Лабортаторная работа №2 по курсо ТМО

Холодова Карина

ИУ5Ц-82Б

Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных

Loan_ID : Уникальный идентификатор заявки на кредит.

Gender : Пол заявителя (например, Male — мужчина, Female — женщина).

Married : Семейное положение (Yes — женат/замужем, No — холост/не замужем).

Dependents : Количество иждивенцев у заявителя (например, 0, 1, 2 и т.д.; NaN указывает на пропущенные данные).

Education : Образование заявителя (Graduate — имеет высшее образование, Not Graduate — не имеет высшего образования).

Self_Employed : Является ли заявитель предпринимателем (Yes — да, No — нет).

ApplicantIncome : Доход основного заявителя.

CoapplicantIncome : Доход созаявителя (если есть; 0.0 означает отсутствие созаявителя).

LoanAmount : Запрашиваемая сумма кредита.

Loan_Amount_Term : Срок кредита в месяцах (например, 360 месяцев = 30 лет).

Credit_History : История кредитования (1.0 — положительная кредитная история, 0.0 или NaN — отсутствие или негативная кредитная история).

Property_Area : Местоположение собственности, для которой запрашивается кредит (Rural — сельская местность, Urban — городская местность, Semiurban — пригород).

Loan_Status : Статус заявки на кредит (Y — одобрено, N — отклонено).

```
# Импорты библиотек: numpy, pandas, matplotlib, seaborn и os

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

import os
```

```
# Загрузка датасета из CSV файла

df = pd.read_csv("loan_data.csv")

# Вывод случайных 5 строк исходного датафрейма

df.sample(5)
```

```
      Loan_ID  Gender Married Dependents Education Self_Employed  \
378  LP002978  Female      No          0  Graduate            No
245  LP002250    Male     Yes          0  Graduate            No
272  LP002368    Male     Yes          2  Graduate            No
212  LP002100    Male      No        NaN  Graduate            No
199  LP002008    Male     Yes          2  Graduate           Yes

     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term
\
378             2900                0.0        71.0             360.0

245             5488                0.0       125.0             360.0

272             5935                0.0       133.0             360.0

212             2833                0.0        71.0             360.0

199             5746                0.0       144.0              84.0


     Credit_History Property_Area Loan_Status
378             1.0         Rural           Y
245             1.0         Rural           Y
272             1.0     Semiurban           Y
212             1.0         Urban           Y
199             NaN         Rural           Y
```

```
# Удаление столбца Loan_ID

df = df.drop(['Loan_ID'], axis=1)

# Вывод случайных трех строк после удаления столбца

df.sample(3)
```

```
    Gender Married Dependents     Education Self_Employed
ApplicantIncome  \
216   Male     Yes         3+  Not Graduate            No
3173
207   Male      No          0      Graduate            No
4917
234   Male     Yes          0      Graduate            No
4817
```

```
     CoapplicantIncome  LoanAmount  Loan_Amount_Term
Credit_History  \
216                0.0        74.0             360.0             1.0

207                0.0       130.0             360.0             0.0

234              923.0       120.0             180.0             1.0


    Property_Area Loan_Status
216     Semiurban           Y
207         Rural           Y
234         Urban           Y
```

```
# Проверка формы DataFrame, чтобы понять количество строк и столбцов

df.shape
```

```
(381, 12)
```

```
# Вывод информации о DataFrame с помощью df.info(), что показывает
типы данных и наличие пропусков

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 381 entries, 0 to 380
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Gender             376 non-null    object
 1   Married            381 non-null    object
 2   Dependents         373 non-null    object
 3   Education          381 non-null    object
 4   Self_Employed      360 non-null    object
 5   ApplicantIncome    381 non-null    int64
 6   CoapplicantIncome  381 non-null    float64
 7   LoanAmount         381 non-null    float64
 8   Loan_Amount_Term   370 non-null    float64
 9   Credit_History     351 non-null    float64
 10  Property_Area      381 non-null    object
 11  Loan_Status        381 non-null    object
dtypes: float64(4), int64(1), object(7)
memory usage: 35.8+ KB
```

```
# Подсчет пропущенных значений в каждом столбце

df.isnull().sum()
```

```
Gender                5
Married               0
Dependents            8
Education             0
Self_Employed        21
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term     11
Credit_History       30
Property_Area         0
Loan_Status           0
dtype: int64
```

```
# One-hot кодирование категориального признака 'Property_Area'

df_encoding = pd.get_dummies(df, columns=['Property_Area'],
prefix='Property_Area')

df_encoding.head(3)
```

```
  Gender Married Dependents     Education Self_Employed
ApplicantIncome  \
0   Male     Yes          1      Graduate            No
4583
