Import neccessery libraries

In [91]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from numpy.polynomial.polynomial import polyfit
from sklearn.linear_model import LogisticRegression
import seaborn as sns
import statsmodels.stats.tests.test_influence
from sklearn.feature_selection import RFE
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

import warnings
warnings.filterwarnings('ignore')
```

Problem

Whether the client has subscribed a term deposit or not Binomial ("yes" or "no")

Import data

In [3]:
```
import os
```

In [4]:
```
os.getcwd()
```

Out[4]:
```
'C:\\Users\\Akarsh\\ASSIGNMENTS'
```

In [5]:
```
os.chdir('C:\\Users\\Akarsh\\Desktop\\assignments\\logistic regression')
```

In [6]:
```
os.getcwd()
```

Out[6]:
```
'C:\\Users\\Akarsh\\Desktop\\assignments\\logistic regression'
```

In [9]:
```
bank_data = pd.read_csv('bank-full.csv',sep = ';',encoding='latin1')
bank_data
```

Out[9]:

	age	job	marital	education	default	balance	housing	loan	contact	day
0	58	management	married	tertiary	no	2143	yes	no	unknown	5
1	44	technician	single	secondary	no	29	yes	no	unknown	5
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5
4	33	unknown	single	unknown	no	1	no	no	unknown	5
...	...	...	...	...	...	...	...	...	...	...

	age	job	marital	education	default	balance	housing	loan	contact	day
45206	51	technician	married	tertiary	no	825	no	no	cellular	17
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17
45208	72	retired	married	secondary	no	5715	no	no	cellular	17
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellular	17

Data understanding

In [12]:

```
bank_data.shape
```

Out[12]:

```
(45211, 17)
```

In [13]:

```
bank_data.isna().sum()
```

Out[13]:

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

In [14]:

```
bank_data.dtypes
```

Out[14]:

```
age           int64
job          object
marital      object
education    object
default      object
balance       int64
housing      object
loan         object
contact      object
day           int64
month        object
duration      int64
campaign      int64
pdays         int64
previous      int64
poutcome     object
y            object
dtype: object
```

In [18]:

```
bank_data.describe()
```

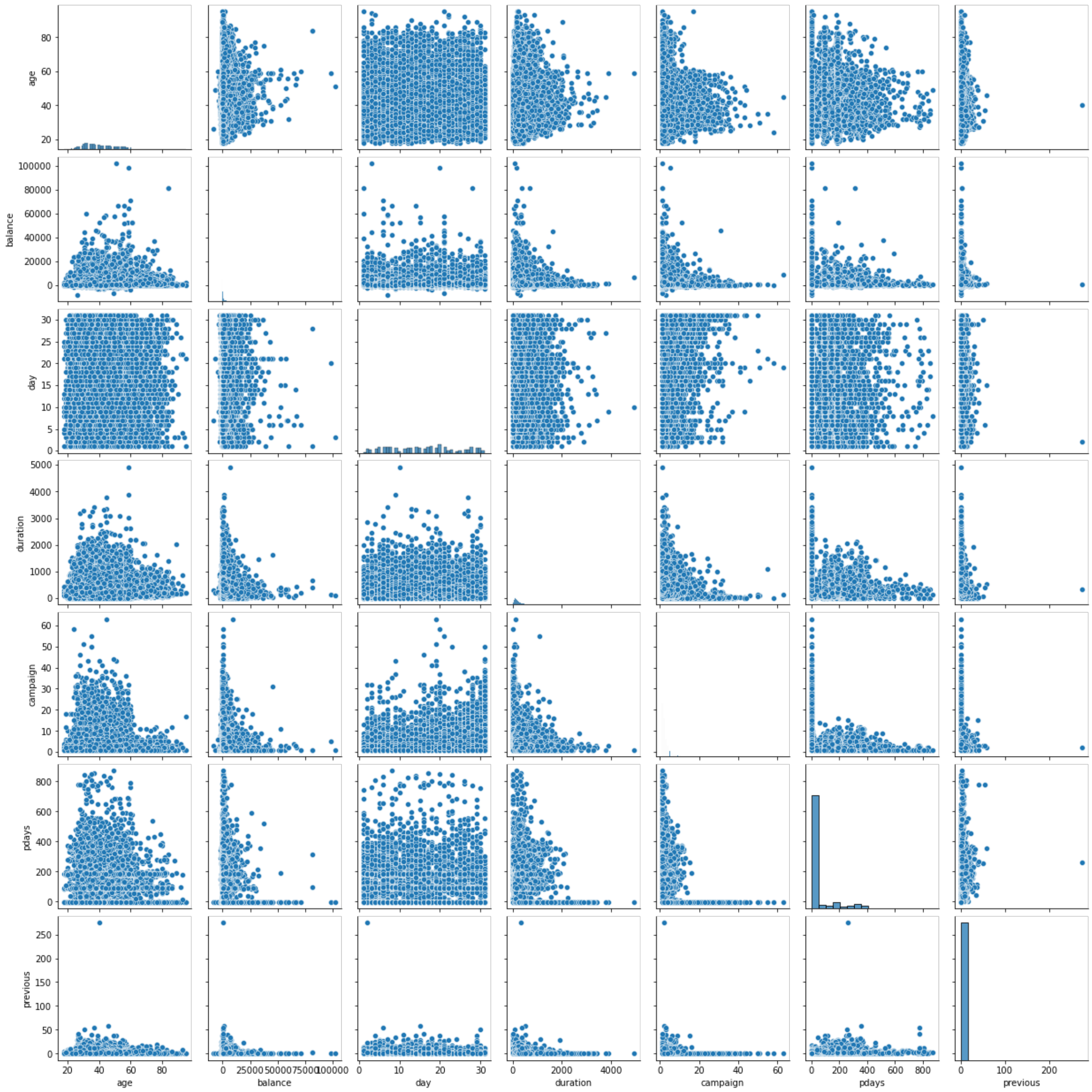
Out[18]:

	age	balance	day	duration	campaign	pdays	
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	452
mean	40.936210	1362.272058	15.806419	258.163080	2.763841	40.197828	
std	10.618762	3044.765829	8.322476	257.527812	3.098021	100.128746	
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.000000	
25%	33.000000	72.000000	8.000000	103.000000	1.000000	-1.000000	
50%	39.000000	448.000000	16.000000	180.000000	2.000000	-1.000000	
75%	48.000000	1428.000000	21.000000	319.000000	3.000000	-1.000000	
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	871.000000	2

In [19]:

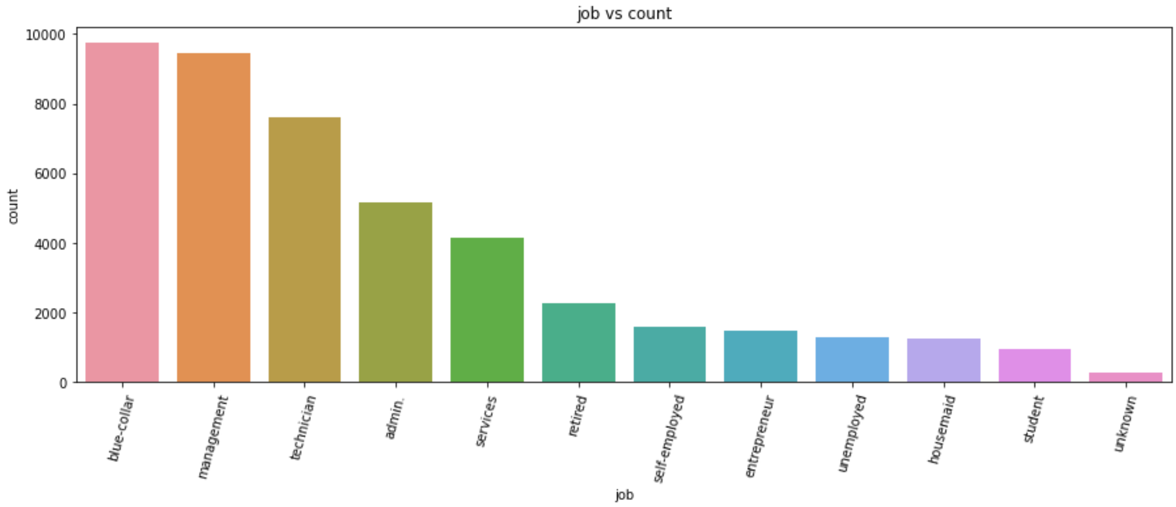
```
sns.pairplot(bank_data)
```

Out[19]: <seaborn.axisgrid.PairGrid at 0x17a0fa4dac0>



In [39]:

```
plt.figure(figsize=(15, 5))
highest_sport = bank_data['job'].value_counts().index
sns.countplot(data=bank_data, x='job', order=highest_sport)
plt.xticks(rotation=75)
plt.title('job vs count')
plt.show()
```



In [41]:

```
bank_data['y'].value_counts()
```

Out[41]:

```
no     39922
yes     5289
Name: y, dtype: int64
```

In [44]:

```
count_no_sub = len(bank_data[bank_data['y']=="no"])
count_sub = len(bank_data[bank_data['y']=="yes"])
```

In [45]:

```
(count_sub / (count_sub + count_no_sub))*100
```

Out[45]:

```
11.698480458295547
```

Percentage of Client Subscribed is 11.70 % in the current data set

In [52]:

```
plt.figure(figsize=(14, 7))
pd.crosstab(bank_data.job,bank_data.y).plot(kind='bar')
plt.title('Subscribed Frequency for Job Title')
plt.xlabel('Job')
plt.ylabel('Frequency of subscribtion')
plt.show()
```

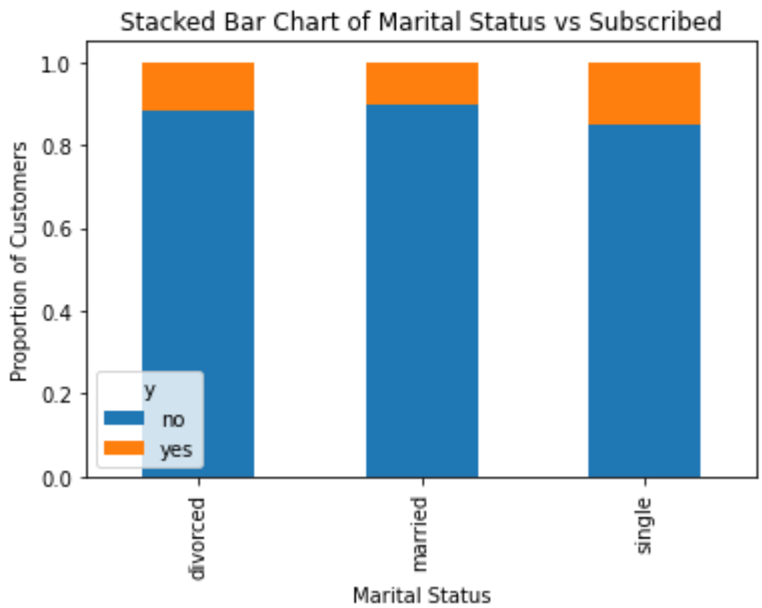
```
<Figure size 1008x504 with 0 Axes>
```

The frequency of subscribtion depends a great deal on the job title. Thus, the job title can be a good predictor of the outcome variable.

In [72]:

```
table=pd.crosstab(bank_data.marital,bank_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 Subscribed')
plt.xlabel('Marital Status')
plt.ylabel('Proportion of Customers')
plt.show()
```

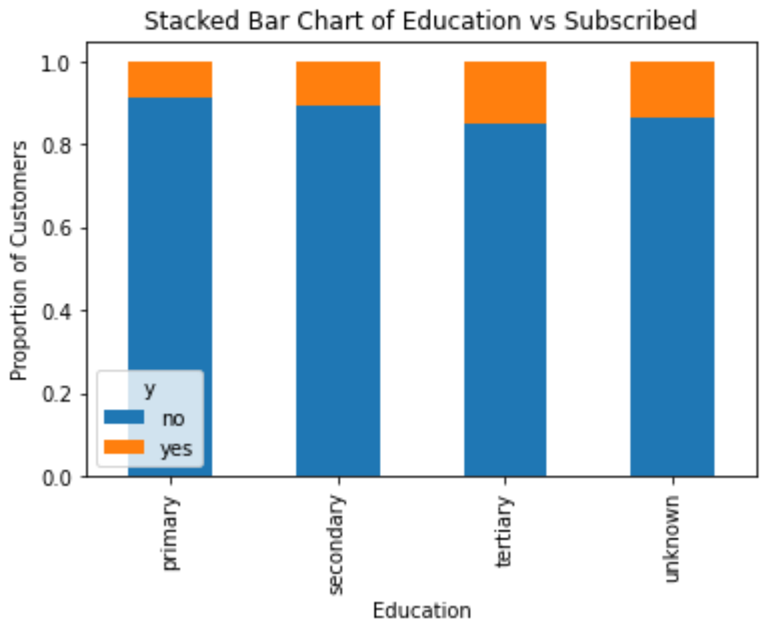
```
<Figure size 1008x504 with 0 Axes>
```



The marital status seem a strong predictor for the outcome variable

In [64]:

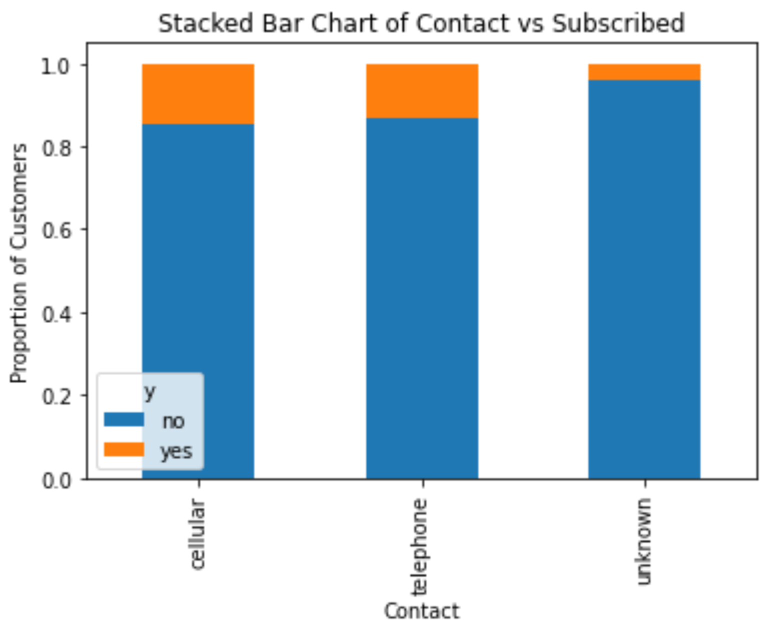
```
table=pd.crosstab(bank_data.education,bank_data.y)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True
plt.title('Stacked Bar Chart of Education vs Subscribed')
plt.xlabel('Education')
plt.ylabel('Proportion of Customers')
plt.show()
```



Education seem a strong predictor for the outcome variable

In [63]:

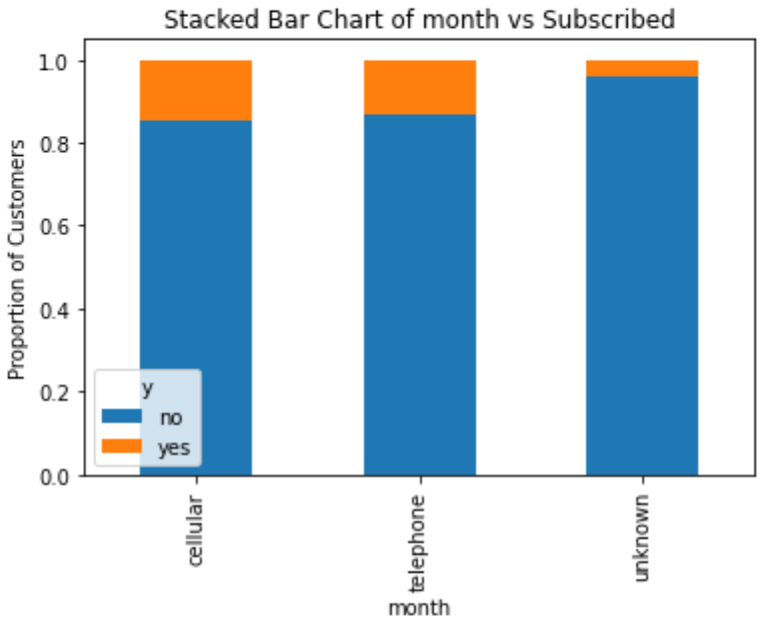
```
table=pd.crosstab(bank_data.contact,bank_data.y)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True
plt.title('Stacked Bar Chart of Contact vs Subscribed')
plt.xlabel('Contact')
plt.ylabel('Proportion of Customers')
plt.show()
```



Contact does not seem a strong predictor for the outcome variable

In [62]:

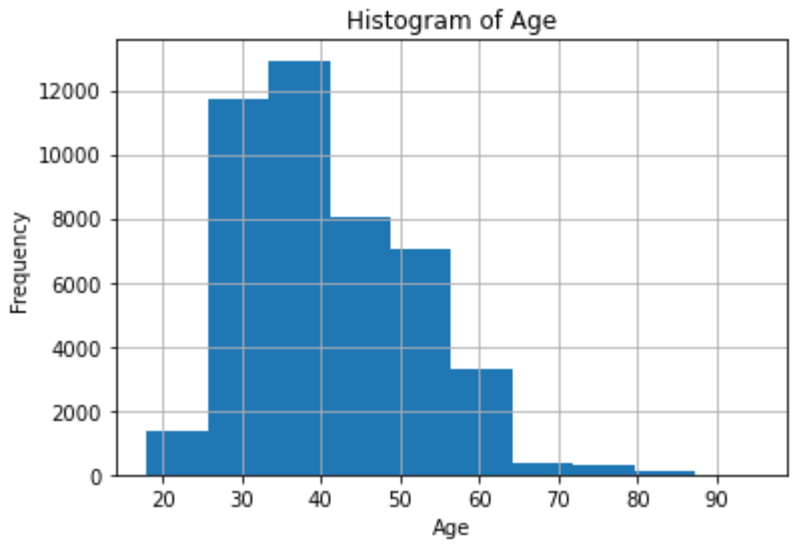
```
table=pd.crosstab(bank_data.contact,bank_data.y)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True
plt.title('Stacked Bar Chart of month vs Subscribed')
plt.xlabel('month')
plt.ylabel('Proportion of Customers')
plt.show()
```



Month might be a good predictor of the outcome variable

In [61]:

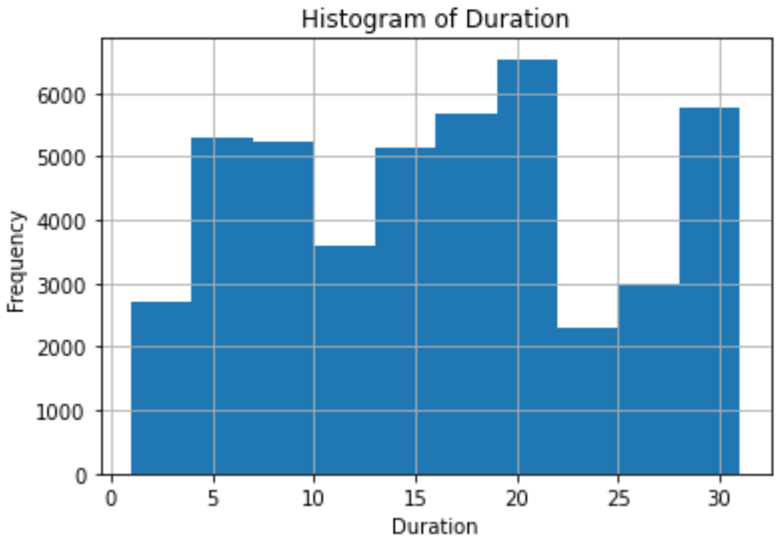
```
bank_data.age.hist()
plt.title('Histogram of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



