import numpy as np *# linear algebra*

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

import os

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('/kaggle/input/loan-prediction-problem-dataset/train\_u6lujuX\_CVtuZ9i.csv')

df.head(10)

test = pd.read\_csv('/kaggle/input/loan-prediction-problem-dataset/test\_Y3wMUE5\_7gLdaTN.csv')

test.head(10)

df.info()

print('Gender Mode: ', df['Gender'].mode())

print('Married Mode: ', df['Married'].mode())

print('Self\_Employed Mode: ', df['Self\_Employed'].mode())

print('Credit\_History Mode: ', df['Credit\_History'].mode())

sns.barplot(x = df['Loan\_Amount\_Term'], y = df['LoanAmount'])

df[['Loan\_Amount\_Term', 'LoanAmount']][df['Loan\_Amount\_Term'].isnull()]

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

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

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

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

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

df.isnull().sum()

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

df.nunique()

df.describe()

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

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

plt.xticks(rotation = 90)

plt.grid()

plt.show()

fig, axs = plt.subplots(figsize = (25,6), ncols = 6, nrows = 2)

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

sns.countplot(x = df['Gender'], hue = df['Loan\_Status'], ax = axs[0,1])

sns.countplot(x = df['Married'], hue = df['Loan\_Status'], ax = axs[0,2])

sns.countplot(x = df['Dependents'], hue = df['Loan\_Status'], ax = axs[0,3])

sns.countplot(x = df['Education'], hue = df['Loan\_Status'], ax = axs[0,4])

sns.countplot(x = df['Self\_Employed'], hue = df['Loan\_Status'], ax = axs[0,5])

sns.countplot(x = df['Credit\_History'], hue = df['Loan\_Status'], ax = axs[1,0])

sns.countplot(x = df['Property\_Area'], hue = df['Loan\_Status'], ax = axs[1,1])

sns.countplot(x = df['Gender'], hue = df['Dependents'], ax = axs[1,2])

sns.countplot(x = df['Loan\_Amount\_Term'], hue = df['Loan\_Status'], ax = axs[1,3])

sns.countplot(x = df['Married'], hue = df['Dependents'], ax = axs[1,4])

sns.countplot(x = df['Education'], hue = df['Self\_Employed'], ax = axs[1,5])

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

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)

data = df

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

plt.xticks(rotation = 90)

plt.show()

df.describe()

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()

x = df.iloc[:,:-1]

y = df.iloc[:,-1]

x.head()

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)

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), "**\n**Classification Report: **\n**", classification\_report(ytest, ypred), '**\n**Confusion Matrix: **\n**', confusion\_matrix(ytest, ypred))

print(f'Training Accuracy: **{**train\_accuracy**}\n**Testing 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))