project2 supervised learning

March 16, 2021

ID: Customer ID

Age: Customer's age in completed years

Experience: #years of professional experience

Income: Annual income of the customer ($000)

ZIP Code: Home Address ZIP

Family: Family size of the customer

CCAvg: Avg. spending on credit cards per month ($000)

Education: Education Level. 1: Undergrad; 2: Graduate; 3: Advanced/Professional

Mortgage: Value of house mortgage if any. ($000)

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 the bank?

1 1) Import the datasets and libraries, check datatype, statistical summary, shape, null values or incorrect imputation. (5 marks)

[184]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from scipy import stats
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import normalize
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import recall_score, roc_auc_score, classification_report,␣
 ↪confusion_matrix, accuracy_score, plot_roc_curve
sns.set()
```

[185]:
```
df = pd.read_csv('Bank_Personal_Loan_Modelling.csv', dtype={'ZIP Code': 'str'})

#replacing the categorical var with actual values
df['Education'] = df['Education'].replace({1: 'Undergrad', 2: 'Graduate', 3:␣
↪'Advanced/Professional'})
df.head(10)
```

[185]:
```
   ID  Age  Experience  Income ZIP Code  Family  CCAvg              Education  \
0   1   25           1      49    91107       4    1.6              Undergrad
1   2   45          19      34    90089       3    1.5              Undergrad
2   3   39          15      11    94720       1    1.0              Undergrad
3   4   35           9     100    94112       1    2.7               Graduate
4   5   35           8      45    91330       4    1.0               Graduate
5   6   37          13      29    92121       4    0.4               Graduate
6   7   53          27      72    91711       2    1.5               Graduate
7   8   50          24      22    93943       1    0.3  Advanced/Professional
8   9   35          10      81    90089       3    0.6               Graduate
9  10   34           9     180    93023       1    8.9  Advanced/Professional

   Mortgage  Personal Loan  Securities Account  CD Account  Online  CreditCard
0         0              0                   1           0       0           0
1         0              0                   1           0       0           0
2         0              0                   0           0       0           0
3         0              0                   0           0       0           0
4         0              0                   0           0       0           1
5       155              0                   0           0       1           0
6         0              0                   0           0       1           0
7         0              0                   0           0       0           1
8       104              0                   0           0       1           0
9         0              1                   0           0       0           0
```

2 2) EDA: Study the data distribution in each attribute and target variable, share your findings (20 marks)

2.0.1 Number of unique in each column?

[186]:
```
for column in df.columns:
    print(column, "has %d unique values"%df[column].nunique())
```

```
ID has 5000 unique values
Age has 45 unique values
Experience has 47 unique values
Income has 162 unique values
ZIP Code has 467 unique values
Family has 4 unique values
CCAvg has 108 unique values
Education has 3 unique values
```

```
Mortgage has 347 unique values
Personal Loan has 2 unique values
Securities Account has 2 unique values
CD Account has 2 unique values
Online has 2 unique values
CreditCard has 2 unique values
```

2.0.2 Number of people with zero mortgage?

[187]:
```
temp = df[df['Mortgage'] == 0].count()
print("there are %d people with $0 mortgage"%temp[0])
```

```
there are 3462 people with $0 mortgage
```

2.0.3 Number of people with zero credit card spending per month?

[188]:
```
temp = df[df['CCAvg']==0].count()
print('%d people have $0 Credit Card spending montly'%temp[0])
```

