SACRAMENTO STATE
COLLEGE OF ENGINEERING & COMPUTER SCIENCE

CSC 215-01 Artificial Intelligence (Spring 2023)

Project 2: Time Series Forecasting using NN, LSTM and CNN

By
Rajvee Modi
Sac State ID - 219712052

Maryam Siddique
Sac State ID – 219485345

Due at 3:00 PM, Monday, March 5th, 2023

Time Series Forecasting using NN, LSTM and CNN

Abstract:
Machine learning models play a significant role in the stock market. These models help us implement features that cleverly predict stock prices for the future. Using past data can help programmers design an AI model to predict the stock price for the upcoming days. In this project, we aim to build machine learning models to predict the stock price for the next day based on the stock price of the past seven days. For this task, we decided to analyze the data for Intel corporation. Our data included 10833 rows and 7 columns regarding the stock values for concurrent dates. Since time series forecasting is a popular field of machine learning, there are various ways to analyze the data and predict values. We will use CNN, LSTM, and NN to predict the stock price for the upcoming day based on last seven days data.

Problem Statement:
In the fast-growing world of social media and technology, it became crucial for businesses and consumers to work ahead of time. In today's world, the primary source of information for most individuals is digital platforms. Hence, people want to stay informed regarding economical factors through online systems. Our goal is to simply forecast the time stamps for stock price prediction in the nearer future. Although time series forecasting has been on the table for a long time, it became one of the fastest growing fields for the search for time series forecasting.

Methodology
We attempted to model our regression problem in several steps. The first step involved processing the data. In data processing, we ensured that the data was fully loaded into the Python notebook and contained zero null rows. By running the isnull() method, our dataset contained zero null values thus, no rows were removed. We further checked our dataset for duplicate data rows by running the duplicated() method and found that there were some duplicate values, so we set those values to false. We also encode the data into a numeric range using the MinMax Scaler. Further, we tested our dataset for categorical features and separated the existing features into discrete and dependent variables. To normalize, we dropped the Date and Adj Close columns as it was not necessary to compare the final outcome of the model. Additionally, we duplicated the Close column to be used for both the input and the output for the model and also tried the min-max scalar to normalize the dataset by applying the lambda function. This normalized the dataset to obtain a cleaner dataset.

Train Test and Split the Dataset
The next step included splitting the dataset to train and test the machine-learning models. We chose our train-test ratio to be 70:30, and 70% of the data was dedicated to training and 30% dedicated to testing our classification models using train_test_split() method. We made sure that each record has 7 * 5 = 35 input features and 1 output feature.

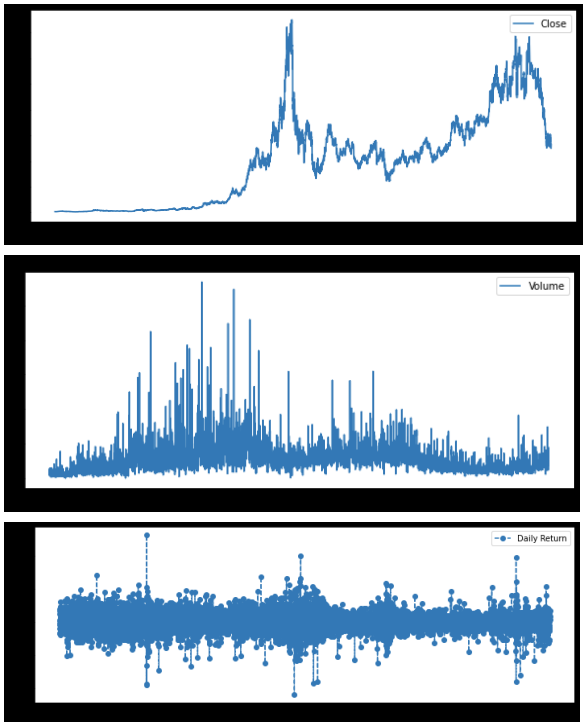
Data Source

The Intel dataset obtained from the resource is used of which 70% is treated as training data and 30% is considered as testing data. The dataset has various attributes used in this stock market. Different configurations of this data set are available with variations

in the number of instances but the number of attributes in each case is the same. The attribute labeled Close tells us the stock price in the data set. For data cleaning, we removed the redundant and missing rows, however. We also checked for missing values to ensure the robustness of our dataset.

