TABLE OF CONTENTS

1. Network Parameters

1.1 Inputs

For the Convultional Neural Network the input is a dataset of financial data from the site "**https://www.nasdaq.com/market-activity/index/spx/historical**". The dataset used contains an overlay of the Schiller P/E for a longer time interval i.e., from 1 Jan 1960 to 31 Dec 2021, onto the S&P500 for the same period. The output from the Convolutional Neural network will be utilized by the Recurrent Neural Network to predict the correct Sell signal.

1.2 Mean Squared Error Loss Function

The Mean Squared Error (MSE), also called L2 Loss, computes the average of the squared differences between actual values and predicted values. MSE is the default loss function for most Pytorch regression problems. The squaring implies that larger mistakes produce even larger errors than smaller ones. The Pytorch L2 Loss is expressed as:

$$\text{loss}(x, y) = (x - y)^2$$

x represents the actual value and y the predicted value.

1.3 Adam Optimizer

Adaptive Moment Estimation is an algorithm for optimization technique for gradient descent. It requires less memory and is efficient. Adam optimizer involves a combination of two gradient descent methodologies: Momentum and Root Mean Square Propagation (RMSP).

$$m_t = \beta m_{t-1} + (1-\beta)\left[\frac{\delta L}{\delta w_t}\right] \qquad v_t = \beta v_{t-1} + (1-\beta) * \left[\frac{\delta L}{\delta w_t}\right]^2$$

Momentum *Root Mean Square Propagation*

1.4 Outputs

The Convolutional Neural Network predicts the ABCD wave pattern in the dataset and outputs it to the Recurrent Neural Network. The output of the CNN+RNN model is the stock prediction for a given day in the given time frame. By using the given dataset we predict the stock price and

calculate the prediction error by comparing the actual price and predicted price. The prediction error ranges from 0 to 1.

2. Python code for Convolutional & Recurrent Neural Network

2.1 Pseudo Code for Convolutional & Recurrent Neural Network follows:

```
# define cnn model
def define_model(learn_rate=0.0001, eps=1e-7, dropout_rate=0.5):
    model = Sequential()
    model.add(Conv2D(16, (3, 3), activation='relu', kernel_initializer='he_uniform', input_shape=(50, 50, 1)))
    model.add(BatchNormalization())
    model.add(MaxPooling2D((2, 2)))
    model.add(Dropout(dropout_rate))
    model.add(Conv2D(16, (3, 3), activation='relu', kernel_initializer='he_uniform'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D((4, 4)))
    model.add(Dropout(dropout_rate))
    model.add(Flatten(name='feat_cnn'))
    model.add(Dense(256, activation='relu', kernel_initializer='he_uniform'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(2, activation='softmax'))
    # compile model
    opt = Adam(learning_rate=learn_rate , epsilon=eps)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, GRU
from tensorflow.keras.optimizers import SGD, Adam
from tensorflow.keras.layers import Dropout
import tensorflow as tf
def set_seed(seed):
    random.seed(seed)
    np.random.seed(seed)
    tf.random.set_seed(seed)
def rnn(shape, lr=0.001, eps=1e-7, dop=0.4):
    set_seed(42)
    model = Sequential()
    model.add(GRU(128, input_shape=shape, return_sequences=True, activation='tanh'))
    model.add(Dropout(dop))
    model.add(GRU(128, activation='tanh'))
    model.add(Dropout(dop))
    model.add(Dense(1))
    opt = Adam(learning_rate=lr , epsilon=eps)
    model.compile(loss='mean_squared_error',  optimizer=opt,metrics = ['mse', 'mae'])
    return model
```

Fig 2.1 Pseudo Code for the convolutional & Recurrent Neural Network

The above is the pseudo code that we are using to implement our convolutional & Recurrent Neural Network. The loss function that we have implemented is the Mean Squared Error. Also we use the Adam Optimizer as it requires less memory and is efficient when working with a

large dataset or parameters. After getting the output from the CNN model we pass it to our RNN model to generate the desired sell point.

3. Training set configuration

3.1 Dataset for Training the Recurrent Neural Network:

- The dataset used for training contains an overlay of the Schiller P/E for a longer time interval i.e., from 1 Jan 1960 to 31 Dec 2021, onto the S&P500 for the same period.
- Below image is a sample of the data that is being used for the training of our CNN.

Date	Close	PE_CAPE
1960-01-04	59.910000	18.338285
1960-01-05	60.389999	18.338285
1960-01-06	60.130001	18.338285
1960-01-07	59.689999	18.338285
1960-01-08	59.500000	18.338285
...	...	...
2021-12-27	4791.190000	38.660000
2021-12-28	4786.350000	38.660000
2021-12-29	4793.060000	38.660000
2021-12-30	4778.730000	38.660000
2021-12-31	4766.180000	38.660000

15606 rows × 2 columns

Fig 3.1: Training data consisting of an overlay of the Schiller P/E onto the S&P500 dataset

3.2 Training the Recurrent Neural Network with dataset:

As you can see in the figure below the training epochs have achieved 0.1826 Mean Squared Error after 10epochs. Hence we train our model for upto 10 epochs in order to ensure that it is neither underfit nor overfit but is in the precise state.

```
Model Defined
Epoch 1/10
```

```
178/178 [==============================] - 5s 5ms/step - loss: 1.4933 - accuracy: 0.7996
Epoch 2/10
178/178 [==============================] - 1s 6ms/step - loss: 0.7435 - accuracy: 0.8941
Epoch 3/10
178/178 [==============================] - 0s 2ms/step - loss: 0.5596 - accuracy: 0.9273
Epoch 4/10
178/178 [==============================] - 0s 2ms/step - loss: 0.4617 - accuracy: 0.9365
Epoch 5/10
178/178 [==============================] - 0s 2ms/step - loss: 0.3261 - accuracy: 0.9460
Epoch 6/10
178/178 [==============================] - 0s 2ms/step - loss: 0.3185 - accuracy: 0.9485
Epoch 7/10
178/178 [==============================] - 0s 2ms/step - loss: 0.2615 - accuracy: 0.9584
Epoch 8/10
178/178 [==============================] - 0s 2ms/step - loss: 0.2446 - accuracy: 0.9548
Epoch 9/10
178/178 [==============================] - 0s 2ms/step - loss: 0.1798 - accuracy: 0.9661
Epoch 10/10
178/178 [==============================] - 0s 2ms/step - loss: 0.1826 - accuracy: 0.9633
Confusion Matrix :-   [[519  15]
 [  3 408]]
```

Fig 3.2: MSE associated with no. of epochs

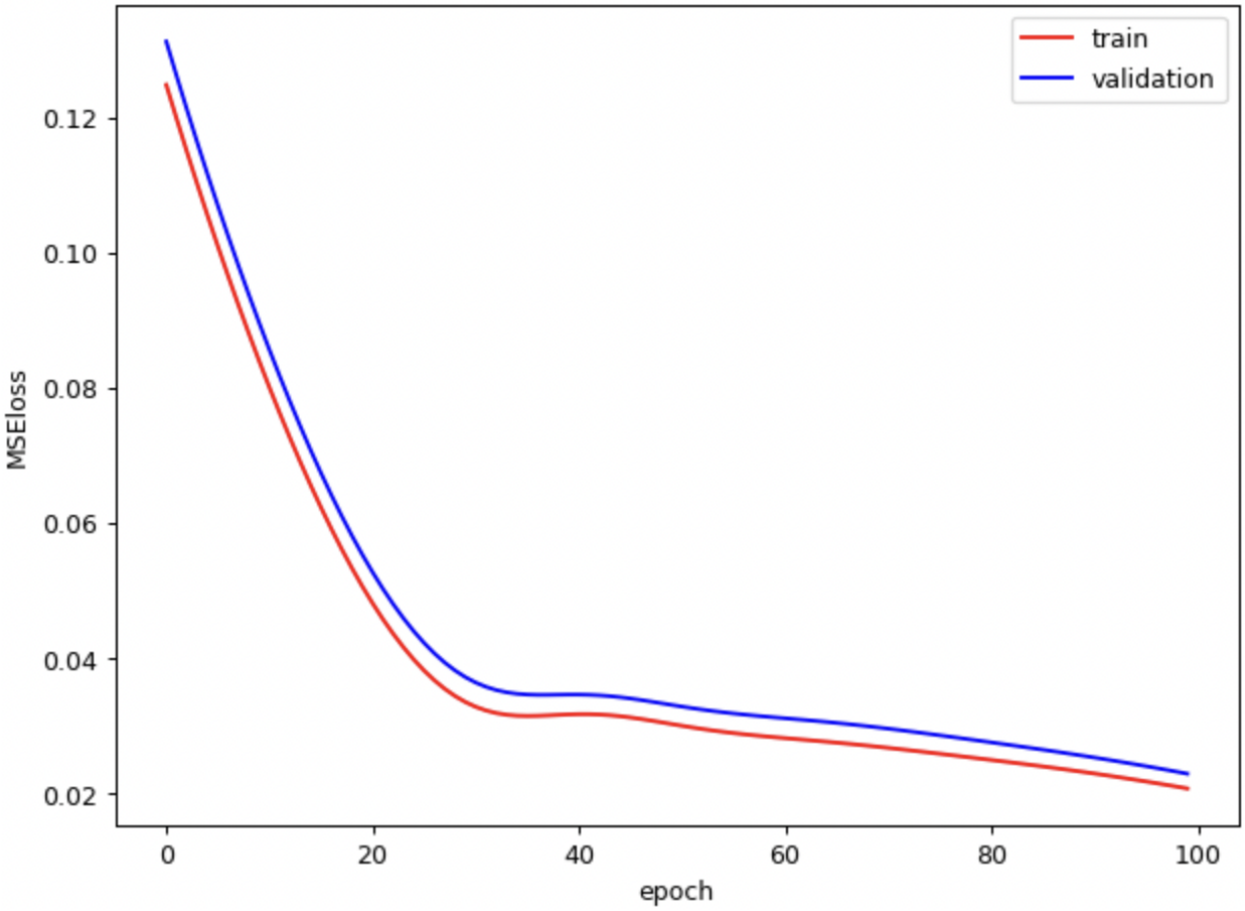

Fig 3.3: MSE vs No. of Epochs Graph for training and validation data

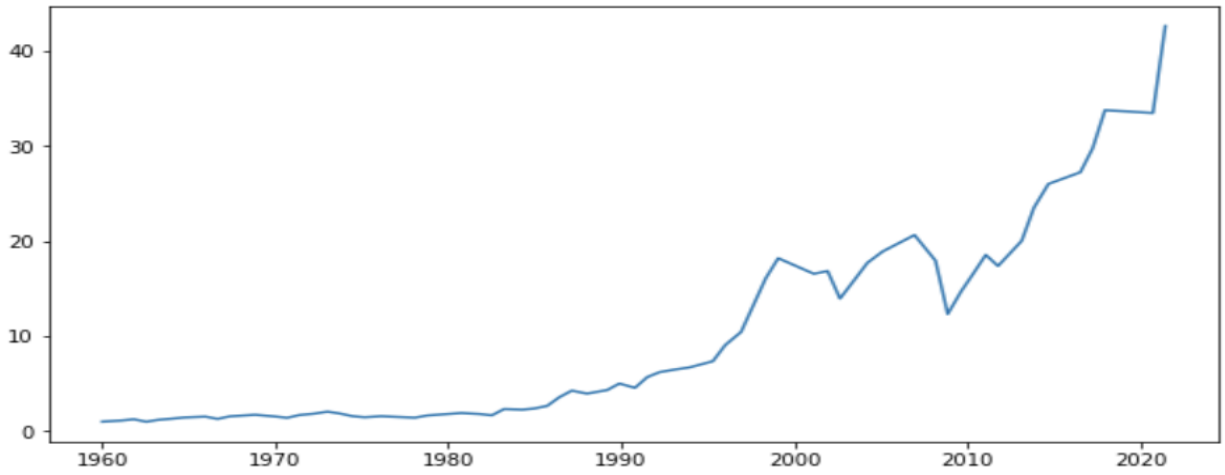

Fig 3.4: Visualizing the results in training dataset

From the plot of MSE loss above, we could see that the MSE of the training set and validation set are quite similar. The loss for the validation set keeps decreasing. Therefore, no overfitting issue is observed. However, based on the prediction plot shown below, there is a significant underfitting issue. We can fix the underfitting issue by the following methods:

- Increase the size or number of parameters in the ML model.
- Increase the complexity or type of the model.
- Increasing the training time until cost function in ML is minimized.

