Importing Neccessary Libraries

```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import BatchNormalization,Dropout,Dense,Flatt
from tensorflow.keras.optimizers import Adam

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

Gathering the data and assessing the data

```
df = pd.read_csv('/content/creditcard.csv')
```

```
df.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	
0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.09
1	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.08
2	1	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.24
3	1	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.37
4	2	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.27

```
df.shape
```

```
(3973, 31)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3973 entries, 0 to 3972
Data columns (total 31 columns):
 #   Column  Non-Null Count  Dtype
```

```
---  ------  --------------  -----
 0   Time    3973 non-null   int64
 1   V1      3973 non-null   float64
 2   V2      3973 non-null   float64
 3   V3      3973 non-null   float64
 4   V4      3973 non-null   float64
 5   V5      3973 non-null   float64
 6   V6      3973 non-null   float64
 7   V7      3973 non-null   float64
 8   V8      3973 non-null   float64
 9   V9      3973 non-null   float64
 10  V10     3973 non-null   float64
 11  V11     3973 non-null   float64
 12  V12     3973 non-null   float64
 13  V13     3973 non-null   float64
 14  V14     3973 non-null   float64
 15  V15     3973 non-null   float64
 16  V16     3973 non-null   float64
 17  V17     3973 non-null   float64
 18  V18     3973 non-null   float64
 19  V19     3973 non-null   float64
 20  V20     3973 non-null   float64
 21  V21     3973 non-null   float64
 22  V22     3973 non-null   float64
 23  V23     3972 non-null   float64
 24  V24     3972 non-null   float64
 25  V25     3972 non-null   float64
 26  V26     3972 non-null   float64
 27  V27     3972 non-null   float64
 28  V28     3972 non-null   float64
 29  Amount  3972 non-null   float64
 30  Class   3972 non-null   float64
dtypes: float64(30), int64(1)
memory usage: 962.3 KB
```

```
df.Class.unique()
```

```
array([ 0.,  1., nan])
```

Uneven class distribution

```
df.Class.value_counts()
```

```
0.0    3970
1.0       2
Name: Class, dtype: int64
```

```
nf = df[df.Class==0]
f = df[df.Class==1]
```

Extracting random entries of class-0

Total entries are 1.5* NO. of class-1 entries

```
nf = nf.sample(738)
```

Creating new dataframe

```
data = f.append(nf,ignore_index=True)
```

```
data.shape
```

```
(740, 31)
```

```
X = data.drop(['Class'],axis=1)
y=data['Class']
```

Train-Test Split

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,stratify=
```

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

```
((592, 30), (148, 30))
```

Applying StandardScaler to obtain all the features in similar range

```
scaler=StandardScaler()
X_train=scaler.fit_transform(X_train)
X_test=scaler.transform(X_test)
```

```
y_train=y_train.to_numpy()
y_test=y_test.to_numpy()
```

Reshaping the input to 3D.

```
X_train=X_train.reshape(X_train.shape[0],X_train.shape[1],1)
X_test=X_test.reshape(X_test.shape[0],X_test.shape[1],1)
```

CNN model

```
model=Sequential()
model.add(Conv1D(32,2,activation='relu',input_shape=X_train[0].shape))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Conv1D(64,2,activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Flatten())
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(1,activation='sigmoid'))


model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 29, 32)            96

 batch_normalization (BatchN  (None, 29, 32)           128
 ormalization)

 dropout (Dropout)           (None, 29, 32)            0

 conv1d_1 (Conv1D)           (None, 28, 64)            4160

 batch_normalization_1 (Batc  (None, 28, 64)           256
 hNormalization)

 dropout_1 (Dropout)         (None, 28, 64)            0

 flatten (Flatten)           (None, 1792)              0

 dense (Dense)               (None, 64)                114752

 dropout_2 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

=================================================================
```

```
Total params: 119,457
Trainable params: 119,265
Non-trainable params: 192
_________________________________________________________________
```

Compiling and Fiting

```
model.compile(optimizer=Adam(learning_rate=0.0001),loss='binary_crossentro
```

```
history = model.fit(X_train,y_train,epochs=20,validation_data=(X_test,y_te
```

```
Epoch 1/20
19/19 [==============================] - 2s 28ms/step - loss: 0.9555 - accuracy: 0.59
Epoch 2/20
19/19 [==============================] - 0s 12ms/step - loss: 0.8238 - accuracy: 0.69
Epoch 3/20
19/19 [==============================] - 0s 12ms/step - loss: 0.6821 - accuracy: 0.72
Epoch 4/20
19/19 [==============================] - 0s 11ms/step - loss: 0.5871 - accuracy: 0.86
Epoch 5/20
19/19 [==============================] - 0s 12ms/step - loss: 0.4421 - accuracy: 0.82
Epoch 6/20
19/19 [==============================] - 0s 11ms/step - loss: 0.3362 - accuracy: 0.85
Epoch 7/20
19/19 [==============================] - 0s 12ms/step - loss: 0.4105 - accuracy: 0.86
Epoch 8/20
19/19 [==============================] - 0s 12ms/step - loss: 0.3677 - accuracy: 0.86
Epoch 9/20
19/19 [==============================] - 0s 12ms/step - loss: 0.2674 - accuracy: 0.90
Epoch 10/20
19/19 [==============================] - 0s 11ms/step - loss: 0.2311 - accuracy: 0.89
Epoch 11/20
19/19 [==============================] - 0s 10ms/step - loss: 0.1998 - accuracy: 0.92
Epoch 12/20
19/19 [==============================] - 0s 11ms/step - loss: 0.1767 - accuracy: 0.92
Epoch 13/20
19/19 [==============================] - 0s 11ms/step - loss: 0.1518 - accuracy: 0.94
Epoch 14/20
19/19 [==============================] - 0s 11ms/step - loss: 0.1737 - accuracy: 0.93
Epoch 15/20
19/19 [==============================] - 0s 11ms/step - loss: 0.1329 - accuracy: 0.95
Epoch 16/20
19/19 [==============================] - 0s 12ms/step - loss: 0.1680 - accuracy: 0.93
Epoch 17/20
19/19 [==============================] - 0s 12ms/step - loss: 0.1268 - accuracy: 0.94
Epoch 18/20
19/19 [==============================] - 0s 13ms/step - loss: 0.1571 - accuracy: 0.94
Epoch 19/20
19/19 [==============================] - 0s 13ms/step - loss: 0.0976 - accuracy: 0.97
Epoch 20/20
19/19 [==============================] - 0s 13ms/step - loss: 0.0808 - accuracy: 0.96
```

```
def plotLearningCurve(history,epochs):
```

```
    epochRange = range(1,epochs+1)
    plt.plot(epochRange,history.history['accuracy'])
    plt.plot(epochRange,history.history['val_accuracy'])
    plt.title('Model Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend(['Train','Validation'],loc='upper left')
    plt.show()

    plt.plot(epochRange,history.history['loss'])
    plt.plot(epochRange,history.history['val_loss'])
    plt.title('Model Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend(['Train','Validation'],loc='upper left')
    plt.show()


plotLearningCurve(history,20)
```