```
106 people have $0 Credit Card spending montly
```

2.0.4 Value counts of all categorical columns

[189]:
```
df.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   object
 5   Family              5000 non-null   int64
 6   CCAvg               5000 non-null   float64
 7   Education           5000 non-null   object
 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(11), object(2)
memory usage: 547.0+ KB
```

[190]:
```
for column in df.select_dtypes(include="object"):
    print(df[column].value_counts())
```

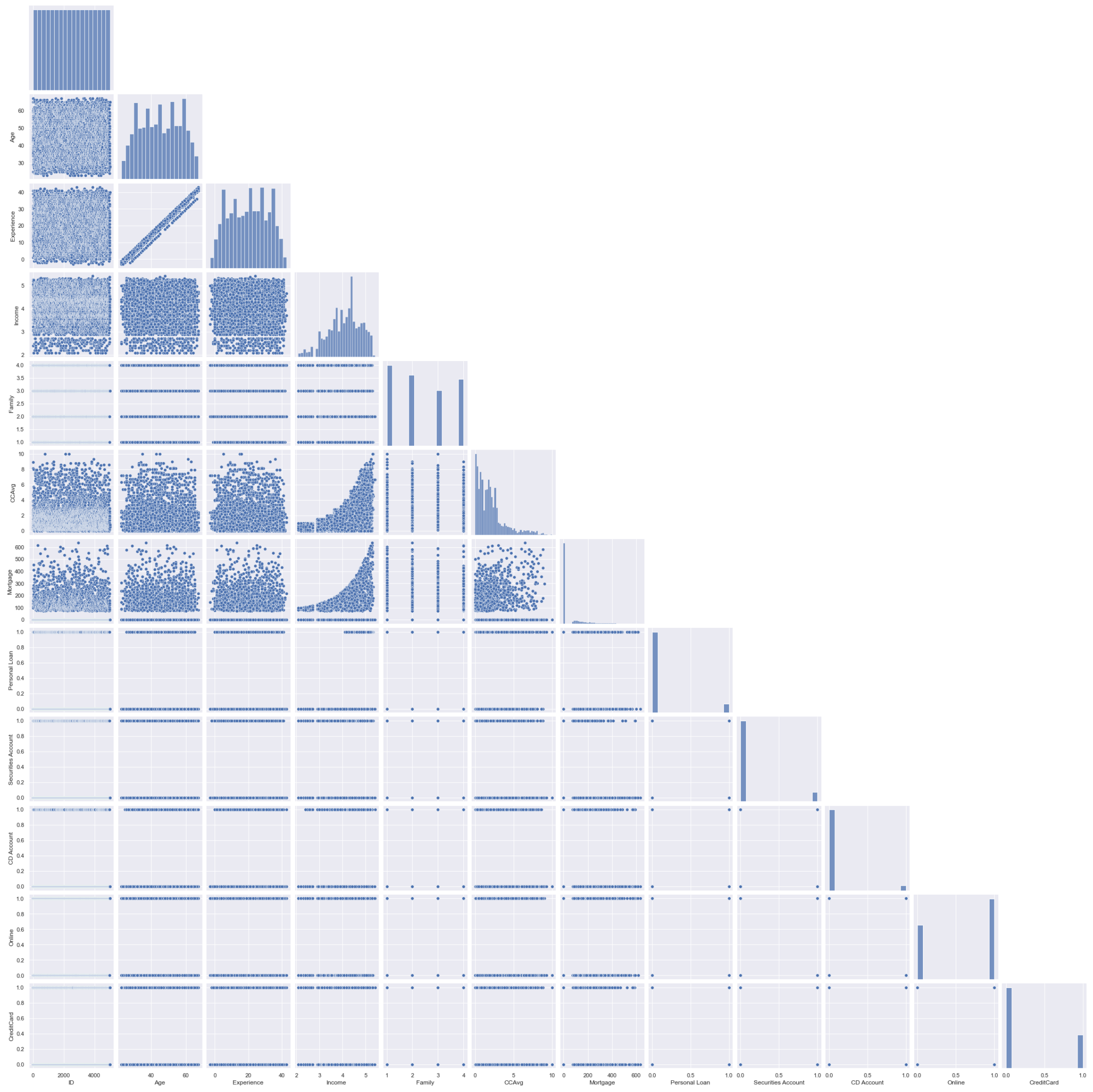
