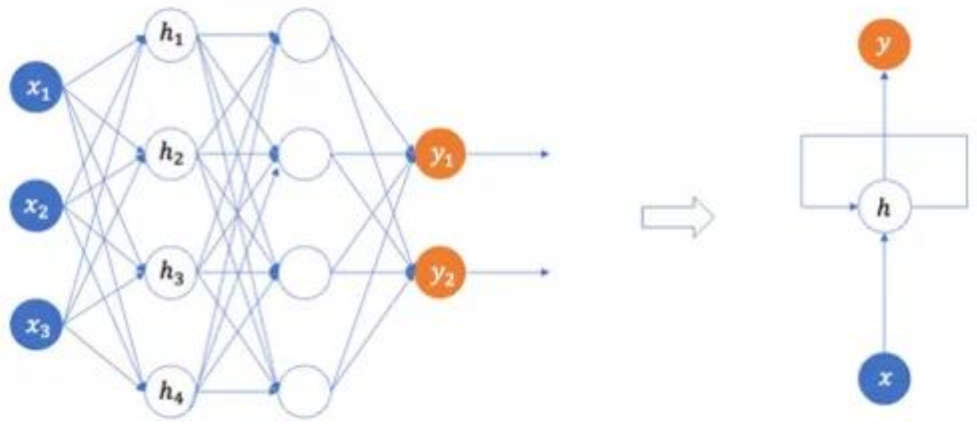
Assignment No: - DL-Group1-4

TITLE	Time series analysis
PROBLEM STATEMENT /DEFINITION	**Recurrent neural network (RNN)** Use the Google stock prices dataset and design a time series analysis and prediction system using RNN.
OBJECTIVE	To build a RNN model to predict the stock price and time analysis .
OUTCOME	To understand the exploratory data analysis,split the training and testing data, Model Evaluation and Prediction by the RNN on the google stock price dataset..
S/W PACKAGES AND HARDWARE/ APPARATUS USED	Jupyter notebook IDE, python3 PC with the configuration as Latest Version of 64 bit Operating Systems, Open Source Fedora-GHz. 8 G.B. RAM, 500 G.B. HDD, 15"Color Monitor, Keyboard, Mouse
REFERENCES	https://www.kaggle.com/code/kandij/google-stock-prediction-using-lstm-keras https://medium.com/analytics-vidhya/share-price-prediction-using-rnn-and-lstm-8776456dea6f
STEPS	**Installing Jupyter notebook with python3** **import the required libraries- numpy,matplotlib.pyplot,pandas,seaborn** **Importing Data (kaggle and scikit-learn library) and Checking out** **Exploratory Data Analysis for stock price Prediction and time series analysis** **Get Data Ready For Training RNN model** **Split Data into Train, Test**
INSTRUCTIONS FOR WRITING JOURNAL	Date Assignment no. Problem definition Learning objective Learning Outcome Concepts related Theory Algorithm Test cases Conclusion/Analysis

Prerequisites: Programming language, machine learning
Concepts related Theory:
Recursive Neural Network (RNN) should be specially designed. RNN translates the provided inputs to machine readable vectors. Then the system processes each of this sequence of vectors one by one, moving from very first vector to the next one in a sequential order. While processing, the system passes the information through the hidden state (memory) to the next step of the sequence. Once the hidden state has collected all the existing information in the system until time period t, it is ready to move towards the next step and in this newer step the hidden step is classified as the previous hidden state defined.The outputs of the previous periods should somewhat become the inputs of the current periods. And the hidden layers will recursively take the inputs of previous periods. The hidden layer receives the inputs from the input layer, and there is a line to connect a hidden layer back to itself to represent the recursive nature.



Training through RNN

1. A single time step of the input is provided to the network.
2. Then calculate its current state using set of current input and the previous state.
3. The current ht becomes ht-1 for the next time step.
4. One can go as many time steps according to the problem and join the information from all the previous states.
5. Once all the time steps are completed the final current state is used to calculate the output.
6. The output is then compared to the actual output i.e the target output and the error is generated.
7. The error is then back-propagated to the network to update the weights and hence the network (RNN) is trained.

