Business Problem 1:

Classify whether application accepted or not using Logistic regression

card

Factor. Was the application for a credit card accepted?

reports

Number of major derogatory reports.

age

Age in years plus twelfths of a year.

income

Yearly income (in USD 10,000).

share

Ratio of monthly credit card expenditure to yearly income.

expenditure

Average monthly credit card expenditure.

owner

Factor. Does the individual own their home?

selfemp

Factor. Is the individual self-employed?

dependents

Number of dependents.

months

Months living at current address.

majorcards

Number of major credit cards held.

Active

Solution:

Here in the given business problem whether the application of the applicant is accepted or rejected later whether the applicant did get the card or not and all other variables are considered as inputs.

Y = cards

X = reports, age, income, share, expenditure, owner, selfemp, dependents, months, majorcards, Active.

Step 1:

Loading Data:

Codes:

```
import pandas as pd

import numpy  as np

import matplotlib.pyplot as plt


cc = pd.read_csv("E:\Data\Assignments\i made\Logistic Regression\creditcard.csv")

cc.head()

cc.columns
```

Step 2:

Data Preprocessing:

1 In the given data, the serial number (sl_no) column is not required hence we need to remove the column.

Codes:

```
cc.drop(["sl"],axis=1,inplace=True)
```

2 As we can observe there are three Boolean columns are present in the dataset, so we need to convert those into zeros and ones.

Codes:

```
from sklearn import preprocessing

le = preprocessing.LabelEncoder()


cc['card'] = le.fit_transform(cc['card'])

cc['owner'] = le.fit_transform(cc['owner'])

cc['selfemp'] = le.fit_transform(cc['selfemp'])
```

3 To check whether the null values are present or not

Codes:

```
cc.isnull().sum()
```

The output for all variables are zero, so it is concluded that there are no values null are present.

Step 3:

Model Building:

Codes:

```
from sklearn.linear_model import LogisticRegression

X = cc.iloc[:,[1,2,3,4,5,6,7,8,9,10,11]]

Y = cc.iloc[:,0]

model1 = LogisticRegression()

model1.fit(X,Y)

model1.coef_

model1.predict_proba (X)
```

Output:

```
([-1.65509892e+00, -3.11259045e-03, -2.05627500e-01,
  -6.17010295e-04,  1.61855364e+00,  6.20454990e-01,
   2.29555725e-01, -6.64242088e-01, -1.60414470e-03,
   3.43439831e-02,  7.65660467e-02]])
```

Step 4:

Predictions:

Codes:

```
y_pred = model1.predict(X)
cc["y_pred"] = y_pred
y_prob = pd.DataFrame(model1.predict_proba(X.iloc[:,:]))
new_df = pd.concat([cc,y_prob],axis=1)
```

Step 5:

Calculating the accuracy of the model, we can either use Confusion matrix or ROC curve to calculate the accuracy.

1 Confusion matrix method:

Codes:

```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(Y,y_pred)
print (confusion_matrix)
type(y_pred)
accuracy = sum(Y==y_pred)/cc.shape[0]
pd.crosstab(y_pred,Y)
```

Output:

```
card    0    1

row_0

0     295    23

1       1  1000
```

Accuracy = True positive + True negative / Total

Accuracy = 1000 + 295 / 1319 = 0.9818043972706596

Accuracy = 0.9818043972706596

2 ROC method:

The function is applicable for binary classification class

```
from sklearn import metrics

fpr, tpr, threshold = metrics.roc_curve(cc.card, y_pred)
```