Most of the customers are in age between 20 and 50 years

In [66]:

```
bank_data.day.hist()
plt.title('Histogram of Duration')
plt.xlabel('Duration')
plt.ylabel('Frequency')
plt.show()
```



In [67]:

```
bank_data['housing'].value_counts()
```

Out[67]:

```
yes    25130
no     20081
Name: housing, dtype: int64
```

In [68]:

```
bank_data['loan'].value_counts()
```

Out[68]:

```
no     37967
yes     7244
Name: loan, dtype: int64
```

Cleaning Data

In [73]:

```
bank_data.isnull().sum()
```

Out[73]:

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

Logistic Regression Model

In [76]:
```
bank_data ['housing'] = bank_data ['housing'].map({'yes': 1, 'no': 0})
```

In [77]:
```
bank_data ['default'] = bank_data ['default'].map({'yes': 1, 'no': 0})
```

In [78]:
```
bank_data ['loan'] = bank_data ['loan'].map({'yes': 1, 'no': 0})
```

In [79]:
```
bank_data ['y'] = bank_data ['y'].map({'yes': 1, 'no': 0})
```

In [80]:
```
bank_data = pd.get_dummies(bank_data, columns=['job'])
```

In [81]:
```
bank_data = pd.get_dummies(bank_data, columns=['marital'])
```

In [82]:
```
bank_data = pd.get_dummies(bank_data, columns=['education'])
```

In [83]:
```
bank_data = pd.get_dummies(bank_data, columns=['month'])
```

In [84]:
```
bank_data = bank_data.drop(['contact','poutcome'],axis=1)
```

In [88]:
```
X = bank_data.loc[:, bank_data.columns != 'y']
y = bank_data.loc[:, bank_data.columns == 'y']
```

