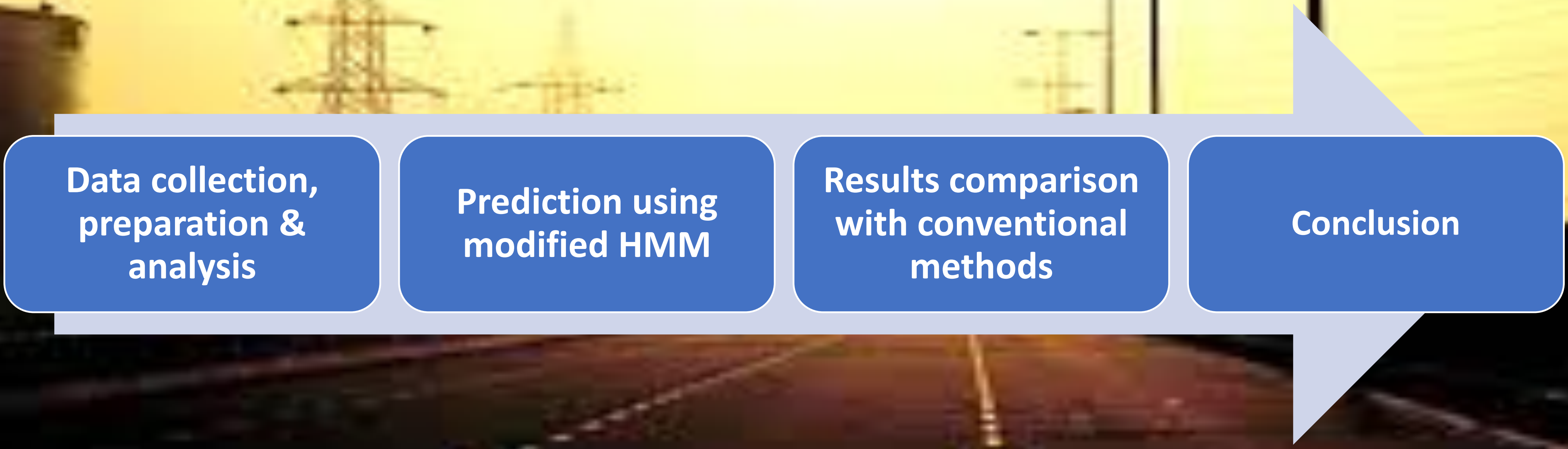
Short-Term Electricity Demand Prediction Using Modified Hidden Markov Model

Neelu Nagpal[a], Pierluigi Siano[b]; Poras Khetarpal[c] ; Simarn Kaur[c] ; Hardik Nagpal[d*]

[a]Maharaja Agrasen Institute of Technology, India [b]University of Salerno, Italy

[c]Bharati Vidyapeeth's College of Engineering, India [d]ZS Associates, India

*hardik.nagpal5@gmail.com


Data collection, preparation & analysis
Prediction using modified HMM
Results comparison with conventional methods
Conclusion

Data Collection & Analysis

- Data collected from state load dispatch center, Delhi, India.
- **Data Size**: Hourly electricity demand of 24 months from January 2018 to December 2019.
- **Data Processing**: Divided into training and learning data (10,000 and 4,000 dataset points).

