

Importing basic python modules to extract and visualize dataset.

```
import os
import pandas as pd
import numpy as np
import warnings
```

Reading the dataset using pandas.

```
bank = pd.read_csv('bank.csv', sep = ',')
```

Gathering information about Dataset by inspecing it.

```
bank.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day
0	59	admin.	married	secondary	no	2343	yes	no	unknown	5
1	56	admin.	married	secondary	no	45	no	no	unknown	5
2	41	technician	married	secondary	no	1270	yes	no	unknown	5
3	55	services	married	secondary	no	2476	yes	no	unknown	5
4	54	admin.	married	tertiary	no	184	no	no	unknown	5

```
bank.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11162 entries, 0 to 11161
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype  
 --- 
 0   age         11162 non-null   int64  
 1   job          11162 non-null   object 
 2   marital     11162 non-null   object 
 3   education   11162 non-null   object 
 4   default     11162 non-null   object 
 5   balance     11162 non-null   int64  
 6   housing     11162 non-null   object 
 7   loan         11162 non-null   object 
 8   contact     11162 non-null   object 
 9   day          11162 non-null   int64  
 10  month        11162 non-null   object 
 11  duration    11162 non-null   int64  
 12  campaign    11162 non-null   int64  
 13  pdays       11162 non-null   int64  
 14  previous    11162 non-null   int64  
 15  poutcome    11162 non-null   object 
 16  deposit     11162 non-null   object
```

```
dtypes: int64(7), object(10)
memory usage: 1.4+ MB
```

We observe that in the dataset there are **17 attributes** in total out of which **deposit** is the dependent Y variable on all rest 16 X variables. The values of deposit is either Yes or No. so later it would needed to be converted to 0 for No and 1 for Yes.

Shape of Dataset is:

```
bank.shape
```

```
(11162, 17)
```

overall description of dataset:

```
bank.describe()
```

	age	balance	day	duration	campaign	pday
count	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000
mean	41.231948	1528.538524	15.658036	371.993818	2.508421	51.330400
std	11.913369	3225.413326	8.420740	347.128386	2.722077	108.758280
min	18.000000	-6847.000000	1.000000	2.000000	1.000000	-1.000000
25%	32.000000	122.000000	8.000000	138.000000	1.000000	-1.000000
50%	39.000000	550.000000	15.000000	255.000000	2.000000	-1.000000
75%	49.000000	1708.000000	22.000000	496.000000	3.000000	20.750000
max	95.000000	81204.000000	31.000000	3881.000000	63.000000	854.000000

Let's figure out the number of unique values assigned to all attributs in dataset of 'object' type.

```
for col in bank.select_dtypes(include='object').columns:
    print(col)
    print(bank[col].unique())
    print('\n')

    job
    ['admin.' 'technician' 'services' 'management' 'retired' 'blue-collar'
     'unemployed' 'entrepreneur' 'housemaid' 'unknown' 'self-employed'
     'student']

    marital
    ['married' 'single' 'divorced']
```

```
education
['secondary' 'tertiary' 'primary' 'unknown']

default
['no' 'yes']

housing
['yes' 'no']

loan
['no' 'yes']

contact
['unknown' 'cellular' 'telephone']

month
['may' 'jun' 'jul' 'aug' 'oct' 'nov' 'dec' 'jan' 'feb' 'mar' 'apr' 'sep']

poutcome
['unknown' 'other' 'failure' 'success']

deposit
['yes' 'no']
```

▼ Exploratory Data Analysis

- Find Unwanted Columns
- Find Missing Values
- Find Features with one value
- Explore the Categorical Features
- Find Categorical Feature Distribution
- Relationship between Categorical Features and Label
- Explore the Numerical Features
- Find Discrete Numerical Features
- Relation between Discrete numerical Features and Labels
- Find Continous Numerical Features
- Distribution of Continous Numerical Features
- Relation between Continous numerical Features and Labels
- Find Outliers in numerical features
- Explore the Correlation between numerical features

- Find Pair Plot
- Check the Data set is balanced or not based on target values in classification

```
df=bank
```

1. Find Unwanted Columns

Take-away:

- there is no unwanted column present in given dataset to remove

2. Find Missing Values

```
features_na = [feature for feature in bank.columns if bank[feature].isnull().sum() > 0]
for feature in features_na:
    print(feature, np.round(bank[feature].isnull().mean(), 4), ' % missing values')
else:
    print("No missing value found")

No missing value found
```

Take-away:

- No missing value found

3. Find Features with One Value

```
for column in df.columns:
    print(column, df[column].nunique())

age 76
job 12
marital 3
education 4
default 2
balance 3805
housing 2
loan 2
contact 3
day 31
month 12
duration 1428
campaign 36
pdays 472
previous 34
poutcome 4
deposit 2
```

Take-away:

- No feature with only one value

4. Explore the Categorical Features

