Suggested code may be subject to a license | 03Akshay/assignments-1

```
import numpy as np
import pandas as pd
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

```
raw_train=pd.read_csv('train.csv')
raw_test=pd.read_csv('test.csv')
raw_train.head()
```

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome |
|---|---|---|---|---|---|---|---|
| **0** | LP001002 | Male | No | 0 | Graduate | No | 5849 |
| **1** | LP001003 | Male | Yes | 1 | Graduate | No | 4583 |
| **2** | LP001005 | Male | Yes | 0 | Graduate | Yes | 3000 |
| **3** | LP001006 | Male | Yes | 0 | Not Graduate | No | 2583 |
| **4** | LP001008 | Male | No | 0 | Graduate | No | 6000 |

Next steps: Generate code with raw_train | View recommended plots | New interactive sheet

```
raw_test.tail()
```

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome |
|---|---|---|---|---|---|---|---|
| **357** | LP002971 | Male | Yes | 3+ | Not Graduate | Yes | 4009 |
| **358** | LP002975 | Male | Yes | 0 | Graduate | No | 4158 |
| **359** | LP002980 | Male | No | 0 | Graduate | No | 3250 |
| **360** | LP002986 | Male | Yes | 0 | Graduate | No | 5000 |
| **361** | LP002989 | Male | No | 0 | Graduate | Yes | 9200 |

```
raw_train.nunique() # shows me all the unique ids present
```

```
Loan_ID              614
Gender                 2
Married                2
Dependents             4
Education              2
Self_Employed          2
ApplicantIncome      505
CoapplicantIncome    287
LoanAmount           203
Loan_Amount_Term      10
Credit_History         2
Property_Area          3
Loan_Status            2
dtype: int64
```

```
raw_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
```

```
memory usage: 62.5+ KB
```

```
raw_train.shape
```

```
(614, 13)
```

```
# To copy all the data into a dataframe
train_df = raw_train.copy()
test_df = raw_test.copy()
```

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

```
test_df.info() # here you can see the loan_Status data is missing
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 362 entries, 0 to 361
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            362 non-null    object
 1   Gender             351 non-null    object
 2   Married            362 non-null    object
 3   Dependents         353 non-null    object
 4   Education          362 non-null    object
 5   Self_Employed      339 non-null    object
 6   ApplicantIncome    362 non-null    int64
 7   CoapplicantIncome  362 non-null    int64
 8   LoanAmount         362 non-null    int64
 9   Loan_Amount_Term   356 non-null    float64
 10  Credit_History     333 non-null    float64
 11  Property_Area      362 non-null    object
dtypes: float64(2), int64(3), object(7)
memory usage: 34.1+ KB
```

```
# From the baove data frame i see that there is data of loan_status for training
# the test data dosent have the loan_status so we can only use it for predicition
```

## Preprocessing

```
train_y = train_df['Loan_Status'].copy()
train_y
```

```
0      Y
1      N
2      Y
3      Y
4      Y
      ..
609    Y
610    Y
611    Y
612    Y
613    N
Name: Loan_Status, Length: 614, dtype: object
```

```
train_df.drop(columns=['Loan_Status'], inplace=True)
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
dtypes: float64(4), int64(1), object(7)
memory usage: 57.7+ KB
```

## Dropping unncessary columns

```
train_df.drop(columns='Loan_ID', inplace=True)
test_df.drop(columns='Loan_ID', inplace=True)
```

```
train_df.columns
```

```
Index(['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed',
       'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area'],
      dtype='object')
```

## Check for dupliates

```
train_df.duplicated()
```

```
0      False
1      False
2      False
3      False
4      False
       ...
609    False
610    False
611    False
612    False
613    False
Length: 614, dtype: bool
```

```
train_df[train_df.duplicated()]
```

No entries Filter

| index | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantInco |
|---|---|---|---|---|---|---|---|

Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables

```
test_df[test_df.duplicated()]
```

1 entry Filter

| index | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantInco |
|---|---|---|---|---|---|---|---|
| 192 | Male | No | 0 | Graduate | Yes | 5833 | |

Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables

```
test_df.drop_duplicates(inplace=True)
test_df
```

1 to 100 of 361 entries Filter

| index | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantInco |
|---|---|---|---|---|---|---|---|
| 0 | Male | Yes | 0 | Graduate | No | 5720 | |
| 1 | Male | Yes | 1 | Graduate | No | 3076 | 1 |
| 2 | Male | Yes | 2 | Graduate | No | 5000 | 1 |
| 3 | Male | Yes | 2 | Graduate | No | 2340 | 2 |
| 4 | Male | No | 0 | Not Graduate | No | 3276 | |
| 5 | Male | Yes | 0 | Not Graduate | Yes | 2165 | 3 |
| 6 | Female | No | 1 | Not Graduate | No | 2226 | |
| 7 | Male | Yes | 2 | Not Graduate | No | 3881 | |
| 8 | Male | Yes | 2 | Graduate | NaN | 13633 | |
| 9 | Male | No | 0 | Not Graduate | No | 2400 | 2 |
| 10 | Male | No | 0 | Not Graduate | No | 3091 | |
| 11 | Male | Yes | 1 | Graduate | NaN | 2185 | 1 |
| 12 | Male | No | 3+ | Graduate | No | 4166 | |
| 13 | Male | Yes | 2 | Graduate | NaN | 12173 | |
| 14 | Female | No | 0 | Graduate | No | 4666 | |
| 15 | Male | No | 1 | Graduate | No | 5667 | |
| 16 | Male | Yes | 2 | Graduate | No | 4583 | 2 |
| 17 | Male | Yes | 3+ | Graduate | No | 3786 | |
| 18 | Male | Yes | 0 | Graduate | No | 9226 | 7 |
| 19 | Male | No | 0 | Graduate | No | 1300 | 3 |
| 20 | Male | Yes | 1 | Not Graduate | No | 1888 | 1 |
| 21 | Female | No | 3+ | Not Graduate | No | 2083 | |
| 22 | NaN | No | 0 | Graduate | No | 3909 | |
| 23 | Female | No | 0 | Not Graduate | No | 3765 | |
| 24 | Male | Yes | 0 | Graduate | No | 5400 | 4 |
| 25 | Male | No | 0 | Graduate | No | 0 | 24 |
| 26 | Male | Yes | 2 | Graduate | No | 4363 | 1 |
| 27 | Male | Yes | 0 | Graduate | No | 7500 | 3 |
| 28 | Male | Yes | 0 | Graduate | No | 3772 | |
| 29 | Male | No | 0 | Graduate | No | 2942 | 2 |
| 30 | Female | No | 0 | Not Graduate | No | 2478 | |
| 31 | Male | Yes | 2 | Graduate | No | 6250 | |
| 32 | Male | No | 0 | Graduate | No | 3268 | 1 |
| 33 | Male | Yes | 0 | Graduate | No | 2783 | 2 |
| 34 | Male | Yes | 0 | Graduate | No | 2740 | 1 |
| 35 | Male | No | 0 | Graduate | No | 3150 | |
| 36 | Male | Yes | 2 | Graduate | NaN | 7350 | 4 |
| 37 | Male | Yes | 0 | Graduate | Yes | 2267 | 2 |
| 38 | Male | No | 0 | Graduate | Yes | 5833 | |
| 39 | Male | No | 0 | Graduate | No | 3643 | 1 |
| 40 | Male | Yes | 0 | Graduate | No | 5629 | |
| 41 | Female | No | 0 | Graduate | No | 3644 | |
| 42 | Male | Yes | 0 | Not Graduate | No | 1750 | 2 |
| 43 | Male | No | 0 | Graduate | No | 6500 | 2 |
| 44 | Female | No | 0 | Graduate | No | 3666 | |
| 45 | Male | Yes | 0 | Graduate | No | 4260 | 3 |
| 46 | Male | Yes | NaN | Not Graduate | No | 4163 | 1 |
| 47 | Male | No | 0 | Not Graduate | No | 2356 | 1 |
| 48 | Male | No | 0 | Graduate | No | 6792 | 3 |
| 49 | Male | Yes | 3+ | Not Graduate | Yes | 8000 | |
| 50 | Male | Yes | 1 | Graduate | No | 2419 | 1 |
| 51 | NaN | Yes | 3+ | Not Graduate | No | 3500 | |
| 52 | Male | Yes | 1 | Graduate | No | 3500 | 3 |
| 53 | Male | Yes | 2 | Graduate | No | 4116 | 1 |
| 54 | Male | Yes | 0 | Not Graduate | Yes | 5293 | |
| 55 | Male | No | 0 | Graduate | No | 2750 | |
| 56 | Female | No | 0 | Not Graduate | No | 4402 | |

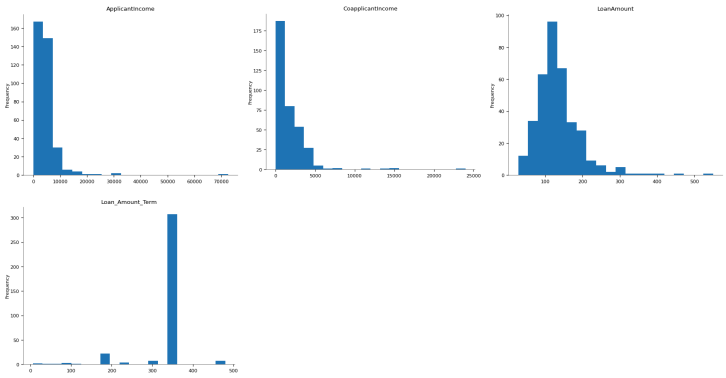
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 57 | Male | Yes | 2 | Graduate | No | 3613 | 3 |
| 58 | Female | Yes | 2 | Graduate | No | 2779 | 3 |
| 59 | Male | Yes | 3+ | Graduate | No | 4720 | |
| 60 | Male | Yes | 0 | Not Graduate | No | 2415 | 1 |
| 61 | Male | Yes | 0 | Graduate | Yes | 7016 | |
| 62 | Female | No | 2 | Graduate | No | 4968 | |
| 63 | Female | No | 0 | Graduate | No | 2101 | 1 |
| 64 | Male | Yes | 3+ | Not Graduate | No | 4490 | |
| 65 | Male | Yes | 0 | Graduate | No | 2917 | 3 |
| 66 | Male | Yes | 0 | Not Graduate | No | 4700 | |
| 67 | Male | Yes | 0 | Graduate | No | 3445 | |
| 68 | Male | Yes | 0 | Graduate | No | 7666 | |
| 69 | Male | Yes | 0 | Graduate | No | 2458 | 5 |
| 70 | Female | No | NaN | Graduate | No | 3250 | |
| 71 | Male | No | 0 | Graduate | No | 4463 | |
| 72 | Male | Yes | 1 | Graduate | NaN | 4083 | 1 |
| 73 | Male | Yes | 0 | Graduate | Yes | 3900 | 2 |
| 74 | Male | Yes | 0 | Not Graduate | No | 4750 | 3 |
| 75 | Male | No | 0 | Graduate | No | 3583 | 3 |
| 76 | Male | Yes | 0 | Graduate | No | 3189 | 2 |
| 77 | Male | No | 0 | Graduate | Yes | 6356 | |
| 78 | Female | Yes | 0 | Graduate | No | 7950 | |
| 79 | Male | Yes | 3+ | Graduate | No | 3829 | 1 |
| 80 | Male | Yes | 3+ | Graduate | No | 72529 | |
| 81 | Male | Yes | 2 | Not Graduate | No | 4136 | |
| 82 | Male | Yes | 0 | Graduate | No | 8449 | |
| 83 | Male | Yes | 0 | Graduate | No | 4456 | |
| 84 | Male | Yes | 2 | Graduate | No | 4635 | 8 |
| 85 | Male | Yes | 0 | Graduate | No | 3571 | 1 |
| 86 | Male | No | 0 | Graduate | No | 3066 | |
| 87 | Male | No | 2 | Not Graduate | No | 3235 | 2 |
| 88 | Female | No | 0 | Graduate | NaN | 5058 | |
| 89 | Male | Yes | 0 | Graduate | Yes | 3188 | 2 |
| 90 | Male | Yes | 3+ | Graduate | No | 13518 | |
| 91 | Male | Yes | 1 | Graduate | No | 4364 | 2 |
| 92 | Male | Yes | 2 | Not Graduate | No | 4766 | 1 |
| 93 | Male | Yes | 1 | Graduate | No | 4609 | 2 |
| 94 | Female | Yes | 3+ | Graduate | No | 6260 | |
| 95 | Male | Yes | 1 | Graduate | No | 3333 | 4 |
| 96 | Male | Yes | 0 | Graduate | No | 3500 | 3 |
| 97 | Male | Yes | 3+ | Graduate | No | 9719 | |
| 98 | Male | Yes | 3+ | Graduate | No | 6835 | |
| 99 | Male | No | 0 | Graduate | No | 4452 | |

Show 100 per page

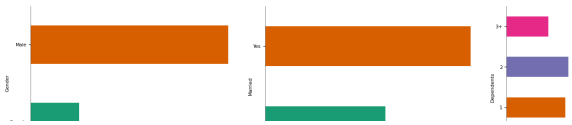
1 2 3 4

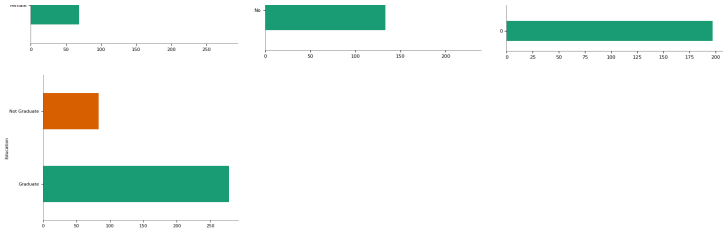
Like what you see? Visit the data table notebook to learn more about interactive tables.

**Distributions**




**Categorical distributions**






2-d distributions

Next

Generate code with test_df | View recommended plots | New interactive sheet

## Misising Value analysis

