Supervised Learning - project 2 - Edouard Toutounji

November 29, 2025

[146]:
```python
# importing necessary libraries for EDA
import numpy as np
import pandas as pd

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

[176]:
```python
# loading the dataset for Bank Personal Loan (BPL_data)
BPL_data = pd.read_csv('Bank_Personal_Loan_Modelling.csv')
```

0.1 1 - Columns description

[148]:
```python
BPL_data.head()
```

[148]:
```
   ID  Age  Experience  Income  ZIP Code  Family  CCAvg  Education  Mortgage  \
0   1   25           1      49     91107       4    1.6          1         0
1   2   45          19      34     90089       3    1.5          1         0
2   3   39          15      11     94720       1    1.0          1         0
3   4   35           9     100     94112       1    2.7          2         0
4   5   35           8      45     91330       4    1.0          2         0

   Personal Loan  Securities Account  CD Account  Online  CreditCard
0              0                   1           0       0           0
1              0                   1           0       0           0
2              0                   0           0       0           0
3              0                   0           0       0           0
4              0                   0           0       0           1
```

[149]:
```python
BPL_data.shape
```

[149]:
```
(5000, 14)
```

[150]:
```python
BPL_data.columns
```

[150]:
```
Index(['ID', 'Age', 'Experience', 'Income', 'ZIP Code', 'Family', 'CCAvg',
       'Education', 'Mortgage', 'Personal Loan', 'Securities Account',
       'CD Account', 'Online', 'CreditCard'],
```

```
      dtype='object')
```

[151]:
```
# all values seem non-null

BPL_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 14 columns):
ID                    5000 non-null int64
Age                   5000 non-null int64
Experience            5000 non-null int64
Income                5000 non-null int64
ZIP Code              5000 non-null int64
Family                5000 non-null int64
CCAvg                 5000 non-null float64
Education             5000 non-null int64
Mortgage              5000 non-null int64
Personal Loan         5000 non-null int64
Securities Account    5000 non-null int64
CD Account            5000 non-null int64
Online                5000 non-null int64
CreditCard            5000 non-null int64
dtypes: float64(1), int64(13)
memory usage: 547.0 KB
```

[152]:
```
# all values seem non-null

