 numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import osimport

for dirname, \_, filenames in os.walk('/kaggle/input'):

    for filename in filenames:

        print(os.path.join(dirname, filename))

import seaborn as sns

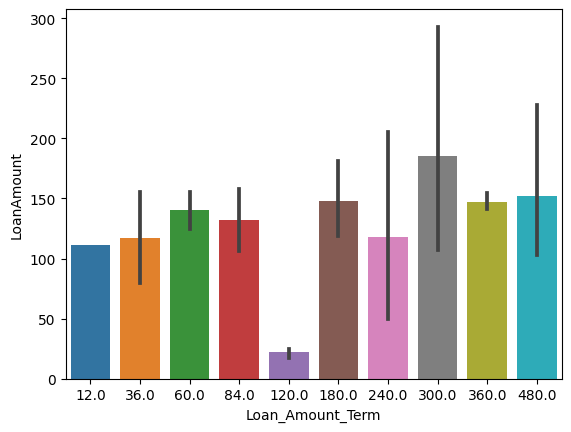
import matplotlib.pyplot as plt

import warnings

warnings.filterwarnings('ignore')

df = pd.read\_csv(r"/content/train\_u6lujuX\_CVtuZ9i[1].csv")

|  | **Loan\_ID** | **Gender** | **Married** | **Dependents** | **Education** | **Self\_Employed** | **ApplicantIncome** | **CoapplicantIncome** | **LoanAmount** | **Loan\_Amount\_Term** | **Credit\_History** | **Property\_Area** | **Loan\_Status** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | LP001002 | Male | No | 0 | Graduate | No | 5849 | 0.0 | NaN | 360.0 | 1.0 | Urban | Y |
| **1** | LP001003 | Male | Yes | 1 | Graduate | No | 4583 | 1508.0 | 128.0 | 360.0 | 1.0 | Rural | N |
| **2** | LP001005 | Male | Yes | 0 | Graduate | Yes | 3000 | 0.0 | 66.0 | 360.0 | 1.0 | Urban | Y |
| **3** | LP001006 | Male | Yes | 0 | Not Graduate | No | 2583 | 2358.0 | 120.0 | 360.0 | 1.0 | Urban | Y |
| **4** | LP001008 | Male | No | 0 | Graduate | No | 6000 | 0.0 | 141.0 | 360.0 | 1.0 | Urban | Y |
| **5** | LP001011 | Male | Yes | 2 | Graduate | Yes | 5417 | 4196.0 | 267.0 | 360.0 | 1.0 | Urban | Y |
| **6** | LP001013 | Male | Yes | 0 | Not Graduate | No | 2333 | 1516.0 | 95.0 | 360.0 | 1.0 | Urban | Y |
| **7** | LP001014 | Male | Yes | 3+ | Graduate | No | 3036 | 2504.0 | 158.0 | 360.0 | 0.0 | Semiurban | N |
| **8** | LP001018 | Male | Yes | 2 | Graduate | No | 4006 | 1526.0 | 168.0 | 360.0 | 1.0 | Urban | Y |
| **9** | LP001020 | Male | Yes | 1 | Graduate | No | 12841 | 10968.0 | 349.0 | 360.0 | 1.0 | Semiurban | N |



Loan\_Amount\_Term LoanAmount

19 NaN 115.0

36 NaN 100.0

44 NaN 96.0

45 NaN 88.0

73 NaN 95.0

112 NaN 152.0

165 NaN 182.0

197 NaN 120.0

223 NaN 175.0

232 NaN 120.0

335 NaN 70.0

367 NaN 124.0

421 NaN 80.0

423 NaN 110.0

df['Dependents'].value\_counts()

0 345

1 102

2 101

3+ 51

Name: Dependents, dtype: int64

df['Dependents'].replace('3+',3,inplace = True)

df['Dependents'].value\_counts()

0 345

1 102

2 101

3 51

Name: Dependents, dtype: int64

df[['Dependents', 'Married']][df['Dependents'].isnull()]

Dependents Married

102 NaN Yes

104 NaN NaN

120 NaN Yes

226 NaN Yes

228 NaN NaN

293 NaN No

301 NaN Yes

332 NaN No

335 NaN Yes

346 NaN Yes

355 NaN No

435 NaN NaN

517 NaN Yes

571 NaN Yes

597 NaN No

df['Gender'].fillna('Male', inplace = True)

df['Married'].fillna('Yes', inplace = True)

df['Self\_Employed'].fillna('No', inplace = True)

df['Credit\_History'].fillna('1.0', inplace = True)

df['LoanAmount'].fillna((df['LoanAmount'].mean()), inplace = True)

df['Loan\_Amount\_Term'].fillna('84', inplace = True)

df['Dependents'].fillna(0, inplace = True)

df['Dependents'] = df['Dependents'].astype('int')

df['Dependents'].dtype

dtype('int64')

df.isnull().sum()