```
categorical_features=[features for features in df.columns if ((df[features].dtype=='O') &
categorical_features

['job',
'marital',
'education',
'default',
'housing',
'loan',
'contact',
'month',
'poutcome']]

for feature in categorical_features:
    print('The feature is {} and number of categories are {}'.format(feature,df[feature].n

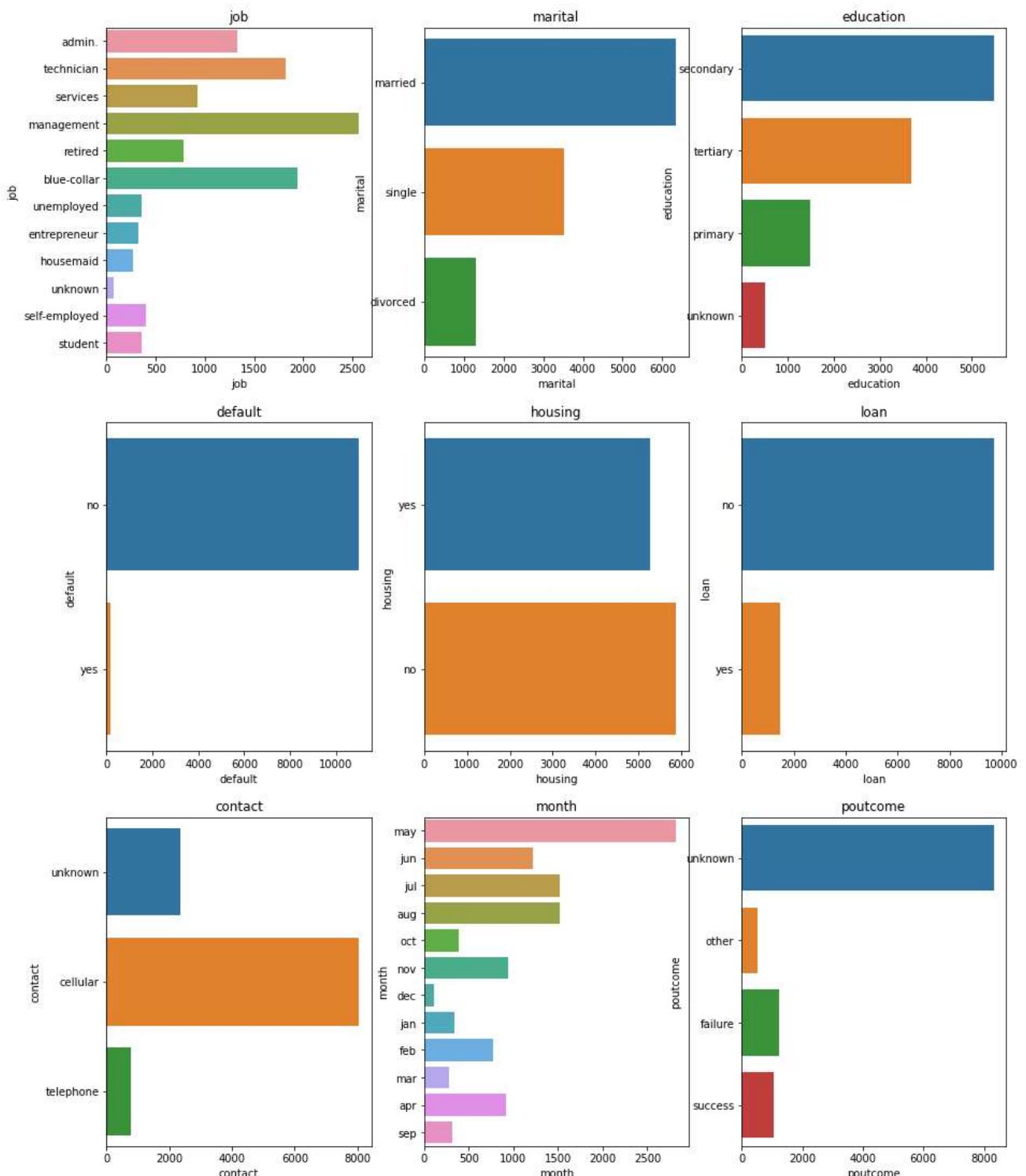
The feature is job and number of categories are 12
The feature is marital and number of categories are 3
The feature is education and number of categories are 4
The feature is default and number of categories are 2
The feature is housing and number of categories are 2
The feature is loan and number of categories are 2
The feature is contact and number of categories are 3
The feature is month and number of categories are 12
The feature is poutcome and number of categories are 4
```

Take-away:

- there are 9 categorical features
- feature job and month has highest number of categorical values

5. Find Categorical Feature Distribution

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(15,80),facecolor='white')
plotnumber=1
for categorical_feature in categorical_features:
    ax = plt.subplot(12,3,plotnumber)
    sns.countplot(y=categorical_feature,data=df)
    plt.xlabel(categorical_feature)
    plt.title(categorical_feature)
    plotnumber+=1
plt.show()
```