BPL_data.isnull().sum()
```

[152]:
```
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
```

[153]:
```
BPL_data.describe()
```

[153]:

	ID	Age	Experience	Income	ZIP Code
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000
mean	2500.500000	45.338400	20.104600	73.774200	93152.503000
std	1443.520003	11.463166	11.467954	46.033729	2121.852197
min	1.000000	23.000000	-3.000000	8.000000	9307.000000
25%	1250.750000	35.000000	10.000000	39.000000	91911.000000
50%	2500.500000	45.000000	20.000000	64.000000	93437.000000
75%	3750.250000	55.000000	30.000000	98.000000	94608.000000
max	5000.000000	67.000000	43.000000	224.000000	96651.000000

	Family	CCAvg	Education	Mortgage	Personal Loan
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000
mean	2.396400	1.937938	1.881000	56.498800	0.096000
std	1.147663	1.747659	0.839869	101.713802	0.294621
min	1.000000	0.000000	1.000000	0.000000	0.000000
25%	1.000000	0.700000	1.000000	0.000000	0.000000
50%	2.000000	1.500000	2.000000	0.000000	0.000000
75%	3.000000	2.500000	3.000000	101.000000	0.000000
max	4.000000	10.000000	3.000000	635.000000	1.000000

	Securities Account	CD Account	Online	CreditCard
count	5000.000000	5000.00000	5000.000000	5000.000000
mean	0.104400	0.06040	0.596800	0.294000
std	0.305809	0.23825	0.490589	0.455637
min	0.000000	0.00000	0.000000	0.000000
25%	0.000000	0.00000	0.000000	0.000000
50%	0.000000	0.00000	1.000000	0.000000
75%	0.000000	0.00000	1.000000	1.000000
max	1.000000	1.00000	1.000000	1.000000

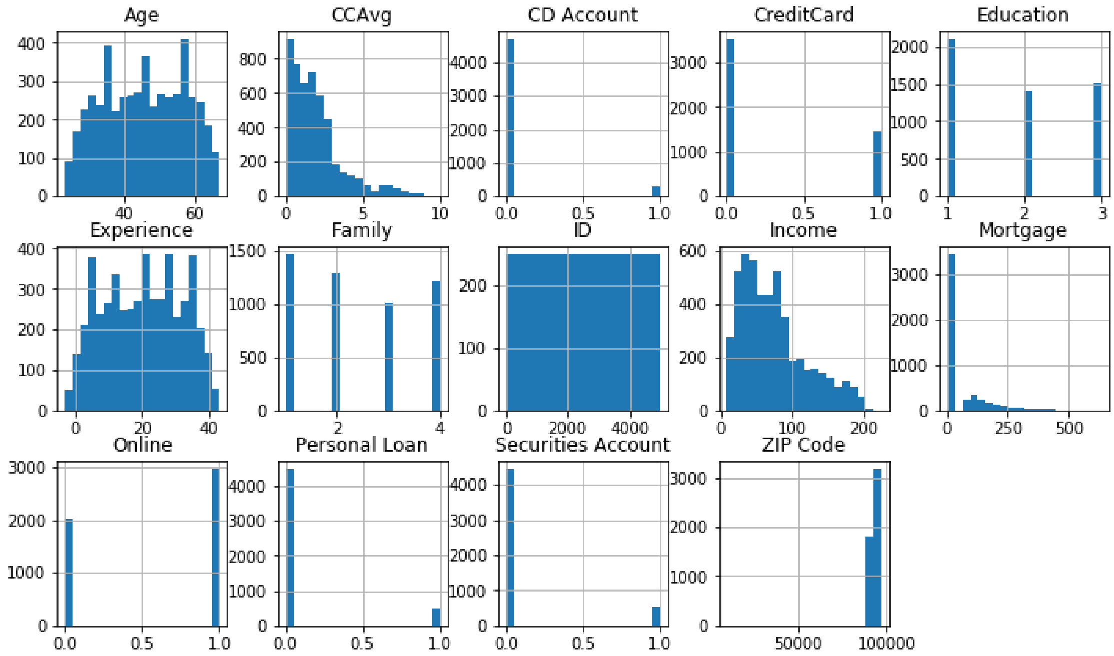
0.1.1 Observations and comments for later analysis:

- No Null values
- "ID" is unecesseary : to be droppped
- "ZIP code" could be irrelevant information : top be dropped
- "Education" : to be transformed into multiple columns via dummy varaiables/one-hot encoding
- The last 5 columns: "Personal Loan", "Securities Account","CD Account", "Online", "CreditCard": logical values(True 1/False 0)

0.2 2 - Univariate distributions, correlations , pairplots and graphical represenations

[154]:
```python
# The distributions of the 2 columns : "ZIP code" and "ID" confirm our initial␣
↪doubt of their irrelevance.

BPL_data.hist(stacked=False, bins=20, figsize=(12,12), layout=(5,5))
```

[154]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1a66e01d50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a657bbed0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a673f0dd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a566228d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a68d64c50>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1a68d78f50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a6937dd90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a693a6590>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a693b1050>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a68e46490>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1a69443e90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a69485c50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a694bdf50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a694f67d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a6952dc90>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1a6956e4d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a695a2cd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a695e2510>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a69619d10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a69658550>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1a6968dd50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a696cf590>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a69702d90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a697435d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a69778dd0>]],
      dtype=object)
```