where,

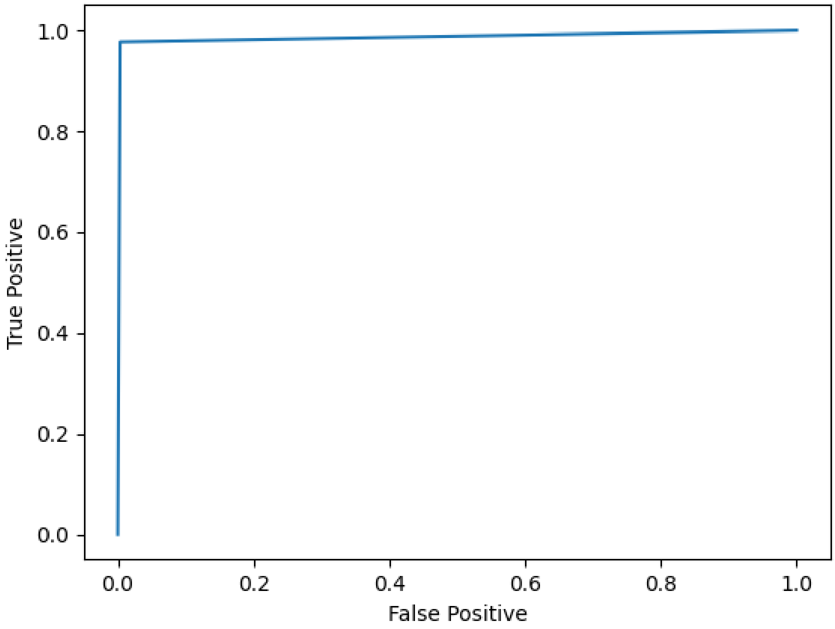
fpr => false positive rate

tpr => true positive rate

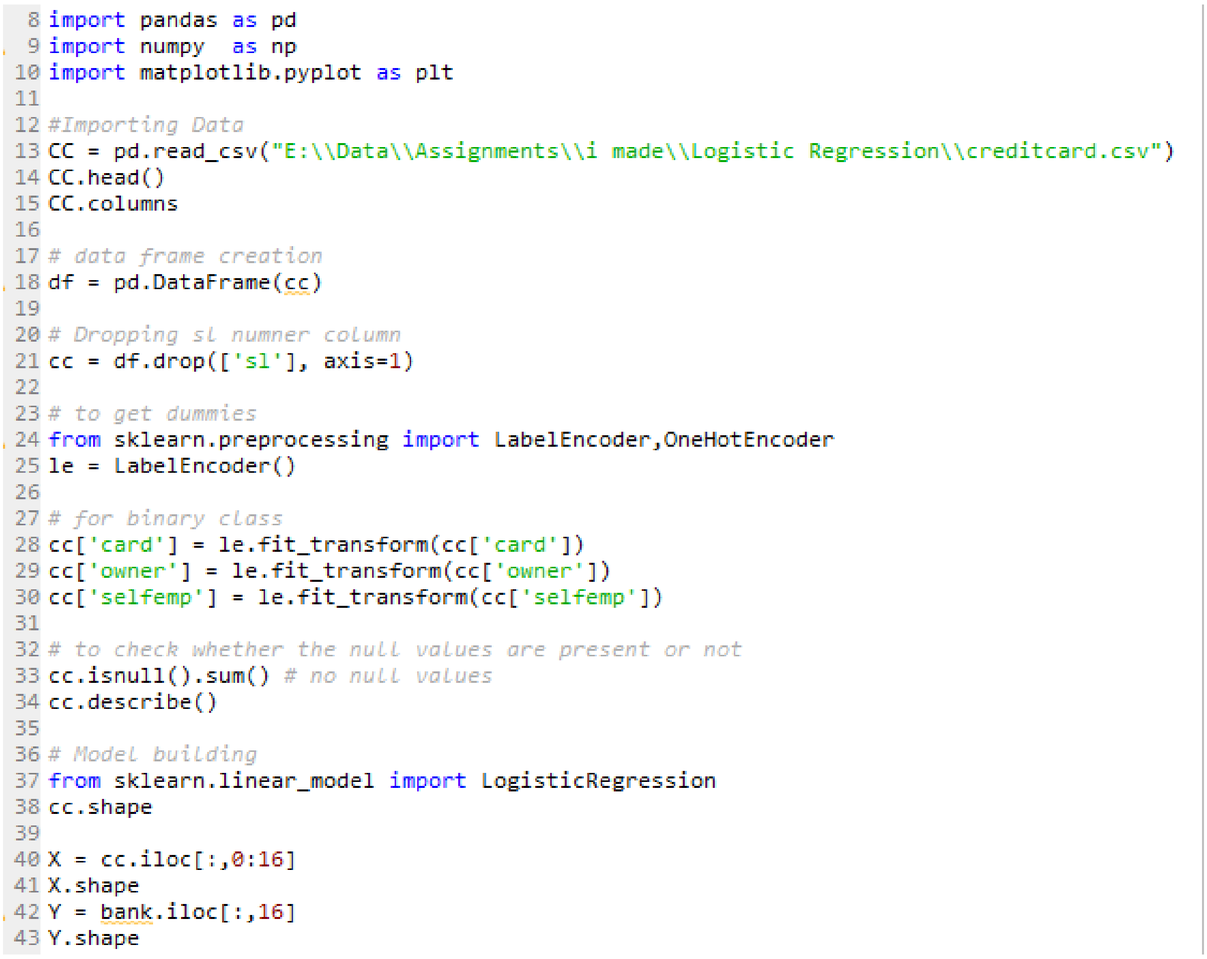
```
plt.plot(fpr,tpr);plt.xlabel("False Positive");plt.ylabel("True Positive")
```

Calculating area under roc curve

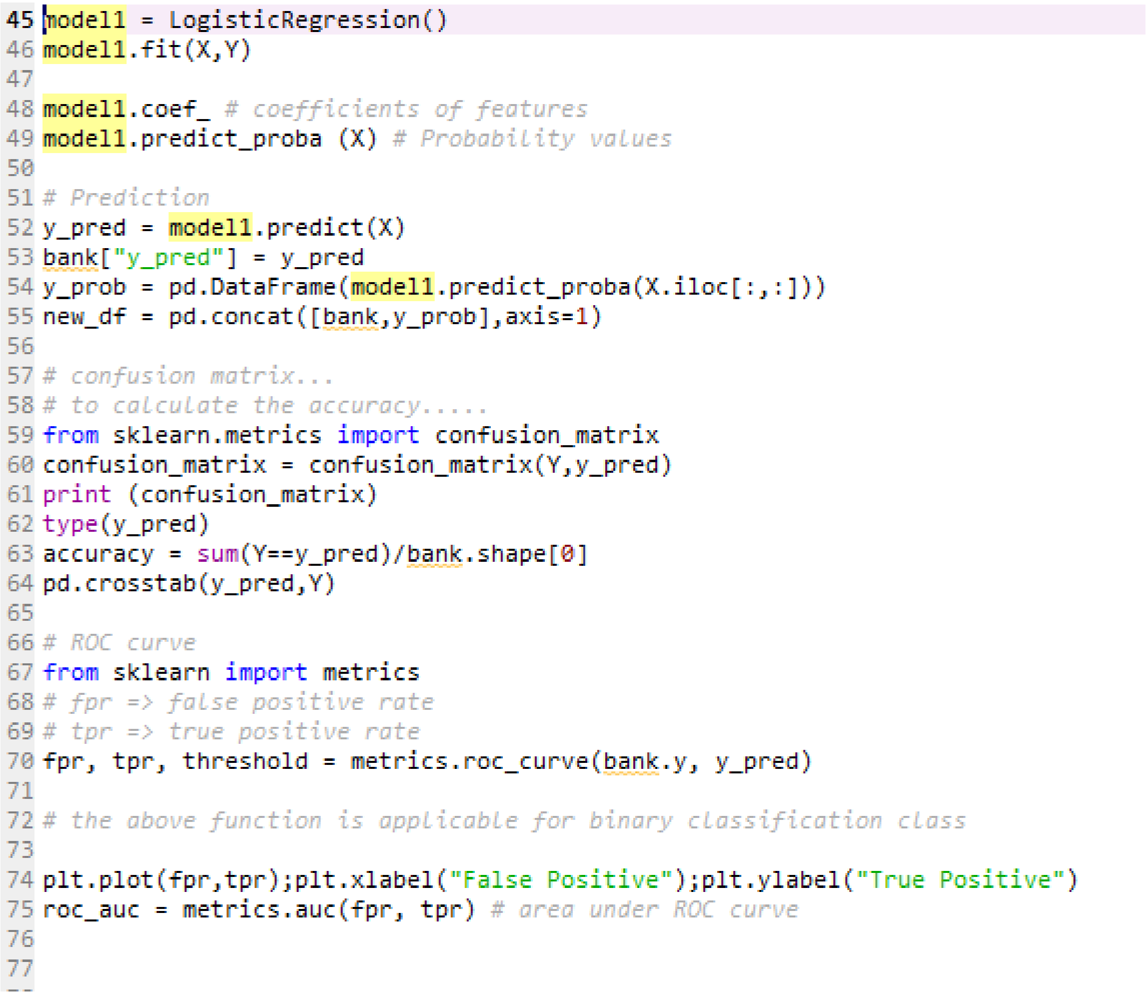
```
roc_auc = metrics.auc(fpr, tpr)
```



Aur_auc = 0.9870693640854931



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

#Importing Data
CC = pd.read_csv("E:\\Data\\Assignments\\i made\\Logistic Regression\\creditcard.csv")
CC.head()
CC.columns

# data frame creation
df = pd.DataFrame(cc)

# Dropping sl numner column
cc = df.drop(['sl'], axis=1)

# to get dummies
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
le = LabelEncoder()

# for binary class
cc['card'] = le.fit_transform(cc['card'])
cc['owner'] = le.fit_transform(cc['owner'])
cc['selfemp'] = le.fit_transform(cc['selfemp'])

# to check whether the null values are present or not
cc.isnull().sum() # no null values
cc.describe()

# Model building
from sklearn.linear_model import LogisticRegression
cc.shape

X = cc.iloc[:,0:16]
X.shape
Y = bank.iloc[:,16]
Y.shape
```



```
model1 = LogisticRegression()
model1.fit(X,Y)

model1.coef_ # coefficients of features
model1.predict_proba (X) # Probability values

# Prediction
y_pred = model1.predict(X)
bank["y_pred"] = y_pred
y_prob = pd.DataFrame(model1.predict_proba(X.iloc[:,:]))
new_df = pd.concat([bank,y_prob],axis=1)

# confusion matrix...
# to calculate the accuracy.....
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(Y,y_pred)
print (confusion_matrix)
type(y_pred)
accuracy = sum(Y==y_pred)/bank.shape[0]
pd.crosstab(y_pred,Y)

# ROC curve
from sklearn import metrics
# fpr => false positive rate
# tpr => true positive rate
fpr, tpr, threshold = metrics.roc_curve(bank.y, y_pred)

# the above function is applicable for binary classification class

plt.plot(fpr,tpr);plt.xlabel("False Positive");plt.ylabel("True Positive")
roc_auc = metrics.auc(fpr, tpr) # area under ROC curve


```

Business Problem 2:

Input variables:

bank client data:

1 - age (numeric)

2 - job : type of job categorical:"admin.","unknown","unemployed","management","housemaid","entrepreneur","student", "blue-collar", "self-employed", "retired", "technician", "services")

3 - marital : marital status (categorical: "married", "divorced", "single" ; note: "divorced" means divorced or widowed)

4 - education (categorical: "unknown","secondary","primary","tertiary")

5 - default: has credit in default? (binary: "yes","no")

6 - balance: average yearly balance, in euros (numeric)

7 - housing: has housing loan? (binary: "yes","no")

8 - loan: has personal loan? (binary: "yes","no")

# related with the last contact of the current campaign:

9 - contact: contact communication type (categorical: "unknown","telephone","cellular")

10 - day: last contact day of the month (numeric)

11 - month: last contact month of year (categorical: "jan", "feb", "mar", ..., "nov", "dec")

12 - duration: last contact duration, in seconds (numeric)

# other attributes:

13 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)

14 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted)

15 - previous: number of contacts performed before this campaign and for this client (numeric)

16 - poutcome: outcome of the previous marketing campaign (categorical: "unknown","other","failure","success")

Output variable (desired target):

17 - y - has the client subscribed a term deposit? (binary: "yes","no")

Missing Attribute Values: None

Solution:

Here in the given business problem whether the client subscribed a term deposit or not deposited. The y variable in the dataset is treated as output and all other variables are considered as inputs.

Dependent = y

Independent = ‘age', 'job', 'marital', 'education', 'default', 'balance', 'housing', 'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays', 'previous', 'poutcome’.

Step 1:

Loading Data:

Codes:

```
import pandas as pd

import numpy  as np

import matplotlib.pyplot as plt


bank = pd.read_csv("E:\Data\Assignments\i made\Logistic Regression\bank.csv")

bank.head()

bank.columns
```

Step 2:

Data Preprocessing:

1 As we can observe there are four Boolean (binary class) columns are present in the dataset, so we need to convert those into zeros and ones.

Codes:

To get dummies

```
from sklearn.preprocessing import LabelEncoder,OneHotEncoder

le = LabelEncoder()
```

For binary class

```
bank['default'] = le.fit_transform(bank['default'])

bank['housing'] = le.fit_transform(bank['housing'])

bank['loan'] = le.fit_transform(bank['loan'])

bank['y'] = le.fit_transform(bank['y'])
```

For more than 2 class

```
bank['job'] = le.fit_transform(bank['job'])

bank['marital'] = le.fit_transform(bank['marital'])

bank['education'] = le.fit_transform(bank['education'])

bank['contact'] = le.fit_transform(bank['contact'])

bank['poutcome'] = le.fit_transform(bank['poutcome'])

bank['month'] = le.fit_transform(bank['month'])
```

2 To check whether the null values are present or not

Codes:

```
bank.isnull().sum()
```

The output for all variables are zero, so it is concluded that there are no null values are present.

Step 3:

Model Building:

Codes:

```
from sklearn.linear_model import LogisticRegression

X = bank.iloc[:,0:16]

Y = bank.iloc[:,0]

model1 = LogisticRegression()

model1.fit(X,Y)

model1.coef_

model1.predict_proba (X)
```

Output:

```
([[ 5.26365457e-03,  6.79923567e-03,  1.74529110e-01,

1.89676601e-01, -3.17584084e-01,  1.79848768e-05,

 -1.04381568e+00, -7.15651301e-01, -6.40323125e-01,

 -6.28799422e-03,  3.56202089e-02,  3.91362534e-03,

 -1.33252062e-01,  3.16245830e-03,  8.43875303e-02,

1.68155146e-01]])  -6.17010295e-04,  1.61855364e+00,

6.20454990e-01, 2.29555725e-01, -6.64242088e-01,

 -1.60414470e-03,   3.43439831e-02,  7.65660467e-02]])
```

Step 4:

Predictions:

Codes:

```
y_pred = model1.predict(X)

bank["y_pred"] = y_pred
```

```
y_prob = pd.DataFrame(model1.predict_proba(X.iloc[:,:]))

new_df = pd.concat([bank,y_prob],axis=1)
```

Step 5:

Calculating the accuracy of the model, we can either use Confusion matrix or ROC curve to calculate the accuracy.

1 Confusion matrix method:

Codes:

```
from sklearn.metrics import confusion_matrix

confusion_matrix = confusion_matrix(Y,y_pred)

print (confusion_matrix)

type(y_pred)

accuracy = sum(Y==y_pred)/cc.shape[0]

pd.crosstab(y_pred,Y)
```

Output:

```
y        0    1

row_0

0     39142  4157

1     780     1132
```

Accuracy = True positive + True negative / Total

Accuracy = 39142 + 1132 / 45213 = 0.8908009112826525

Accuracy = 0.8908009112826525

2 ROC method:

The function is applicable for binary classification class

