In [1]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Flatten, Dense, Dropout, BatchNormalizatic
from tensorflow.keras.layers import Conv1D, MaxPooling1D
from keras.layers import SpatialDropout1D
from tensorflow.keras.optimizers import Adam
print(tf.__version__)
```

```
2.11.0
```

In [41]:
```
import pandas as pd
import numpy as np
import  seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [42]:
```
data = pd.read_csv('creditcard.csv')
data.head()
```

Out[42]:

	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
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0

5 rows × 31 columns

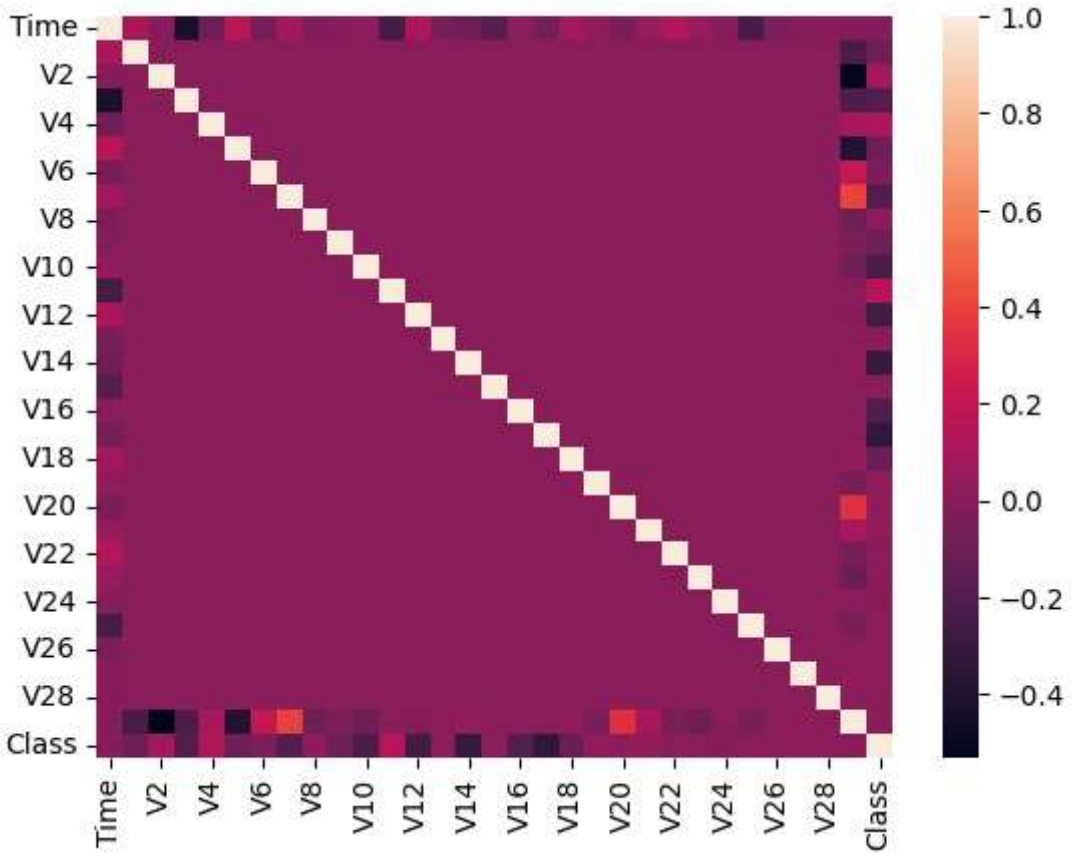
In [5]:
```
data.shape
```

Out[5]:
```
(284807, 31)
```

In [6]:
```
data.isnull().sum()
```

Out[6]:
```
Time      0
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
Amount    0
Class     0
dtype: int64
```

In [8]:
```
dataplot=sns.heatmap(data.corr())
plt.show()
```



In [44]:
```
data['Class'].value_counts()
```

Out[44]:
```
0    284315
1       492
Name: Class, dtype: int64
```

In [45]:
```
non_fraud = data[data['Class']==0]
fraud = data[data['Class']==1]
```

In [46]:
```
non_fraud.shape, fraud.shape
```

Out[46]:
```
((284315, 31), (492, 31))
```

In [47]:
```
non_fraud = non_fraud.sample(fraud.shape[0])
non_fraud.shape
```

Out[47]:
```
(492, 31)
```

In [48]:
```
data = fraud.append(non_fraud, ignore_index=True)
data
```

```
C:\Users\Administrator\AppData\Local\Temp\ipykernel_7648\96343452.py:1: Futur
eWarning: The frame.append method is deprecated and will be removed from pand
as in a future version. Use pandas.concat instead.
  data = fraud.append(non_fraud, ignore_index=True)
```

Out[48]:

	Time	V1	V2	V3	V4	V5	V6	V7	
0	406.0	-2.312227	1.951992	-1.609851	3.997906	-0.522188	-1.426545	-2.537387	1.391
1	472.0	-3.043541	-3.157307	1.088463	2.288644	1.359805	-1.064823	0.325574	-0.067
2	4462.0	-2.303350	1.759247	-0.359745	2.330243	-0.821628	-0.075788	0.562320	-0.399
3	6986.0	-4.397974	1.358367	-2.592844	2.679787	-1.128131	-1.706536	-3.496197	-0.248
4	7519.0	1.234235	3.019740	-4.304597	4.732795	3.624201	-1.357746	1.713445	-0.496
...	...	...	...	...	...	...	...	...	
979	125205.0	0.080544	0.744861	-0.029200	-0.743787	0.664837	-0.666655	0.905852	-0.067
980	79826.0	1.242763	0.056145	0.045949	0.109549	-0.435421	-1.330599	0.307449	-0.283
981	73745.0	-0.670076	1.037461	-0.678171	-1.056501	2.244517	3.200812	-0.241885	1.363
982	132947.0	-1.143580	1.173592	1.107021	-0.453706	-0.646007	-0.120689	0.002402	0.470
983	75904.0	1.104687	0.043815	0.970525	1.418752	-0.849291	-0.659503	-0.127395	-0.042

984 rows × 31 columns

In [49]:
```
data['Class'].value_counts()
```

Out[49]:
```
1    492
0    492
Name: Class, dtype: int64
```

In [50]:
```
X = data.drop('Class', axis = 1)
y = data['Class']
```

In [51]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, rar
```

In [52]:
```
X_train.shape, X_test.shape
```

Out[52]:
```
((787, 30), (197, 30))
```

In [53]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [54]:
```
y_train = y_train.to_numpy()
y_test = y_test.to_numpy()
```

In [65]:
```
X_train.shape
```

Out[65]:
```
(787, 30, 1)
```

In [66]:
```
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)
```

In [67]:
```
X_train.shape, X_test.shape
```

Out[67]:
```
((787, 30, 1), (197, 30, 1))
```

CNN MODEL

In [68]:
```
epochs = 20
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'))
```

In [69]:
```
model.summary()
```

```
Model: "sequential_18"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_14 (Conv1D)          (None, 29, 32)            96

 batch_normalization_5 (Batc  (None, 29, 32)           128
 hNormalization)

 dropout_16 (Dropout)        (None, 29, 32)            0

 conv1d_15 (Conv1D)          (None, 28, 64)            4160

 batch_normalization_6 (Batc  (None, 28, 64)           256
 hNormalization)

 dropout_17 (Dropout)        (None, 28, 64)            0

 flatten_2 (Flatten)         (None, 1792)              0

 dense_24 (Dense)            (None, 64)                114752

 dropout_18 (Dropout)        (None, 64)                0

 dense_25 (Dense)            (None, 1)                 65

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

In [70]:
```
model.compile(optimizer=Adam(lr=0.0001), loss = 'binary_crossentropy', metrics
```

```
WARNING:absl:`lr` is deprecated, please use `learning_rate` instead, or use t
he legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
```

In [71]:
```
history = model.fit(X_train, y_train, epochs=epochs, validation_data=(X_test,
```

```
Epoch 1/20
25/25 [==============================] - 2s 15ms/step - loss: 0.5397 - accura
cy: 0.8475 - val_loss: 0.5386 - val_accuracy: 0.5330
Epoch 2/20
25/25 [==============================] - 0s 8ms/step - loss: 0.2522 - accurac
y: 0.9199 - val_loss: 0.5274 - val_accuracy: 0.5025
Epoch 3/20
25/25 [==============================] - 0s 8ms/step - loss: 0.2320 - accurac
y: 0.9276 - val_loss: 0.5684 - val_accuracy: 0.4975
Epoch 4/20
25/25 [==============================] - 0s 8ms/step - loss: 0.1492 - accurac
y: 0.9339 - val_loss: 0.6860 - val_accuracy: 0.4975
Epoch 5/20
25/25 [==============================] - 0s 8ms/step - loss: 0.1779 - accurac
y: 0.9352 - val_loss: 0.7443 - val_accuracy: 0.4975
Epoch 6/20
25/25 [==============================] - 0s 8ms/step - loss: 0.1560 - accurac
y: 0.9504 - val_loss: 0.6206 - val_accuracy: 0.4975
Epoch 7/20
25/25 [==============================] - 0s 8ms/step - loss: 0.1346 - accurac
y: 0.9466 - val_loss: 0.7656 - val_accuracy: 0.4975
Epoch 8/20
25/25 [==============================] - 0s 7ms/step - loss: 0.1454 - accurac
y: 0.9555 - val_loss: 0.6463 - val_accuracy: 0.5076
Epoch 9/20
25/25 [==============================] - 0s 7ms/step - loss: 0.1426 - accurac
y: 0.9479 - val_loss: 0.7070 - val_accuracy: 0.5127
Epoch 10/20
25/25 [==============================] - 0s 7ms/step - loss: 0.1302 - accurac
y: 0.9555 - val_loss: 0.4579 - val_accuracy: 0.7005
Epoch 11/20
25/25 [==============================] - 0s 7ms/step - loss: 0.1426 - accurac
y: 0.9504 - val_loss: 0.5595 - val_accuracy: 0.6041
Epoch 12/20
25/25 [==============================] - 0s 7ms/step - loss: 0.1171 - accurac
y: 0.9517 - val_loss: 0.3684 - val_accuracy: 0.8173
Epoch 13/20
25/25 [==============================] - 0s 8ms/step - loss: 0.1274 - accurac
y: 0.9581 - val_loss: 0.2945 - val_accuracy: 0.8680
Epoch 14/20
25/25 [==============================] - 0s 10ms/step - loss: 0.1134 - accura
cy: 0.9492 - val_loss: 0.2776 - val_accuracy: 0.8883
Epoch 15/20
25/25 [==============================] - 0s 8ms/step - loss: 0.1178 - accurac
y: 0.9670 - val_loss: 0.2099 - val_accuracy: 0.9289
Epoch 16/20
25/25 [==============================] - 0s 8ms/step - loss: 0.1209 - accurac
y: 0.9606 - val_loss: 0.1825 - val_accuracy: 0.9239
Epoch 17/20
25/25 [==============================] - 0s 8ms/step - loss: 0.1097 - accurac
y: 0.9606 - val_loss: 0.1928 - val_accuracy: 0.9289
Epoch 18/20
25/25 [==============================] - 0s 8ms/step - loss: 0.1118 - accurac
y: 0.9606 - val_loss: 0.1852 - val_accuracy: 0.9340
Epoch 19/20
25/25 [==============================] - 0s 8ms/step - loss: 0.0978 - accurac
y: 0.9657 - val_loss: 0.1660 - val_accuracy: 0.9239
```

