BANK LOAN PREDICTION

This case is about a bank (Thera Bank) whose management wants to explore ways of converting its liability customers to personal loan customers (while retaining them as depositors). A campaign that the bank ran last year for liability customers showed a healthy conversion rate of over 9% success. This has encouraged the retail marketing department to devise campaigns with better target marketing to increase the success ratio with minimal budget.

ATTRIBUTE INFORMATION:

ID :Customer ID
Age : Customer's age in completed years
Experience : #years of professional experience
Income : Annual income of the customer
ZIP Code : Home Address ZIP code.
Family : family size of the customer
CCAvg : Avg. spending on credit cards per month
Education : Education Level. 1: Undergrad; 2: Graduate; 3: Advanced/Professional
Mortgage : Value of house mortgage if any.
Personal Loan : Did this customer accept the personal loan offered in the last campaign?
Securities Account : Does the customer have a securities account with the bank?
CD Account : Does the customer have a certificate of deposit (CD) account with the bank?
Online : Does the customer use internet banking facilities?
Credit card : Does the customer use a credit card issued by UniversalBank?

In [1]:
```
import  numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
```

In [2]:
```
data=pd.read_csv("Bank_Personal_Loan_Modelling.csv")
```

In [3]:
```
data.head()
```

Out[3]:

	ID	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Online	CreditCard
0	1	25	1	49	91107	4	1.6	1	0	0	1	0	0	0
1	2	45	19	34	90089	3	1.5	1	0	0	1	0	0	0
2	3	39	15	11	94720	1	1.0	1	0	0	0	0	0	0
3	4	35	9	100	94112	1	2.7	2	0	0	0	0	0	0
4	5	35	8	45	91330	4	1.0	2	0	0	0	0	0	1