```
from sklearn import metrics
```

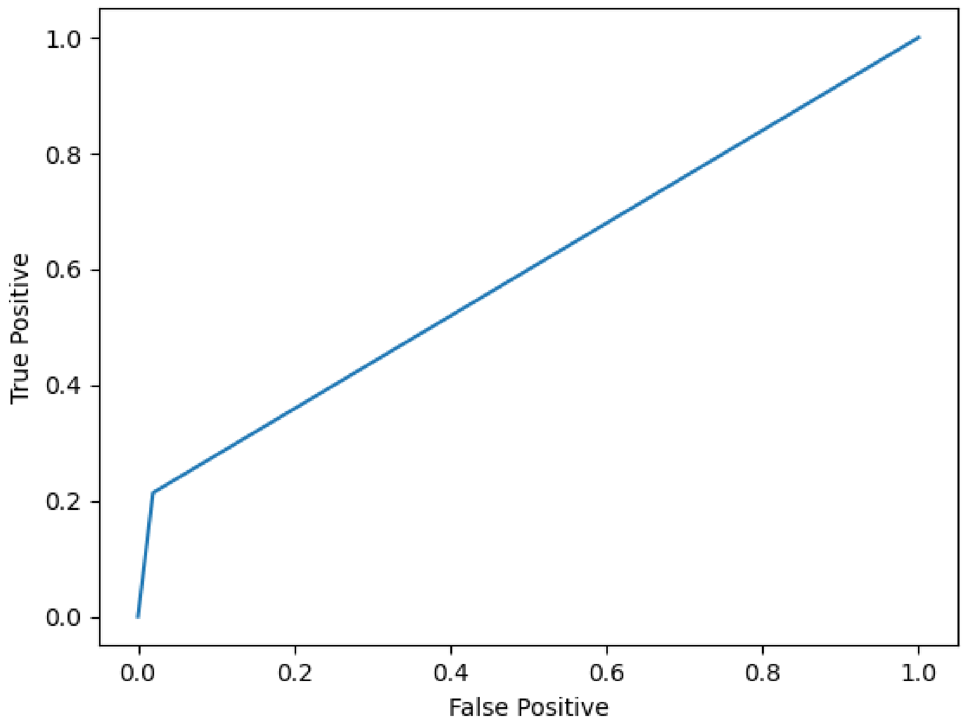
```
fpr, tpr, threshold = metrics.roc_curve(bank.card, y_pred)
```

where,

fpr => false positive rate

tpr => true positive rate