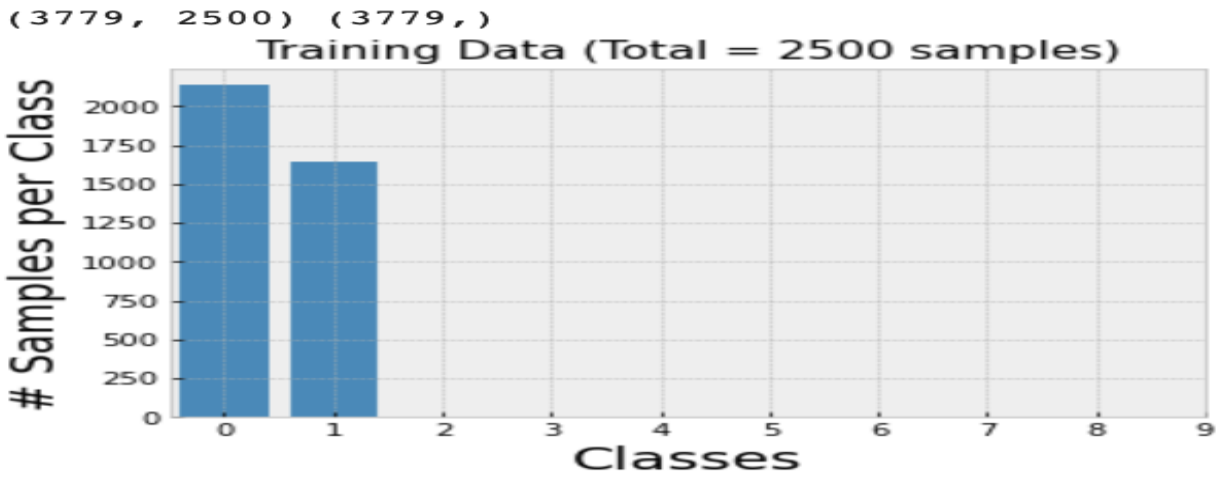

Fig 3.5: Visualizing the samples per classes in training dataset

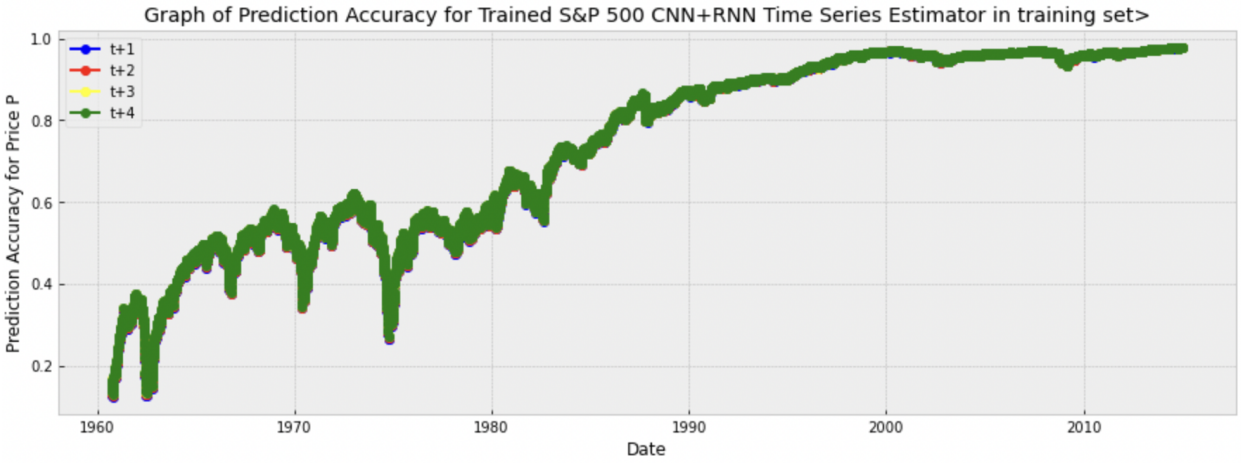

Fig 3.6: Prediction Error Graph as a function of timestamp for Training dataset

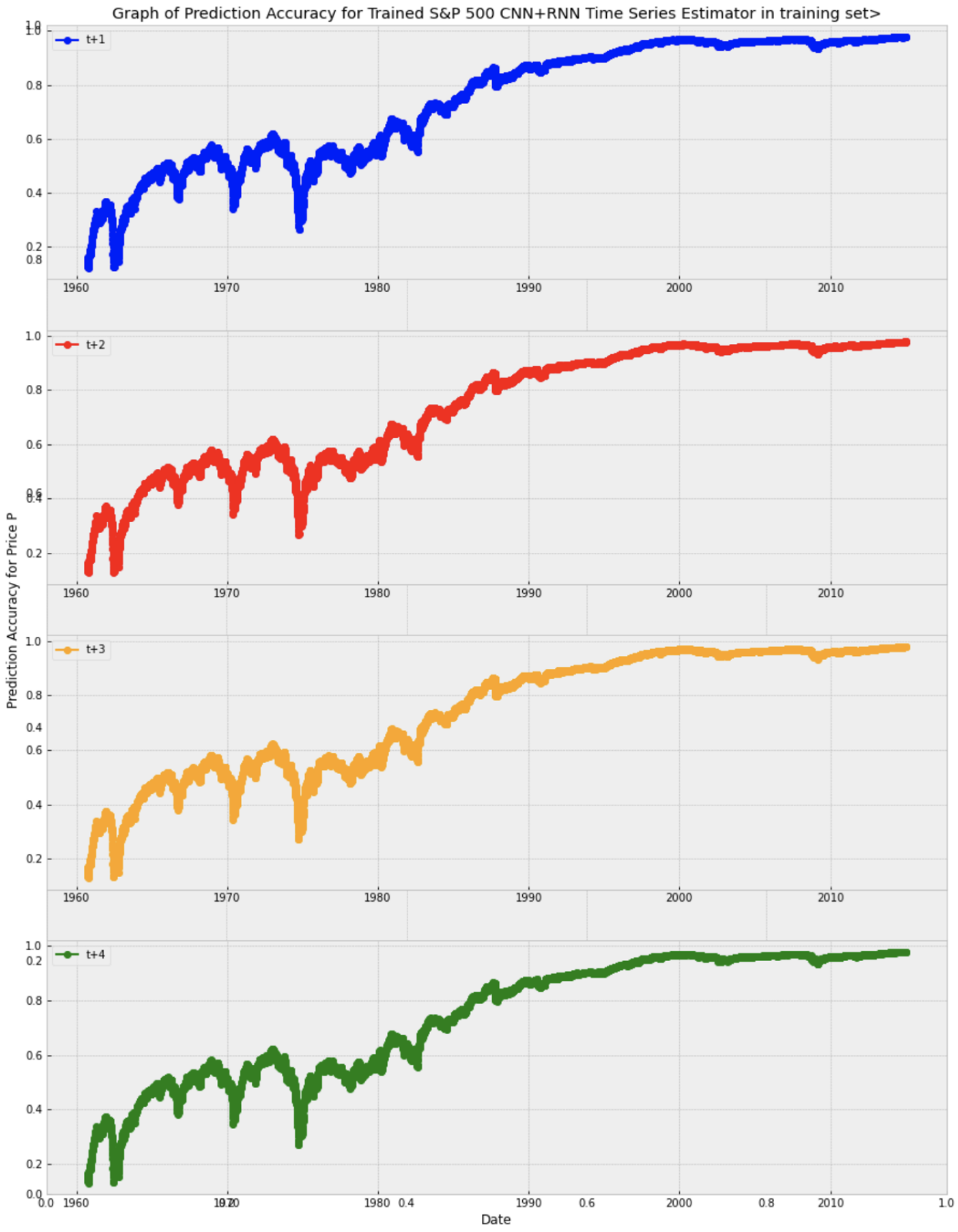

Fig 3.7: Prediction Error Graph as a function of timestamp for Training dataset for next 4 datapoint in dataset

4. Testing set configuration

4.1 Dataset for Testing the Recurrent Neural Network:

- The dataset used for testing also contains an overlay of the Schiller P/E for a longer time interval i.e., from 1 Jan 1960 to 31 Dec 2021, onto the S&P500 for the same period.
- The NYSE and NASDAQ average about 253 trading days a year. We use 20% for testing, i.e. first 200 days for training, and consecutive 53 data points for testing.

	Date	Open	High	Low	Close	PE_CAPE
0	1960-01-04	59.910000	59.910000	59.910000	59.910000	18.338285
1	1960-01-05	60.389999	60.389999	60.389999	60.389999	18.338285
2	1960-01-06	60.130001	60.130001	60.130001	60.130001	18.338285
3	1960-01-07	59.689999	59.689999	59.689999	59.689999	18.338285
4	1960-01-08	59.500000	59.500000	59.500000	59.500000	18.338285

Fig 4.1: Testing data consisting of an overlay of the Schiller P/E onto the S&P500 dataset

4.2 Testing the Recurrent Neural Network with dataset:

We first train the Recurrent Neural Network model with the 80% training dataset that we have chosen and then test it with the remaining 20% to ensure it functions correctly.

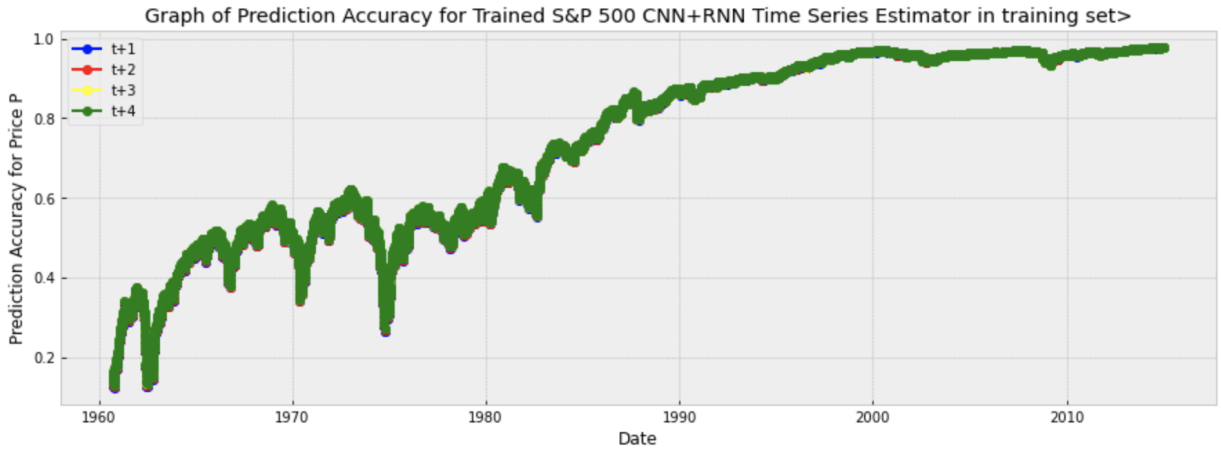

Fig 4.2: Visualizing the results in testing dataset

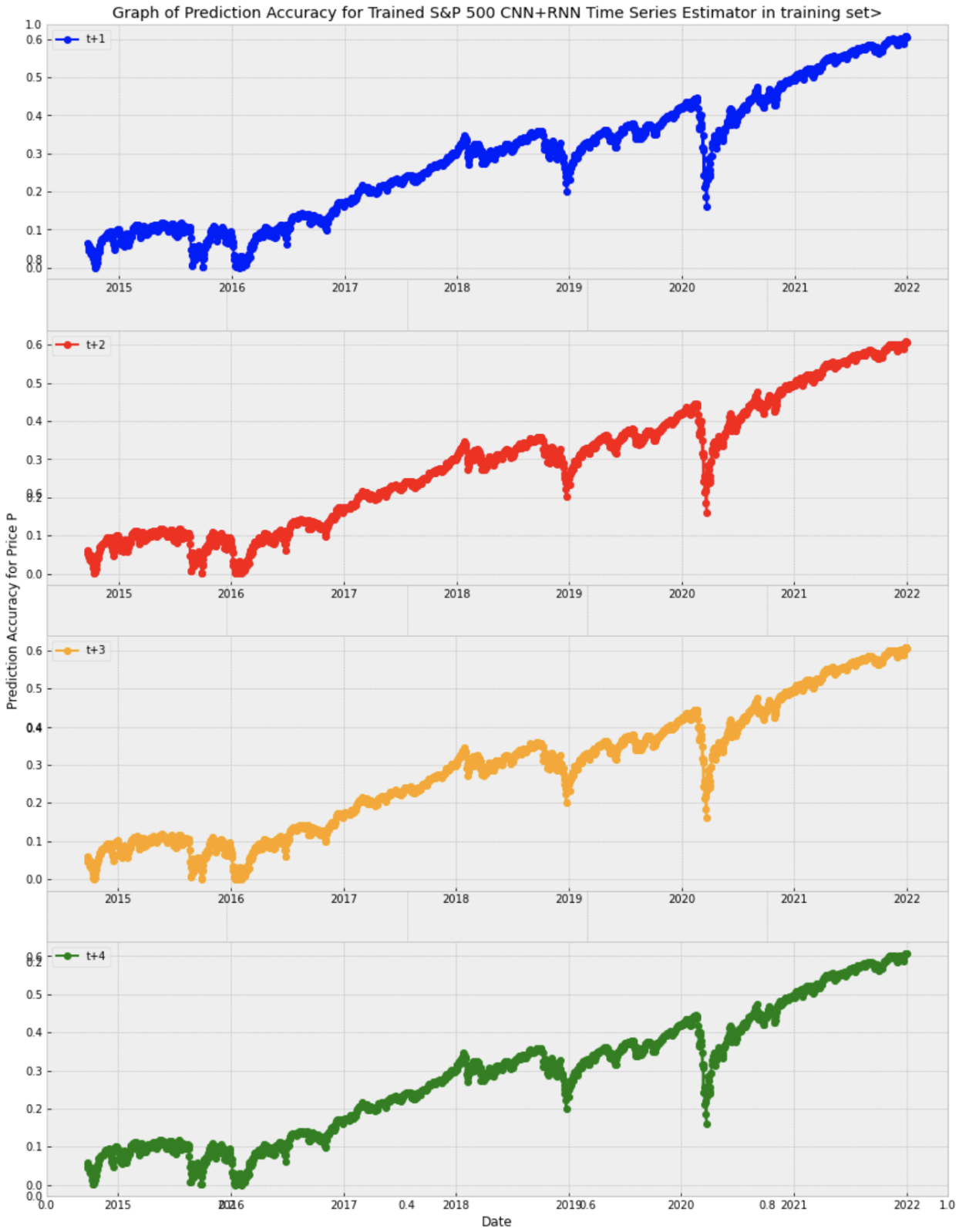

Fig 4.3: Prediction Error Graph as a function of timestamp for Testing dataset for next 4 datapoint in dataset

5. Optimizing Convolutional and Recurrent Neural Network performance to yield maximum achievable accuracy

In order to optimize the model, we increased the number layer from 1 to 2, the batch size from 64 to 128 and the learning rate from 0.001 to 0.005 to speed up the learning process.

```
Epoch 1/25
107/107 [==============================] - 4s 25ms/step - loss: 0.0434 - mse: 0.0434 - mae: 0.1568
Epoch 2/25
107/107 [==============================] - 3s 24ms/step - loss: 0.0360 - mse: 0.0360 - mae: 0.1385
Epoch 3/25
107/107 [==============================] - 3s 24ms/step - loss: 0.0382 - mse: 0.0382 - mae: 0.1424
Epoch 4/25
107/107 [==============================] - 3s 24ms/step - loss: 0.0108 - mse: 0.0108 - mae: 0.0670
Epoch 5/25
107/107 [==============================] - 3s 24ms/step - loss: 0.0020 - mse: 0.0020 - mae: 0.0312
Epoch 6/25
107/107 [==============================] - 3s 24ms/step - loss: 0.0016 - mse: 0.0016 - mae: 0.0277
Epoch 7/25
107/107 [==============================] - 3s 25ms/step - loss: 0.0014 - mse: 0.0014 - mae: 0.0257
Epoch 8/25
107/107 [==============================] - 3s 26ms/step - loss: 0.0012 - mse: 0.0012 - mae: 0.0239
Epoch 9/25
107/107 [==============================] - 3s 26ms/step - loss: 0.0012 - mse: 0.0012 - mae: 0.0233
Epoch 10/25
107/107 [==============================] - 3s 25ms/step - loss: 0.0011 - mse: 0.0011 - mae: 0.0224
Epoch 11/25
107/107 [==============================] - 3s 24ms/step - loss: 0.0011 - mse: 0.0011 - mae: 0.0225
Epoch 12/25
107/107 [==============================] - 3s 24ms/step - loss: 0.0010 - mse: 0.0010 - mae: 0.0226
Epoch 13/25
107/107 [==============================] - 3s 24ms/step - loss: 8.3598e-04 - mse: 8.3598e-04 - mae: 0.0200
Epoch 14/25
107/107 [==============================] - 3s 24ms/step - loss: 7.7468e-04 - mse: 7.7468e-04 - mae: 0.0192
Epoch 15/25
107/107 [==============================] - 3s 24ms/step - loss: 8.6222e-04 - mse: 8.6222e-04 - mae: 0.0204
Epoch 16/25
107/107 [==============================] - 3s 24ms/step - loss: 7.1387e-04 - mse: 7.1387e-04 - mae: 0.0186
Epoch 17/25
107/107 [==============================] - 3s 24ms/step - loss: 8.4284e-04 - mse: 8.4284e-04 - mae: 0.0201
Epoch 18/25
107/107 [==============================] - 3s 24ms/step - loss: 7.6443e-04 - mse: 7.6443e-04 - mae: 0.0193
Epoch 19/25
107/107 [==============================] - 3s 25ms/step - loss: 8.1354e-04 - mse: 8.1354e-04 - mae: 0.0203
Epoch 20/25
107/107 [==============================] - 3s 24ms/step - loss: 6.8207e-04 - mse: 6.8207e-04 - mae: 0.0188
Epoch 21/25
107/107 [==============================] - 3s 24ms/step - loss: 6.9323e-04 - mse: 6.9323e-04 - mae: 0.0188
Epoch 22/25
107/107 [==============================] - 3s 26ms/step - loss: 6.7555e-04 - mse: 6.7555e-04 - mae: 0.0187
Epoch 23/25
107/107 [==============================] - 3s 26ms/step - loss: 6.8511e-04 - mse: 6.8511e-04 - mae: 0.0188
Epoch 24/25
107/107 [==============================] - 3s 26ms/step - loss: 7.3040e-04 - mse: 7.3040e-04 - mae: 0.0194
Epoch 25/25
107/107 [==============================] - 3s 26ms/step - loss: 6.5276e-04 - mse: 6.5276e-04 - mae: 0.0186
<keras.callbacks.History at 0x2b05ee868ac0>
```

Fig 5.1: MSE & MAE associated with no. of epochs

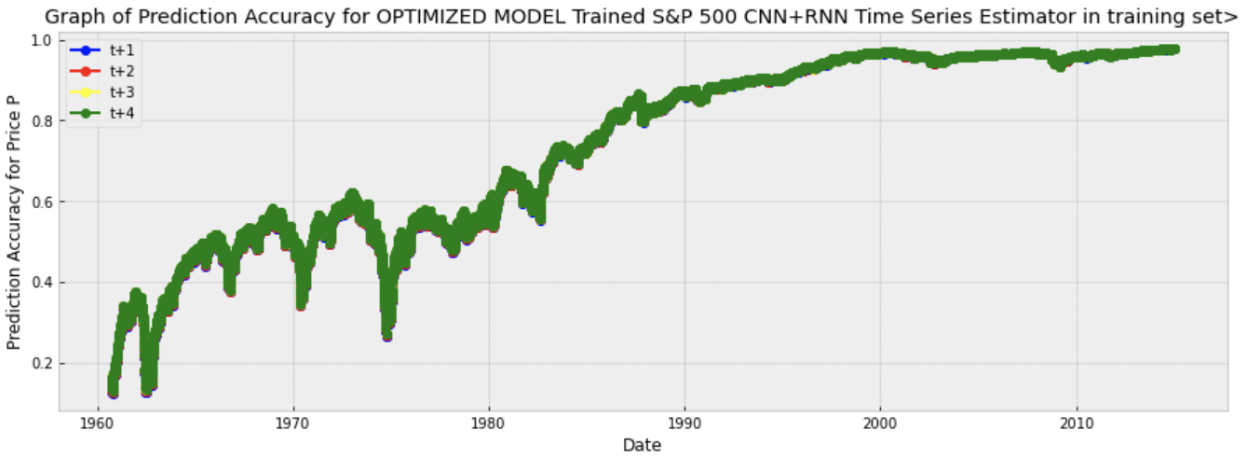

Fig 5.2: Prediction Error Graph as a function of timestamp after optimizing CNN+RNN for next 1 datapoint in Training dataset

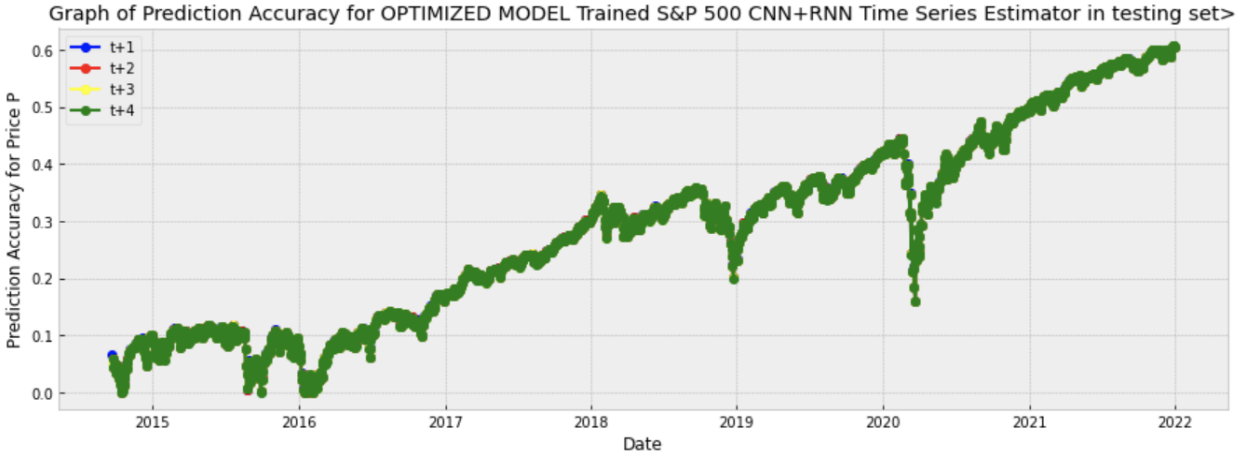

Fig 5.3: Prediction Error Graph as a function of timestamp after optimizing CNN+RNN for next 1 datapoint in Testing dataset