Loan\_ID 0

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

Loan\_Status 0

dtype: int64

[31]

df.drop('Loan\_ID', axis = 1, inplace = True)

df.nunique()

df.nunique()

df.describe()

Dependents ApplicantIncome CoapplicantIncome LoanAmount

count 614.000000 614.000000 614.000000 614.000000

mean 0.744300 5403.459283 1621.245798 146.412162

std 1.009623 6109.041673 2926.248369 84.037468

min 0.000000 150.000000 0.000000 9.000000

25% 0.000000 2877.500000 0.000000 100.250000

50% 0.000000 3812.500000 1188.500000 129.000000

75% 1.000000 5795.000000 2297.250000 164.750000

max 3.000000 81000.000000 41667.000000 700.000000

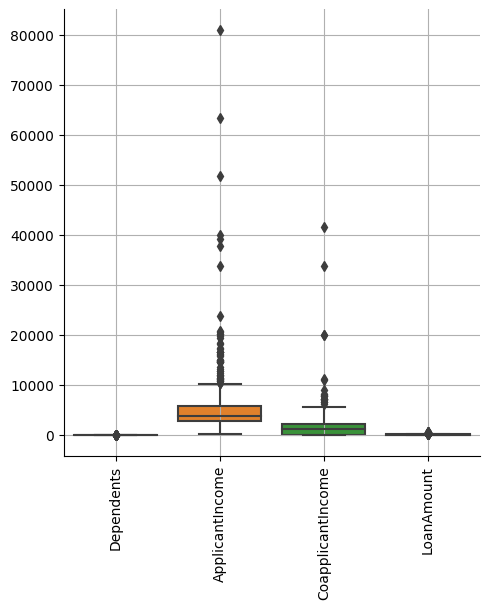
plt.figure(figsize = (10,4))

sns.catplot(data = df, kind = 'box')

plt.xticks(rotation = 90)

plt.grid()

plt.show()



fig, axs = plt.subplots(figsize = (20,3), ncols = 5)

sns.countplot(x = df['ApplicantIncome'], hue = df['Loan\_Status'], fill = True ,ax = axs[0])

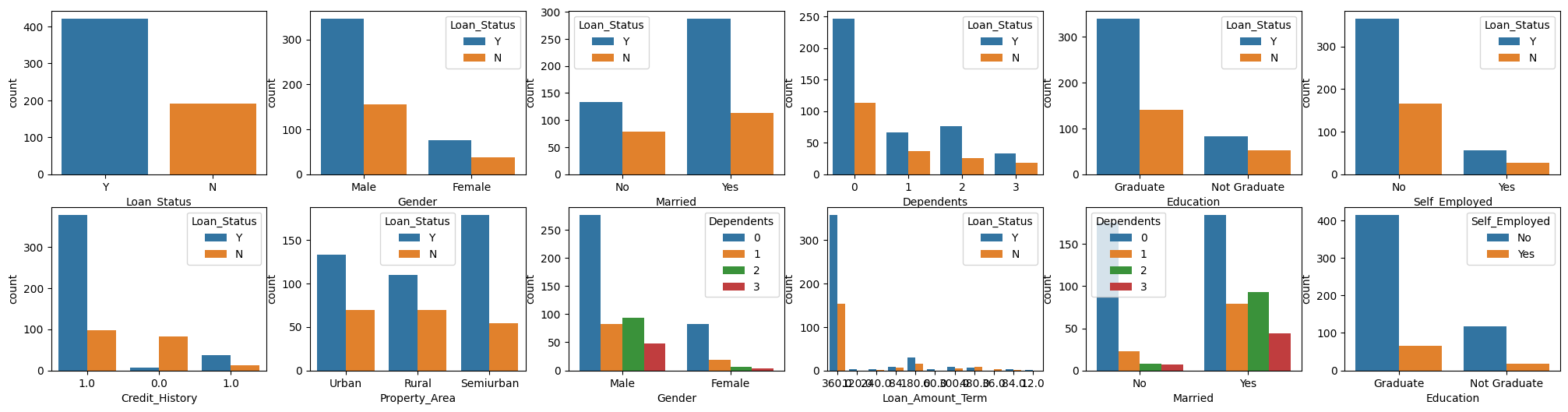
sns.countplot(x = df['CoapplicantIncome'], hue = df['Loan\_Status'], fill = True ,ax = axs[1])

sns.countplot(x = df['LoanAmount'], hue = df['Loan\_Status'], fill = True ,ax = axs[2])

