Attribute information For bank dataset

Input variables:

bank client data:
1 - age (numeric)
2 - job : type of job (categorical: "admin.","unknown","unemployed","management","housemaid","entrepreneur","student", "blue-collar","self-employed","retired","technician","services")
3 - marital : marital status (categorical: "married","divorced","single"; note: "divorced" means divorced or widowed)
4 - education (categorical: "unknown","secondary","primary","tertiary")
5 - default: has credit in default? (binary: "yes","no")
6 - balance: average yearly balance, in euros (numeric)
7 - housing: has housing loan? (binary: "yes","no")
8 - loan: has personal loan? (binary: "yes","no")

related with the last contact of the current campaign:

9 - contact: contact communication type (categorical: "unknown","telephone","cellular")
10 - day: last contact day of the month (numeric)
11 - month: last contact month of year (categorical: "jan", "feb", "mar", ..., "nov", "dec")
12 - duration: last contact duration, in seconds (numeric)

other attributes:

13 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
14 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted)
15 - previous: number of contacts performed before this campaign and for this client (numeric)
16 - poutcome: outcome of the previous marketing campaign (categorical: "unknown","other","failure","success")

Output variable (desired target):

17 - y - has the client subscribed a term deposit? (binary: "yes","no")

8. Missing Attribute Values: None

In [33]:
```
# Importing the Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
bank_data=pd.read_csv('bank-full.csv', sep = ';')
bank_data.head()
```

Out[3]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previou
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	

EDA

In [4]:
```
bank_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        45211 non-null  int64 
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64 
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64 
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64 
 12  campaign   45211 non-null  int64 
 13  pdays      45211 non-null  int64 
 14  previous   45211 non-null  int64 
 15  poutcome   45211 non-null  object
 16  y          45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

In [5]:
```
bank_data.dtypes
```

Out[5]:
```
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 [6]:
```
bank_data.describe()
```

Out[6]:

	age	balance	day	duration	campaign	pdays	previous
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000
mean	40.936210	1362.272058	15.806419	258.163080	2.763841	40.197828	0.580323
std	10.618762	3044.765829	8.322476	257.527812	3.098021	100.128746	2.303441
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.000000	0.000000
25%	33.000000	72.000000	8.000000	103.000000	1.000000	-1.000000	0.000000
50%	39.000000	448.000000	16.000000	180.000000	2.000000	-1.000000	0.000000
75%	48.000000	1428.000000	21.000000	319.000000	3.000000	-1.000000	0.000000
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	871.000000	275.000000

