**PROJECT CODE:**  
  
**EXPLORATORY DATA ANALYSIS PART:**

import cudf

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.feature\_selection import chi2, f\_classif

from sklearn.preprocessing import LabelEncoder

from scipy.stats import zscore

df = cudf.read\_csv('credit\_risk\_dataset.csv')

df = df.dropna()

print(df.head())

print(df.shape)

print(df.dtypes)

print(df.isnull().sum())

print(df.describe())

loan\_status\_counts = df['loan\_status'].value\_counts().to\_pandas()

loan\_status\_counts.plot(kind='bar')

plt.xlabel('Loan Status')

plt.ylabel('Frequency')

plt.title('Distribution of Loan Status')

plt.show()

person\_age\_hist = df['person\_age'].to\_pandas()

person\_age\_hist.hist()

plt.xlabel('Person Age')

plt.ylabel('Frequency')

plt.title('Distribution of Person Age')

plt.show()

person\_income\_hist= df['person\_income'].to\_pandas()

person\_income\_hist.hist()

plt.xlabel('Person Income')

plt.ylabel('Frequency')

plt.title('Distribution of Person Income')

plt.show()

person\_home\_hist= df['person\_home\_ownership'].value\_counts().to\_pandas()

person\_home\_hist.hist()

plt.xlabel('Person Home Ownership')

plt.ylabel('Frequency')

plt.title('Distribution of Person Home Ownership')

plt.show()

person\_emp\_hist= df['person\_emp\_length'].to\_pandas()

person\_emp\_hist.hist()

plt.xlabel('Person Employment Length')

plt.ylabel('Frequency')

plt.title('Distribution of Person Employment Length')

plt.show()

loan\_intent\_counts = df['loan\_intent'].value\_counts().to\_pandas()

loan\_intent\_counts.plot(kind='bar')

plt.xlabel('Loan Intent')

plt.ylabel('Frequency')

plt.title('Distribution of Loan Intent')

plt.show()

loan\_grade\_hist=df['loan\_grade'].value\_counts().to\_pandas()

loan\_grade\_hist.hist()

plt.xlabel('Loan Grade')

plt.ylabel('Frequency')

plt.title('Distribution of Loan Grade')

plt.show()

df['loan\_amnt'].to\_pandas().hist()

plt.xlabel('Loan Amount')

plt.ylabel('Frequency')

plt.title('Distribution of Loan Amount')

plt.show()

df['loan\_int\_rate'].to\_pandas().hist()

plt.xlabel('Loan Interest Rate')

plt.ylabel('Frequency')

plt.title('Distribution of Loan Interest Rate')

plt.show()

df['loan\_percent\_income'].to\_pandas().hist()

plt.xlabel('Loan Percent Income')

plt.ylabel('Frequency')

plt.title('Distribution of Loan Percent Income')

plt.show()

le = LabelEncoder()

df['person\_home\_ownership'] = le.fit\_transform(df['person\_home\_ownership'].to\_numpy())

df['loan\_intent'] = le.fit\_transform(df['loan\_intent'].to\_numpy())

df['loan\_grade'] = le.fit\_transform(df['loan\_grade'].to\_numpy())

df['cb\_person\_default\_on\_file'] = le.fit\_transform(df['cb\_person\_default\_on\_file'].to\_numpy())

corr = df.corr()

plt.figure(figsize=(12,10))

sns.heatmap(corr.to\_pandas(), annot=True, cmap='coolwarm')

plt.title('Correlation Matrix')

plt.show()

cat\_vars = ['person\_home\_ownership', 'loan\_intent', 'loan\_grade', 'cb\_person\_default\_on\_file']

chi\_scores = chi2(df[cat\_vars].to\_pandas(), df['loan\_status'].to\_pandas())

p\_values = pd.Series(chi\_scores[1], index=cat\_vars)

p\_values.sort\_values(ascending=False, inplace=True)

print(p\_values)