Advantages of Recurrent Neural Network

1. An RNN remembers each and every information through time. It is useful in time series prediction only because of the feature to remember previous inputs as well. This is called Long Short Term Memory.
2. Recurrent neural network are even used with convolutional layers to extend the effective

pixel neighborhood.

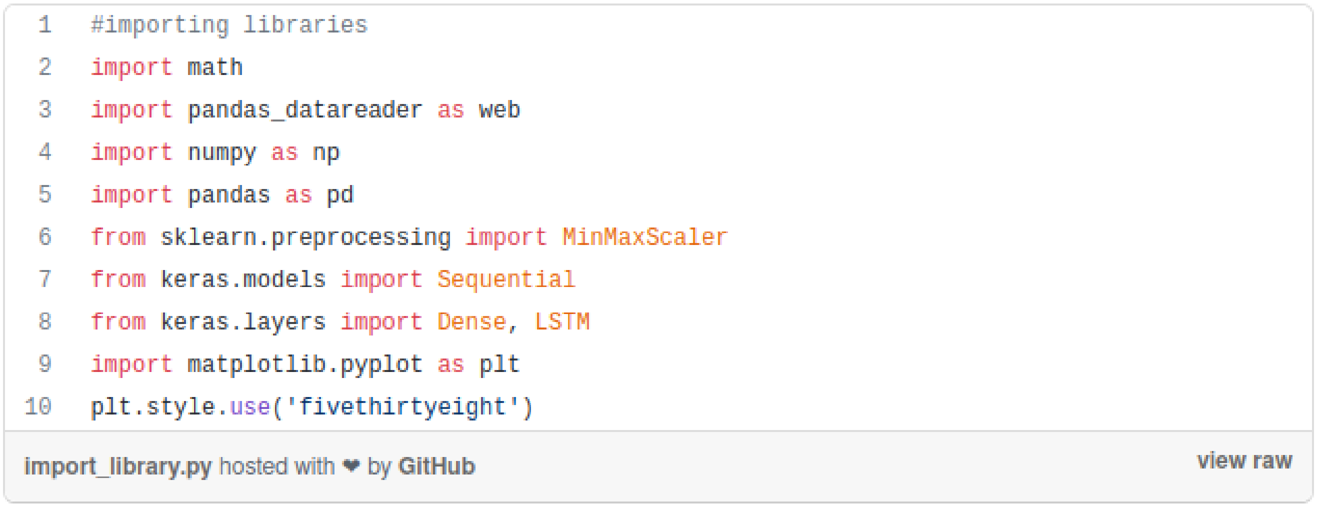
Disadvantages of Recurrent Neural Network

1. Gradient vanishing and exploding problems.
2. Training an RNN is a very difficult task.
3. It cannot process very long sequences if using tanh or relu as an activation function.

For example , we have considered the amazon stock data for prediction and explained in detail in the similar way Google stock prediction should be implemented and plot the graph

Algorithm:

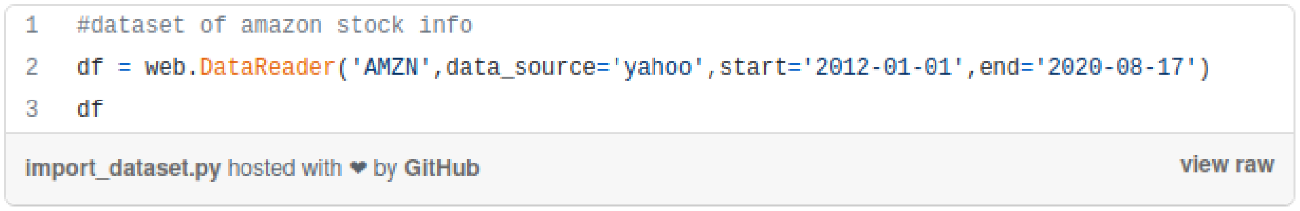
Import Libraries: Install the required libraries and setup for the environment for the assignment. importing SciKit-Learn, Pandas, Seaborn, Matplotlib ,Tensorflow and Numpy.



```python
#importing libraries
import math
import pandas_datareader as web
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
```

import_library.py hosted with ❤ by GitHub view raw

Importing Data and Checking out: As data is in the CSV file, we will read the CSV using pandas read_csv function and check the first 5 rows of the data frame using head().