In [7]:
```
bank_data.isna().sum()
```

Out[7]:
```
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 [8]:
```
bank_data.nunique()
```

Out[8]:
```
age            77
job            12
marital         3
education       4
default         2
balance      7168
housing         2
loan            2
contact         3
day            31
month          12
duration     1573
campaign       48
pdays         559
previous       41
poutcome        4
y               2
dtype: int64
```

Data Preprocessing & Visualization

In [9]:
```
# Renaming target variable 'y' to 'Deposit' and moving it to the first position
dep = bank_data['y']
#Drop the deposit column
bank_data.drop(labels=['y'], axis=1,inplace = True)
bank_data.insert(0, 'Deposit', dep)
bank_data.head()
```

Out[9]:

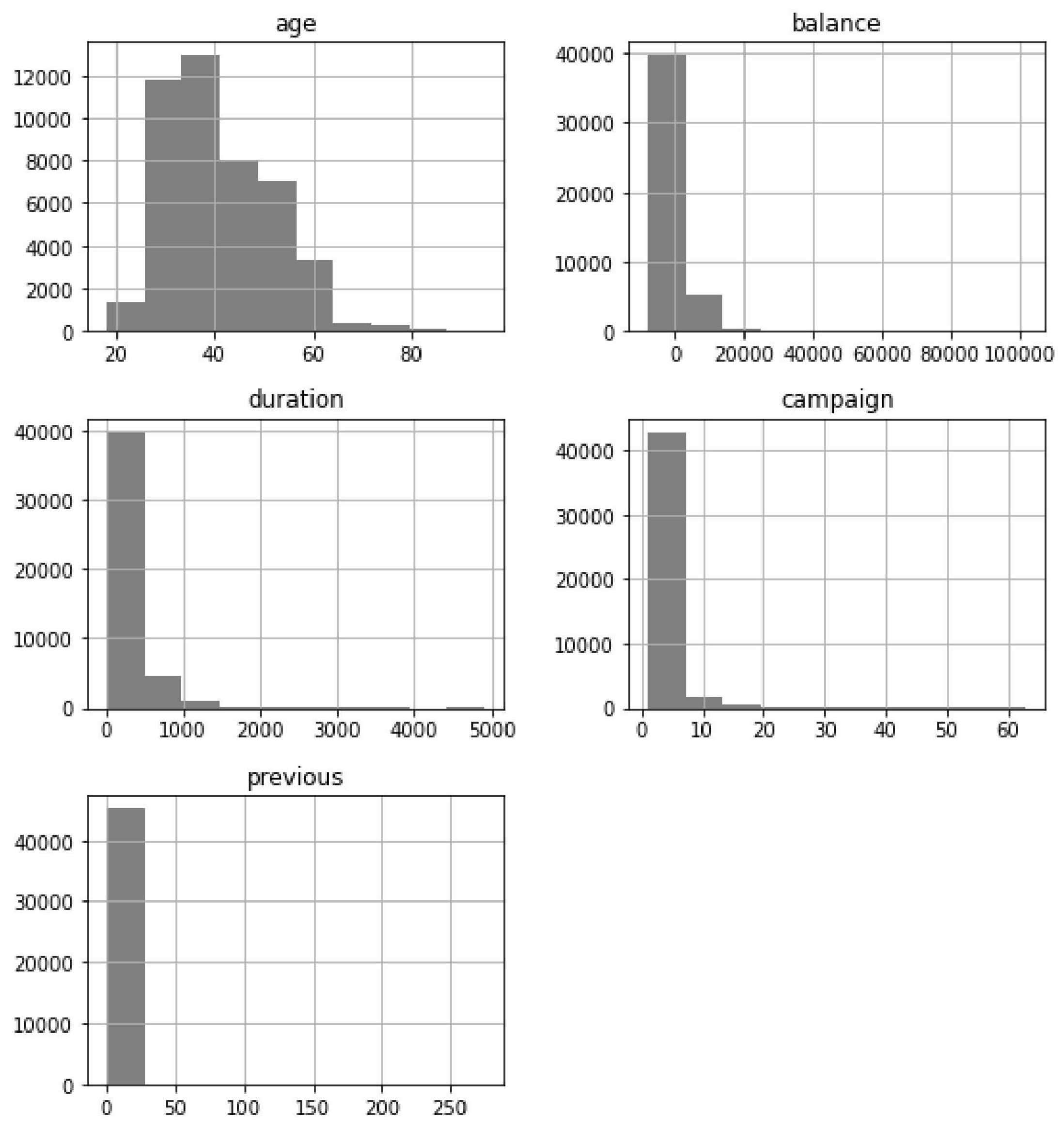
	Deposit	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays
0	no	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1
1	no	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1
2	no	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1
3	no	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1
4	no	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1

In [10]:
```
bank_data['Deposit'].value_counts()
```

Out[10]:
```
no     39922
yes     5289
Name: Deposit, dtype: int64
```

In [12]:
```
# Plotting numeric data distribution.