In [4]:
```
data.shape
```

Out[4]:
```
(5000, 14)
```

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   ID                  5000 non-null   int64  
 1   Age                 5000 non-null   int64  
 2   Experience          5000 non-null   int64  
 3   Income              5000 non-null   int64  
 4   ZIP Code            5000 non-null   int64  
 5   Family              5000 non-null   int64  
 6   CCAvg               5000 non-null   float64
 7   Education           5000 non-null   int64  
 8   Mortgage            5000 non-null   int64  
 9   Personal Loan       5000 non-null   int64  
 10  Securities Account  5000 non-null   int64  
 11  CD Account          5000 non-null   int64  
 12  Online              5000 non-null   int64  
 13  CreditCard          5000 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 546.9 KB
```

CHECKING MISSING VALUE

In [13]:
```
data.isnull().sum()
```

Out[13]:
```
ID                    0
Age                   0
Experience            0
Income                0
ZIP Code              0
Family                0
CCAvg                 0
Education             0
Mortgage              0
Personal Loan         0
Securities Account    0
CD Account            0
Online                0
CreditCard            0
dtype: int64
```

In [12]:
```
data=data.drop('ID',axis=1)
```

In [13]:
```
data.head(20)
```

Out[13]:

	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Online	CreditCard
0	25	1	49	91107	4	1.6	1	0	0	1	0	0	0
1	45	19	34	90089	3	1.5	1	0	0	1	0	0	0
2	39	15	11	94720	1	1.0	1	0	0	0	0	0	0
3	35	9	100	94112	1	2.7	2	0	0	0	0	0	0
4	35	8	45	91330	4	1.0	2	0	0	0	0	0	1
5	37	13	29	92121	4	0.4	2	155	0	0	0	1	0
6	53	27	72	91711	2	1.5	2	0	0	0	0	1	0
7	50	24	22	93943	1	0.3	3	0	0	0	0	0	1
8	35	10	81	90089	3	0.6	2	104	0	0	0	1	0
9	34	9	180	93023	1	8.9	3	0	1	0	0	0	0
10	65	39	105	94710	4	2.4	3	0	0	0	0	0	0
11	29	5	45	90277	3	0.1	2	0	0	0	1	0	0
12	48	23	114	93106	2	3.8	3	0	0	1	0	0	0
13	59	32	40	94920	4	2.5	2	0	0	0	0	1	0
14	67	41	112	91741	1	2.0	1	0	0	1	0	0	1
15	60	30	22	95054	1	1.5	3	0	0	0	0	1	1
16	38	14	130	95010	4	4.7	3	134	1	0	0	0	0
17	42	18	81	94305	4	2.4	1	0	0	0	0	0	0
18	46	21	193	91604	2	8.1	3	0	1	0	0	0	0
19	55	28	21	94720	1	0.5	2	0	0	1	0	0	1

CORRELATION BETWEEN VARIABLES

In [14]:
```
data.corr()
```

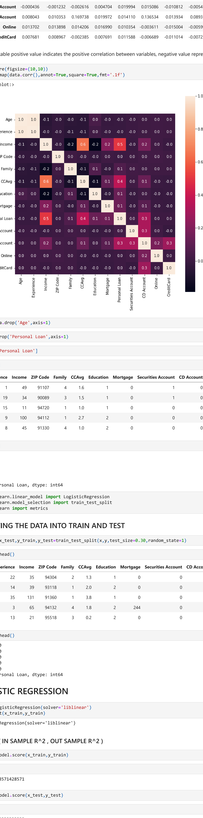
Out[14]:

	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Online	CreditCard
Age	1.000000	0.994215	-0.055269	-0.029216	-0.046418	-0.052012	0.041334	-0.012539	-0.007726	-0.000436	0.008043	0.013702	0.007681
Experience	0.994215	1.000000	-0.046574	-0.028626	-0.052563	-0.050077	0.013152	-0.010582	-0.007413	-0.001232	0.010353	0.013898	0.008967
Income	-0.055269	-0.046574	1.000000	-0.016410	-0.157501	0.645984	-0.187524	0.206806	0.502462	-0.002616	0.169738	0.014206	-0.002385
ZIP Code	-0.029216	-0.028626	-0.016410	1.000000	0.011778	-0.004061	-0.017377	0.007383	0.000107	0.004704	0.019972	0.016990	0.007691
Family	-0.046418	-0.052563	-0.157501	0.011778	1.000000	-0.109275	0.064570	-0.020445	-0.061367	0.019994	-0.014110	0.010154	0.011588
CCAvg	-0.052012	-0.050077	0.645984	-0.004061	-0.109275	1.000000	-0.136124	0.109905	0.366889	0.015086	0.136534	-0.003611	-0.006689
Education	0.041334	0.013152	-0.187524	-0.017377	0.064929	-0.136124	1.000000	0.033327	0.136722	-0.010812	0.013934	-0.015004	-0.011014
Mortgage	-0.012539	-0.010582	0.206806	0.007383	-0.020445	0.109905	0.033327	1.000000	0.142095	-0.005411	0.089311	-0.005995	-0.007231
Personal Loan	-0.007726	-0.007413	0.502462	0.000107	-0.061367	0.366889	0.136722	0.142095	1.000000	0.021954	0.316355	0.006278	0.002802
Securities Account	-0.000436	-0.001232	-0.002616	0.004704	0.019994	0.015086	-0.010812	-0.005411	0.021954	1.000000	0.317034	0.012627	-0.015028
CD Account	0.008043	0.010353	0.169738	0.019972	0.014110	0.136534	0.013934	0.089311	0.316355	0.317034	1.000000	0.175880	0.278644
Online	0.013702	0.013898	0.014206	0.016990	0.010154	-0.003611	-0.015004	-0.005995	0.006278	0.012627	0.175880	1.000000	0.004210
CreditCard	0.007681	0.008967	-0.002385	0.007691	0.011588	-0.006689	-0.011014	-0.007231	0.002802	-0.015028	0.278644	0.004210	1.000000

In above table positive value indicates the positive correlation between variables, negative value represents negative correlation among variables and zero represents no relationship between variables.

In [20]:
```
plt.figure(figsize=(10,10))
sns.heatmap(data.corr(),annot=True,square=True,fmt='.1f')
<AxesSubplot:>
```

Out[20]:



In [15]:
```
data=data.drop('Age',axis=1)
```

In [16]:
```
e=data.drop('Personal Loan',axis=1)
```

In [17]:
```
y=data['Personal Loan']
```

In [18]:
```
x.head()
```

Out[18]:

	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Securities Account	CD Account	Online	CreditCard
0	1	49	91107	4	1.6	1	0	1	0	0	0
1	19	34	90089	3	1.5	1	0	1	0	0	0
2	15	11	94720	1	1.0	1	0	0	0	0	0
3	9	100	94112	1	2.7	2	0	0	0	0	0
4	8	45	91330	4	1.0	2	0	0	0	0	1

In [19]:
```
y.head()
```

Out[19]:
```
0    0
1    0
2    0
3    0
4    0
Name: Personal Loan, dtype: int64
```

In [20]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

SPLITTING THE DATA INTO TRAIN AND TEST

In [21]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=1)
```

In [22]:
```
x_train.head()
```

Out[22]:

	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Securities Account	CD Account	Online	CreditCard
1334	22	35	94304	2	1.3	1	0	0	0	1	0
4768	14	39	93118	1	2.0	2	0	0	0	1	0
65	35	131	91360	1	3.8	1	0	0	0	1	1
177	3	65	94132	4	1.8	2	244	0	0	0	0
4489	13	21	95518	3	0.2	2	0	0	0	1	0

In [23]:
```
y_train.head()
```

Out[23]:
```
1334    0
4768    0
65      0
177     0
4489    0
Name: Personal Loan, dtype: int64
```

LOGISTIC REGRESSION

In [24]:
```
model=LogisticRegression(solver='liblinear')
model.fit(x_train,y_train)
```

Out[24]:
```
LogisticRegression(solver='liblinear')
```

SCORE (IN SAMPLE R^2 , OUT SAMPLE R^2)

In [25]:
```
model1=model.score(x_train,y_train)
```

In [26]:
```
model1
```

Out[26]:
```
0.9091428571428571
```

In [27]:
```
model2=model.score(x_test,y_test)
```

In [28]:
```
model2
```

Out[28]:
```
0.9073333333333333
```

In [29]:
```
y_predict=model.predict(x_test)#for given x we have to predict y thats y we take x test
```

CONFUSION MATRIX

In [30]:
```
cm=metrics.confusion_matrix(y_test,y_predict)
```

In [31]:
```
cm
```

Out[31]:
```
array([[1338,   33],
       [ 106,  413]], dtype=int64)
```

NAIVE BAYES

In [32]:
```
from sklearn.naive_bayes import GaussianNB
```

In [33]:
```
gaussian=GaussianNB()
```

In [34]:
```
gaussian.fit(x_train,y_train)
```

Out[34]:
```
GaussianNB()
```

SCORE (IN SAMPLE R^2 , OUT SAMPLE R^2)

In [35]:
```
model3=gaussian.score(x_train,y_train)
```

In [36]:
```
model3
```

Out[36]:
```
0.8948571428571429
```

In [37]:
```
model4=gaussian.score(x_test,y_test)
```

In [38]:
```
model4
```

Out[38]:
```
0.8833333333333333
```

CONFUSION MATRIX

In [39]:
```
cm1=metrics.confusion_matrix(y_test,y_predict)
```

In [40]:
```
cm1
```

Out[40]:
```
array([[1338,   33],
       [ 106,  413]], dtype=int64)
```

K NEAREST NEIGHBOR CLASSIFIER

In [41]:
```
from scipy.stats import zscore
from sklearn.neighbors import KNeighborsClassifier
```

In [42]:
```
scaled=x.apply(zscore)
```

In [43]:
```
x_train,x_test,y_train,y_test=train_test_split(scaled,y,test_size=0.30,random_state=1)
```

In [44]:
```
knn=KNeighborsClassifier(n_neighbors=3,weights='distance')
knn.fit(x_train,y_train)
```

Out[44]:
```
KNeighborsClassifier(n_neighbors=3, weights='distance')
```

SCORE (IN SAMPLE R^2 , OUT SAMPLE R^2)

In [45]:
```
model5=knn.score(x_train,y_train)
```

In [46]:
```
model5
```

Out[46]:
```
1.0
```

In [47]:
```
model6=knn.score(x_test,y_test)
```

In [48]:
```
model6
```

Out[48]:
```
0.95
```

In [49]:
```
y_predict=knn.predict(x_test)
```

CONFUSION MATRIX

In [50]:
```
cm=metrics.confusion_matrix(y_test,y_predict)
```

In [51]:
```
cm
```

Out[51]:
```
array([[1339,   32],
       [  63,  86]], dtype=int64)
```

In [54]:
```
result=pd.DataFrame({'CLASSIFIER':['LOGISTIC REGRESSION','NAIVE BAYES','K NEAREST NEIGHBOR'],'SCORES':[0.9091428,0.8948571,1.0]})
```

In [55]:
```
result
```

Out[55]:

	CLASSIFIER	SCORES
0	LOGISTIC REGRESSION	0.909143
1	NAIVE BAYES	0.894857
2	K NEAREST NEIGHBOR	1.000000

CONCLUSION :

Out of above three classifier" K NEAREST NEIGHBOR "shows highest score .It means k nearest neighbors train our model with highest accuracy.

The advantage of using k nearest neighbors is:

1.Time complexity is O(n).
2.Can make predictions without training.
3.can be used for both classification and regression.

In [ ]:
```

```