Loan prediction

August 31, 2023

[139]:
```
import csv
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OrdinalEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB, GaussianNB, BernoulliNB,␣
↪CategoricalNB
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_auc_score, roc_curve, auc
from sklearn.metrics import precision_recall_curve, average_precision_score
from sklearn.metrics import log_loss, balanced_accuracy_score
```

[140]:
```
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
```

TRAINING DATA

[141]:
```
print(train.head())
```

```
    Loan_ID Gender Married Dependents     Education Self_Employed  \
0  LP001002   Male      No          0      Graduate            No
1  LP001003   Male     Yes          1      Graduate            No
2  LP001005   Male     Yes          0      Graduate           Yes
3  LP001006   Male     Yes          0  Not Graduate            No
4  LP001008   Male      No          0      Graduate            No

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5849                0.0         NaN             360.0
1             4583             1508.0       128.0             360.0
2             3000                0.0        66.0             360.0
3             2583             2358.0       120.0             360.0
4             6000                0.0       141.0             360.0

   Credit_History Property_Area Loan_Status
```

```
0              1.0         Urban           Y
1              1.0         Rural           N
2              1.0         Urban           Y
3              1.0         Urban           Y
4              1.0         Urban           Y
```

[142]:
```python
print(train.isnull().sum())    # finding missing values
```

```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

[143]:
```python
print(train['Gender'].unique())
```

```
['Male' 'Female' nan]
```

[144]:
```python
print(train['Dependents'].unique())
```

```
['0' '1' '2' '3+' nan]
```

[145]:
```python
print(train['Education'].unique())
```

```
['Graduate' 'Not Graduate']
```

[146]:
```python
print(train['Self_Employed'].unique())
```

```
['No' 'Yes' nan]
```

[147]:
```python
print(train['Property_Area'].unique())
```

```
['Urban' 'Rural' 'Semiurban']
```

[148]:
```python
encoder = OrdinalEncoder()
df = train.copy()
cate = ['Dependents']
df[cate] = encoder.fit_transform(train[cate])
Dependents = df[cate]
```

[149]:
```
imputer = SimpleImputer()
depn = Dependents.copy()
depn = pd.DataFrame(imputer.fit_transform(Dependents))
```

[150]:
```
train['Dependents'] = depn
#print(train)
```

[151]:
```
X = train.copy()
X = X.drop(['Loan_ID', 'Loan_Status'], axis=1)       # dropping columns
```

[152]:
```
cate_col = (X.dtypes == 'object')
cate_col = list(cate_col[cate_col].index)
print("Categorical Variables : ", cate_col)
```

```
Categorical Variables :  ['Gender', 'Married', 'Education', 'Self_Employed',
'Property_Area']
```

[153]:
```
X[cate_col] = encoder.fit_transform(df[cate_col])
#print(X)
```

[154]:
```
X_prep = X.copy()
X_prep = pd.DataFrame(imputer.fit_transform(X))
X_prep.columns = X.columns
#print(X_prep)                     # preprocessed X - features
```

[155]:
```
df2 = train.copy()
cate2 = ['Loan_Status']
df2[cate2] = encoder.fit_transform(train[cate2])
y_prep = df2[cate2].copy()
```

[156]:
```
#print(y_prep)
```

TEST DATA

[157]:
```
print(test.head())
```

```
    Loan_ID Gender Married Dependents     Education Self_Employed  \
0  LP001015   Male     Yes          0      Graduate            No
1  LP001022   Male     Yes          1      Graduate            No
2  LP001031   Male     Yes          2      Graduate            No
3  LP001035   Male     Yes          2      Graduate            No
4  LP001051   Male      No          0  Not Graduate            No

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5720                  0       110.0             360.0
1             3076               1500       126.0             360.0
2             5000               1800       208.0             360.0
3             2340               2546       100.0             360.0
```

