Data Visualization and Pre-processing

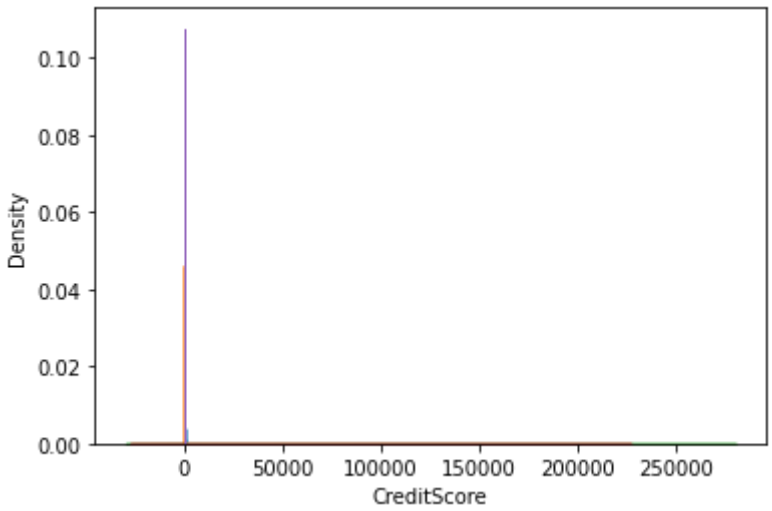
1. **Download the dataset**: **Dataset**
2. **Load the dataset**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
data=pd.read_csv('/content/chrun_modelling.csv')
```

3.**Perform Below Visualizations**

```
sns.kdeplot(data['CreditScore'], shade=True)
sns.kdeplot(data['Age'], shade=True)
sns.kdeplot(data['Balance'], shade=True)
sns.kdeplot(data['EstimatedSalary'], shade=True)
sns.kdeplot(data['Tenure'], shade=True)
```

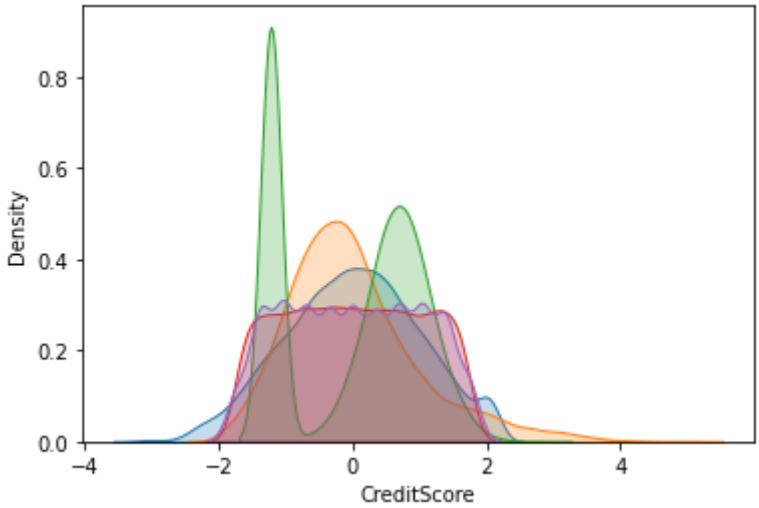
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc73ba72d90>
```




```
from sklearn.preprocessing import StandardScaler
stand= StandardScaler()
for column in ['CreditScore','Age','Balance','EstimatedSalary','Tenure']:
    data[column] = stand.fit_transform(data[column].values.reshape(-1,1))
```

