

1.DOWNLOAD THE DATASET

```
import pandas as pd
import numpy as np
```

1. LOAD THE DATASET

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

```
df.head()
```

	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

	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	

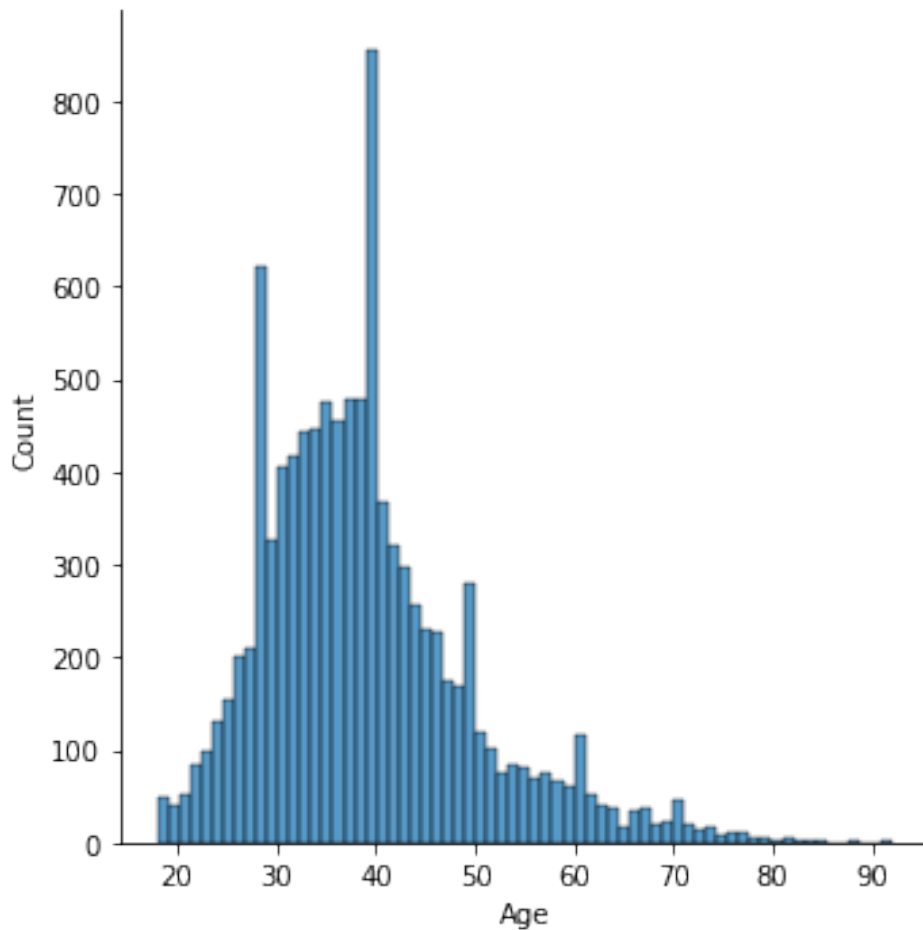
	EstimatedSalary	Exited
0	101348.88	1
1	112542.58	0
2	113931.57	1
3	93826.63	0
4	79084.10	0

3.VISUALIZATION

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
#Univariate Analysis
sns.displot(df['Age'])
```

