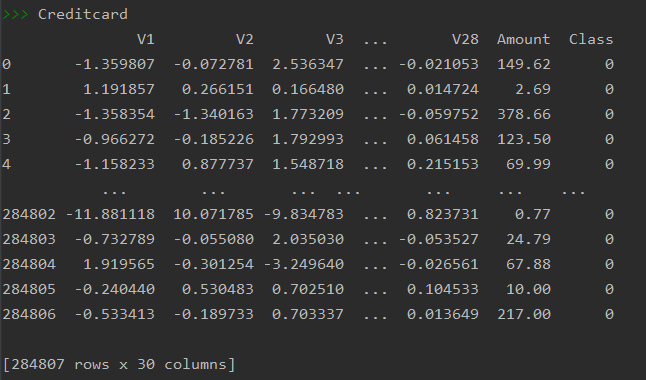
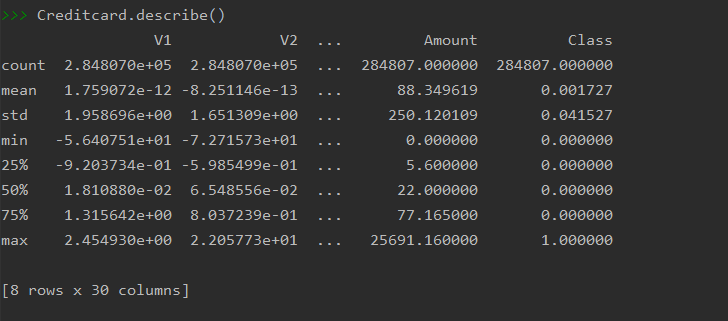
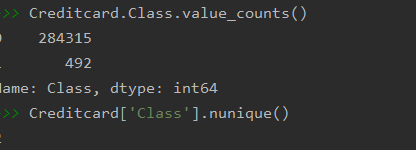
#Librarys

import pandas as pd  
import numpy as np  
import seaborn as sns  
import matplotlib.pyplot as plt  
from sklearn.preprocessing import StandardScaler  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import r2\_score, mean\_absolute\_error, mean\_squared\_error  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import classification\_report, confusion\_matrix

Creditcard=pd.read\_csv('creditcard.csv')  
Creditcard

  
Creditcard.columns  
Creditcard.dtypes  
Creditcard.head(5)  
Creditcard.info()  
Creditcard.isnull().sum()  
Creditcard.describe()

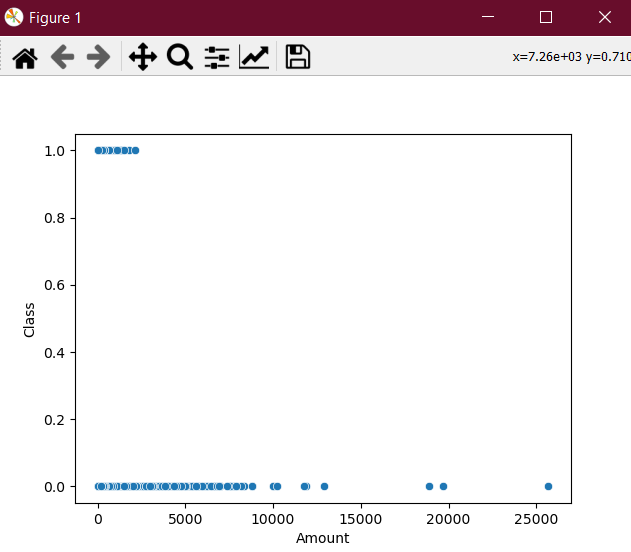
  
Creditcard.Class.value\_counts()

