

1. Importing Required Package

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

```
%matplotlib inline
```

1. Loading the Dataset

```
df=pd.read_csv('/content/Churn_Modelling.csv')
df
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender
Age \						
0	1	15634602	Hargrave	619	France	Female
42						
1	2	15647311	Hill	608	Spain	Female
41						
2	3	15619304	Onio	502	France	Female
42						
3	4	15701354	Boni	699	France	Female
39						
4	5	15737888	Mitchell	850	Spain	Female
43						
...
...						
9995	9996	15606229	Obijiaku	771	France	Male
39						
9996	9997	15569892	Johnstone	516	France	Male
35						
9997	9998	15584532	Liu	709	France	Female
36						
9998	9999	15682355	Sabbatini	772	Germany	Male
42						
9999	10000	15628319	Walker	792	France	Female
28						

	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	\
0	2	0.00	1	1		1
1	1	83807.86	1	0		1
2	8	159660.80	3	1		0
3	1	0.00	2	0		0
4	2	125510.82	1	1		1
...
9995	5	0.00	2	1		0
9996	10	57369.61	1	1		1
9997	7	0.00	1	0		1
9998	3	75075.31	2	1		0
9999	4	130142.79	1	1		0

	EstimatedSalary	Exited
0	101348.88	1
1	112542.58	0
2	113931.57	1
3	93826.63	0
4	79084.10	0
...
9995	96270.64	0
9996	101699.77	0
9997	42085.58	1
9998	92888.52	1
9999	38190.78	0

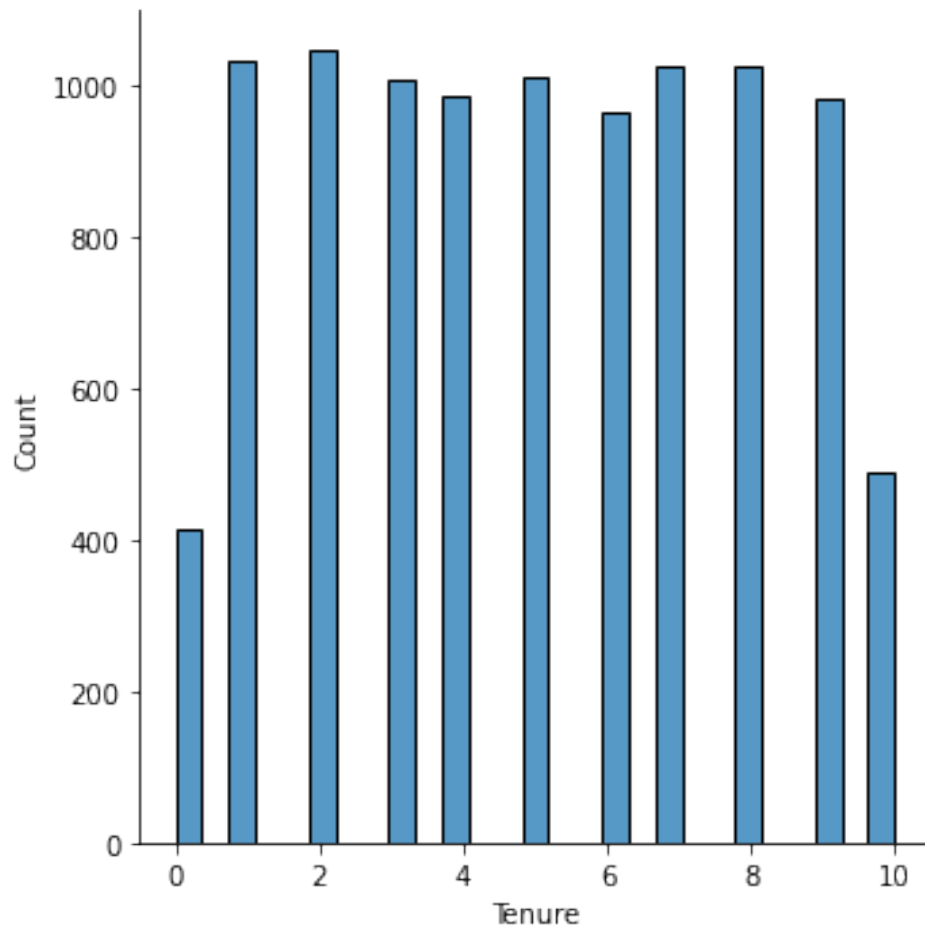
[10000 rows x 14 columns]

1. Visualizations

3.1 Univariate Analysis

