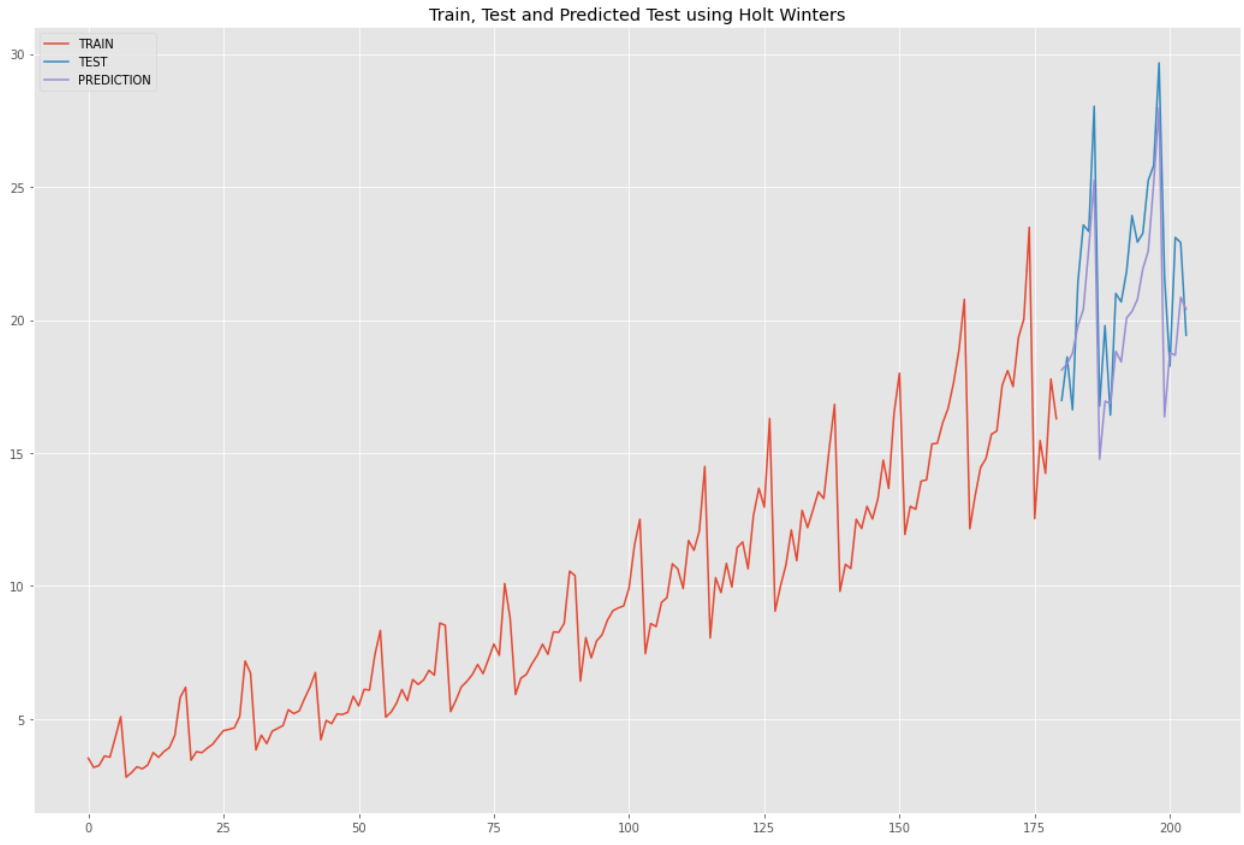
NAME : AMMAAR AHMAD
ROLL NO: 1801CS08
CS575 - End Semester

Colab Link -
https://colab.research.google.com/drive/1xkIQuJxQyl59MOyihfjB9MYKdBzdtMR6?usp=sharing

