In [23]:
```
import pandas as pd   #for data analysis
import numpy as np    #for numerical calculations
```

In [24]:
```
mydata=pd.read_csv(r"E:\BOOKS pdf\EXTRA STUFFS\Internships\DataSets-master\ChurnData.csv")
```

In [25]:
```
mydata
```

Out[25]:

	tenure	age	address	income	ed	employ	equip	callcard	wireless	longmon	...	pager	internet	callwait	confer	ebill	loglong	logtoll	lninc	custcat	churn
0	11	33	7	136	5	5	0	1	1	4.40	...	1	0	1	1	0	1.482	3.033	4.913	4	1
1	33	33	12	33	2	0	0	0	0	9.45	...	0	0	0	0	0	2.246	3.240	3.497	1	1
2	23	30	9	30	1	2	0	0	0	6.30	...	0	0	0	1	0	1.841	3.240	3.401	3	0
3	38	35	5	76	2	10	1	1	1	6.05	...	1	1	1	1	1	1.800	3.807	4.331	4	0
4	7	35	14	80	2	15	0	1	0	7.10	...	0	0	1	1	0	1.960	3.091	4.382	3	0
5	68	52	17	120	1	24	0	1	0	20.70	...	0	0	0	0	0	3.030	3.240	4.787	1	0
6	42	40	7	37	2	8	1	1	1	8.25	...	0	1	1	1	1	2.110	3.157	3.611	4	0
7	9	21	1	17	2	2	0	0	0	2.90	...	0	0	0	0	0	1.065	3.240	2.833	1	0
8	35	50	26	140	2	21	0	1	0	6.50	...	0	0	1	1	0	1.872	3.314	4.942	3	0
9	49	51	27	63	4	19	0	1	0	12.85	...	0	1	1	0	1	2.553	3.248	4.143	2	0
10	56	52	28	49	2	12	0	1	0	24.75	...	0	0	0	0	0	3.209	3.240	3.892	2	0
11	47	40	16	127	4	12	1	1	0	19.70	...	0	1	0	0	1	2.981	3.240	4.844	2	0
12	56	50	1	80	2	24	0	1	1	28.80	...	1	0	1	1	0	3.360	4.016	4.382	4	0
13	69	51	11	438	4	23	1	1	0	29.00	...	1	1	0	1	0	3.367	3.240	6.082	4	0
14	16	27	5	37	3	5	0	0	0	6.00	...	0	0	0	0	0	1.792	3.240	3.611	1	0
15	4	35	16	161	5	6	1	0	1	3.40	...	1	1	1	1	1	1.224	3.168	5.081	4	1
16	27	51	3	80	5	11	1	0	0	7.10	...	1	1	0	0	1	1.960	3.240	4.382	2	0
17	52	61	3	53	5	1	1	1	1	12.25	...	0	1	0	1	1	2.506	3.240	3.970	2	0
18	64	25	4	76	3	2	1	1	0	24.05	...	0	0	0	1	1	3.180	3.240	4.331	3	0
19	12	24	2	19	1	0	0	1	0	4.00	...	0	0	1	1	0	1.386	3.209	2.944	3	1
20	35	61	23	41	2	11	0	1	0	9.60	...	0	0	0	0	0	2.262	3.240	3.714	1	0
21	13	54	2	31	4	2	0	0	0	5.85	...	0	1	0	0	1	1.766	3.240	3.434	1	0
22	45	22	2	36	4	0	1	0	0	9.95	...	0	0	0	0	0	2.298	2.691	3.584	2	1
23	3	37	13	24	1	3	0	0	0	2.00	...	0	0	0	1	1	0.693	3.240	3.178	1	0
24	53	22	1	25	4	0	1	1	0	12.05	...	0	1	0	0	1	2.489	3.240	3.219	2	0