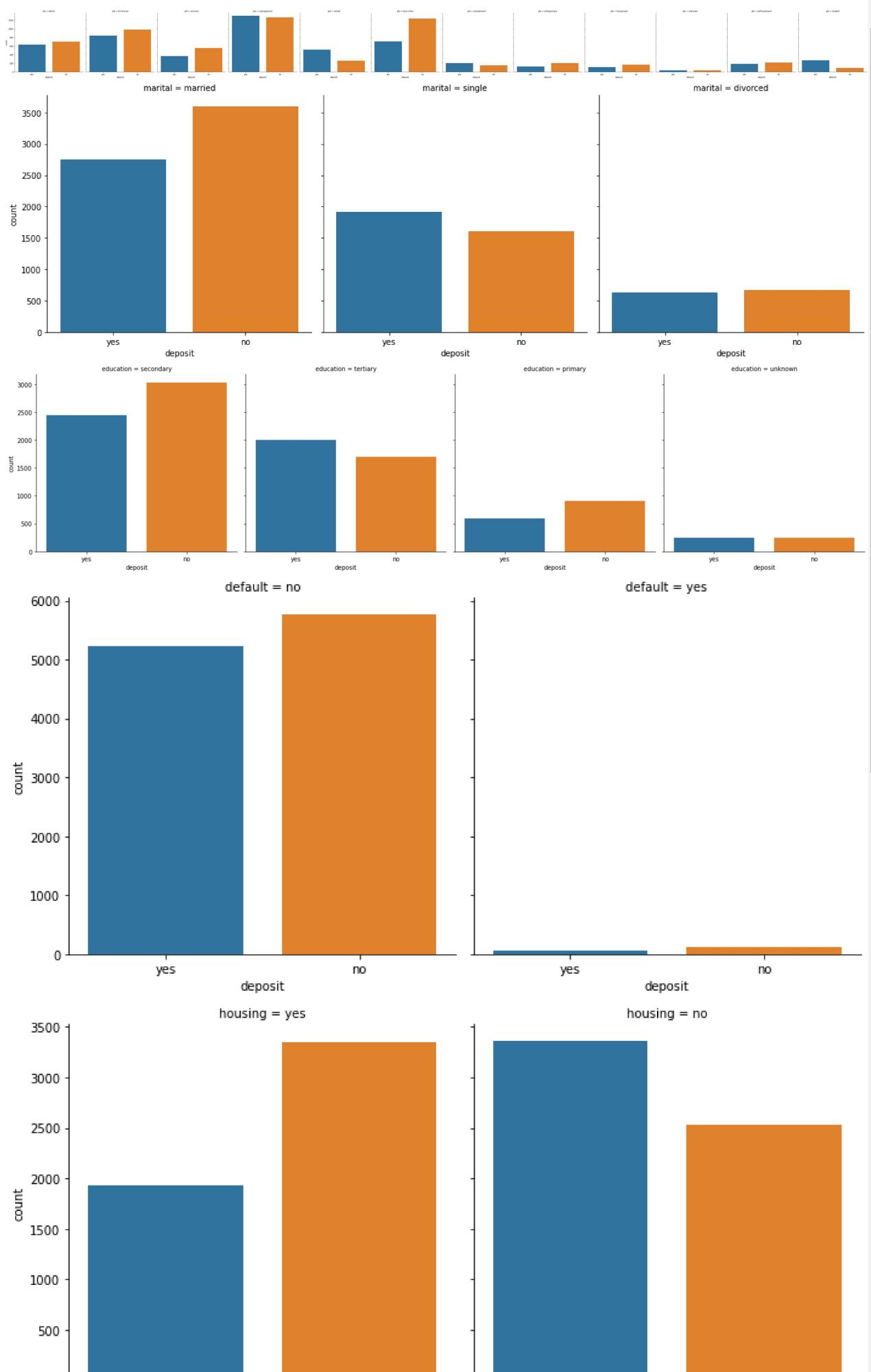
Take-away:

- client with job type as management records are high in given dataset and housemaid are very less
- client who married are high in records in given dataset and divorced are less
- client whoes education background is secondary are in high numbers in given dataset
- defualt feature seems to be does not play importand role as it has value of no at high ratio to value yes which can drop
- data in month of may is high and less in dec

