Importing Essential Libraries and Our Dataset

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
dataset = pd.read_csv('creditcard.csv')
```

In [4]:
```
dataset.head()
```

Out[4]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817

5 rows × 31 columns

Data Pre-Processing

In [7]:
```
from sklearn.preprocessing import StandardScaler
dataset['normalizedAmount'] = StandardScaler().fit_transform(dataset['Amount'].value
dataset = dataset.drop(['Amount'], axis = 1)
```

In [9]:
```
dataset.head()
```

Out[9]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817

5 rows × 31 columns

In [10]:
```
dataset = dataset.drop(['Time'], axis = 1)
dataset.head()
```

Out[10]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787

	V1	V2	V3	V4	V5	V6	V7	V8	V9	
1	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-(
2	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	(
3	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-(
4	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	(

5 rows × 30 columns

In [20]:
```
dataset.isnull().sum()
```

Out[20]:
```
V1                  0
V2                  0
V3                  0
V4                  0
V5                  0
V6                  0
V7                  0
V8                  0
V9                  0
V10                 0
V11                 0
V12                 0
V13                 0
V14                 0
V15                 0
V16                 0
V17                 0
V18                 0
V19                 0
V20                 0
V21                 0
V22                 0
V23                 0
V24                 0
V25                 0
V26                 0
V27                 0
V28                 0
Class               0
normalizedAmount    0
dtype: int64
```

In [17]:
```
X = dataset.drop(columns = ['Class'])
y = dataset['Class']
```

In [18]:
```
X.head()
```

Out[18]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	(
1	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-(
2	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	(
3	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-(
4	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	(

5 rows × 29 columns

In [19]:
```
y.head()
```

Out[19]:
```
0    0
1    0
2    0
3    0
4    0
Name: Class, dtype: int64
```

In [21]:
```
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_st
```

In [22]:
```
X_train.shape
```

Out[22]:
```
(199364, 29)
```

In [23]:
```
X_test.shape
```

Out[23]:
```
(85443, 29)
```

In [24]:
```
y_train.shape
```

Out[24]:
```
(199364,)
```

In [25]:
```
y_test.shape
```

Out[25]:
```
(85443,)
```

In [26]:
```
X_train = np.array(X_train)
X_test = np.array(X_test)
y_train = np.array(y_train)
y_test = np.array(y_test)
```

Deep Neural Network

In [37]:
```
import tensorflow.keras
tensorflow.__version__
```

Out[37]:
```
'2.3.1'
```

In [43]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
```

In [44]:
```
model = Sequential([
    Dense(units = 16, input_dim = 29, activation = 'relu'),
    Dense(units = 24, activation = 'relu'),
    Dropout(0.5),
    Dense(20, activation = 'relu'),
    Dense(24, activation = 'relu'),
    Dense(1, activation = 'sigmoid')
])
```

In [46]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
```

```
dense (Dense)                (None, 16)                480
_________________________________________________________________
dense_1 (Dense)              (None, 24)                408
_________________________________________________________________
dropout (Dropout)            (None, 24)                0
_________________________________________________________________
dense_2 (Dense)              (None, 20)                500
_________________________________________________________________
dense_3 (Dense)              (None, 24)                504
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 25
=================================================================
Total params: 1,917
Trainable params: 1,917
Non-trainable params: 0
_________________________________________________________________
```

In [48]:
```
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy
model.fit(X_train, y_train, batch_size = 15, epochs = 5)
```

```
Epoch 1/5
13291/13291 [==============================] - 27s 2ms/step - loss: 0.0092 - accurac
y: 0.9985
Epoch 2/5
13291/13291 [==============================] - 29s 2ms/step - loss: 0.0041 - accurac
y: 0.9992 0s - loss: 0.0041 - accura
Epoch 3/5
13291/13291 [==============================] - 28s 2ms/step - loss: 0.0037 - accurac
y: 0.9993
Epoch 4/5
13291/13291 [==============================] - 27s 2ms/step - loss: 0.0035 - accurac
y: 0.9993
Epoch 5/5
13291/13291 [==============================] - 22s 2ms/step - loss: 0.0034 - accurac
y: 0.9994
```

Out[48]:
```
<tensorflow.python.keras.callbacks.History at 0x15c2f1e3520>
```

In [49]:
```
score = model.evaluate(X_test, y_test)
```

