In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import accuracy_score,confusion_matrix,
precision_score,recall_score,f1_score,auc,precision_recall_curve
from sklearn import preprocessing

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow.keras import optimizers
```

In [3]:
```
data=pd.read_csv('Churn_Modelling.csv')
data.head(10)
```

Out[3]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMembe
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	
5	6	15574012	Chu	645	Spain	Male	44	8	113755.78	2	1	
6	7	15592531	Bartlett	822	France	Male	50	7	0.00	2	1	
7	8	15656148	Obinna	376	Germany	Female	29	4	115046.74	4	1	
8	9	15792365	He	501	France	Male	44	4	142051.07	2	0	
9	10	15592389	H?	684	France	Male	27	2	134603.88	1	1	

In [4]:
```python
data.tail(10)
```

Out[4]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActive
9990	9991	15798964	Nkemakonam	714	Germany	Male	33	3	35016.60	1	1	
9991	9992	15769959	Ajuluchukwu	597	France	Female	53	4	88381.21	1	1	
9992	9993	15657105	Chukwualuka	726	Spain	Male	36	2	0.00	1	1	
9993	9994	15569266	Rahman	644	France	Male	28	7	155060.41	1	1	
9994	9995	15719294	Wood	800	France	Female	29	2	0.00	2	0	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	

In [5]:
```python
data=data.drop(['RowNumber','CustomerId','Surname'],axis=1)
data.head()
```

Out[5]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

In [6]:
```python
data.shape
```

Out[6]:
```
(10000, 11)
```

In [7]:
```
data.isnull().sum()
```

Out[7]:
```
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

In [8]:
```
data.isna().sum()
```

Out[8]:
```
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

In [9]:
```
data.duplicated().sum()
```

Out[9]:
```
0
```

In [10]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   CreditScore      10000 non-null  int64  
 1   Geography        10000 non-null  object 
 2   Gender           10000 non-null  object 
 3   Age              10000 non-null  int64  
 4   Tenure           10000 non-null  int64  
 5   Balance          10000 non-null  float64
 6   NumOfProducts    10000 non-null  int64  
 7   HasCrCard        10000 non-null  int64  
 8   IsActiveMember   10000 non-null  int64  
 9   EstimatedSalary  10000 non-null  float64
 10  Exited           10000 non-null  int64  
dtypes: float64(2), int64(7), object(2)
memory usage: 859.5+ KB
```

In [11]:
```
data.describe()
```

Out[11]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exite
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.00000
mean	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.20370
std	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.40276
min	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.00000
25%	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.00000
50%	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.00000
75%	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.00000
max	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.00000

In [12]:
```
data['Exited'].value_counts(normalize=True)
```

Out[12]:
```
0    0.7963
1    0.2037
Name: Exited, dtype: float64
```

In [13]:
```
data['Geography'].value_counts(normalize=True)
```

Out[13]:
```
France     0.5014
Germany    0.2509
Spain      0.2477
Name: Geography, dtype: float64
```

In [14]:
```
data[data['Geography']=='France']['Exited'].value_counts(normalize=True)
```

Out[14]:
```
0    0.838452
1    0.161548
Name: Exited, dtype: float64
```

In [15]:
```
data[data['Geography']=='Germany']['Exited'].value_counts(normalize=True)
```

Out[15]:
```
0    0.675568
1    0.324432
Name: Exited, dtype: float64
```

In [16]:
```
data[data['Geography']=='Spain']['Exited'].value_counts(normalize=True)
```

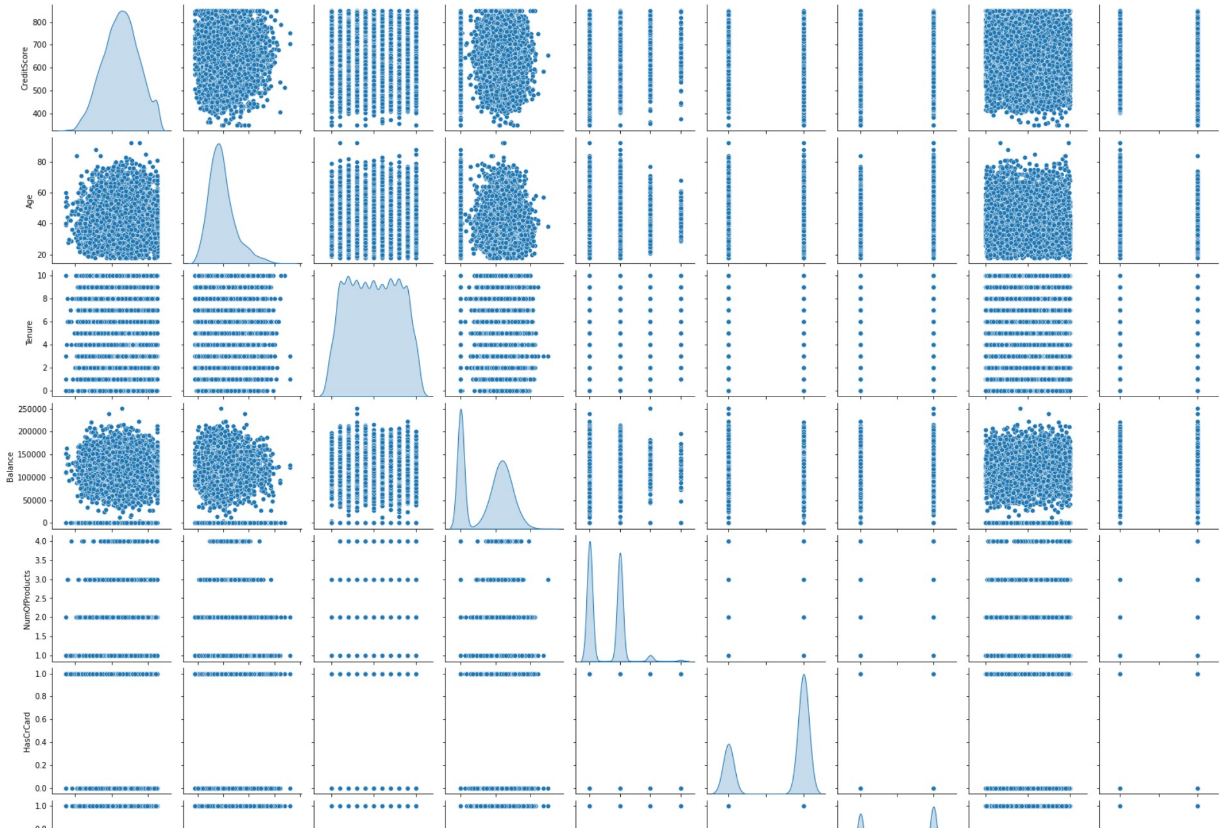
Out[16]:
```
0    0.833266
1    0.166734
Name: Exited, dtype: float64
```

In [17]:
```
data['Gender'].value_counts(normalize=True)
```

Out[17]:
```
Male      0.5457
Female    0.4543
Name: Gender, dtype: float64
```

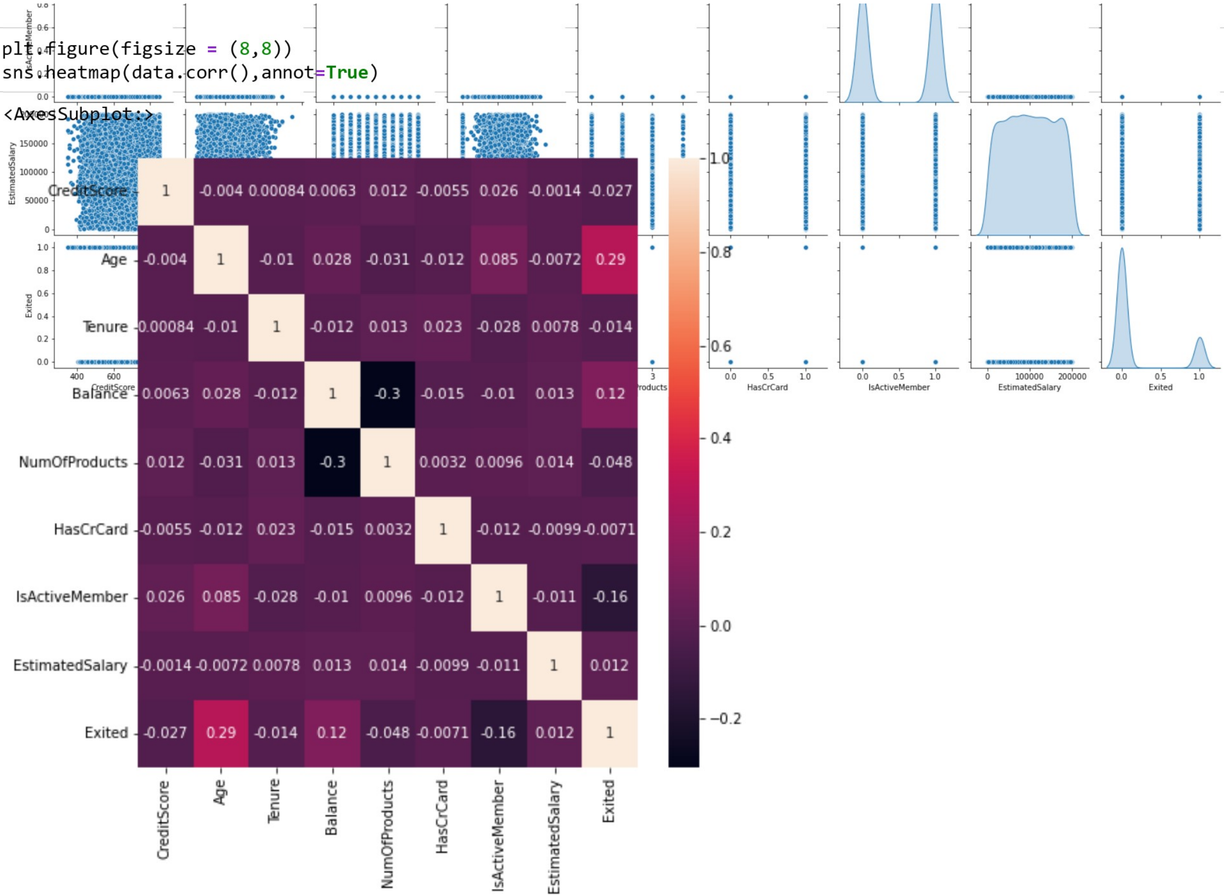
```
In [18]: sns.pairplot(data,diag_kind='kde')
```

```
Out[18]: <seaborn.axisgrid.PairGrid at 0x1d8aa4368e0>
```