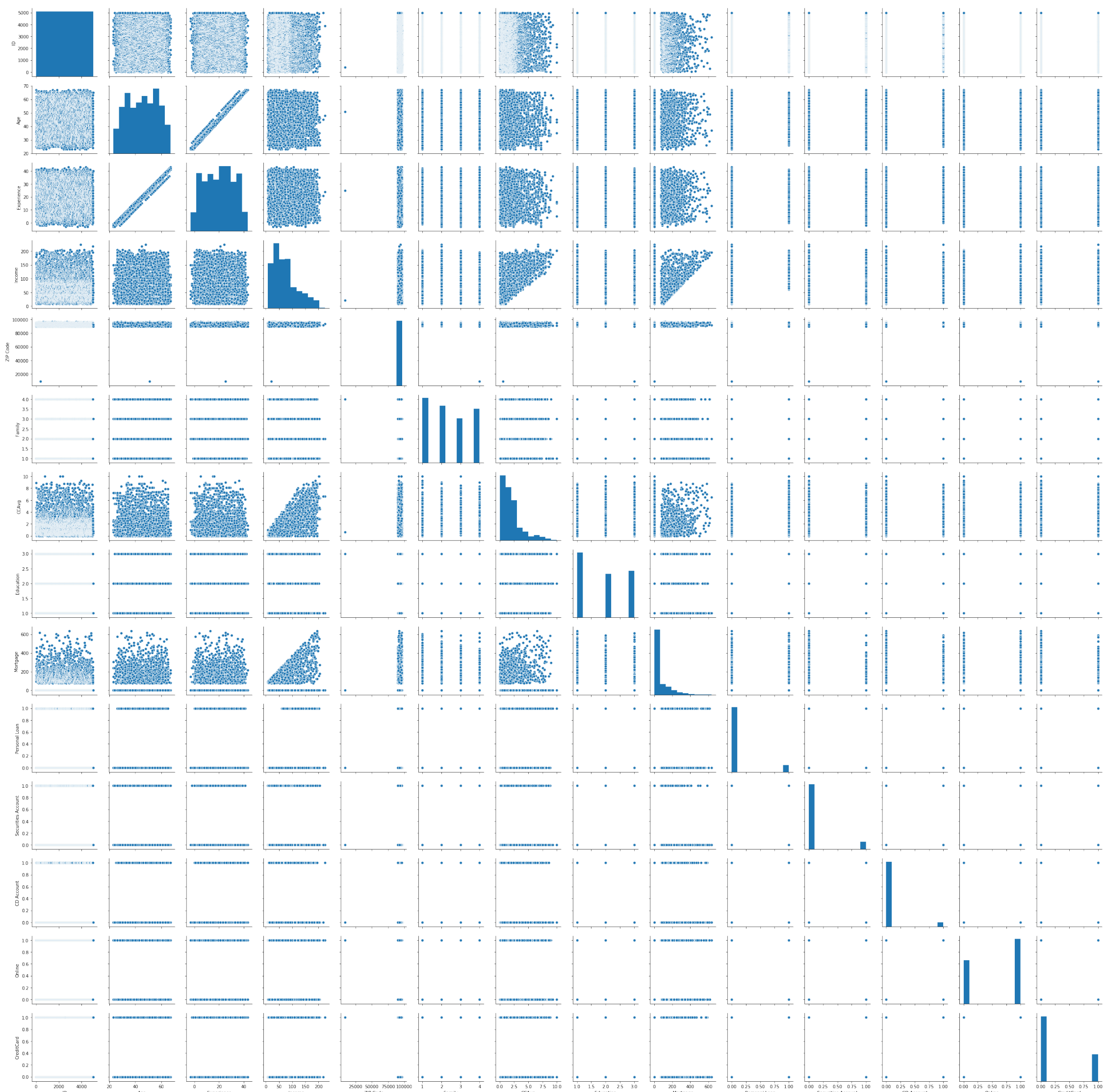
[155]:
```
BPL_corr = BPL_data.corr()
```

[156]:
```
# correlation visually

BPL_corr.style.background_gradient(cmap='PuBu')
```

[156]: <pandas.io.formats.style.Styler at 0x1a69aa5b90>

[138]:
```
# now mixing both types of graph in one-shot

sns.pairplot(BPL_data)
```

[138]: <seaborn.axisgrid.PairGrid at 0x1a605de350>



0.2.1 Conclusions:

- Strong linear correlation between : "Age"/"Income".
- Observable linear correlations were mainly between : "Personal Loan" and "Income"/"Mortgage"/"CCAvg"/"CD Account".
- The BPL_data will be reduced("Zip Code" and "ID") and the "Personal Loan" column shall be created as a separate taget column