1   Male     Yes          0      Graduate           Yes
3000
2   Male     Yes          0  Not Graduate            No
2583

   CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History  \
0             1508.0       128.0             360.0             1.0
1                0.0        66.0             360.0             1.0
2             2358.0       120.0             360.0             1.0

  Loan_Status  Property_Area_Rural  Property_Area_Semiurban  \
0           N                 True                    False
1           Y                False                    False
2           Y                False                    False

   Property_Area_Urban
0                False
1                 True
2                 True
```

```
# Список столбцов, которые требуют бинарного кодирования или
преобразования

bit_columns = [
    'Gender',
    'Married',
```

```
    'Education',
    'Self_Employed',
    'Loan_Status',
    'Property_Area_Rural',
    'Property_Area_Semiurban',
    'Property_Area_Urban'
]

# Проверка уникальных значений в выбранных столбцах

for col in bit_columns:
    print(f"{col}:\n{df_encoding[col].unique()}\n")
```

```
Gender:
['Male' 'Female' nan]

Married:
['Yes' 'No']

Education:
['Graduate' 'Not Graduate']

Self_Employed:
['No' 'Yes' nan]

Loan_Status:
['N' 'Y']

Property_Area_Rural:
[ True False]

Property_Area_Semiurban:
[False  True]

Property_Area_Urban:
[False  True]
```

```
# Преобразование булевых значений в целочисленные для столбцов
'Property_Area'

df_encoding['Property_Area_Urban'] =
df_encoding['Property_Area_Urban'].astype(int)
df_encoding['Property_Area_Semiurban'] =
df_encoding['Property_Area_Semiurban'].astype(int)
df_encoding['Property_Area_Rural'] =
df_encoding['Property_Area_Rural'].astype(int)

# Замена категориальных значений на числовые в столбцах 'Education',
'Married', 'Loan_Status'
```

```
df_encoding['Education'].replace(['Graduate', 'Not Graduate'], [1, 0],
inplace=True)
df_encoding['Married'].replace(['Yes', 'No'], [1, 0], inplace=True)
df_encoding['Loan_Status'].replace(['Y', 'N'], [1, 0], inplace=True)
```

```
/var/folders/8l/5pgwt05s0h5_ftplv2qxvwlm0000gn/T/
ipykernel_45901/3290626206.py:3: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df_encoding['Education'].replace(['Graduate', 'Not Graduate'], [1,
0], inplace=True)
/var/folders/8l/5pgwt05s0h5_ftplv2qxvwlm0000gn/T/ipykernel_45901/32906
26206.py:3: FutureWarning: Downcasting behavior in `replace` is
deprecated and will be removed in a future version. To retain the old
behavior, explicitly call `result.infer_objects(copy=False)`. To opt-
in to the future behavior, set
`pd.set_option('future.no_silent_downcasting', True)`
  df_encoding['Education'].replace(['Graduate', 'Not Graduate'], [1,
0], inplace=True)
/var/folders/8l/5pgwt05s0h5_ftplv2qxvwlm0000gn/T/ipykernel_45901/32906
26206.py:4: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df_encoding['Married'].replace(['Yes', 'No'], [1, 0], inplace=True)
/var/folders/8l/5pgwt05s0h5_ftplv2qxvwlm0000gn/T/ipykernel_45901/32906
26206.py:4: FutureWarning: Downcasting behavior in `replace` is
deprecated and will be removed in a future version. To retain the old
behavior, explicitly call `result.infer_objects(copy=False)`. To opt-
in to the future behavior, set
`pd.set_option('future.no_silent_downcasting', True)`
```

```
  df_encoding['Married'].replace(['Yes', 'No'], [1, 0], inplace=True)
/var/folders/8l/5pgwt05s0h5_ftplv2qxvwlm0000gn/T/ipykernel_45901/32906
26206.py:5: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df_encoding['Loan_Status'].replace(['Y', 'N'], [1, 0], inplace=True)
/var/folders/8l/5pgwt05s0h5_ftplv2qxvwlm0000gn/T/ipykernel_45901/32906
26206.py:5: FutureWarning: Downcasting behavior in `replace` is
deprecated and will be removed in a future version. To retain the old
behavior, explicitly call `result.infer_objects(copy=False)`. To opt-
in to the future behavior, set
`pd.set_option('future.no_silent_downcasting', True)`
  df_encoding['Loan_Status'].replace(['Y', 'N'], [1, 0], inplace=True)

df_encoding.sample(4)

     Gender  Married Dependents  Education Self_Employed
ApplicantIncome  \
138    Male        0           0          1            No
2971
226    Male        1           0          1            No
4750
239    Male        1           0          1           NaN
3333
250  Female        0           0          1            No
3180

     CoapplicantIncome  LoanAmount  Loan_Amount_Term
Credit_History  \
138             2791.0       144.0             360.0              1.0

226             2333.0       130.0             360.0              1.0

239             2500.0       128.0             360.0              1.0

250                0.0        71.0             360.0              0.0


     Loan_Status  Property_Area_Rural  Property_Area_Semiurban  \
138            1                    0                        1
226            1                    0                        0
```

```
239             1                       0                           1
250             0                       0                           0

     Property_Area_Urban
138                    0