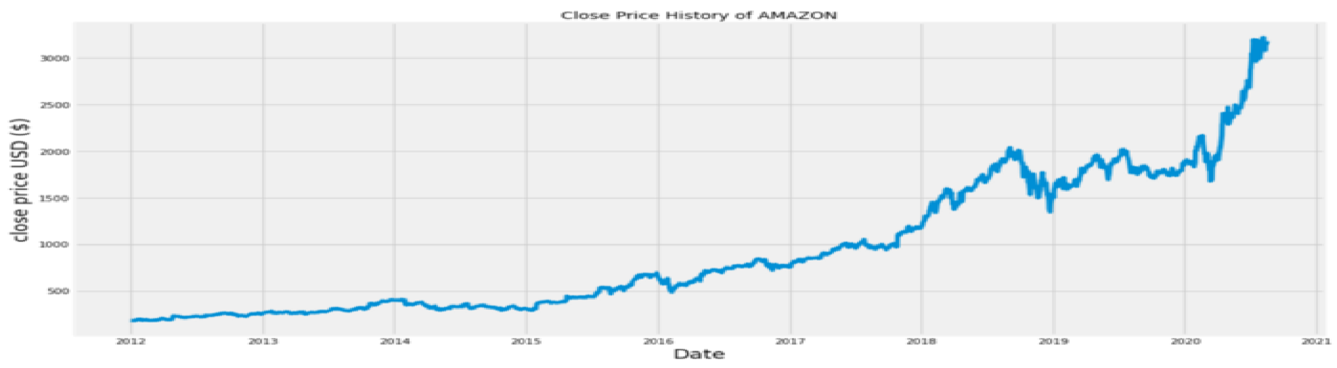
```python
#dataset of amazon stock info
df = web.DataReader('AMZN',data_source='yahoo',start='2012-01-01',end='2020-08-17')
df
```

import_dataset.py hosted with ❤ by GitHub view raw

Visualize the data using matplotlib

```
#Visualize the closing price history of amazon
plt.figure(figsize=(16,8))
plt.title('Close Price History of AMAZON')
plt.plot(df['Close'])
plt.xlabel('Date',fontsize=18)
plt.ylabel('close price USD ($)',fontsize=18)
plt.show()
```

import_visualize.py hosted with ❤ by GitHub view raw

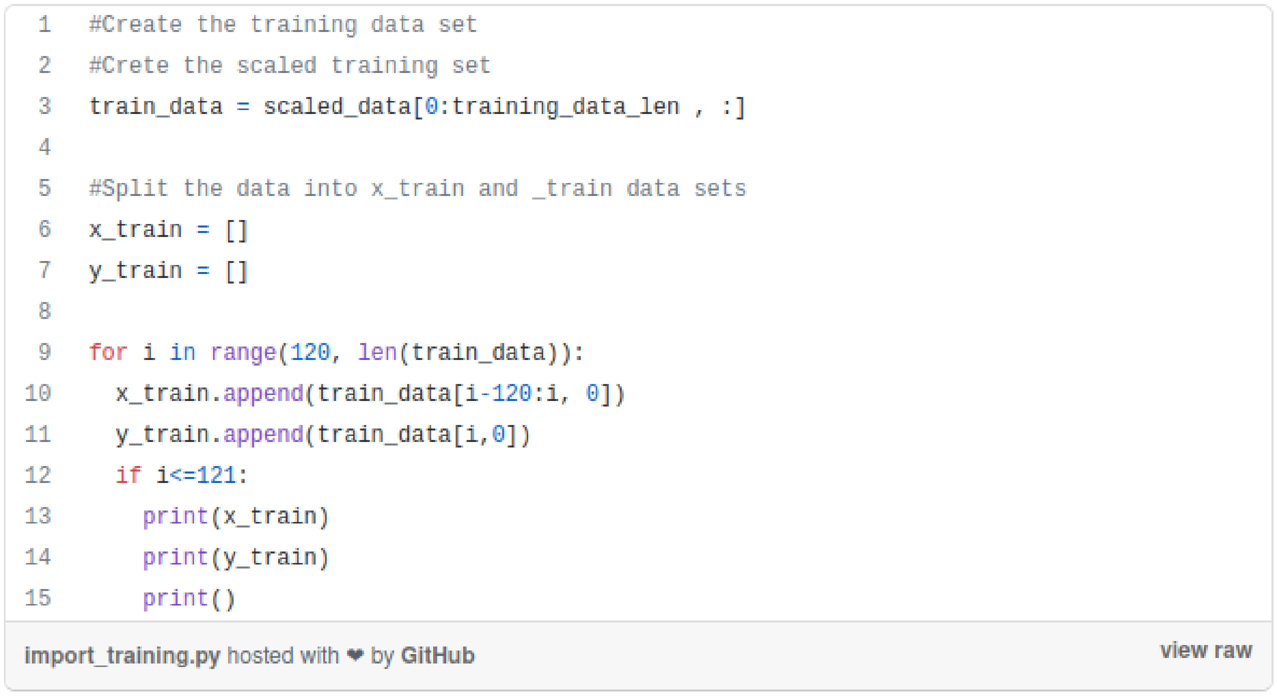


13.Training the data

```
#Create a new data frame with only the 'Close column'
data = df.filter(['Close'])

#Convert the dataframe to numpy array
dataset = data.values

#Get the number of rows to train the model on
training_data_len = math.ceil(len(dataset) * .8 )

training_data_len
```

import_columns.py hosted with ❤ by GitHub view raw

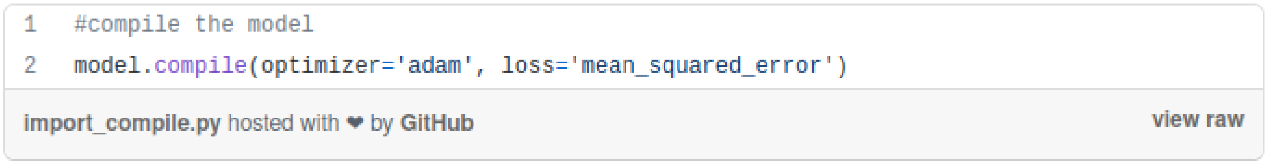
Creating the Training data



```
#Create the training data set
#Crete the scaled training set
train_data = scaled_data[0:training_data_len , :]

#Split the data into x_train and _train data sets
x_train = []
y_train = []

for i in range(120, len(train_data)):
  x_train.append(train_data[i-120:i, 0])
  y_train.append(train_data[i,0])
  if i<=121:
    print(x_train)
    print(y_train)
    print()
```

import_training.py hosted with ❤ by GitHub view raw

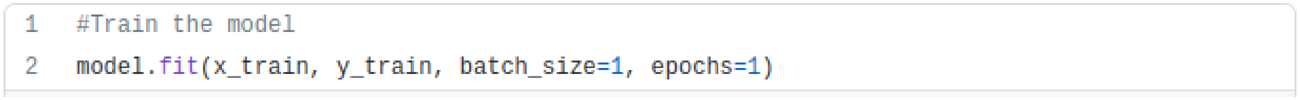
Building the Long Short Term Model



```
#compile the model
model.compile(optimizer='adam', loss='mean_squared_error')
```

import_compile.py hosted with ❤ by GitHub view raw

→Compiling the model



```
#Train the model
model.fit(x_train, y_train, batch_size=1, epochs=1)
```

Training the model

```
#Train the model
model.fit(x_train, y_train, batch_size=1, epochs=1)
```

Creating the test data

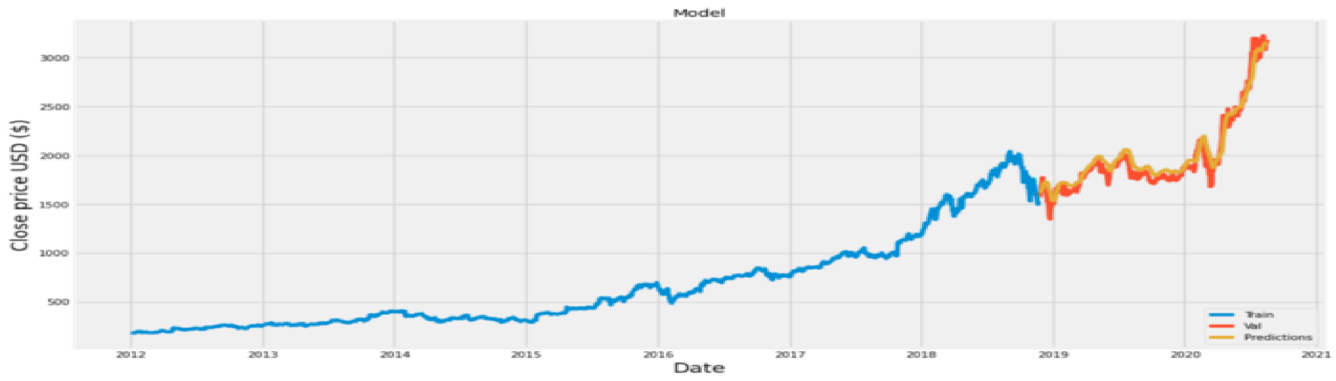
```
#Create the testing data set
#Create a new array containing scaled values from index 1616 to 2170
test_data = scaled_data[training_data_len - 120: , :]

#Create the data sets x_test and y_test
x_test = []
y_test = dataset[training_data_len:, :]
for i in range(120, len(test_data)):
  x_test.append(test_data[i-120:i, 0])
```

To get predicted price values

```
#Get the model predicted price values
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
```

Evaluation and Visualizing the predicted values of the amazon stock prices

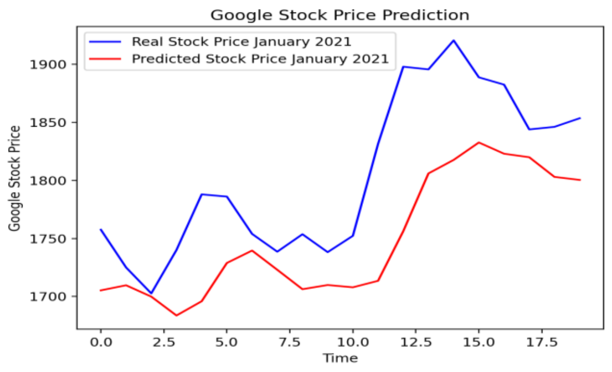
```
#plot the data
train = data[:training_data_len]
valid = data[training_data_len:]
valid['Predictions'] = predictions

#Visualize the data
plt.figure(figsize=(16,8))
plt.title('Model')
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close price USD ($)', fontsize=18)
plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
plt.legend(['Train', 'Val', 'Predictions'], loc='lower right')
plt.show()
```



Conclusion-We have analyzed a RNN Model on amazon stock prediction and in similar way Google stock prediction can be calculated whose prediction model will be as below graph.

Prediction Results

Comparing the real Google stock values and predicted Google stock values that are generated using RNN model, for the test period, the first month of 2021. RNN based on 5 LSTMs was able to properly predict all upward and downward trends as we see that the red line corresponding to the predicted stock prices follows the same pattern as the blue line which corresponds to the real stock prices.



Review Questions:

1. What's the difference between Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) and in which cases would use each one?
2. How many dimensions must the inputs of an RNN layer have? What does each dimension represent? What about its outputs?
3. What are the main difficulties when training RNNs?
4. What are the uses of using RNN in NLP?
5. What's the difference between Traditional Feedforward Networks and Recurrent Neural Networks?
6. How to calculate the output of a Recurrent Neural Network (RNN)?
7. How does LSTM compare to RNN?