Data Analysis

We performed sns plot analysis on the target feature and hist plot analysis on the discrete and continuous features as shown in the figures below.



Machine learning Models

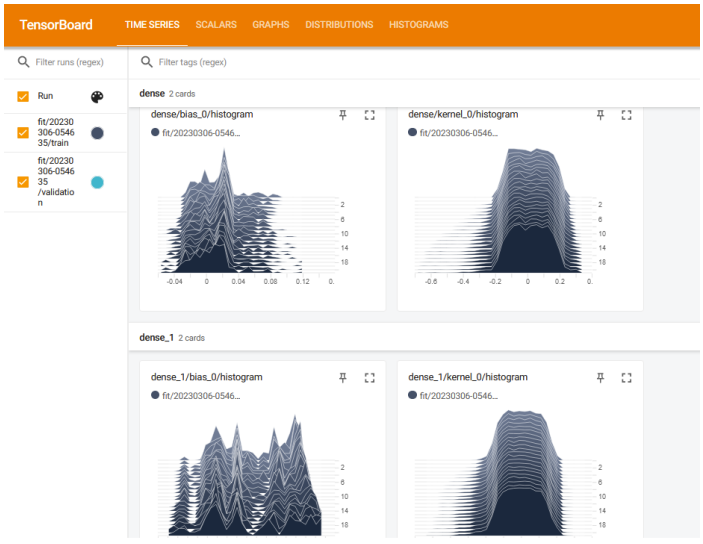
We used three models to evaluate our regression problem. We developed a fully connected neural network (FCNN), LTSM, and a CNN model to evaluate the regression problem. We also included hyperparameter tuning for all three models to optimize the results.

1. **Fully Connected Neural Network**

We implemented a Neural network to obtain the intended calculations. For this model we processed our data and splitted the dataset for training and testing again with a normalized dataset. We used Relu as our activation function for the hidden layer and softmax for our output layer. The number of layers were a multiple of 5 for our model and we used the adam optimizer to minimize the loss in our model. We chose Relu as our activation function because it is used in almost all the convolutional neural networks or deep learning. We further used theTensorboard to visualize the hidden models. We calculated the following parameters using the Tensorflow API.

```
102/102 [==============================] - 0s 1ms/step
Final score (MSE): 0.7470951345651581
Final score (RMSE): 0.8643466518504934
```

The tensorboard allowed us to visualize the epoch loss within each layer as shown in the figure below.



1.1: **FIND EFFICIENT N FOR FCNN MODEL (Additional Feature)**

We further implemented a fully connected neural network to find the best N values for our dataset. For this, we

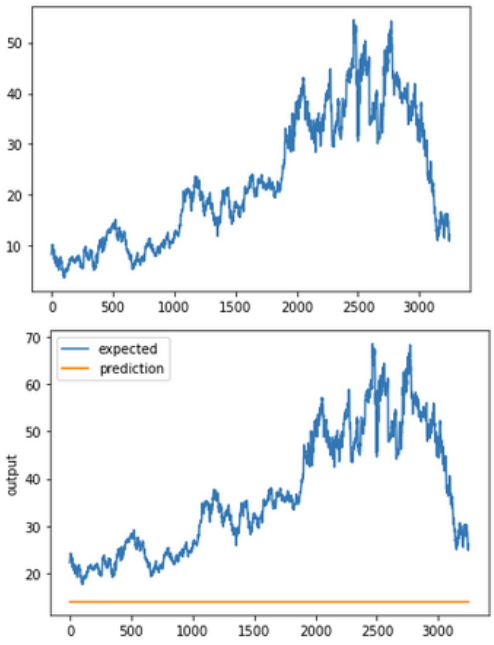
slightly modified our FCNN model to obtain the results as shown in the figure below. For the FCNN model our optimal sequence number is 7.

```
Epoch 28: early stopping
102/102 [==============================] - 0s 1ms/step
Final score (MSE): 0.8204774549784147
Final score (RMSE): 0.9058021058588983
current best rmse: 0.8541889854348671
current best sequence number: 7
Final RMSE: 0.8541889854348671
Optimal Sequence Number: 7
```

