**Assignment-2**

| Assignment Date | 24 September 2022 |
| --- | --- |
| Student Name | Divya Rani.R |
| Student Roll Number | 111619104024 |
| Maximum Marks | 2 Marks |

**Task-1**

**Download the Dataset:**

**Churn\_Modelling.csv**

**Task-2:**

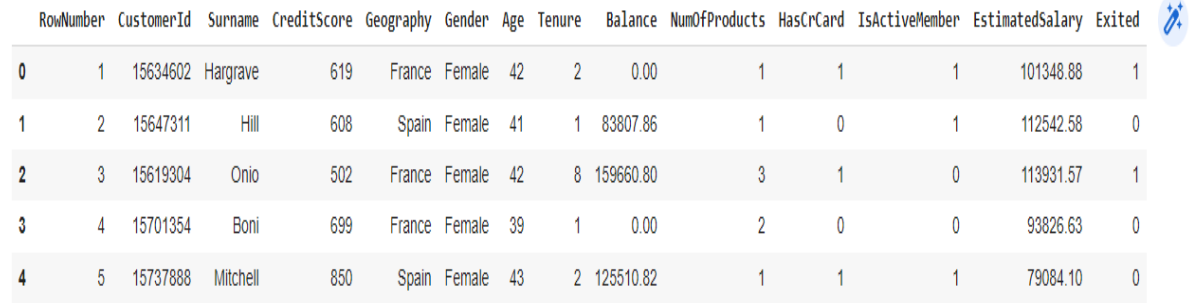
**Load the Dataset:**

**Solution:**

ds=pd.read\_csv("gdrive/My Drive/Churn\_Modelling.csv")

df=pd.DataFrame(ds)

df.head()

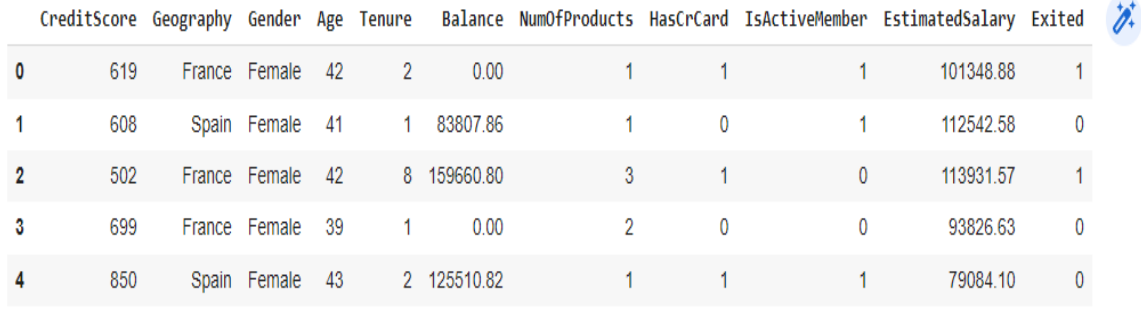
 df['HasCrCard'] **=** df['HasCrCard']**.**astype('category')

df['IsActiveMember'] **=** df['IsActiveMember']**.**astype('category')

df['Exited'] **=** df['Exited']**.**astype('category')

df **=** df**.**drop(columns**=**['RowNumber', 'CustomerId', 'Surname'])

df**.**head()



**Task-3:**

**3. Perform Below Visualizations.**

● Univariate Analysis

● Bi - Variate Analysis

● Multi - Variate Analysis

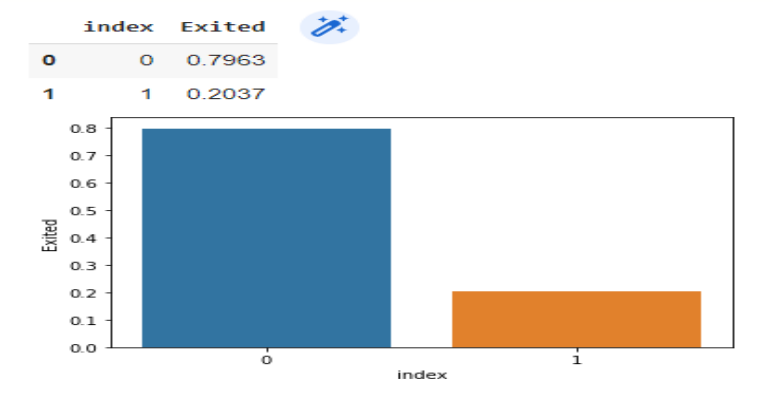
**Solution:**

import seaborn as sn

density = df['Exited'].value\_counts(normalize=True).reset\_index()

sn.barplot(data=density, x='index', y='Exited', );

density



import matplotlib.pyplot as plt

categorical = df.drop(columns=['CreditScore', 'Age', 'Tenure', 'Balance', 'EstimatedSalary'])

rows = int(np.ceil(categorical.shape[1] / 2)) - 1

# create sub-plots anf title them

fig, axes = plt.subplots(nrows=rows, ncols=2, figsize=(10,6))

axes = axes.flatten()

for row in range(rows):

cols = min(2, categorical.shape[1] - row\*2)