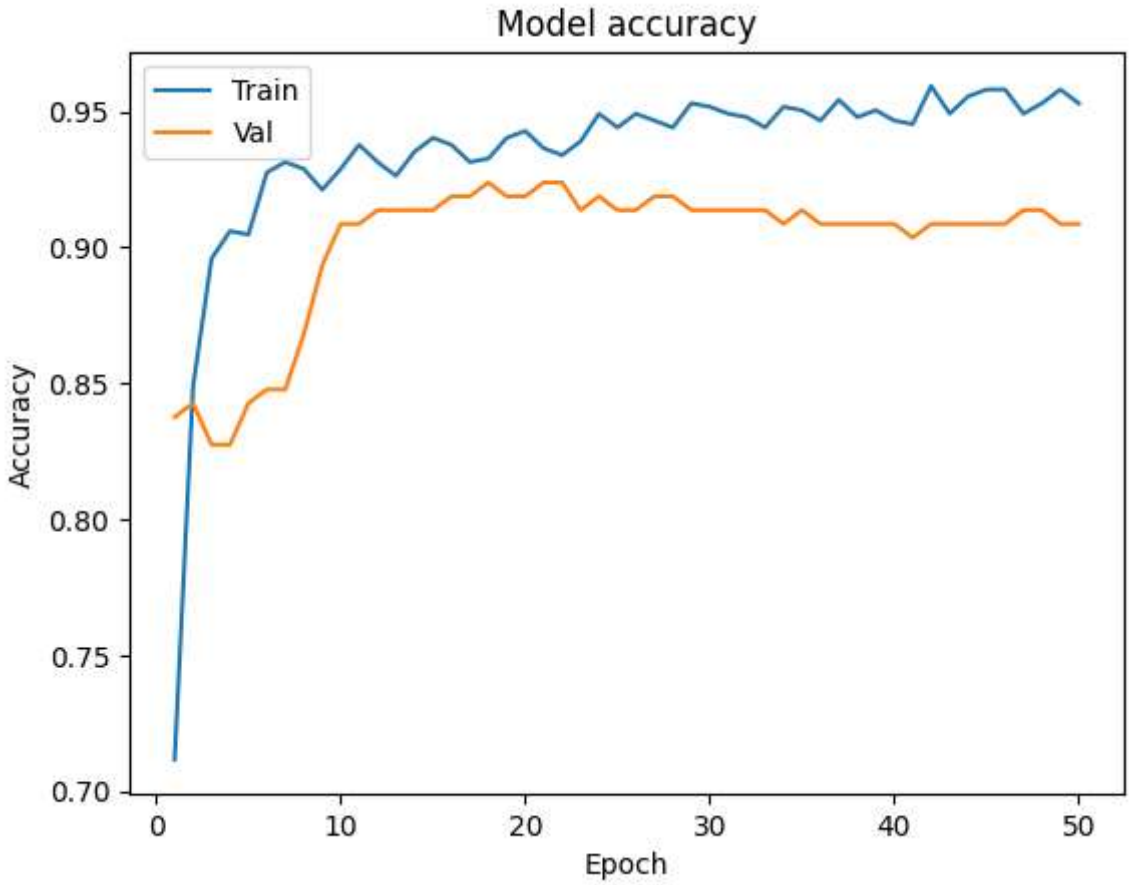
```
Epoch 20/20
25/25 [==============================] - 0s 8ms/step - loss: 0.1006 - accurac
y: 0.9619 - val_loss: 0.1658 - val_accuracy: 0.9239
```

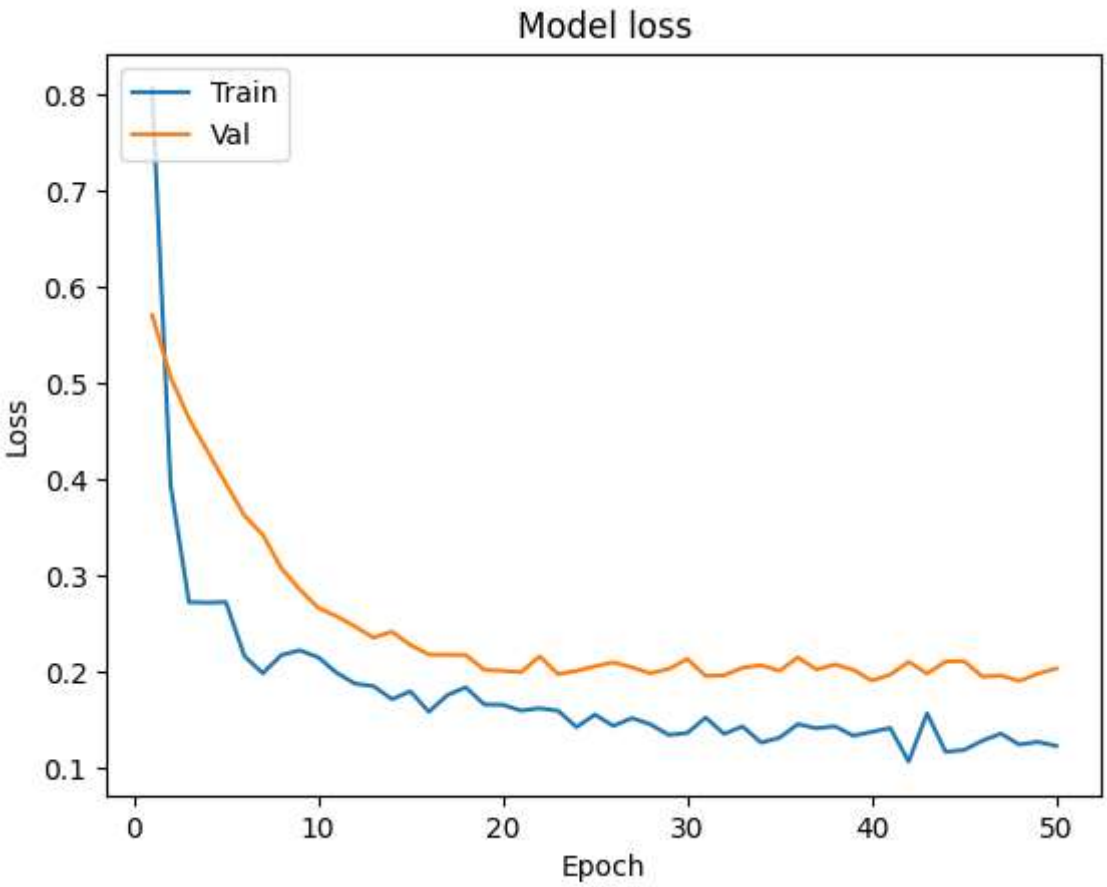
In [72]:
```
model = Sequential()
model.add(Conv1D(16, 2, activation='relu', input_shape = X_train[0].shape))

model.add(Dense(16, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(16, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_19"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_16 (Conv1D)          (None, 29, 16)            48

 dense_26 (Dense)            (None, 29, 16)            272

 dropout_19 (Dropout)        (None, 29, 16)            0

 dense_27 (Dense)            (None, 29, 16)            272

=================================================================
Total params: 592
Trainable params: 592
Non-trainable params: 0
_________________________________________________________________
```