```
4              3276                     0         78.0                360.0

   Credit_History Property_Area
0             1.0         Urban
1             1.0         Urban
2             1.0         Urban
3             NaN         Urban
4             1.0         Urban
```

[158]:
```python
print(test.isnull().sum())
```

```
Loan_ID               0
Gender               11
Married               0
Dependents           10
Education             0
Self_Employed        23
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            5
Loan_Amount_Term      6
Credit_History       29
Property_Area         0
dtype: int64
```

[159]:
```python
print(test['Gender'].unique())
```

```
['Male' 'Female' nan]
```

[160]:
```python
print(test['Dependents'].unique())
```

```
['0' '1' '2' '3+' nan]
```

[161]:
```python
print(test['Education'].unique())
```

```
['Graduate' 'Not Graduate']
```

[162]:
```python
print(test['Self_Employed'].unique())
```

```
['No' 'Yes' nan]
```

[163]:
```python
print(test['Property_Area'].unique())
```

```
['Urban' 'Semiurban' 'Rural']
```

[164]:
```python
encoder = OrdinalEncoder()
df3 = test.copy()
cate = ['Dependents']
```

```
df3[cate] = encoder.fit_transform(test[cate])
Dependents2 = df3[cate]
```

[165]:
```
imputer = SimpleImputer()
depn2 = Dependents2.copy()
depn2 = pd.DataFrame(imputer.fit_transform(Dependents2))
```

[166]:
```
test['Dependents'] = depn2
#print(test)
```

[167]:
```
X2 = test.copy()
X2 = X2.drop(['Loan_ID'], axis=1)       # dropping columns
#print(X2)
```

[168]:
```
X2[cate_col] = encoder.fit_transform(df3[cate_col])
#print(X2)
```

[169]:
```
X_test_prep = X2.copy()
X_test_prep = pd.DataFrame(imputer.fit_transform(X2))
X_test_prep.columns = X2.columns
#print(X_test_prep)                    # preprocessed X - features
```

PREPROCESSED DATA

[170]:
```
print(X_prep)
```

```
     Gender  Married  Dependents  Education  Self_Employed  ApplicantIncome  \
0       1.0      0.0         0.0        0.0            0.0           5849.0
1       1.0      1.0         1.0        0.0            0.0           4583.0
2       1.0      1.0         0.0        0.0            1.0           3000.0
3       1.0      1.0         0.0        1.0            0.0           2583.0
4       1.0      0.0         0.0        0.0            0.0           6000.0
..      …        …          …          …              …              …
609     0.0      0.0         0.0        0.0            0.0           2900.0
610     1.0      1.0         3.0        0.0            0.0           4106.0
611     1.0      1.0         1.0        0.0            0.0           8072.0
612     1.0      1.0         2.0        0.0            0.0           7583.0
613     0.0      0.0         0.0        0.0            1.0           4583.0

     CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  \
0                  0.0  146.412162             360.0             1.0
1               1508.0  128.000000             360.0             1.0
2                  0.0   66.000000             360.0             1.0
3               2358.0  120.000000             360.0             1.0
4                  0.0  141.000000             360.0             1.0
..                 …           …                …               …
609                0.0   71.000000             360.0             1.0
610                0.0   40.000000             180.0             1.0
```

```
611                240.0  253.000000            360.0              1.0
612                  0.0  187.000000            360.0              1.0
613                  0.0  133.000000            360.0              0.0

     Property_Area
0              2.0
1              0.0
2              2.0
3              2.0
4              2.0
..             …
609            0.0
610            0.0
611            2.0
612            2.0
613            1.0

[614 rows x 11 columns]
```

[171]:
```
print(y_prep)
```

```
     Loan_Status
0            1.0
1            0.0
2            1.0
3            1.0
4            1.0
..           …
609          1.0
610          1.0
611          1.0
612          1.0
613          0.0

