```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

maldata = pd.read_csv("", sep="|", low_memory = True)
maldata.head()
maldata.shape
maldata.dscribe()

legit = maldata[0:41323].drop(["legitimate"], axis=1)
mal = maldata[41323::].drop(["legitimate"], axis=1)
print("The shape of the legit dataset is: %s samples, %s features"%(legit.shape[0],
legit.shape[1]))
print("The shape of the mal dataset is: %s samples, %s features" %(mal.shape[0],
mal.shape[1]))

fig = plt.figure()
ax = fig.add_axes([0, 0, 1, 1])
ax.hist(maldata['legitimate'], 20)
plt.show()

y = maldata['legitimate']
maldata = maldata.drop(['legitimate'], axis = 1)

maldata = maldata.drop(['Name'], axis=1)
maldata = maldata.drop(['md5'], axis=1)
print(" The Name and md5 variable are removed successfully")

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(maldata, y, test_size=0.2,
random_state=42)
x_train.shape

from sklearn.metrics import
f1_score,accuracy_score,plot_confusion_matrix,auc,confusion_matrix


titles_options = [("Confusion matrix, without normalization", None),
("Normalized confusion matrix", 'true')]

for title, normalize in titles_options:
disp = plot_confusion_matrix(model, x_test, y_test,
display_labels = 'legitimate',
cmap = plt.cm.Blues,
normalize = normalize)

disp.ax_.set_title(title)

print(title)
print(disp.confusion_matrix)

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

model = Sequential()
model.add(Dense(16, input_dim=54, activation="relu"))
model.add(Dense(8, activation="relu"))
model.add(Dense(4, activation="relu"))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss="binary_crossentropy", optimizer="rmspop", metrics=["accuracy"])
```

```
model.fit(x_train, y_train, epochs=5, batch_size=32)

trainPred = model.predict(x_train)

trainPred = [1 if y >= 0.5 else 0 for y in trainPred]
accuracy_score(y_train, trainPred)

y_prediction = model.predict(x_test)
y_prediction = [1 if y >= 0.5 else 0 for y in y_prediction]
accuracy_score(y_test, y_prediction)

confusion_matrix(y_test, y_prediction)

f1_score(y_test, y_prediction)
```