1.2: HYPER-PARAMETER TUNING USING OPTIMAL SEQUENCE NUMBER

We then proceeded to perform hyper-parameter tuning for the FCNN model using the best n value obtained in the previous step. Refer to the figure below to see the optimal result after parameterization.

```
=================================================================
Total params: 28,927,996
Trainable params: 28,927,996
Non-trainable params: 0
_________________________________________________________________
None
************************* Prediction in progress ********
102/102 [==============================] - 0s 2ms/step
Pred Shape: (3248, 1)
************************* Prediction Completed *********
Final score (MSE): 678.4543585337412
Final score (RMSE): 26.047156438539336
```

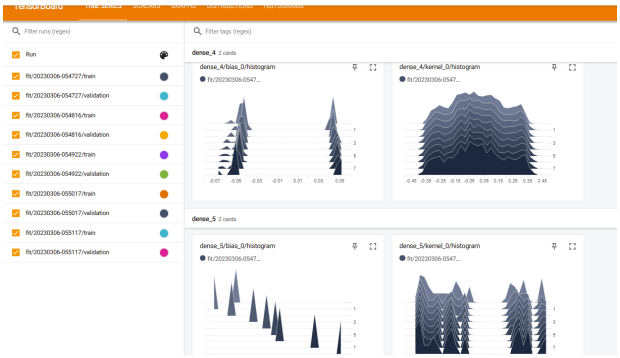


2. LSTM

The LSTM model sees the input dataset as a sequence, so it's able to learn patterns from sequenced data better than the FCNN, it can also learn patterns from long sequences as shown in the model evaluation figure below.

```
102/102 [==============================] - 1s 4ms/step
Final score (MSE): 1.598444217518986
Final score (RMSE): 1.2642959374762643
```

We also used a tensorboard to visualize the epoch loss within each layer as shown in the figure below.



2.1: FIND EFFICIENT N FOR LSTM MODEL (Additional Feature)

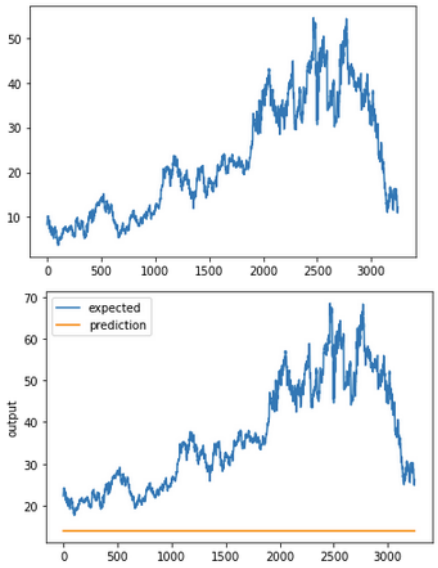
We further implemented a LSTM model to find the best N values for our dataset. For this, we slightly modified our model to obtain the results as shown in the figure below. For the LSTM model our optimal sequence number is also 7.

```
102/102 [==============================] - 1s 5ms/step
Final score (MSE): 1.5191343926974645
Final score (RMSE): 1.2325317004837906
current best rmse: 1.179693815088543
current best sequence number: 7
Final RMSE: 1.179693815088543
Optimal Sequence Number: 7
```

2.2: HYPER-PARAMETER TUNING USING OPTIMAL SEQUENCE NUMBER

We then proceeded to perform hyper-parameter tuning for the LSTM model using the best n value obtained in the previous step. Refer to the figure below to see the optimal result after parameterization.

```
Total params: 315,873,048
Trainable params: 315,873,048
Non-trainable params: 0
_________________________________________________________________
None
************************** Prediction in progress
102/102 [==============================] - 6s 26ms
Pred Shape: (3248, 1)
************************** Prediction Completed **
Final score (MSE): 680.5311817967729
Final score (RMSE): 26.086992578616126
```

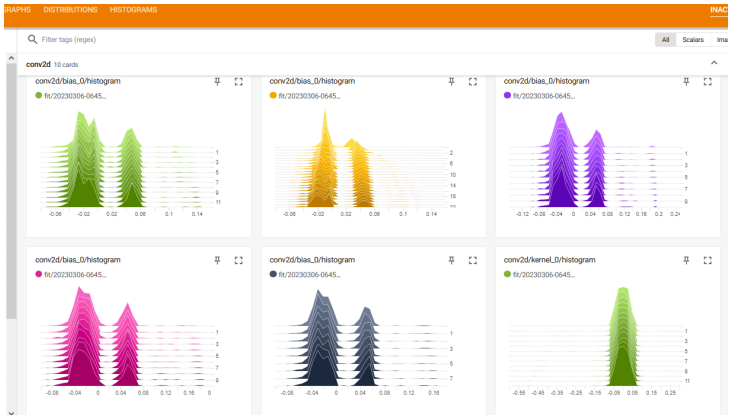


3. CNN Model

For the CNN model we will use one convolutional hidden layer followed by a max pooling layer. The filter maps are then flattened before being interpreted by a Dense layer and outputting a prediction. The convolutional layer identifies patterns between the timesteps. And we also split the dataset into images with 7 pixels and 5 channels to obtain the results as shown in the figure below.

```
Epoch 11: early stopping
102/102 [==============================] - 0s 2ms/step
Final score (MSE): 0.8941354521653577
Final score (RMSE): 0.8941354521653577
```

We also used a tensorboard to visualize the epoch loss within each layer as shown in the figure below.



3.1: FIND EFFICIENT N FOR CNN MODEL (Additional Feature)

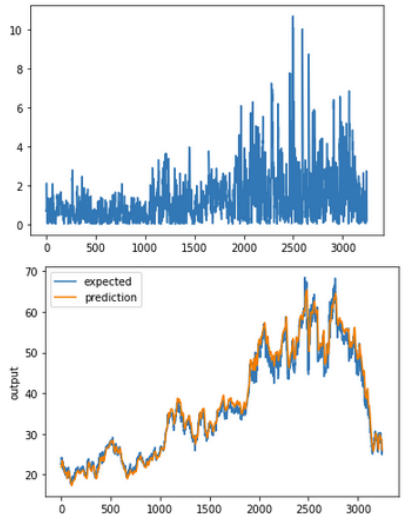
We further implemented a CNN model to find the best N values for our dataset. For this, we slightly modified our model to obtain the results as shown in the figure below. For the LSTM model our optimal sequence number is 11.

```
Final score (MSE): 1.0825999221831495
Final score (RMSE): 1.0825999221831495
current best rmse: 0.9591251891863247
current best sequence number: 11
Final RMSE: 0.9591251891863247
Optimal Sequence Number: 11
```

3.2: HYPER-PARAMETER TUNING USING OPTIMAL SEQUENCE NUMBER

We then proceeded to perform hyper-parameter tuning for the CNN model using the best n value obtained in the previous step. Refer to the figure below to see the optimal result after parameterization.

```
=================================================================
Total params: 3,022,426
Trainable params: 3,022,426
Non-trainable params: 0
_________________________________________________________________
Epoch 1/1000
60/60 - 5s - loss: 71.1181 - val_loss: 28.4629 - 5s/epoch -
Epoch 2/1000
60/60 - 1s - loss: 10.3135 - val_loss: 3.3674 - 554ms/epoch
Epoch 3/1000
60/60 - 0s - loss: 7.2841 - val_loss: 30.4524 - 417ms/epoch
Epoch 4/1000
60/60 - 0s - loss: 6.6058 - val_loss: 6.0412 - 420ms/epoch -
Epoch 5/1000
60/60 - 0s - loss: 5.7822 - val_loss: 3.9807 - 418ms/epoch -
Epoch 6/1000
60/60 - 0s - loss: 5.3921 - val_loss: 11.6306 - 420ms/epoch
Epoch 7/1000
60/60 - 0s - loss: 5.3941 - val_loss: 4.1563 - 429ms/epoch -
Epoch 7: early stopping
102/102 [==============================] - 0s 3ms/step
Final score (MSE): 1.8349736846838682
Final score (RMSE): 1.8349736846838682
```



Second Additional Feature: Multi-Output Regression Feature

Multi-output regression predicts two or more numerical variables. Unlike normal regression where a single value is predicted for each sample, multi-output regression requires specialized machine learning algorithms that support outputting multiple variables for each prediction. Deep learning neural networks are an example of an algorithm that natively supports multi-output regression problems.

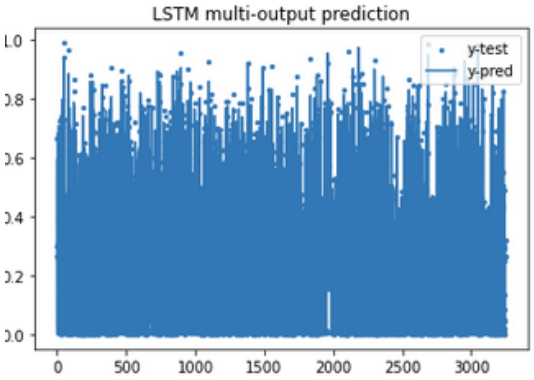
We implemented a similar model using LSTM to predict the values for the next five days. As shown in the figure below the model outputs the MSE score and plots the outcome.

```
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 64)                16896

dense (Dense)                (None, 1)                 65

=================================================================
Total params: 16,961
Trainable params: 16,961
Non-trainable params: 0
_________________________________________________________________
```

```
MSE:0.0003
```



Experimental Results and Analysis:

As mentioned above we divided our project into several sections to evaluate the best model based on the five categories: RMSE and MSE score. Below is a table to indicate the values for all the models.

Model	RMSE	MSE
FCNN	**0.8643466518504934**	**0.7470951345651581**
FCNN N = 7	0.8541889854348671	0.8204774549784147
HPT FCNN	1.013272406206701	1.026720969179917 3
LSTM	1.2642959374762643	1.598444217518986
LSTM N = 7	1.179693815088543	1.519134392697464 5
HPT LSTM	2.5097327905556317	6.298758679990159

CNN	0.8941354521653577	0.8941354521653577
CNN N = 11	0.9591251891863247	1.0825999221831495
HPT CNN	1.8349736846838682	1.8349736846838682

Considering the RMSE and MSE score after the hyper parameter tuning our best model is CNN as it outputs the lowest RMSE and MSE score after hyper parameterization with the optimal sequence obtained in our additional feature

Challenges encountered and their solution:

For this project we faced two major challenges: one was to understand the different combination of layers for each layer needed for optimum results. Another challenge we encountered was to reshape the y vector for the multi-output regression feature.

Learning:

This project provided us with a broad picture of how to begin with pre-processing and data cleaning. The hands-on activities during class lectures helped us map out ideas and put them into action during our project. We learned about various data mining techniques and were able to share this knowledge and improve our project ideas as a team. It was a good learning experience for the team to gain a thorough understanding of the topics covered in class.

Task Division and Project Reflection

Rajvee Modi

1. Analysis of the dataset and figuring out the attributes that are needed to be considered and finalize the approach of how to solve the problem.
2. Implementation of data cleaning and pre-processing.
3. Implementation of normalization, Hyperparameter Tuning
4. Strategy and implementation of finding the best N value for each model
5. Implementation of FCNN, LSTM and CNN

Maryam Siddique:

1. Analysis of the dataset and figuring out the attributes that are needed to be considered and finalize the approach of how to solve the problem.
2. Implementation of data cleaning and pre-processing.
3. Comparison of models implemented.
4. Implementation of Tensorboard.
5. Implementation of the Multi-Output regression feature.