Data Preprocessing Assignment

K.Mohith Saran
2-7-2021
Assignment-3

In [1]:
```
#import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [4]:
```
#importing the csv file from computer
pd.read_csv('Churn_Modelling1.csv')
```

Out[4]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedS
0	1	15634602	Hargrave	619	France	Female	42.0	2	0.00	1	1	1	10134
1	2	15647311	Hill	608	Spain	Female	41.0	1	83807.86	1	0	1	11254
2	3	15619304	Onio	502	France	Female	42.0	8	159660.80	3	1	0	11393
3	4	15701354	Boni	699	France	Female	39.0	1	0.00	2	0	0	9382
4	5	15737888	Mitchell	850	Spain	Female	NaN	2	125510.82	1	1	1	7908
...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	France	Male	39.0	5	0.00	2	1	0	9627
9996	9997	15569892	Johnstone	516	France	Male	35.0	10	57369.61	1	1	1	10169
9997	9998	15584532	Liu	709	France	Female	36.0	7	0.00	1	0	1	4208
9998	9999	15682355	Sabbatini	772	Germany	Male	42.0	3	75075.31	2	1	0	9288
9999	10000	15628319	Walker	792	France	Female	28.0	4	130142.79	1	1	0	3819

10000 rows × 14 columns

In [5]:
```
dataset=pd.read_csv('Churn_Modelling1.csv')
```

In [6]:
```
dataset
```

Out[6]:

Number	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
1	15634602	Hargrave	619	France	Female	42.0	2	0.00	1	1	1	101348.88	1
2	15647311	Hill	608	Spain	Female	41.0	1	83807.86	1	0	1	112542.58	0
3	15619304	Onio	502	France	Female	42.0	8	159660.80	3	1	0	113931.57	1
4	15701354	Boni	699	France	Female	39.0	1	0.00	2	0	0	93826.63	0
5	15737888	Mitchell	850	Spain	Female	NaN	2	125510.82	1	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
9996	15606229	Obijiaku	771	France	Male	39.0	5	0.00	2	1	0	96270.64	0
9997	15569892	Johnstone	516	France	Male	35.0	10	57369.61	1	1	1	101699.77	0
9998	15584532	Liu	709	France	Female	36.0	7	0.00	1	0	1	42085.58	1
9999	15682355	Sabbatini	772	Germany	Male	42.0	3	75075.31	2	1	0	92888.52	1
10000	15628319	Walker	792	France	Female	28.0	4	130142.79	1	1	0	38190.78	0

× 14 columns

In [7]:
```
type(dataset)
```

Out[7]:
```
pandas.core.frame.DataFrame
```

In [8]:
```
dataset.info() #to check how many rows are non null
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   RowNumber        10000 non-null  int64
 1   CustomerId       10000 non-null  int64
 2   Surname          10000 non-null  object
 3   CreditScore      10000 non-null  int64
 4   Geography        9997 non-null   object
 5   Gender           9997 non-null   object
 6   Age              9996 non-null   float64
 7   Tenure           10000 non-null  int64
 8   Balance          9998 non-null   float64
 9   NumOfProducts    10000 non-null  int64
 10  HasCrCard        10000 non-null  int64
 11  IsActiveMember   10000 non-null  int64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64
dtypes: float64(3), int64(8), object(3)
memory usage: 1.1+ MB
```

In [11]:
```
dataset.isnull().any() #there are 4 null valued rows out of 14 rows
```

Out[11]:
```
RowNumber          False
CustomerId         False
Surname            False
CreditScore        False
Geography           True
Gender              True
Age                 True
Tenure             False
Balance             True
NumOfProducts      False
HasCrCard          False
IsActiveMember     False
EstimatedSalary    False
```

```
Exited             False
dtype: bool
```

In [19]:
```
dataset.isnull().sum()
```

Out[19]:
```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          3
Gender             3
Age                4
Tenure             0
Balance            2
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

In [20]:
```
dataset[dataset['Age'].isnull()].index.tolist()
```

Out[20]:
```
[4, 28, 43, 59]
```

In [21]:
```
dataset[dataset['Geography'].isnull()].index.tolist()
```

Out[21]:
```
[16, 30, 41]
```

In [22]:
```
dataset[dataset['Gender'].isnull()].index.tolist()
```

Out[22]:
```
[6, 21, 32]
```

In [23]:
```
dataset[dataset['Balance'].isnull()].index.tolist()
```

Out[23]:
```
[10, 26]
```

In [24]:
```
dataset['Age'].fillna(dataset['Age'].mean(),inplace=True)
```

In [27]:
```
dataset['Balance'].fillna(dataset['Balance'].mean(),inplace=True)
```