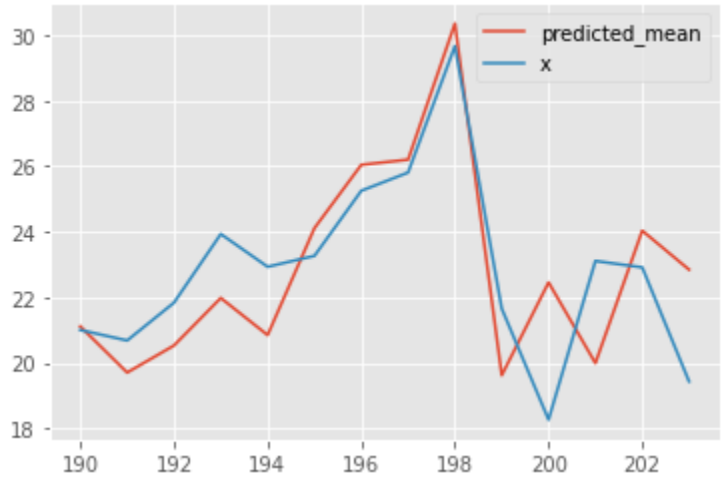
A10.csv - Dataset

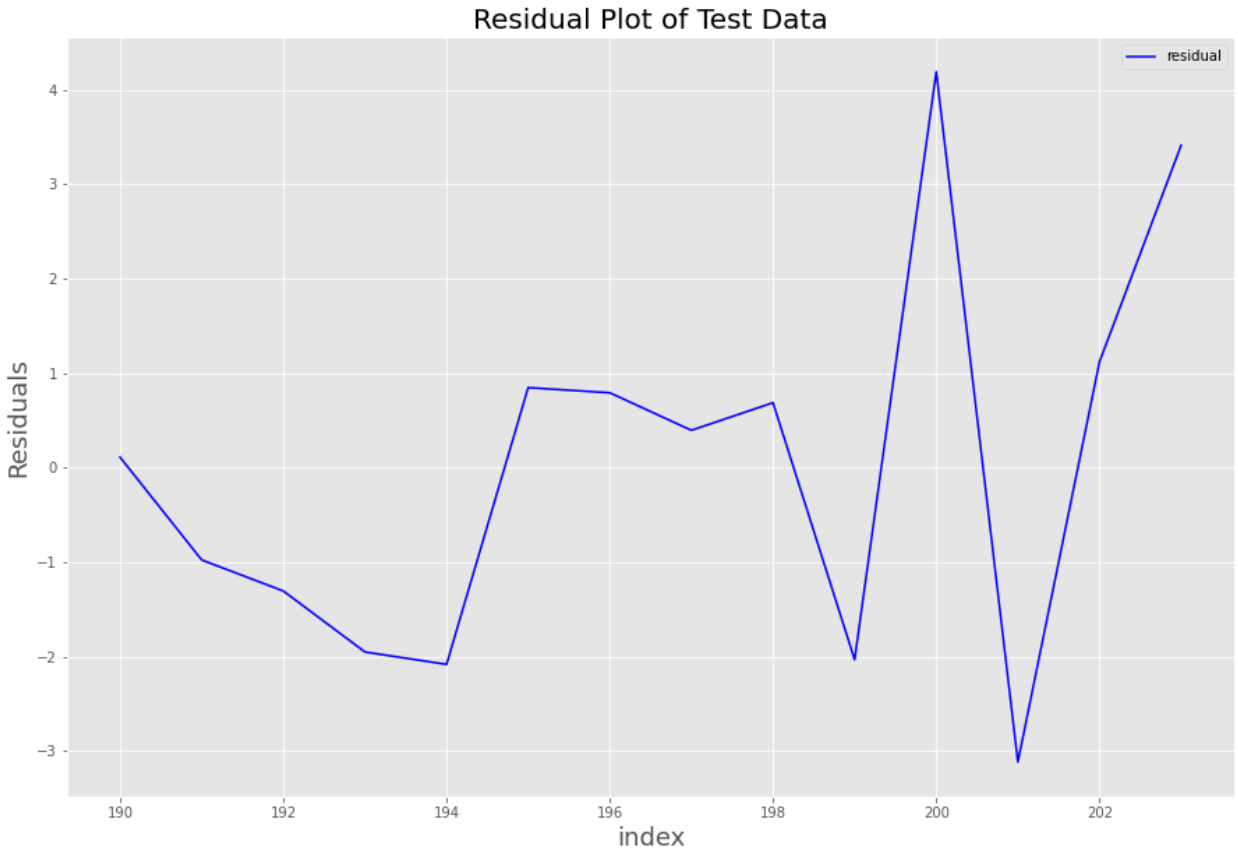
Q1.



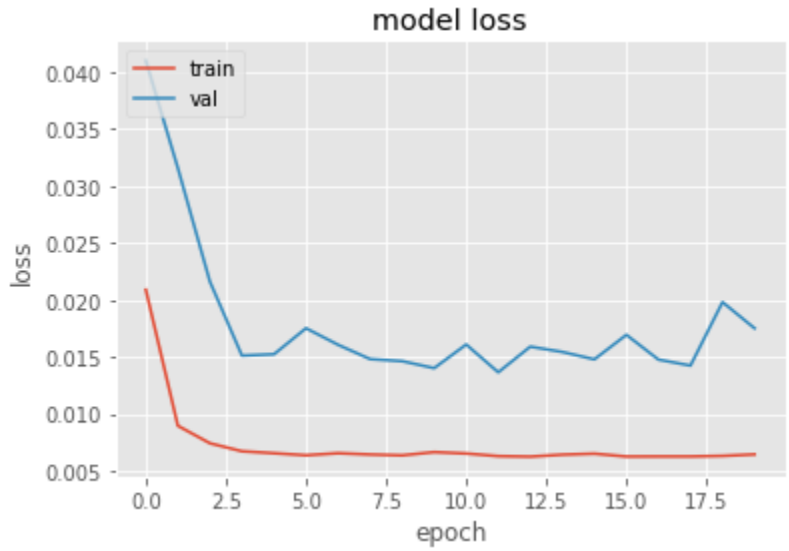
Holt Winter Model with Multiplicative Seasonal as well as Trend give best results RMSE = 5.80

