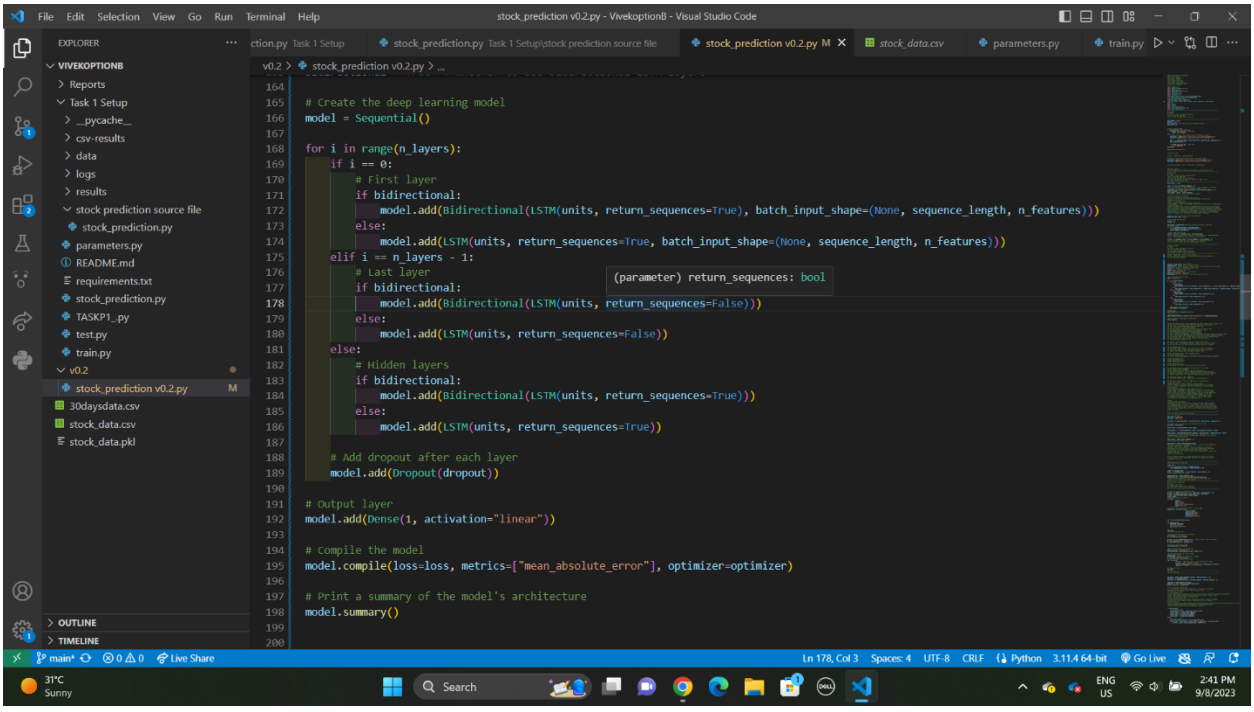
COS30018 - Option B - Task 4: Machine Learning 1

Alright lets start again with this project. I will explain what I have changed in this project to make the new version that is v0.3. I will try to keep it short, to the point and crisp.

Write a function that takes as input several parameters including the number of layers, the size of each layer, the layer name and return a Deep Learning model. Again, our reference project (P1) will give you an example of how this can be done. You can reuse this example, extend it, and most importantly, explain in detail all the code in your program.

For the first step , I have changed the existing model with the one I have created by using the example present in P1.



This LSTM model is built based on user-specified parameters, such as sequence length, number of input features, number of LSTM units in each layer, number of LSTM layers, dropout rate, loss function, optimizer, and whether to use bidirectional LSTM layers.

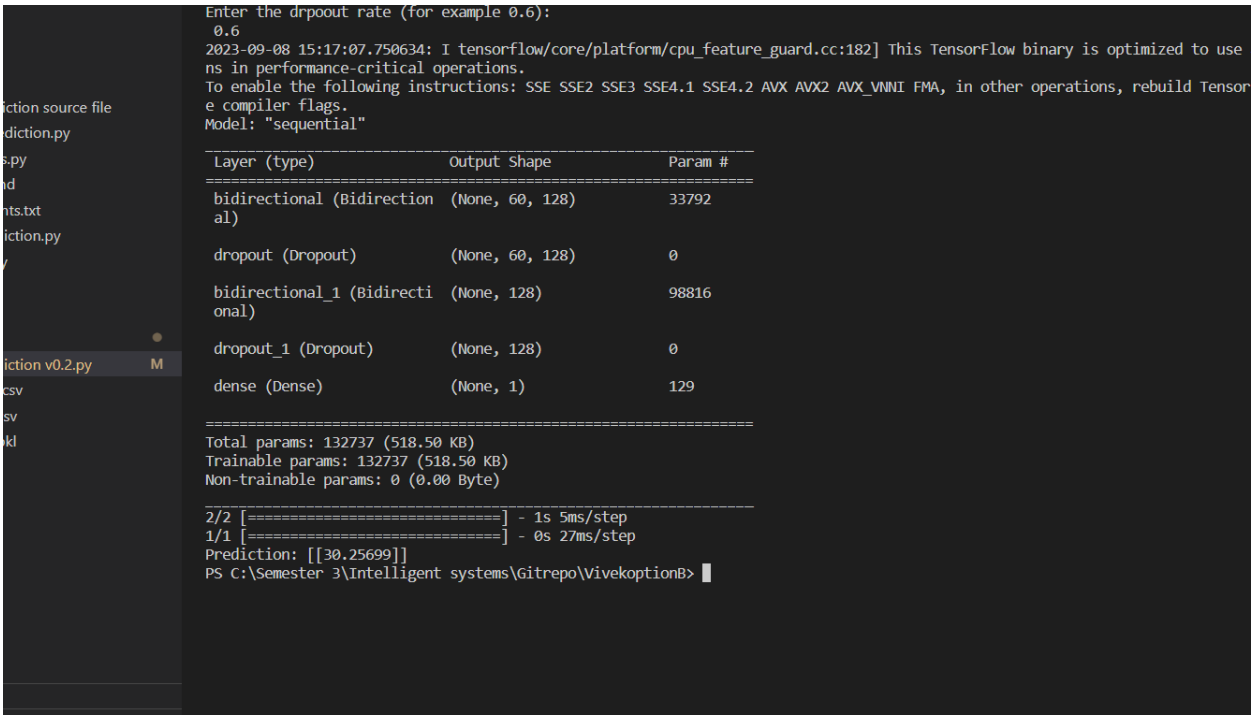
- The sequence_length determines the number of previous days' stock prices to consider as input for predicting the next day's price. This is a hyperparameter that you can adjust.
- The n_features specifies the number of input features. In this case, it's set to 1 since the input is the closing price of the stock.
- The units parameter specifies the number of LSTM units (neurons) in each layer.
- The n_layers parameter specifies how many LSTM layers are stacked in the model.
- The dropout rate is a regularization technique to prevent overfitting by randomly dropping a fraction of neurons during training.
- The loss function is used to measure the model's error during training, and "mean_absolute_error" is chosen as the loss function for this code.
- The optimizer is the optimization algorithm used during training. "rmsprop" is selected as the optimizer.
- The bidirectional parameter determines whether to use bidirectional LSTM layers, which can capture information from both past and future time steps.

The LSTM model is built layer by layer within a loop, where each layer consists of LSTM units followed by dropout layers. The last layer is a dense layer with a linear activation function, responsible for the final prediction

For the information related to the terms I have used info from this website: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/

Now I have used the example and did some tweaks to match my code from the one that is given in P1. I have first used hardcoded values to fill in the hyperparameters and check that whether my code is working or not. After successfully checking the code was working. I changed the code lines so that the parameters takes the user inputs. The errors I faced was with the inputs and had to parse the data into integers and float as required.

Then after executing the code the summary would look like this:



I have also added comments in respectively to tell what line of code does what like how many layers are there and what does it accomplish.

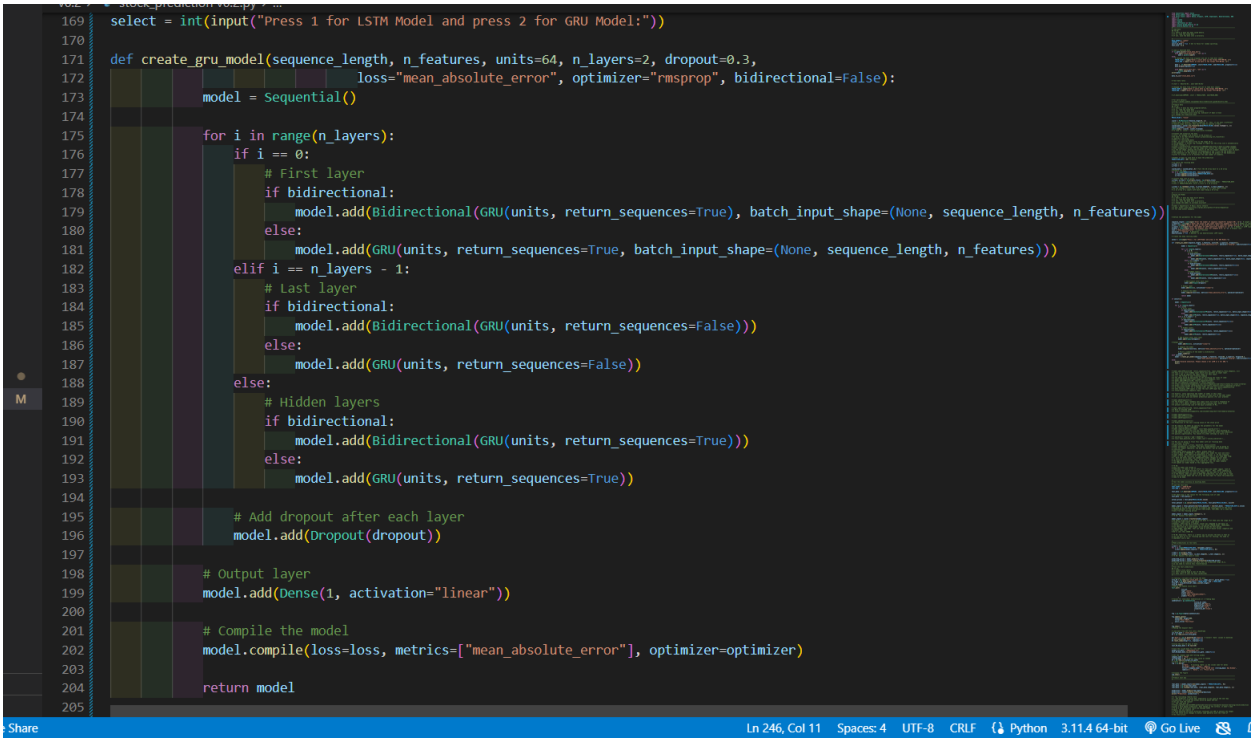
By creating this model, it gives me a good grasp on how these deep learning models actually work and how we can manipulate the input data to make it different. For example , I changed the loss method and fed in different values and experimented. I took the reference to understand the concepts from here : https://builtin.com/data-science/loss-functions-deep-learning-python

Moving on to the Second task:

Use the above function to experiment with different DL networks (e.g., LSTM, RNN, GRU, etc.) and with different hyperparameter configurations (e.g. different numbers of layers and layer sizes, number of epochs, batch sizes, etc.

I was quite unsure about this task, so to make this one I made another function which would add GRU layers and add it to the learning model to get the output. It was really similar to the model that I made , instead of writing LSTM all had to do was replace it with GRU, import the

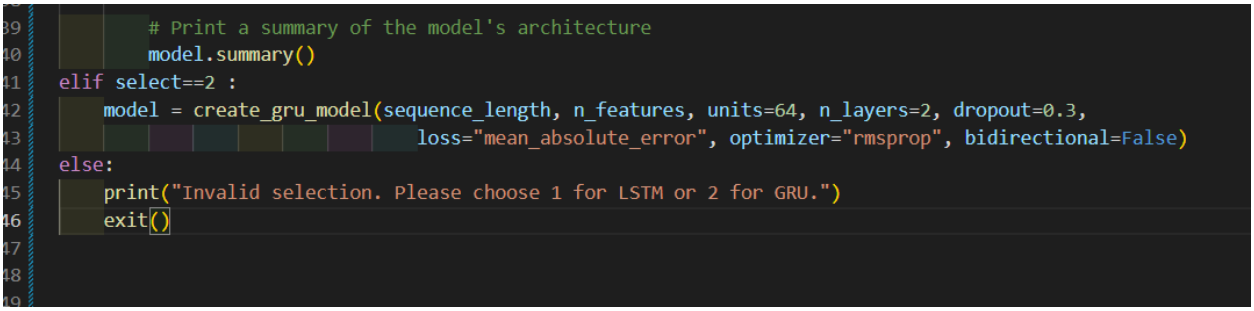
GRU library from keras library. But after this I had some errors in the code due to bad indentation and to make it right I had to give it some time and rewrite that portion where I ahd to define it first and also did a couple of tweaks here and there. I used the same methods top implement the addition of layers similar to the previous model.



```
select = int(input("Press 1 for LSTM Model and press 2 for GRU Model:"))

def create_gru_model(sequence_length, n_features, units=64, n_layers=2, dropout=0.3,
                     loss="mean_absolute_error", optimizer="rmsprop", bidirectional=False):
    model = Sequential()

    for i in range(n_layers):
        if i == 0:
            # First layer
            if bidirectional:
                model.add(Bidirectional(GRU(units, return_sequences=True), batch_input_shape=(None, sequence_length, n_features)))
            else:
                model.add(GRU(units, return_sequences=True, batch_input_shape=(None, sequence_length, n_features)))
        elif i == n_layers - 1:
            # Last layer
            if bidirectional:
                model.add(Bidirectional(GRU(units, return_sequences=False)))
            else:
                model.add(GRU(units, return_sequences=False))
        else:
            # Hidden layers
            if bidirectional:
                model.add(Bidirectional(GRU(units, return_sequences=True)))
            else:
                model.add(GRU(units, return_sequences=True))

        # Add dropout after each layer
        model.add(Dropout(dropout))

    # Output layer
    model.add(Dense(1, activation="linear"))

    # Compile the model
    model.compile(loss=loss, metrics=["mean_absolute_error"], optimizer=optimizer)

    return model
```