In [29]:
```
dataset.isnull().any()
```

Out[29]:
```
RowNumber          False
CustomerId         False
Surname            False
CreditScore        False
Geography           True
Gender              True
Age                False
Tenure             False
Balance            False
NumOfProducts      False
HasCrCard          False
IsActiveMember     False
EstimatedSalary    False
Exited             False
dtype: bool
```

In [64]:
```
#since Gender and Geography are the categorical values it must use mode and replace the null values with mode
dataset['Geography'] = dataset['Geography'].fillna(dataset['Geography'].mode()[0])
```

In [62]:
```
dataset['Gender'] = dataset['Gender'].fillna(dataset['Gender'].mode()[0])
```

In [65]:
```
dataset.isnull().any()
```

Out[65]:
```
RowNumber          False
CustomerId         False
Surname            False
CreditScore        False
Geography          False
Gender             False
Age                False
Tenure             False
Balance            False
NumOfProducts      False
HasCrCard          False
IsActiveMember     False
EstimatedSalary    False
Exited             False
dtype: bool
```

In [53]:
```
dataset['Age']=dataset['Age'].round()
```

In [66]:
```
dataset
```

Out[66]:

Number	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
1	15634602	Hargrave	619	France	Female	42.0	2	0.00	1	1	1	101348.88	1
2	15647311	Hill	608	Spain	Female	41.0	1	83807.86	1	0	1	112542.58	0
3	15619304	Onio	502	France	Female	42.0	8	159660.80	3	1	0	113931.57	1
4	15701354	Boni	699	France	Female	39.0	1	0.00	2	0	0	93826.63	0
5	15737888	Mitchell	850	Spain	Female	39.0	2	125510.82	1	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
9996	15606229	Obijiaku	771	France	Male	39.0	5	0.00	2	1	0	96270.64	0
9997	15569892	Johnstone	516	France	Male	35.0	10	57369.61	1	1	1	101699.77	0
9998	15584532	Liu	709	France	Female	36.0	7	0.00	1	0	1	42085.58	1
9999	15682355	Sabbatini	772	Germany	Male	42.0	3	75075.31	2	1	0	92888.52	1
10000	15628319	Walker	792	France	Female	28.0	4	130142.79	1	1	0	38190.78	0

× 14 columns

In [55]:
```
dataset['Gender'].mode()
```

Out[55]:
```
0    Male
dtype: object
```

In [67]:
```
dataset['Gender'][6]
```

Out[67]:
```
'Male'
```

In [71]:
```
dataset['Surname'].unique
```

Out[71]:
```
<bound method Series.unique of 0        Hargrave
1            Hill
2            Onio
3            Boni
4        Mitchell
           ...
9995     Obijiaku
9996    Johnstone
9997          Liu
9998     Sabbatini
9999        Walker
Name: Surname, Length: 10000, dtype: object>
```

In [72]:
```
dataset['Surname'].unique()
```

Out[72]:
```
array(['Hargrave', 'Hill', 'Onio', ..., 'Kashiwagi', 'Aldridge',
       'Burbidge'], dtype=object)
```

In [75]:
```
dataset['Geography'].unique()
```

Out[75]:
```
array(['France', 'Spain', 'Germany'], dtype=object)
```

In [76]:
```
dataset['Gender'].unique()
```

Out[76]:
```
array(['Female', 'Male'], dtype=object)
```

In [77]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
dataset['Geography']=le.fit_transform(dataset['Geography'])
dataset['Gender']=le.fit_transform(dataset['Gender'])
dataset['Surname']=le.fit_transform(dataset['Surname'])
```

In [78]:
```
dataset
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	1115	619	0	0	42.0	2	0.00	1	1	1	101348.88	1
1	2	15647311	1177	608	2	0	41.0	1	83807.86	1	0	1	112542.58	0
2	3	15619304	2040	502	0	0	42.0	8	159660.80	3	1	0	113931.57	1
3	4	15701354	289	699	0	0	39.0	1	0.00	2	0	0	93826.63	0
4	5	15737888	1822	850	2	0	39.0	2	125510.82	1	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	1999	771	0	1	39.0	5	0.00	2	1	0	96270.64	0
9996	9997	15569892	1336	516	0	1	35.0	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	1570	709	0	0	36.0	7	0.00	1	0	1	42085.58	1
9998	9999	15682355	2345	772	1	1	42.0	3	75075.31	2	1	0	92888.52	1
9999	10000	15628319	2751	792	0	0	28.0	4	130142.79	1	1	0	38190.78	0