```
94720    169
94305    127
95616    116
90095     71
93106     57
        …
94970      1
90813      1
94598      1
90068      1
9307       1
Name: ZIP Code, Length: 467, dtype: int64
Undergrad                2096
Advanced/Professional    1501
Graduate                 1403
Name: Education, dtype: int64
```

2.0.5 plot Univariate and Bivariate data

[191]:
```
df['Income'] = np.log(df['Income'])
```

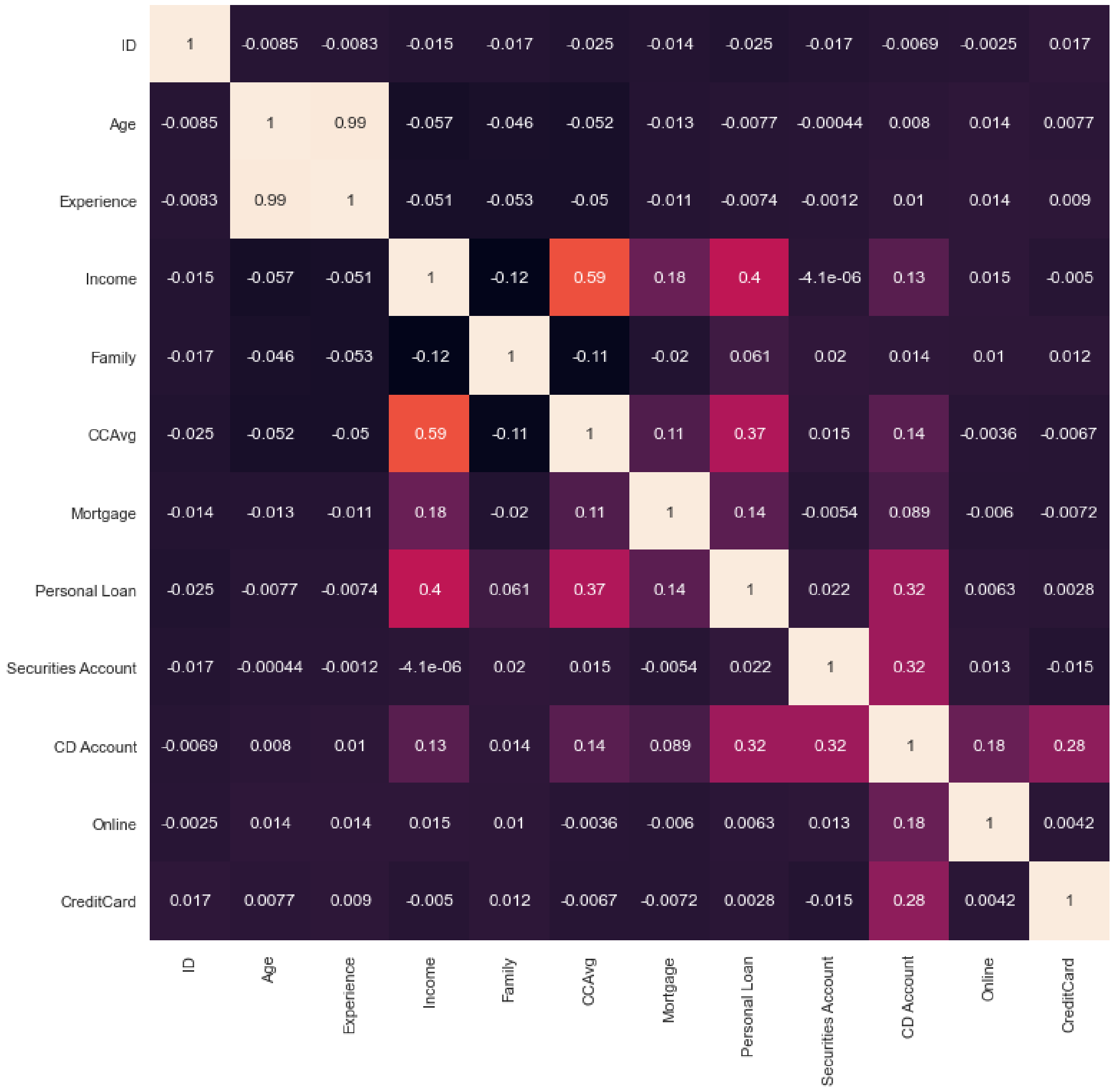
[192]:
```
plt.figure(figsize=(20,20))
sns.pairplot(data=df, corner=True)
plt.show()
```