```
sns.displot(df.Tenure)
```

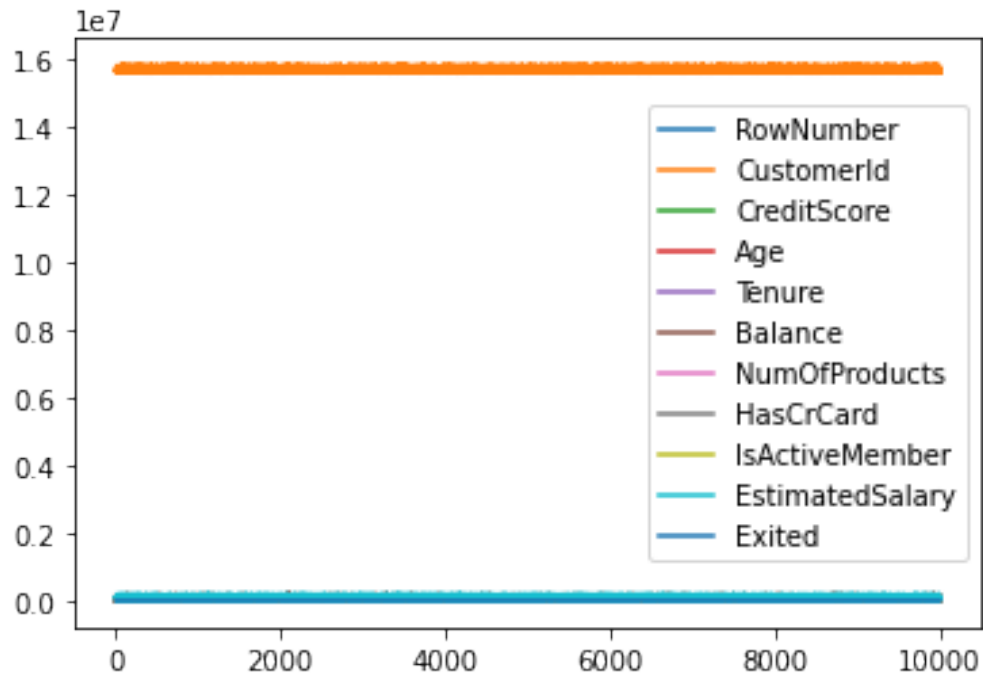
```
<seaborn.axisgrid.FacetGrid at 0x7fc35cdb3e10>
```



3.2 Bi-Variate Analysis