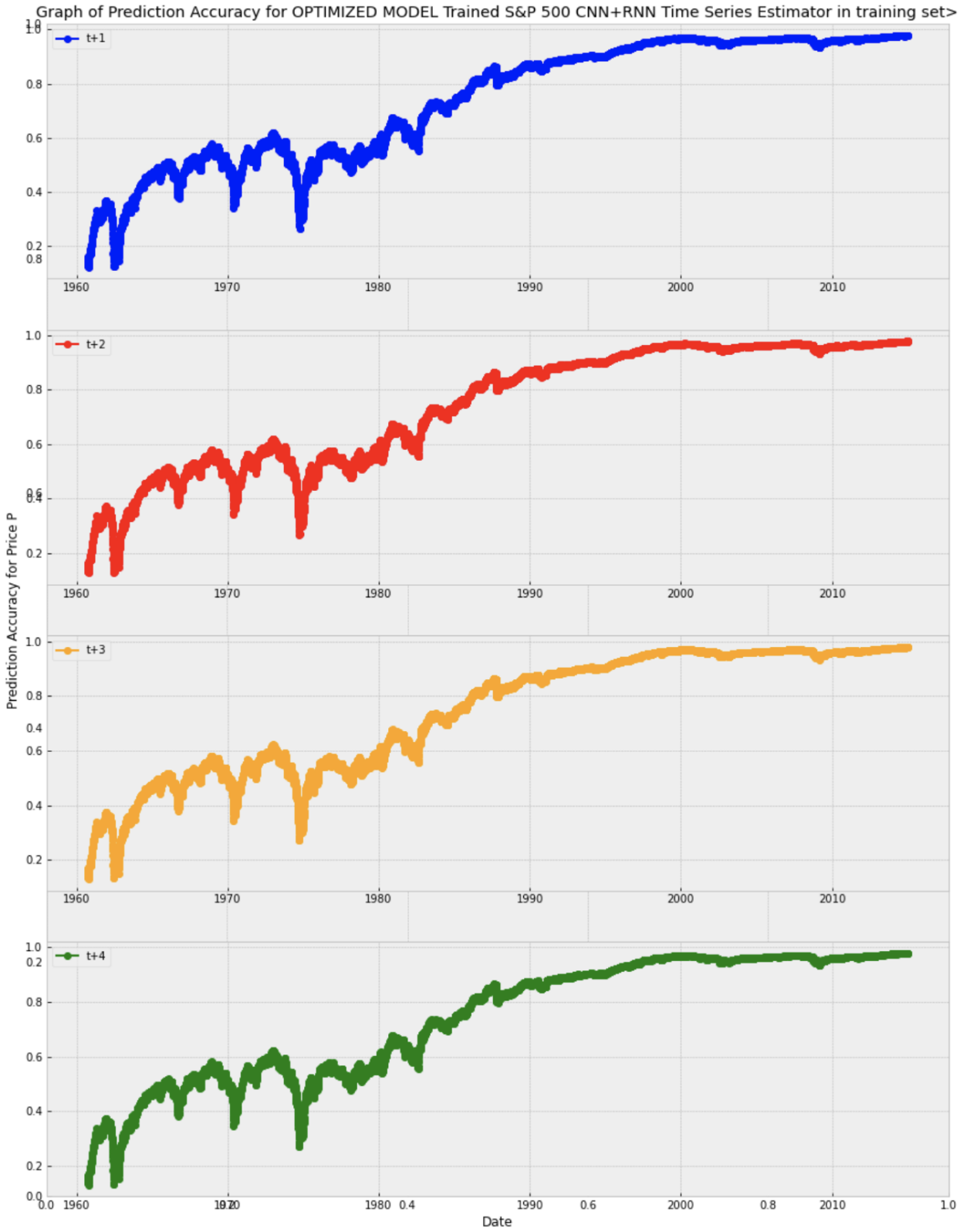

Fig 5.4: Prediction Error Graph as a function of timestamp after optimizing CNN+RNN for next 4 data points in training dataset

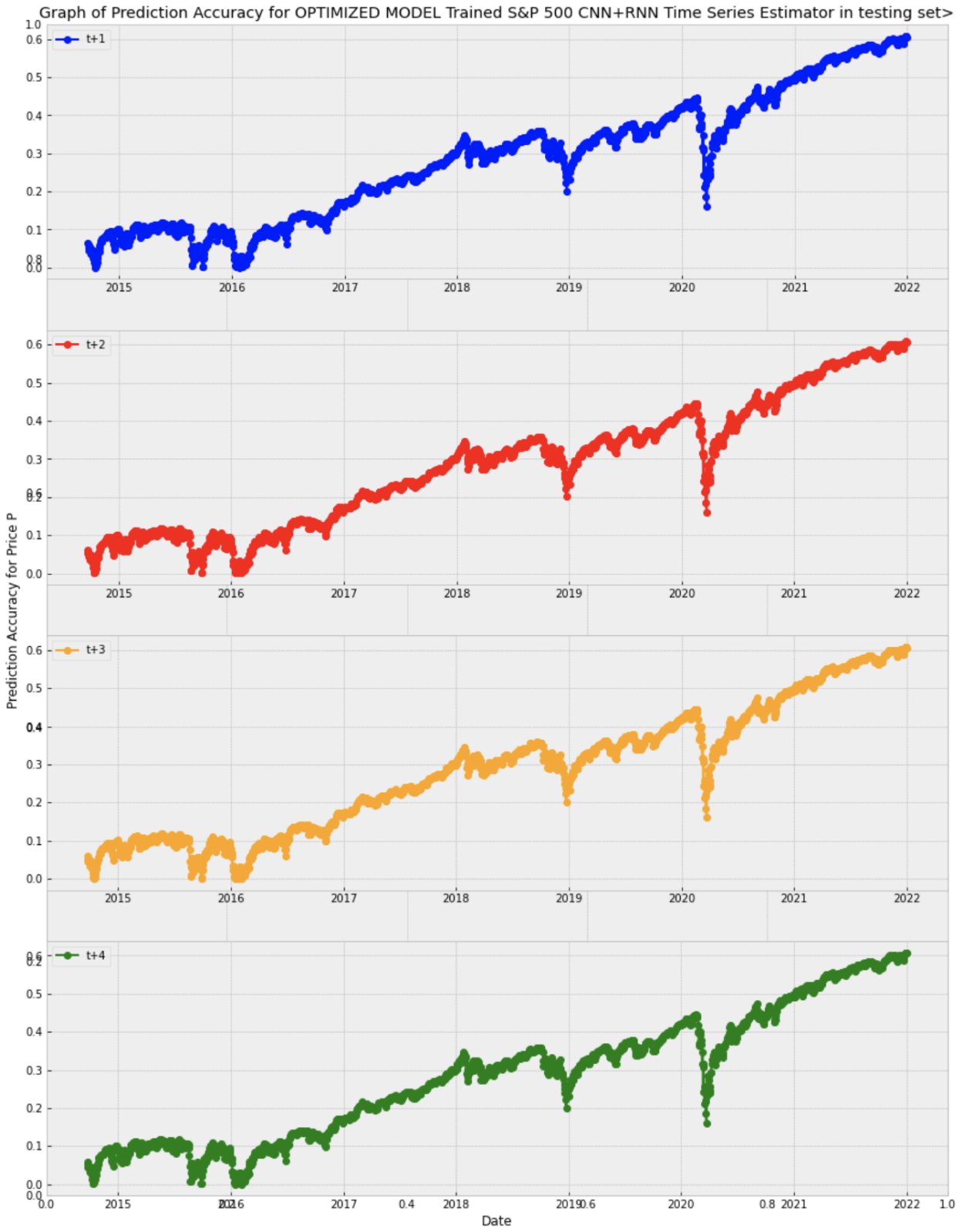

Fig 5.5: Prediction Error Graph as a function of timestamp after optimizing CNN+RNN for next 4 data points in testing dataset

6. Unoptimized CNN+RNN results VS Optimized CNN+RNN results

The below graphs show the difference between the actual prices (green line) against the predicted prices (red line) for the unoptimized and optimized graphs respectively. It is clearly visible how the predicted prices are mismatched with the actual prices in the unoptimized graph. Whereas in the optimized graph the predicted prices align similarly to the actual prices.

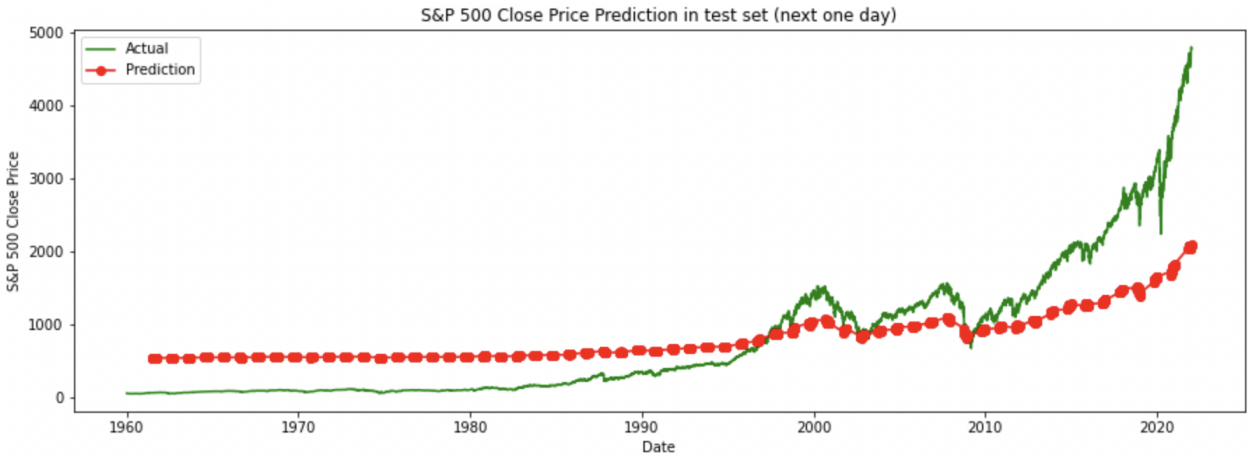

Fig 6.1: Visualizing the results in before optimizing the CNN+RNN

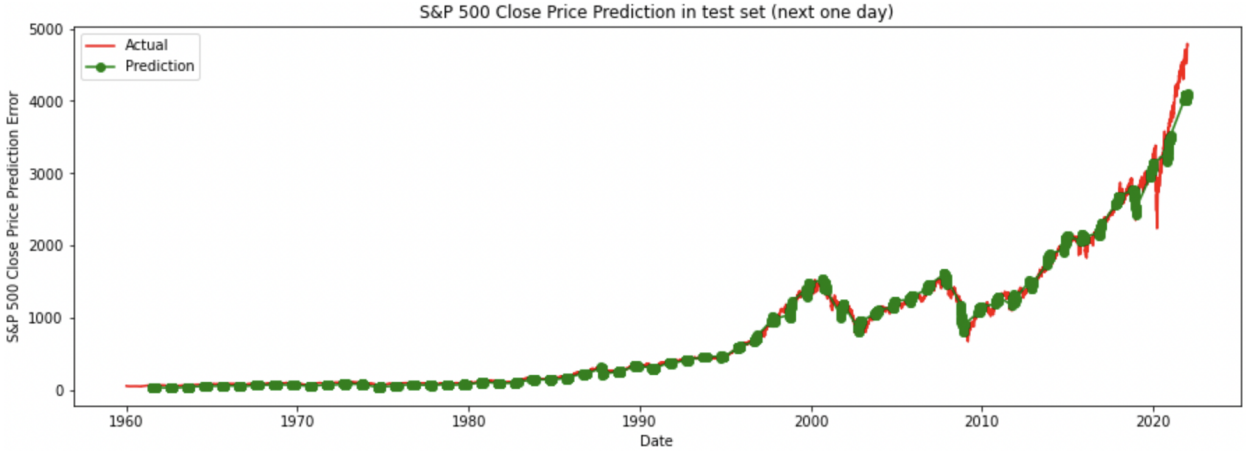

Fig 6.2: Visualizing the results after optimizing CNN+RNN

7. CNN+RNN output results for noise free & noise-corrupted input as table and graph of Date and Prediction Error:

7.1 Prediction Error values for noise corrupted data in CNN+RNN output:

Next 1 Data point in the Dataset

	0.000	0.001	0.002	0.003	0.005	0.010	0.020	0.030	0.050	0.100
1961-07-07	0.415717	0.416029	0.415956	0.415594	0.415124	0.415491	0.415548	0.418384	0.412025	0.412857
1961-07-10	0.410238	0.409148	0.305718	0.607583	0.348587	0.591615	0.280046	0.369158	0.915342	3.971183
1961-07-11	0.408194	0.405424	0.397784	0.414182	0.425306	0.458767	0.545206	0.289896	0.703866	0.351839
1961-07-12	0.405252	0.407094	0.406691	0.404176	0.411628	0.379338	0.376500	0.352725	0.453657	0.437054
1961-07-13	0.403735	0.377410	0.395766	0.395455	0.256642	0.548432	0.626117	0.464673	0.264956	0.124766
...	...	...	...	...	...	...	...	...	...	...
2021-12-21	0.129428	0.129252	0.129802	0.129488	0.129409	0.129451	0.125155	0.126841	0.121953	0.138904
2021-12-22	0.139670	0.139868	0.139890	0.140112	0.140472	0.140245	0.140062	0.139529	0.146044	0.119206
2021-12-23	0.142144	0.142097	0.141955	0.142584	0.142102	0.142683	0.140216	0.138214	0.138441	0.123557
2021-12-27	0.149267	0.149256	0.149213	0.149269	0.149214	0.149421	0.149212	0.148567	0.149548	0.149376
2021-12-28	0.142608	0.142608	0.142609	0.142607	0.142605	0.142601	0.142624	0.142602	0.142589	0.142509

Fig 7.1: Date vs Prediction Error Table of Noise Corrupted Input for next 1 Data point in the Dataset

Next 2 Data points in the Dataset

	0.000	0.001	0.002	0.003	0.005	0.010	0.020	0.030	0.050	0.100
1961-07-08	0.410252	0.410259	0.409783	0.409703	0.411937	0.409563	0.413723	0.406693	0.409020	0.380868
1961-07-11	0.405193	0.399320	0.456093	0.515601	0.496453	0.578934	0.313683	0.301327	0.839941	5.955050
1961-07-12	0.399987	0.399737	0.395404	0.388345	0.411865	0.381121	0.337620	0.429933	0.275694	0.883223
1961-07-13	0.396087	0.396687	0.394734	0.400524	0.397199	0.409967	0.411543	0.400810	0.356925	0.387346
1961-07-14	0.402505	0.429504	0.390609	0.493496	0.375708	0.482518	0.316229	0.677899	0.161600	0.754558
...	...	...	...	...	...	...	...	...	...	...
2021-12-22	0.132014	0.131812	0.132233	0.132137	0.131416	0.130993	0.135372	0.136679	0.138291	0.152529
2021-12-23	0.139146	0.139265	0.138876	0.139004	0.140094	0.140668	0.138200	0.144791	0.138677	0.124691
2021-12-24	0.147509	0.147517	0.147844	0.147398	0.146642	0.147074	0.152817	0.152208	0.151078	0.156561
2021-12-28	0.141886	0.141897	0.141892	0.141870	0.141873	0.141931	0.142089	0.142123	0.141929	0.139408
2021-12-29	0.137205	0.137205	0.137205	0.137206	0.137205	0.137205	0.137206	0.137207	0.137211	0.137216

3223 rows × 10 columns

Fig 7.2: Date vs Prediction Error Table of Noise Corrupted Input for next 2 Data Points in the Dataset

Next 3 Data points in the Dataset

	0.000	0.001	0.002	0.003	0.005	0.010	0.020	0.030	0.050	0.100
1961-07-09	0.266236	0.266190	0.266135	0.266184	0.265042	0.265487	0.273369	0.268887	0.262140	0.257728
1961-07-12	0.257666	0.275542	0.279758	0.137019	0.170955	0.570176	0.299520	0.633686	1.563569	1.478296
1961-07-13	0.250933	0.244595	0.251793	0.239996	0.248780	0.212080	0.301634	0.348250	0.034262	0.748543
1961-07-14	0.256575	0.257194	0.257610	0.259074	0.261297	0.263398	0.275441	0.309150	0.315847	0.089328
1961-07-15	0.254593	0.253617	0.235942	0.162596	0.236478	0.119234	0.328297	0.664557	1.390123	0.195487
...	...	...	...	...	...	...	...	...	...	...
2021-12-23	0.111741	0.111652	0.112091	0.111114	0.111292	0.111924	0.101012	0.111848	0.123385	0.077558
2021-12-24	0.124093	0.124076	0.124281	0.123163	0.124204	0.121581	0.123338	0.118259	0.126202	0.137391
2021-12-25	0.120642	0.120805	0.121069	0.120435	0.118060	0.122045	0.111656	0.117243	0.120905	0.086008
2021-12-29	0.116633	0.116617	0.116645	0.116676	0.116639	0.116828	0.116819	0.116481	0.116687	0.117026
2021-12-30	0.106560	0.106559	0.106560	0.106560	0.106559	0.106559	0.106558	0.106555	0.106552	0.106556