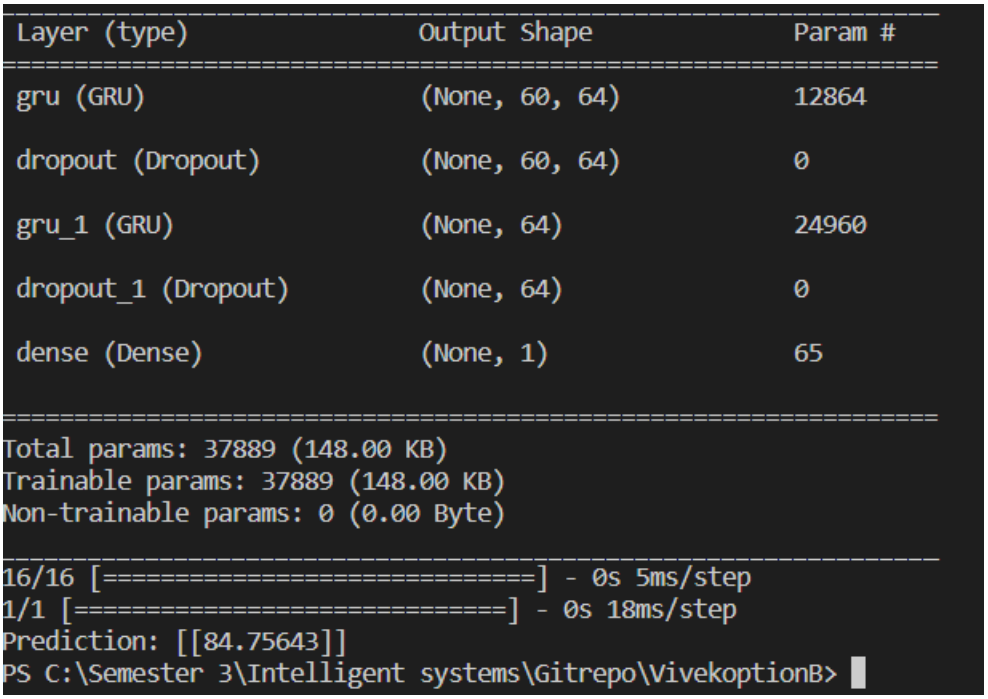
This the GRU model function.

At last, to make it work we have already input the values of hyperparameters . I put up an if else condition to select which DL model we can use.



```
        # Print a summary of the model's architecture
        model.summary()
elif select==2 :
    model = create_gru_model(sequence_length, n_features, units=64, n_layers=2, dropout=0.3,
                             loss="mean_absolute_error", optimizer="rmsprop", bidirectional=False)
else:
    print("Invalid selection. Please choose 1 for LSTM or 2 for GRU.")
    exit()
```

After selecting the options it executes and gives us the summary as well.



I hope this is all required for now that satisfies creating a new deep learning model , testing out with another DL Network and taking the parameters that can be changed by the user via input function.

In conclusion I have learnt about the working of different neural network how we can change the layers and also add different layers to our learning model.

NOTE: I also tried making a separate model with different types of layers like LSTM,GRU, ANN, DROPUTS and tried to merge it into a single one but it was getting way too complex and not working at all so I had to scrap that idea. So in the end , I think this satisfies the requirements and will wait for the feedback :)