In [24]:
```
def plot_learningCurve(history, epoch):
  # Plot training & validation accuracy values
  epoch_range = range(1, epoch+1)
  plt.plot(epoch_range, history.history['accuracy'])
  plt.plot(epoch_range, history.history['val_accuracy'])
  plt.title('Model accuracy')
  plt.ylabel('Accuracy')
  plt.xlabel('Epoch')
  plt.legend(['Train', 'Val'], loc='upper left')
  plt.show()

  # Plot training & validation loss values
  plt.plot(epoch_range, history.history['loss'])
  plt.plot(epoch_range, history.history['val_loss'])
  plt.title('Model loss')
  plt.ylabel('Loss')
  plt.xlabel('Epoch')
  plt.legend(['Train', 'Val'], loc='upper left')
  plt.show()
```

In [25]:
```
plot_learningCurve(history, epochs)
```





Adding MaxPool Layer

In [26]:
```
epochs = 50
model = Sequential()
model.add(Conv1D(32, 2, activation='relu', input_shape = X_train[0].shape))
model.add(BatchNormalization())
model.add(MaxPool1D(2))
model.add(Dropout(0.2))

model.add(Conv1D(64, 2, activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool1D(2))
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.compile(optimizer=Adam(lr=0.0001), loss = 'binary_crossentropy', metrics
history = model.fit(X_train, y_train, epochs=epochs, validation_data=(X_test,
plot_learningCurve(history, epochs)
```

```
WARNING:absl:`lr` is deprecated, please use `learning_rate` instead, or us
e the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.

Epoch 1/50
25/25 [==============================] - 4s 30ms/step - loss: 0.5877 - acc
uracy: 0.7408 - val_loss: 0.5620 - val_accuracy: 0.8477
Epoch 2/50
25/25 [==============================] - 0s 11ms/step - loss: 0.3311 - acc
uracy: 0.8704 - val_loss: 0.5066 - val_accuracy: 0.8629
Epoch 3/50
25/25 [==============================] - 0s 13ms/step - loss: 0.3050 - acc
uracy: 0.8844 - val_loss: 0.4616 - val_accuracy: 0.8883
Epoch 4/50
25/25 [==============================] - 0s 13ms/step - loss: 0.2693 - acc
uracy: 0.9136 - val_loss: 0.4165 - val_accuracy: 0.9036
Epoch 5/50
25/25 [==============================] - 0s 13ms/step - loss: 0.2259 - acc
uracy: 0.9174 - val_loss: 0.3729 - val_accuracy: 0.9086
Epoch 6/50
25/25 [==============================] - 0s 13ms/step - loss: 0.2303 - acc
```