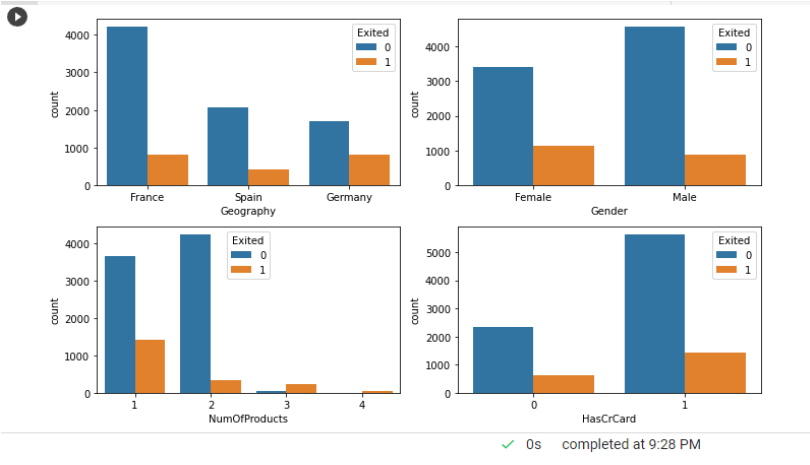
for col in range(cols):

col\_name = categorical.columns[2 \* row + col]

ax = axes[row\*2 + col]

sn.countplot(data=categorical, x=col\_name, hue="Exited", ax=ax);

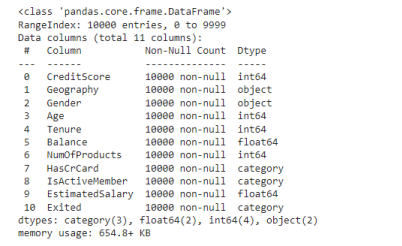
plt.tight\_layout()



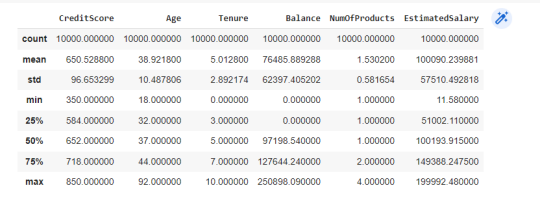
**Task-4:**

**Perform descriptive statistics on the dataset**

df.info()



df.describe()

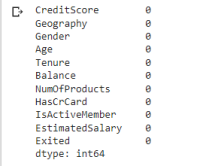


**Task-5:**

**Handle the Missing values.**

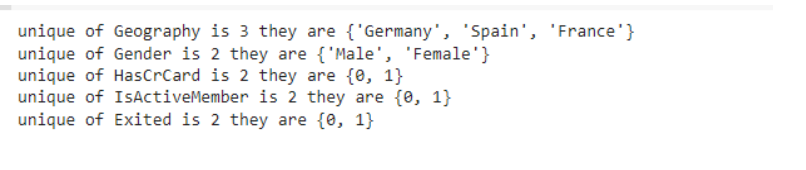
**Solution:**

df.isna().sum()



for i in df:

if df[i].dtype=='object' or df[i].dtype=='category':

print("unique of "+i+" is "+str(len(set(df[i])))+" they are "+str(set(df[i])))

**Task-6:**

**Find the outliers and replace the outliers**

**Solution:**

finding whether the outlier is present

def box\_scatter(data, x, y):

fig, (ax1, ax2) = plt.subplots(nrows=2, ncols=1, figsize=(16,6))

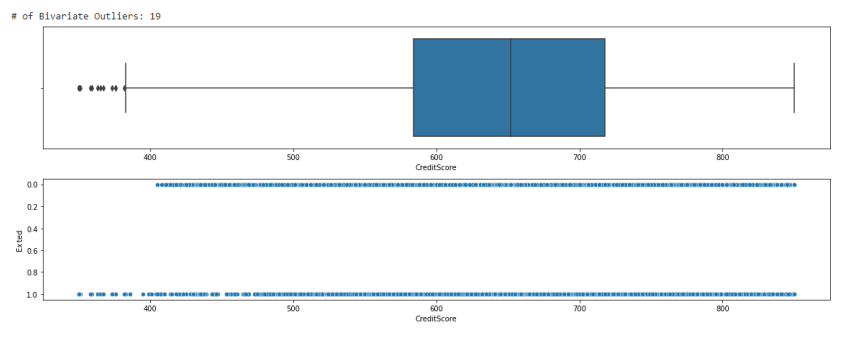
sn.boxplot(data=data, x=x, ax=ax1)

sn.scatterplot(data=data, x=x,y=y,ax=ax2)

box\_scatter(df,'CreditScore','Exited');