In [19]:
```python
plt.figure(figsize = (8,8))
sns.heatmap(data.corr(),annot=True)
```

Out[19]: <AxesSubplot:>



In [20]:
```python
data1 = pd.get_dummies(data, columns=['Geography','Gender'])
data1.head()
```

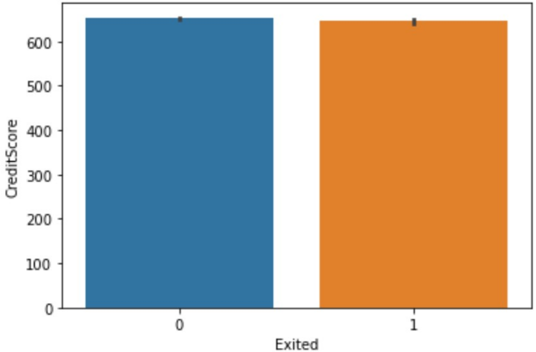
Out[20]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Geography_France	Geography
0	619	42	2	0.00	1	1	1	101348.88	1	1	
1	608	41	1	83807.86	1	0	1	112542.58	0	0	
2	502	42	8	159660.80	3	1	0	113931.57	1	1	
3	699	39	1	0.00	2	0	0	93826.63	0	1	
4	850	43	2	125510.82	1	1	1	79084.10	0	0	

In [21]:
```python
data1.dtypes
```

Out[21]:
```
CreditScore            int64
Age                    int64
Tenure                 int64
Balance              float64
NumOfProducts          int64
HasCrCard              int64
IsActiveMember         int64
EstimatedSalary      float64
Exited                 int64
Geography_France       uint8
Geography_Germany      uint8
Geography_Spain        uint8
Gender_Female          uint8
Gender_Male            uint8
dtype: object
```

In [22]:
```
for i in data1.columns:
    sns.barplot(x='Exited',y=i,data=data1)
    plt.show()
```



In [23]:
```
x=data1.drop('Exited',axis=1)
x.head()
```

Out[23]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Geography_France	Geography_Germa
0	619	42	2	0.00	1	1	1	101348.88	1	
1	608	41	1	83807.86	1	0	1	112542.58	0	
2	502	42	8	159660.80	3	1	0	113931.57	1	
3	699	39	1	0.00	2	0	0	93826.63	1	
4	850	43	2	125510.82	1	1	1	79084.10	0	

In [24]:
```
x.dtypes
```

Out[24]:
```
CreditScore            int64
Age                    int64
Tenure                 int64
Balance              float64
NumOfProducts          int64
HasCrCard              int64
IsActiveMember         int64
EstimatedSalary      float64
Geography_France       uint8
Geography_Germany      uint8
Geography_Spain        uint8
Gender_Female          uint8
Gender_Male            uint8
dtype: object
```

In [25]:
```
x=x.astype('float64')
x.dtypes
```

Out[25]:
```
CreditScore          float64
Age                  float64
Tenure               float64
Balance              float64
NumOfProducts        float64
HasCrCard            float64
IsActiveMember       float64
EstimatedSalary      float64
Geography_France     float64
Geography_Germany    float64
Geography_Spain      float64
Gender_Female        float64
Gender_Male          float64
dtype: object
```