226                    1
239                    0
250                    1
```

```
# Заполнение пропусков в 'Gender' и 'Self_Employed' значением
'NotGiven'

df_encoding['Gender'].fillna('NotGiven', inplace=True)
df_encoding['Self_Employed'].fillna('NotGiven', inplace=True)
```

```
/var/folders/8l/5pgwt05s0h5_ftplv2qxvwlm0000gn/T/
ipykernel_45901/3757713171.py:3: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df_encoding['Gender'].fillna('NotGiven', inplace=True)
/var/folders/8l/5pgwt05s0h5_ftplv2qxvwlm0000gn/T/ipykernel_45901/37577
13171.py:4: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df_encoding['Self_Employed'].fillna('NotGiven', inplace=True)
```

```
for col in bit_columns:
    print(f"{col}:\n{df_encoding[col].unique()}\n")
```

```
Gender:
['Male' 'Female' 'NotGiven']
```

```
Married:
[1 0]

Education:
[1 0]

Self_Employed:
['No' 'Yes' 'NotGiven']

Loan_Status:
[0 1]

Property_Area_Rural:
[1 0]

Property_Area_Semiurban:
[0 1]

Property_Area_Urban:
[0 1]
```

```
# One-hot кодирование для 'Gender' и 'Self_Employed'

df_encoding = pd.get_dummies(df_encoding, columns=['Gender',
'Self_Employed'], prefix=['Gender', 'Self_Employed'], dtype=int)

df_encoding.sample(5)
```

```
     Married Dependents  Education  ApplicantIncome  CoapplicantIncome
\
209        0           0          1             2500                0.0

130        1           0          1             3173             3021.0

296        0           0          1             4166                0.0

348        1           0          1             2785             2016.0

134        1           2          1             5000                0.0


     LoanAmount  Loan_Amount_Term  Credit_History  Loan_Status  \
209        67.0             360.0             1.0            1
130       137.0             360.0             1.0            1
296        98.0             360.0             0.0            0
348       110.0             360.0             1.0            1
134        72.0             360.0             0.0            0

     Property_Area_Rural  Property_Area_Semiurban  Property_Area_Urban
```

```
\
209                     0                           0                     1

130                     0                           0                     1

296                     0                           1                     0

348                     1                           0                     0

134                     0                           1                     0


     Gender_Female  Gender_Male  Gender_NotGiven  Self_Employed_No  \
209              1            0                0                 1
130              0            1                0                 1
296              0            1                0                 1
348              0            1                0                 1
134              0            1                0                 0

     Self_Employed_NotGiven  Self_Employed_Yes
209                       0                  0
130                       0                  0
296                       0                  0
348                       0                  0
134                       1                  0
```

```
# Заполнение пропусков в 'Loan_Amount_Term' нулями (временное решение)

df_encoding['Loan_Amount_Term'].fillna(0, inplace=True)
```

```
/var/folders/8l/5pgwt05s0h5_ftplv2qxvwlm0000gn/T/
ipykernel_45901/2617159699.py:3: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df_encoding['Loan_Amount_Term'].fillna(0, inplace=True)
```

```
# Список числовых признаков для масштабирования

feature_for_scaling = ['ApplicantIncome', 'CoapplicantIncome',
'LoanAmount', 'Loan_Amount_Term']
```

```python
# Проверка наличия пропусков в выбранных признаках

for col in feature_for_scaling:
    print(f"{col}:\n{df_encoding[df_encoding[col].isnull()].shape[0]}\
n")
```

```
ApplicantIncome:
0

CoapplicantIncome:
0

LoanAmount:
0

Loan_Amount_Term:
0
```

```
pip install scikit-learn
```

```
Defaulting to user installation because normal site-packages is not
writeable
Collecting scikit-learn
  Downloading scikit_learn-1.6.1-cp39-cp39-
macosx_12_0_arm64.whl.metadata (31 kB)
Requirement already satisfied: numpy>=1.19.5 in
/Users/kkholodova/Library/Python/3.9/lib/python/site-packages (from
scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in
/Users/kkholodova/Library/Python/3.9/lib/python/site-packages (from
scikit-learn) (1.13.1)
Collecting joblib>=1.2.0 (from scikit-learn)
  Downloading joblib-1.4.2-py3-none-any.whl.metadata (5.4 kB)
Collecting threadpoolctl>=3.1.0 (from scikit-learn)
  Downloading threadpoolctl-3.6.0-py3-none-any.whl.metadata (13 kB)
Downloading scikit_learn-1.6.1-cp39-cp39-macosx_12_0_arm64.whl (11.1
MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.1/11.1 MB 1.0 MB/s eta
0:00:0000:0100:010m
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 301.8/301.8 kB 1.5 MB/s eta
0:00:00a 0:00:01
[notice] A new release of pip is available: 24.0 -> 25.0.1
[notice] To update, run:
/Library/Developer/CommandLineTools/usr/bin/python3 -m pip install --
upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