In [93]:
```
logreg = LogisticRegression()
rfe = RFE(logreg, 20)
rfe = rfe.fit(X, y.values.ravel())
print(rfe.support_)
print(rfe.ranking_)
```

```
[False  True False  True  True False False False False False False False
 False  True False  True False False  True False False False False  True
 False  True False False  True  True  True  True False  True  True  True
  True  True  True  True  True]
[19  1 22  1  1 18 20  9 21 10 17  7  4  1 14  1 13 12  1 15  5  3  8  1
 16  1  2 11  1  1  1  1  6  1  1  1  1  1  1  1  1]
```

As per Recursive Feature Elimination (RFE) analysis we can exclude all the variables which are False

In [94]:
```
X = bank_data[['default', 'housing', 'loan', 'job_housemaid', 'job_retired
y = bank_data.loc[:, bank_data.columns == 'y']
```

In [95]:
```
logit=sm.Logit(y,X)
result = logit.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.329035
         Iterations 7
```

In [96]:

```python
result.summary()
```

Out[96]:

Logit Regression Results

Dep. Variable:	y	**No. Observations:**	45211
Model:	Logit	**Df Residuals:**	45191
Method:	MLE	**Df Model:**	19
Date:	Wed, 23 Feb 2022	**Pseudo R-squ.:**	0.08823
Time:	16:18:01	**Log-Likelihood:**	-14876.
converged:	True	**LL-Null:**	-16315.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
default	-0.3787	0.147	-2.583	0.010	-0.666	-0.091
housing	-0.8781	0.032	-27.113	0.000	-0.942	-0.815
loan	-0.5724	0.052	-11.078	0.000	-0.674	-0.471
job_housemaid	-0.3289	0.107	-3.063	0.002	-0.539	-0.118
job_retired	0.4625	0.060	7.766	0.000	0.346	0.579
job_student	0.3142	0.083	3.793	0.000	0.152	0.477
marital_married	-0.4290	0.030	-14.327	0.000	-0.488	-0.370
education_primary	-0.4093	0.050	-8.128	0.000	-0.508	-0.311
education_unknown	-0.1917	0.076	-2.526	0.012	-0.340	-0.043
month_aug	-1.6473	0.044	-37.054	0.000	-1.734	-1.560
month_dec	0.2455	0.143	1.721	0.085	-0.034	0.525
month_feb	-1.0118	0.056	-18.069	0.000	-1.122	-0.902
month_jan	-1.6548	0.091	-18.198	0.000	-1.833	-1.477
month_jul	-1.4992	0.048	-31.550	0.000	-1.592	-1.406
month_jun	-1.4979	0.050	-30.193	0.000	-1.595	-1.401
month_mar	0.5074	0.097	5.253	0.000	0.318	0.697
month_may	-1.5674	0.044	-35.900	0.000	-1.653	-1.482
month_nov	-1.4004	0.057	-24.622	0.000	-1.512	-1.289
month_oct	0.1817	0.079	2.290	0.022	0.026	0.337
month_sep	0.2701	0.088	3.053	0.002	0.097	0.443

All variables have significant p value

In [97]:

```
logreg.fit(X, y)
```

Out[97]:

```
LogisticRegression()
```

In [99]:

```
y_pred = logreg.predict(X)
print('Accuracy of logistic regression classifier on test set: {:.2f}'.forr
```

```
Accuracy of logistic regression classifier on test set: 0.88
```

In [100…

```
print(classification_report(y, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.99      0.94     39922
           1       0.49      0.09      0.15      5289

    accuracy                           0.88     45211
   macro avg       0.69      0.54      0.54     45211
weighted avg       0.84      0.88      0.84     45211
```

In [101…

```
confusion_matrix(y, y_pred)
```

Out[101…

```
array([[39455,   467],
       [ 4833,   456]], dtype=int64)
```

1 - Confusion Matrix The result is telling us that we have 39455+456 correct predictions and 4833+467 incorrect predictions.

2 - Accuracy == 84% Of the entire data set, 84% of the clients will subcribe

In [ ]:

```
```