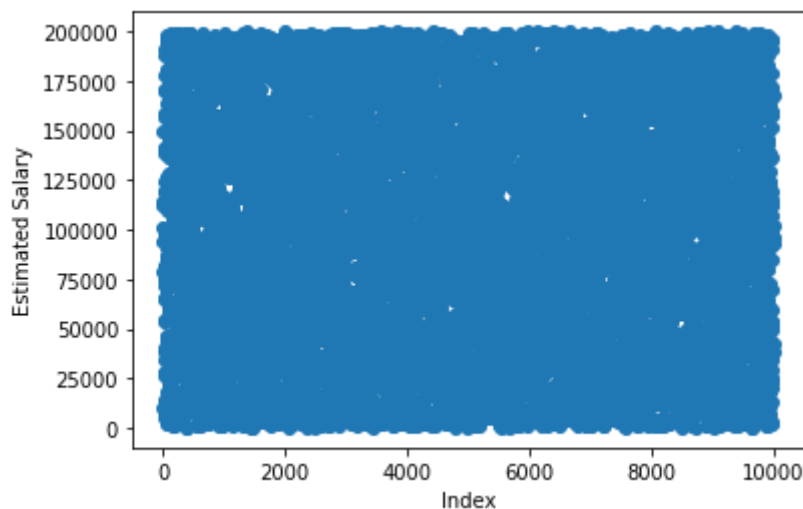


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
Value = pd.read_csv('/content/Churn_Modelling.csv') #loading the data set
Value.head(10)
```

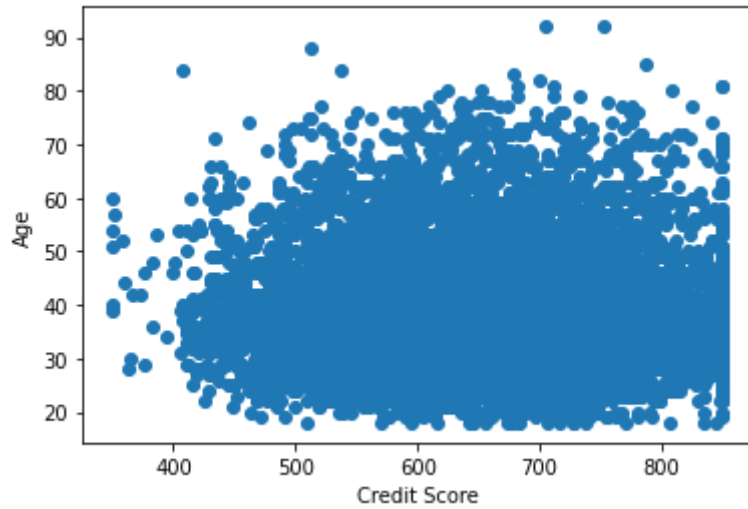
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Ba
0	1	15634602	Hargrave	619	France	Female	42	2	
1	2	15647311	Hill	608	Spain	Female	41	1	838
2	3	15619304	Onio	502	France	Female	42	8	1596
3	4	15701354	Boni	699	France	Female	39	1	
4	5	15737888	Mitchell	850	Spain	Female	43	2	1255
5	6	15574012	Chu	645	Spain	Male	44	8	1137
6	7	15592531	Bartlett	822	France	Male	50	7	
7	8	15656148	Obinna	376	Germany	Female	29	4	1150
8	9	15792365	He	501	France	Male	44	4	1420
9	10	15592389	H?	684	France	Male	27	2	1346

```
plt.scatter(Value.index, Value['EstimatedSalary']) #univariate analysis
plt.xlabel('Index')
plt.ylabel('Estimated Salary')
plt.show()
```



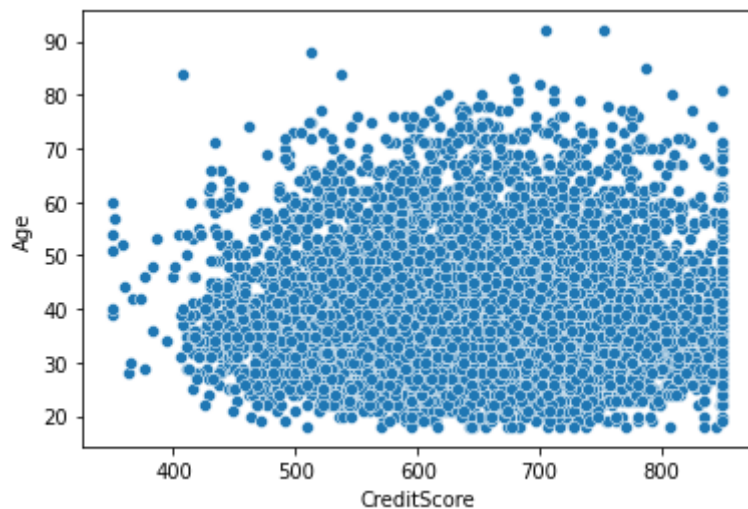
▼ Data Visualization

```
plt.scatter(Value['CreditScore'],Value['Age']) #Bivariate analysis
plt.xlabel('Credit Score')
plt.ylabel('Age')
plt.show()
```

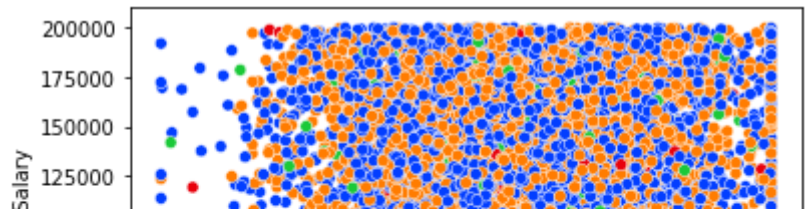


```
sns.scatterplot(x='CreditScore',y='Age',data=Value) #bivariant analysis
```

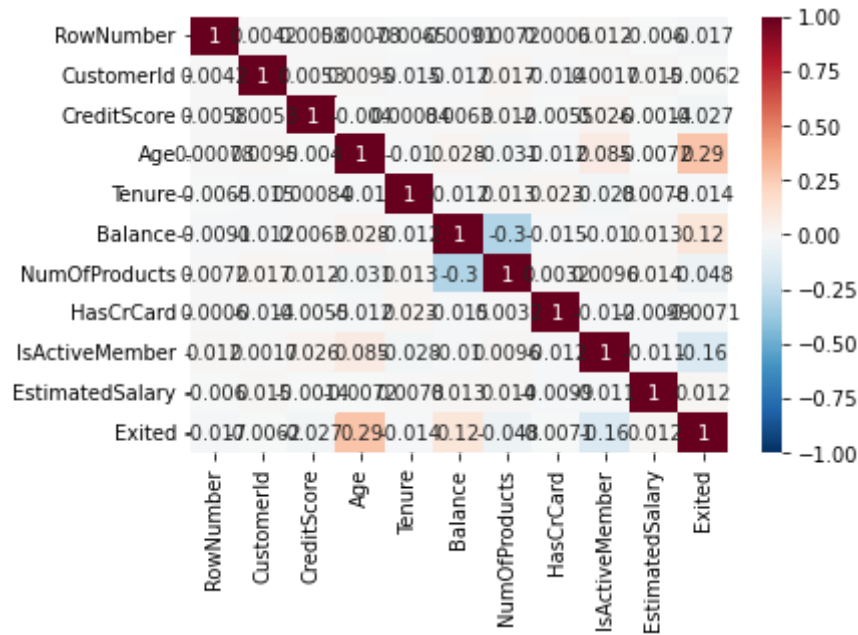
<matplotlib.axes._subplots.AxesSubplot at 0x7f10a8222350>



```
sns.scatterplot(
    x='CreditScore',
    y='EstimatedSalary',
    data=Value,
    palette='bright',
    hue='NumOfProducts'); ##multivariant analysis
```



```
sns.heatmap(
    Value.corr(),
    cmap='RdBu_r',
    annot=True,
    vmin=-1, vmax=1); ##multivariant analysis
```



```
Value.describe(include='all')
```

	RowNumber	CustomerId	Surname	CreditScore	Geography
count	10000.00000	1.000000e+04	10000	10000.000000	10000
unique	NaN	NaN	2932	NaN	NaN
top	NaN	NaN	Smith	NaN	France
freq	NaN	NaN	32	NaN	501
mean	5000.50000	1.569094e+07	NaN	650.528800	NaN
std	2886.89568	7.193619e+04	NaN	96.653299	NaN
min	1.00000	1.556570e+07	NaN	350.000000	NaN
25%	2500.75000	1.562853e+07	NaN	584.000000	NaN
50%	5000.50000	1.569094e+07	NaN	650.528800	NaN
75%	7500.25000	1.575387e+07	NaN	716.642400	NaN
max	10000.00000	1.599551e+07	NaN	900.000000	NaN

Handling Missing values and outliers

Value.isnull().sum() # finding missing value and found no missing value present in this da

```

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

```

```

qnt = Value.quantile(q=[0.25,0.75])
qnt

```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance
0.25	2500.75	15628528.25	584.0	32.0	3.0	0.00

```

IQR=qnt.loc[0.75]-qnt.loc[0.25]
IQR

```

```

RowNumber      4999.5000
CustomerId      124705.5000
CreditScore     134.0000
Age             12.0000
Tenure          4.0000
Balance        127644.2400
NumOfProducts   1.0000
HasCrCard       1.0000
IsActiveMember  1.0000
EstimatedSalary 98386.1375
Exited          0.0000
dtype: float64

```

```

Upper_exterm= qnt.loc[0.75]+1.5*IQR
lower_exterm= qnt.loc[0.25]-1.5*IQR

```

Upper_exterm

```

RowNumber      1.499950e+04
CustomerId      1.594029e+07

```

```
CreditScore      9.190000e+02
Age              6.200000e+01
Tenure           1.300000e+01
Balance          3.191106e+05
NumOfProducts   3.500000e+00
HasCrCard        2.500000e+00
IsActiveMember   2.500000e+00
EstimatedSalary  2.969675e+05
Exited           0.000000e+00
dtype: float64
```

lower_exterm

```
RowNumber      -4.998500e+03
CustomerId      1.544147e+07
CreditScore     3.830000e+02
Age             1.400000e+01
Tenure          -3.000000e+00
Balance         -1.914664e+05
NumOfProducts   -5.000000e-01
HasCrCard       -1.500000e+00
IsActiveMember  -1.500000e+00
EstimatedSalary -9.657710e+04
Exited           0.000000e+00
dtype: float64
```

```
print(Value[Value['Balance']>319110.6],
Value[Value['Balance']<-191466.4]) ##no outlier in Balance
```

```
Empty DataFrame
Columns: [RowNumber, CustomerId, Surname, CreditScore, Geography, Gender, Age, Tenure]
Index: [] Empty DataFrame
Columns: [RowNumber, CustomerId, Surname, CreditScore, Geography, Gender, Age, Tenure]
Index: []
```



```
print(Value[Value['Age']>62]) ##outlier in Age
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	\
58	59	15623944	T'ien	511	Spain	Female	66	
85	86	15805254	Ndukaku	652	Spain	Female	75	
104	105	15804919	Dunbabin	670	Spain	Female	65	
158	159	15589975	Maclean	646	France	Female	73	
181	182	15789669	Hsia	510	France	Male	65	
...	
9753	9754	15705174	Chiedozie	656	Germany	Male	68	
9765	9766	15777067	Thomas	445	France	Male	64	
9832	9833	15814690	Chukwujekwu	595	Germany	Female	64	
9894	9895	15704795	Vagin	521	France	Female	77	
9936	9937	15653037	Parks	609	France	Male	77	

	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	\
58	4	0.00	1	1	0	
85	10	0.00	2	1	1	
104	1	0.00	1	1	1	
158	6	97259.25	1	0	1	
181	2	0.00	2	1	1	

```

...      ...      ...      ...      ...
9753      7  153545.11      1      1      1
9765      2  136770.67      1      0      1
9832      2  105736.32      1      1      1
9894      6      0.00      2      1      1
9936      1      0.00      1      0      1

```

```

      EstimatedSalary  Exited
58      1643.11      1
85      114675.75      0
104      177655.68      1
158      104719.66      0
181      48071.61      0
...      ...      ...
9753      186574.68      0
9765      43678.06      0
9832      89935.73      1
9894      49054.10      0
9936      18708.76      0

```

[359 rows x 14 columns]

#replaced with mean value

```
Value['Age']= np.where(Value['Age']>62,Value['Age'].mean(),Value['Age'])
```

```
Value[Value['Age']<14] #no outlier lower exterm here
```

RowNumber CustomerId Surname CreditScore Geography Gender

```
print(Value[Value['Tenure']>13],
```

```
Value[Value['Tenure']<-3]) ##no outlier Tenure
```

Empty DataFrame

Columns: [RowNumber, CustomerId, Surname, CreditScore, Geography, Gender, Age, Tenure]

Index: [] Empty DataFrame

Columns: [RowNumber, CustomerId, Surname, CreditScore, Geography, Gender, Age, Tenure]

Index: []

```
print(Value[Value['CreditScore']>919]) ##no outlier in Upper exterm of Credit Score
```

Empty DataFrame

Columns: [RowNumber, CustomerId, Surname, CreditScore, Geography, Gender, Age, Tenure]