0.3 3 - Reshaping BPL_data

- Droppping columns: 'ZIP Code', 'ID'
- Making Dummy variables from 'Education' after replacing numerical values
- Rechecking correlations

- Checking the Target column 'Personal Loan' and analysing its distribution

[177]:
```
# Reducing the BPL _data Dataframe

BPL_data_dropped= BPL_data.drop(['ZIP Code', 'ID'], axis=1)
```

[180]:
```
BPL_data_dropped.shape
```

[180]:
```
(5000, 12)
```

[183]:
```
# splitting education into 3 different columns after renaming the numeric␣
↪values

BPL_data_dropped['Education'] = BPL_data_dropped['Education'].replace({1:
↪'Undergrad', 2:'Graduate', 3:'Adv/Prof'})
BPL_data_dropped.head()
```

[183]:
```
   Age  Experience  Income  Family  CCAvg  Education  Mortgage  Personal Loan  \
0   25           1      49       4    1.6  Undergrad         0              0
1   45          19      34       3    1.5  Undergrad         0              0
2   39          15      11       1    1.0  Undergrad         0              0
3   35           9     100       1    2.7   Graduate         0              0
4   35           8      45       4    1.0   Graduate         0              0

   Securities Account  CD Account  Online  CreditCard
0                   1           0       0           0
1                   1           0       0           0
2                   0           0       0           0
3                   0           0       0           0
4                   0           0       0           1
```

[184]:
```
# creating Dummy Variables for 'Education' column
BPL_data_dropped_dummies = pd.get_dummies( BPL_data_dropped,␣
↪columns=['Education'])
BPL_data_dropped_dummies.head()
```

[184]:
```
   Age  Experience  Income  Family  CCAvg  Mortgage  Personal Loan  \
0   25           1      49       4    1.6         0              0
1   45          19      34       3    1.5         0              0
2   39          15      11       1    1.0         0              0
3   35           9     100       1    2.7         0              0
4   35           8      45       4    1.0         0              0

   Securities Account  CD Account  Online  CreditCard  Education_Adv/Prof  \
0                   1           0       0           0                   0
1                   1           0       0           0                   0
2                   0           0       0           0                   0
3                   0           0       0           0                   0
```

```
4                   0            0       0            1                   0

   Education_Graduate  Education_Undergrad
0                   0                    1
1                   0                    1
2                   0                    1
3                   1                    0
4                   1                    0
```

[185]:
```python
# rechecking the correlation matrix after dropping the 2 columns above.
#Nothing alarming changed so our hunch was correct

BPL_data_dropped_dummies.corr().style.background_gradient(cmap='PuBu')
```

[185]: <pandas.io.formats.style.Styler at 0x1a69c93610>

0.3.1 Target Column Distribution : majority of bank customers did NOT buy Personal loans

[186]:
```python
# Count analysis
BPL_data_dropped_dummies['Personal Loan'].value_counts()
```