In [84]:
```
x=dataset.iloc[:,0:13].values
```

In [85]:
```
x
```

Out[85]:
```
array([[1.0000000e+00, 1.5634602e+07, 1.1150000e+03, ..., 1.0000000e+00,
        1.0000000e+00, 1.0134888e+05],
       [2.0000000e+00, 1.5647311e+07, 1.1770000e+03, ..., 0.0000000e+00,
        1.0000000e+00, 1.1254258e+05],
       [3.0000000e+00, 1.5619304e+07, 2.0400000e+03, ..., 1.0000000e+00,
        0.0000000e+00, 1.1393157e+05],
       ...,
       [9.9980000e+03, 1.5584532e+07, 1.5700000e+03, ..., 0.0000000e+00,
        1.0000000e+00, 4.2085580e+04],
       [9.9990000e+03, 1.5682355e+07, 2.3450000e+03, ..., 1.0000000e+00,
        0.0000000e+00, 9.2888520e+04],
       [1.0000000e+04, 1.5628319e+07, 2.7510000e+03, ..., 1.0000000e+00,
        0.0000000e+00, 3.8190780e+04]])
```

In [86]:
```
#Converting into array
x=np.array(x)
```

In [87]:
```
x
```

Out[87]:
```
array([[1.0000000e+00, 1.5634602e+07, 1.1150000e+03, ..., 1.0000000e+00,
        1.0000000e+00, 1.0134888e+05],
       [2.0000000e+00, 1.5647311e+07, 1.1770000e+03, ..., 0.0000000e+00,
        1.0000000e+00, 1.1254258e+05],
       [3.0000000e+00, 1.5619304e+07, 2.0400000e+03, ..., 1.0000000e+00,
        0.0000000e+00, 1.1393157e+05],
       ...,
       [9.9980000e+03, 1.5584532e+07, 1.5700000e+03, ..., 0.0000000e+00,
        1.0000000e+00, 4.2085580e+04],
       [9.9990000e+03, 1.5682355e+07, 2.3450000e+03, ..., 1.0000000e+00,
        0.0000000e+00, 9.2888520e+04],
       [1.0000000e+04, 1.5628319e+07, 2.7510000e+03, ..., 1.0000000e+00,
        0.0000000e+00, 3.8190780e+04]])
```

In [88]:
```
y=dataset.iloc[:,-1:].values
```

In [89]:
```
y
```

Out[89]:
```
array([[1],
       [0],
       [1],
       ...,
       [1],
       [1],
       [0]], dtype=int64)
```

In [91]:
```
x.shape
```

Out[91]:
```
(10000, 13)
```

In [92]:
```
x.ndim
```

Out[92]:
```
2
```

In [93]:
```
y.ndim
```

Out[93]:
```
2
```

In [94]:
```
y.shape
```

Out[94]:
```
(10000, 1)
```

In [95]:
```
#converting dataset to array using onehotencoder
from sklearn.preprocessing import OneHotEncoder
oh=OneHotEncoder()
```

In [96]:
```
z=oh.fit_transform(x[:,0:1]).toarray()
z
```

Out[96]:
```
array([[1., 0., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 0., 0., 1.]])
```

In [97]:
```
x=np.concatenate((x,z),axis=1)
```

In [99]:
```
x.shape
```

Out[99]:
```
(10000, 10013)
```

In [100]:
```
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)
```

In [101]:
```
x_train.shape
```

Out[101]:
```
(8000, 10013)
```

In [102]:
```
x_test.shape
```

Out[102]:
```
(2000, 10013)
```

In [103]:
```
y_train.shape
```

Out[103]:
```
(8000, 1)
```

In [104]:
```
y_test.shape
```

Out[104]:
```
(2000, 1)
```

In [ ]:
```

```