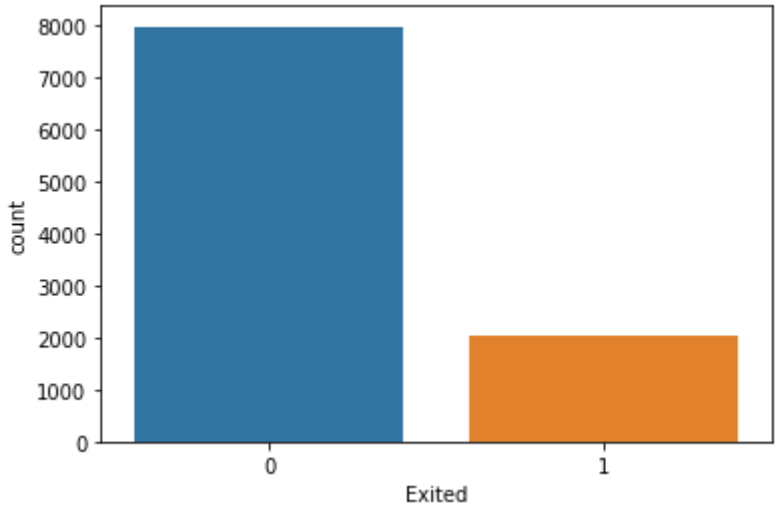
```
sns.kdeplot(data['CreditScore'], shade=True)
sns.kdeplot(data['Age'], shade=True)
sns.kdeplot(data['Balance'], shade=True)
sns.kdeplot(data['EstimatedSalary'], shade=True)
sns.kdeplot(data['Tenure'], shade=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc73b4a6fd0>
```




```
sns.countplot(data['Exited'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: P
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fc73ac92950>
```



4. **Perform descriptive statistics on the dataset**

```
data.describe()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Bal
count	10000.00000	1.000000e+04	1.000000e+04	1.000000e+04	1.000000e+04	1.000000

5 .**Handle the Missing values**

```
data.isnull().sum()
```

```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

6.**Find the outliers and replace the outliers**

```
lowerlimit=data['Age'].quantile(0.05)
lowerlimit
data[data['Age']<lowerlimit]
upperlimit=data['Age'].quantile(0.95)
upperlimit
data[data['Age']<upperlimit]
data=data[(data['Age']>lowerlimit)&(data['Age']<upperlimit)]
data
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure
0	1	15634602	Hargrave	619	France	Female	42	2

7.Check for Categorical columns and perform encoding

```
x = pd.get_dummies(x)

x.head()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	
0	1	15634602	-0.326221	0.293517	-1.041760	-1.225848	1	
1	2	15647311	-0.440036	0.198164	-1.387538	0.117350	1	
2	3	15619304	-1.536794	0.293517	1.032908	1.333053	3	
3	4	15701354	0.501521	0.007457	-1.387538	-1.225848	2	
4	5	15737888	2.063884	0.388871	-1.041760	0.785728	1	

5 rows × 2944 columns

8.Split the data into dependent and independent variables

```
x = data.iloc[:,0:10]
y = data.iloc[:,10]

print(x.shape)
print(y.shape)
```

```
(10000, 10)
(10000,)
```

9.Scale the independent variables

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x,y,test_size=0.25,random_state=0)
sc = StandardScaler()
x_train=sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)

x_train = pd.DataFrame(x_train)
x_train.head()
```

	0	1	2	3	4	5	6	7
0	-0.702176	-1.343330	-0.735507	0.015266	0.008860	0.673160	2.535034	-0.016332
1	-1.485722	1.558330	1.024427	-0.652609	0.008860	-1.207724	0.804242	-0.016332
2	-0.524522	-0.655156	0.808295	-0.461788	1.393293	-0.356937	0.804242	-0.016332
3	-1.167396	1.200594	0.396614	-0.080145	0.008860	-0.009356	-0.926551	-0.016332
4	-1.451159	0.778798	-0.467915	1.255605	0.701077	-1.207724	0.804242	-0.016332

5 rows × 2044 columns

10. Split the data into training and testing

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x,y,test_size=0.25,random_state=0)
print(' x_train.shape : ',x_train.shape)
print(' y_train.shape : ',y_train.shape)
print(' x_test.shape : ',x_test.shape)
print(' y_test.shape : ',y_test.shape)
```

```
 x_train.shape :  (7500, 10)
 y_train.shape :  (7500,)
 x_test.shape :  (2500, 10)
 y_test.shape :  (2500,)
```