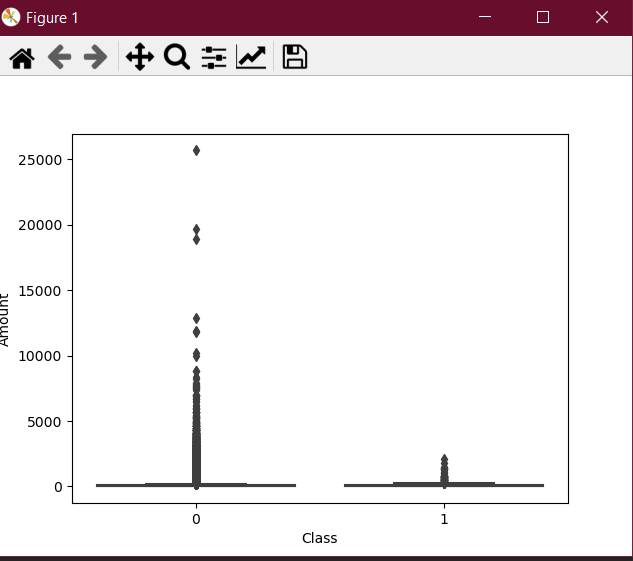
  
Creditcard['Class'].nunique()

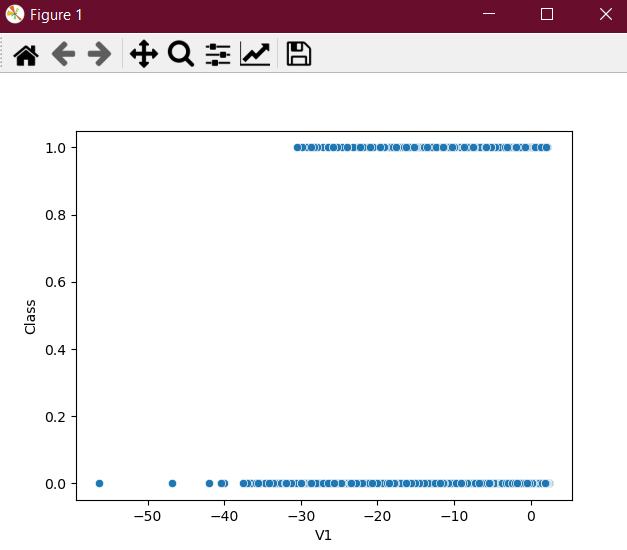
print('Fraud Percentage: {}'.format(round((Creditcard['Class'].value\_counts()[1] / len(Creditcard)) \* 100, 2)))  
print('Non Fraud Percentage: {}'.format(round((Creditcard['Class'].value\_counts()[0] / len(Creditcard)) \* 100, 2)))