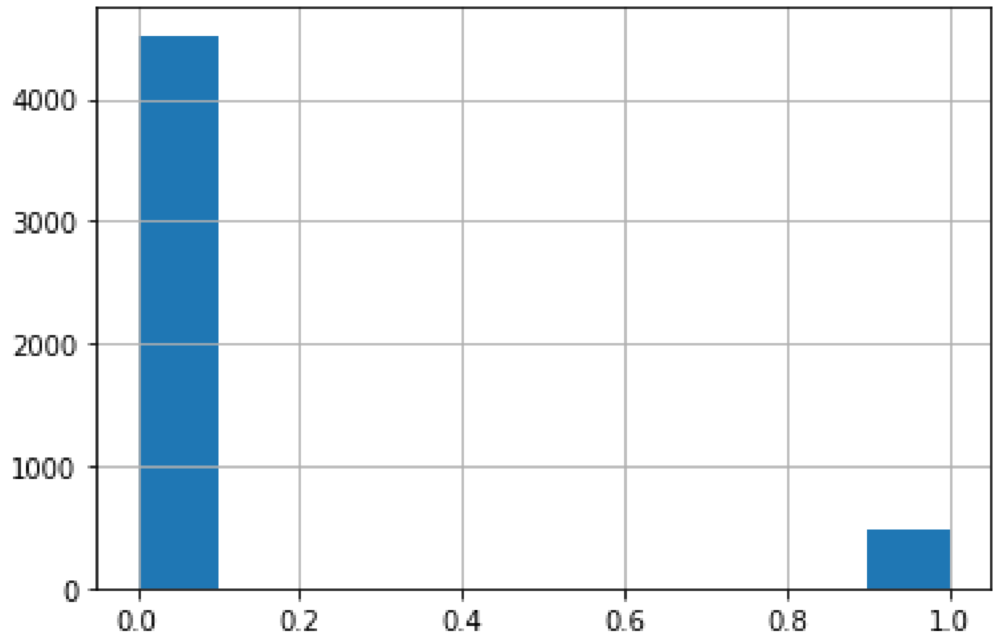
[186]:
```
0    4520
1     480
Name: Personal Loan, dtype: int64
```

[187]:
```python
# Percentage analysis
BPL_data_dropped_dummies['Personal Loan'].value_counts(normalize=True)
```

[187]:
```
0    0.904
1    0.096
Name: Personal Loan, dtype: float64
```

[188]:
```python
# Graphical Distribution
BPL_data_dropped_dummies['Personal Loan'].hist()
```

[188]: <matplotlib.axes._subplots.AxesSubplot at 0x1a69cabb90>



0.4 4 - Defining X and y – then splitting data into: Train(70%) / Test(30%) ratio

[189]:
```
# rechecking columns in BPL_data
BPL_data_dropped_dummies.columns
```

[189]:
```
Index(['Age', 'Experience', 'Income', 'Family', 'CCAvg', 'Mortgage',
       'Personal Loan', 'Securities Account', 'CD Account', 'Online',
       'CreditCard', 'Education_Adv/Prof', 'Education_Graduate',
       'Education_Undergrad'],
      dtype='object')
```

[190]:
```
X = BPL_data_dropped_dummies.drop('Personal Loan', axis=1)
# rechecking columns in X
X.columns
```

[190]:
```
Index(['Age', 'Experience', 'Income', 'Family', 'CCAvg', 'Mortgage',
       'Securities Account', 'CD Account', 'Online', 'CreditCard',
       'Education_Adv/Prof', 'Education_Graduate', 'Education_Undergrad'],
      dtype='object')
```

[191]:
```
y = BPL_data_dropped_dummies['Personal Loan']
y.describe()
```

[191]:
```
count    5000.000000
mean        0.096000
std         0.294621
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000
Name: Personal Loan, dtype: float64
```

0.4.1 Splitting the Data

- X_train , y_train : will be used ONCE (in an interative manner I assume) to fit the optimal model (Gradient Descent ?)
- X_test , y_test : will be subsequently used for prediction and performance measure

[ ]:
```
# importing the necesseary library
from sklearn.model_selection import train_test_split
```

[195]:
```
(X_train, X_test , y_train , y_test )= train_test_split (X , y ,test_size=0.30,␣
 ↪random_state=1 )
```

[205]:
```
# checking ratios
print('Ratios of  X_train and X_test resp. : '+ str(len (X_train)/len(X)) + '␣
 ↪and '+ str(len (X_test)/len(X)))
print('Ratios of  y_train and y_test resp. : '+ str(len (y_train)/len(y)) + '␣
 ↪and '+ str(len (y_test)/len(y)))
```

```
Ratios of  X_train and X_test resp. : 0.7 and 0.3
Ratios of  y_train and y_test resp. : 0.7 and 0.3
```

0.5 5 - Logistic Regression

[206]:
```
# importing the necesseary libraries

from sklearn.linear_model import LogisticRegression
from sklearn import metrics
```