```
<Figure size 1440x1440 with 0 Axes>
```



```
[193]: plt.figure(figsize=(12,12))
       sns.heatmap(df.corr(), annot = True, cbar=False)
```

```
[193]: <AxesSubplot:>
```



[194]:
```
df.describe()
```

[194]:
```
                ID          Age   Experience       Income       Family  \
count  5000.000000  5000.000000  5000.000000  5000.000000  5000.000000
mean   2500.500000    45.338400    20.104600     4.085451     2.396400
std    1443.520003    11.463166    11.467954     0.696455     1.147663
min       1.000000    23.000000    -3.000000     2.079442     1.000000
25%    1250.750000    35.000000    10.000000     3.663562     1.000000
50%    2500.500000    45.000000    20.000000     4.158883     2.000000
75%    3750.250000    55.000000    30.000000     4.584967     3.000000
max    5000.000000    67.000000    43.000000     5.411646     4.000000

             CCAvg     Mortgage  Personal Loan  Securities Account  \
count  5000.000000  5000.000000     5000.000000         5000.000000
mean      1.937938    56.498800        0.096000            0.104400
std       1.747659   101.713802        0.294621            0.305809
min       0.000000     0.000000        0.000000            0.000000
25%       0.700000     0.000000        0.000000            0.000000
50%       1.500000     0.000000        0.000000            0.000000
75%       2.500000   101.000000        0.000000            0.000000
max      10.000000   635.000000        1.000000            1.000000

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

[195]:
```
#replacing negative experience values with median
#df[df['Experience']<0]['Experience'] = df['Experience'].median()
df.loc[df['Experience'] < 0, ['Experience']] = df['Experience'].median()

df.describe()
```

[195]:
```
                ID          Age   Experience       Income       Family  \
count  5000.000000  5000.000000  5000.000000  5000.000000  5000.000000
mean   2500.500000    45.338400    20.327600     4.085451     2.396400
std    1443.520003    11.463166    11.253035     0.696455     1.147663
min       1.000000    23.000000     0.000000     2.079442     1.000000
25%    1250.750000    35.000000    11.000000     3.663562     1.000000
50%    2500.500000    45.000000    20.000000     4.158883     2.000000
75%    3750.250000    55.000000    30.000000     4.584967     3.000000
max    5000.000000    67.000000    43.000000     5.411646     4.000000

             CCAvg     Mortgage  Personal Loan  Securities Account  \
count  5000.000000  5000.000000    5000.000000         5000.000000
mean      1.937938    56.498800       0.096000            0.104400
std       1.747659   101.713802       0.294621            0.305809
min       0.000000     0.000000       0.000000            0.000000
25%       0.700000     0.000000       0.000000            0.000000
50%       1.500000     0.000000       0.000000            0.000000
75%       2.500000   101.000000       0.000000            0.000000
max      10.000000   635.000000       1.000000            1.000000

       CD Account       Online   CreditCard
count  5000.00000  5000.000000  5000.000000
```

```
mean      0.06040     0.596800     0.294000
std       0.23825     0.490589     0.455637
min       0.00000     0.000000     0.000000
25%       0.00000     0.000000     0.000000
50%       0.00000     1.000000     0.000000
75%       0.00000     1.000000     1.000000
max       1.00000     1.000000     1.000000
```

3 3) Split the data into training and test set in the ratio of 70:30 respectively (5 marks)

[196]:
```
#one hot encoding of categorical vars
#zipcode = pd.get_dummies(df['ZIP Code'], drop_first=True)
education = pd.get_dummies(df['Education'], drop_first=True)
df = df.join(education)
df.head()
```

[196]:
```
   ID  Age  Experience    Income  ZIP Code  Family  CCAvg  Education  Mortgage  \
0   1   25           1  3.891820     91107       4    1.6  Undergrad         0
1   2   45          19  3.526361     90089       3    1.5  Undergrad         0
2   3   39          15  2.397895     94720       1    1.0  Undergrad         0
3   4   35           9  4.605170     94112       1    2.7   Graduate         0
4   5   35           8  3.806662     91330       4    1.0   Graduate         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

   Graduate  Undergrad
0         0          1
1         0          1
2         0          1
3         1          0
4         1          0
```

[ ]:
```

```

[197]:
```
#split vars
x = df.drop('Personal Loan', axis=1)
y = df['Personal Loan']
x.drop(['ZIP Code','Education', 'ID','Experience'], axis=1, inplace=True)
x.head()
```