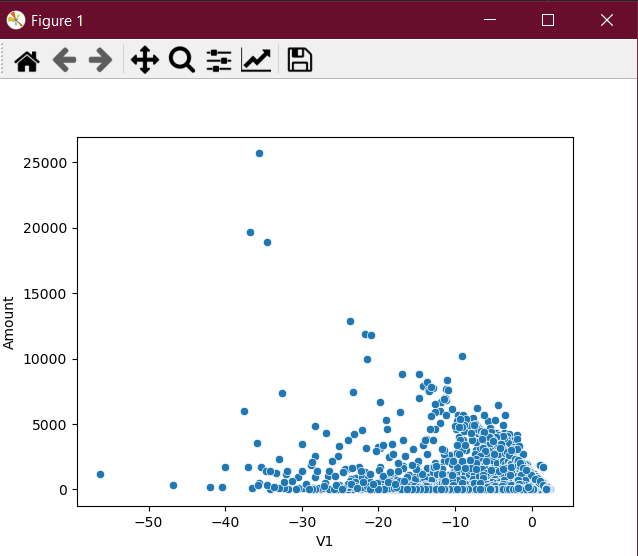


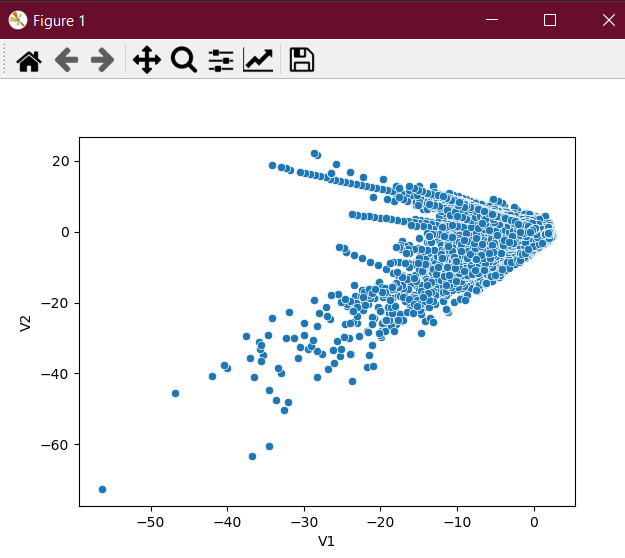
Creditcard.drop("Time", axis=1, inplace=True)  
Creditcard.shape  
  
# visulization  
  
sns.scatterplot(x='Amount', y='Class', data=Creditcard)

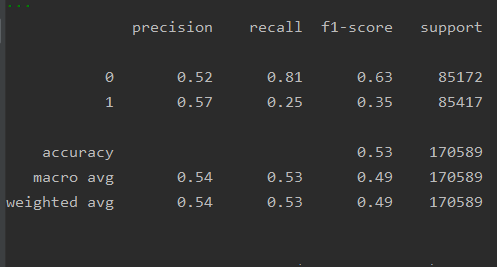
  
sns.boxplot(x="Class", y="Amount", data=Creditcard)

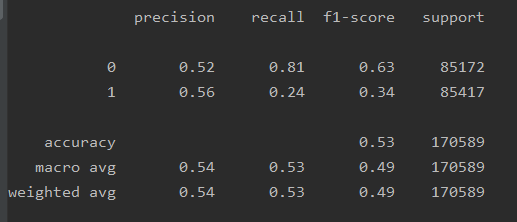
  
plt.ylim(0, 5000)  
plt.show()  
plt.close();  
  
plt.close()  
Creditcard  
sns.scatterplot(x='V1', y='Class', data=Creditcard)

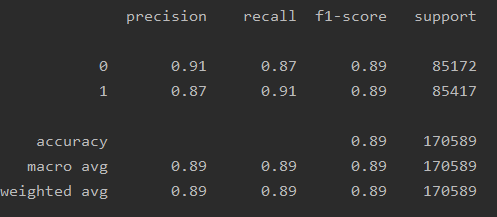
  
sns.scatterplot(x='V1', y='Amount', data=Creditcard)

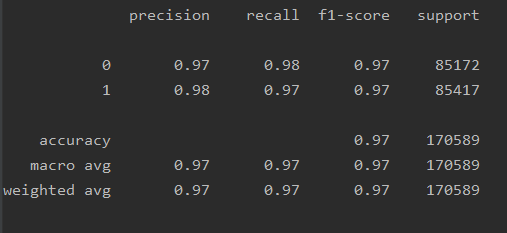
  
sns.scatterplot(x='V2', y='Amount', data=Creditcard)  
sns.scatterplot(x='V3', y='Amount', data=Creditcard)  
sns.scatterplot(x='V4', y='Amount', data=Creditcard)  
sns.scatterplot(x='V5', y='Amount', data=Creditcard)  
sns.scatterplot(x='V6', y='Amount', data=Creditcard)  
sns.scatterplot(x='V7', y='Amount', data=Creditcard)  
sns.scatterplot(x='V8', y='Amount', data=Creditcard)  
sns.scatterplot(x='V9', y='Amount', data=Creditcard)  
sns.scatterplot(x='V10', y='Amount', data=Creditcard)  
sns.scatterplot(x='V11', y='Amount', data=Creditcard)  
sns.scatterplot(x='V12', y='Amount', data=Creditcard)  
sns.scatterplot(x='V13', y='Amount', data=Creditcard)  
sns.scatterplot(x='V14', y='Amount', data=Creditcard)  
sns.scatterplot(x='V15', y='Amount', data=Creditcard)  
sns.scatterplot(x='V16', y='Amount', data=Creditcard)  
sns.scatterplot(x='V17', y='Amount', data=Creditcard)  
sns.scatterplot(x='V18', y='Amount', data=Creditcard)  
sns.scatterplot(x='V19', y='Amount', data=Creditcard)  
sns.scatterplot(x='V20', y='Amount', data=Creditcard)  
sns.scatterplot(x='V21', y='Amount', data=Creditcard)  
sns.scatterplot(x='V22', y='Amount', data=Creditcard)  
sns.scatterplot(x='V23', y='Amount', data=Creditcard)  
sns.scatterplot(x='V24', y='Amount', data=Creditcard)  
sns.scatterplot(x='V25', y='Amount', data=Creditcard)  
sns.scatterplot(x='V26', y='Amount', data=Creditcard)  
sns.scatterplot(x='V27', y='Amount', data=Creditcard)  
sns.scatterplot(x='V28', y='Amount', data=Creditcard)  
sns.scatterplot(x='V27', y='Amount', data=Creditcard)  
sns.scatterplot(x='V1', y='V2', data=Creditcard)