```
2671/2671 [==============================] - 3s 1ms/step - loss: 0.0035 - accuracy:
0.9994
```

In [50]:
```
print(score)
```

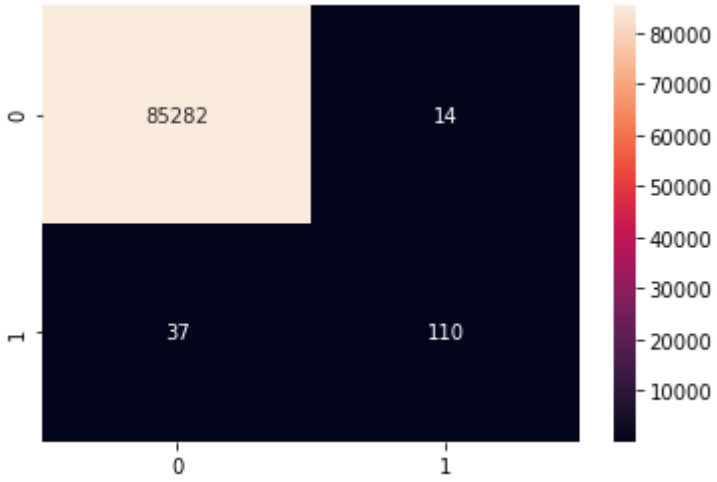
```
[0.003530588699504733, 0.9994031190872192]
```

In [51]:
```
y_pred = model.predict(X_test)
y_test = pd.DataFrame(y_test)
```

In [52]:
```
from sklearn.metrics import confusion_matrix, classification_report, precision_score
```

In [58]:
```
cm = confusion_matrix(y_test, y_pred.round())
sns.heatmap(cm, annot = True, fmt = 'g')
```

Out[58]:
```
<AxesSubplot:>
```



In [60]:
```
from imblearn.over_sampling import SMOTE
```

In [61]:
```
X_resample, y_resample = SMOTE().fit_sample(X, y)
```

In [62]:
```
y_resample = pd.DataFrame(y_resample)
X_resample = pd.DataFrame(X_resample)
```

In [63]:
```
X_train, X_test, y_train, y_test = train_test_split(X_resample, y_resample, test_siz
```

In [64]:
```
X_train = np.array(X_train)
X_test = np.array(X_test)
y_train = np.array(y_train)
y_test = np.array(y_test)
```

In [65]:
```
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy
model.fit(X_train, y_train, batch_size = 15, epochs = 5)
```

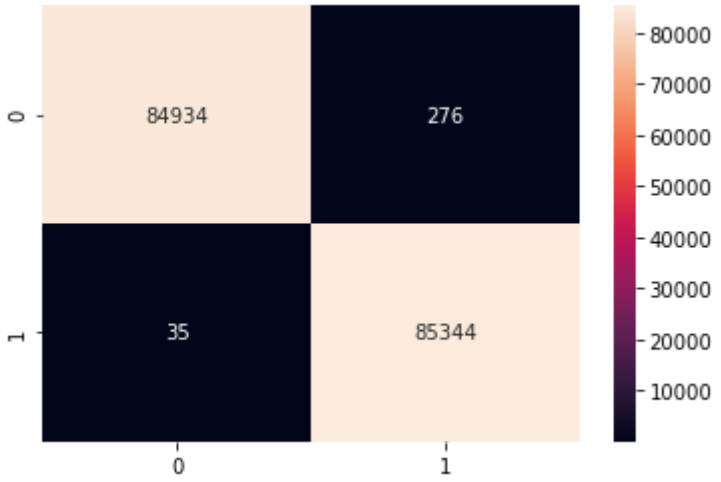
```
Epoch 1/5
26537/26537 [==============================] - 47s 2ms/step - loss: 0.0293 - accurac
y: 0.9902
Epoch 2/5
26537/26537 [==============================] - 48s 2ms/step - loss: 0.0124 - accurac
y: 0.9968
Epoch 3/5
26537/26537 [==============================] - 49s 2ms/step - loss: 0.0103 - accurac
y: 0.9974
Epoch 4/5
26537/26537 [==============================] - 49s 2ms/step - loss: 0.0090 - accurac
y: 0.9978
Epoch 5/5
26537/26537 [==============================] - 47s 2ms/step - loss: 0.0084 - accurac
y: 0.9980
```

Out[65]:
```
<tensorflow.python.keras.callbacks.History at 0x15c312e96a0>
```

In [66]:
```
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 = 'g')
```

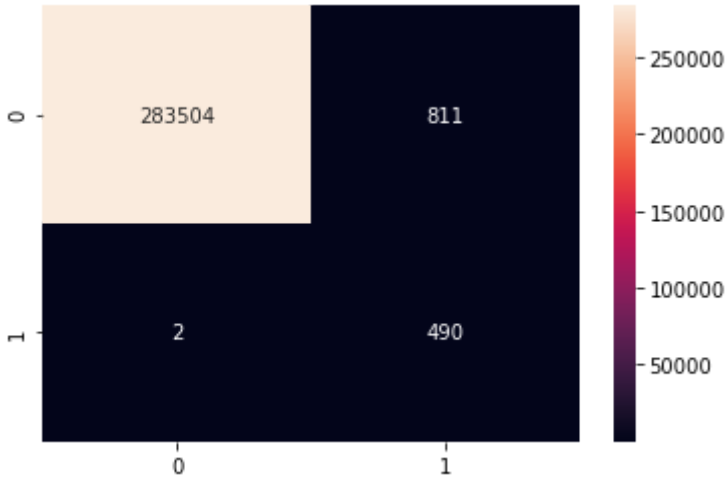
Out[66]:
```
<AxesSubplot:>
```



In [67]:
```
y_pred = model.predict(X)
y_test = pd.DataFrame(y)

cm = confusion_matrix(y_test, y_pred.round())
sns.heatmap(cm, annot = True, fmt = 'g')
```

Out[67]: <AxesSubplot:>



THE END