[197]:
```
   Age    Income  Family  CCAvg  Mortgage  Securities Account  CD Account  \
0   25  3.891820       4    1.6         0                   1           0
1   45  3.526361       3    1.5         0                   1           0
2   39  2.397895       1    1.0         0                   0           0
3   35  4.605170       1    2.7         0                   0           0
4   35  3.806662       4    1.0         0                   0           0

   Online  CreditCard  Graduate  Undergrad
0       0           0         0          1
1       0           0         0          1
2       0           0         0          1
3       0           0         1          0
4       0           1         1          0
```

[198]:
```
#x['Age'] = (x['Age'] - x['Age'].min())/(x['Age'].max() - x['Age'].min())
#x['Experience'] = (x['Experience'] - x['Experience'].min())/(x['Experience'].
 ↪max() - x['Experience'].min())
#x['Family'] = (x['Family'] - x['Family'].min())/(x['Family'].max() -␣
 ↪x['Family'].min())
#x['Income'] = (x['Income'] - x['Income'].min())/(x['Income'].max() -␣
 ↪x['Income'].min())
#x['CCAvg'] = (x['CCAvg'] - x['CCAvg'].min())/(x['CCAvg'].max() - x['CCAvg'].
 ↪min())

#x.head()
```

[199]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,random_state=0,␣
 ↪test_size=.3)
```

[200]:
```
#verify split sizes
print("Percentage training data", len(x_train)/(len(x_train)+len(x_test)))
print("Percentage training results", len(y_train)/(len(y_train)+len(y_test)))
print('\n\n')
print("Percentage testing data", len(x_test)/(len(x_train)+len(x_test)))
print("Percentage testing results", len(y_test)/(len(y_train)+len(y_test)))
```

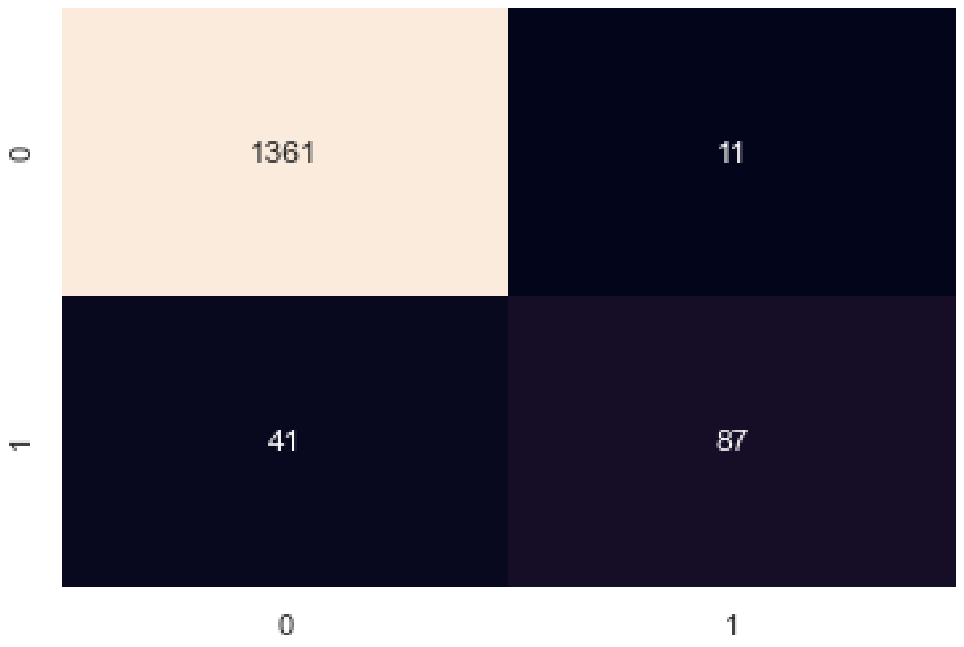
```
Percentage training data 0.7
Percentage training results 0.7



Percentage testing data 0.3
Percentage testing results 0.3
```

4 4) Use the Logistic Regression model to predict whether the customer will take a personal loan or not. Print all the metrics related to evaluating the model performance (accuracy, recall, precision, f1score, and roc_auc_score). Draw a heatmap to display confusion matrix (15 marks)

[201]:
```
model = LogisticRegression(solver='newton-cg', max_iter=100)
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
```

[202]:
```
matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(matrix, cbar=False, annot=True, fmt='.4g')
```

[202]:
```
<AxesSubplot:>
```



[203]:
```
print(classification_report(y_test, y_pred))
print('Accuracy of the model is', accuracy_score(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      1372
           1       0.89      0.68      0.77       128

    accuracy                           0.97      1500
   macro avg       0.93      0.84      0.88      1500
```