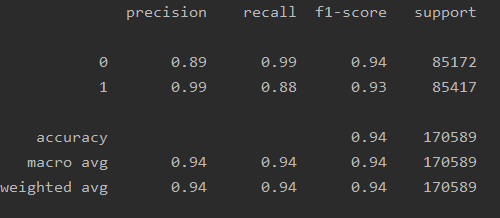
  
sns.scatterplot(x='V1', y='V3', data=Creditcard)  
sns.scatterplot(x='V1', y='V4', data=Creditcard)  
sns.scatterplot(x='V1', y='V5', data=Creditcard)  
sns.scatterplot(x='V1', y='V6', data=Creditcard)  
sns.scatterplot(x='V1', y='V7', data=Creditcard)  
sns.scatterplot(x='V1', y='V8', data=Creditcard)  
sns.scatterplot(x='V1', y='V9', data=Creditcard)  
sns.scatterplot(x='V1', y='V10', data=Creditcard)  
sns.scatterplot(x='V1', y='V11', data=Creditcard)  
sns.scatterplot(x='V1', y='V12', data=Creditcard)  
sns.scatterplot(x='V1', y='V13', data=Creditcard)  
sns.scatterplot(x='V1', y='V14', data=Creditcard)  
sns.scatterplot(x='V1', y='V15', data=Creditcard)  
sns.scatterplot(x='V1', y='V16', data=Creditcard)  
sns.scatterplot(x='V1', y='V17', data=Creditcard)  
sns.scatterplot(x='V1', y='V18', data=Creditcard)  
sns.scatterplot(x='V1', y='V19', data=Creditcard)  
sns.scatterplot(x='V1', y='V20', data=Creditcard)  
sns.scatterplot(x='V1', y='V21', data=Creditcard)  
sns.scatterplot(x='V1', y='V22', data=Creditcard)  
sns.scatterplot(x='V1', y='V23', data=Creditcard)  
sns.scatterplot(x='V1', y='V24', data=Creditcard)  
sns.scatterplot(x='V1', y='V25', data=Creditcard)  
sns.scatterplot(x='V1', y='V26', data=Creditcard)  
sns.scatterplot(x='V1', y='V27', data=Creditcard)  
sns.scatterplot(x='V1', y='V28', data=Creditcard)  
  
# model Building  
# LogisticRegression  
X = Creditcard[['Amount']]  
y = Creditcard['Class']  
sc = StandardScaler()  
sc  
X  
X = sc.fit\_transform(X)  
X  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)  
log\_reg = LogisticRegression(random\_state=0)  
log\_reg.fit(X\_train, y\_train)  
log\_reg.coef\_  
log\_reg.intercept\_  
y\_pred = log\_reg.predict(X\_test)  
confusion\_matrix(y\_test, y\_pred)  
  
print(classification\_report(y\_test, y\_pred))  
  