3223 rows × 10 columns

Fig 7.3: Date vs Prediction Error Table of Noise Corrupted Input for next 3 Data points in the Dataset

Next 4 Data points in the Dataset

	0.000	0.001	0.002	0.003	0.005	0.010	0.020	0.030	0.050	0.100
1961-07-10	0.260741	0.260860	0.261046	0.260998	0.262602	0.262214	0.259679	0.261541	0.261273	0.262282
1961-07-13	0.251244	0.220819	0.275270	0.184436	0.162567	0.020858	0.787619	1.019813	0.994667	4.947416
1961-07-14	0.254586	0.253435	0.257854	0.253054	0.273190	0.336284	0.192307	0.428209	0.012326	0.060500
1961-07-15	0.249473	0.251104	0.249919	0.249028	0.251860	0.253020	0.266128	0.253457	0.176115	0.416914
1961-07-16	0.248163	0.290157	0.192078	0.157946	0.340474	0.159968	0.156970	1.411771	0.747893	2.173364
...	...	...	...	...	...	...	...	...	...	...
2021-12-24	0.147790	0.147516	0.147824	0.147755	0.146647	0.145903	0.147235	0.140305	0.155839	0.154901
2021-12-25	0.146852	0.146775	0.146873	0.147477	0.147853	0.143828	0.147067	0.149781	0.156042	0.137376
2021-12-26	0.144621	0.144593	0.144842	0.144028	0.146039	0.143588	0.150466	0.136853	0.146131	0.155869
2021-12-30	0.137170	0.137168	0.137186	0.137158	0.137249	0.137308	0.137276	0.137005	0.137961	0.139781
2021-12-31	0.128615	0.128615	0.128614	0.128615	0.128615	0.128612	0.128612	0.128625	0.128637	0.128602

3223 rows × 10 columns

Fig 7.4: Date vs Prediction Error Table of Noise Corrupted Input for next 4 Data points in the Dataset

The above images show the Date VS Prediction Error values for the Conventional and Recurrent Neural Network with noise increasing from 0.001 to 0.1 values.