```
weighted avg       0.96      0.97      0.96      1500

Accuracy of the model is 0.9653333333333334
```

[204]:
```
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
specificity = tn / (tn+fp)
print('specificity of the model is', specificity)
```

```
specificity of the model is 0.9919825072886297
```

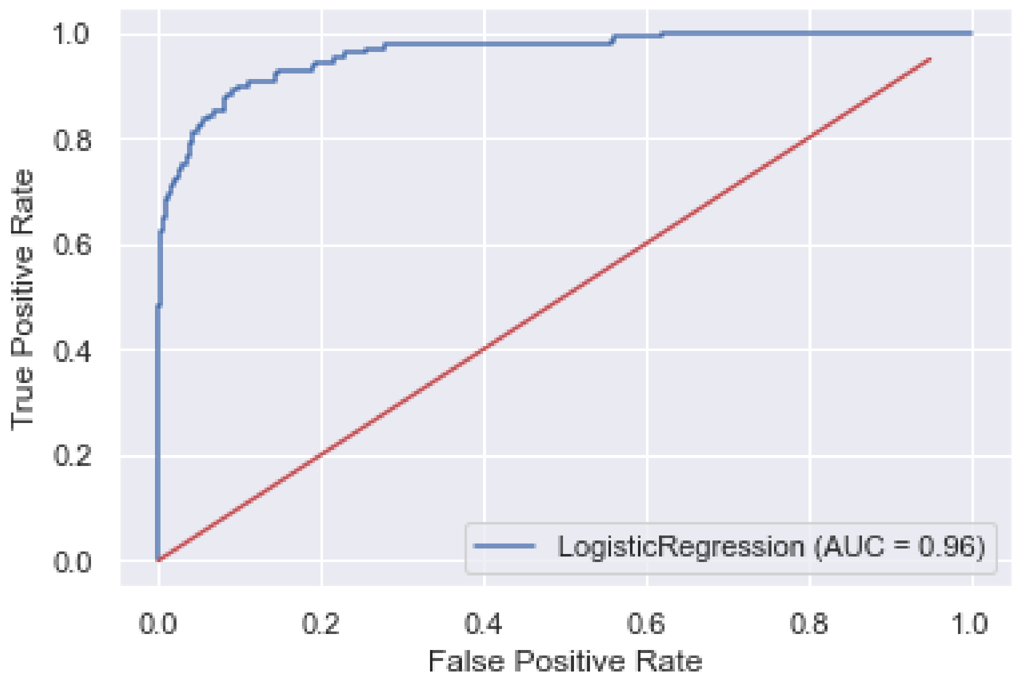
[205]:
```
roc_auc_score(y_test, y_pred)
```

[205]:
```
0.8358350036443147
```

[206]:
```
from sklearn.model_selection import cross_val_score
model1 = LogisticRegression(solver='newton-cg')
scoring_metrics=['precision','recall', 'accuracy','f1','roc_auc']
for i in range(len(scoring_metrics)):
    score=cross_val_score(model1, x, y, cv=10, scoring=scoring_metrics[i])
    print('The mean %s score is %-.2f'%(scoring_metrics[i],score.mean()))
```

```
The mean precision score is 0.90
The mean recall score is 0.69
The mean accuracy score is 0.96
The mean f1 score is 0.78
The mean roc_auc score is 0.97
```

[207]:
```
x = np.arange(0,1,.05)
plot_roc_curve(model, x_test, y_test)
plt.plot(x,x, 'r-')
plt.show()
```



[208]:
```
import statsmodels.api as sm

logit = sm.Logit( y_train, sm.add_constant( x_train ) )

lg = logit.fit()