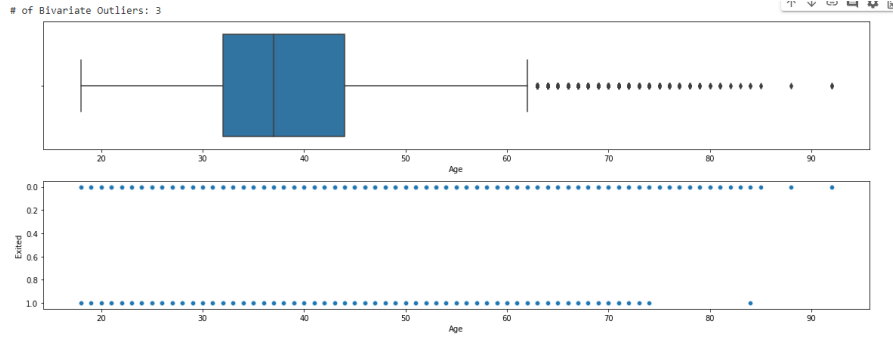
plt.tight\_layout()

print(f"# of Bivariate Outliers: {len(df.loc[df['CreditScore'] < 400])}")

 box\_scatter(df,'Age','Exited');

plt.tight\_layout()

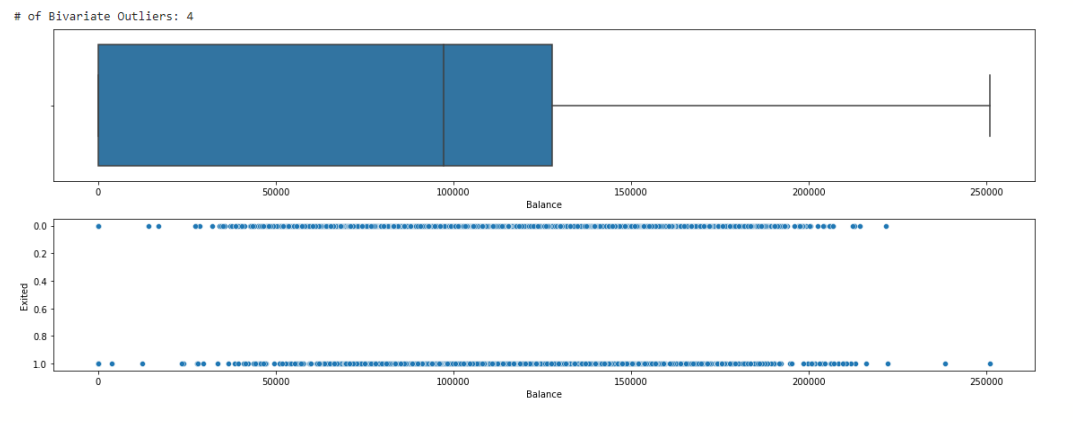
print(f"# of Bivariate Outliers: {len(df.loc[df['Age'] > 87])}")



box\_scatter(df,'Balance','Exited');

plt.tight\_layout()

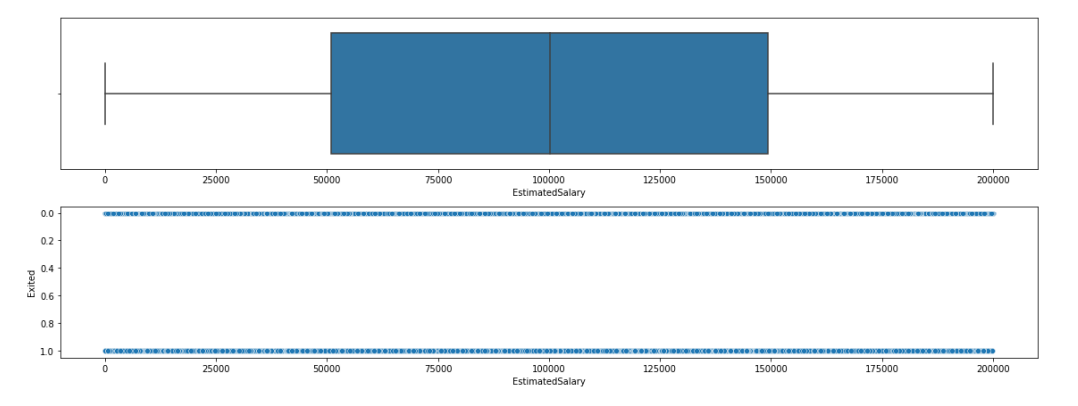
print(f"# of Bivariate Outliers: {len(df.loc[df['Balance'] > 220000])}")