[614 rows x 1 columns]
```

[172]:
```
print(X_test_prep)
```

```
     Gender  Married  Dependents  Education  Self_Employed  ApplicantIncome  \
0       1.0      1.0         0.0        0.0            0.0           5720.0
1       1.0      1.0         1.0        0.0            0.0           3076.0
2       1.0      1.0         2.0        0.0            0.0           5000.0
3       1.0      1.0         2.0        0.0            0.0           2340.0
4       1.0      0.0         0.0        1.0            0.0           3276.0
..      …        …           …          …              …              …
362     1.0      1.0         3.0        1.0            1.0           4009.0
363     1.0      1.0         0.0        0.0            0.0           4158.0
364     1.0      0.0         0.0        0.0            0.0           3250.0
```

```
365     1.0      1.0          0.0         0.0             0.0            5000.0
366     1.0      0.0          0.0         0.0             1.0            9200.0

     CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  \
0                  0.0       110.0             360.0        1.000000
1               1500.0       126.0             360.0        1.000000
2               1800.0       208.0             360.0        1.000000
3               2546.0       100.0             360.0        0.825444
4                  0.0        78.0             360.0        1.000000
..                 …           …                 …             …
362             1777.0       113.0             360.0        1.000000
363              709.0       115.0             360.0        1.000000
364             1993.0       126.0             360.0        0.825444
365             2393.0       158.0             360.0        1.000000
366                0.0        98.0             180.0        1.000000

     Property_Area
0              2.0
1              2.0
2              2.0
3              2.0
4              2.0
..             …
362            2.0
363            2.0
364            1.0
365            0.0
366            0.0

[367 rows x 11 columns]
```

TRAINING DATA SPLIT

[173]:
```
X_train, X_valid, y_train, y_valid = train_test_split(X_prep, y_prep,␣
 ↪test_size=0.25, random_state=0)
```

[174]:
```
X_train.reset_index(inplace=True)            # resetting index
X_train = X_train.drop('index', axis=1)
```

[175]:
```
y_train.reset_index(inplace=True)
y_train = y_train.drop('index', axis=1)
```

[176]:
```
X_valid.reset_index(inplace=True)
X_valid = X_valid.drop('index', axis=1)
```

[177]:
```
y_valid.reset_index(inplace=True)
y_valid = y_valid.drop('index', axis=1)
```

[178]:
```
### 1. NAIVE BAYES CLASSIFIER
```

[179]:
```
model = GaussianNB()
```

[180]:
```
model.fit(X_train, y_train)
pred = model.predict(X_valid)
print(classification_report(y_valid, pred))
```

```
              precision    recall  f1-score   support

         0.0       0.83      0.47      0.60        43
         1.0       0.82      0.96      0.89       111

    accuracy                           0.82       154
   macro avg       0.83      0.71      0.74       154
weighted avg       0.83      0.82      0.81       154


/home/niranjan/.local/lib/python3.8/site-
packages/sklearn/utils/validation.py:1111: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the shape of y
to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

[181]:
```
matrix = confusion_matrix(y_valid, pred)
matrix_df = pd.DataFrame(matrix)
print(matrix_df)
```

```
    0    1
0  20   23
1   4  107
```

0 - No loan | 1 - loan accepted

[182]:
```
test_pred = model.predict(X_test_prep)
print(test_pred)
```

```
[1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 0. 0. 1. 0. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 1. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1.
 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0.
 1. 1. 1. 0. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 0. 1.
 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 0. 1.
 0. 1. 1. 1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 0. 0. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 0. 1. 1. 1. 1. 0. 0. 0. 1. 1.
 1. 0. 1. 0. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 0. 1. 0. 1. 1. 1. 1. 0. 0. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1.
```

```
 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.
 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1.]
```