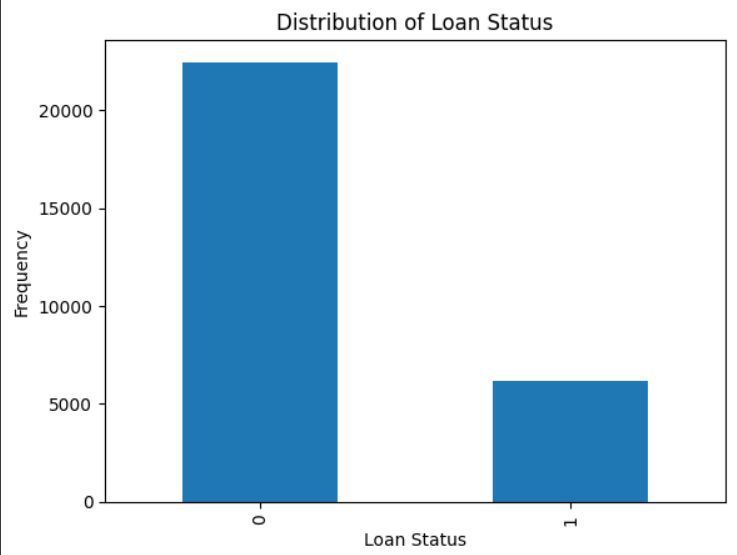
num\_vars = ['person\_age', 'person\_income', 'person\_emp\_length', 'loan\_amnt', 'loan\_int\_rate', 'loan\_percent\_income', 'cb\_person\_cred\_hist\_length']

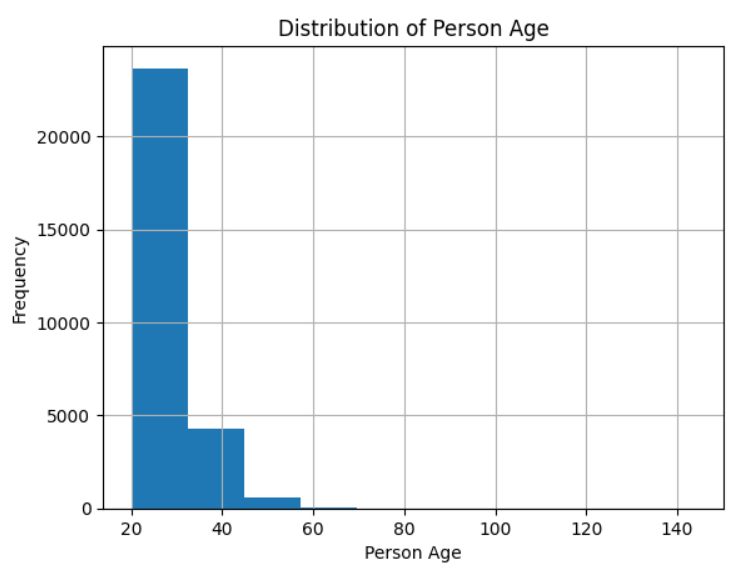
f\_scores = f\_classif(df[num\_vars].to\_pandas(), df['loan\_status'].to\_pandas())

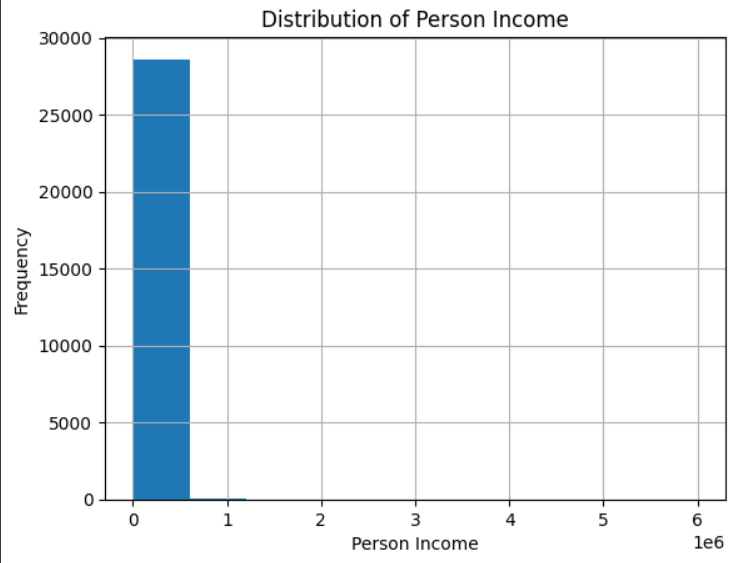
p\_values = pd.Series(f\_scores[1], index=num\_vars)