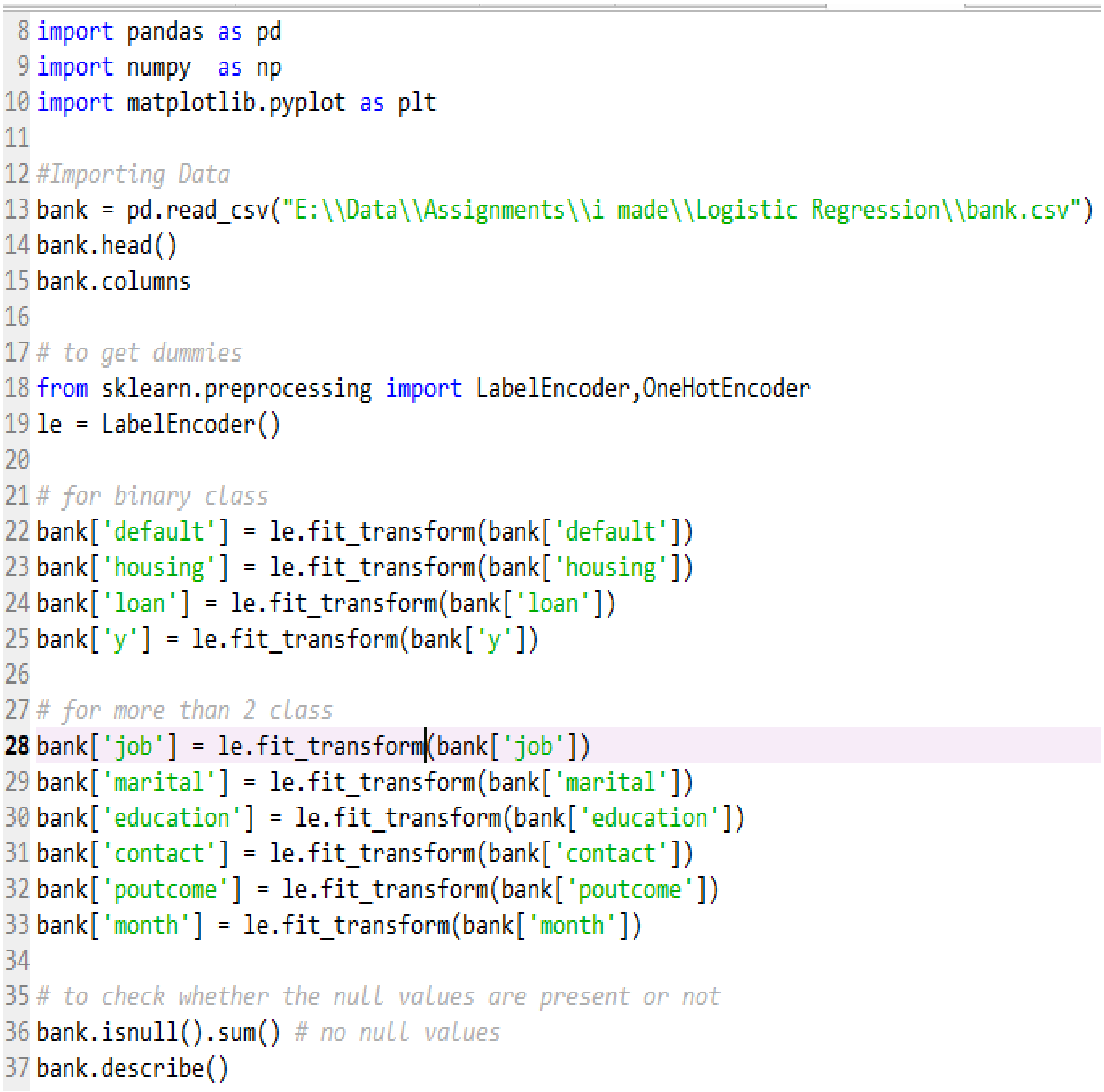
```
plt.plot(fpr,tpr);plt.xlabel("False Positive");plt.ylabel("True Positive")
```



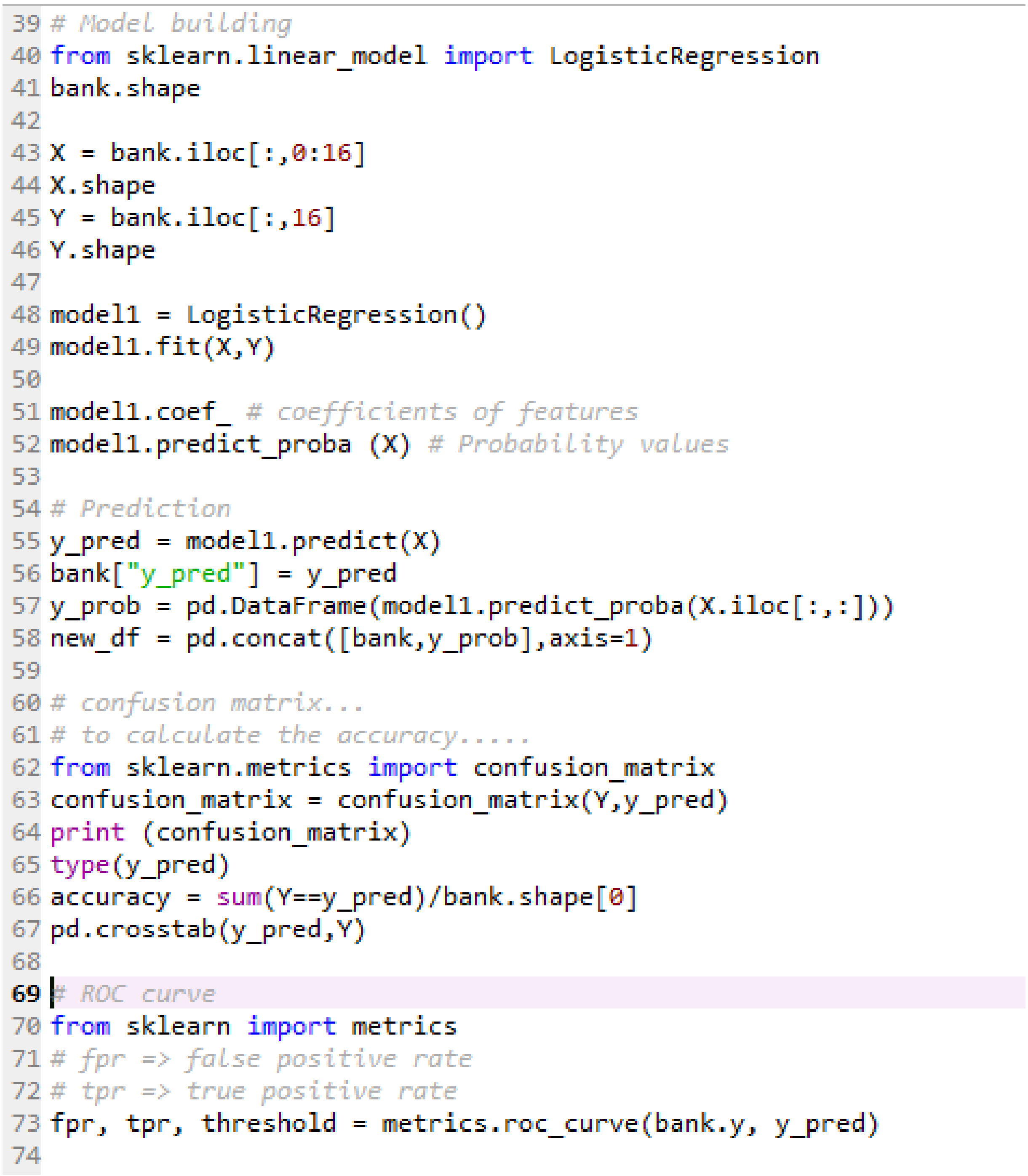
Calculating area under roc curve