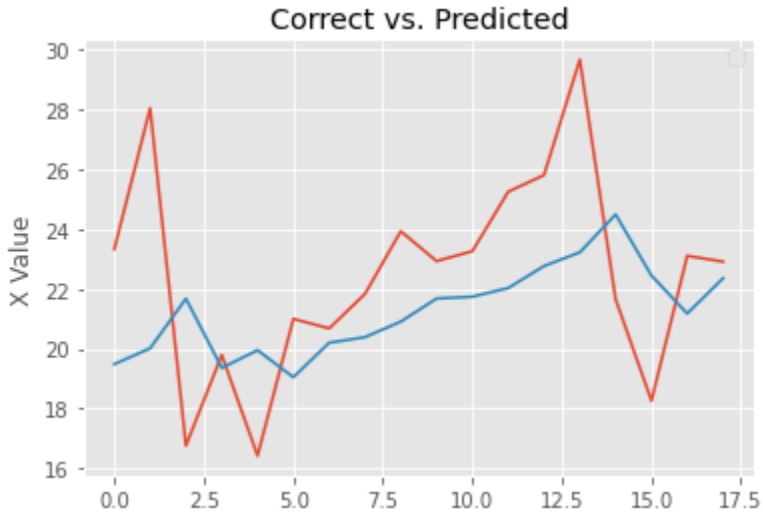
Q2. ARIMA (1,2,2) (1,0,1,12) gives the best results with RMSE = 2.019. Their residuals were random with p value = 0 indicating stationarity





Q3. The LSTM Model with 15 timestamps as input gave the best output among LSTM. TImestamps were kept 15 due to seasonal periods of 12 detected in SARIMA above.



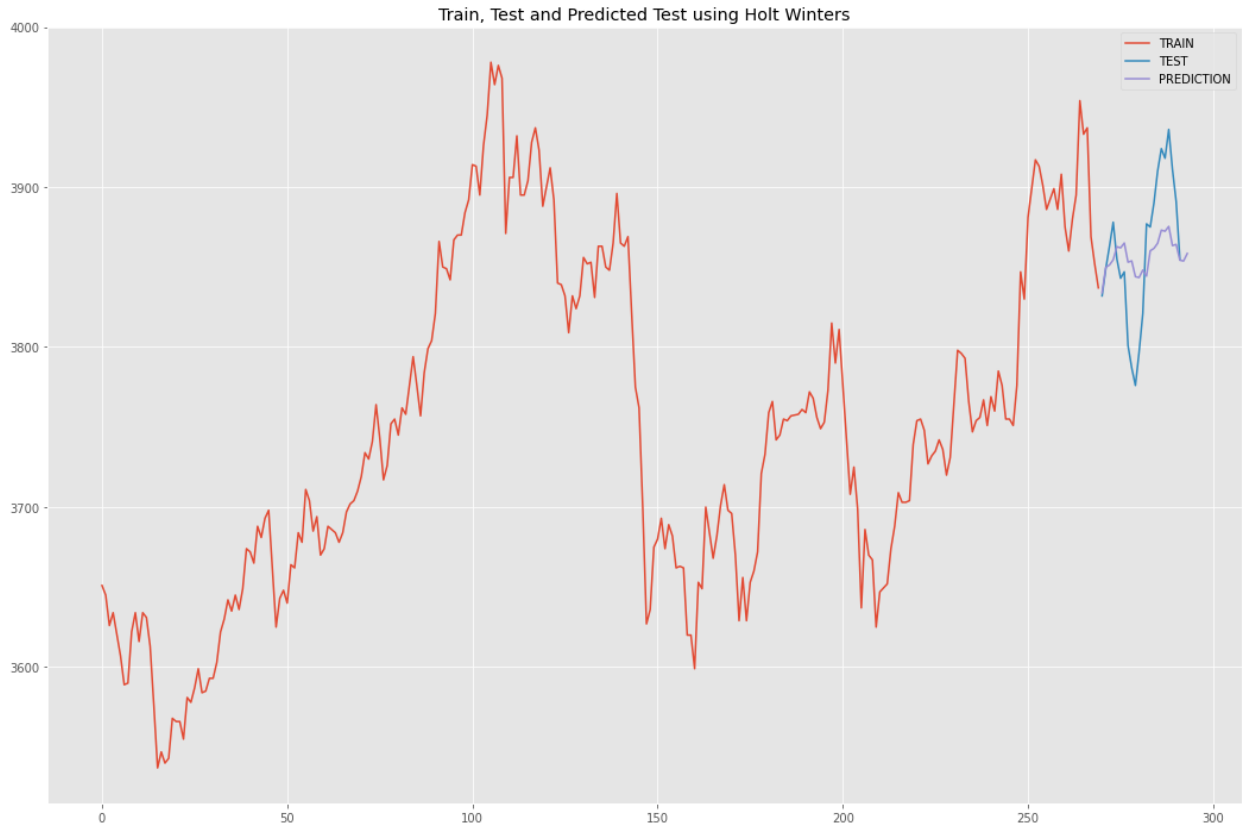


RMSE = 3.541196

Among the 3 SARIMA model performs best
Conclusion : SARIMA performs better that LSTM on this dataset
Explanation : Training Dataset is very little for LSTM to learn hence poor performance

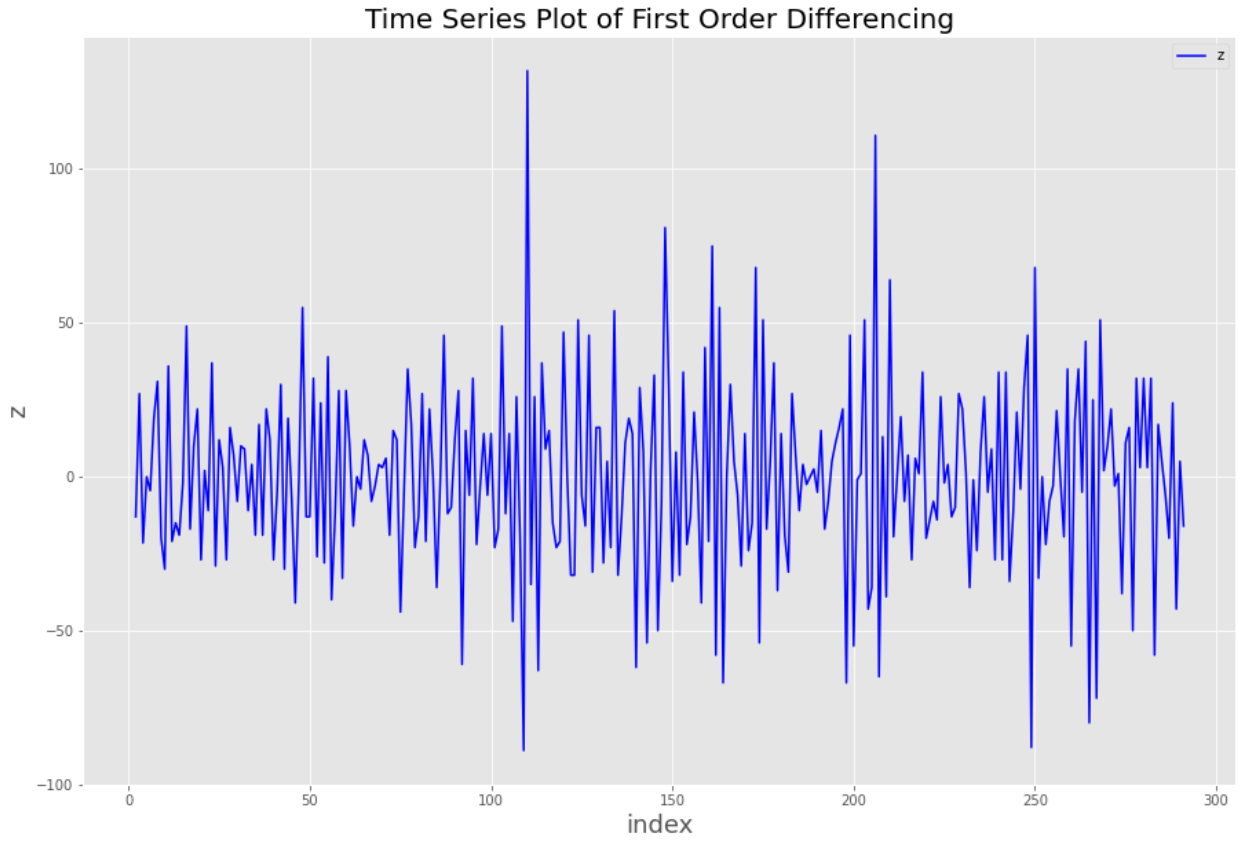
J0.csv - Dataset

Q1.

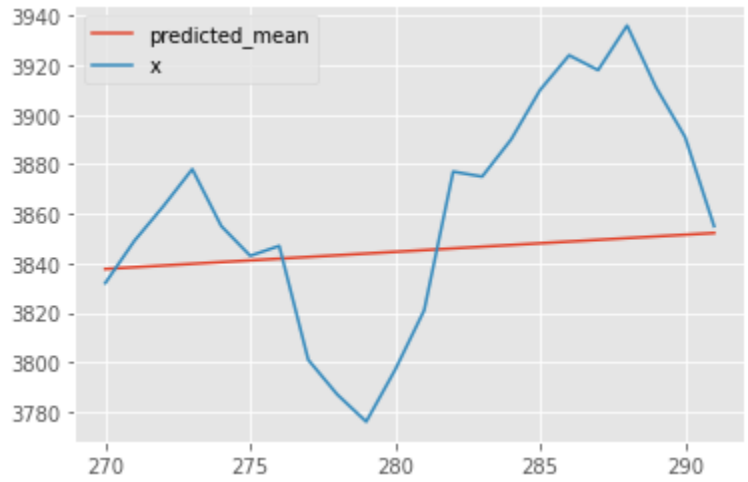


Holt WInter Additive Seasonal and Multiplicative Trend performs best

Q2.