In [26]:
```
y=data1['Exited']
y.head()
```

Out[26]:
```
0    1
1    0
2    1
3    0
4    0
Name: Exited, dtype: int64
```

In [27]:
```
# Split the data up in train and test sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30, random_state=75)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(7000, 13)
(3000, 13)
(7000,)
(3000,)
```

In [28]:
```
from sklearn.preprocessing import StandardScaler

# Define the scaler
scaler = StandardScaler().fit(x_train)

# Scale the train set
x_train = scaler.transform(x_train)

# Scale the test set
x_test = scaler.transform(x_test)
```

In [39]:
```
model = Sequential()

# Adding the layres with 13 inputs and fully connected neurons.
# using 'ELU' and 'ReLU' activation functions in the hidden layers
# using one sigmoid function at the output as it's a classification model
model.add(Dense(128, input_shape = (13,), activation = 'elu'))
model.add(Dense(64, activation = 'relu'))
model.add(Dense(32, activation = 'swish'))
model.add(Dense(16, activation = 'elu'))
model.add(Dense(64, activation = 'relu'))
model.add(Dense(32, activation = 'swish'))
model.add(Dense(16, activation = 'elu'))
model.add(Dense(8, activation = 'relu'))
model.add(Dense(4, activation = 'swish'))
model.add(Dense(1, activation = 'sigmoid'))

# choose the optimizer, learning rate, loss function,metrics
optm=optimizers.Adam(lr=0.0015)
model.compile(optimizer = optm, loss = 'binary_crossentropy', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_10 (Dense)            (None, 128)               1792

 dense_11 (Dense)            (None, 64)                8256

 dense_12 (Dense)            (None, 32)                2080

 dense_13 (Dense)            (None, 16)                528

 dense_14 (Dense)            (None, 64)                1088

 dense_15 (Dense)            (None, 32)                2080

 dense_16 (Dense)            (None, 16)                528
```

```
 dense_17 (Dense)            (None, 8)                 136

 dense_18 (Dense)            (None, 4)                 36

 dense_19 (Dense)            (None, 1)                 5

=================================================================
Total params: 16,529
Trainable params: 16,529
Non-trainable params: 0

C:\Users\DRISTI\anaconda3\lib\site-packages\keras\optimizers\optimizer_v2\adam.py:114: UserWarning: The `lr` argu
ment is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)
```

In [40]:

```
epoch=850
model_hist=model.fit(x_train, y_train,batch_size = 1000,validation_split = 0.2, epochs=epoch, verbose = 1)
model_hist
hist  = pd.DataFrame(model_hist.history)
hist['epoch'] = model_hist.epoch
print(hist)
plt.plot(hist['accuracy'])
plt.plot(hist['val_accuracy'])
plt.plot(hist['val_loss'])
plt.plot(hist['loss'])
plt.legend(("train" , "valid","val_loss","loss") , loc =0)
```

```
Epoch 1/850
6/6 [==============================] - 2s 41ms/step - loss: 0.6551 - accuracy: 0.7729 - val_loss: 0.6015 - val_ac
curacy: 0.7879
Epoch 2/850
6/6 [==============================] - 0s 13ms/step - loss: 0.5674 - accuracy: 0.7907 - val_loss: 0.5187 - val_ac
curacy: 0.7879
Epoch 3/850
6/6 [==============================] - 0s 10ms/step - loss: 0.4955 - accuracy: 0.7907 - val_loss: 0.4838 - val_ac
curacy: 0.7879
Epoch 4/850
6/6 [==============================] - 0s 10ms/step - loss: 0.4783 - accuracy: 0.7907 - val_loss: 0.4558 - val_ac
curacy: 0.7879
Epoch 5/850
6/6 [==============================] - 0s 13ms/step - loss: 0.4546 - accuracy: 0.7907 - val_loss: 0.4381 - val_ac
curacy: 0.7879
Epoch 6/850
6/6 [==============================] - 0s 14ms/step - loss: 0.4407 - accuracy: 0.7907 - val_loss: 0.4216 - val_ac
curacy: 0.7879
Epoch 7/850
```

In [31]:
```
y_preds = model.predict(x_test)

# Predict the results using 0.5 as a threshold
print((y_preds>0.5).astype(int))

# Identify the model evaluation loss and accuracy
results = model.evaluate(x_test, y_test)
```

```
94/94 [==============================] - 0s 1ms/step
[[0]
 [0]
 [0]
 ...
 [0]
 [0]
 [0]]
94/94 [==============================] - 0s 2ms/step - loss: 2.4743 - accuracy: 0.8193
```