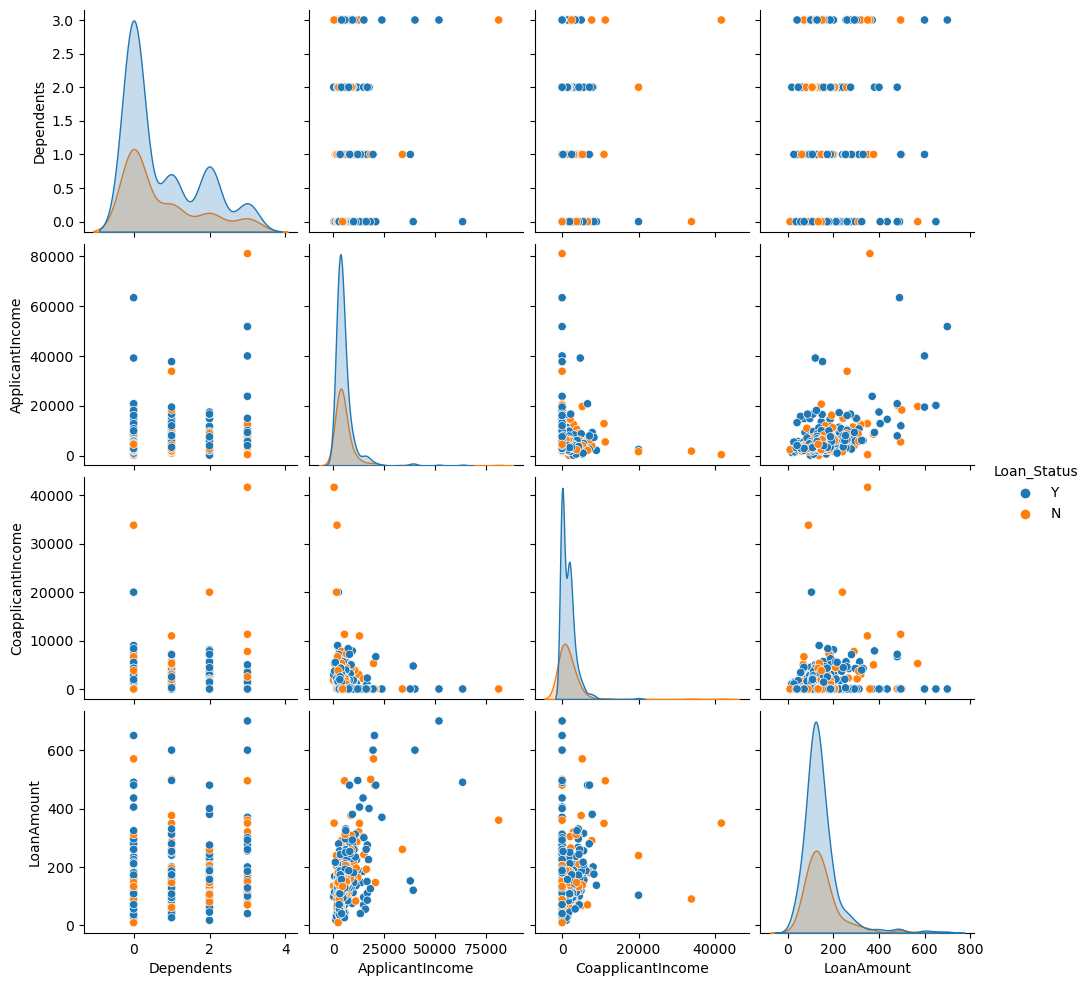
sns.countplot(x = df['Loan\_Amount\_Term'], hue = df['Loan\_Status'], fill = True ,ax = axs[3])

sns.countplot(x = df['ApplicantIncome'], hue = df['Gender'], fill = True ,ax = axs[4])

plt.show()



sns.pairplot(df, hue = 'Loan\_Status')



obj\_col = df.select\_dtypes('object').columns

obj\_col

Index(['Gender', 'Married', 'Education', 'Self\_Employed', 'Loan\_Amount\_Term', 'Credit\_History', 'Property\_Area', 'Loan\_Status'], dtype='object')

from sklearn.preprocessing import OrdinalEncoder

oe = OrdinalEncoder()

df[obj\_col] = df[obj\_col].astype(str)

df[obj\_col] = oe.fit\_transform(df[obj\_col])

df.head(3)