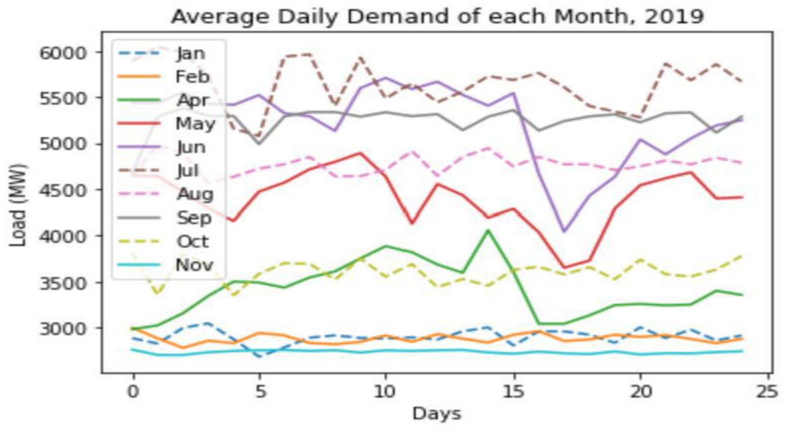


Figure 2: Average Daily Demand of Each Month

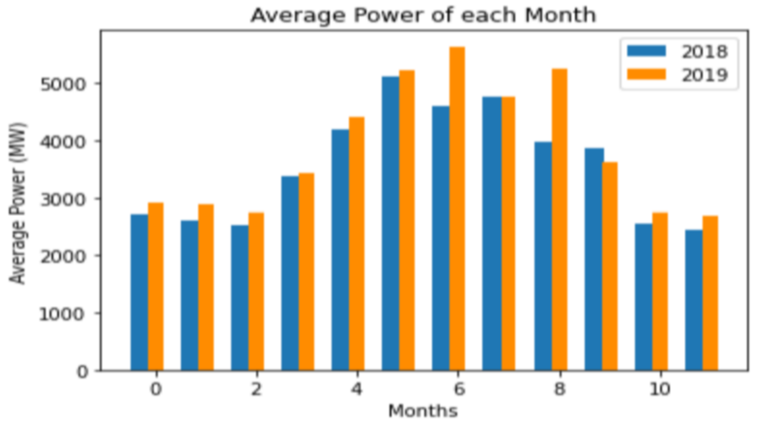


Figure 3: Total Average Power of Each Month

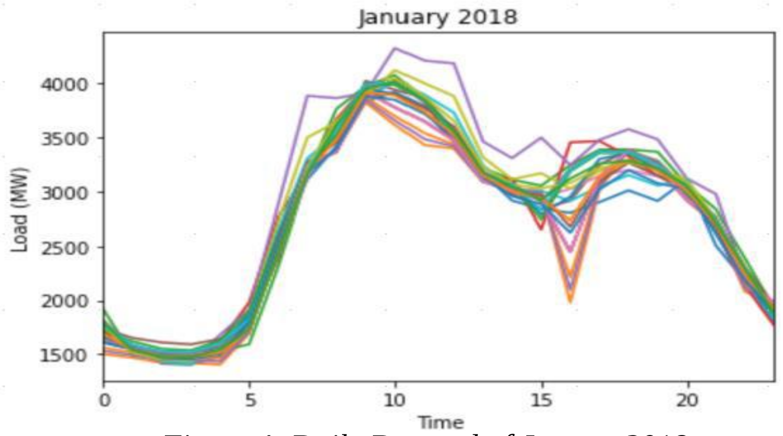


Figure 4: Daily Demand of January2018

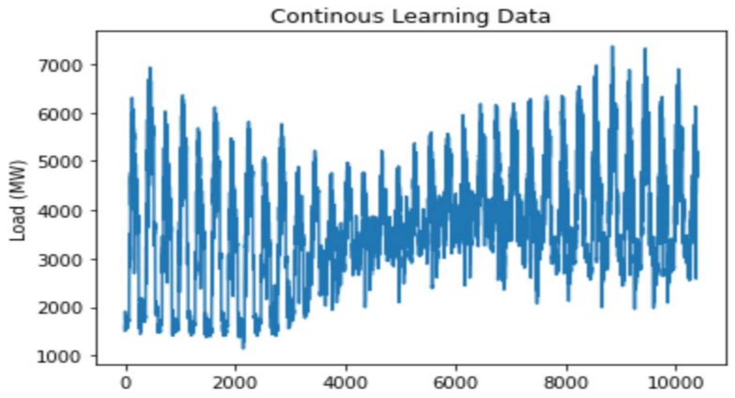


Figure 5: Daily Demand of January 2019

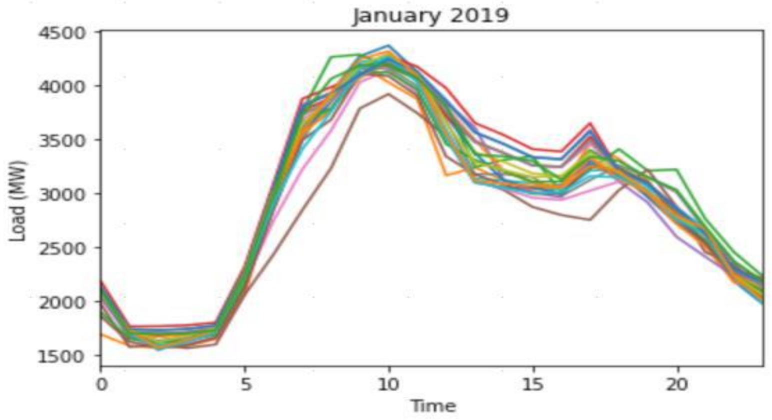


Figure 1:
Hourly Electricity Demand of Delhi

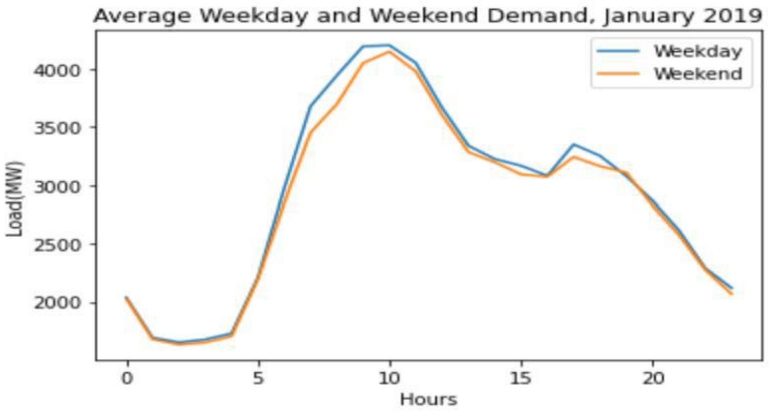
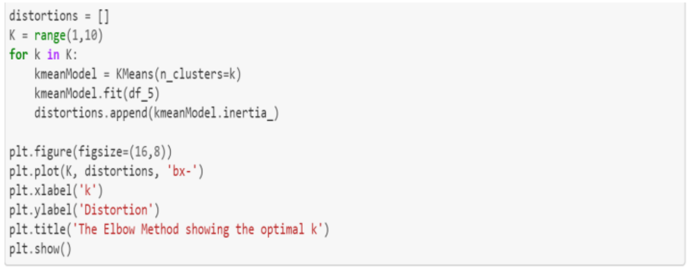


Figure 6: Weekend vs Weekday Demand

Hidden Markov Model

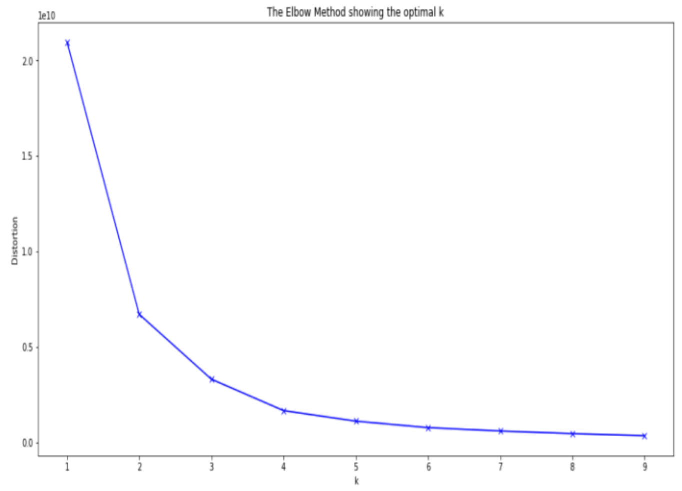
- Compute Likelihood – Forward Algorithm