First Order Differencing Plot. It pass the ADF Test
P value of first lag = 5.1879078792128 91e-29

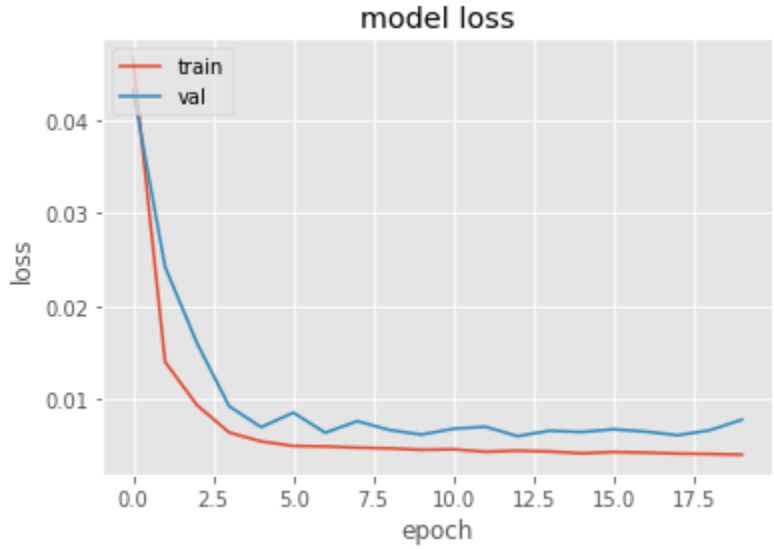


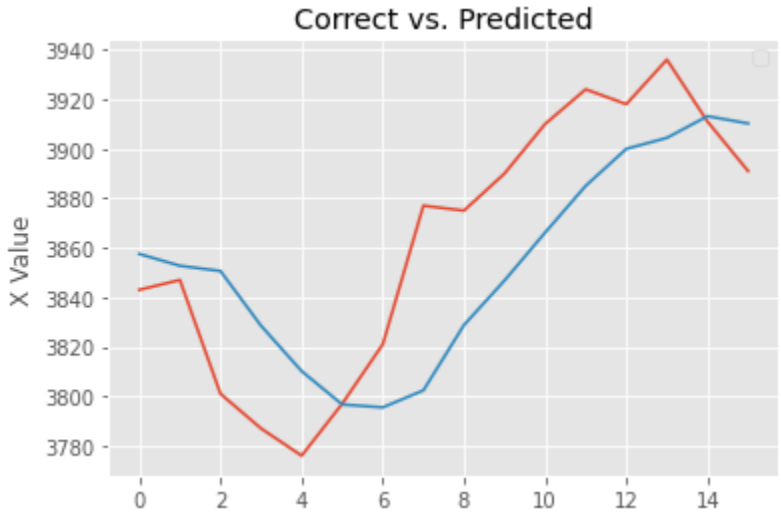
There was no dependency between the current timestamp and previous time stamp. It was random as predicted by the

auto_arima model. So ARIMA (0,1,0) was used to fit the training dataset. RMSE Loss = 45.2830

Q3

LSTM with lookback = 5 was trained with Adam Optimizer and MSE Loss was calculated. Training Epochs = 20



