from pandas import\*

from numpy import\*

from scipy.stats import\*

from matplotlib import\*

from seaborn import\*

from sklearn.preprocessing import\*

# Import you data and perform basic data exploration phase

df=read\_csv(r"/content/Expresso\_churn\_dataset.csv")

# Display general information about the dataset

df.head()

df.columns

df.dtypes

df.shape

df.info()

df.describe()

# Handle Missing and corrupted values

df.isnull().sum()

df.duplicated().sum()

df.isna().any()

df.isnull().sum()

df.duplicated().sum()

df.REGION.fillna(df.REGION.mode()[0],inplace=True)

df.MONTANT.fillna(df.MONTANT.mean(),inplace=True)

df.FREQUENCE\_RECH.fillna(df.FREQUENCE\_RECH.mean(),inplace=True)

df.REVENUE.fillna(df.REVENUE.mean(),inplace=True)

df.ARPU\_SEGMENT.fillna(df.ARPU\_SEGMENT.mean(),inplace=True)

df.FREQUENCE.fillna(df.FREQUENCE.mean(),inplace=True)

df.DATA\_VOLUME.fillna(df.DATA\_VOLUME.mean(),inplace=True)

df.ORANGE.fillna(0.0,inplace=True)

df.TIGO.fillna(1,inplace=True)

df.ZONE1.fillna(0.0,inplace=True)

df.ZONE2.fillna(0.0,inplace=True)

df.REGULARITY.fillna(1,inplace=True)

df.MRG.fillna(df.MRG.mode()[0],inplace=True)

df.TOP\_PACK.fillna("All-net 500F=2000F;5d",inplace=True)

df.CHURN.fillna(0.0,inplace=True)

df.FREQ\_TOP\_PACK.fillna(1,inplace=True)

df.ON\_NET.loc[df.ZONE1==0.0]=0

df.ON\_NET.loc[df.ZONE2==0.0]=0

df.ON\_NET.fillna(0.0,inplace=True)

df.isnull().sum()

# Handle outliers, if they exist

# Perform feature scaling if needed

a=df.drop(["user\_id","REGION"],axis=1)

for i in a.select\_dtypes(object).columns:

a[i]=LabelEncoder().fit\_transform(a[i])

a.corr()

heatmap(a.corr())

X=a.corr()

ytarget=X

model=[LogisticRegression(),SVC(),DecisionTreeClassifier(),RandomForestClassifier(),GradientBoostingClassifier()]

for i in model:

i.fit(X,ytarget)

print(f"{i}{classification\_report(ytarget,i.predict(X))}")

print(cross\_val\_score(i,X,ytarget,cv=5))

model=GradientBoostingClassifier().fit(X,ytarget)

pred=model.predict(X)

ConfusionMatrixDisplay(confusion\_matrix(ytarget,pred)).plot()