	tenure	age	address	income	ed	employ	equip	callcard	wireless	longmon	...	pager	internet	callwait	confer	ebill	loglong	logtoll	lninc	custcat	churn
25	17	42	6	131	5	6	1	0	1	5.80	...	0	1	0	0	1	1.758	3.240	4.875	2	1
26	59	43	4	101	2	22	0	1	0	13.65	...	0	0	0	1	0	2.614	3.240	4.615	2	0
27	57	37	11	108	4	9	1	1	0	21.80	...	0	1	1	1	0	3.082	3.305	4.682	3	0
28	3	24	2	20	2	3	0	1	0	3.35	...	0	1	1	1	0	1.209	3.114	2.996	3	0
29	4	47	5	123	4	11	1	1	0	2.50	...	0	1	0	0	1	0.916	3.240	4.812	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
170	16	49	17	41	2	5	1	0	1	4.10	...	1	1	0	0	0	1.411	2.639	3.714	4	0
171	59	26	3	41	4	1	1	1	1	12.65	...	0	1	0	0	0	2.538	3.240	3.714	2	1
172	9	40	13	38	4	7	1	1	1	3.35	...	1	1	1	1	1	1.209	3.045	3.638	4	1
173	12	55	13	36	1	5	1	0	0	5.95	...	0	1	0	0	0	1.783	3.240	3.584	2	1
174	3	32	4	58	2	11	1	1	1	2.75	...	1	0	0	0	1	1.012	2.757	4.060	4	1
175	52	39	6	119	3	18	0	1	0	10.50	...	1	0	0	0	1	2.351	3.219	4.779	1	0
176	18	69	28	11	1	17	0	0	0	3.85	...	0	0	0	0	0	1.348	3.240	2.398	1	0
177	43	29	4	33	1	13	0	1	1	22.05	...	0	0	1	1	0	3.093	2.931	3.497	3	0
178	37	33	4	41	3	8	1	0	0	9.20	...	0	0	0	0	1	2.219	3.240	3.714	2	1
179	51	46	8	107	2	21	0	1	1	17.30	...	1	0	1	1	0	2.851	3.248	4.673	3	0
180	56	53	23	100	5	14	1	1	0	14.15	...	0	1	0	0	1	2.650	3.240	4.605	2	0
181	72	55	24	82	3	25	1	1	0	62.30	...	0	0	1	0	1	4.132	3.240	4.407	2	0
182	32	44	10	201	2	24	0	1	0	7.65	...	1	0	1	1	0	2.035	3.332	5.303	3	0
183	51	49	29	45	1	16	0	1	0	15.70	...	0	0	1	0	0	2.754	3.240	3.807	1	0
184	26	55	13	61	1	26	0	1	0	4.25	...	0	0	1	1	0	1.447	3.367	4.111	3	0
185	34	40	21	23	4	9	1	1	1	5.95	...	1	1	1	1	1	1.783	3.248	3.135	4	0
186	20	25	4	33	4	0	0	1	1	4.55	...	0	1	0	0	1	1.515	2.773	3.497	1	0
187	58	36	13	39	2	8	0	1	1	16.40	...	1	1	1	1	1	2.797	3.644	3.664	4	1
188	25	38	19	56	1	19	1	1	1	10.55	...	0	0	0	1	0	2.356	3.240	4.025	3	0
189	66	50	2	333	5	24	0	1	0	10.30	...	1	1	0	0	0	2.332	3.240	5.808	2	0
190	71	48	25	288	3	19	0	1	0	30.90	...	0	0	0	0	0	3.431	3.240	5.663	1	0