Gender Married Dependents Education Self\_Employed ApplicantIncome CoapplicantIncome LoanAmount Loan\_Amount\_Term Credit\_History Property\_Area Loan\_Status

0 1.0 0.0 0 0.0 0.0 5849 0.0 146.412162 6.0 1.0 2.0 1.0

1 1.0 1.0 1 0.0 0.0 4583 1508.0 128.000000 6.0 1.0 0.0 0.0

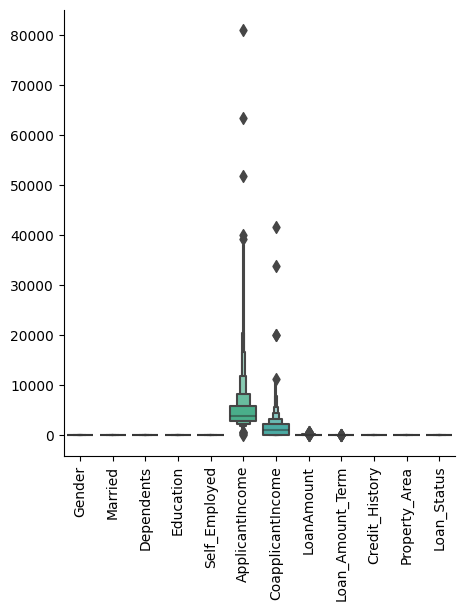
2 1.0 1.0 0 0.0 1.0 3000 0.0 66.000000 6.0 1.0 2.0 1.0

data = df

sns.catplot(data = df, kind = 'boxen')

plt.xticks(rotation = 90)

plt.show()



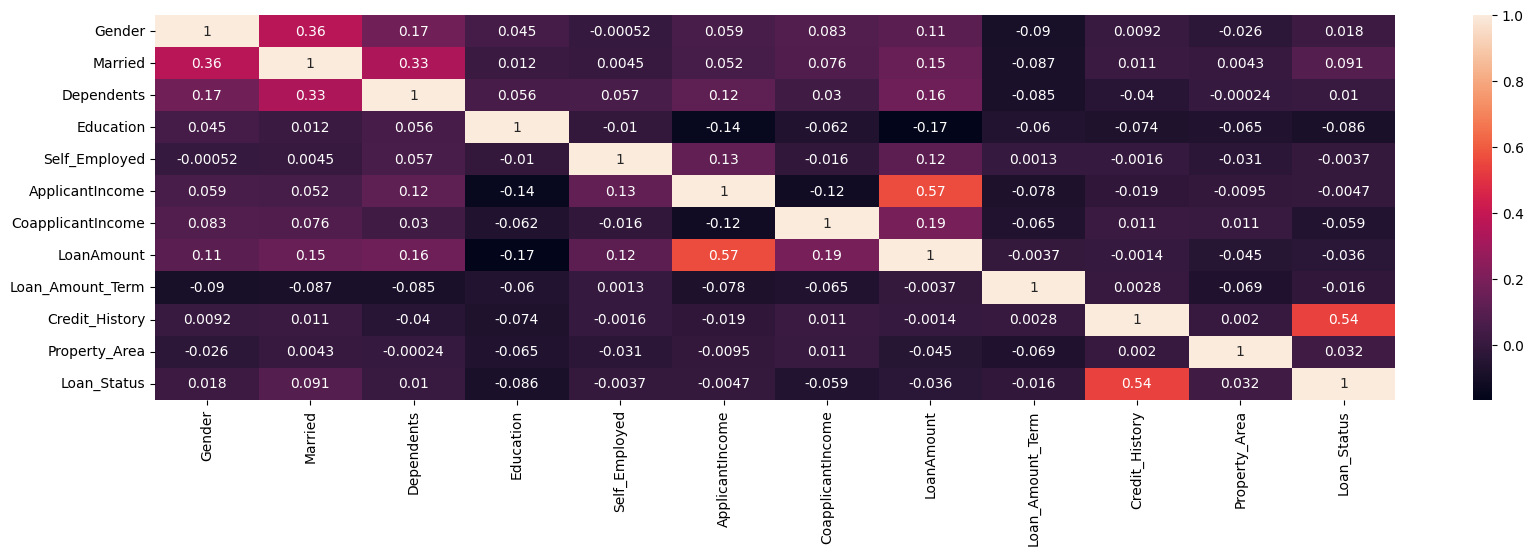
df.describe()