Architecture of 14 Layers

In [27]:
```
epochs = 100
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(100, activation='relu'))
model.add(Dense(50, activation='relu'))
model.add(Dense(25, activation='relu'))


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

In [28]:
```
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_6 (Conv1D)           (None, 29, 32)            96

 batch_normalization_6 (Batc  (None, 29, 32)           128
 hNormalization)

 dropout_9 (Dropout)         (None, 29, 32)            0

 conv1d_7 (Conv1D)           (None, 28, 64)            4160

 batch_normalization_7 (Batc  (None, 28, 64)           256
 hNormalization)

 dropout_10 (Dropout)        (None, 28, 64)            0

 flatten_3 (Flatten)         (None, 1792)              0

 dense_6 (Dense)             (None, 64)                114752

 dropout_11 (Dropout)        (None, 64)                0

 dense_7 (Dense)             (None, 100)               6500

 dense_8 (Dense)             (None, 50)                5050

 dense_9 (Dense)             (None, 25)                1275

 dense_10 (Dense)            (None, 1)                 26

=================================================================
Total params: 132,243
Trainable params: 132,051
Non-trainable params: 192
_________________________________________________________________
```

In [29]:
```
model.compile(optimizer=Adam(lr=0.0001), loss = 'binary_crossentropy', metrics
```

```
WARNING:absl:`lr` is deprecated, please use `learning_rate` instead, or use t
he legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
```

In [30]:
```
history = model.fit(X_train, y_train, epochs=epochs, validation_data=(X_test,
```

```
Epoch 1/100
25/25 [==============================] - 5s 33ms/step - loss: 0.3987 - acc
uracy: 0.8272 - val_loss: 0.5518 - val_accuracy: 0.8985
Epoch 2/100
25/25 [==============================] - 0s 14ms/step - loss: 0.2461 - acc
uracy: 0.9123 - val_loss: 0.4877 - val_accuracy: 0.9340
Epoch 3/100
25/25 [==============================] - 0s 16ms/step - loss: 0.2070 - acc
uracy: 0.9327 - val_loss: 0.4258 - val_accuracy: 0.9188
Epoch 4/100
25/25 [==============================] - 0s 15ms/step - loss: 0.1922 - acc
uracy: 0.9288 - val_loss: 0.4292 - val_accuracy: 0.7614
Epoch 5/100
25/25 [==============================] - 0s 16ms/step - loss: 0.1903 - acc
uracy: 0.9314 - val_loss: 0.3944 - val_accuracy: 0.8883
Epoch 6/100
25/25 [==============================] - 0s 15ms/step - loss: 0.1675 - acc
uracy: 0.9403 - val_loss: 0.3480 - val_accuracy: 0.9188
Epoch 7/100
```

Architecture of 17 Layers

In [47]:
```
epochs = 100
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(Conv1D(64, 2, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.25))

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

model.add(Dense(100, activation='relu'))
model.add(Dense(50,activation='relu'))
model.add(Dense(25,activation='relu'))

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

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_22 (Conv1D)          (None, 29, 32)            96

 batch_normalization_22 (Bat  (None, 29, 32)           128
 chNormalization)

 dropout_29 (Dropout)        (None, 29, 32)            0

 conv1d_23 (Conv1D)          (None, 28, 64)            4160

 batch_normalization_23 (Bat  (None, 28, 64)           256
 chNormalization)

 dropout_30 (Dropout)        (None, 28, 64)            0

 conv1d_24 (Conv1D)          (None, 27, 64)            8256

 batch_normalization_24 (Bat  (None, 27, 64)           256
 chNormalization)

 dropout_31 (Dropout)        (None, 27, 64)            0

 flatten_7 (Flatten)         (None, 1728)              0

 dense_28 (Dense)            (None, 64)                110656

 dropout_32 (Dropout)        (None, 64)                0

 dense_29 (Dense)            (None, 3)                 195

 dense_30 (Dense)            (None, 1)                 4

=================================================================
Total params: 124,007
Trainable params: 123,687
Non-trainable params: 320
_________________________________________________________________
```

In [50]:
```
model.compile(loss = 'binary_crossentropy', metrics=['accuracy'])
```

In [51]:
```
history = model.fit(X_train, y_train, epochs=epochs, validation_data=(X_test,
```