	tenure	age	address	income	ed	employ	equip	callcard	wireless	longmon	...	pager	internet	callwait	confer	ebill	loglong	logtoll	lninc	custcat	churn
191	6	20	0	25	2	0	0	1	0	1.90	...	0	0	1	1	0	0.642	3.033	3.219	3	0
192	26	30	4	76	3	7	1	0	0	9.45	...	1	0	0	0	1	2.246	3.240	4.331	1	0
193	61	52	21	82	1	18	0	1	0	12.10	...	0	0	0	1	0	2.493	3.240	4.407	3	0
194	57	60	20	14	2	27	0	1	0	16.10	...	0	0	1	1	0	2.779	2.639	2.639	3	0
195	55	44	24	83	1	23	0	1	0	17.35	...	0	0	0	1	0	2.854	3.199	4.419	3	0
196	34	23	3	24	1	7	0	1	0	6.00	...	0	0	1	1	0	1.792	3.332	3.178	3	0
197	6	32	10	47	1	10	0	1	0	3.85	...	0	0	1	1	0	1.348	3.168	3.850	3	0
198	24	30	0	25	4	5	0	1	1	8.70	...	1	1	1	1	1	2.163	3.866	3.219	4	1
199	61	50	16	190	2	22	1	1	1	16.85	...	0	1	0	0	1	2.824	3.240	5.247	2	0

200 rows × 28 columns

In [26]:
```
mydata.shape
```

Out[26]:
```
(200, 28)
```

In [27]:
```
mydata.keys()
```

Out[27]:
```
Index(['tenure', 'age', 'address', 'income', 'ed', 'employ', 'equip',
       'callcard', 'wireless', 'longmon', 'tollmon', 'equipmon', 'cardmon',
       'wiremon', 'longten', 'tollten', 'cardten', 'voice', 'pager',
       'internet', 'callwait', 'confer', 'ebill', 'loglong', 'logtoll',
       'lninc', 'custcat', 'churn'],
      dtype='object')
```

In [28]:
```
mydata.head(1)
```

Out[28]:

	tenure	age	address	income	ed	employ	equip	callcard	wireless	longmon	...	pager	internet	callwait	confer	ebill	loglong	logtoll	lninc	custcat	churn
0	11	33	7	136	5	5	0	1	1	4.4	...	1	0	1	1	0	1.482	3.033	4.913	4	1

1 rows × 28 columns

In [29]:
```
#Selecting the data which is useful and use it further for analysis
X_data=mydata[["tenure","age","address","income","ed","employ","equip","callcard","wireless"]]
```

Positive correlation is a relation between two variables means both the variables move in same direction. If one variable increases than other variable also increases and if it is decreases than it also decreases . A negative correlation is just inverse of this...

In [30]:
```
mydata.corr().T  #is also used to skip the repeated values not in use
```

Out[30]:

	tenure	age	address	income	ed	employ	equip	callcard	wireless	longmon	...	pager	internet	callwait	confer	ebill	loglon
tenure	1.000000	0.431802	0.456328	0.109383	-0.070503	0.445755	-0.117102	0.426530	-0.070590	0.763134	...	0.018791	-0.164921	-0.009747	0.080650	-0.099128	0.8643
age	0.431802	1.000000	0.746566	0.211275	-0.071509	0.622553	-0.071357	0.170404	-0.065527	0.373547	...	0.006803	-0.078395	0.020002	0.030625	-0.048279	0.3794
address	0.456328	0.746566	1.000000	0.132807	-0.145550	0.520926	-0.148977	0.209204	-0.146478	0.421782	...	-0.105812	-0.191058	-0.019967	-0.030494	-0.172171	0.4093
income	0.109383	0.211275	0.132807	1.000000	0.141241	0.345161	-0.010741	-0.019969	-0.029635	0.041808	...	0.056977	0.102809	0.081133	-0.031556	-0.041392	0.0655
ed	-0.070503	-0.071509	-0.145550	0.141241	1.000000	-0.213886	0.488041	-0.071178	0.267670	-0.072735	...	0.258698	0.552996	-0.016247	-0.132215	0.427315	-0.0545
employ	0.445755	0.622553	0.520926	0.345161	-0.213886	1.000000	-0.174470	0.266612	-0.101187	0.363386	...	0.038381	-0.250044	0.119708	0.173247	-0.151965	0.3771
equip	-0.117102	-0.071357	-0.148977	-0.010741	0.488041	-0.174470	1.000000	-0.087051	0.386735	-0.097618	...	0.308633	0.623509	-0.034021	-0.103499	0.603133	-0.1130
callcard	0.426530	0.170404	0.209204	-0.019969	-0.071178	0.266612	-0.087051	1.000000	0.220118	0.322514	...	0.251069	-0.067146	0.370878	0.311056	-0.045058	0.3510
wireless	-0.070590	-0.065527	-0.146478	-0.029635	0.267670	-0.101187	0.386735	0.220118	1.000000	-0.073043	...	0.667535	0.343631	0.389670	0.382925	0.321433	-0.0426
longmon	0.763134	0.373547	0.421782	0.041808	-0.072735	0.363386	-0.097618	0.322514	-0.073043	1.000000	...	-0.001372	-0.223929	0.032913	0.060614	-0.124605	0.9016
tollmon	0.100214	0.053595	-0.015564	-0.003701	-0.000712	0.117962	-0.090516	0.352994	0.462751	0.083771	...	0.499124	0.036393	0.678454	0.651039	-0.032998	0.0822
equipmon	-0.043274	-0.030402	-0.125212	0.000275	0.492999	-0.141651	0.941065	0.026955	0.527573	-0.070876	...	0.461458	0.631338	0.104341	0.056846	0.602010	-0.0624
cardmon	0.489857	0.203202	0.297395	-0.013513	-0.070990	0.219413	-0.056590	0.629955	0.195387	0.493439	...	0.216653	-0.129166	0.222755	0.232357	-0.036768	0.4675
wiremon	0.043067	-0.003685	-0.113143	-0.027998	0.288571	-0.045539	0.367579	0.259638	0.885879	-0.020392	...	0.704724	0.359273	0.435465	0.426042	0.347533	0.0236
longten	0.796355	0.393370	0.447256	0.048374	-0.083032	0.385254	-0.098271	0.342553	-0.088461	0.984808	...	-0.011369	-0.217916	0.013932	0.058190	-0.119885	0.8626
tollten	0.398869	0.187221	0.129141	-0.000262	-0.016257	0.240498	-0.097151	0.292555	0.328256	0.338036	...	0.390620	-0.017063	0.507160	0.515239	-0.064126	0.3478
cardten	0.676920	0.336903	0.406964	0.019358	-0.069499	0.309183	-0.053837	0.448182	0.047773	0.673179	...	0.097579	-0.154030	0.065429	0.084221	-0.051008	0.6299
voice	-0.050686	0.019555	-0.069627	0.085525	0.233628	-0.035922	0.264481	0.226109	0.649721	-0.080503	...	0.608335	0.243845	0.399721	0.392889	0.288019	-0.0585
pager	0.018791	0.006803	-0.105812	0.056977	0.258698	0.038381	0.308633	0.251069	0.667535	-0.001372	...	1.000000	0.266193	0.426690	0.352743	0.288752	0.0157
internet	-0.164921	-0.078395	-0.191058	0.102809	0.552996	-0.250044	0.623509	-0.067146	0.343631	-0.223929	...	0.266193	1.000000	0.019419	-0.029911	0.512987	-0.2022
callwait	-0.009747	0.020002	-0.019967	0.081133	-0.016247	0.119708	-0.034021	0.370878	0.389670	0.032913	...	0.426690	0.019419	1.000000	0.566910	-0.061492	0.0285
confer	0.080650	0.030625	-0.030494	-0.031556	-0.132215	0.173247	-0.103499	0.311056	0.382925	0.060614	...	0.352743	-0.029911	0.566910	1.000000	-0.090542	0.0676
ebill	-0.099128	-0.048279	-0.172171	-0.041392	0.427315	-0.151965	0.603133	-0.045058	0.321433	-0.124605	...	0.288752	0.512987	-0.061492	-0.090542	1.000000	-0.1333
loglong	0.864388	0.379413	0.409357	0.065595	-0.054581	0.377186	-0.113065	0.351030	-0.042637	0.901631	...	0.015771	-0.202230	0.028577	0.067662	-0.133310	1.0000
logtoll	0.310045	0.093600	0.018386	-0.156498	-0.007227	0.068718	-0.027882	0.080060	0.178317	0.247302	...	0.231647	0.038008	0.147224	0.235203	0.024784	0.2972