| **Gender** | **Married** | **Dependents** | **Education** | **Self\_Employed** | **ApplicantIncome** | **CoapplicantIncome** | **LoanAmount** | **Loan\_Amount\_Term** | **Credit\_History** | **Property\_Area** | **Loan\_Status** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 614.000000 | 614.000000 | 614.000000 | 614.000000 | 614.000000 | 614.000000 | 614.000000 | 614.000000 | 614.000000 | 614.000000 | 614.000000 | 614.000000 |
| **mean** | 0.817590 | 0.653094 | 0.744300 | 0.218241 | 0.133550 | 5403.459283 | 1621.245798 | 146.412162 | 5.739414 | 0.855049 | 1.037459 | 0.687296 |
| **std** | 0.386497 | 0.476373 | 1.009623 | 0.413389 | 0.340446 | 6109.041673 | 2926.248369 | 84.037468 | 1.325067 | 0.352339 | 0.787482 | 0.463973 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 150.000000 | 0.000000 | 9.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 2877.500000 | 0.000000 | 100.250000 | 6.000000 | 1.000000 | 0.000000 | 0.000000 |
| **50%** | 1.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 3812.500000 | 1188.500000 | 129.000000 | 6.000000 | 1.000000 | 1.000000 | 1.000000 |
| **75%** | 1.000000 | 1.000000 | 1.000000 | 0.000000 | 0.000000 | 5795.000000 | 2297.250000 | 164.750000 | 6.000000 | 1.000000 | 2.000000 | 1.000000 |
| **max** | 1.000000 | 1.000000 | 3.000000 | 1.000000 | 1.000000 | 81000.000000 | 41667.000000 | 700.000000 | 10.000000 | 1.000000 | 2.000000 | 1.000000 |

**plt.figure(figsize = (20,5))**

**sns.heatmap(df.corr(), annot = True)**

**plt.show()**

****

from sklearn.preprocessing import StandardScaler

ss = StandardScaler()

df.iloc[:,:-1] = ss.fit\_transform(df.iloc[:,:-1])

df.head()

**Gender Married Dependents Education Self\_Employed ApplicantIncome CoapplicantIncome LoanAmount Loan\_Amount\_Term Credit\_History Property\_Area Loan\_Status**

**0 0.472343 -1.372089 -0.737806 -0.528362 -0.392601 0.072991 -0.554487 0.000000 0.196819 0.411733 1.223298 1.0**

**1 0.472343 0.728816 0.253470 -0.528362 -0.392601 -0.134412 -0.038732 -0.219273 0.196819 0.411733 -1.318513 0.0**

**2 0.472343 0.728816 -0.737806 -0.528362 2.547117 -0.393747 -0.554487 -0.957641 0.196819 0.411733 1.223298 1.0**

**3 0.472343 0.728816 -0.737806 1.892641 -0.392601 -0.462062 0.251980 -0.314547 0.196819 0.411733 1.223298 1.0**

**4 0.472343 -1.372089 -0.737806 -0.528362 -0.392601 0.097728 -0.554487 -0.064454 0.196819 0.411733 1.223298 1.0**

x = df.iloc[:,:-1]

y = df.iloc[:,-1]

x.head()

|  | **Gender** | **Married** | **Dependents** | **Education** | **Self\_Employed** | **ApplicantIncome** | **CoapplicantIncome** | **LoanAmount** | **Loan\_Amount\_Term** | **Credit\_History** | **Property\_Area** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.472343 | -1.372089 | -0.737806 | -0.528362 | -0.392601 | 0.072991 | -0.554487 | 0.000000 | 0.196819 | 0.411733 | 1.223298 |
| **1** | 0.472343 | 0.728816 | 0.253470 | -0.528362 | -0.392601 | -0.134412 | -0.038732 | -0.219273 | 0.196819 | 0.411733 | -1.318513 |
| **2** | 0.472343 | 0.728816 | -0.737806 | -0.528362 | 2.547117 | -0.393747 | -0.554487 | -0.957641 | 0.196819 | 0.411733 | 1.223298 |
| **3** | 0.472343 | 0.728816 | -0.737806 | 1.892641 | -0.392601 | -0.462062 | 0.251980 | -0.314547 | 0.196819 | 0.411733 | 1.223298 |
| **4** | 0.472343 | -1.372089 | -0.737806 | -0.528362 | -0.392601 | 0.097728 | -0.554487 | -0.064454 | 0.196819 | 0.411733 | 1.223298 |