```
df.plot.line()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc359e7c410>
```

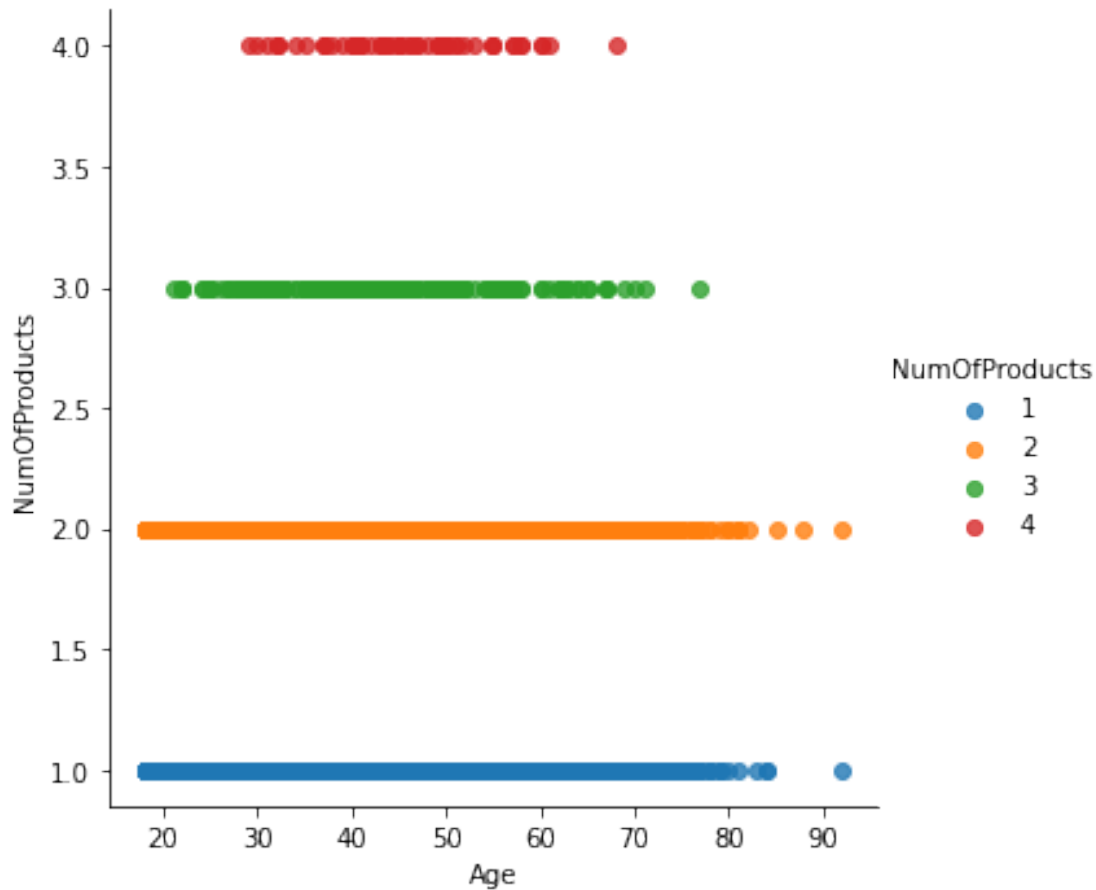


3.3 Multi - Variate Analysis

```
sns.lmplot("Age", "NumOfProducts", df, hue="NumOfProducts",
fit_reg=False);
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variables as keyword args: x, y,
data. From version 0.12, the only valid positional argument will be
`data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
```

```
FutureWarning
```



1. Perform descriptive statistics on the dataset
`df.describe()`

	RowNumber	CustomerId	CreditScore	Age
Tenure \				
count	10000.00000	1.000000e+04	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800
std	2886.89568	7.193619e+04	96.653299	10.487806
min	1.00000	1.556570e+07	350.000000	18.000000
25%	2500.75000	1.562853e+07	584.000000	32.000000
50%	5000.50000	1.569074e+07	652.000000	37.000000
75%	7500.25000	1.575323e+07	718.000000	44.000000
max	10000.00000	1.581569e+07	850.000000	92.000000
	Balance	NumOfProducts	HasCrCard	IsActiveMember \

count	10000.000000	10000.000000	10000.000000	10000.000000
mean	76485.889288	1.530200	0.70550	0.515100
std	62397.405202	0.581654	0.45584	0.499797
min	0.000000	1.000000	0.00000	0.000000
25%	0.000000	1.000000	0.00000	0.000000
50%	97198.540000	1.000000	1.00000	1.000000
75%	127644.240000	2.000000	1.00000	1.000000
max	250898.090000	4.000000	1.00000	1.000000

	EstimatedSalary	Exited
count	10000.000000	10000.000000
mean	100090.239881	0.203700
std	57510.492818	0.402769
min	11.580000	0.000000
25%	51002.110000	0.000000
50%	100193.915000	0.000000
75%	149388.247500	0.000000
max	199992.480000	1.000000

1. Handle the Missing values

```
data = pd.read_csv('/content/Churn_Modelling.csv')
pd.isnull(data["Gender"])
```

```
0      False
1      False
2      False
3      False
4      False
...
9995   False
9996   False
9997   False
9998   False
9999   False
```

Name: Gender, Length: 10000, dtype: bool

1. Find the outliers and replace the outliers

```
df["Tenure"] = np.where(df["Tenure"] > 10, np.median(df["Tenure"])
df["Tenure"]
```

```
0      2
1      1
2      8
3      1
4      2
...
9995   5
9996  10
9997   7
9998   3
```