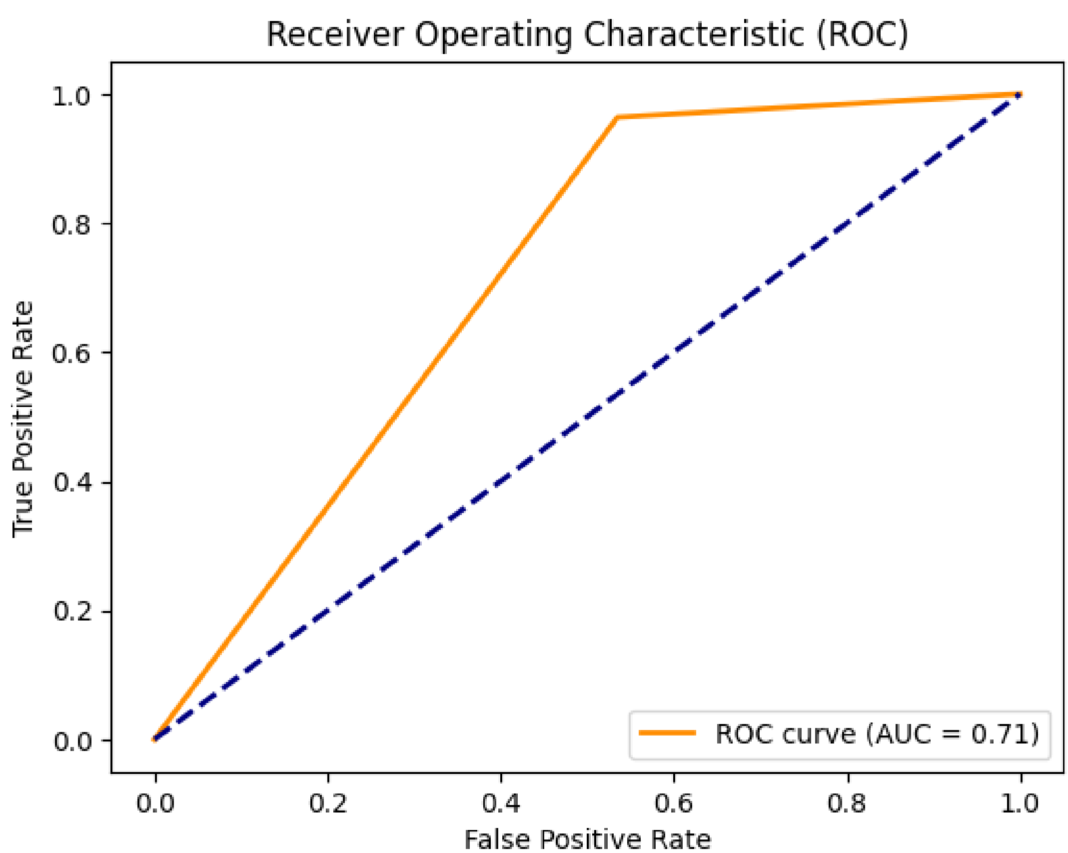
[183]:

```
print(roc_auc)
```

```
0.7010266080033523
```