bank_data.hist(bins=10, figsize=(14,10), color='grey')
plt.show()
```



In [14]:
```
# One-Hot Encoding of categrical variables
data=pd.get_dummies(bank_data,columns=['job','marital','education','contact','poutcome'])
pd.set_option("display.max.columns", None)
data.head()
```

Out[14]:

wn	contact_cellular	contact_telephone	contact_unknown	poutcome_failure	poutcome_other	poutcome_success	poutcome_unknown
0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	1
1	0	0	1	0	0	0	1
1	0	0	1	0	0	0	1

In [15]:
```
# Convert the columns that contain a Yes or No. (Binary Columns)
def convert_to_int(data, new_column, target_column):
    data[new_column] = data[target_column].apply(lambda x: 0 if x == 'no' else 1)
    return data[new_column].value_counts()
```

In [16]:
```
convert_to_int(data, "deposit_int", "Deposit") #Create a deposit int
convert_to_int(data, "housing_int", "housing") # Create housingint column
convert_to_int(data, "loan_int", "loan") #Create a loan_int column
convert_to_int(data, "default_int", "default") #Create a default_int column
```

Out[16]:
```
0    44396
1      815
Name: default_int, dtype: int64
```

In [17]:
```
# Drop the binary columns and leave the same column in the form of integers 0 = No and 1 = Yes
data.drop(['housing', 'loan', 'default'], axis=1, inplace=True)
```

In [18]:
```
# Find and Replace Encoding for month categorical varaible
data['month'].value_counts()
```

Out[18]:
```
may    13766
jul     6895
aug     6247
jun     5341
nov     3970
apr     2932
feb     2649
jan     1403
oct      738
sep      579
mar      477
dec      214
Name: month, dtype: int64
```

In [19]:
```
order={'month':{'jan':1,'feb':2,'mar':3,'apr':4,'may':5,'jun':6,'jul':7,'aug':8,'sep':9,'oct':10,'nov':11,'d
```

In [20]:
```
data=data.replace(order)
data.head()
```

Out[20]:

	Deposit	age	balance	day	month	duration	campaign	pdays	previous	job_admin.	job_blue-collar	job_entrepreneur	job_housemai
0	no	58	2143	5	5	261	1	-1	0	0	0	0	(
1	no	44	29	5	5	151	1	-1	0	0	0	0	(
2	no	33	2	5	5	76	1	-1	0	0	0	1	(
3	no	47	1506	5	5	92	1	-1	0	0	1	0	(
4	no	33	1	5	5	198	1	-1	0	0	0	0	(

In [21]:
```
data.drop(['Deposit'], axis=1, inplace=True)
# Rename deposit_int column for Deposit and then move it to the first
data = data.rename(columns={"deposit_int": "deposit"})
first = data['deposit']
data.drop(labels=['deposit'], axis=1,inplace = True)
# insert (loc, column, values) --> loc is the same as position in the column.
data.insert(0, 'deposit', first)
data["deposit"].value_counts()
```

Out[21]:
```
0    39922
1     5289
Name: deposit, dtype: int64
```

In [22]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 38 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   deposit              45211 non-null  int64
 1   age                  45211 non-null  int64
 2   balance              45211 non-null  int64
 3   day                  45211 non-null  int64
 4   month                45211 non-null  int64
 5   duration             45211 non-null  int64
 6   campaign             45211 non-null  int64
 7   pdays                45211 non-null  int64
 8   previous             45211 non-null  int64
 9   job_admin.           45211 non-null  uint8
 10  job_blue-collar      45211 non-null  uint8
 11  job_entrepreneur     45211 non-null  uint8
 12  job_housemaid        45211 non-null  uint8
 13  job_management       45211 non-null  uint8
 14  job_retired          45211 non-null  uint8
 15  job_self-employed    45211 non-null  uint8
 16  job_services         45211 non-null  uint8
 17  job_student          45211 non-null  uint8
 18  job_technician       45211 non-null  uint8
 19  job_unemployed       45211 non-null  uint8
 20  job_unknown          45211 non-null  uint8
 21  marital_divorced     45211 non-null  uint8
 22  marital_married      45211 non-null  uint8
 23  marital_single       45211 non-null  uint8
 24  education_primary    45211 non-null  uint8
 25  education_secondary  45211 non-null  uint8
 26  education_tertiary   45211 non-null  uint8
 27  education_unknown    45211 non-null  uint8
 28  contact_cellular     45211 non-null  uint8
 29  contact_telephone    45211 non-null  uint8
 30  contact_unknown      45211 non-null  uint8
 31  poutcome_failure     45211 non-null  uint8
 32  poutcome_other       45211 non-null  uint8
 33  poutcome_success     45211 non-null  uint8
 34  poutcome_unknown     45211 non-null  uint8
 35  housing_int          45211 non-null  int64
 36  loan_int             45211 non-null  int64
 37  default_int          45211 non-null  int64
dtypes: int64(12), uint8(26)
memory usage: 5.3 MB
```

Model Building

In [23]:
```
# Dividing our data into input and output variables
X=pd.concat([data.iloc[:,1:]],axis=1)
y=data.iloc[:,0:1]
```

In [28]:
```
# Logistic regression model
classifier=LogisticRegression()
classifier.fit(X,y)
```