from sklearn.model\_selection import train\_test\_split

xtrain, xtest, ytrain, ytest = train\_test\_split(x,y,random\_state = 4, test\_size = 0.25, stratify = y)

def mymodel(model):

    model.fit(xtrain,ytrain)

    ypred = model.predict(xtest)

    train\_accuracy = model.score(xtrain,ytrain)

    test\_accuracy = model.score(xtest, ytest)

    print(str(model)[:-2], 'Accuracy')

    print('Accuracy: ', accuracy\_score(ytest,ypred), "\nClassification Report: \n", classification\_report(ytest, ypred), '\nConfusion Matrix: \n', confusion\_matrix(ytest, ypred))

    print(f'Training Accuracy: {train\_accuracy}\nTesting Accuracy: {test\_accuracy}')

    print()

    print()

    return model

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.naive\_bayes import GaussianNB

from sklearn.ensemble import RandomForestClassifier

knn = mymodel(KNeighborsClassifier())

svc = mymodel(SVC())

dt= mymodel(DecisionTreeClassifier())

lr = mymodel(LogisticRegression())

gnb = mymodel(GaussianNB())

rfc = mymodel(RandomForestClassifier(n\_estimators = 80, max\_depth = 10, min\_samples\_leaf = 12))

KNeighborsClassifier Accuracy

Accuracy: 0.8246753246753247

Classification Report:

precision recall f1-score support

0.0 0.86 0.52 0.65 48

1.0 0.82 0.96 0.88 106

accuracy 0.82 154

macro avg 0.84 0.74 0.77 154

weighted avg 0.83 0.82 0.81 154

Confusion Matrix:

[[ 25 23]

[ 4 102]]

Training Accuracy: 0.8217391304347826

Testing Accuracy: 0.8246753246753247

SVC Accuracy

Accuracy: 0.8246753246753247

Classification Report:

precision recall f1-score support

0.0 1.00 0.44 0.61 48

1.0 0.80 1.00 0.89 106

accuracy 0.82 154

macro avg 0.90 0.72 0.75 154

weighted avg 0.86 0.82 0.80 154

Confusion Matrix:

[[ 21 27]

[ 0 106]]

Training Accuracy: 0.8108695652173913

Testing Accuracy: 0.8246753246753247

DecisionTreeClassifier Accuracy

Accuracy: 0.7597402597402597

Classification Report:

precision recall f1-score support

0.0 0.63 0.56 0.59 48

1.0 0.81 0.85 0.83 106

accuracy 0.76 154

macro avg 0.72 0.71 0.71 154

weighted avg 0.75 0.76 0.76 154

Confusion Matrix:

[[27 21]

[16 90]]

Training Accuracy: 1.0

Testing Accuracy: 0.7597402597402597

LogisticRegression Accuracy

Accuracy: 0.8376623376623377

Classification Report:

precision recall f1-score support

0.0 1.00 0.48 0.65 48

1.0 0.81 1.00 0.89 106

accuracy 0.84 154

macro avg 0.90 0.74 0.77 154

weighted avg 0.87 0.84 0.82 154

Confusion Matrix:

[[ 23 25]

[ 0 106]]

Training Accuracy: 0.8043478260869565

Testing Accuracy: 0.8376623376623377

GaussianNB Accuracy

Accuracy: 0.8311688311688312

Classification Report:

precision recall f1-score support

0.0 0.96 0.48 0.64 48

1.0 0.81 0.99 0.89 106

accuracy 0.83 154

macro avg 0.88 0.73 0.76 154

weighted avg 0.85 0.83 0.81 154

Confusion Matrix:

[[ 23 25]

[ 1 105]]

Training Accuracy: 0.8021739130434783

Testing Accuracy: 0.8311688311688312

RandomForestClassifier(max\_depth=10, min\_samples\_leaf=12, n\_estimators=8 Accuracy

Accuracy: 0.8376623376623377

Classification Report:

precision recall f1-score support

0.0 1.00 0.48 0.65 48

1.0 0.81 1.00 0.89 106

accuracy 0.84 154

macro avg 0.90 0.74 0.77 154

weighted avg 0.87 0.84 0.82 154

Confusion Matrix:

[[ 23 25]

[ 0 106]]

Training Accuracy: 0.8

Testing Accuracy: 0.8376623376623377