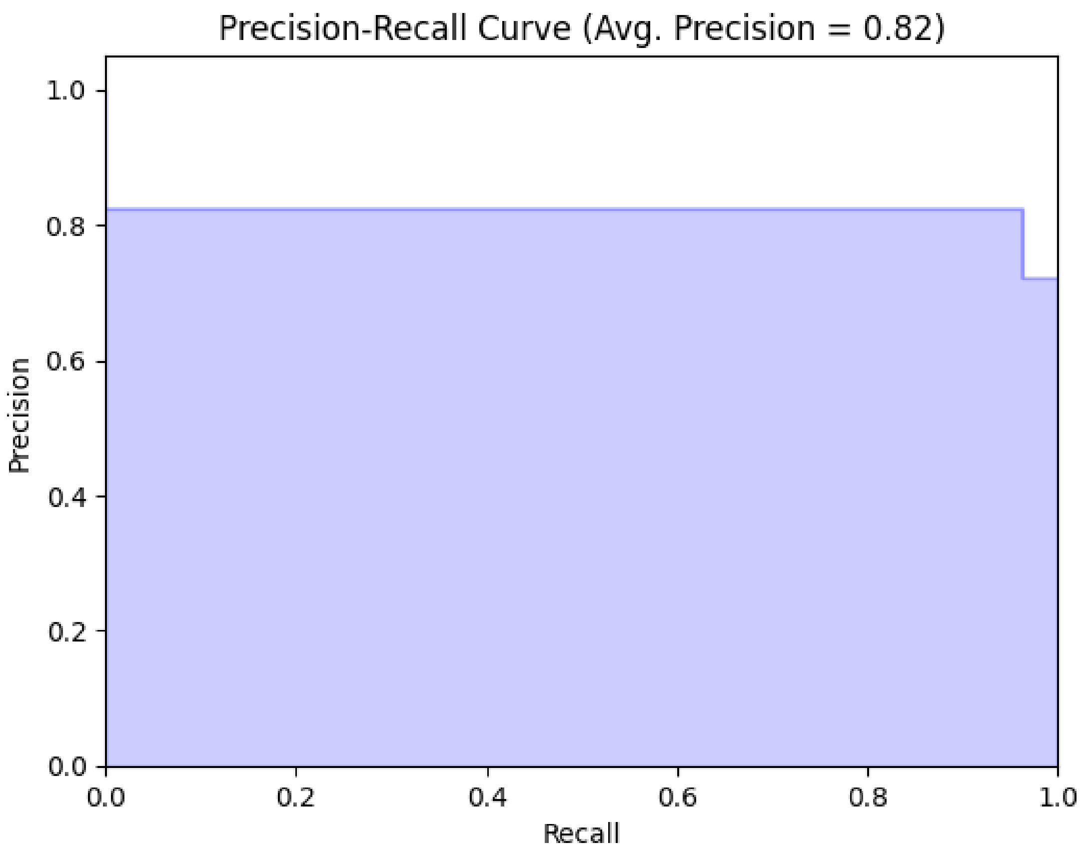
[184]:

```
fpr, tpr, thresholds = roc_curve(y_valid, pred)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (AUC = {:.2f})'.
 ↪format(roc_auc))
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc='lower right')
plt.show()

print("AUC ROC Score:",roc_auc)
```



AUC ROC Score: 0.7145401215168657

[185]:
```
precision, recall, thresholds = precision_recall_curve(y_valid, pred)
average_precision = average_precision_score(y_valid, pred)

plt.figure()
plt.step(recall, precision, color='b', alpha=0.2, where='post')
plt.fill_between(recall, precision, step='post', alpha=0.2, color='b')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.ylim([0.0, 1.05])
plt.xlim([0.0, 1.0])
plt.title('Precision-Recall Curve (Avg. Precision = {:.2f})'.
 ↪format(average_precision))
plt.show()
```



[186]:
```
# other classification metrics

print('avg precision:', average_precision)

balanced_accuracy = balanced_accuracy_score(y_valid, pred)
print('balanced accuracy:', balanced_accuracy)

logloss = log_loss(y_valid, pred)
print("Log Loss:", logloss)
```

```
avg precision: 0.8193905193905194
balanced accuracy: 0.7145401215168657
Log Loss: 6.0556191779446955
```

[187]:
```
### 2. RFC
```

[188]:
```
model2 = RandomForestClassifier(n_estimators=100, criterion='entropy')
```

[189]:
```
model2.fit(X_train, y_train)
pred2 = model2.predict(X_valid)
print(classification_report(y_valid, pred2))
```