6. Relationship between Categorical Features and Label

```
for categorical_feature in categorical_features:  
    sns.catplot(x='deposit', col=categorical_feature, kind='count', data= df)  
plt.show()
```

→



```
#Check target label split over categorical features and find the count
for categorical_feature in categorical_features:
    print(df.groupby(['deposit',categorical_feature]).size())
```

```
deposit  job
no      admin.          703
       blue-collar     1236
       entrepreneur    205
       housemaid       165
       management      1265
       retired         262
       self-employed    218
       services         554
       student          91
       technician       983
       unemployed       155
       unknown          36
yes     admin.          631
       blue-collar     708
       entrepreneur    123
       housemaid       109
       management      1301
       retired         516
       self-employed    187
       services         369
       student          269
       technician       840
       unemployed       202
       unknown          34
dtype: int64
deposit  marital
no      divorced        671
       married         3596
       single          1606
yes     divorced        622
       married         2755
       single          1912
dtype: int64
deposit  education
no      primary          909
       secondary        3026
       tertiary         1693
       unknown          245
yes     primary          591
       secondary        2450
       tertiary         1996
       unknown          252
dtype: int64
deposit  default
no      no              5757
       yes             116
yes     no              5237
       yes             52
dtype: int64
deposit  housing
no      no              2527
       yes             3346
yes     no              3354
       yes             1935
dtype: int64
deposit  loan
no      no              4897
```

Take-away:

- retired client has high interest on deposit
- client who has housing loan seems to be not interested much on deposit
- if pre campagin outcome that is poutcome=success then, there is high chance of client to show interest on deposit
- in month of March, September, October and December, client show high interest to deposit
- in month of may, records are high but client interst ratio is very less

7. Explore the Numerical Features

```
numerical_features = [feature for feature in df.columns if ((df[feature].dtypes != 'O') &
print('Number of numerical variables: ', len(numerical_features))

df[numerical_features].head()
```

Number of numerical variables: 7

	age	balance	day	duration	campaign	pdays	previous
0	59	2343	5	1042	1	-1	0
1	56	45	5	1467	1	-1	0
2	41	1270	5	1389	1	-1	0
3	55	2476	5	579	1	-1	0
4	54	184	5	673	2	-1	0

Take-away:

- there are 7 numerical features

8. Find Discrete Numerical Features

```
discrete_feature=[feature for feature in numerical_features if len(df[feature].unique())<2
print("Discrete Variables Count: {}".format(len(discrete_feature)))
```

Discrete Variables Count: 0

Take-away:

- there is no Discrete Variables in give dataset

9. Relation between Discrete numerical Features and Labels

- NA

10. Find Continous Numerical Features

```
continuous_features=[feature for feature in numerical_features if feature not in discrete_
print("Continuous feature Count {}".format(len(continuous_features)))
```

Continuous feature Count 7

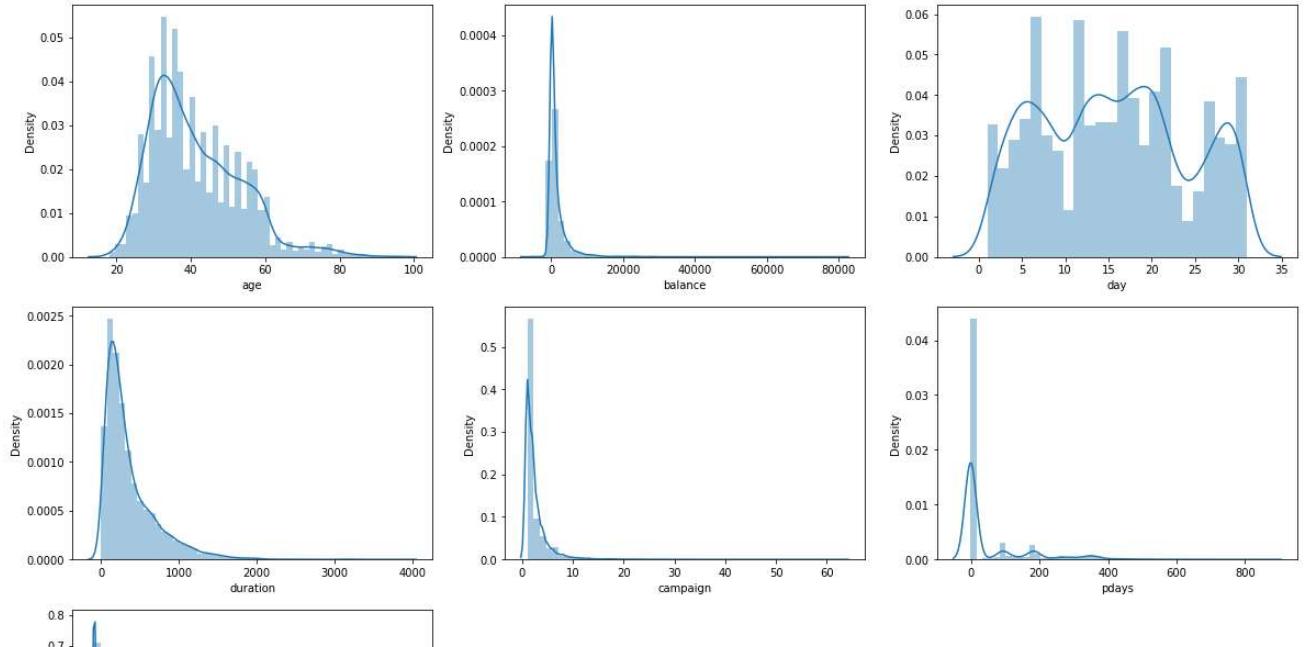
Take-away:

- there are 7 continuous numerical features

11. Distribution of Continous Numerical Features

```
plt.figure(figsize=(20,60), facecolor='white')
plotnumber =1
for continuous_feature in continuous_features:
    ax = plt.subplot(12,3,plotnumber)
    sns.distplot(df[continuous_feature])
    plt.xlabel(continuous_feature)
    plotnumber+=1
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
```

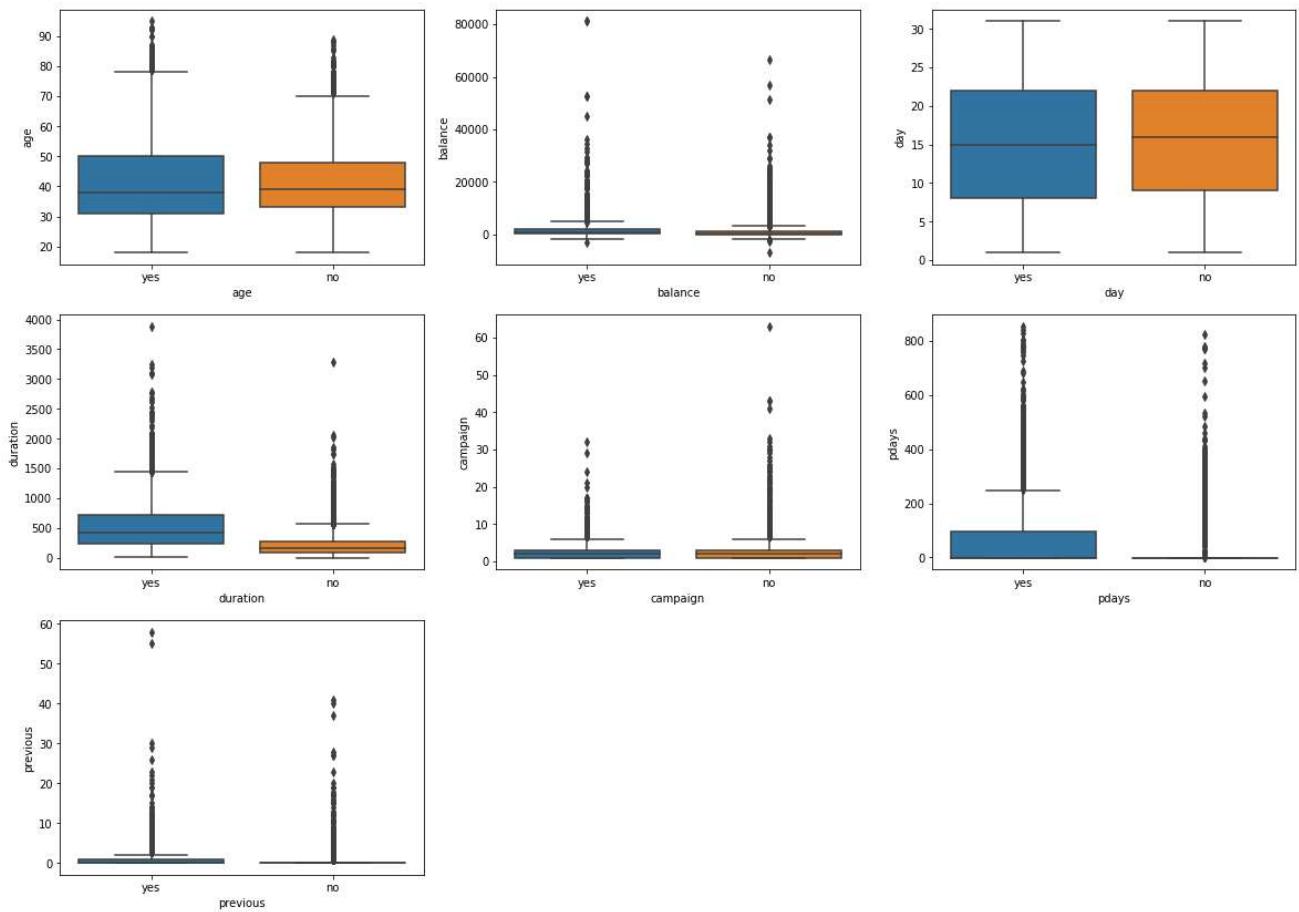


Take-away:

- it seems age, days distributed normally
- balance, duration, compaign, pdays and previous heavily skewed towards left and seems to be have some outliers.