[255]:
```
# fitting the model and measuring its score

model = LogisticRegression(solver="liblinear")   # this parameter isn't clear␣
 ↪but I read it's for small Datasets

model.fit (X_train , y_train)
print('Training Data Score: '+ str(model.score(X_train, y_train)))
print('Testing  Data Score: '+ str(model.score(X_test, y_test)))
```

```
Training Data Score: 0.9585714285714285
Testing  Data Score: 0.9573333333333334
```

[248]:
```python
# Saving coefficients and intercept:

Coefficients = pd.DataFrame(model.coef_, columns = [X_train.columns])
Intercept    = pd.DataFrame(model.intercept_, columns=['Intercept'])

Model_Parameters= pd.concat([Intercept, Coefficients], axis=1)
print(Model_Parameters.T)
```

```
                              0
Intercept             -1.469267
(Age,)                -0.311426
(Experience,)          0.311550
(Income,)              0.054166
(Family,)              0.584145
(CCAvg,)               0.194660
(Mortgage,)            0.000933
(Securities Account,) -0.819830
(CD Account,)          3.140831
(Online,)             -0.581268
(CreditCard,)         -0.868540
(Education_Adv/Prof,)  0.862584
(Education_Graduate,)  0.639760
(Education_Undergrad,) -2.971611
```

0.5.1 Predicting the likelihood of buying Personal Loan on the X_test data

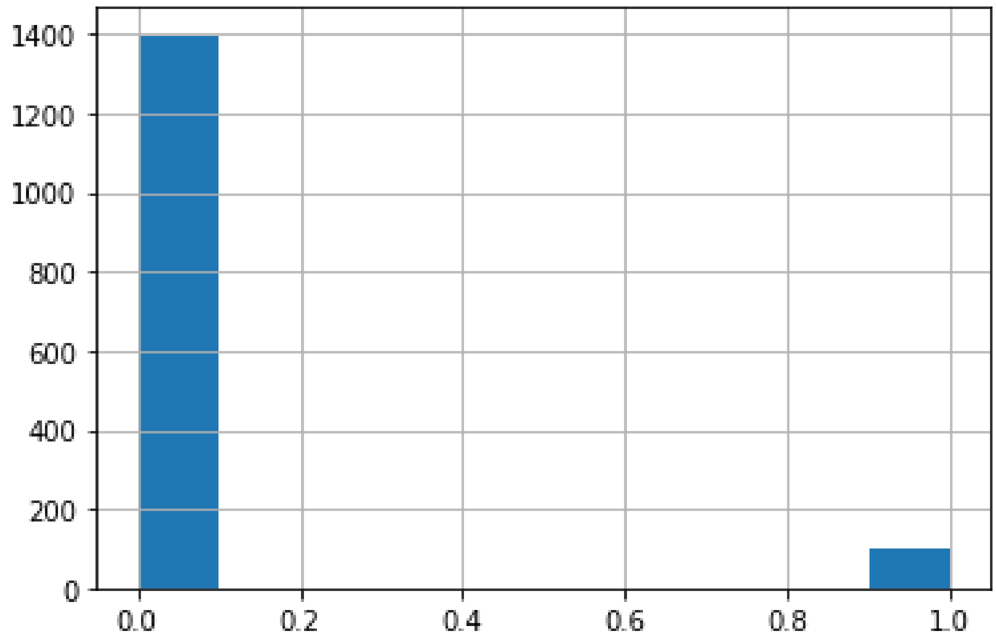
[307]:
```python
y_pred_test = model.predict(X_test)
```

[308]:
```python
print(str( pd.Series(y_pred_test).value_counts() ))

print('\n' +
      str( pd.Series(y_pred_test).value_counts(normalize=True) ))
```

```
0    1397
1     103
dtype: int64

0    0.931333
1    0.068667
dtype: float64
```