# smote  
from imblearn.over\_sampling import SMOTE  
  
smote\_oversample = SMOTE()  
X, y = smote\_oversample.fit\_resample(X, y)  
X.shape  
y.shape  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=0)  
log\_reg = LogisticRegression(random\_state=0)  
log\_reg.fit(X\_train, y\_train)  
y\_pred = log\_reg.predict(X\_test)  
print(classification\_report(y\_test, y\_pred))

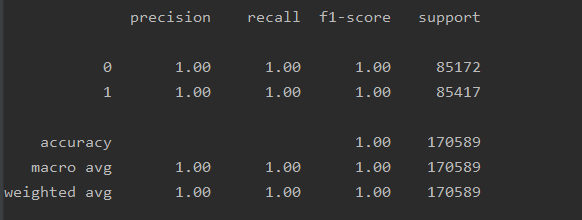
  
  
# LinearSVC  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.svm import LinearSVC  
  
svc\_reg = LinearSVC(random\_state=0)  
svc\_reg.fit(X\_train, y\_train)  
y\_pred = svc\_reg.predict(X\_test)  
print(classification\_report(y\_test, y\_pred))

  
  
# RandomForest  
rf\_reg = RandomForestClassifier(random\_state=0)  
rf\_reg.fit(X\_train, y\_train)  
y\_pred = rf\_reg.predict(X\_test)  
print(classification\_report(y\_test, y\_pred))

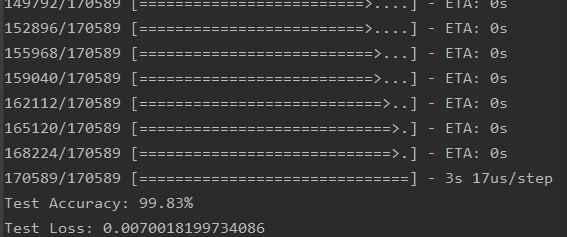
  
  
# scaling data using standard scaller  
X = Creditcard.drop(['Class'], axis=1)  
y = Creditcard['Class']  
sc = StandardScaler()  
smote\_oversample = SMOTE()  
X, y = smote\_oversample.fit\_resample(X, y)  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=0)  
  
log\_reg = LogisticRegression(random\_state=0)  
log\_reg.fit(X\_train, y\_train)  
y\_pred = log\_reg.predict(X\_test)  
print(classification\_report(y\_test, y\_pred))

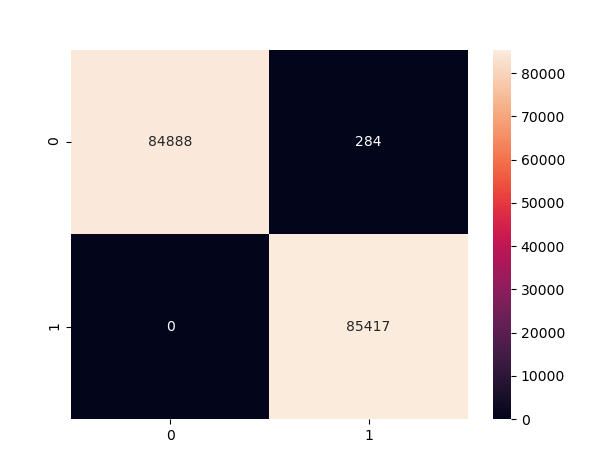
  
  
X = sc.fit\_transform(X)  
svc\_reg = LinearSVC(random\_state=0)  
svc\_reg.fit(X\_train, y\_train)  
y\_pred = svc\_reg.predict(X\_test)  
print(classification\_report(y\_test, y\_pred))