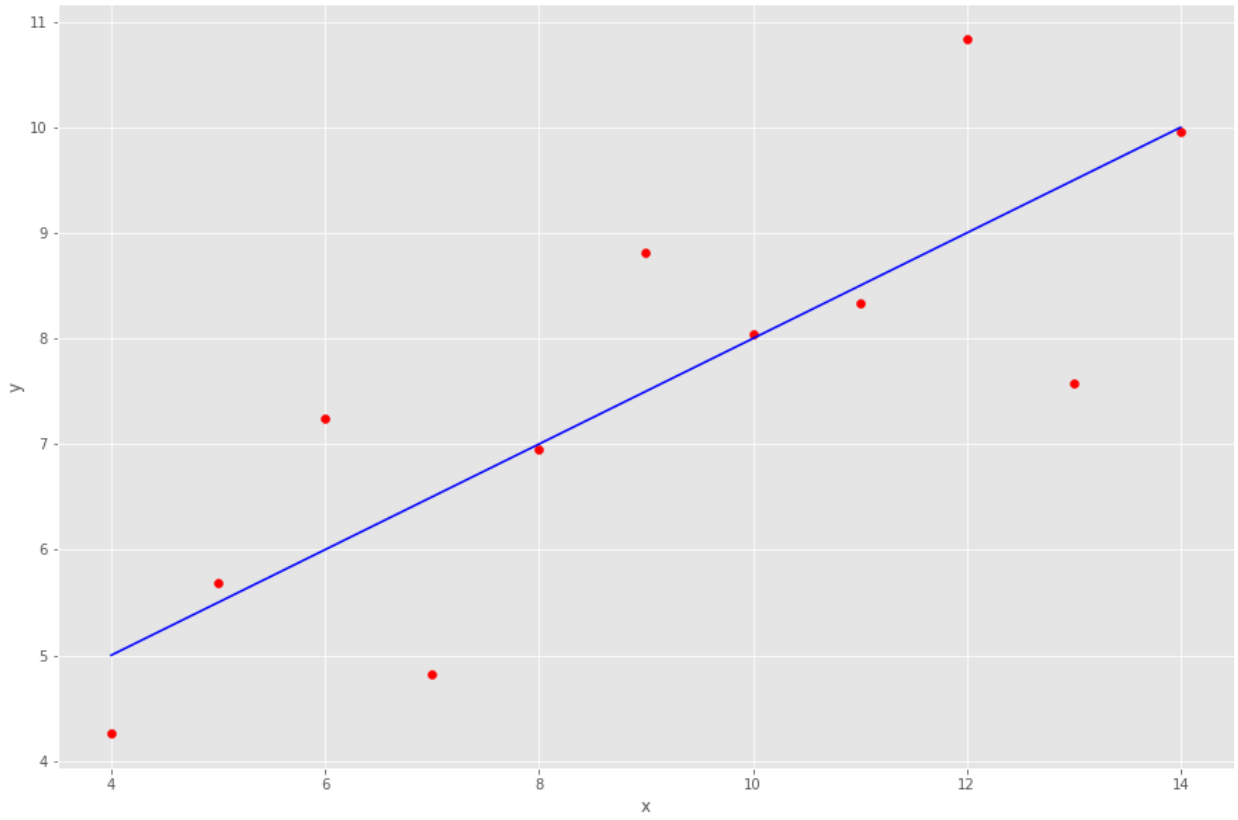


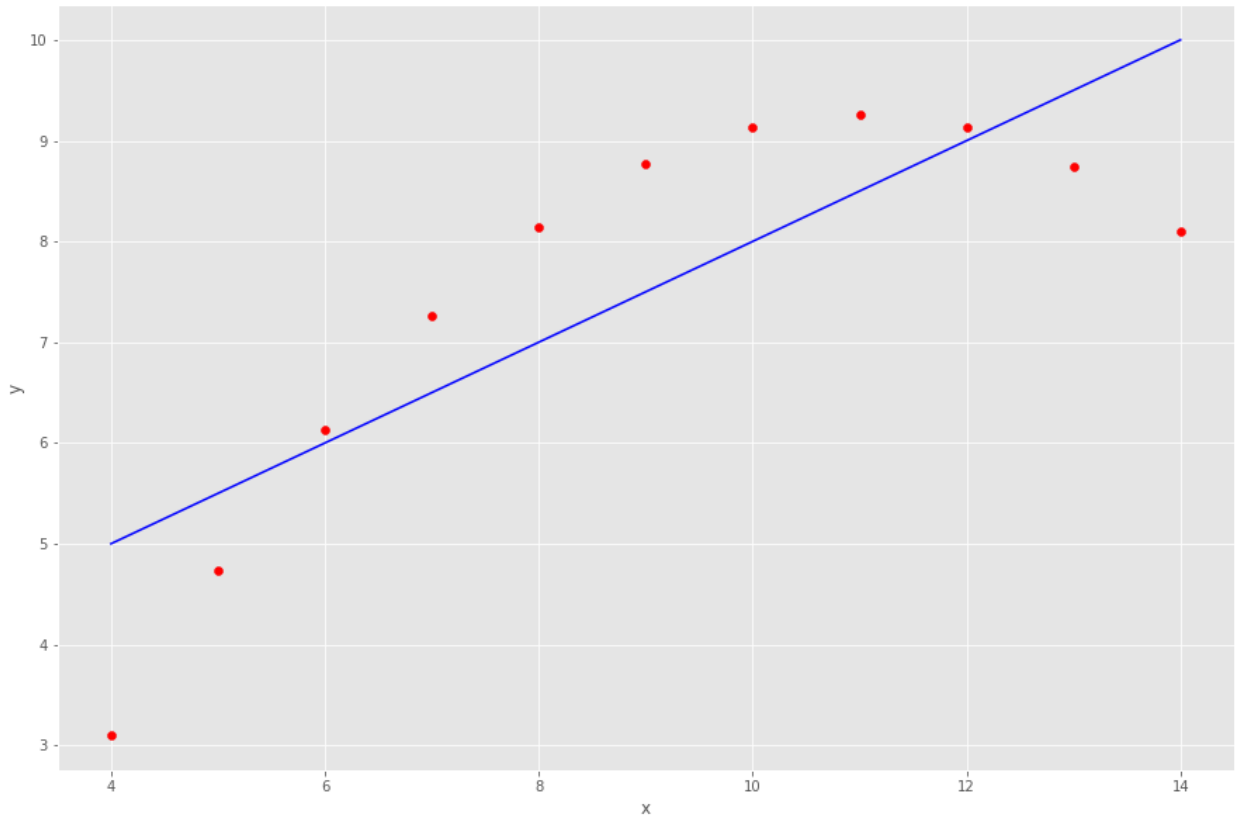
RMSE = 36.185