```
<seaborn.axisgrid.FacetGrid at 0x7f9e4a1e0c50>
```



#Bivariate Analysis

```
sns.scatterplot(df['Tenure'],df['Age'])
```

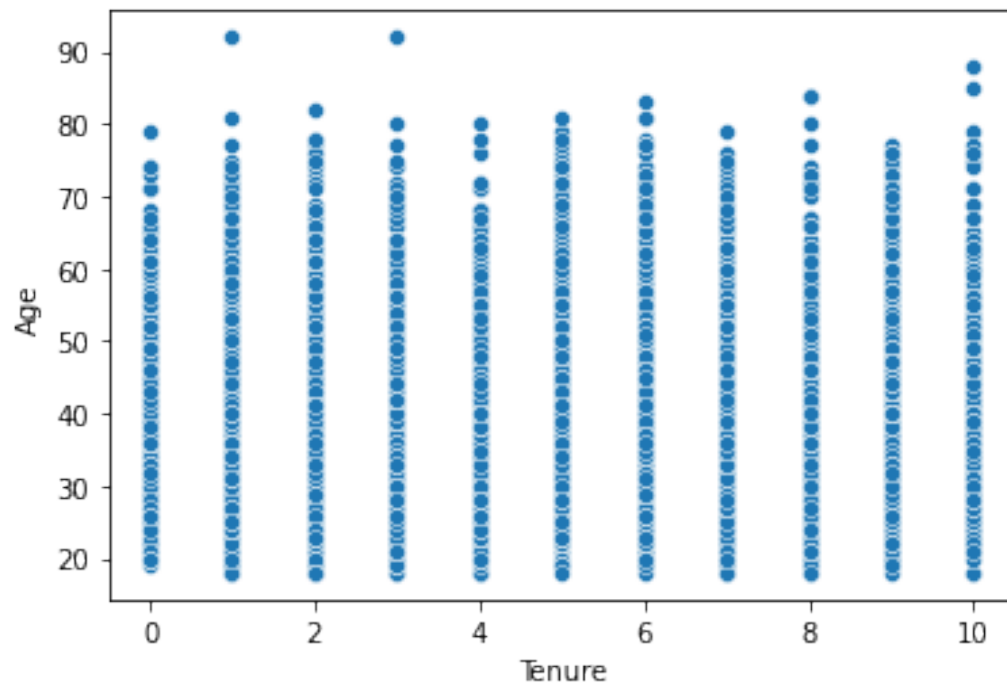
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:

FutureWarning: Pass the following variables as keyword args: x, y.

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

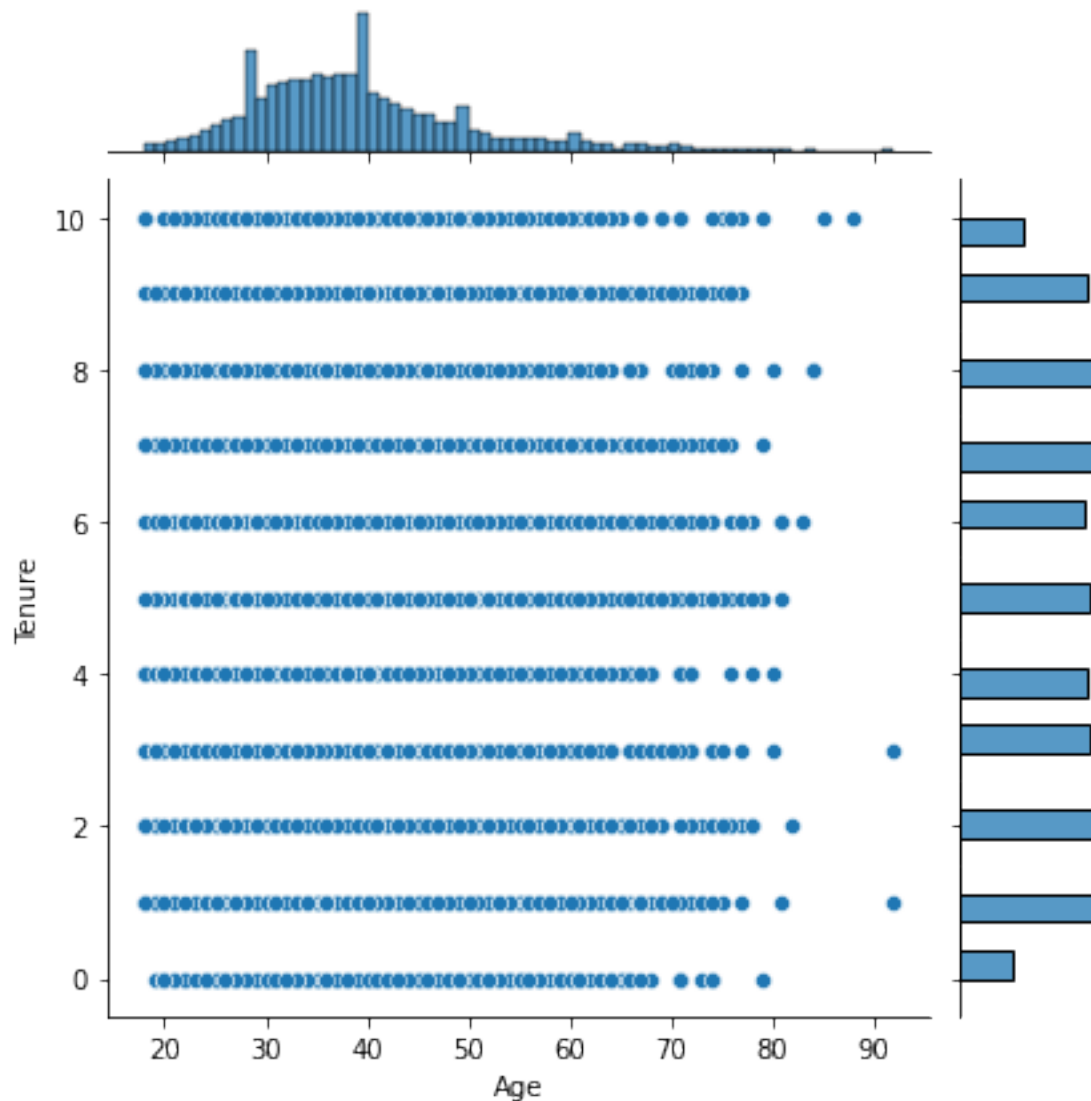
<matplotlib.axes._subplots.AxesSubplot at 0x7f9e3a934210>