Index: []

```
Value[Value['CreditScore']<383] #outliers
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	G
7	8	15656148	Obinna	376	Germany	F
942	943	15804586	Lin	376	France	F
1193	1194	15779947	Thomas	363	Spain	F
1405	1406	15612494	Panicucci	359	France	F
1631	1632	15685372	Azubuike	350	Spain	
1838	1839	15758813	Campbell	350	Germany	
1962	1963	15692416	Aikenhead	358	Spain	F
2473	2474	15679249	Chou	351	Germany	F
2579	2580	15597896	Ozoemena	365	Germany	
8154	8155	15791533	Ch'ien	367	Spain	
8723	8724	15803202	Onyekachi	350	France	
8762	8763	15765173	Lin	350	France	F

```
#replaced with mean value
Value['CreditScore']= np.where(Value['CreditScore']<383,Value['CreditScore'].mean(),Value[

print(Value[Value['EstimatedSalary']>296967.5],
      Value[Value['EstimatedSalary']<=965771])  ##no outlier

Empty DataFrame
Columns: [RowNumber, CustomerId, Surname, CreditScore, Geography, Gender, Age, Tenure]
Index: []
Empty DataFrame
Columns: [RowNumber, CustomerId, Surname, CreditScore, Geography, Gender, Age, Tenure]
Index: []

pd.get_dummies(Value,columns=['Gender','Geography']) ## one hot encoding but it is not rec
```

	RowNumber	CustomerId	Surname	CreditScore	Age	Tenure
0	1	15634602	Hargrave	619.0	42.0	2
1	2	15647311	Hill	608.0	41.0	1
2	3	15619304	Onio	502.0	42.0	8
3	4	15701354	Boni	699.0	39.0	1
4	5	15737888	Mitchell	850.0	43.0	2
...
9995	9996	15606229	Obijiaku	771.0	39.0	5
9996	9997	15569892	Johnstone	516.0	35.0	10

```
pd.get_dummies(Value, columns=['Surname'])
```

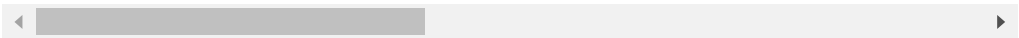
	RowNumber	CustomerId	CreditScore	Geography	Gender	Age
0	1	15634602	619.0	France	Female	42.0
1	2	15647311	608.0	Spain	Female	41.0
2	3	15619304	502.0	France	Female	42.0
3	4	15701354	699.0	France	Female	39.0
4	5	15737888	850.0	Spain	Female	43.0
...
9995	9996	15606229	771.0	France	Male	39.0
9996	9997	15569892	516.0	France	Male	35.0
9997	9998	15584532	709.0	France	Female	36.0
9998	9999	15682355	772.0	Germanv	Male	42.0

```
from sklearn.preprocessing import LabelEncoder
var = LabelEncoder()
```

```
## Label encoding is perfect for data preprocessing
Value['Surname']=var.fit_transform(Value['Surname']) ## not much considered for data pred
Value['Gender']=var.fit_transform(Value['Gender'])
Value['Geography']=var.fit_transform(Value['Geography'])
```

Value

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gen
0	1	15634602	1115	619.0	0	
1	2	15647311	1177	608.0	2	
2	3	15619304	2040	502.0	0	
3	4	15701354	289	699.0	0	
4	5	15737888	1822	850.0	2	
...
9995	9996	15606229	1999	771.0	0	
9996	9997	15569892	1336	516.0	0	
9997	9998	15584532	1570	709.0	0	



Separating dependent and Independent Variables

```
y = Value['EstimatedSalary']  
x = Value.drop(columns=['EstimatedSalary','RowNumber'],axis=1)  
x
```

	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	N
0	15634602	1115	619.0	0	0	42.0	2	0.00	
1	15647311	1177	608.0	2	0	41.0	1	83807.86	

```
Names = x.columns
```

```
## scaling the independent variable
```

```
from sklearn.preprocessing import scale
```

```
x=scale(x)
```

```
x=pd.DataFrame(x,columns=Names)
```

```
...      ...      ...      ...      ...      ...      ...      ...      ...
```

```
x.head()
```

	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bala
0	-0.783213	-0.464183	-0.332960	-0.901886	-1.095988	0.482051	-1.041760	-1.225
1	-0.606534	-0.390911	-0.447549	1.515067	-1.095988	0.366388	-1.387538	0.117
2	-0.995885	0.628988	-1.551769	-0.901886	-1.095988	0.482051	1.032908	1.333
3	0.144767	-1.440356	0.500414	-0.901886	-1.095988	0.135061	-1.387538	-1.225
4	0.652659	0.371354	2.073407	1.515067	-1.095988	0.597715	-1.041760	0.785

```
# Splitting data inti training and test SET
```

```
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)
```

```
len(x_train) ## 80% of total data
```

```
8000
```

[Colab paid products](#) - [Cancel contracts here](#)