12. Relation between Continous numerical Features and Labels

```
plt.figure(figsize=(20,60), facecolor='white')
plotnumber = 1
for feature in continuous_features:
    ax = plt.subplot(12,3,plotnumber)
    sns.boxplot(x="deposit", y= df[feature], data=df)
    plt.xlabel(feature)
    plotnumber+=1
plt.show()
```



Take-away:

- client shows interest on deposit who had discussion for longer duration

13. Find Outliers in numerical features

```
#boxplot on numerical features to find outliers
plt.figure(figsize=(20,60), facecolor='white')
plotnumber =1
for numerical_feature in numerical_features:
    ax = plt.subplot(12,3,plotnumber)
    sns.boxplot(df[numerical_feature])
```

```
plt.xlabel(numerical_feature)
plotnumber+=1
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
    FutureWarning
```

Take-away:

- age, balance, duration, compaign, pdays and previous has some outliers

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
```

14. Explore the Correlation between numerical features

```
cor_mat=df.corr()
fig = plt.figure(figsize=(15,7))
sns.heatmap(cor_mat, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcc1be2fb90>
```

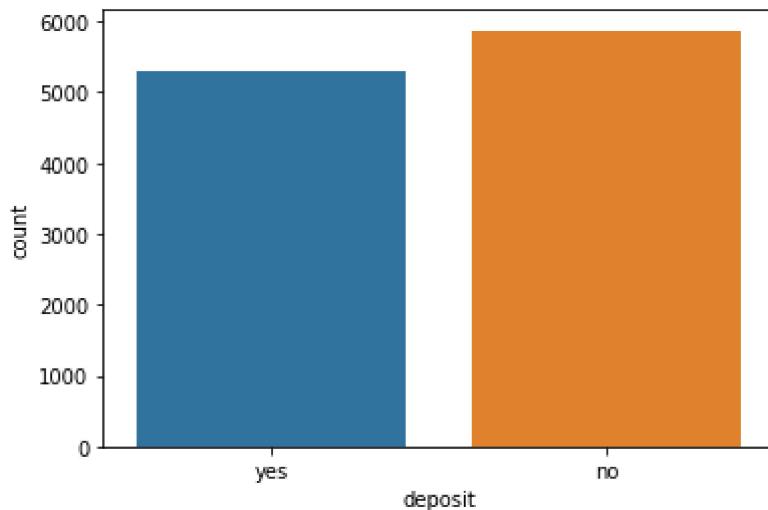


Take-away:

- it seems no feature is heavily correlated with other features

15. Check the Data set is balanced or not based on target values in classification

```
sns.countplot(x='deposit', data=df)
plt.show()
```



```
df['deposit'].groupby(df['deposit']).count()
```

```
deposit
no      5873
yes     5289
Name: deposit, dtype: int64
```

Take-away:

- given dataset seems to be balanced.

▼ Feature Engineering

Removing Outliers in age, balance, duration, compaign, pdays as well as dropping unwanted attributes to improve accuracy of training.

```
df2=df.copy()
```

default features does not play important role

```
df2.groupby(['deposit','default']).size()
```

```
deposit  default
no       no        5757
          yes        116
yes      no        5237
          yes         52
dtype: int64
```

```
df2.drop(['default'],axis=1, inplace=True)
```

```
df2.groupby(['deposit','pdays']).size()
```

```

deposit  pdays
no      -1      4940
       1       2
       2       6
       5       2
       6       2
       ...
yes     804      1
       805      1
       828      1
       842      1
       854      1
Length: 732, dtype: int64

```

drop pdays as it has -1 value for around 40%+

```
df2.drop(['pdays'], axis=1, inplace=True)
```

remove outliers in feature age.

```
df2.groupby('age', sort=True)['age'].count()
```

```

age
18     8
19    13
20    20
21    30
22    48
...
89     1
90     2
92     2
93     2
95     1
Name: age, Length: 76, dtype: int64

```

these can be ignored and values lies in between 18 to 95

remove outliers in feature balance.