```python
# Масштабирование данных с использованием StandardScaler

from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()

df_encoding[feature_for_scaling] =
scaler.fit_transform(df_encoding[feature_for_scaling])

df_encoding.sample(5)
```

```
     Married Dependents  Education  ApplicantIncome  CoapplicantIncome
\
137        1           2          1         0.302659           0.188098

242        0           1          1         0.061465          -0.546371

101        1           0          1        -0.804574           0.879363

223        1           1          1         0.208156          -0.546371

345        0           0          0        -0.156455          -0.546371


     LoanAmount  Loan_Amount_Term  Credit_History  Loan_Status  \
137    0.388865          0.327910             1.0            1
242    0.282937         -1.709055             1.0            1
101    0.847884          0.327910             1.0            1
223   -1.341287          0.327910             1.0            0
345   -0.882267         -3.338627             1.0            0

     Property_Area_Rural  Property_Area_Semiurban  Property_Area_Urban
\
137                    0                        1                    0

242                    0                        0                    1

101                    1                        0                    0

223                    0                        0                    1

345                    0                        1                    0


     Gender_Female  Gender_Male  Gender_NotGiven  Self_Employed_No  \
137              0            1                0                 1
242              0            1                0                 0
101              0            1                0                 1
223              0            1                0                 1
345              0            1                0                 1

     Self_Employed_NotGiven  Self_Employed_Yes
137                       0                  0
242                       1                  0
101                       0                  0
```

```
223                         0                   0
345                         0                   0
```

```
# Обработка пропусков в 'Dependents' и one-hot кодирование
```

```
df_encoding.Dependents.fillna('NotGiven', inplace=True)
```

```
/var/folders/8l/5pgwt05s0h5_ftplv2qxvwlm0000gn/T/
ipykernel_45901/3878776685.py:3: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df_encoding.Dependents.fillna('NotGiven', inplace=True)
```

```
df_encoding.Dependents.isnull().sum()
```

```
0
```

```
df_encoding = pd.get_dummies(df_encoding, columns=['Dependents'],
prefix=['Dependents'], dtype=int)
```

```
df_encoding.Credit_History.isnull().sum()
```

```
30
```

```
df_encoding.Credit_History.unique()
```

```
array([ 1., nan,  0.])
```

```
# Обработка пропусков в 'Credit_History' и преобразование в dummy-
переменные
```

```
df_encoding.Credit_History.fillna('NotGiven', inplace=True)
```

```
/var/folders/8l/5pgwt05s0h5_ftplv2qxvwlm0000gn/T/
ipykernel_45901/2425284162.py:3: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
```

