In [1]:

```
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
%matplotlib inline

df = pd.read_csv("creditcard.csv")
df.head()
```

Out[1]:

4	V5	V6	V7	V8	V9	...	V21	V22	V23	
5	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.0
4	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.3
0	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.6
1	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.1
4	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.1

In [4]:

```
from sklearn.preprocessing import StandardScaler

std_scaler = StandardScaler()

df_amt = std_scaler.fit_transform(df[['Amount']])
df_time = std_scaler.fit_transform(df[['Time']])

df_amt = pd.DataFrame(df_amt)
df_time = pd.DataFrame(df_time)
df.drop(['Amount',"Time"],axis = "columns",inplace=True)
```

In [5]:

```
df.insert(0, 'scaled_amount', df_amt)
df.insert(1, 'scaled_time', df_time)
```

In [8]:

```
X = df.drop("Class",axis = "columns")
y = df["Class"]
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state = 2)
X_train_res, y_train_res = sm.fit_resample(X, y)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_train_res, y_train_res, test_size=0
```

In [10]:

```
X_train.shape,X_test.shape
```

Out[10]:

```
((454904, 30), (113726, 30))
```

In [11]:

```
import tensorflow as tf
from tensorflow import keras


model = keras.Sequential([
    keras.layers.Dense(30, input_shape=(30,), activation='relu'),
    keras.layers.Dense(10, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])


model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

model.fit(X_train, y_train, epochs=10)
```

```
Epoch 1/10
14216/14216 [==============================] - 11s 744us/step - loss: 0.03
00 - accuracy: 0.9899
Epoch 2/10
14216/14216 [==============================] - 12s 834us/step - loss: 0.00
77 - accuracy: 0.9983
Epoch 3/10
14216/14216 [==============================] - 10s 737us/step - loss: 0.00
55 - accuracy: 0.9987
Epoch 4/10
14216/14216 [==============================] - 11s 748us/step - loss: 0.00
45 - accuracy: 0.9989
Epoch 5/10
14216/14216 [==============================] - 11s 755us/step - loss: 0.00
39 - accuracy: 0.9991
Epoch 6/10
14216/14216 [==============================] - 11s 759us/step - loss: 0.00
36 - accuracy: 0.9992
Epoch 7/10
14216/14216 [==============================] - 10s 732us/step - loss: 0.00
29 - accuracy: 0.9993
Epoch 8/10
14216/14216 [==============================] - 10s 735us/step - loss: 0.00
28 - accuracy: 0.9994
Epoch 9/10
14216/14216 [==============================] - 10s 724us/step - loss: 0.00
24 - accuracy: 0.9995
Epoch 10/10
14216/14216 [==============================] - 10s 736us/step - loss: 0.00
22 - accuracy: 0.9995
```

Out[11]:

```
<keras.src.callbacks.History at 0x1f334b41940>
```

In [12]:

```
model.evaluate(X_test, y_test)
```

```
3554/3554 [==============================] - 2s 636us/step - loss: 0.0031
- accuracy: 0.9994
```

Out[12]:

```
[0.0030587059445679188, 0.9993668794631958]
```

In [13]:

```
yp = model.predict(X_test)
y_pred = []
for element in yp:
    if element > 0.5:
        y_pred.append(1)
    else:
        y_pred.append(0)
from sklearn.metrics import confusion_matrix , classification_report

print(classification_report(y_test,y_pred))
```

```
3554/3554 [==============================] - 2s 558us/step
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56832
           1       1.00      1.00      1.00     56894

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726
```

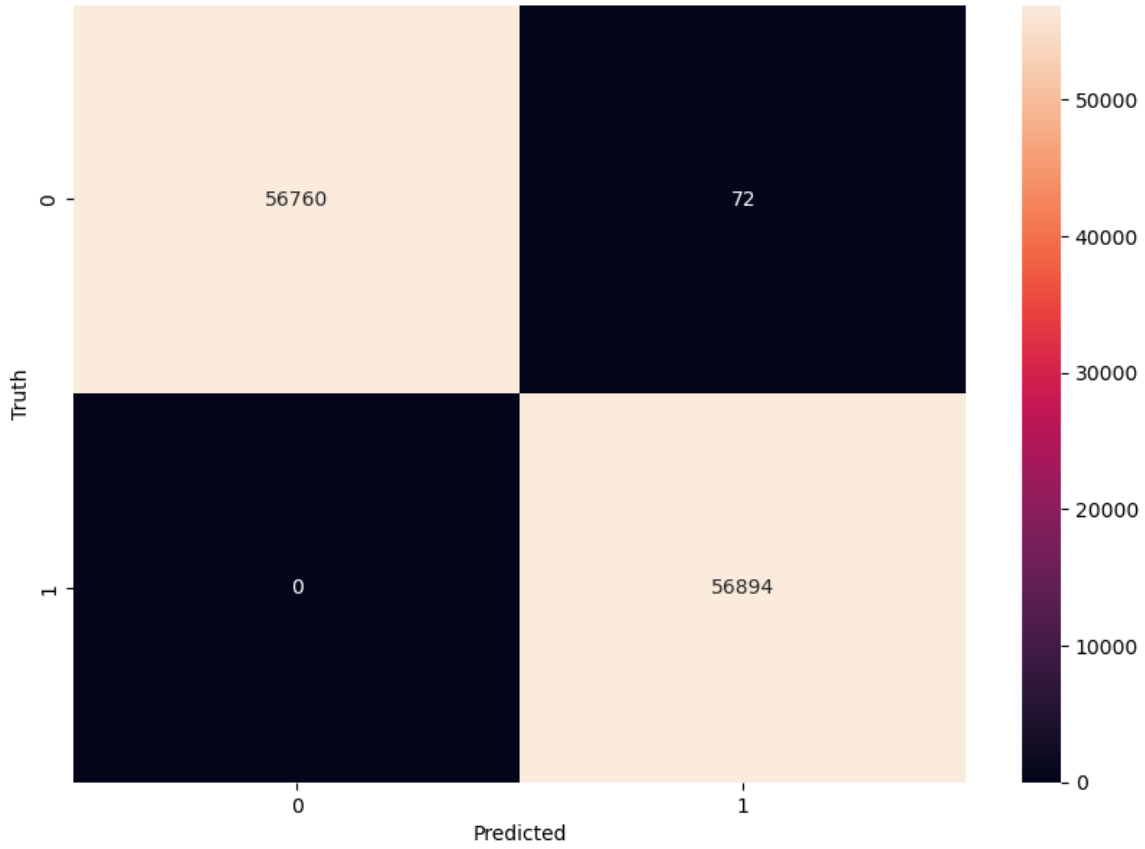
In [14]:

```
import seaborn as sn
cm = tf.math.confusion_matrix(labels=y_test,predictions=y_pred)

plt.figure(figsize = (10,7))
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[14]:

```
Text(95.72222222222221, 0.5, 'Truth')
```



In [15]:

```
y_pred[:5]
```

Out[15]:

```
[0, 0, 0, 1, 1]
```

In [16]:

```
y_test[:5]
```

Out[16]:

```
81766     0
172447    0
67445     0
306017    1
433208    1
Name: Class, dtype: int64
```

In [ ]: