_data_X, os_data_y.values.ravel())
print(rfe.support_)
print(rfe.ranking_)
```

```
C:\Users\alokr\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:444: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\alokr\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:444: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
[False False False False False False False False False False False  True
  True False  True False  True  True False  True False False  True  True
  True False  True  True False  True  True  True False False False  True
  True  True  True  True  True  True  True False False False False False
 False  True  True False False  True  True  True  True  True  True False
 False]
[3 3 2 3 2 2 2 3 3 3 2 1 1 3 1 3 1 1 3 1 2 3 1 1 1 3 1 1 3 1 1 1 3 2 3 1 1
 1 1 1 1 1 1 2 2 3 2 3 2 1 1 3 3 1 1 1 1 1 1 3 3]
```

In [32]:

```
cols=['euribor3m', 'job_blue-collar', 'job_housemaid', 'marital_unknown', 'education_illiterate', 'default_no', 'default_unknown',
      'contact_cellular', 'contact_telephone', 'month_apr', 'month_aug', 'month_dec', 'month_jul', 'month_jun', 'month_mar',
```

```
      'month_may', 'month_nov', 'month_oct', "poutcome_failure", "poutcome_success"]
X=os_data_X[cols]
y=os_data_y['y']
```

Implementing the model

In [34]:
```
import statsmodels.api as sm
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.455620
         Iterations 7
                           Results: Logit
=====================================================================
Model:                Logit            Pseudo R-squared:  0.343
Dependent Variable:   y                AIC:               46635.3319
Date:                 2022-08-28 22:15 BIC:               46812.1760
No. Observations:     51134            Log-Likelihood:    -23298.
Df Model:             19               LL-Null:           -35443.
Df Residuals:         51114            LLR p-value:       0.0000
Converged:            1.0000           Scale:             1.0000
No. Iterations:       7.0000
---------------------------------------------------------------------
                      Coef.  Std.Err.    z     P>|z|   [0.025  0.975]
---------------------------------------------------------------------
euribor3m             0.1616   0.0082  19.8265 0.0000  0.1457  0.1776
job_blue-collar      -0.9958   0.0381 -26.1079 0.0000 -1.0706 -0.9210
job_housemaid        -1.6288   0.1377 -11.8284 0.0000 -1.8987 -1.3589
marital_unknown      -1.1082   0.4206  -2.6348 0.0084 -1.9325 -0.2838
education_illiterate  0.2414   0.6654   0.3629 0.7167 -1.0626  1.5455
default_no            0.7945   0.0371  21.4416 0.0000  0.7219  0.8671
default_unknown      -0.4725   0.0570  -8.2934 0.0000 -0.5842 -0.3608
contact_cellular      1.5117   0.0442  34.2385 0.0000  1.4252  1.5983
contact_telephone    -0.3720   0.0574  -6.4795 0.0000 -0.4846 -0.2595
month_apr            -2.1763   0.0546 -39.8467 0.0000 -2.2834 -2.0693
month_aug            -3.6205   0.0529 -68.4339 0.0000 -3.7241 -3.5168
month_dec            -1.7415   0.1714 -10.1593 0.0000 -2.0775 -1.4056
month_jul            -3.4496   0.0530 -65.1405 0.0000 -3.5534 -3.3458
month_jun            -2.0951   0.0529 -39.5739 0.0000 -2.1988 -1.9913
month_mar            -1.0935   0.0955 -11.4524 0.0000 -1.2807 -0.9064
month_may            -2.5250   0.0441 -57.2184 0.0000 -2.6115 -2.4385
month_nov            -3.6152   0.0577 -62.6892 0.0000 -3.7282 -3.5021
month_oct            -1.0505   0.0856 -12.2766 0.0000 -1.2183 -0.8828
poutcome_failure     -0.8991   0.0462 -19.4564 0.0000 -0.9896 -0.8085
poutcome_success      2.4595   0.0662  37.1402 0.0000  2.3297  2.5893
=====================================================================
```

The p-values for four variables are very high, therefore, I will remove them.

In [35]:
```
cols=['euribor3m', 'job_blue-collar', 'job_housemaid', 'marital_unknown', 'education_illiterate',
      'month_apr', 'month_aug', 'month_dec', 'month_jul', 'month_jun', 'month_mar',
      'month_may', 'month_nov', 'month_oct', "poutcome_failure", "poutcome_success"]
X=os_data_X[cols]
y=os_data_y['y']
```

In [36]:
```
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.547516
         Iterations 7
                           Results: Logit
=====================================================================
Model:                Logit            Pseudo R-squared:  0.210
Dependent Variable:   y                AIC:               56025.3883
Date:                 2022-08-28 22:15 BIC:               56166.8635
No. Observations:     51134            Log-Likelihood:    -27997.
Df Model:             15               LL-Null:           -35443.
Df Residuals:         51118            LLR p-value:       0.0000
Converged:            1.0000           Scale:             1.0000
No. Iterations:       7.0000
---------------------------------------------------------------------
                      Coef.  Std.Err.    z     P>|z|   [0.025  0.975]
---------------------------------------------------------------------
euribor3m             0.1726   0.0055  31.1231 0.0000  0.1617  0.1835
job_blue-collar      -1.0761   0.0360 -29.9182 0.0000 -1.1465 -1.0056
job_housemaid        -1.6936   0.1293 -13.1030 0.0000 -1.9469 -1.4403
marital_unknown      -1.1130   0.4136  -2.6908 0.0071 -1.9237 -0.3023
education_illiterate  0.1325   0.6595   0.2009 0.8408 -1.1602  1.4251
month_apr            -0.2660   0.0414  -6.4324 0.0000 -0.3470 -0.1849
month_aug            -1.6700   0.0393 -42.4900 0.0000 -1.7471 -1.5930
month_dec            -0.1384   0.1606  -0.8616 0.3889 -0.4532  0.1764
month_jul            -1.6077   0.0391 -41.1482 0.0000 -1.6843 -1.5311
month_jun            -1.3552   0.0394 -34.4031 0.0000 -1.4324 -1.2780
month_mar             0.7367   0.0859   8.5771 0.0000  0.5684  0.9050
month_may            -1.5298   0.0302 -50.7234 0.0000 -1.5889 -1.4707
month_nov            -1.7487   0.0467 -37.4200 0.0000 -1.8403 -1.6571
month_oct             0.4962   0.0751   6.6078 0.0000  0.3490  0.6434
poutcome_failure      0.0003   0.0419   0.0063 0.9949 -0.0818  0.0824
poutcome_success      3.1910   0.0595  53.6365 0.0000  3.0744  3.3076
=====================================================================
```

Logistic Regression Model Fitting

In [37]:
```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

```
C:\Users\alokr\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:444: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[37]:
```
▾ LogisticRegression
LogisticRegression()
```

In [38]:
```
y_pred = logreg.predict(X_test)
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))
```

```
Accuracy of logistic regression classifier on test set: 0.84
```

Confusion Matrix

In [39]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
```

```
[[6850  816]
 [1708 5967]]
```

In [40]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.80      0.89      0.84      7666
           1       0.88      0.78      0.83      7675

    accuracy                           0.84     15341
   macro avg       0.84      0.84      0.83     15341
weighted avg       0.84      0.84      0.83     15341
```

Interpretation

Of the entire test set, 74% of the promoted term deposit were the term deposit that the customers liked. Of the entire test set, 74% of the customer's preferred term deposit were promoted.

In [41]:
```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test, logreg.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```

In [ ]: