CM₂

Note: Important section have **blue** heading

#Fitting the classifier to the training set

Creating the input layer and first hidden layer

epochs=100)

with relu activation and 1 output layer

model 3.add(Dense(24, activation='relu'))

model 3.add(Dense(3, activation='softmax'))

metrics=['accuracy'])

Creating the input layer and first hidden layer

Creating the input layer and first hidden layer

model 2.add(Dense(24, activation='sigmoid'))

Critical Code blocks to implement models

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

model 1 = Sequential()

Building the model

```
# 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
```

```
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
```

model 2.add(Dense(24, input dim=X train.shape[1], activation='sigmoid')) #Adding second hidden layer

model 2 = Sequential()

In [46]:

%%time

```
#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
 %%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,
```

Model 3: Making the network deeper - 1 input layer, 4 hidden layers

Building the model

model 3 = Sequential()

#Adding third hidden layer

#Adding the output layer

model 3.compile(optimizer='adam',

#Compiling the ANN

In [54]:

#Adding second hidden layer model 3.add(Dense(24, activation='relu'))

model 3.add(Dense(24, input dim=X train.shape[1], activation='relu'))

loss='SparseCategoricalCrossentropy',

#Adding fourth hidden layer model 3.add(Dense(24, activation='relu'))

```
Training the model
 %%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
```

#Adding second hidden layer model 4.add(Dense(24, activation='relu')) model_4.add(Dropout(0.3))

model 4.add(Dense(24, input dim=X train.shape[1], activation='relu'))

#Adding third hidden layer

Building the model

model 4 = Sequential()

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

loss='SparseCategoricalCrossentropy',

metrics=['accuracy'])

model_6.add(LSTM(64, input_shape=(X_train.shape[1], 1)))

epochs=100)

model 5.add(SimpleRNN(64, input shape=(X train.shape[1], 1)))

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

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

Training the model

Building the model

model_5 = Sequential()

#Adding first hidden layer

Creating the input layer and RNN layer

model_5.add(Dense(32, activation='relu'))

model 5.compile(optimizer='adam',

```
%%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
```

model_6 = Sequential() # Creating the input layer and LSTM layer

```
model_6.add(Dense(32, activation='relu'))
#Adding second hidden layer
                loss='SparseCategoricalCrossentropy',
                metrics=['accuracy'])
                      validation_data = (rnn_val, y_val),
```

#Adding first 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', Training the model %%time #Fitting the classifier to the training set history = model 6.fit(rnn train, y train, batch size=64,