CM2

Note: Important section have **blue** heading

Critical Code blocks to implement models

Model 1: 1 input layer, 2 hidden layers with relu activation and 1 output layer

Building the model

In [38]:

```
model_1 = Sequential()

# Creating the input layer and first hidden layer
model_1.add(Dense(24, input_dim=X_train.shape[1], activation='relu'))

#Adding second hidden layer
model_1.add(Dense(24, activation='relu'))

#Adding the output layer
model_1.add(Dense(3, activation='softmax'))

#Compiling the ANN
model_1.compile(optimizer='adam',
                loss='SparseCategoricalCrossentropy',
                metrics=['accuracy'])
```

Training the model

In [ ]:

```
%%time
#Fitting the classifier to the training set
history = model_1.fit(X_train, y_train,
                      validation_data = (X_val, y_val),
                      batch_size=64,
                      epochs=100)
```

Model 2: 1 input layer, 2 hidden layers with sigmoid activation and 1 output layer

Building the model

In [46]:

```
model_2 = Sequential()

# Creating the input layer and first hidden layer
model_2.add(Dense(24, input_dim=X_train.shape[1], activation='sigmoid'))

#Adding second hidden layer
model_2.add(Dense(24, activation='sigmoid'))

#Adding the output layer
model_2.add(Dense(3, activation='softmax'))

#Compiling the ANN
model_2.compile(optimizer='adam',
                loss='SparseCategoricalCrossentropy',
                metrics=['accuracy'])
```

Training the model

In [ ]:

```
%%time
#Fitting the classifier to the training set
history = model_2.fit(X_train, y_train,
                      validation_data = (X_val, y_val),
                      batch_size=64,
                      epochs=100)
```

Model 3: Making the network deeper - 1 input layer, 4 hidden layers with relu activation and 1 output layer

Building the model

In [54]:

```
model_3 = Sequential()

# Creating the input layer and first hidden layer
model_3.add(Dense(24, input_dim=X_train.shape[1], activation='relu'))

#Adding second hidden layer
model_3.add(Dense(24, activation='relu'))

#Adding third hidden layer
model_3.add(Dense(24, activation='relu'))

#Adding fourth hidden layer
model_3.add(Dense(24, activation='relu'))

#Adding the output layer
model_3.add(Dense(3, activation='softmax'))

#Compiling the ANN
model_3.compile(optimizer='adam',
                loss='SparseCategoricalCrossentropy',
                metrics=['accuracy'])
```

Training the model

In [ ]:

```
%%time
#Fitting the classifier to the training set
history = model_3.fit(X_train, y_train,
                      validation_data = (X_val, y_val),
                      batch_size=64,
                      epochs=100)
```

Model 4: Making the network deeper - 1 input layer, 4 hidden layers with relu activation and 1 output layer with dropout

Building the model

In [62]:

```
model_4 = Sequential()

# Creating the input layer and first hidden layer
model_4.add(Dense(24, input_dim=X_train.shape[1], activation='relu'))

#Adding second hidden layer
model_4.add(Dense(24, activation='relu'))

model_4.add(Dropout(0.3))

#Adding third hidden layer
model_4.add(Dense(24, activation='relu'))

model_4.add(Dropout(0.3))

#Adding fourth hidden layer
model_4.add(Dense(24, activation='relu'))

#Adding the output layer
model_4.add(Dense(3, activation='softmax'))

#Compiling the ANN
model_4.compile(optimizer='adam',
                loss='SparseCategoricalCrossentropy',
                metrics=['accuracy'])
```

Training the model

In [ ]:

```
%%time
#Fitting the classifier to the training set
history = model_4.fit(X_train, y_train,
                      validation_data = (X_val, y_val),
                      batch_size=64,
                      epochs=100)
```

Model 5: 1 input layer, 1 RNN layer, 2 hidden layers and 1 output layer

Building the model

In [71]:

```
model_5 = Sequential()

# Creating the input layer and RNN layer
model_5.add(SimpleRNN(64, input_shape=(X_train.shape[1], 1)))

#Adding first hidden layer
model_5.add(Dense(32, activation='relu'))

#Adding second hidden layer
model_5.add(Dense(32, activation='relu'))

#Adding the output layer
model_5.add(Dense(3, activation='softmax'))

model_5.compile(optimizer='adam',
                loss='SparseCategoricalCrossentropy',
                metrics=['accuracy'])
```

Training the model

In [ ]:

```
%%time
#Fitting the classifier to the training set
history = model_5.fit(rnn_train, y_train,
                      validation_data = (rnn_val, y_val),
                      batch_size=64,
                      epochs=100)
```

Model 6: 1 input layer, 1 LSTM layer, 2 hidden layers and 1 output layer

Building the model

In [79]:

```
model_6 = Sequential()

# Creating the input layer and LSTM layer
model_6.add(LSTM(64, input_shape=(X_train.shape[1], 1)))

#Adding first hidden layer
model_6.add(Dense(32, activation='relu'))

#Adding second hidden layer
model_6.add(Dense(32, activation='relu'))

#Adding the output layer
model_6.add(Dense(3, activation='softmax'))

model_6.compile(optimizer='adam',
                loss='SparseCategoricalCrossentropy',
                metrics=['accuracy'])
```

Training the model

In [ ]:

```
%%time
#Fitting the classifier to the training set
history = model_6.fit(rnn_train, y_train,
                      validation_data = (rnn_val, y_val),
                      batch_size=64,
                      epochs=100)
```