lg.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.106185
         Iterations 10
```

[208]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:          Personal Loan   No. Observations:                 3500
Model:                          Logit   Df Residuals:                     3488
Method:                           MLE   Df Model:                           11
Date:                Sun, 17 Jan 2021   Pseudo R-squ.:                  0.6746
Time:                        12:46:01   Log-Likelihood:                -371.65
converged:                       True   LL-Null:                       -1142.2
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================
======
                         coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
------
const                -37.6892      2.376    -15.865      0.000     -42.345
-33.033
Age                    0.0048      0.008      0.577      0.564      -0.012
0.021
Income                 7.5530      0.485     15.584      0.000       6.603
8.503
Family                 0.5768      0.094      6.114      0.000       0.392
0.762
CCAvg                  0.2078      0.054      3.853      0.000       0.102
0.313
Mortgage               0.0010      0.001      1.416      0.157      -0.000
0.002
Securities Account    -0.7063      0.360     -1.961      0.050      -1.412
-0.000
CD Account             3.8856      0.428      9.089      0.000       3.048
4.724
Online                -0.9057      0.206     -4.402      0.000      -1.309
-0.502
CreditCard            -1.0245      0.259     -3.948      0.000      -1.533
-0.516
Graduate               0.0330      0.237      0.139      0.889      -0.432
0.498
Undergrad             -4.2565      0.318    -13.383      0.000      -4.880
-3.633
================================================================================
======

Possibly complete quasi-separation: A fraction 0.33 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
"""
```

5 5) Find out coefficients of all the attributes and show the output in a data frame with column names (10 marks)

[209]:
```
df_coef = pd.DataFrame()
coef = model.coef_
columns = x_train.columns
df_coef['Coefs'] = coef[0,:]
df_coef['Feature'] = (columns)
df_coef = df_coef.append({'Coefs': model.intercept_, 'Feature': "Intercept"},␣
 ↪ignore_index = True)
```

```
df_coef
```

```
[209]:                   Coefs             Feature
       0             0.00362349                 Age
       1                6.11849              Income
       2                0.49818              Family
       3               0.216408               CCAvg
       4             0.00112513            Mortgage
       5              -0.478079  Securities Account
       6                3.10203          CD Account
       7              -0.719543              Online
       8              -0.779656          CreditCard
       9              0.0970267            Graduate
       10              -3.51098           Undergrad
       11   [-30.99324688449752]           Intercept
```