	tenure	age	address	income	ed	employ	equip	callcard	wireless	longmon	...	pager	internet	callwait	confer	ebill	loglor
lninc	0.246353	0.313359	0.212929	0.680313	0.206718	0.540052	0.083494	0.156920	0.033558	0.122550	...	0.159049	0.067611	0.069897	-0.004963	0.039137	0.1824
custcat	0.134237	0.041055	-0.016841	0.030725	0.013127	0.131292	0.174955	0.407553	0.598156	0.072519	...	0.653884	0.160904	0.687960	0.673629	0.104775	0.1204
churn	-0.376860	-0.287697	-0.260659	-0.090790	0.216112	-0.337969	0.275284	-0.311451	0.174356	-0.292026	...	0.124623	0.254838	-0.052885	-0.081361	0.254838	-0.3368

28 rows × 28 columns

In [31]:
```
Y_data=mydata["churn"]   #take the churn data split
```

In [32]:
```
Y_data.head(1)
```

Out[32]:
```
0    1
Name: churn, dtype: int64
```

In [33]:
```
print(type(X_data))
print(type(Y_data))
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
```

In [34]:
```
XA=X_data.values
YA=Y_data.values
```

In [35]:
```
print(type(XA))
print(type(YA))
```

```
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

In [36]:
```
X_data.describe().T  #descriptive variable
```

Out[36]:

	count	mean	std	min	25%	50%	75%	max
tenure	200.0	35.505	21.640971	1.0	16.75	33.5	55.25	72.0
age	200.0	41.165	13.076803	19.0	31.00	40.0	51.00	76.0
address	200.0	11.650	10.158419	0.0	3.00	9.0	18.00	48.0
income	200.0	75.130	128.430468	9.0	31.00	48.0	80.00	1668.0
ed	200.0	2.825	1.285550	1.0	2.00	3.0	4.00	5.0
employ	200.0	10.225	8.957430	0.0	3.00	7.5	17.00	44.0
equip	200.0	0.425	0.495584	0.0	0.00	0.0	1.00	1.0
callcard	200.0	0.705	0.457187	0.0	0.00	1.0	1.00	1.0
wireless	200.0	0.290	0.454901	0.0	0.00	0.0	1.00	1.0

In [37]:
```
print(type(XA))
print(type(YA))
```

```
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

In [38]:
```
print("mean of XA {} and Std of XA {}".format(XA.mean(),XA.std()))
```

```
mean of XA 19.768888888888889 and Std of XA 50.071370494623736
```

In [39]:
```
#feature scaling all values of max in one
```

In [41]:
```
from sklearn.preprocessing import StandardScaler
```

In [42]:
```
XA=StandardScaler().fit(XA).transform(XA) #you may have to update upper to run
print(type(XA))
```

```
<class 'numpy.ndarray'>
```

```
C:\Users\ANKIT SINGH\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning: Data with input dtype int64 was converted t
o float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
C:\Users\ANKIT SINGH\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning: Data with input dtype int64 was converted t
o float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

In [43]:
```
Xm=round(XA.mean())
Xstd=XA.std()
```

In [44]:
```
print("mean of XA {} and Std of XA {}".format(Xm,Xstd))
```