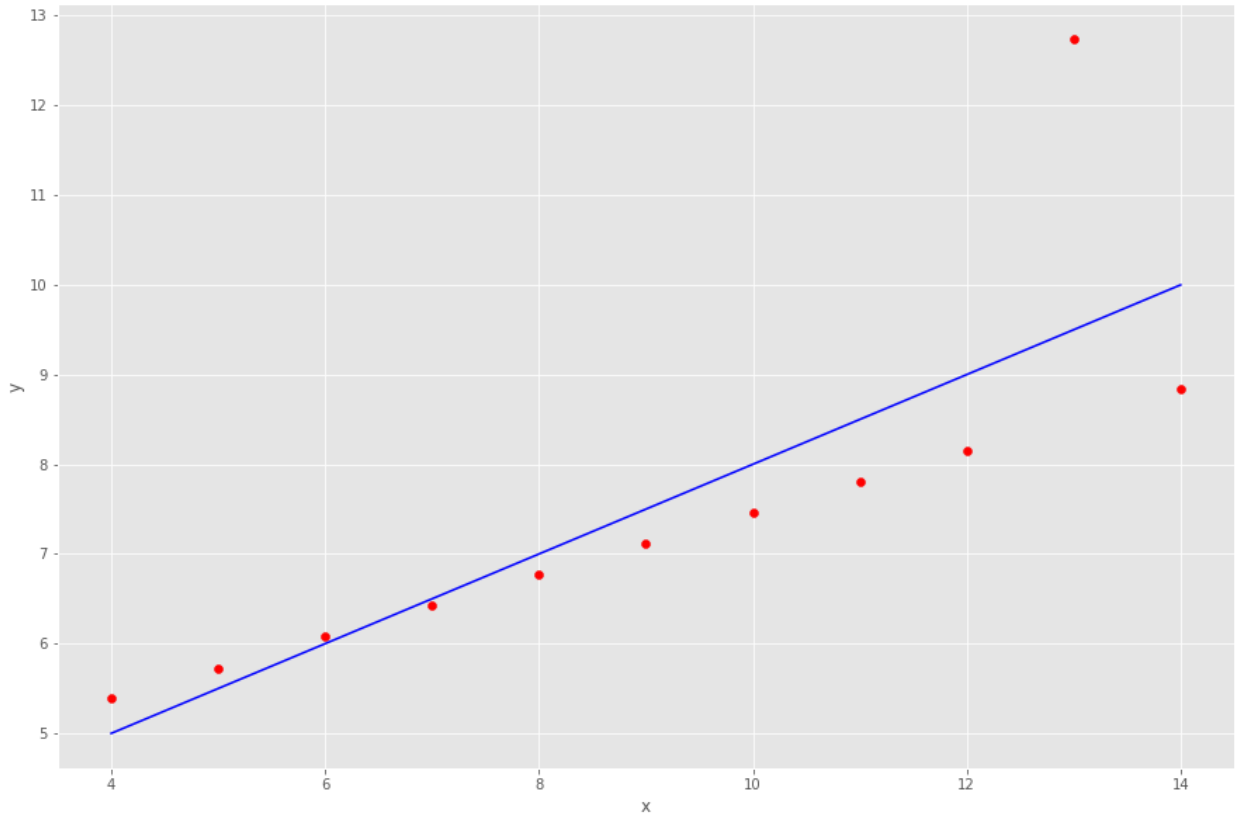
Among the 3. LSTM performs best as there is randomness in this dataset

