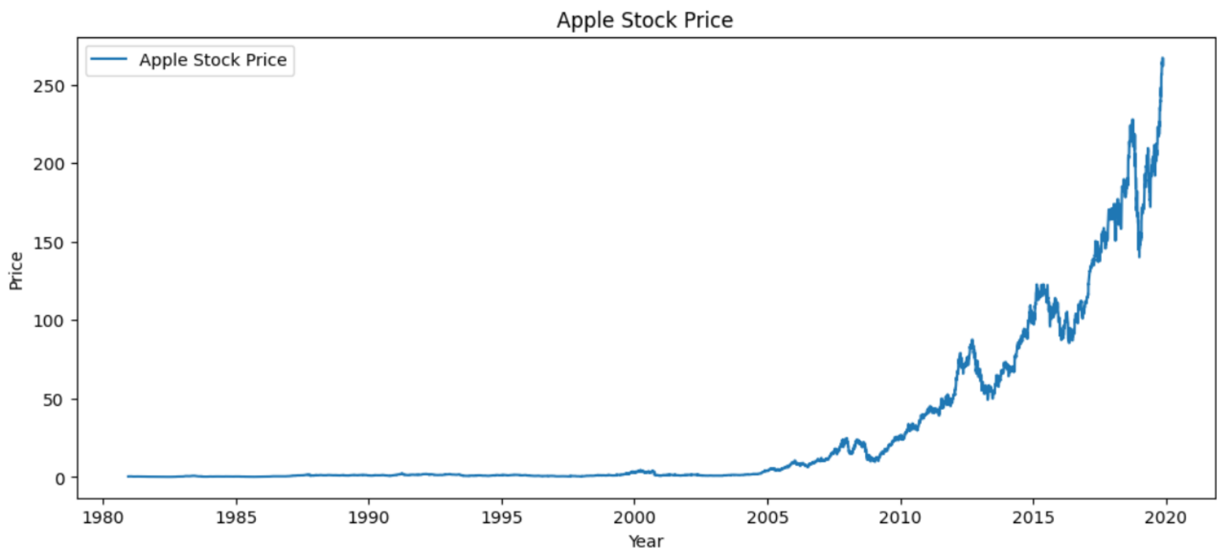
Apple Stock Predictions Using SimpleRNN - Documentation

- Importing necessary libraries: pandas, NumPy, Matplotlib, Scikit-learn and Tensorflow)

Approach-1:

Data-Preprocessing:

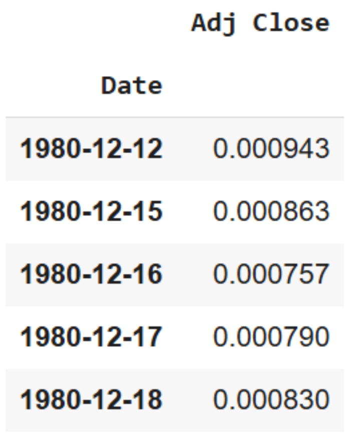
- Loading the dataset into the dataframe.
- Dataset Shape: (9823, 7) --- 9823 rows and 7 columns.
- Getting basic info of the dataset.
- The ‘Date’ column is of object data type, so it is converted into datetime format using pandas.
- For time-series predictions, the ‘Date’ column is set as **index.**
- Checking Null values (1 Null Values).
- Used **forward fill** to handle missing values, since stock prices are sequential and cannot be dropped arbitrarily.
- Plotted the ‘Adj Close’ (**Target variable)** column to analyse the trends and patterns.



I. The stock price remains flat from 1980 till 2005.
II. After 2005 a sharp upward trend growth happens with several peaks and drops.

Scaling the data:

- Using Min-Max scaler, as neural networks works best for scaled values.
- Setting feature range between **0** to **1.** And selected the target variable as 'Adj Close'



	Adj Close
Date	
1980-12-12	0.000943
1980-12-15	0.000863
1980-12-16	0.000757
1980-12-17	0.000790
1980-12-18	0.000830

Creating Time-sequences:

- Defined a function to create sequences.
- Applied that function into the dataset where we selected the **target variable (‘Adj Close’).**
- Used **60 time steps**, where the model learns from the past 60 days to predict the next day's price.
- After applying the shape of X,y variables are,

  X.shape = (9763, 60, 1)

  I. 9763 → Number of samples (data points available after sequence creation)

  II. 60 → Each sample contains 60 past days (time steps) as input

  III. 1 → Only one feature (Adj Close price)

  y.shape = (9763,)

  I. 9763 → The number of target values (corresponding next-day prices)

Train Test Split:

- The training and testing data were split with an 80-20 ratio:

```
Train Shape: (7810, 60, 1), Test Shape: (1953, 60, 1)
```

  I. Total samples before splitting = 9763.

  II. Training set (80%) → 7810 sequences

  III. Testing set (20%) → 1953 sequences

  IV. Each sample contains → 60 past days as input (time_steps = 60)

  V. Features per timestep → 1 (only Adj Close price).

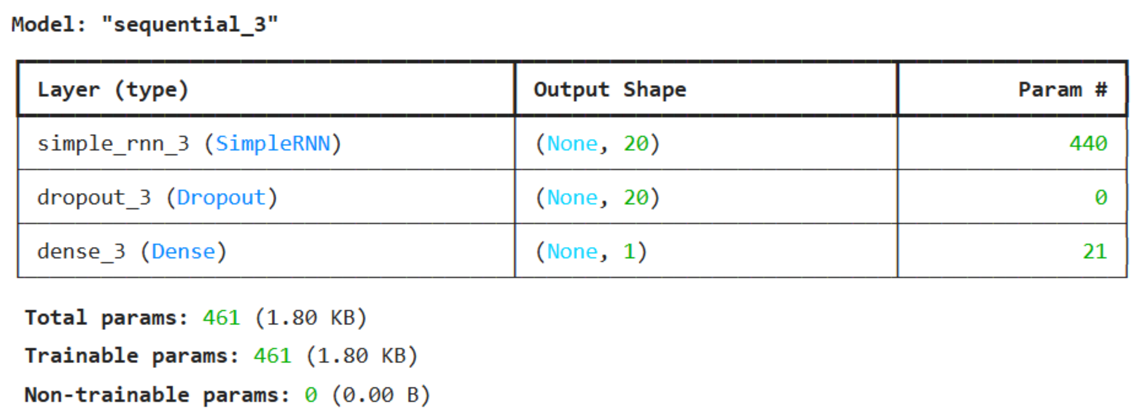
<u>Defining the RNN:</u>

- Since predicting the Apple Stock Prices is Time Series, we are using Sequential with SimpleRNN.
  - I. **20 hidden units** are used in the SimpleRNN layer.
  - II. **ReLU** activation is used to introduce non-linearity and improve training stability.
  - III. **return_sequences=False,** meaning the layer returns only the final output of the sequence.
  - IV. **input_shape** defines the number of time steps and features per step.
  - I. **Dropout=0.4,** where randomly it turns off 40% of neurons during training which helps to prevent overfitting
  - II. **Dense=1,** where it is the output layer which predicts stock price.

<u>Compiling the Model:</u>

- Used the Adam optimizer with a learning rate of 1e-4.
- Loss is Mean Squared Error.

Model Summary:

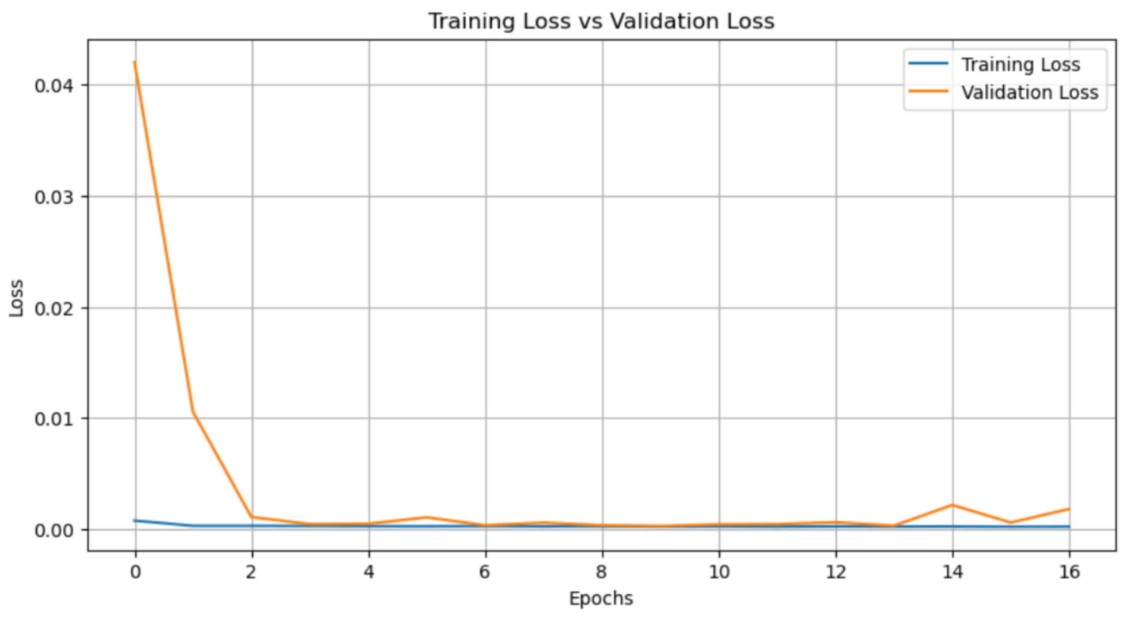


```
Model: "sequential_3"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ simple_rnn_3 (SimpleRNN)             │ (None, 20)                  │             440 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dropout_3 (Dropout)                  │ (None, 20)                  │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_3 (Dense)                      │ (None, 1)                   │              21 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 461 (1.80 KB)
 Trainable params: 461 (1.80 KB)
 Non-trainable params: 0 (0.00 B)
```

Training the Model:

- Initially early stopping and model checkpoints are used, where early stopping prevents unnecessary learning if there is no improvement in val loss.
- Model Checkpoint is used to save the best model.
- Although the model was set to train for 50 epochs, it stopped at 27 due to early stopping.

Training Loss vs Validation Loss:



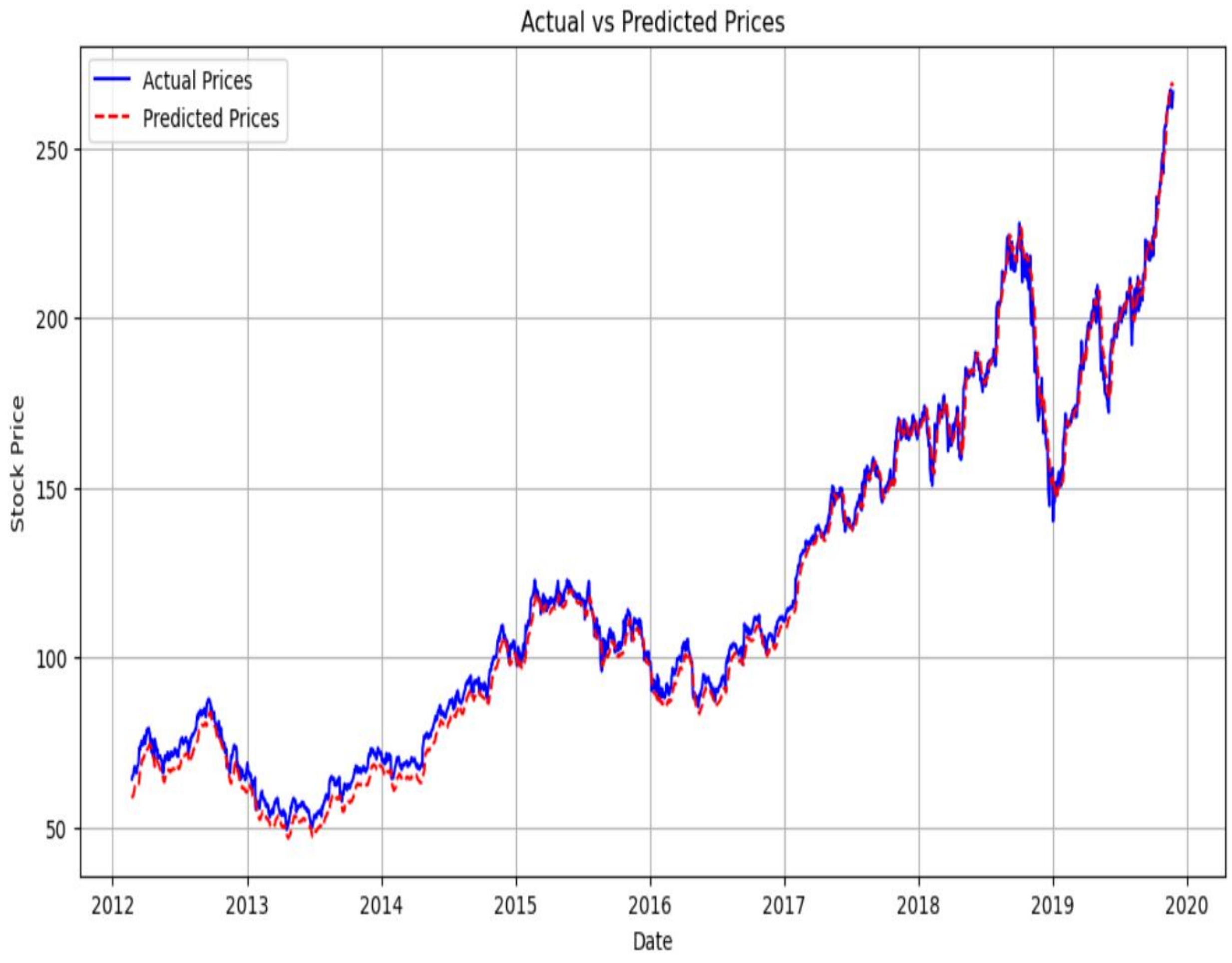
- Validation loss starts around **0.4** and drops down in first few epochs.
- At Epoch 4 the validation loss reaches near zero, indicates the model is fit and learning very well.
- Training and validation loss are closely aligned.

Predictions:

- Predictions are made on test data.
- Applied inverse transformation using MinMaxScaler to convert predictions back to original scale.
- Calculated the Mean Squared Error.

  Mean Squared Error (MSE): 18.370028660377844
- Lower Mean Squared Error indicates lower error, which means the predicted stock prices are closer to the actual stock prices.

Actual vs Predicted Prices Visualization:



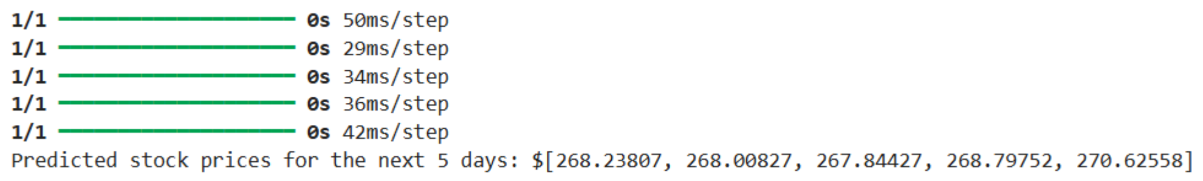
- Predicted stock prices closely follow the actual prices.
- This suggests that SimpleRNN model has learned the trends well.

Predictions of 1,5 and 10 days behavior:

1. Next Day stock price prediction:

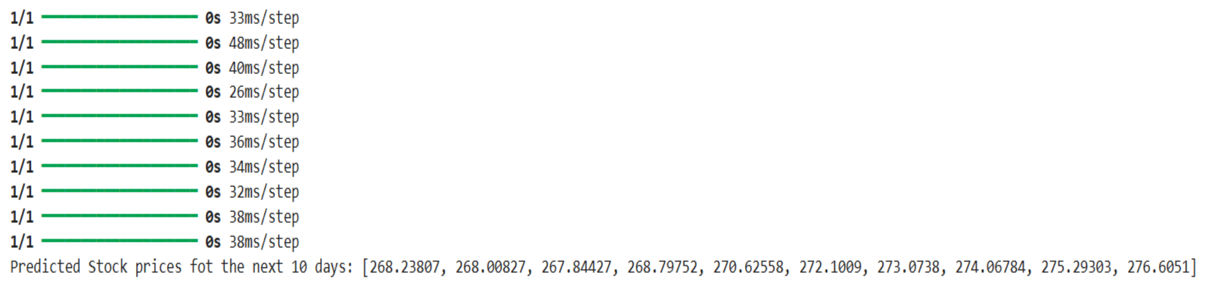
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
Predicted stock price for the next day: $268.23807
```

2. Next 5 days stock price prediction:



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
Predicted stock prices for the next 5 days: $[268.23807, 268.00827, 267.84427, 268.79752, 270.62558]
```

3. Next 10 days stock price predictions:



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
Predicted Stock prices fot the next 10 days: [268.23807, 268.00827, 267.84427, 268.79752, 270.62558, 272.1009, 273.0738, 274.06784, 275.29303, 276.6051]
```

Difficulties Faced:

- Initially only executed with 7 Epochs, then increased the dropout as 0.4, learning rate = 0.0005 and early stopping patience to 7 then executed/learned until 17 Epochs.
- Re-executed the training after a few days; this time, the model trained up to 22 epochs. The loss curves and MSE changed significantly **(MSE = 769.4176).**
- The best-performing model was previously saved as Final_AAPL_SimpleRNN.h5.So opened a new jupyter notebook and done data pre-processing, scaling and create_sequences function and loaded the above mentioned model and predicted the MSE which now we achieved same as previous: **18.37002.**