SUBMITTED BY- MAZIZ YAMIN RAHMAN(102008721)

COS30018 - Option B - Task 4: Machine Learning 1

In this task, we undertake the process to refactor the code which will build the machine learning model

Arguments to Create the model of Machine Learning

1. *We need to create a method and list all the arguments which are required to build the model – network type, no.of layers etc. Here, we are building a layer and then add the dropout. Then we will build another layer and again adding the dropout which can be automated. The network type is LSTM.*

```
def create_model(sequence_length, n_features, units, cell, n_layers, dropout,
                 loss, optimizer, bidirectional):
    ...

create_model()
```

2. *We do not want the value to be passed every time we build the model. So, some default parameters have to be defined.*

```
def create_model(sequence_length, n_features, units=256, cell=LSTM, n_layers=2,
dropout=0.3,
                 loss="mean_absolute_error", optimizer="rmsprop", bidirectional=False):
    ...

create_model()
```

BUILDING THE LAYERS

We will now start building the layers and adding it to the model. This is what we did before:

```
1.# First layer
  model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.shape[1], 1)))
  model.add(Dropout(0.2))
  # second layer
  model.add(LSTM(units=50, return_sequences=True))
  model.add(Dropout(0.2))
  # third layer
  model.add(LSTM(units=50))
  model.add(Dropout(0.2))
  model.add(Dense(units=1))  # Prediction of the next closest price      }
```

2.*The code above shows that arguments were passed inside the 'model.add()' were different but the had same statements. So, we use 'if' statements to make is automated.*

```
model = Sequential()
    for i in range(n_layers):
        if i == 0:
            # first layer
            if bidirectional:
```

```
            model.add(Bidirectional(cell(units, return_sequences=True),
batch_input_shape=(None,                                    sequence_length,
n_features)))
        else:
            model.add(cell(units, return_sequences=True, batch_input_shape=(None,
sequence_length,
```

3. *After the first layer is built, we start building the next layer.*
4. *Since all the layers will be of the same type after the first layer so, we do this*

```
elif i == n_layers - 1:
        # last layer
        if bidirectional:
            model.add(Bidirectional(cell(units, return_sequences=False)))
        else:
            model.add(cell(units, return_sequences=False))
```

5. *There will be a hidden layer in the model which will have the argument return_sequences set to true.*

```
else:
        # Hidden layers
        if bidirectional:
            model.add(bidirectional(cell(units, return_sequences=True))
        else:
            model.add(cell(units, return_sequences=True))
```

6. *There will be a dropout layer after every layer and then we will compile.*

```
model.add(Dropout(dropout))
model.compile(optimizer=optimizer, loss=loss)
```

7. *Then we return the model back to the caller. In this way we have automated the model building process using some extra arguments.*
8. *Call the create_model function and fix that model to predict the stock prices.*

```
model = create_model(50, 1)

    model.fit(x_train, y_train, epochs=3, batch_size=32)
```

```
Epoch 1/3
59/59 [==============================] - 6s 31ms/step - loss: 0.0814
Epoch 2/3
59/59 [==============================] - 2s 30ms/step - loss: 0.0137
Epoch 3/3
59/59 [==============================] - 2s 29ms/step - loss: 0.0098
Prediction: [[272.62918 277.91855 263.2683  282.53778 284.4823  284.66907 280.11395
  279.08786 299.38794 279.0322  279.7595  277.38968 256.7931  277.34222
  261.40802 259.29123 276.05197 283.90436 281.59863 268.08954 284.7468
  282.28146 282.17276 288.3153  279.49765 270.25662 287.29953 283.63687
  274.65863 284.77905 284.1058  286.50677 268.20084 261.18234 278.19394
  287.5572  273.8412  286.99182 279.50378 301.32874 278.97485 275.76498
  282.37573 275.6428  277.7693  271.4638  264.56726 276.30392 280.83017
  264.7434 ]]
```


FB
FB
200
150
100
50
2013
2014
2015
2016
2017
2018
2019