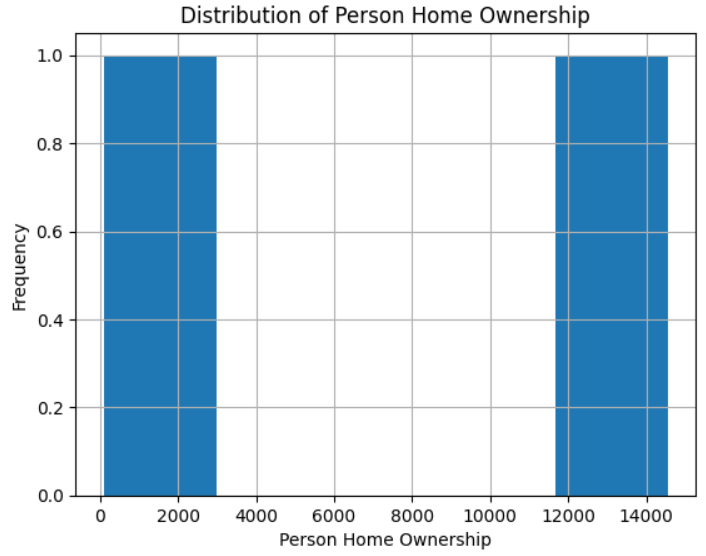
p\_values.sort\_values(ascending=False, inplace=True)

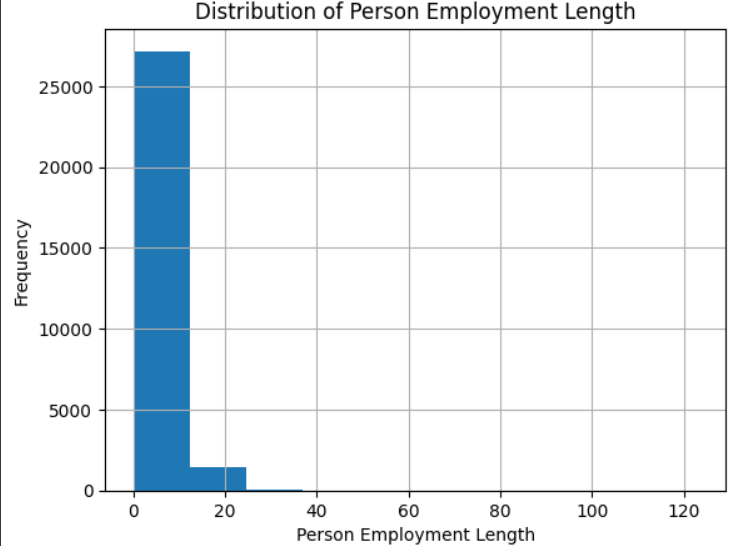
print(p\_values)

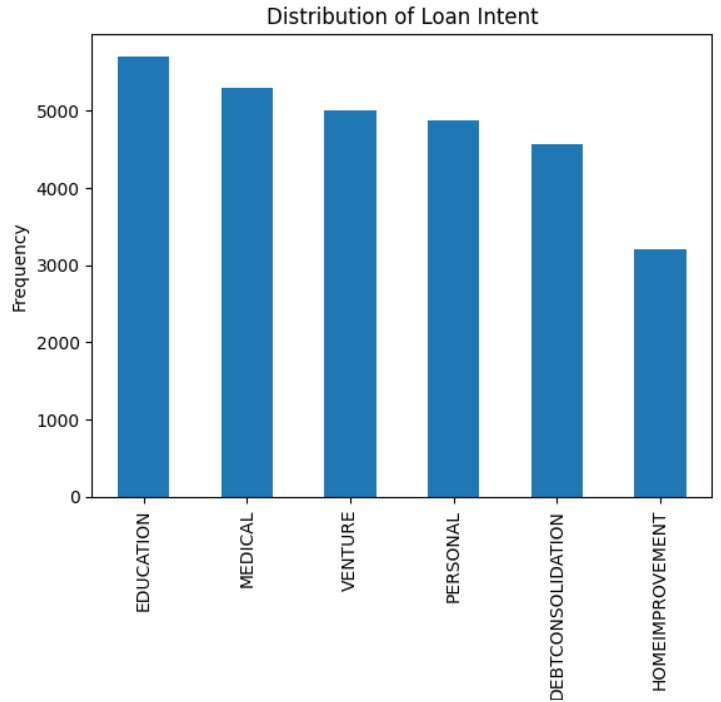
**HISTOGRAMS FOR DISTRIBUTIONS OF EACH FEATURES:**

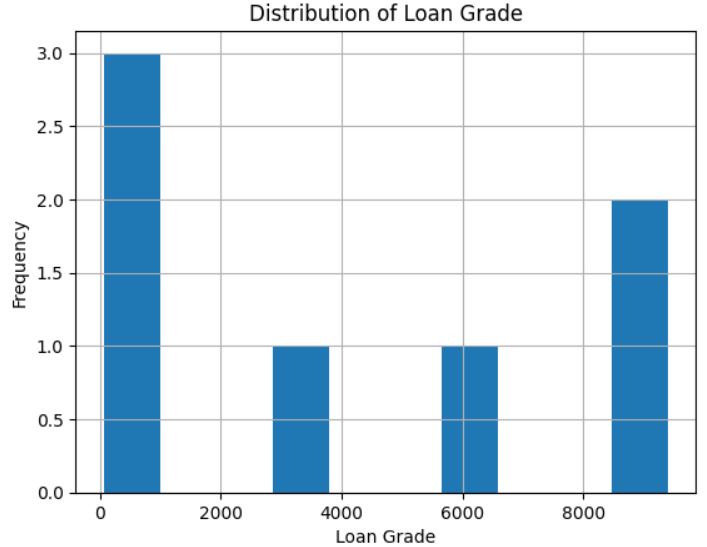
****

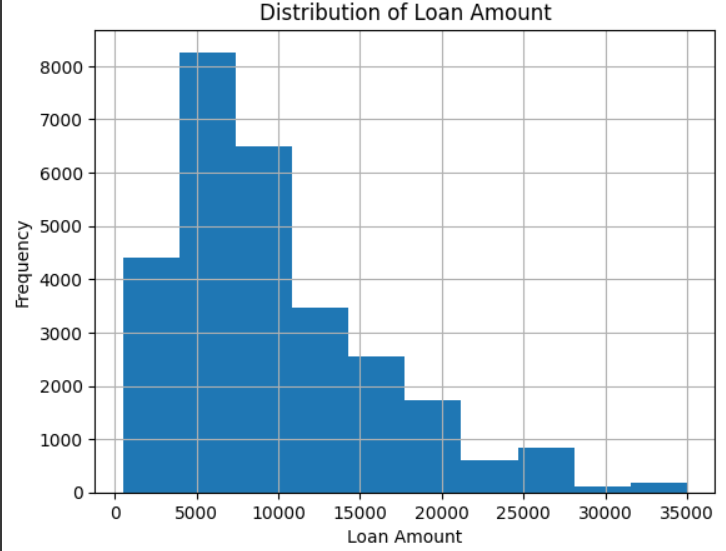
****

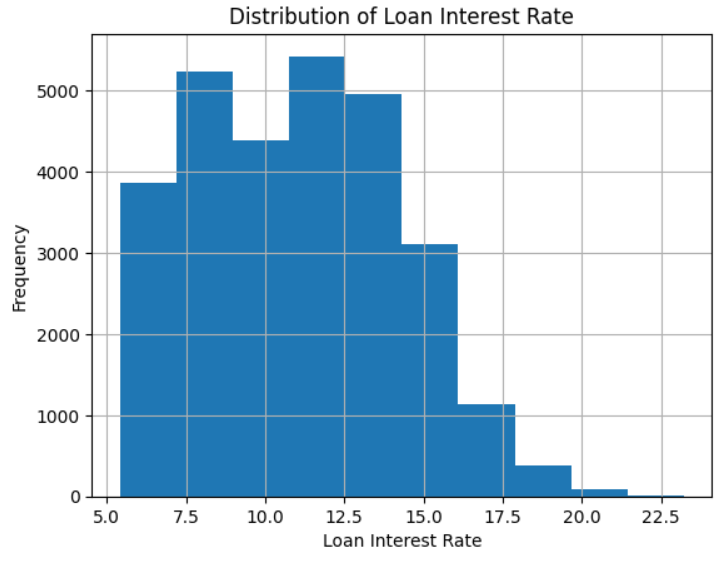
****

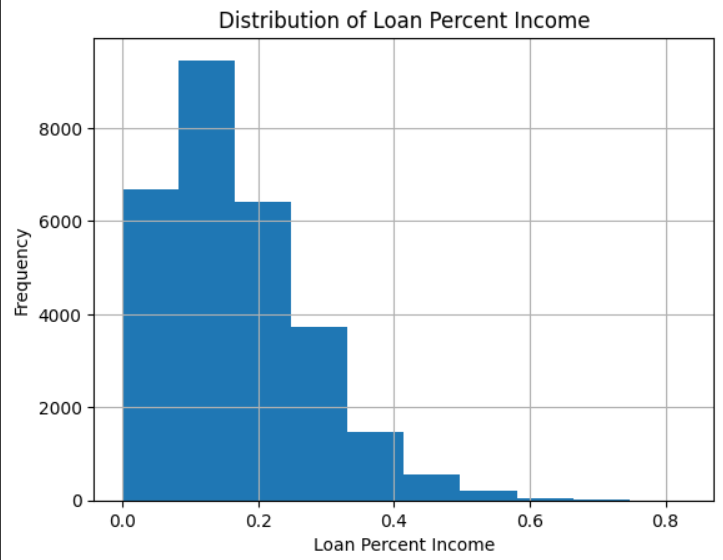
****

****

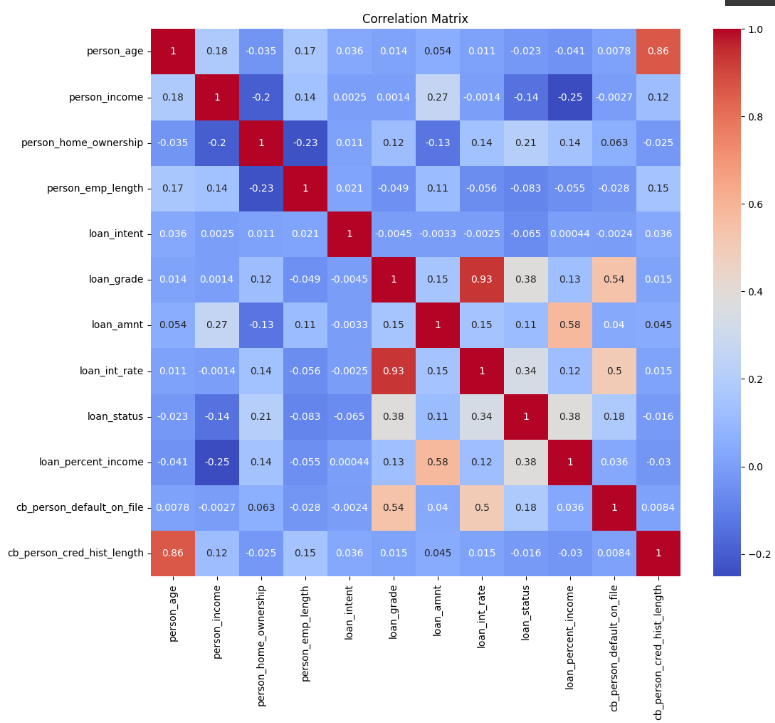
****

****

****

****

**CORRELATION MATRIX:**

****

**MODEL BUILDING FOR PREDICTION PART:**

import torch

import cudf

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.preprocessing import StandardScaler, LabelEncoder, OneHotEncoder

from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

from scipy.stats import zscore

df = cudf.read\_csv('credit\_risk\_dataset.csv')

df = df.dropna()

X = df.drop('loan\_status', axis=1)

y = df['loan\_status']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=42)

scaler = StandardScaler()

X\_train\_num = X\_train.drop(['person\_home\_ownership', 'loan\_intent', 'loan\_grade', 'cb\_person\_default\_on\_file'], axis=1)