```
roc_auc = metrics.auc(fpr, tpr)
```

Aur_auc = 0.5972455088708668



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

#Importing Data
bank = pd.read_csv("E:\\Data\\Assignments\\i made\\Logistic Regression\\bank.csv")
bank.head()
bank.columns

# to get dummies
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
le = LabelEncoder()

# for binary class
bank['default'] = le.fit_transform(bank['default'])
bank['housing'] = le.fit_transform(bank['housing'])
bank['loan'] = le.fit_transform(bank['loan'])
bank['y'] = le.fit_transform(bank['y'])

# for more than 2 class
bank['job'] = le.fit_transform(bank['job'])
bank['marital'] = le.fit_transform(bank['marital'])
bank['education'] = le.fit_transform(bank['education'])
bank['contact'] = le.fit_transform(bank['contact'])
bank['poutcome'] = le.fit_transform(bank['poutcome'])
bank['month'] = le.fit_transform(bank['month'])

# to check whether the null values are present or not
bank.isnull().sum() # no null values
bank.describe()
```



```
# Model building
from sklearn.linear_model import LogisticRegression
bank.shape

X = bank.iloc[:,0:16]
X.shape
Y = bank.iloc[:,16]
Y.shape

model1 = LogisticRegression()
model1.fit(X,Y)

model1.coef_ # coefficients of features
model1.predict_proba (X) # Probability values

# Prediction
y_pred = model1.predict(X)
bank["y_pred"] = y_pred
y_prob = pd.DataFrame(model1.predict_proba(X.iloc[:,:]))
new_df = pd.concat([bank,y_prob],axis=1)

# confusion matrix...
# to calculate the accuracy.....
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(Y,y_pred)
print (confusion_matrix)
type(y_pred)
accuracy = sum(Y==y_pred)/bank.shape[0]
pd.crosstab(y_pred,Y)

# ROC curve
from sklearn import metrics
# fpr => false positive rate
# tpr => true positive rate
fpr, tpr, threshold = metrics.roc_curve(bank.y, y_pred)

```