[309]:
```python
pd.Series(y_pred_test).hist()
```

[309]: <matplotlib.axes._subplots.AxesSubplot at 0x1a6b1204d0>



0.5.2 Conclusion :

The predicted Distribution in PL buys matches the Original Data distbution in y

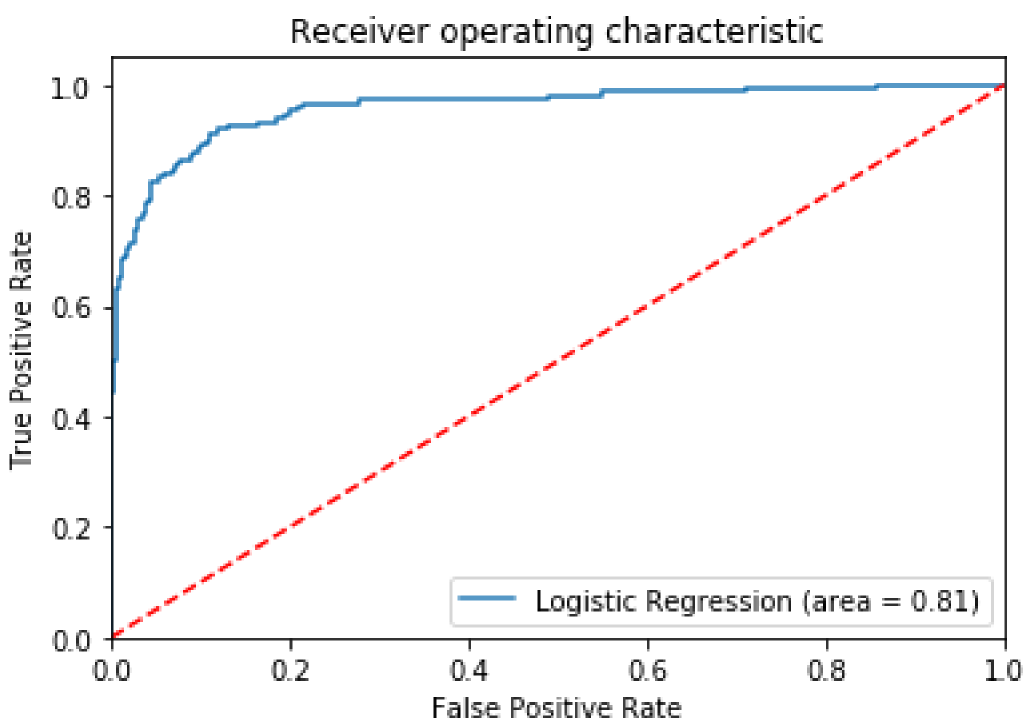
0.6 6 - Confusion matrices for the model

[310]:
```
# confusion matrix library
from sklearn.metrics import confusion_matrix

# computing the matrices on BOTH training and testing Data , expected to be␣
↪similar in terms of ratios
y_pred_train = model.predict(X_train)

mat_train = confusion_matrix(y_train, y_pred_train)
mat_test  = confusion_matrix(y_test , y_pred_test)

print("confusion matrix for training data = \n",mat_train)
print("\n confusion matrix for testing data = \n",mat_test)
```

```
confusion matrix for training data =
 [[3134   35]
 [ 110  221]]

 confusion matrix for testing data =
 [[1342    9]
 [  55   94]]
```

0.7 7 - Model Performance in predicting PL purchases and final comments

[311]:
```
# THIS IS HONESTLY COPY PASTED AS IS FROM CASE STUDY, slighlty modified for our␣
↪model

#AUC ROC curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve

logit_roc_auc = roc_auc_score(y_test, model.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, model.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```



0.7.1 Final Thoughts

This Logistic Regression model whether in the training or testing portions seems to correclty predict the original data distribution of the 'Personal Loan' column to a high level. It was seen in : 1- the Model Scores after fitting. 2- The Distribution in y__pred__test: the predicted on X__test. 3- The confusion matrices on both : the predicted on X__test and X__train 4- Finally in the AUC-ROC plot This was mainly achieved by noticing the unnecessary column for 'Zip Code' and 'ID' , removing them as they may add noise; in addition to modiying 'Education' into Dummy variables so that they can add logical meaning to the predictitive model of the likelihood of PL purchase.

Thanks , Edouard Toutounji - jan 31, 2020.

```
[ ]:
```