```
#Multivariate Analysis
```

```
sns.jointplot(x='Age',y='Tenure',data=df)
```

```
<seaborn.axisgrid.JointGrid at 0x7f9e3a4665d0>
```



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

	count	mean	std	min	\
RowNumber	10000.0	5.000500e+03	2886.895680	1.00	
CustomerId	10000.0	1.569094e+07	71936.186123	15565701.00	
CreditScore	10000.0	6.505288e+02	96.653299	350.00	
Age	10000.0	3.892180e+01	10.487806	18.00	
Tenure	10000.0	5.012800e+00	2.892174	0.00	
Balance	10000.0	7.648589e+04	62397.405202	0.00	
NumOfProducts	10000.0	1.530200e+00	0.581654	1.00	
HasCrCard	10000.0	7.055000e-01	0.455840	0.00	
IsActiveMember	10000.0	5.151000e-01	0.499797	0.00	
EstimatedSalary	10000.0	1.000902e+05	57510.492818	11.58	
Exited	10000.0	2.037000e-01	0.402769	0.00	

	25%	50%	75%	max
RowNumber	2500.75	5.000500e+03	7.500250e+03	10000.00
CustomerId	15628528.25	1.569074e+07	1.575323e+07	15815690.00
CreditScore	584.00	6.520000e+02	7.180000e+02	850.00
Age	32.00	3.700000e+01	4.400000e+01	92.00
Tenure	3.00	5.000000e+00	7.000000e+00	10.00
Balance	0.00	9.719854e+04	1.276442e+05	250898.09
NumOfProducts	1.00	1.000000e+00	2.000000e+00	4.00
HasCrCard	0.00	1.000000e+00	1.000000e+00	1.00
IsActiveMember	0.00	1.000000e+00	1.000000e+00	1.00
EstimatedSalary	51002.11	1.001939e+05	1.493882e+05	199992.48
Exited	0.00	0.000000e+00	0.000000e+00	1.00

1. Handle the Missing values

```
df.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
```

1. Find the outliers and replace the outliers

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

```
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
print(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

```
outliers = df[((df<(Q1-1.5*IQR)) | (df>(Q3+1.5*IQR)))]
outliers
```

```

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1:
FutureWarning: Automatic reindexing on DataFrame vs Series comparisons
is deprecated and will raise ValueError in a future version. Do
`left, right = left.align(right, axis=1, copy=False)` before e.g.
`left == right`
    """Entry point for launching an IPython kernel.

```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age
\ 0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...
9995	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9996	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9997	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9998	NaN	NaN	NaN	NaN	NaN	NaN	NaN

9999	NaN	NaN	NaN	NaN	NaN	NaN	NaN
------	-----	-----	-----	-----	-----	-----	-----

	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	\
0	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN
...
9995	NaN	NaN	NaN	NaN	NaN	NaN
9996	NaN	NaN	NaN	NaN	NaN	NaN
9997	NaN	NaN	NaN	NaN	NaN	NaN
9998	NaN	NaN	NaN	NaN	NaN	NaN
9999	NaN	NaN	NaN	NaN	NaN	NaN

	EstimatedSalary	Exited
0	NaN	1.0
1	NaN	NaN
2	NaN	1.0
3	NaN	NaN
4	NaN	NaN
...
9995	NaN	NaN
9996	NaN	NaN
9997	NaN	1.0
9998	NaN	1.0
9999	NaN	NaN

[10000 rows x 14 columns]

1. Check for Categorical columns and perform encoding.
df.dtypes

```

RowNumber      int64
CustomerId     int64
Surname        object
CreditScore    int64
Geography      object
Gender         object
Age           int64
Tenure        int64
Balance       float64
NumOfProducts int64
HasCrCard     int64
IsActiveMember int64
EstimatedSalary float64
Exited        int64
dtype: object

```

```
obj_df = df.select_dtypes(include=['object']).copy()
obj_df.head()
```

	Surname	Geography	Gender
0	Hargrave	France	Female
1	Hill	Spain	Female
2	Onio	France	Female
3	Boni	France	Female
4	Mitchell	Spain	Female

8. Split the data into dependent and independent variables.

```
x = df.iloc[:,0:13].values;
y = df.iloc[:,13:14].values;
```

x

```
array([[1, 15634602, 'Hargrave', ..., 1, 1, 101348.88],
       [2, 15647311, 'Hill', ..., 0, 1, 112542.58],
       [3, 15619304, 'Onio', ..., 1, 0, 113931.57],
       ...,
       [9998, 15584532, 'Liu', ..., 0, 1, 42085.58],
       [9999, 15682355, 'Sabbatini', ..., 1, 0, 92888.52],
       [10000, 15628319, 'Walker', ..., 1, 0, 38190.78]],
      dtype=object)
```

y

```
array([[1],
       [0],
       [1],
       ...,
       [1],
       [1],
       [0]])
```

1. Scale the independent values

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
scaler.fit_transform(y)
```

```
array([[ 1.97716468],
       [-0.50577476],
       [ 1.97716468],
       ...,
       [ 1.97716468],
       [ 1.97716468],
       [-0.50577476]])
```

1. Split the data into training and testing


```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest =
train_test_split(x,y,test_size=0.3,random_state=0)
```