7.2 Prediction Error VS Date graphs for CNN+RNN output:

Graphs for Next 1 Data point in the Dataset

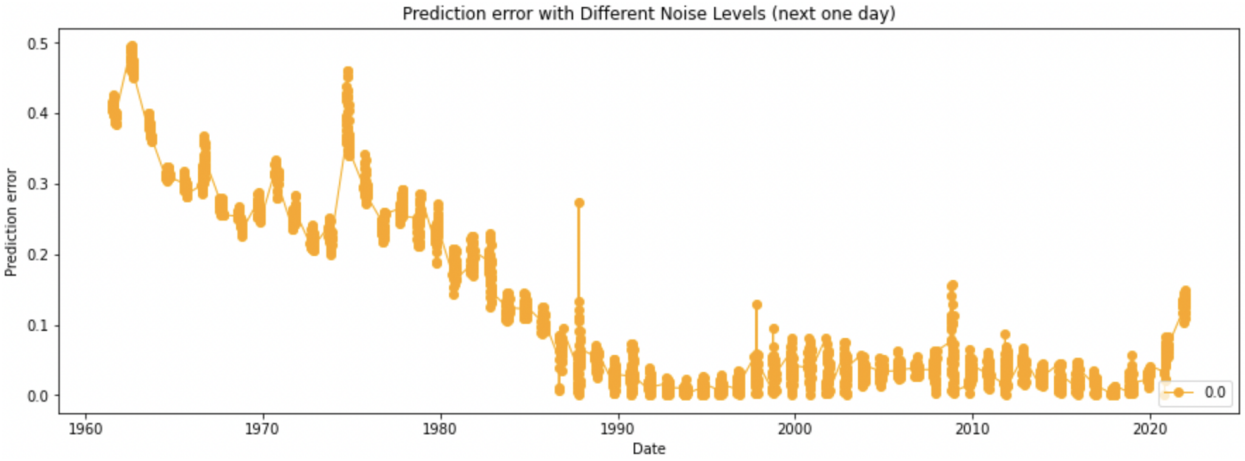

Fig 7.5: Prediction Error vs Date graph for CNN+RNN with 0 noise level

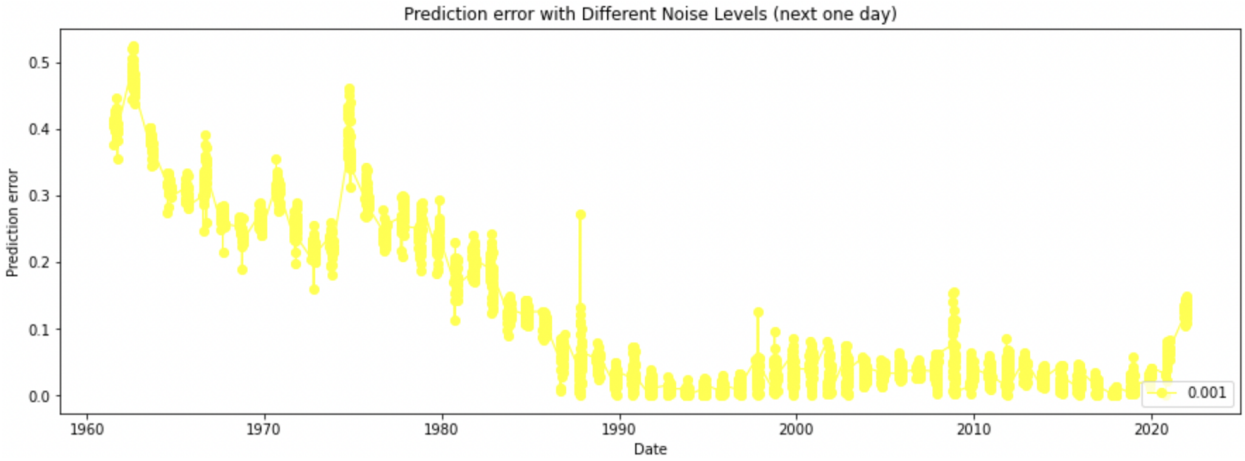

Fig 7.6: Prediction Error vs Date graph for CNN+RNN with 0.001 noise level

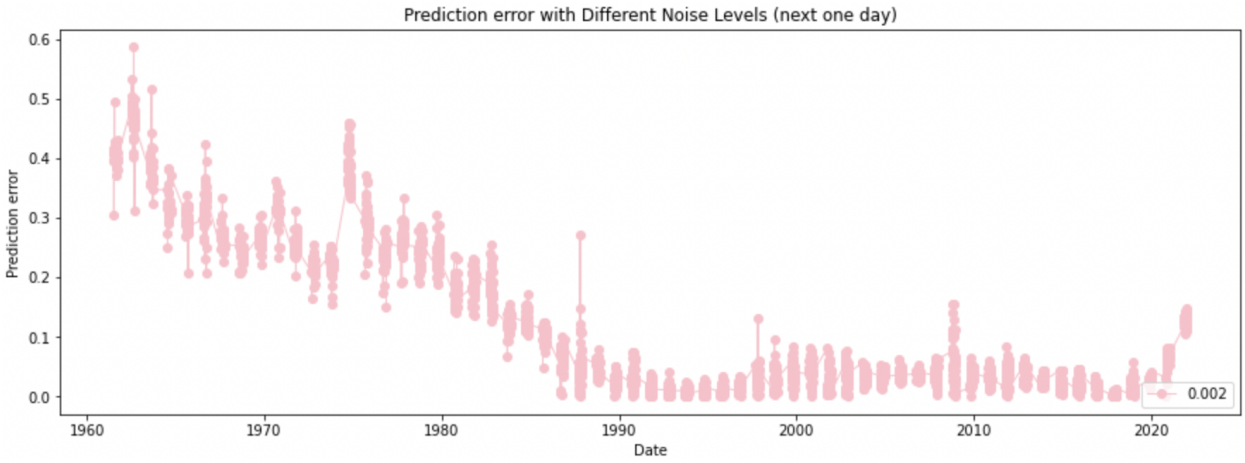

Fig 7.7: Prediction Error vs Date graph for CNN+RNN with 0.002 noise level

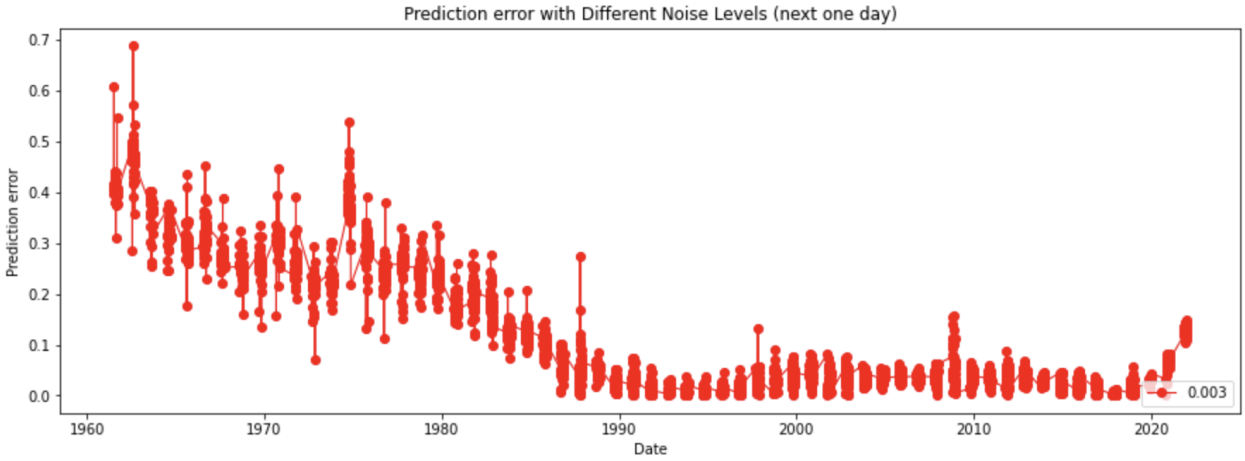

Fig 7.8: Prediction Error vs Date graph for CNN+RNN with 0.003 noise level

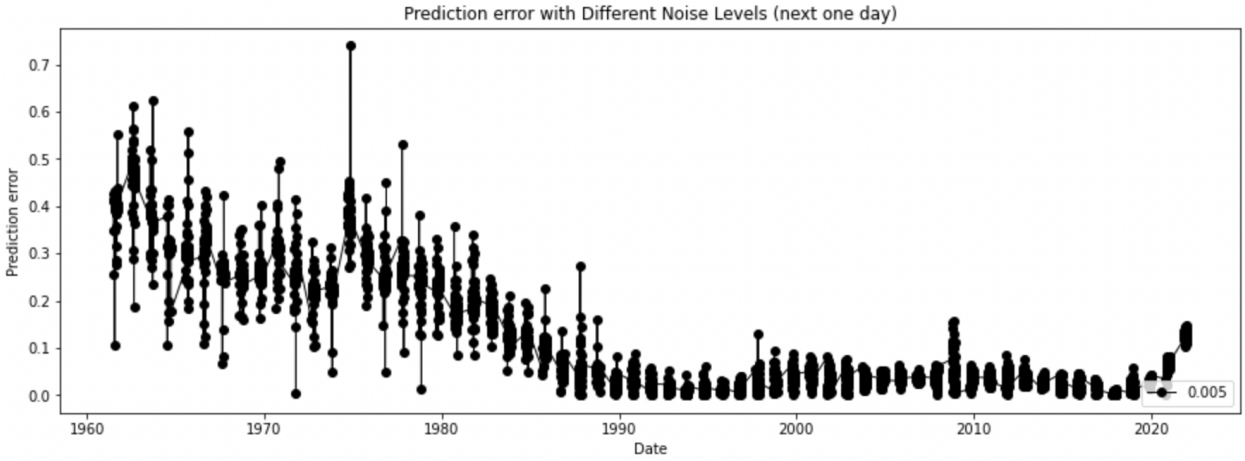

Fig 7.9: Prediction Error vs Date graph for CNN+RNN with 0.005 noise level

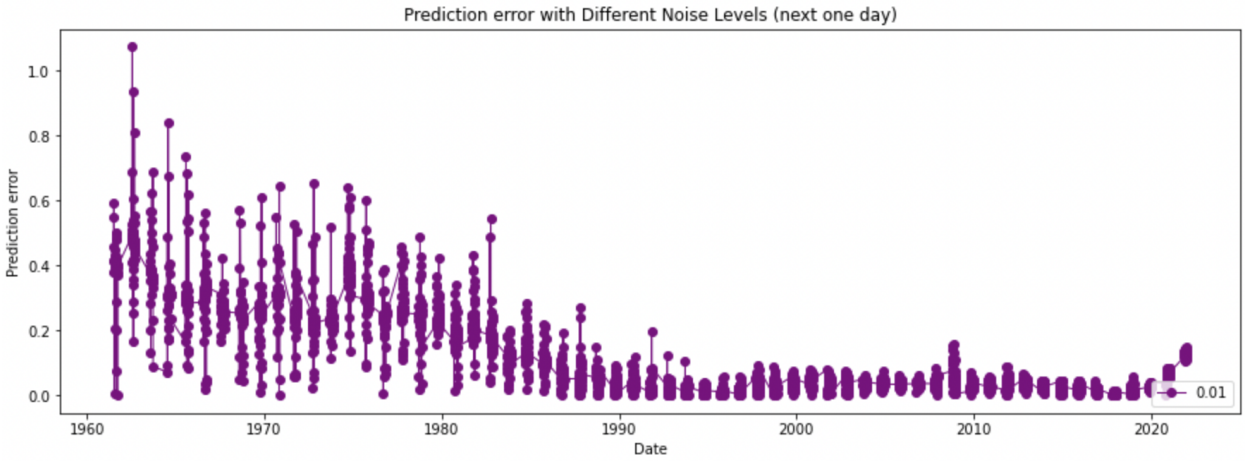

Fig 7.10: Prediction Error vs Date graph for CNN+RNN with 0.01 noise level

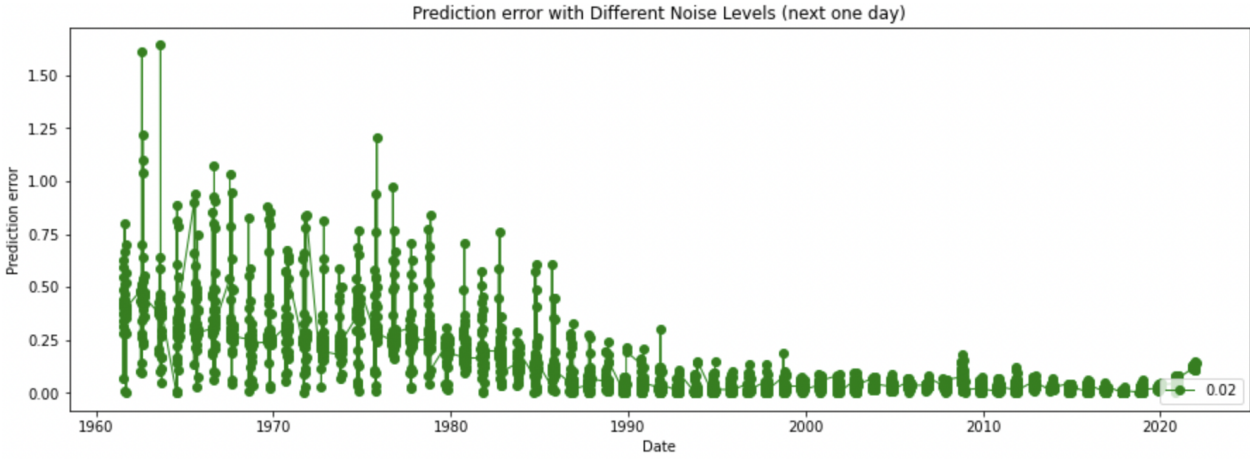

Fig 7.11: Prediction Error vs Date graph for CNN+RNN with 0.02 noise level

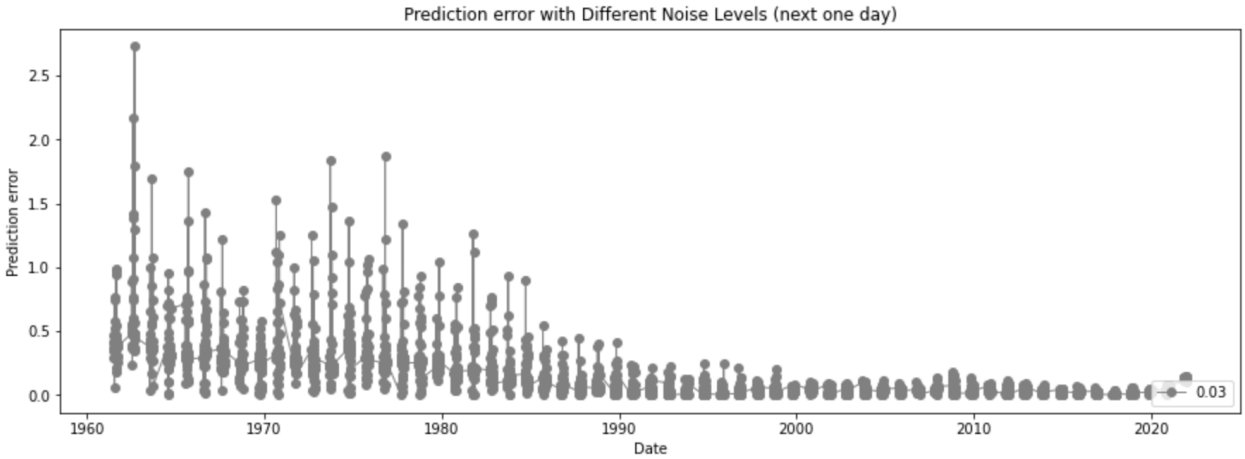

Fig 7.12: Prediction Error vs Date graph for CNN+RNN with 0.03 noise level

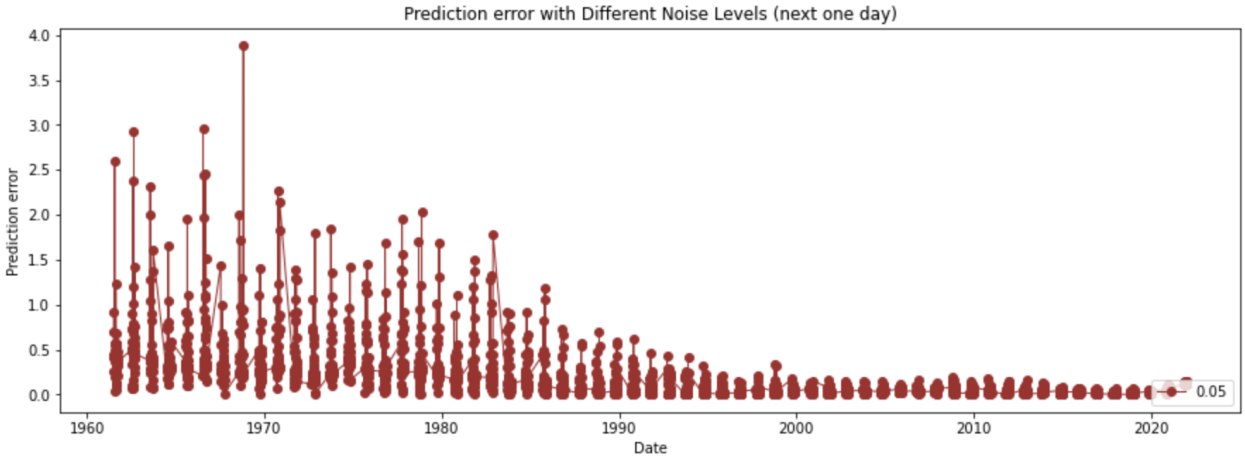

Fig 7.13: Prediction Error vs Date graph for CNN+RNN with 0.05 noise level

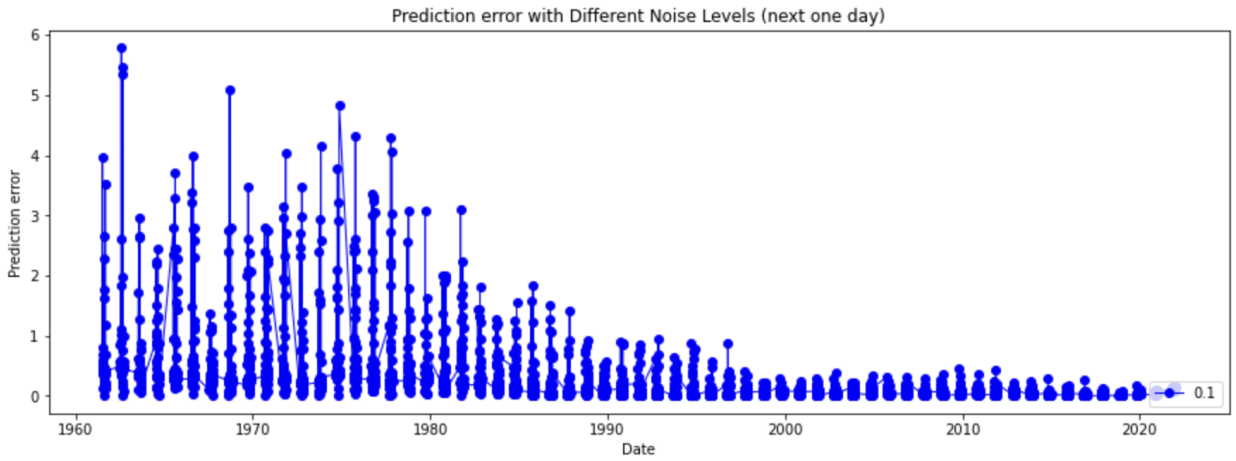

Fig 7.14: Prediction Error vs Date graph for CNN+RNN with 0.1 noise level

Graphs for Next 2 Data points in the Dataset

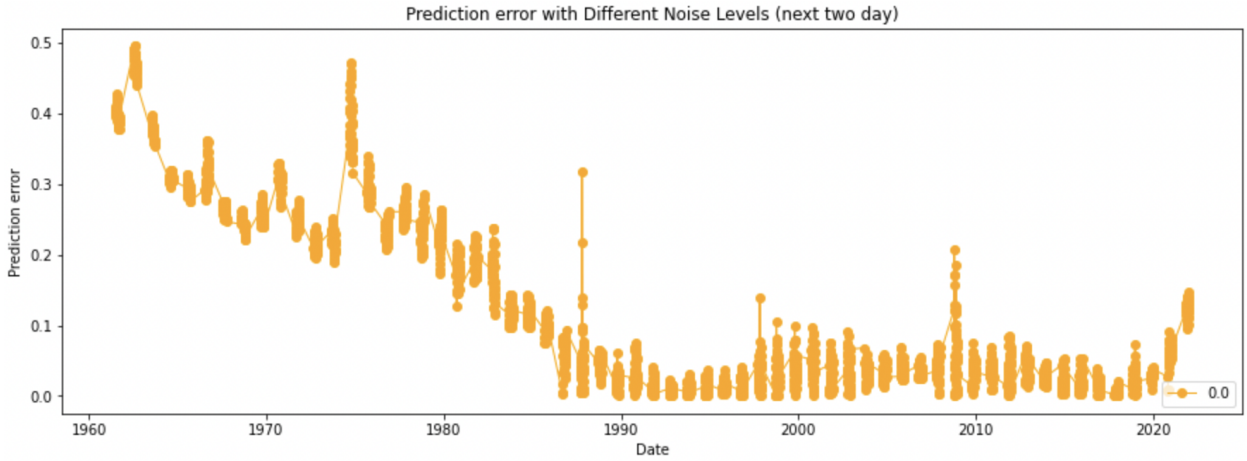

Fig 7.15: Prediction Error vs Date graph for CNN+RNN with 0 noise level

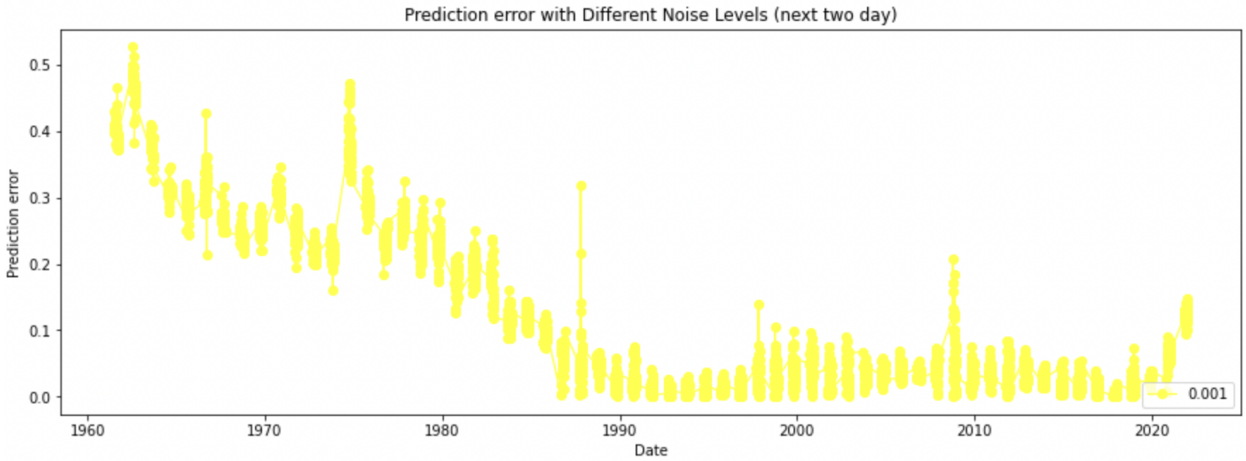

Fig 7.16: Prediction Error vs Date graph for CNN+RNN with 0.001 noise level

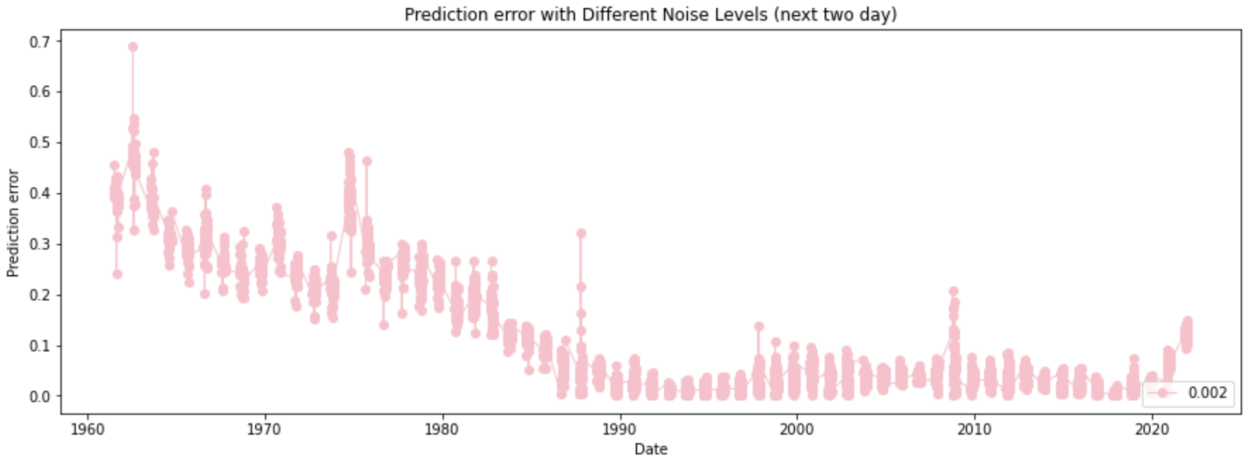

Fig 7.17: Prediction Error vs Date graph for CNN+RNN with 0.002 noise level

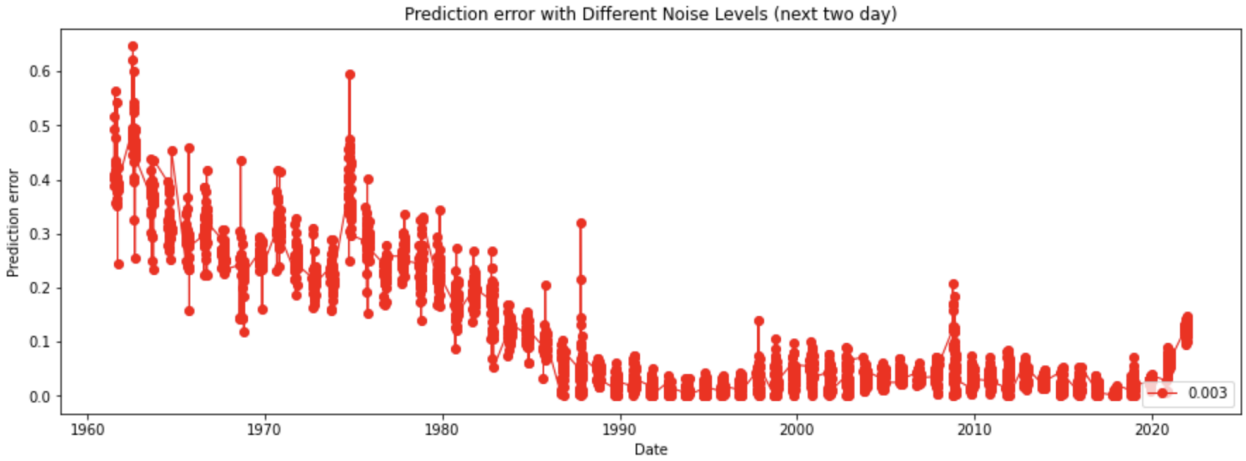

Fig 7.18: Prediction Error vs Date graph for CNN+RNN with 0.003 noise level

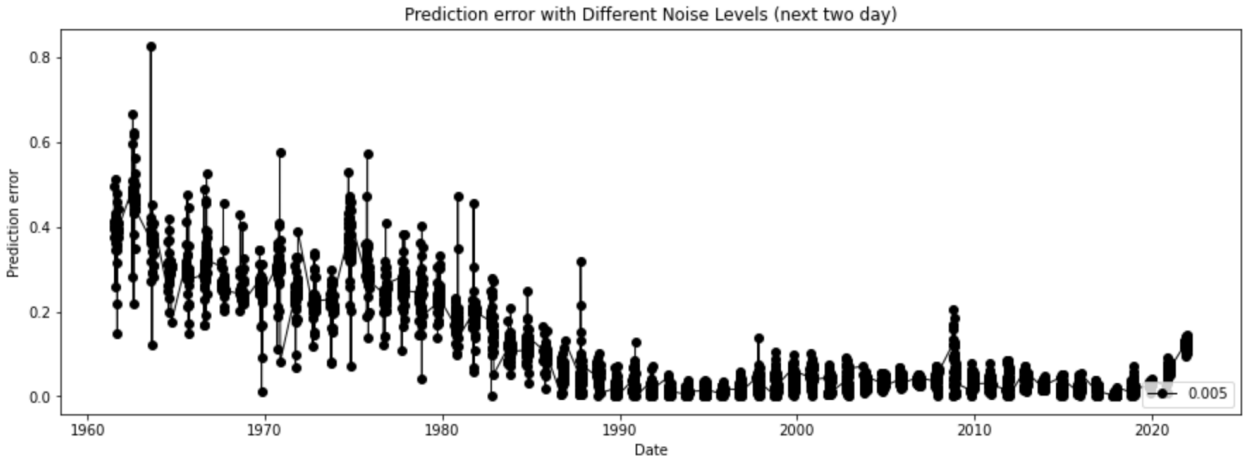

Fig 7.19: Prediction Error vs Date graph for CNN+RNN with 0.005 noise level

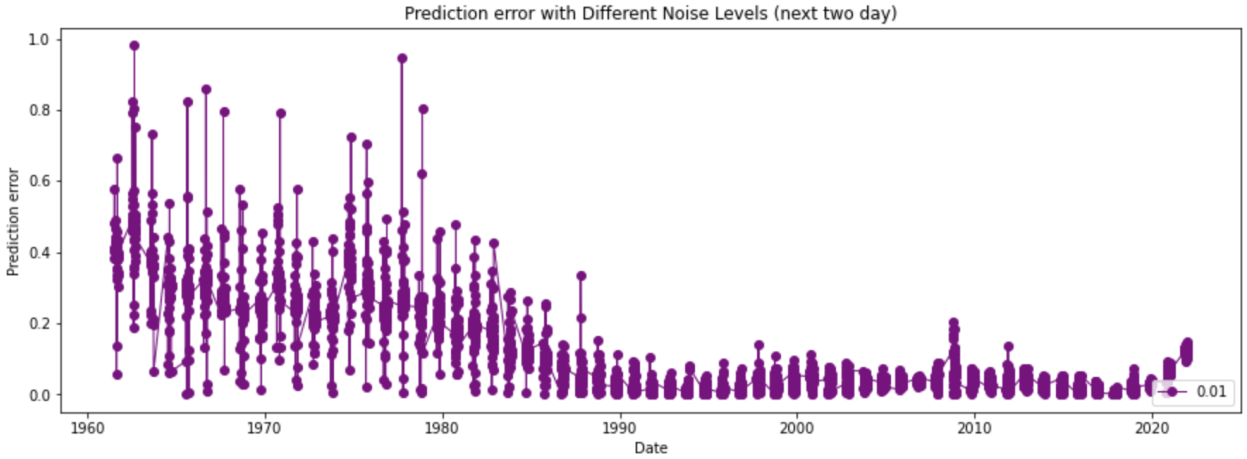

Fig 7.20: Prediction Error vs Date graph for CNN+RNN with 0.01 noise level

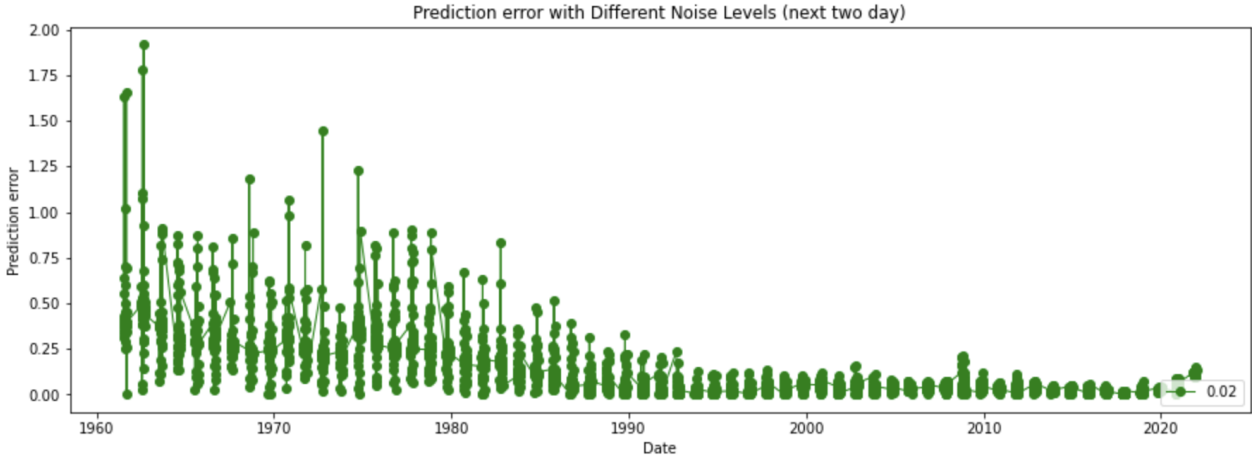

Fig 7.21: Prediction Error vs Date graph for CNN+RNN with 0.02 noise level

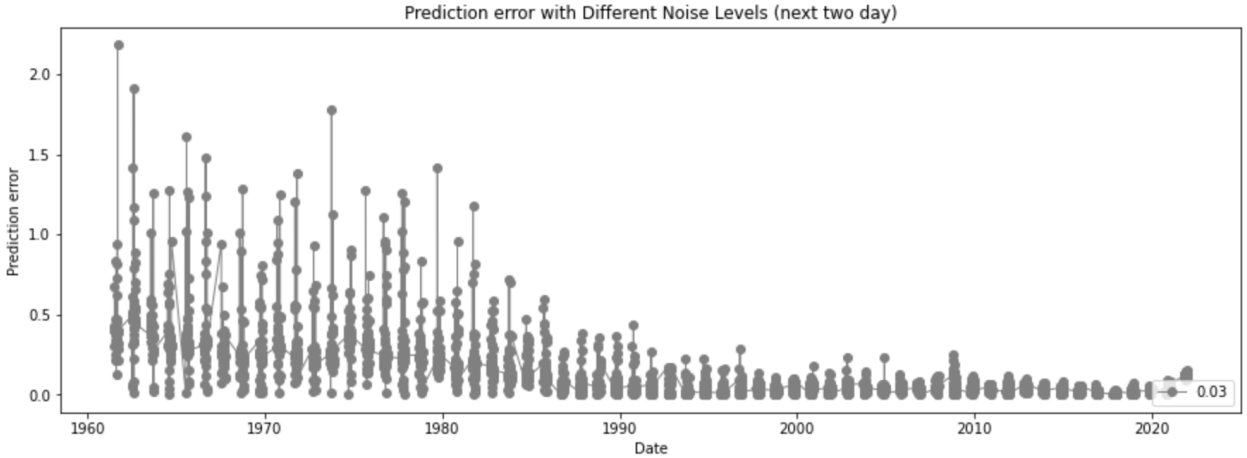

Fig 7.22: Prediction Error vs Date graph for CNN+RNN with 0.03 noise level

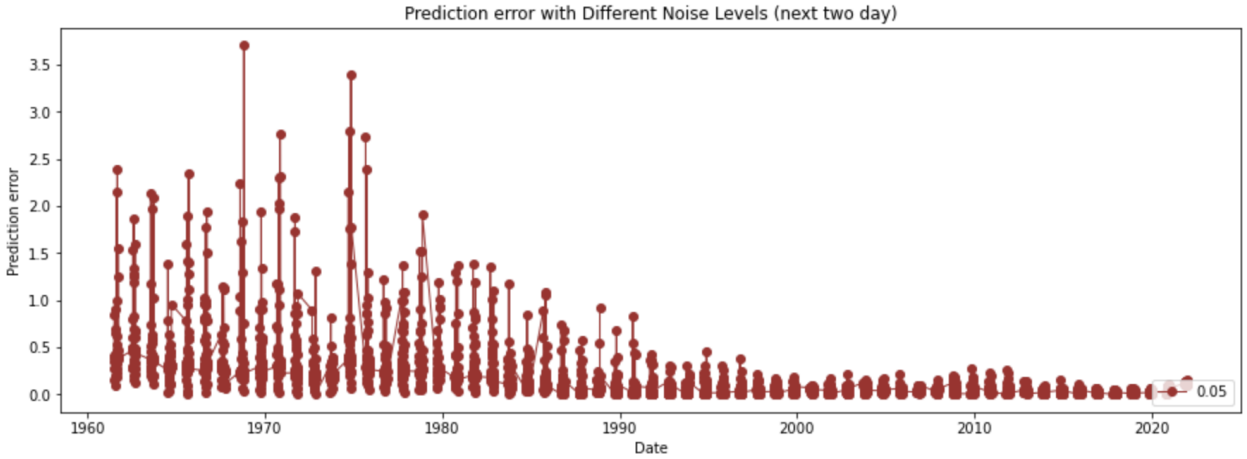

Fig 7.23: Prediction Error vs Date graph for CNN+RNN with 0.05 noise level

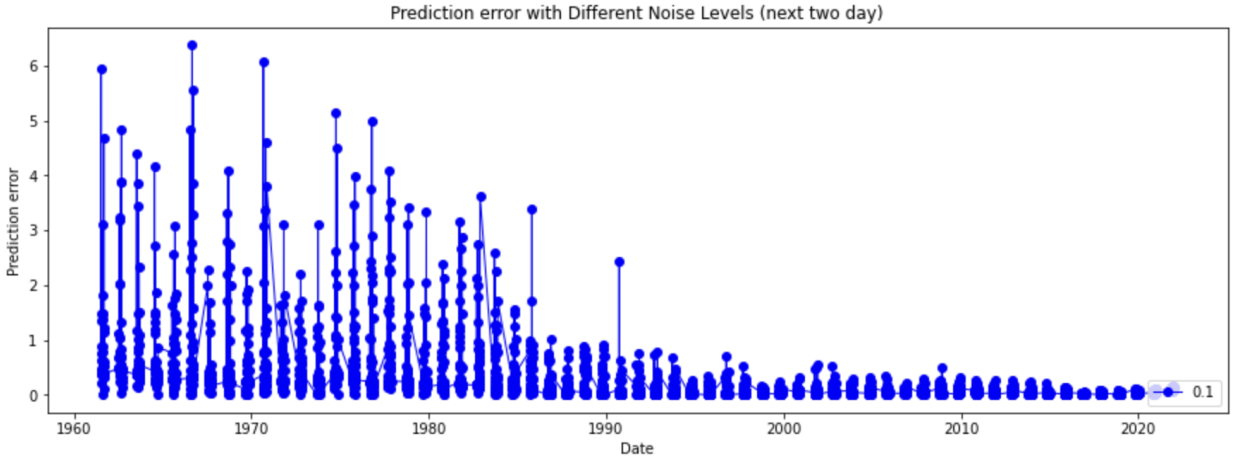

Fig 7.24: Prediction Error vs Date graph for CNN+RNN with 0.1 noise level

Graphs for Next 3 Data points in the Dataset

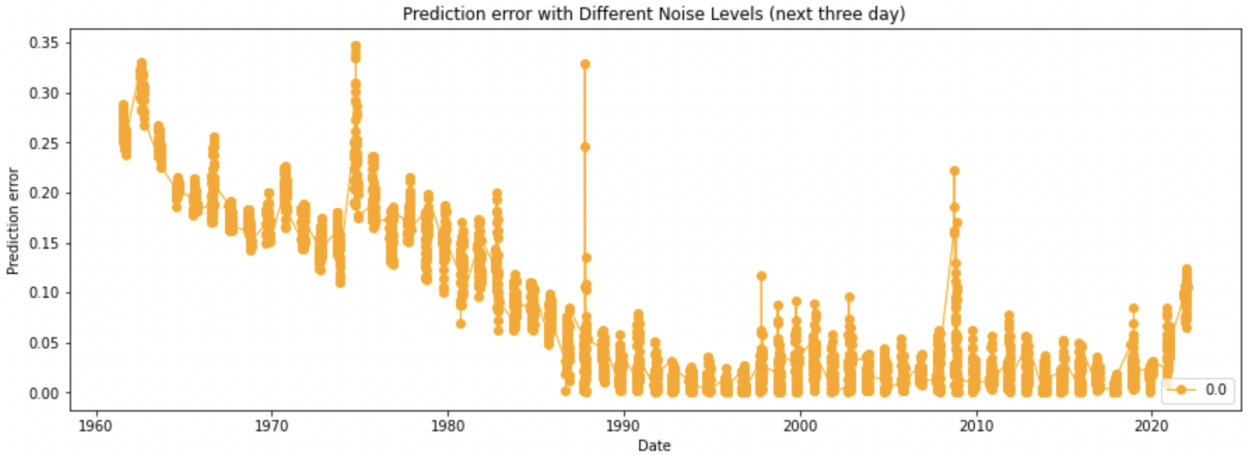

Fig 7.25: Prediction Error vs Date graph for CNN+RNN with 0 noise level

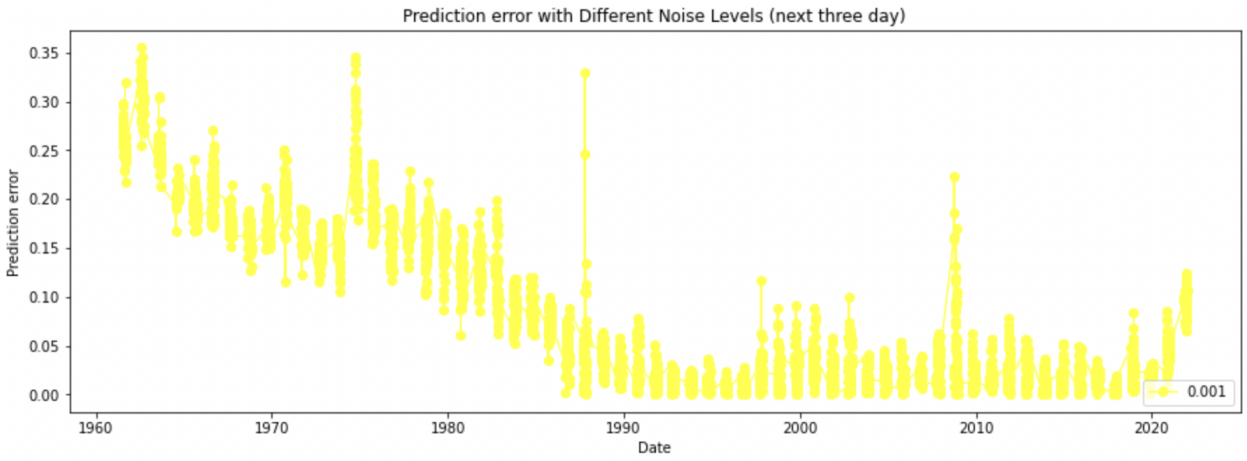

Fig 7.26: Prediction Error vs Date graph for CNN+RNN with 0.001 noise level

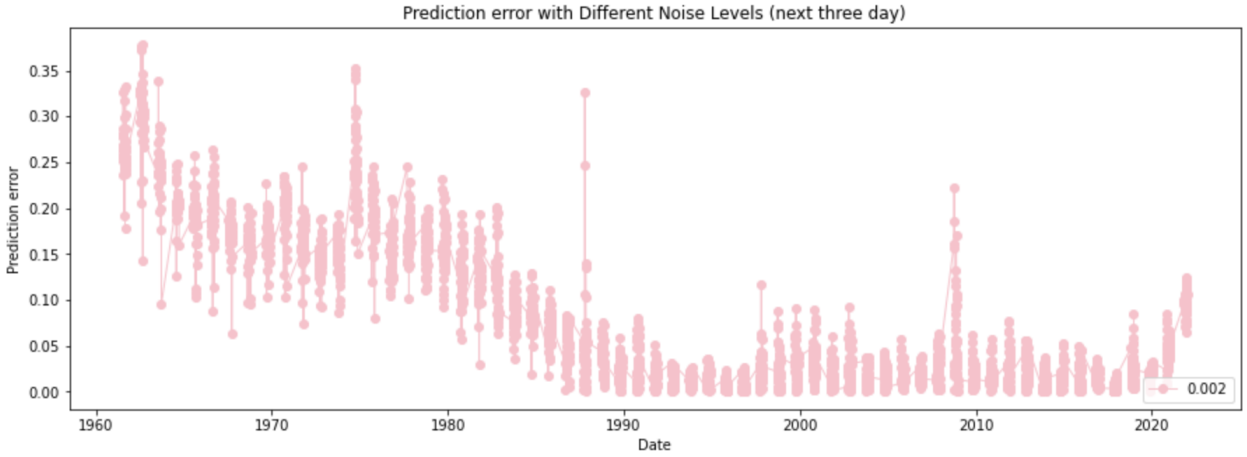

Fig 7.27: Prediction Error vs Date graph for CNN+RNN with 0.002 noise level

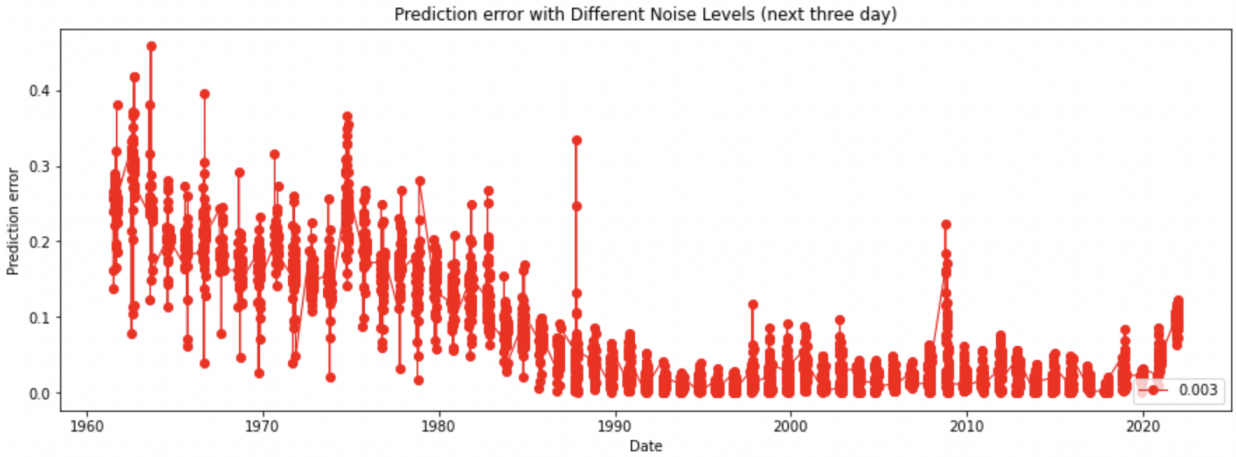

Fig 7.28: Prediction Error vs Date graph for CNN+RNN with 0.003 noise level

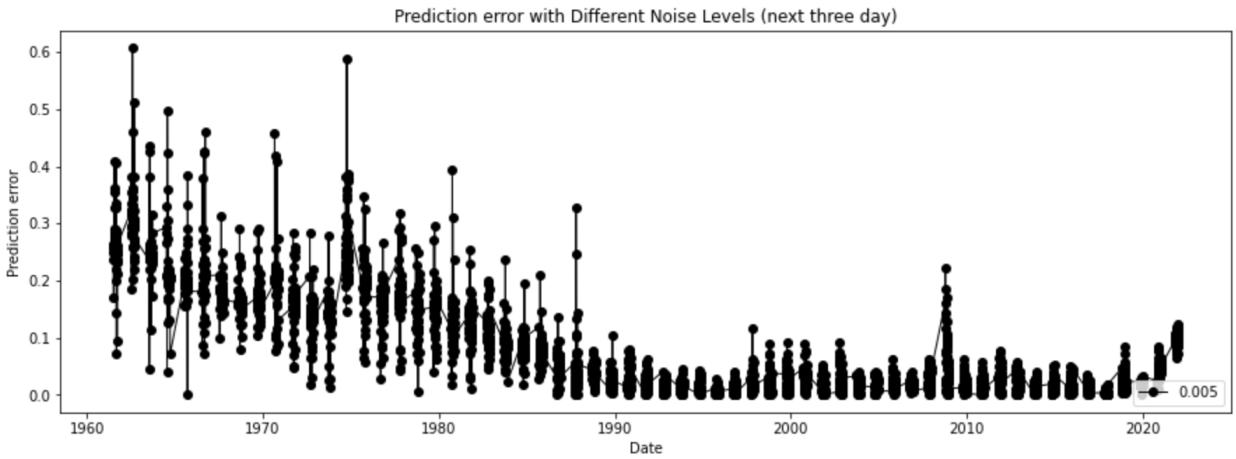

Fig 7.29: Prediction Error vs Date graph for CNN+RNN with 0.005 noise level

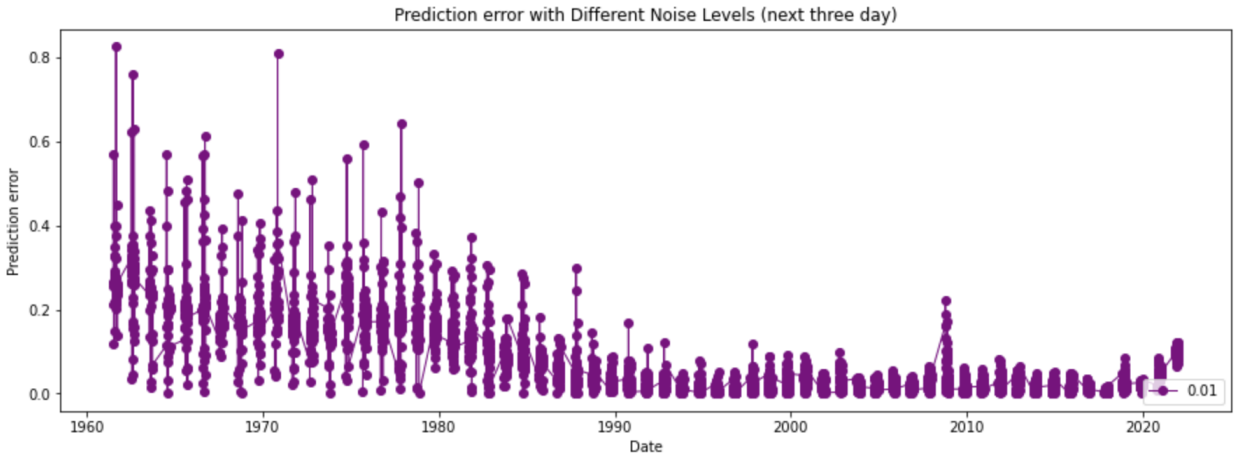

Fig 7.30: Prediction Error vs Date graph for CNN+RNN with 0.01 noise level

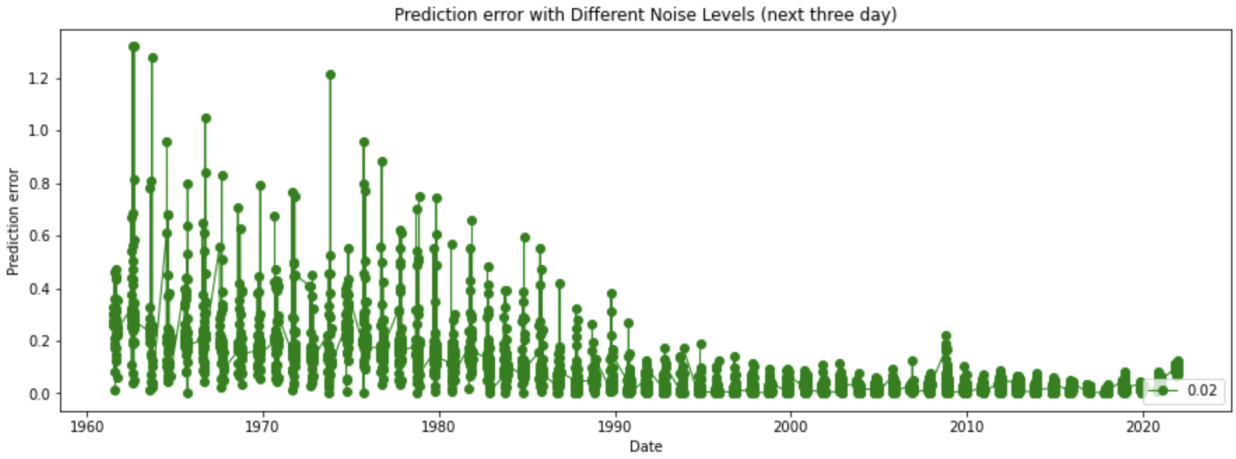

Fig 7.31: Prediction Error vs Date graph for CNN+RNN with 0.02 noise level

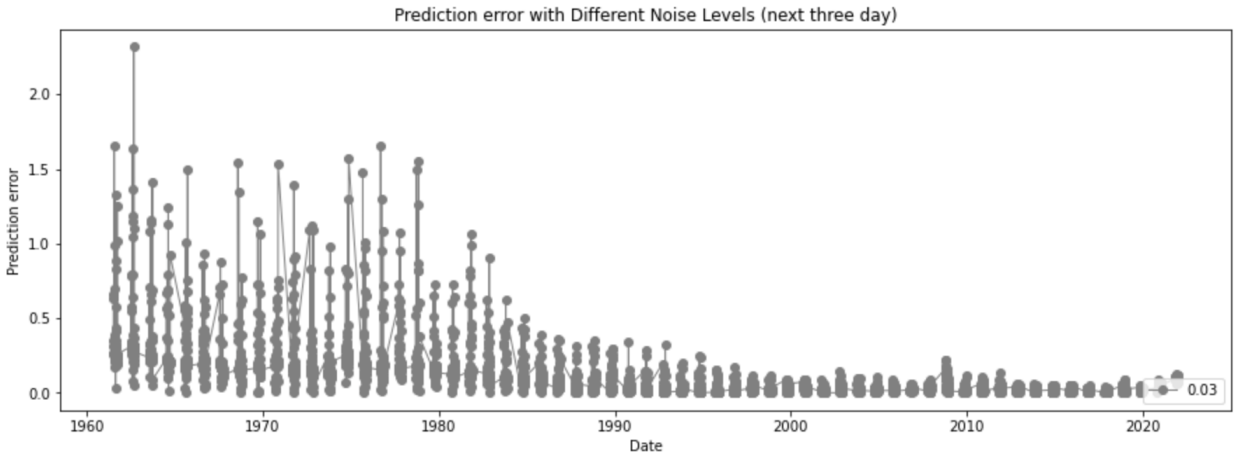

Fig 7.32: Prediction Error vs Date graph for CNN+RNN with 0.03 noise level

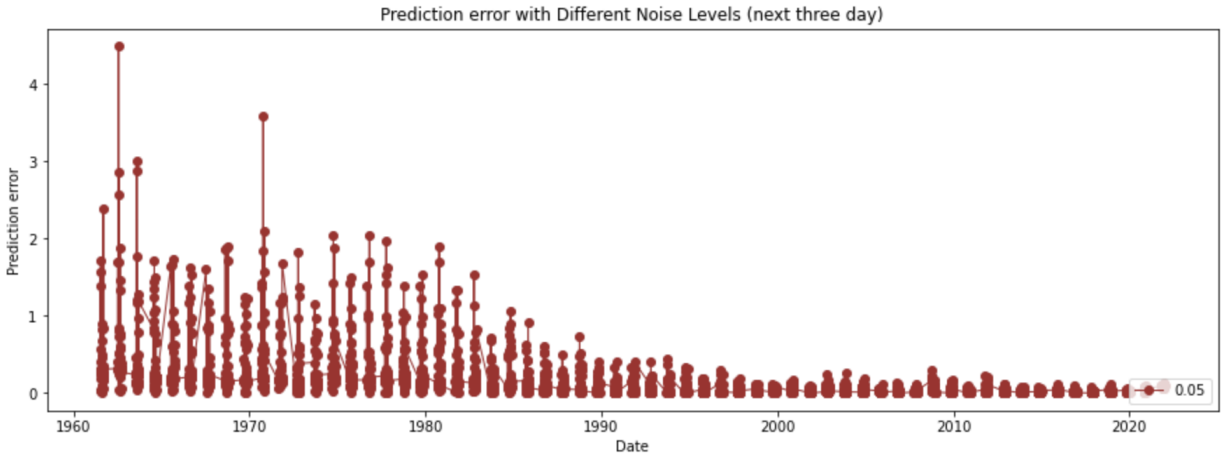

Fig 7.33: Prediction Error vs Date graph for CNN+RNN with 0.05 noise level

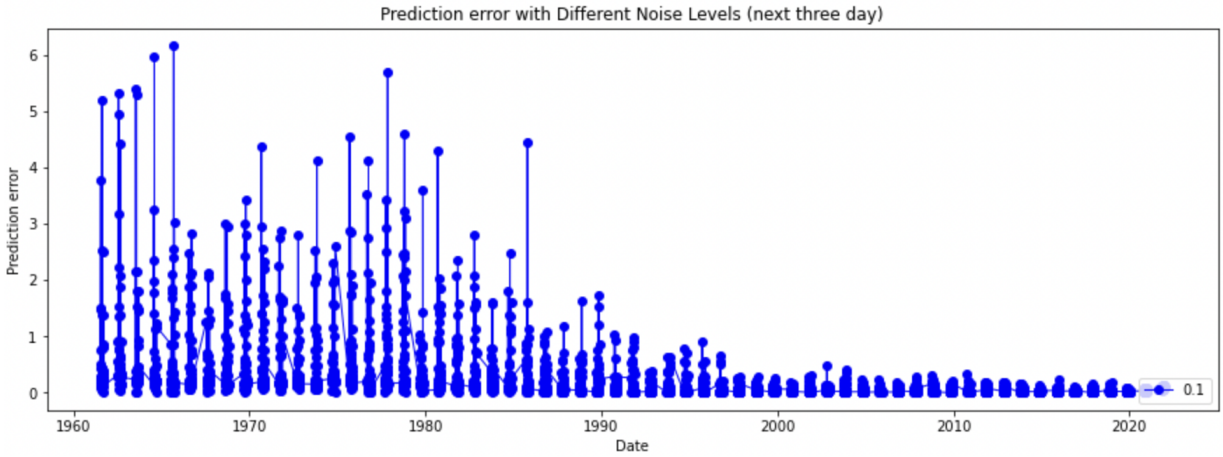

Fig 7.34: Prediction Error vs Date graph for CNN+RNN with 0.1 noise level

Graphs for Next 4 Data points in the Dataset

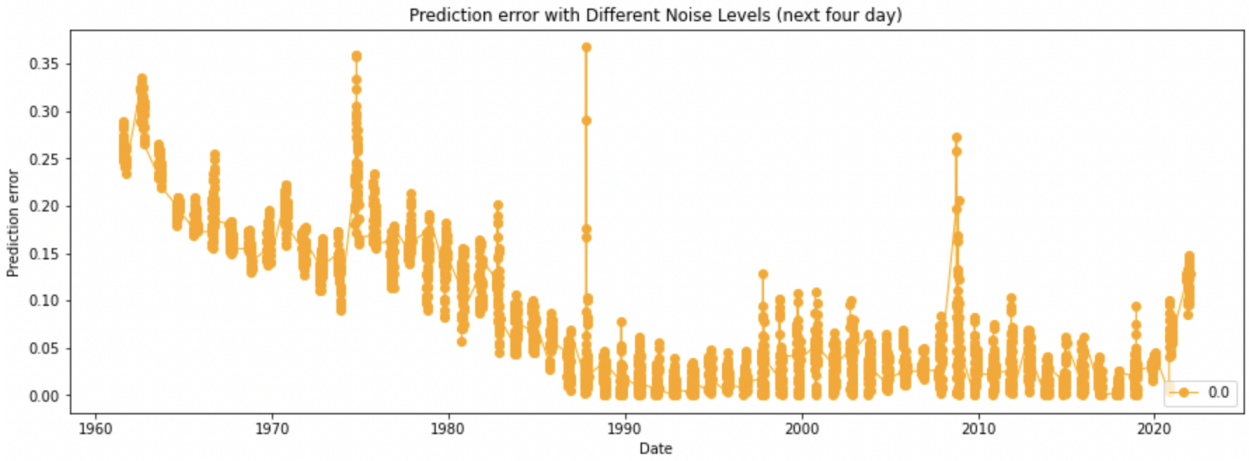

Fig 7.35: Prediction Error vs Date graph for CNN+RNN with 0 noise level

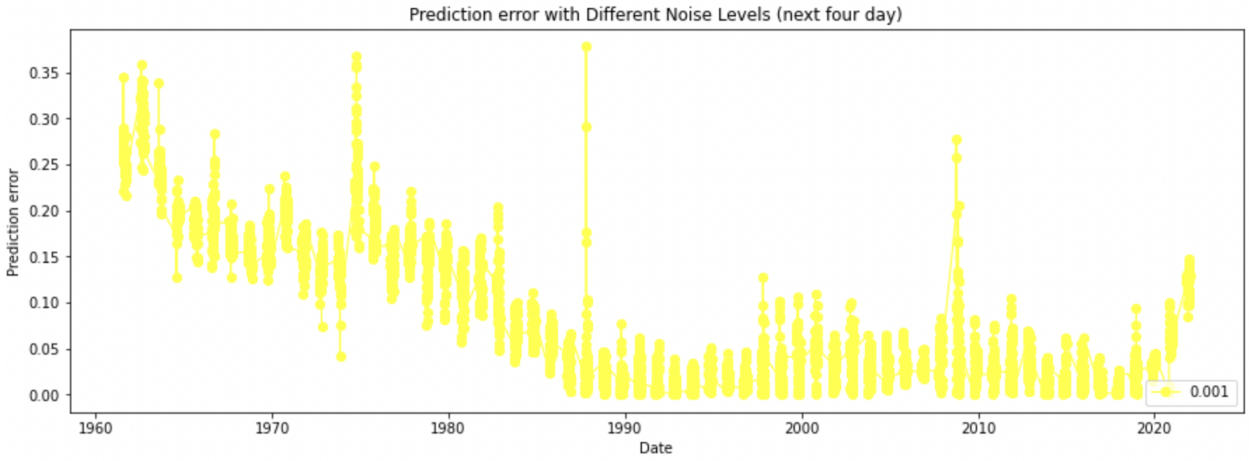

Fig 7.36: Prediction Error vs Date graph for CNN+RNN with 0.001 noise level

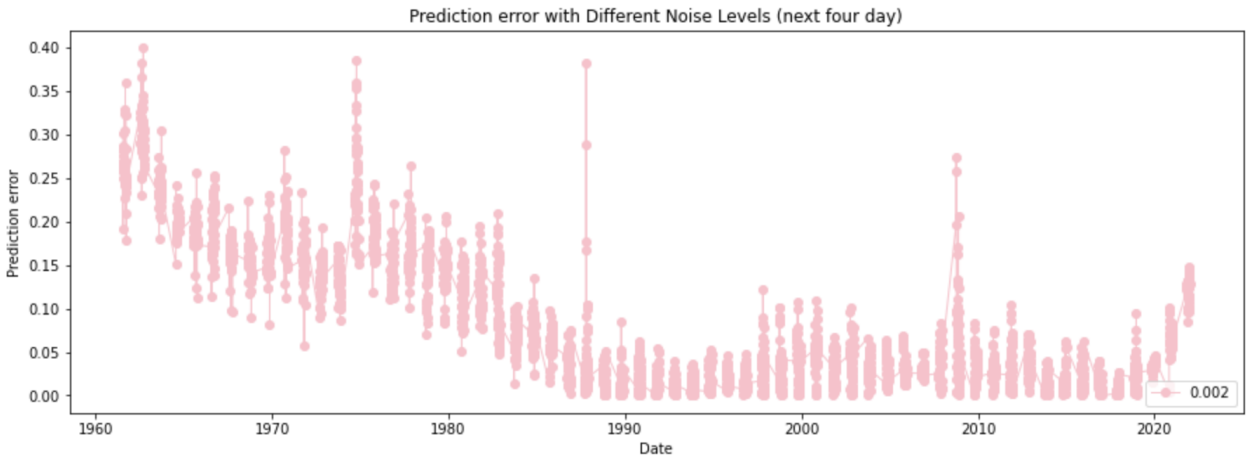

Fig 7.37: Prediction Error vs Date graph for CNN+RNN with 0.002 noise level

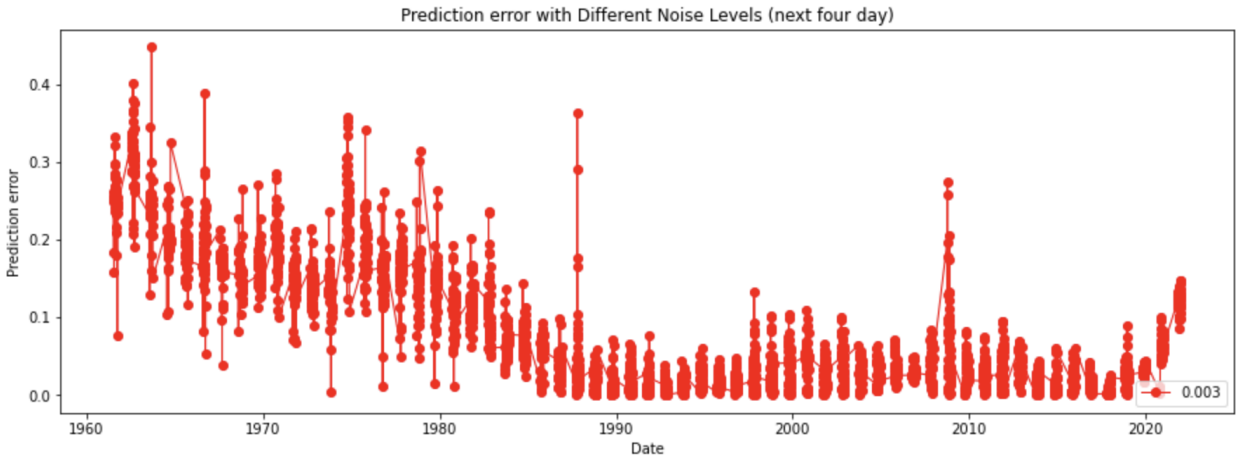

Fig 7.38: Prediction Error vs Date graph for CNN+RNN with 0.001 noise level

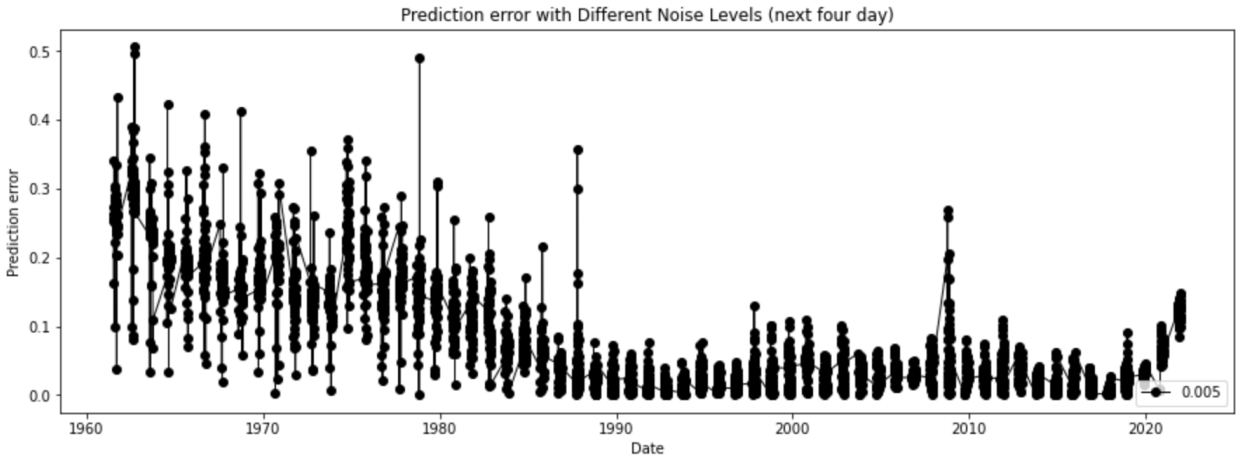

Fig 7.39: Prediction Error vs Date graph for CNN+RNN with 0.005 noise level

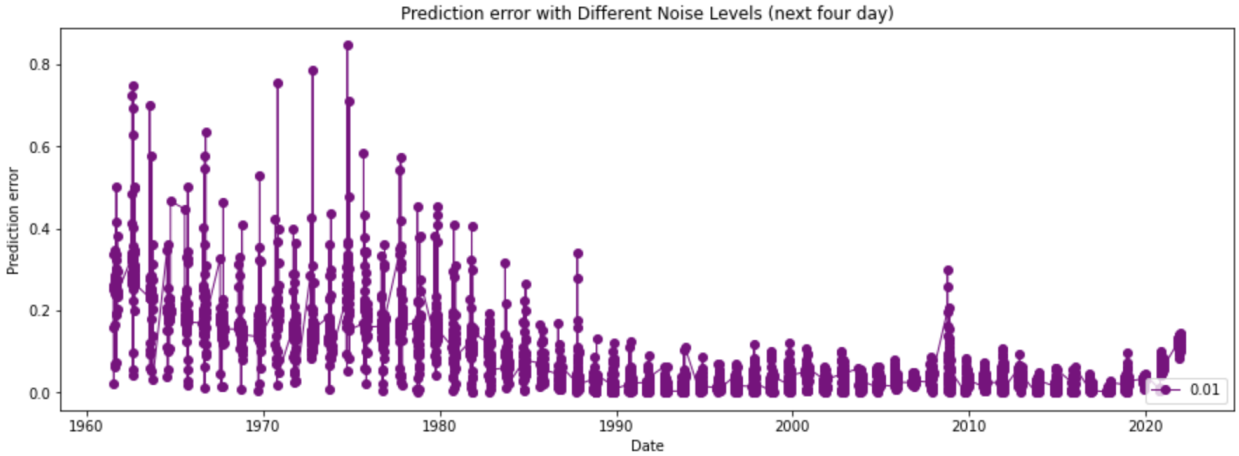

Fig 7.40: Prediction Error vs Date graph for CNN+RNN with 0.01 noise level

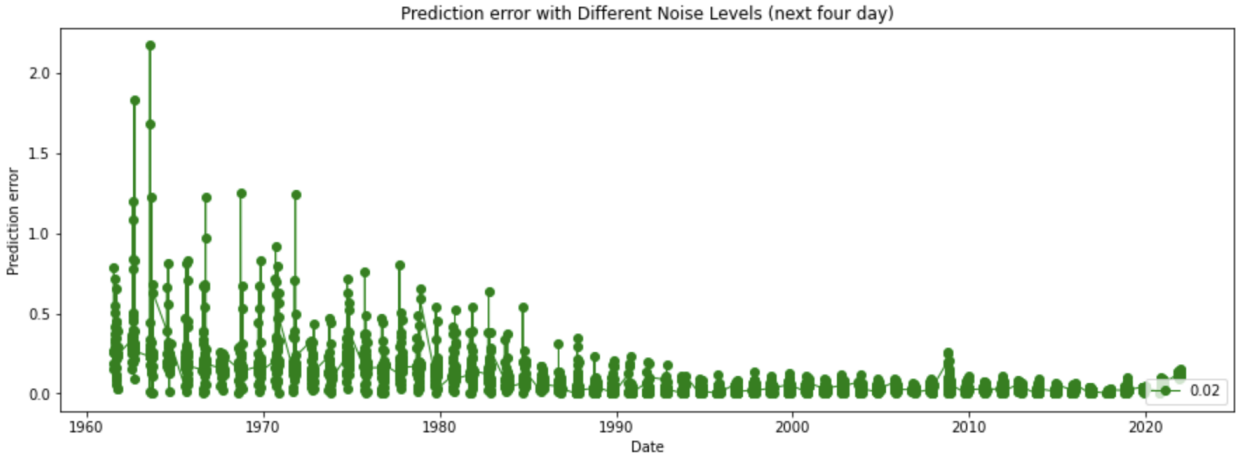

Fig 7.41: Prediction Error vs Date graph for CNN+RNN with 0.02 noise level

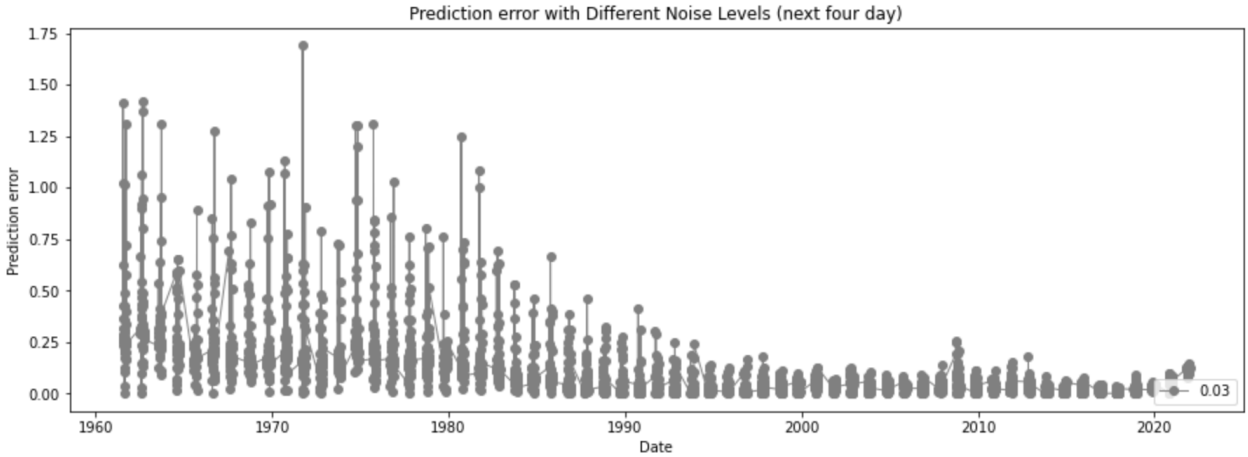

Fig 7.42: Prediction Error vs Date graph for CNN+RNN with 0.03 noise level

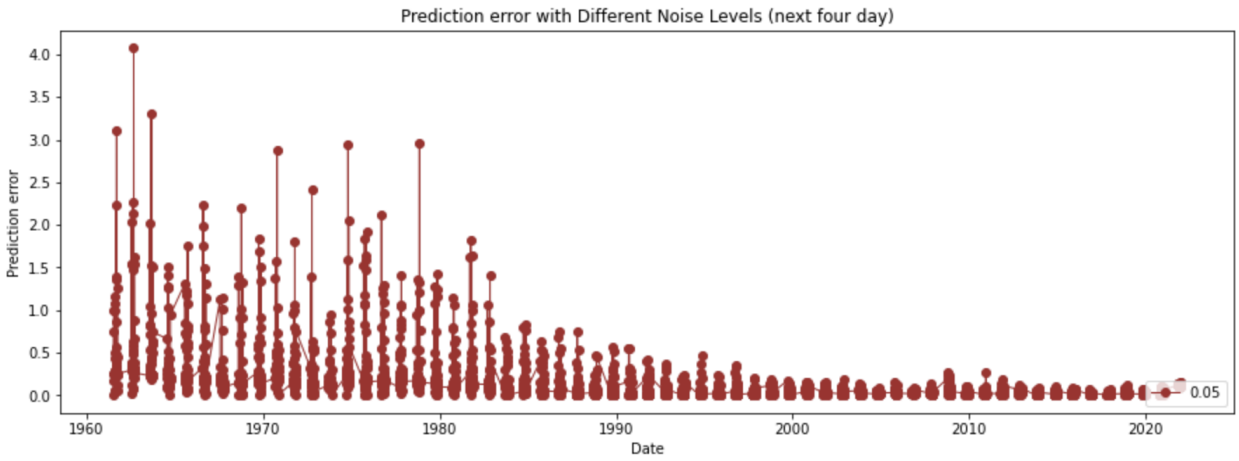

Fig 7.43: Prediction Error vs Date graph for CNN+RNN with 0.05 noise level

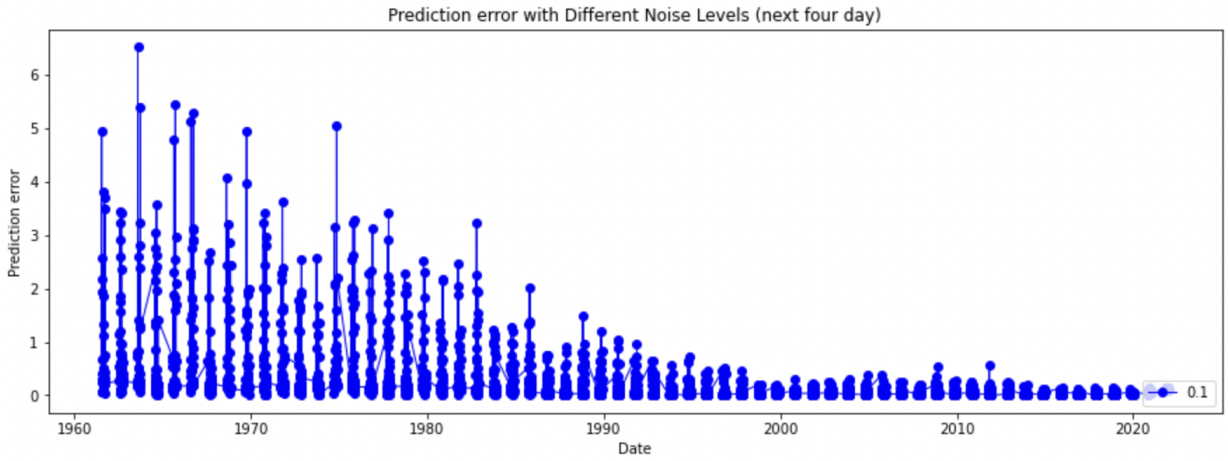

Fig 7.44: Prediction Error vs Date graph for CNN+RNN with 0.1 noise level