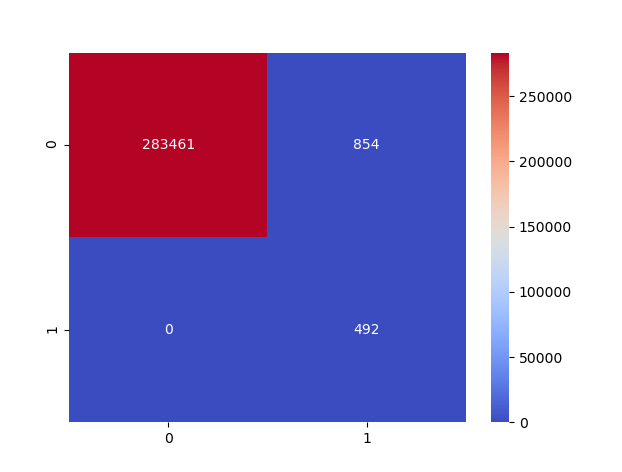
  
  
rf\_reg = RandomForestClassifier(random\_state=0)  
rf\_reg.fit(X\_train, y\_train)  
y\_pred = rf\_reg.predict(X\_test)  
print(classification\_report(y\_test, y\_pred))



ANN  
  
from sklearn.preprocessing import StandardScaler  
Creditcard['Amount(Normalized)'] = StandardScaler().fit\_transform(Creditcard['Amount'].values.reshape(-1,1)  
  
Creditcard.iloc[:,[29,31]].head()  
  
Creditcard = Creditcard.drop(columns = ['Amount', 'Time'], axis=1) # This columns are not necessary anymore.  
  
X = Creditcard.drop('Class', axis=1)  
y = Creditcard['Class']  
  
#train-test split  
  
from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)  
# We are transforming data to numpy array to implementing with keras  
X\_train = np.array(X\_train)  
X\_test = np.array(X\_test)  
y\_train = np.array(y\_train)  
y\_test = np.array(y\_test)  
  
#ANN  
from keras.models import Sequential  
from keras.layers import Dense, Dropout  
model = Sequential([  
 Dense(units=20, input\_dim = X\_train.shape[1], activation='relu'),  
 Dense(units=24,activation='relu'),  
 Dropout(0.5),  
 Dense(units=20,activation='relu'),  
 Dense(units=24,activation='relu'),  
 Dense(1, activation='sigmoid')  
])  
model.summary()  
  
model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])  
model.fit(X\_train, y\_train, batch\_size=30, epochs=5)  
  
model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])  
model.fit(X\_train, y\_train, batch\_size=30, epochs=5)  
  
score = model.evaluate(X\_test, y\_test)  
print('Test Accuracy: {:.2f}%\nTest Loss: {}'.format(score[1]\*100,score[0]))  
  
#SMOTE  
  
from imblearn.over\_sampling import SMOTE  
X\_smote, y\_smote = SMOTE().fit\_sample(X, y)  
X\_smote = pd.DataFrame(X\_smote)  
y\_smote = pd.DataFrame(y\_smote)  
y\_smote.iloc[:,0].value\_counts()  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_smote, y\_smote, test\_size=0.3, random\_state=0)  
X\_train = np.array(X\_train)  
X\_test = np.array(X\_test)  
y\_train = np.array(y\_train)  
y\_test = np.array(y\_test)  
model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])  
model.fit(X\_train, y\_train, batch\_size = 30, epochs = 5)  
  
#Accurancy  
score = model.evaluate(X\_test, y\_test)  
print('Test Accuracy: {:.2f}%\nTest Loss: {}'.format(score[1]\*100,score[0]))

#Output  
  
  
#Confusion matrix1  
y\_pred = model.predict(X\_test)  
y\_test = pd.DataFrame(y\_test)  
cm = confusion\_matrix(y\_test, y\_pred.round())  
sns.heatmap(cm, annot=True, fmt='.0f')  
plt.show()

  
#Confusion matrix2  
y\_pred2 = model.predict(X)  
y\_test2 = pd.DataFrame(y)  
cm2 = confusion\_matrix(y\_test2, y\_pred2.round())  
sns.heatmap(cm2, annot=True, fmt='.0f', cmap='coolwarm')  
plt.show()

  
  
scoreNew = model.evaluate(X, y)  
print('Test Accuracy: {:.2f}%\nTest Loss: {}'.format(scoreNew[1]\*100,scoreNew[0]))  
  
print(classification\_report(y\_test2, y\_pred2.round()))