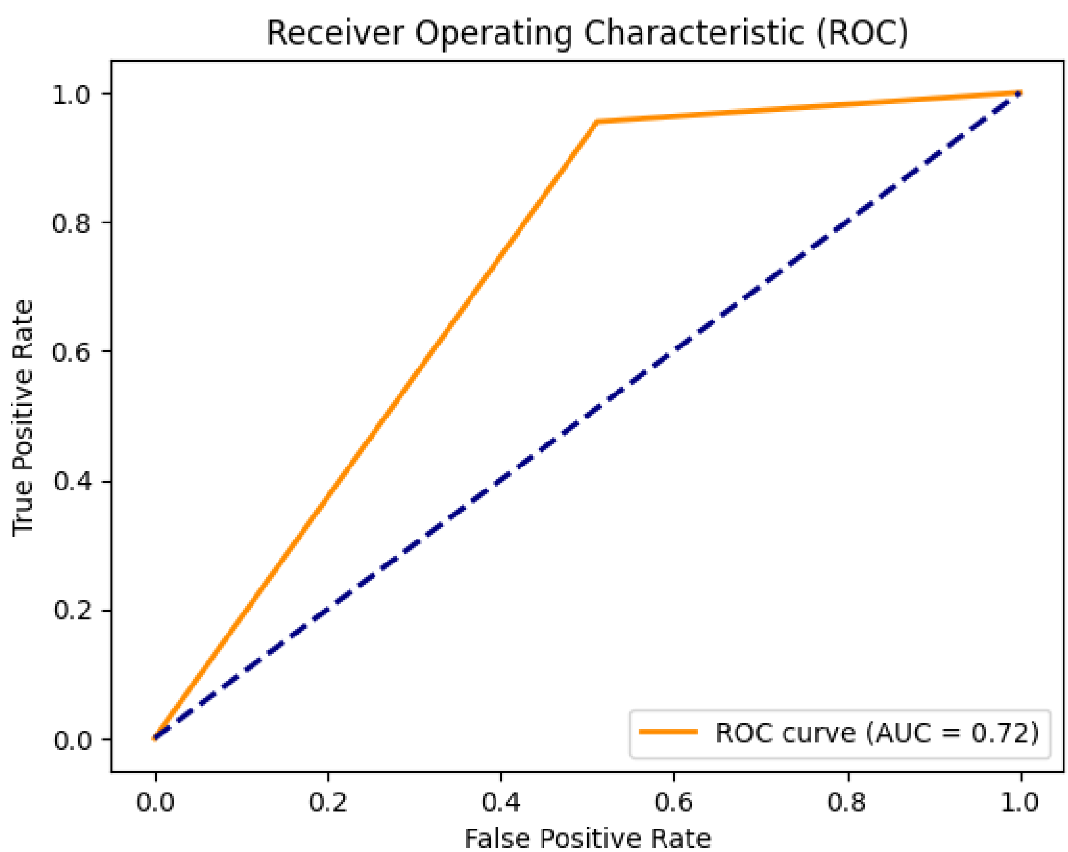
```
              precision    recall  f1-score   support

         0.0       0.81      0.49      0.61        43
         1.0       0.83      0.95      0.89       111

    accuracy                           0.82       154
   macro avg       0.82      0.72      0.75       154
weighted avg       0.82      0.82      0.81       154


/tmp/ipykernel_7019/669114507.py:1: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  model2.fit(X_train, y_train)
```

[190]:
```
matrix = confusion_matrix(y_valid, pred2)
matrix_df = pd.DataFrame(matrix)
print(matrix_df)
```

```
    0    1
0  21   22
1   5  106
```

[191]:
```
fpr, tpr, thresholds = roc_curve(y_valid, pred2)
roc_auc = auc(fpr, tpr)
```

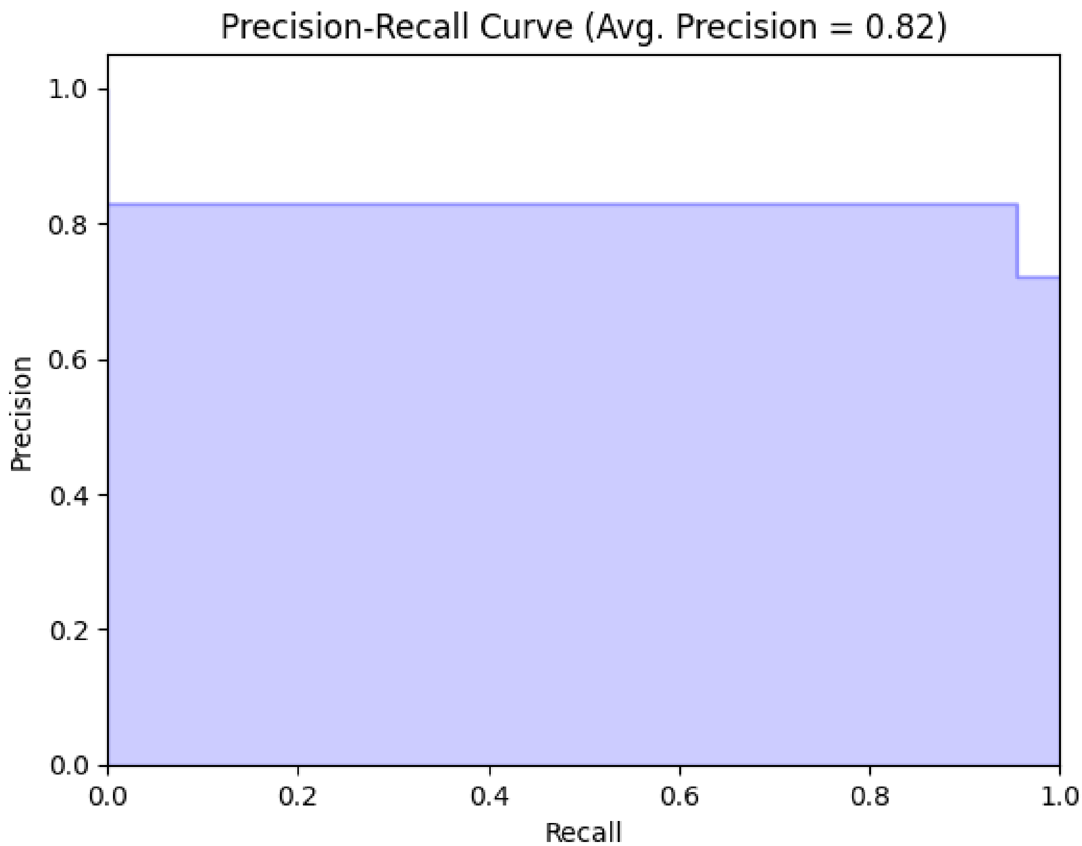
```
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (AUC = {:.2f})'.
    ↪format(roc_auc))
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc='lower right')
plt.show()

print("AUC ROC Score:",roc_auc)
```



AUC ROC Score: 0.7216635239891054

[192]:
```
precision, recall, thresholds = precision_recall_curve(y_valid, pred2)
average_precision = average_precision_score(y_valid, pred2)

plt.figure()
plt.step(recall, precision, color='b', alpha=0.2, where='post')
plt.fill_between(recall, precision, step='post', alpha=0.2, color='b')
```

```
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.ylim([0.0, 1.05])
plt.xlim([0.0, 1.0])
plt.title('Precision-Recall Curve (Avg. Precision = {:.2f})'.
 ↪format(average_precision))
plt.show()
```



[193]:
```
# other metrics

print('avg precision:', average_precision)

balanced_accuracy = balanced_accuracy_score(y_valid, pred2)
print('balanced accuracy:', balanced_accuracy)

logloss = log_loss(y_valid, pred2)
print("Log Loss:", logloss)
```

```
avg precision: 0.8232896045396045
balanced accuracy: 0.7216635239891054
```

```
Log Loss: 6.05561398575359
```

[ ]:

```
```

[ ]:

```
```