The above graphs represent the scatter-plot graph of Prediction Error versus Date with each noise standard deviation (stdev) value represented on a logarithmically-scaled abscissa whose values range from 0 to 0.1 in steps of 0.001, and with the Prediction Error values represented on the linearly-scaled ordinate.

As we increase the standard deviation of the gaussian noise we can observe that values of prediction error are slightly distorted. We can infer from above figures that for a standard deviation greater than or equal to 0.05 is affecting the prediction error values as there is a distortion in the prediction values due to the newly introduced gaussian noise.

8. Graphs of Money Generated with different models

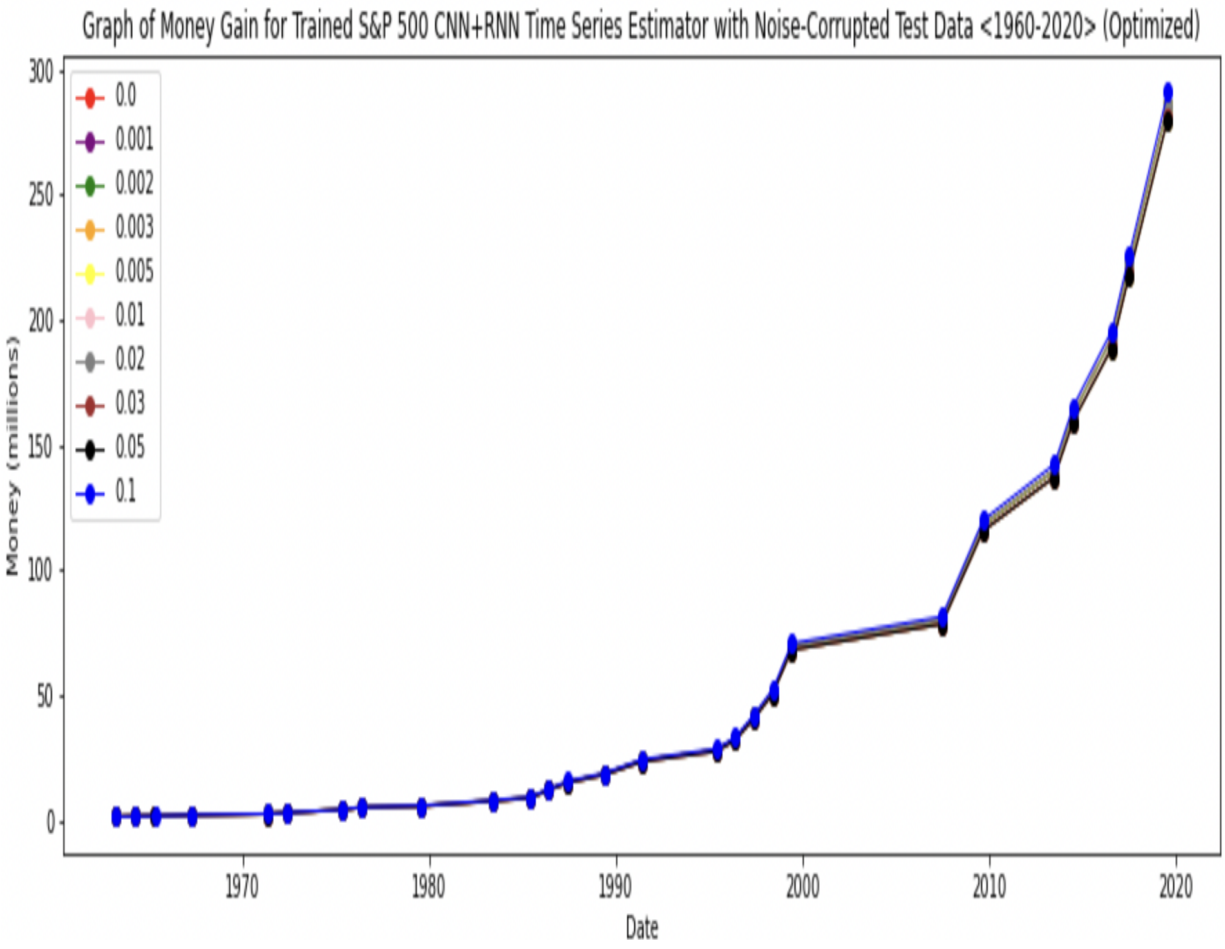

Fig 8.1: Money left at the end of trading activity using optimized CNN+RNN algorithm

9. Discussion

- 20% of the dataset was used for testing, i.e. first 200 days for training, and consecutive 53 data points for testing. We can vary the testing and training dataset percentage to get optimal prediction.

- The CNN+RNN model that we designed suffers from underfitting. In order to overcome the underfitting issue, we increase the number layer from 1 to 2, increase hidden unit size from 8 to 32. In addition, we increase the learning rate from 0.001 to 0.003 to speed up the learning process.

- During the training phase, we randomly added noise on the incoming data which helps our CNN+RNN generalize and perform similarly on noise corrupted data.

- We have only tested our CNN+RNN with 1and 2 hidden layers, experimenting with more hidden layers might improve performance of our neural network.

- In 2020, our model will be more than fourteen billion using CNN algorithm, which is much larger than using correlation algorithm in step 2.