```
mean of XA -0.0 and Std of XA 1.0
```

In [46]:
```
#XA
```

In [47]:
```
#split the train and test
```

In [50]:
```
Xtrain=XA[:160]
Xtest=XA[160:]
Ytrain=YA[:160]
Ytest=YA[160:]
```

In [51]:
```
from sklearn.linear_model import LogisticRegression
```

In [54]:
```
#when approximation available than assumption made
trainer=LogisticRegression(solver='lbfgs')
```

In [55]:
```
learner=trainer.fit(Xtrain,Ytrain)
```

In [56]:
```
YA=Ytest   #Xtest
Yp=learner.predict(Xtest)
```

In [57]:
```
Yp
```

Out[57]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [58]:
```
YA
```

Out[58]:
```
array([1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0], dtype=int64)
```

In [60]:
```
#1st value in list 1st ke hone ki probability
#2nd value in list 2nd ke hone ki probability
learner.predict_proba(Xtest)
```

Out[60]:
```
array([[0.56861894, 0.43138106],
       [0.87514647, 0.12485353],
       [0.85375126, 0.14624874],
       [0.7867313 , 0.2132687 ],
       [0.97477988, 0.02522012],
       [0.85682198, 0.14317802],
       [0.97359155, 0.02640845],
       [0.40833143, 0.59166857],
       [0.93339964, 0.06660036],
       [0.8602275 , 0.1397725 ],
       [0.32436542, 0.67563458],
       [0.68524198, 0.31475802],
       [0.41618619, 0.58381381],
       [0.52957088, 0.47042912],
       [0.48309057, 0.51690943],
       [0.95052113, 0.04947887],
       [0.82411268, 0.17588732],
       [0.88336326, 0.11663674],
       [0.53810824, 0.46189176],
       [0.94540193, 0.05459807],
       [0.87492369, 0.12507631],
       [0.96217219, 0.03782781],
       [0.94482184, 0.05517816],
       [0.95336597, 0.04663403],
       [0.95769376, 0.04230624],
       [0.58423353, 0.41576647],
       [0.54284667, 0.45715333],
       [0.89671182, 0.10328818],
       [0.72356638, 0.27643362],
       [0.97721969, 0.02278031],
       [0.9719511 , 0.0280489 ],
       [0.63580111, 0.36419889],
       [0.42525186, 0.57474814],
       [0.97523915, 0.02476085],
       [0.98064944, 0.01935056],
       [0.96646491, 0.03353509],
       [0.86901205, 0.13098795],
       [0.78110976, 0.21889024],
```

```
       [0.66991219, 0.33008781],
       [0.92396559, 0.07603441]])
```

In [61]:
```
yprob=learner.predict_proba(Xtest)
yprob[:2]
```

Out[61]:
```
array([[0.56861894, 0.43138106],
       [0.87514647, 0.12485353]])
```

In [62]:
```
Y_1X=yprob[:,0]
Y_0X=yprob[:,1]
```

In [63]:
```
Table=pd.DataFrame({"Y(1|X)":Y_1X,"Y(0|X)":Y_0X})
Table
```

Out[63]:

	Y(1|X)	Y(0|X)
0	0.568619	0.431381
1	0.875146	0.124854
2	0.853751	0.146249
3	0.786731	0.213269
4	0.974780	0.025220
5	0.856822	0.143178
6	0.973592	0.026408
7	0.408331	0.591669
8	0.933400	0.066600
9	0.860227	0.139773
10	0.324365	0.675635
11	0.685242	0.314758
12	0.416186	0.583814
13	0.529571	0.470429
14	0.483091	0.516909
15	0.950521	0.049479
16	0.824113	0.175887
17	0.883363	0.116637
18	0.538108	0.461892
19	0.945402	0.054598
20	0.874924	0.125076
21	0.962172	0.037828
22	0.944822	0.055178
23	0.953366	0.046634

	Y(1|X)	Y(0|X)
24	0.957694	0.042306
25	0.584234	0.415766
26	0.542847	0.457153
27	0.896712	0.103288
28	0.723566	0.276434
29	0.977220	0.022780
30	0.971951	0.028049
31	0.635801	0.364199
32	0.425252	0.574748
33	0.975239	0.024761
34	0.980649	0.019351
35	0.966465	0.033535
36	0.869012	0.130988
37	0.781110	0.218890
38	0.669912	0.330088
39	0.923966	0.076034

In [64]:
```
from sklearn.metrics import jaccard_similarity_score,accuracy_score
```

In [65]:
```
jss=jaccard_similarity_score(Ytest,Yp)
acc=accuracy_score(Ytest,Yp)
```

In [66]:
```
acc
```

Out[66]: 0.775

In [67]:
```
jss
```

Out[67]: 0.775