```
df2.groupby(['deposit', 'balance'], sort=True)['balance'].count()
```

```

deposit  balance
no      -6847      1
       -2712      1
       -2282      1
       -2049      1
       -1965      1
       ...
yes      34646      1
       36252      1
       45248      1
       52587      2

```

```
81204      2  
Name: balance, Length: 5082, dtype: int64
```

these outlier should not be remove as balance goes high, client show interest on deposit

remove outliers in feature duration.

```
df2.groupby(['deposit','duration'],sort=True)['duration'].count()
```

```
deposit  duration  
no        2          1  
         3          1  
         4          2  
         5          4  
         6          6  
         ..  
yes       3094        1  
        3102        1  
        3183        1  
        3253        1  
        3881        1  
Name: duration, Length: 2157, dtype: int64
```

these outlier should not be remove as duration goes high, client show interest on deposit

remove outliers in feature campaign.

```
df2.groupby(['deposit','campaign'],sort=True)['campaign'].count()
```

```
deposit  campaign  
no        1        2237  
         2        1627  
         3        703  
         4        454  
         5        239  
         6        173  
         7        92  
         8        96  
         9        51  
        10        38  
        11        24  
        12        25  
        13        24  
        14        11  
        15         9  
        16         7  
        17         8  
        18         8  
        19         5  
        20         4  
        21         7  
        22         4
```

	23	3
	24	4
	25	3
	26	3
	27	1
	28	1
	29	1
	30	4
	31	1
	32	1
	33	1
	41	1
	43	2
	63	1
yes	1	2561
	2	1401
	3	618
	4	317
	5	139
	6	92
	7	47
	8	32
	9	21
	10	14
	11	16
	12	4
	13	6
	14	4
	15	4
	16	2
	17	6
	20	1
	21	1
	24	1
	--	

```
df3 = df2[df2['campaign'] < 33]
```

```
df3.groupby(['deposit','campaign'],sort=True)['campaign'].count()
```

deposit	campaign	
no	1	2237
	2	1627
	3	703
	4	454
	5	239
	6	173
	7	92
	8	96
	9	51
	10	38
	11	24
	12	25
	13	24
	14	11
	15	9
	16	7
	17	8
	18	8

	19	5
	20	4
	21	7
	22	4
	23	3
	24	4
	25	3
	26	3
	27	1
	28	1
	29	1
	30	4
	31	1
	32	1
yes	1	2561
	2	1401
	3	618
	4	317
	5	139
	6	92
	7	47
	8	32
	9	21
	10	14
	11	16
	12	4
	13	6
	14	4
	15	4
	16	2
	17	6
	20	1
	21	1
	24	1
	29	1
	32	1

Name: campaign, dtype: int64

remove outliers in feature previous.

```
df3.groupby(['deposit','previous'],sort=True)['previous'].count()
```

deposit	previous	
no		
	0	4935
	1	304
	2	237
	3	141
	4	73
	5	44
	6	34
	7	23
	8	21
	9	10
	10	4
	11	8
	12	6
	13	2
	14	1

```

15          4
16          2
17          8
18          1
19          2
20          1
23          1
27          2
28          1
37          1
40          1
41          1
yes      0    3384
        1    583
        2    456
        3    294
        4    171
        5    121
        6     83
        7     54
        8     39
        9     24
       10     26
       11     15
       12     10
       13     9
       14     5
       15     1
       17     3
       19     2
       20     1
       21     1
       22     1
       23     1
       26     1
       29     1
       30     1
       55     1
       58     1

```

Name: previous, dtype: int64

```
df4 = df3[df3['previous'] < 31]
```

creating dummy categorical features for all features with multiple categorical values

```

cat_columns = ['job', 'marital', 'education', 'contact', 'month', 'poutcome']
for col in cat_columns:
    df4 = pd.concat([df4.drop(col, axis=1), pd.get_dummies(df4[col], prefix=col, prefix_sep='_')])

bool_columns = ['housing', 'loan', 'deposit']
for col in bool_columns:
    df4[col+'_new']=df4[col].apply(lambda x : 1 if x == 'yes' else 0)
    df4.drop(col, axis=1, inplace=True)

```

```
df4.head()
```

	age	balance	day	duration	campaign	previous	job_blue-collar	job_entrepreneur	job...
0	59	2343	5	1042	1	0	0		0
1	56	45	5	1467	1	0	0		0
2	41	1270	5	1389	1	0	0		0
3	55	2476	5	579	1	0	0		0
4	54	184	5	673	2	0	0		0

5 rows × 41 columns



▼ Split Dataset into Training set and Test set

```
X = df4.drop(['deposit_new'],axis=1)
y = df4['deposit_new']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=0)
```

```
len(X_train)
```

```
8921
```

```
len(X_test)
```

```
2231
```

Machine Learning Models and evaluation.

▼ Logistic Regression

```
from sklearn.linear_model import LogisticRegression

logistic_model = LogisticRegression(max_iter=10000)
result = logistic_model.fit(X_train, y_train)
result.score(X_train, y_train)
```

```
0.8253559018047304
```

```
predict = result.predict(X_test)
```

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, predict))
```

0.8265351860152398

```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test, predict)
```

0.8250241069153792

▼ Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X_train,y_train)
rfc.score(X_test,y_test)
```

0.8458090542357687

```
predict_rfc=rfc.predict(X_test)
```

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, predict_rfc))
```

0.8458090542357687

```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test, predict_rfc)
```

0.8471508689777055

▼ XGBoost Classifier

```
from xgboost import XGBClassifier
model_xgb = XGBClassifier(objective='binary:logistic',learning_rate=0.1,max_depth=10,n_estimators=10)
model_xgb.fit(X_train,y_train)
model_xgb.score(X_test,y_test)
```

0.8583594800537876

```
predict_xgb=model_xgb.predict(X_test)
```

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, predict_xgb))
```

```
0.8583594800537876
```

```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test, predict_xgb)
```

```
0.8595885054357465
```

▼ Observations and conclusions

let's see the Confusion Matrix of all the 3 models.

▼ Logistic Regression

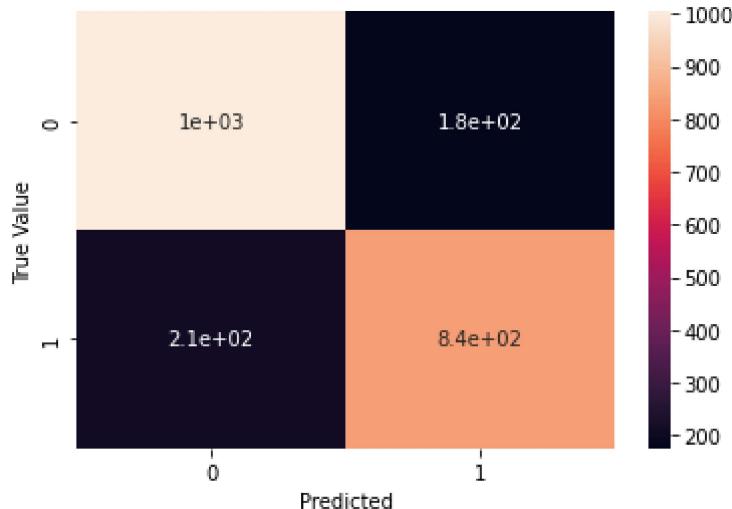
```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test, predict)
```

```
0.8250241069153792
```

```
from sklearn.metrics import confusion_matrix
lr = confusion_matrix(y_test,predict)
lr

array([[1004,  175],
       [ 212,  840]])
```

```
from matplotlib import pyplot as plt
import seaborn as sn
sn.heatmap(lr, annot=True)
plt.xlabel('Predicted')
plt.ylabel('True Value')
plt.show()
```



▼ Random Forest Classifier

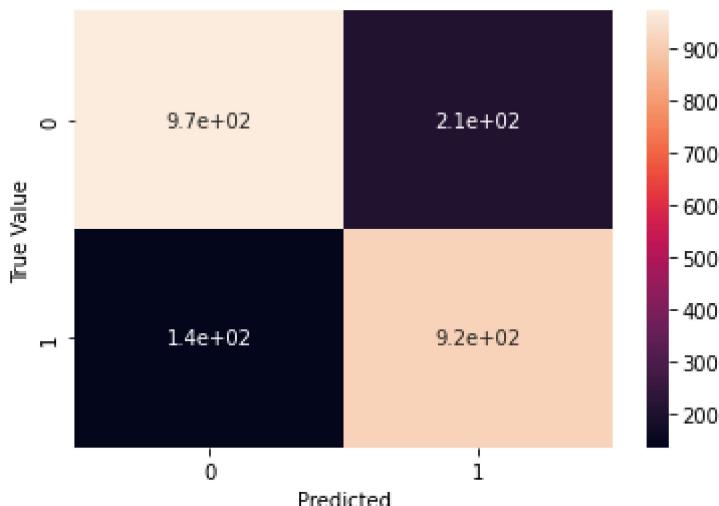
```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test, predict_rfc)

0.8471508689777055

from sklearn.metrics import confusion_matrix
cx = confusion_matrix(y_test, rfc.predict( X=X_test))
cx

array([[971, 208],
       [136, 916]])
```

```
from matplotlib import pyplot as plt
import seaborn as sn
sn.heatmap(cx, annot=True)
plt.xlabel('Predicted')
plt.ylabel('True Value')
plt.show()
```



▼ XGBoost Classifier

```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test, predict_xgb)

0.8595885054357465
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, model_xgb.predict(X_test))
cm
```