```
Epoch 1/100
25/25 [==============================] - 2s 20ms/step - loss: 0.1232 - acc
uracy: 0.9733 - val_loss: 0.5547 - val_accuracy: 0.9188
Epoch 2/100
25/25 [==============================] - 0s 10ms/step - loss: 0.1233 - acc
uracy: 0.9809 - val_loss: 0.5359 - val_accuracy: 0.9239
Epoch 3/100
25/25 [==============================] - 0s 10ms/step - loss: 0.1172 - acc
uracy: 0.9822 - val_loss: 0.5835 - val_accuracy: 0.9137
Epoch 4/100
25/25 [==============================] - 0s 10ms/step - loss: 0.1287 - acc
uracy: 0.9746 - val_loss: 0.5435 - val_accuracy: 0.9239
Epoch 5/100
25/25 [==============================] - 0s 11ms/step - loss: 0.1063 - acc
uracy: 0.9848 - val_loss: 0.6482 - val_accuracy: 0.9137
Epoch 6/100
25/25 [==============================] - 0s 10ms/step - loss: 0.1308 - acc
uracy: 0.9771 - val_loss: 0.5573 - val_accuracy: 0.9239
Epoch 7/100
```

Architecture of 20 Layers

In [31]:
```
epochs = 100
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(Conv1D(64, 2, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Conv1D(64, 2, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.25))

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

model.add(Dense(100, activation='relu'))
model.add(Dense(50, activation='relu'))
model.add(Dense(25, activation='relu'))

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

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_8 (Conv1D)           (None, 29, 32)            96

 batch_normalization_8 (Batc  (None, 29, 32)           128
 hNormalization)

 dropout_12 (Dropout)        (None, 29, 32)            0

 conv1d_9 (Conv1D)           (None, 28, 64)            4160

 batch_normalization_9 (Batc  (None, 28, 64)           256
 hNormalization)

 dropout_13 (Dropout)        (None, 28, 64)            0

 conv1d_10 (Conv1D)          (None, 27, 64)            8256

 batch_normalization_10 (Bat  (None, 27, 64)           256
 chNormalization)

 dropout_14 (Dropout)        (None, 27, 64)            0

 conv1d_11 (Conv1D)          (None, 26, 64)            8256

 batch_normalization_11 (Bat  (None, 26, 64)           256
 chNormalization)

 dropout_15 (Dropout)        (None, 26, 64)            0

 flatten_4 (Flatten)         (None, 1664)              0

 dense_11 (Dense)            (None, 64)                106560

 dropout_16 (Dropout)        (None, 64)                0

 dense_12 (Dense)            (None, 100)               6500

 dense_13 (Dense)            (None, 50)                5050

 dense_14 (Dense)            (None, 25)                1275

 dense_15 (Dense)            (None, 1)                 26

=================================================================
Total params: 141,075
Trainable params: 140,627
Non-trainable params: 448
_________________________________________________________________
```

In [32]:
```
model.compile(optimizer=Adam(lr=0.0001), loss = 'binary_crossentropy', metrics
```

```
WARNING:absl:`lr` is deprecated, please use `learning_rate` instead, or use t
he legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
```

In [33]:
```
history = model.fit(X_train, y_train, epochs=epochs, validation_data=(X_test,
```

```
Epoch 1/100
25/25 [==============================] - 7s 47ms/step - loss: 0.5069 - acc
uracy: 0.7421 - val_loss: 0.6665 - val_accuracy: 0.5228
Epoch 2/100
25/25 [==============================] - 1s 24ms/step - loss: 0.2991 - acc
uracy: 0.8971 - val_loss: 0.5551 - val_accuracy: 0.8071
Epoch 3/100
25/25 [==============================] - 1s 23ms/step - loss: 0.2367 - acc
uracy: 0.9098 - val_loss: 0.4415 - val_accuracy: 0.9188
Epoch 4/100
25/25 [==============================] - 1s 24ms/step - loss: 0.2317 - acc
uracy: 0.9136 - val_loss: 0.3537 - val_accuracy: 0.9137
Epoch 5/100
25/25 [==============================] - 1s 25ms/step - loss: 0.2033 - acc
uracy: 0.9314 - val_loss: 0.2989 - val_accuracy: 0.9188
Epoch 6/100
25/25 [==============================] - 1s 21ms/step - loss: 0.2117 - acc
uracy: 0.9288 - val_loss: 0.2739 - val_accuracy: 0.9137
Epoch 7/100
```

In [ ]:
```
```