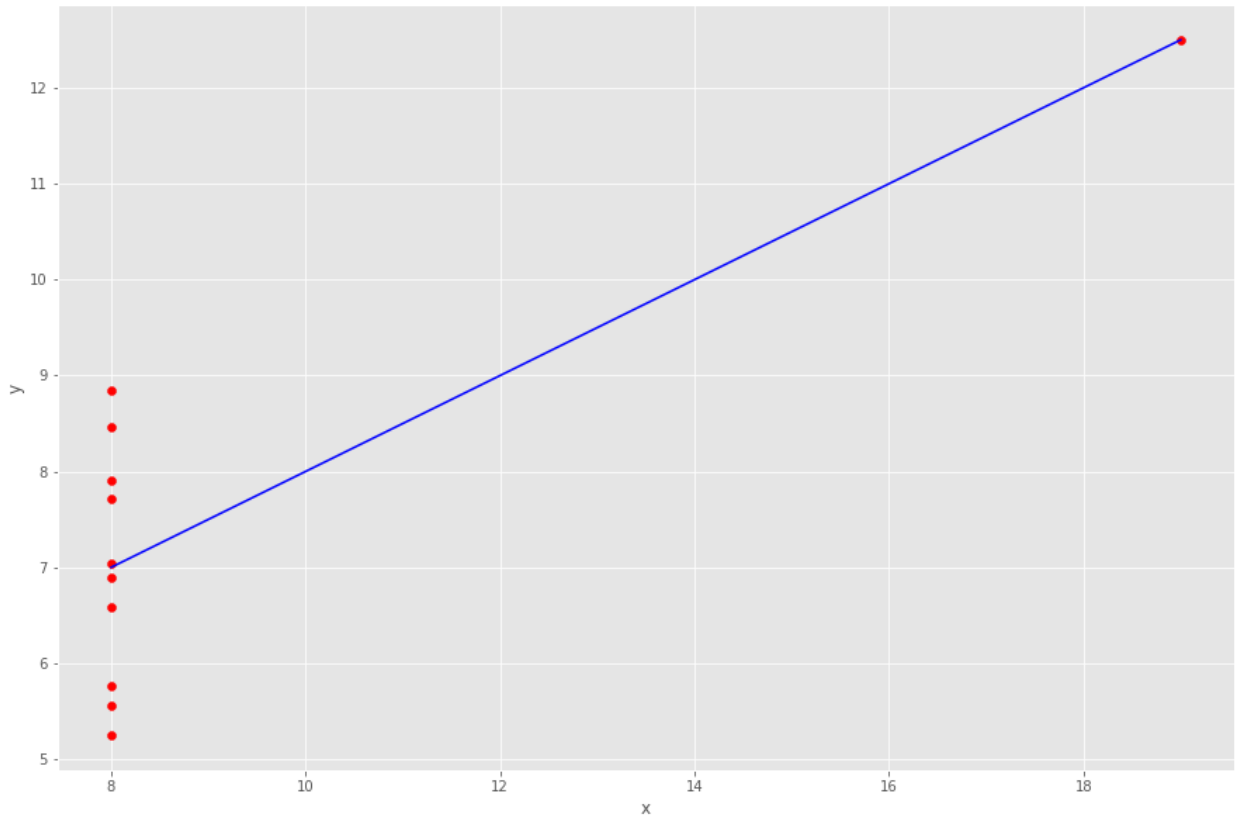
A4.csv - Dataset

	x1	y1	x2	y2	x3	y3	x4	y4
0	4	4.26	4	3.10	4	5.39	8	6.58
1	5	5.68	5	4.74	5	5.73	8	5.76
2	6	7.24	6	6.13	6	6.08	8	7.71
3	7	4.82	7	7.26	7	6.42	8	8.84
4	8	6.95	8	8.14	8	6.77	8	8.47
5	9	8.81	9	8.77	9	7.11	8	7.04
6	10	8.04	10	9.14	10	7.46	8	5.25
7	11	8.33	11	9.26	11	7.81	8	5.56
8	12	10.84	12	9.13	12	8.15	8	7.91
9	13	7.58	13	8.74	13	12.74	8	6.89
10	14	9.96	14	8.10	14	8.84	19	12.50


11
10
9
8
7
6
5
4
y
4
6
8
10
12
14
x


10
9
8
7
6
5
4
3
y
4
6
8
10
12
14
x


13
12
11
10
9
8
7
6
5
y
4
6
8
10
12
14
x


12
11
10
9
8
7
6
5
y
8
10
12
14
16
18
x

Custom Linear Regression was implemented.

```
def calculate(x, y):
  mean_x = np.mean(x)
  mean_y = np.mean(y)
  cov = np.cov(x, y=y, bias = True)
  var_x = cov[0][0]
  var_y = cov[1][1]
  cov_xy = cov[0][1]
  return mean_x, mean_y, var_x, var_y, cov_xy
```

```
[106]
def regression(x, y):
  mean_x, mean_y, var_x, var_y, cov_xy = calculate(x,y)
  beta = cov_xy/var_x
  alpha = mean_y - beta*mean_x
  y_hat = beta*x + alpha
  print(f'The Coefficient of Regression = {beta}')
  print(f'The intercept on y axis = {alpha}')
  ssy = np.sum((y - mean_y)**2)
  ssr = np.sum((y_hat - mean_y)**2)
  r2 = ssr/ssy
  print(f'The R2 coefficient = {r2}')
  plt.figure(figsize = (15,10))
  plt.scatter(x, y, color = 'red')
  plt.plot(x, y_hat, 'blue')
  plt.xlabel('x')
  plt.ylabel('y')
  plt.show()
```

Inbuilt SK Learn Linear Regression was used.
Both gave the exact same results for all 4 subset of data.

This Dataset (x4, y4) has one outlier and thus line is not a good fit for this dataset. Best Regression line should have been x = 8 (Parallel to Y axis)