5.1 For test data show all the rows where the predicted class is not equal to the observed class.

```
[210]: new = x_test.copy()
       new['y_predicted'] = y_pred
       new['y_observed'] = y_test
       new[new['y_observed']!=new['y_predicted']]
```

```
[210]:        Age    Income  Family  CCAvg  Mortgage  Securities Account  CD Account  \
       1161   36  5.198497       3   1.40         0                   0           0
       464    43  4.418841       4   3.60         0                   0           0
       3288   56  4.941642       4   0.50       292                   0           0
       4583   52  4.418841       1   3.10         0                   0           0
       1793   35  4.727388       3   0.80         0                   0           0
       2951   26  4.882802       2   2.40         0                   0           0
       3217   65  4.543295       4   4.10       120                   0           1
       1328   60  4.976734       4   6.90       380                   0           0
       3343   62  4.828314       1   1.00         0                   0           0
       349    26  4.094345       2   3.00       132                   0           0
       3141   57  4.875197       3   0.60         0                   0           0
       896    50  5.081404       3   3.40       212                   0           0
       3608   59  5.308268       1   4.70       553                   0           0
       1022   27  4.770685       1   3.30         0                   0           0
       1176   29  4.634729       4   3.40         0                   0           0
       4156   37  5.262690       1   8.60         0                   0           0
       4225   43  5.318120       2   8.80         0                   0           0
       3959   43  4.812184       3   1.30         0                   0           0
       895    43  4.430817       4   2.60       289                   1           1
       2533   54  4.709530       1   1.10         0                   0           0
       3318   46  4.653960       4   3.20         0                   0           0
       2714   46  5.062595       3   5.40       432                   0           0
```

```
3651   49  4.941642       1   1.90        0                   0           0
3312   47  5.247024       2   8.80        0                   0           0
1478   65  5.075174       4   3.80      237                   0           0
3357   32  4.718499       1   2.70      408                   1           1
1784   54  4.779123       3   2.00        0                   1           1
1499   52  4.510860       1   4.30        0                   0           1
4302   52  4.442651       3   3.40        0                   0           0
1870   63  4.700480       1   4.10        0                   0           0
1069   44  4.317488       2   3.50        0                   0           0
528    64  4.804021       4   0.20      378                   0           0
2024   36  4.727388       4   0.20        0                   0           0
4993   45  5.384495       2   6.67        0                   0           0
1177   28  4.262680       1   3.30      149                   1           1
2285   48  4.736198       1   2.40        0                   0           0
1632   31  4.532599       2   3.10        0                   0           0
4016   53  5.153292       4   2.70      427                   0           0
2345   65  4.488636       1   4.10      299                   0           1
927    65  4.553877       3   3.70      138                   0           0
4418   59  4.976734       4   1.80      198                   0           0
3983   39  4.532599       4   3.60        0                   0           0
85     27  4.691348       4   1.80        0                   0           0
1277   45  5.267858       2   8.80      428                   0           0
227    47  4.997212       2   7.50        0                   0           1
382    65  4.890349       4   2.00        0                   0           0
3804   47  5.313206       2   8.80        0                   0           0
4168   60  4.934474       4   0.40        0                   0           0
3517   30  4.553877       1   3.90      146                   0           0
1889   56  4.709530       4   0.30      372                   1           1
3489   36  5.036953       3   6.40        0                   1           0
4179   29  4.510860       1   3.40        0                   0           0

      Online  CreditCard  Graduate  Undergrad  y_predicted  y_observed
1161       0           0         0          1            0           1
464        0           1         0          0            0           1
3288       0           0         0          1            0           1
4583       1           0         0          1            0           1
1793       1           0         0          0            0           1
2951       0           1         0          0            0           1
3217       1           1         0          1            0           1
1328       0           1         0          1            0           1
3343       1           0         0          0            0           1
349        0           0         0          1            0           1
3141       1           0         0          1            0           1
896        1           0         0          1            0           1
3608       0           0         0          1            1           0
1022       1           0         1          0            0           1
1176       1           0         0          1            0           1
```

```
4156        0           0         0          1             1           0
4225        1           0         0          1             1           0
3959        1           0         0          1             0           1
895         1           1         0          0             1           0
2533        1           0         1          0             0           1
3318        0           0         0          1             0           1
2714        0           1         0          1             0           1
3651        0           1         0          0             0           1
3312        0           0         0          1             1           0
1478        1           0         0          1             0           1
3357        1           1         1          0             1           0
1784        0           0         0          1             0           1
1499        1           1         1          0             0           1
4302        0           0         0          0             0           1
1870        0           0         0          0             0           1
1069        1           0         0          1             0           1
528         1           0         0          1             0           1
2024        0           0         0          1             0           1
4993        1           0         0          1             1           0
1177        1           0         1          0             0           1
2285        1           0         0          0             0           1
1632        1           0         1          0             0           1
4016        1           0         0          1             0           1
2345        1           0         0          1             0           1
927         0           1         1          0             0           1
4418        1           0         0          1             0           1
3983        1           0         0          0             0           1
85          0           0         0          0             1           0
1277        0           0         0          1             1           0
227         1           1         0          1             1           0
382         0           1         0          1             0           1
3804        1           0         0          1             1           0
4168        1           0         0          1             0           1
3517        0           1         0          0             0           1
1889        1           0         0          1             0           1
3489        0           0         0          1             0           1
4179        0           0         0          0             0           1
```

6 6) Give conclusion related to the Business understanding of your model? (5 marks)

Data from a single test/train split showed:

The model predicted 98 people would open savings accounts. Of those, 87 opened accounts, resulting in 89% precision.
128 people opened savings accounts. Of those persons, the model correctly identified 87, resulting in 68% recall.

From a business perspective, the bank is trying to maximize number of people opening accounts. Therefore having a model with high precision allows the bank to focus efforts people who are likely to open accounts. Ideally, the bank would probably prefer a model with higher recall since the cost of a non-open is relatively low. While the gains from one additional opened account are high.

10 Fold cross validation showed the following metrics for this model:

The mean precision score is 0.90

The mean recall score is 0.69

The mean accuracy score is 0.96

The mean f1 score is 0.78

The mean roc_auc score is 0.97

```
[ ]:
```