Out[28]:
```
LogisticRegression()
```

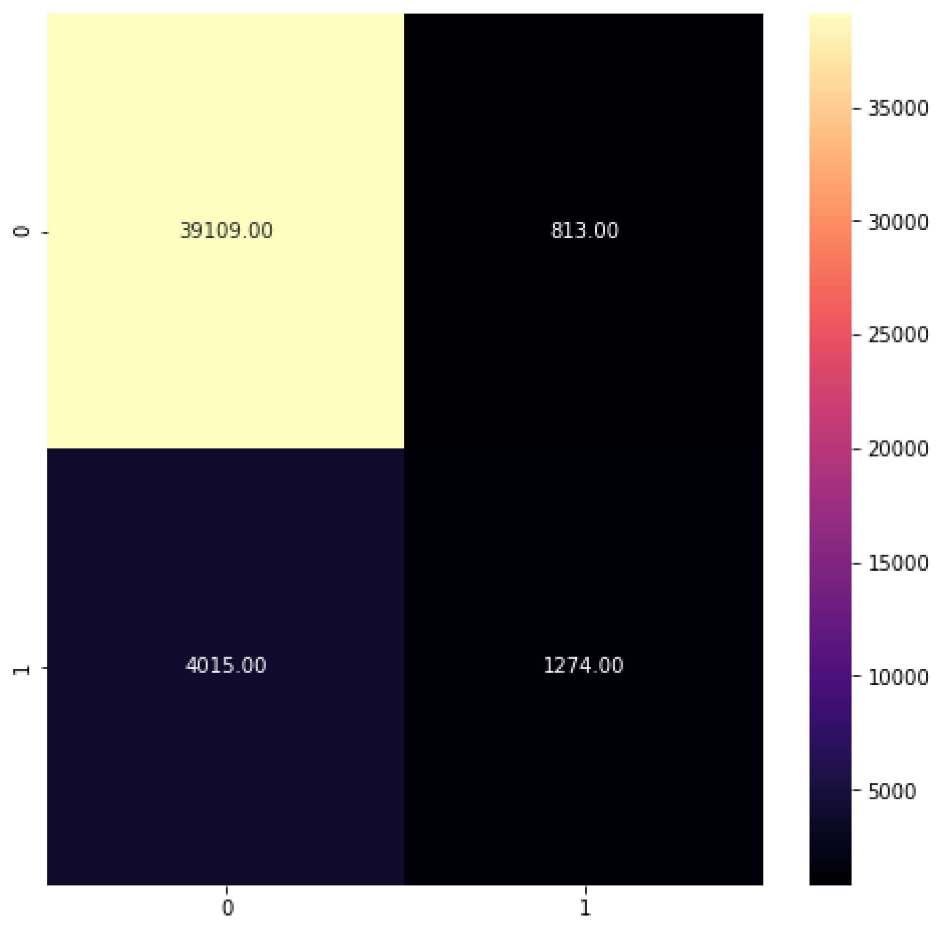
Model prediction and evaluation

In [25]:
```
# Predict for x dataset
y_pred=classifier.predict(X)
y_pred
```

Out[25]:
```
array([0, 0, 0, ..., 1, 0, 0], dtype=int64)
```

In [34]:
```
confusion_matrix = confusion_matrix(y,y_pred)
confusion_matrix
fig, ax = plt.subplots(figsize=(8, 8))
sns.heatmap(confusion_matrix, cmap='magma', annot=True, fmt=".2f")
```

Out[34]:
```
<AxesSubplot:>
```



In [35]:
```
confusion_matrix
```

Out[35]:
```
array([[39109,   813],
       [ 4015,  1274]], dtype=int64)
```

In [36]:
```
# The model accuracy is calculated by (a+d)/(a+b+c+d)
(39109+1274)/(39109+813+4015+1274)
```

Out[36]:
```
0.8932118289796731
```

Model accuracy is : 89.31%

In [37]:
```
classifier.predict_proba(X)[:,1]
```

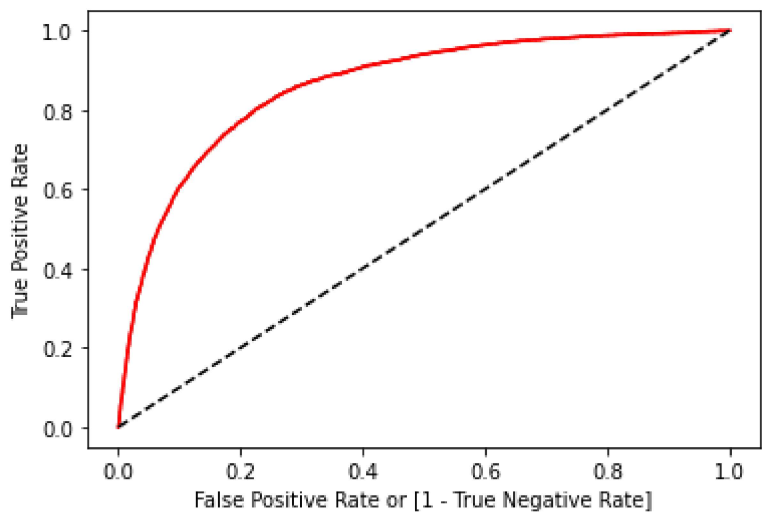
Out[37]:
```
array([0.04534893, 0.01988662, 0.0116982 , ..., 0.83260083, 0.07658997,
       0.11785517])
```

In [38]:

```
# ROC Curve plotting and finding AUC value
fpr,tpr,thresholds=roc_curve(y,classifier.predict_proba(X)[:,1])
plt.plot(fpr,tpr,color='red')
auc=roc_auc_score(y,y_pred)

plt.plot(fpr,tpr,color='red',label='logit model(area  = %0.2f)'%auc)
plt.plot([0,1],[0,1],'k--')
plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
plt.ylabel('True Positive Rate')
plt.show()

print('auc accuracy:',auc)
```



```
auc accuracy: 0.6102562906535205
```