```
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df_encoding.Credit_History.fillna('NotGiven', inplace=True)
/var/folders/8l/5pgwt05s0h5_ftplv2qxvwlm0000gn/T/ipykernel_45901/24252
84162.py:3: FutureWarning: Setting an item of incompatible dtype is
deprecated and will raise an error in a future version of pandas.
Value 'NotGiven' has dtype incompatible with float64, please
explicitly cast to a compatible dtype first.
  df_encoding.Credit_History.fillna('NotGiven', inplace=True)
```

```
df_encoding.Credit_History.unique()
```

```
array([1.0, 'NotGiven', 0.0], dtype=object)
```

```
df_encoding = pd.get_dummies(df_encoding, columns=['Credit_History'],
prefix=['Credit_History'], dtype=int)

# Итоговая проверка структуры данных после всех преобразований

df_encoding.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 381 entries, 0 to 380
Data columns (total 24 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Married                  381 non-null    int64
 1   Education                381 non-null    int64
 2   ApplicantIncome          381 non-null    float64
 3   CoapplicantIncome        381 non-null    float64
 4   LoanAmount               381 non-null    float64
 5   Loan_Amount_Term         381 non-null    float64
 6   Loan_Status              381 non-null    int64
 7   Property_Area_Rural      381 non-null    int64
 8   Property_Area_Semiurban  381 non-null    int64
 9   Property_Area_Urban      381 non-null    int64
 10  Gender_Female            381 non-null    int64
 11  Gender_Male              381 non-null    int64
 12  Gender_NotGiven          381 non-null    int64
 13  Self_Employed_No         381 non-null    int64
 14  Self_Employed_NotGiven   381 non-null    int64
 15  Self_Employed_Yes        381 non-null    int64
 16  Dependents_0             381 non-null    int64
 17  Dependents_1             381 non-null    int64
 18  Dependents_2             381 non-null    int64
 19  Dependents_3+            381 non-null    int64
 20  Dependents_NotGiven      381 non-null    int64
 21  Credit_History_0.0       381 non-null    int64
```

```
 22  Credit_History_1.0       381 non-null    int64
 23  Credit_History_NotGiven  381 non-null    int64
dtypes: float64(4), int64(20)
memory usage: 71.6 KB
```

```
df_encoding.sample(4)
```

```
     Married  Education  ApplicantIncome  CoapplicantIncome
LoanAmount  \
157        0          1        -0.107793           -0.546371
1.165667
158        0          0        -0.703019            0.380591   -
0.034846
96         0          1        -0.423037            0.344659
0.530102
333        0          0        -0.635315            0.010576   -
0.387938

     Loan_Amount_Term  Loan_Status  Property_Area_Rural  \
157           0.32791            0                    0
158           0.32791            1                    1
96            0.32791            1                    1
333           0.32791            1                    0

     Property_Area_Semiurban  Property_Area_Urban  ...  \
157                        0                    1  ...
158                        0                    0  ...
96                         0                    0  ...
333                        1                    0  ...

     Self_Employed_NotGiven  Self_Employed_Yes  Dependents_0
Dependents_1  \
157                       0                  0             0
0
158                       0                  1             1
0
96                        1                  0             1
0
333                       0                  0             0
1

     Dependents_2  Dependents_3+  Dependents_NotGiven
Credit_History_0.0  \
157             1              0                    0
0
158             0              0                    0
0
96              0              0                    0
0
333             0              0                    0
```

```
     Credit_History_1.0  Credit_History_NotGiven
157                   1                        0
158                   1                        0
96                    1                        0
333                   1                        0

[4 rows x 24 columns]
```

```
df_encoding.columns
```

```
Index(['Married', 'Education', 'ApplicantIncome', 'CoapplicantIncome',
       'LoanAmount', 'Loan_Amount_Term', 'Loan_Status',
'Property_Area_Rural',
       'Property_Area_Semiurban', 'Property_Area_Urban',
'Gender_Female',
       'Gender_Male', 'Gender_NotGiven', 'Self_Employed_No',
       'Self_Employed_NotGiven', 'Self_Employed_Yes', 'Dependents_0',
       'Dependents_1', 'Dependents_2', 'Dependents_3+',
'Dependents_NotGiven',
       'Credit_History_0.0', 'Credit_History_1.0',
'Credit_History_NotGiven'],
      dtype='object')
```