9

box\_scatter(df,'EstimatedSalary','Exited');

plt.tight\_layout()



**Removing of Outliers**

for i in df:

if df[i].dtype=='int64' or df[i].dtypes=='float64':

q1=df[i].quantile(0.25)

q3=df[i].quantile(0.75)

iqr=q3-q1

upper=q3+1.5\*iqr

lower=q1-1.5\*iqr

df[i]=np.where(df[i] >upper, upper, df[i])

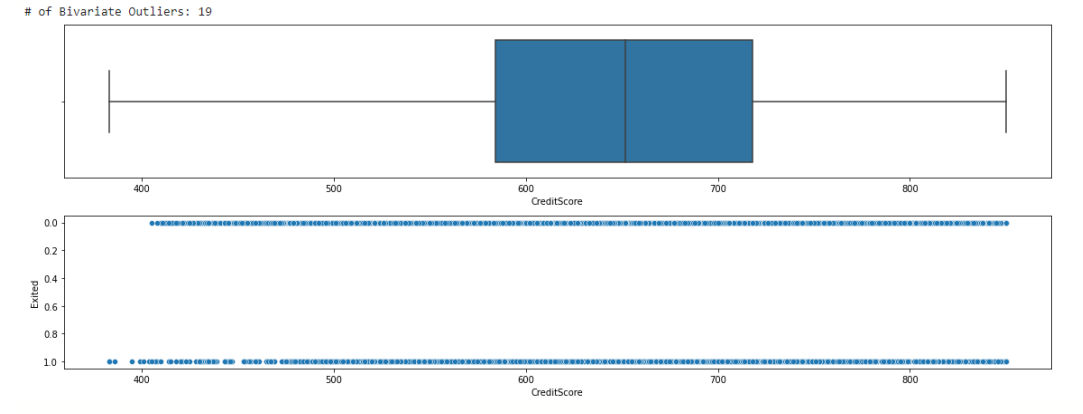
df[i]=np.where(df[i] <lower, lower, df[i])

After removing the outliers the boxplot will be like

box\_scatter(df,'CreditScore','Exited');

plt.tight\_layout()

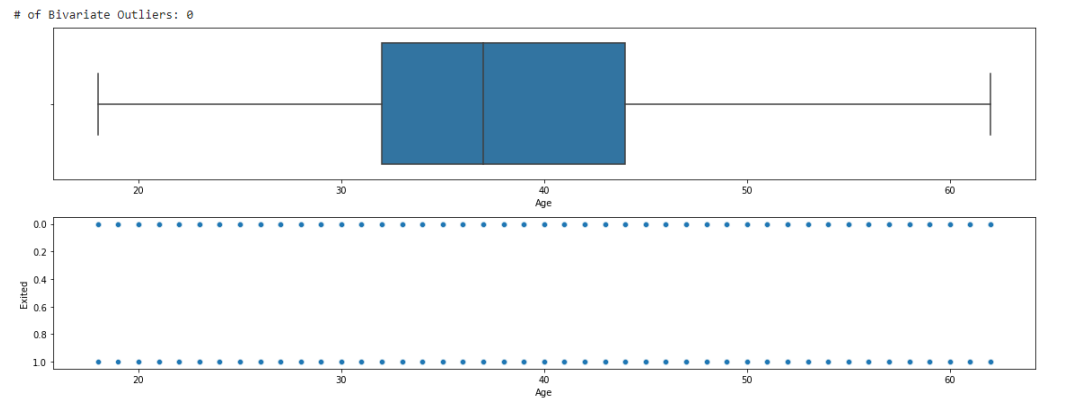
print(f"# of Bivariate Outliers: {len(df.loc[df['CreditScore'] < 400])}")



box\_scatter(df,'Age','Exited');

plt.tight\_layout()