```
train_df.isna().sum()
```

```
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
dtype: int64
```

## When we find missing values we can either remove them or impute values like:

## for all numeric values---> mean

## for all Categorical values ---> mode

```
# to check which are numerical and whicha re categorical
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Gender             601 non-null    object
 1   Married            611 non-null    object
 2   Dependents         599 non-null    object
 3   Education          614 non-null    object
 4   Self_Employed      582 non-null    object
 5   ApplicantIncome    614 non-null    int64
 6   CoapplicantIncome  614 non-null    float64
 7   LoanAmount         592 non-null    float64
 8   Loan_Amount_Term   600 non-null    float64
 9   Credit_History     564 non-null    float64
 10  Property_Area      614 non-null    object
dtypes: float64(4), int64(1), object(6)
memory usage: 52.9+ KB
```

```
train_df.columns
```

```
Index(['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed',
       'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
```

```
       'Loan_Amount_Term', 'Credit_History', 'Property_Area'],
      dtype='object')
```

```
numeical_columns = ['ApplicantIncome','CoapplicantIncome','LoanAmount','Loan_Amount_Term'] # Removed extra space after 'LoanAmount'
cate_columns= ['Gender','Married','Dependents','Education','Self_Employed','Credit_History','Property_Area'] # Removed extra spaces :
```

```
cat_imputer = SimpleImputer(strategy="most_frequent")
cat_imputer.fit(train_df[cate_columns])

train_df[cate_columns] = cat_imputer.transform(train_df[cate_columns])
test_df[cate_columns] = cat_imputer.transform(test_df[cate_columns]) # Changed train_df to test_df to impute on the test data
```

```
num_imputer = SimpleImputer(strategy="mean")
num_imputer.fit(train_df[numeical_columns])

train_df[numeical_columns] = num_imputer.transform(train_df[numeical_columns])
test_df[numeical_columns] = num_imputer.transform(test_df[numeical_columns]) # Changed train_df to test_df to impute on the test data
```

```
train_df.isna().sum()
```

```
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
dtype: int64
```

## Preprocessing