```
9999      4
Name: Tenure, Length: 10000, dtype: object
```

```
1. Check for Categorical columns and perform encoding
pd.get_dummies(df, columns=["Gender", "Age"], prefix=["Age",
"Gender"]).head()
```

RowNumber	CustomerId	Surname	CreditScore	Geography	Tenure
0	1	Hargrave	619	France	2
1	2	Hill	608	Spain	1
2	3	Onio	502	France	8
3	4	Boni	699	France	1
4	5	Mitchell	850	Spain	2

	NumOfProducts	HasCrCard	IsActiveMember	...	Gender_78	Gender_79
0	1	1	1	...	0	0
1	1	0	1	...	0	0
2	3	1	0	...	0	0
3	2	0	0	...	0	0
4	1	1	1	...	0	0

	Gender_80	Gender_81	Gender_82	Gender_83	Gender_84	Gender_85	\
0	0	0	0	0	0	0	
1	0	0	0	0	0	0	
2	0	0	0	0	0	0	
3	0	0	0	0	0	0	
4	0	0	0	0	0	0	

	Gender_88	Gender_92
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0

```
[5 rows x 84 columns]
```

```
1. Split the data into dependent and independent variables
```

8.1 Split the data into Independent variables

```
X = df.iloc[:, :-2].values
print(X)

[[1 15634602 'Hargrave' ... 1 1 1]
 [2 15647311 'Hill' ... 1 0 1]
 [3 15619304 'Onio' ... 3 1 0]
 ...
 [9998 15584532 'Liu' ... 1 0 1]
 [9999 15682355 'Sabbatini' ... 2 1 0]
 [10000 15628319 'Walker' ... 1 1 0]]
```

8.2 Split the data into Dependent variables

```
import pandas as pd
df = pd.read_csv('/content/Churn_Modelling.csv')
Y = df.iloc[:, -1].values
print(Y)
```

```
[1 0 1 ... 1 1 0]
```

1. Scale the independent variables

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
```

```
df[["RowNumber"]]=scaler.fit_transform(df[["RowNumber"]])
print(df)
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender
Age \						
0	0.0000	15634602	Hargrave	619	France	Female
42						
1	0.0001	15647311	Hill	608	Spain	Female
41						
2	0.0002	15619304	Onio	502	France	Female
42						
3	0.0003	15701354	Boni	699	France	Female
39						
4	0.0004	15737888	Mitchell	850	Spain	Female
43						
...
...						
9995	0.9996	15606229	Obijiaku	771	France	Male
39						
9996	0.9997	15569892	Johnstone	516	France	Male
35						
9997	0.9998	15584532	Liu	709	France	Female
36						
9998	0.9999	15682355	Sabbatini	772	Germany	Male
42						
9999	1.0000	15628319	Walker	792	France	Female

	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	\
0	2	0.00	1	1		1
1	1	83807.86	1	0		1
2	8	159660.80	3	1		0
3	1	0.00	2	0		0
4	2	125510.82	1	1		1
...
9995	5	0.00	2	1		0
9996	10	57369.61	1	1		1
9997	7	0.00	1	0		1
9998	3	75075.31	2	1		0
9999	4	130142.79	1	1		0

	EstimatedSalary	Exited
0	101348.88	1
1	112542.58	0
2	113931.57	1
3	93826.63	0
4	79084.10	0
...
9995	96270.64	0
9996	101699.77	0
9997	42085.58	1
9998	92888.52	1
9999	38190.78	0

[10000 rows x 14 columns]

1. Split the data into training and testing

```

from sklearn.model_selection import train_test_split
train_size=0.8
X = df.drop(columns = ['Tenure']).copy()
y = df['Tenure']
X_train, X_rem, y_train, y_rem = train_test_split(X,y, train_size=0.8)
test_size = 0.5
X_valid, X_test, y_valid, y_test = train_test_split(X_rem,y_rem,
test_size=0.5)
print(X_train.shape), print(y_train.shape)
print(X_valid.shape), print(y_valid.shape)
print(X_test.shape), print(y_test.shape)

(8000, 13)
(8000,)
(1000, 13)
(1000,)
(1000, 13)
(1000,)

```

(None, None)