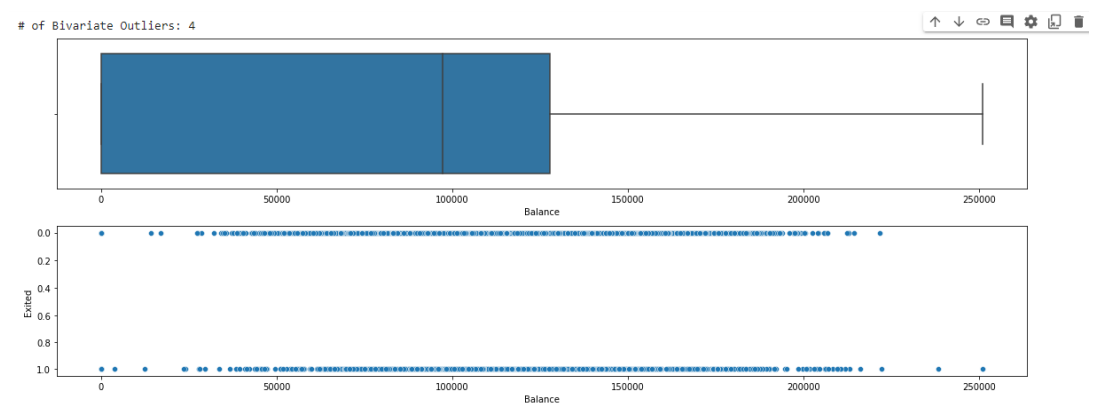
print(f"# of Bivariate Outliers: {len(df.loc[df['Age'] > 87])}")



box\_scatter(df,'Balance','Exited');

plt.tight\_layout()

print(f"# of Bivariate Outliers: {len(df.loc[df['Balance'] > 220000])}")



**TASK-7**

**Check for Categorical columns and perform encoding. Solution:**

**from** sklearn.preprocessing **import** LabelEncoder

encoder**=**LabelEncoder()

**for** i **in** df:

**if** df[i]**.**dtype**==**'object' **or** df[i]**.**dtype**==**'category':

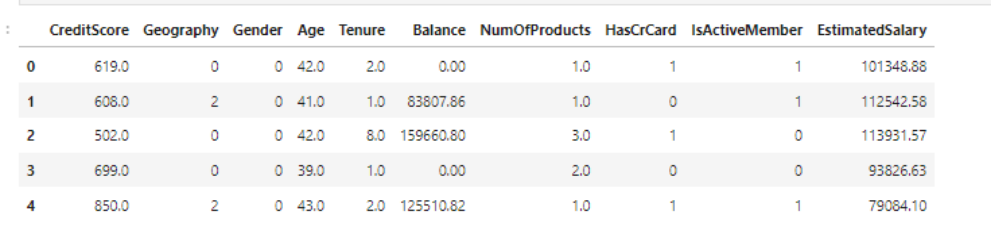
df[i]**=**encoder**.**fit\_transform(df[i])

**TASK-8**

**Split the data into dependent and independent variables.** Solution:

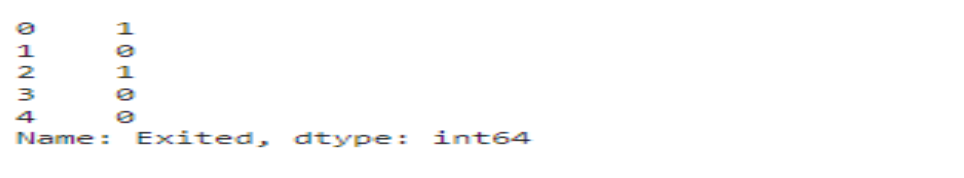
x**=**df**.**iloc[:,:**-**1]

x**.**head()



y**=**df**.**iloc[:,**-**1]

y**.**head()



**TASK-9:**

**Scale the independent variables**

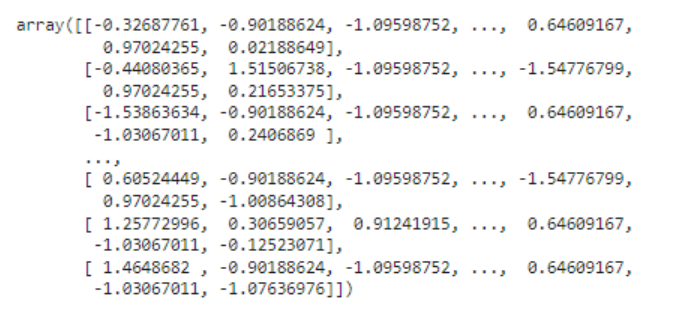
**Solution:**

**from** sklearn.preprocessing **import** StandardScaler

scaler**=**StandardScaler()

x**=**scaler**.**fit\_transform(x)

x



**TASK-10:**

**Split the data into training and testing**

**Solution:**

**from** sklearn.model\_selection **import** train\_test\_split

x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(x,y,test\_size**=**0.33)