```
train_df.head()
```

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | Coapplican |
|---|---|---|---|---|---|---|---|
| **0** | Male | No | 0 | Graduate | No | 5849.0 | |
| **1** | Male | Yes | 1 | Graduate | No | 4583.0 | |
| **2** | Male | Yes | 0 | Graduate | Yes | 3000.0 | |
| **3** | Male | Yes | 0 | Not Graduate | No | 2583.0 | |

Next steps: Generate code with train_df | View recommended plots | New interactive sheet

```
train_df['ApplicantIncome'] = train_df['ApplicantIncome'] + train_df['CoapplicantIncome']
test_df['ApplicantIncome'] = test_df['ApplicantIncome'] + test_df['CoapplicantIncome']

#drop the co-applicant income column
train_df.drop(columns='CoapplicantIncome', inplace=True)
test_df.drop(columns='CoapplicantIncome', inplace=True)

```

```
train_df.head()
```

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | LoanAmount |
|---|---|---|---|---|---|---|---|
| **0** | Male | No | 0 | Graduate | No | 5849.0 | 146.412162 |
| **1** | Male | Yes | 1 | Graduate | No | 9107.0 | 128.000000 |
| **2** | Male | Yes | 0 | Graduate | Yes | 3000.0 | 66.000000 |
| **3** | Male | Yes | 0 | Not Graduate | No | 9657.0 | 120.000000 |

Next steps: Generate code with train_df | View recommended plots | New interactive sheet

```
test_df.tail()
```

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | LoanAmou |
|---|---|---|---|---|---|---|---|
| **357** | Male | Yes | 3+ | Not Graduate | Yes | 5786.0 | 113 |
| **358** | Male | Yes | 0 | Graduate | No | 4867.0 | 115 |
| **359** | Male | No | 0 | Graduate | No | 5243.0 | 126 |
| **360** | Male | Yes | 0 | Graduate | No | 7393.0 | 158 |

```
# Application of label_encoder
train_df.nunique()
```

```
Gender                2
Married               2
Dependents            4
Education             2
Self_Employed         2
ApplicantIncome     583
LoanAmount          204
Loan_Amount_Term     11
Credit_History        2
Property_Area         3
dtype: int64
```

```
train_df.Dependents.unique()
```

```
array(['0', '1', '2', '3+'], dtype=object)
```

```
train_df.Property_Area.unique()
```

```
array(['Urban', 'Rural', 'Semiurban'], dtype=object)
```

```
for col in cate_columns:
  train_df[col] = LabelEncoder().fit_transform(train_df[col])
  test_df[col] = LabelEncoder().fit_transform(test_df[col])

```

```
train_df.head()
```

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | LoanAmount | Loan_Amount_Term | Credit_History | Property_Ar |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 0 | 5849.0 | 146.412162 | 360.0 | 1 | |
| **1** | 1 | 1 | 1 | 0 | 0 | 9107.0 | 128.000000 | 360.0 | 1 | |
| **2** | 1 | 1 | 0 | 0 | 1 | 3000.0 | 66.000000 | 360.0 | 1 | |
| **3** | 1 | 1 | 0 | 1 | 0 | 9657.0 | 120.000000 | 360.0 | 1 | |
| **4** | 1 | 0 | 0 | 0 | 0 | 6000.0 | 141.000000 | 360.0 | 1 | |

Next steps: Generate code with `train_df` | View recommended plots | New interactive sheet

```
numeical_columns.remove('CoapplicantIncome')
```

```
# log transformation
train_df[numeical_columns]= np.log(train_df[numeical_columns])
test_df[numeical_columns]= np.log(test_df[numeical_columns])
```

## Scaling

```
min_max_scaler = MinMaxScaler()
train_df = min_max_scaler.fit_transform(train_df)
test_df = min_max_scaler.transform(test_df)
```

## Building the Model

```
X_train, X_test, y_train, y_test = train_test_split(train_df, train_y, test_size=0.2, random_state=42)
```

```
log = LogisticRegression()
log.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
y_pred_test = log.predict(X_test) # predicted values
```

```
acc = accuracy_score(y_test, y_pred_test)
acc
```

```
0.7886178861788617
```