X\_train\_num = scaler.fit\_transform(X\_train\_num.to\_numpy())

X\_test\_num = X\_test.drop(['person\_home\_ownership', 'loan\_intent', 'loan\_grade', 'cb\_person\_default\_on\_file'], axis=1)

X\_test\_num = scaler.transform(X\_test\_num.to\_numpy())

encoder = OneHotEncoder(sparse=False)

X\_train\_cat = X\_train[['person\_home\_ownership', 'loan\_intent', 'loan\_grade', 'cb\_person\_default\_on\_file']]

X\_train\_cat = encoder.fit\_transform(X\_train\_cat.to\_pandas())

X\_test\_cat = X\_test[['person\_home\_ownership', 'loan\_intent', 'loan\_grade', 'cb\_person\_default\_on\_file']]

X\_test\_cat = encoder.transform(X\_test\_cat.to\_pandas())

X\_train = np.concatenate((X\_train\_num, X\_train\_cat), axis=1)

X\_test = np.concatenate((X\_test\_num, X\_test\_cat), axis=1)

X\_train = torch.tensor(X\_train, dtype=torch.float32).cuda()

y\_train = torch.tensor(y\_train.values, dtype=torch.float32).cuda()

X\_test = torch.tensor(X\_test, dtype=torch.float32).cuda()

y\_test = torch.tensor(y\_test.values, dtype=torch.float32).cuda()

models = [

    KNeighborsClassifier(),

    LogisticRegression(max\_iter=10000),

    SVC(),

    DecisionTreeClassifier(),

    RandomForestClassifier()

]

model\_names = ['KNN', 'Logistic Regression', 'Support Vector Machines', 'Decision Trees', 'Random Forest']

accuracies = []

for model in models:

    model.fit(X\_train.cpu().numpy(), y\_train.cpu().numpy())

    accuracy = accuracy\_score(y\_test.cpu().numpy(), model.predict(X\_test.cpu().numpy()))

    accuracies.append(accuracy)

results = dict(zip(model\_names, accuracies))

for name, acc in results.items():

    print(f'{name}: {acc:.2f}')

plt.figure(figsize=(10, 5))

plt.bar(results.keys(), results.values(), color=['blue', 'orange', 'green', 'red', 'purple'])

plt.xlabel('Model')

plt.ylabel('Accuracy')

plt.title('Comparison of Model Performance')

plt.show()

**FINAL OUTPUT OF MULTIPLE MODELS:**  
  
**Executing the code in the Google CoLab and below is the screenshot of the execution:**