- Viterbi Algorithm : A dynamic programming **algorithm** for finding the most likely sequence of hidden states—called the **Viterbi** path—that results in a sequence of observed events, especially in the context of Markov information sources and hidden Markov models (HMM).
- Baum-Welch Algorithm : A special case of the EM **algorithm** used to find the unknown parameters of HMM. It makes use of the forward-backward **algorithm** to compute the statistics for the expectation step.

HMM Output Parameter

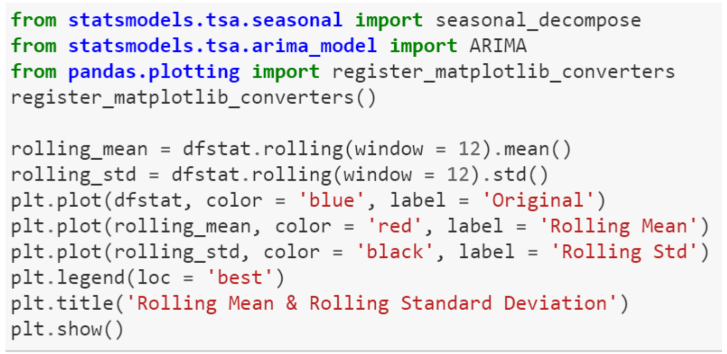


```
distortions = []
K = range(1,10)
for k in K:
    kmeanModel = KMeans(n_clusters=k)
    kmeanModel.fit(df_5)
    distortions.append(kmeanModel.inertia_)

plt.figure(figsize=(16,8))
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```

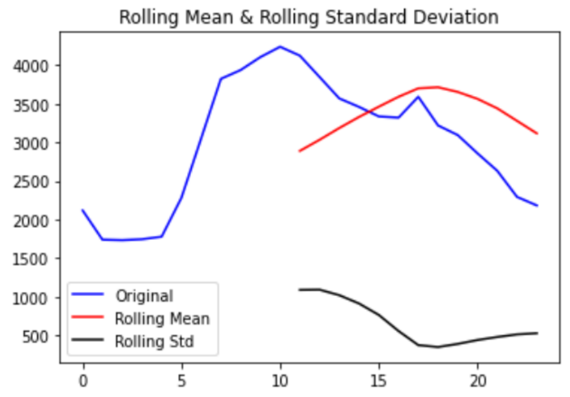


Non-Stationery Test

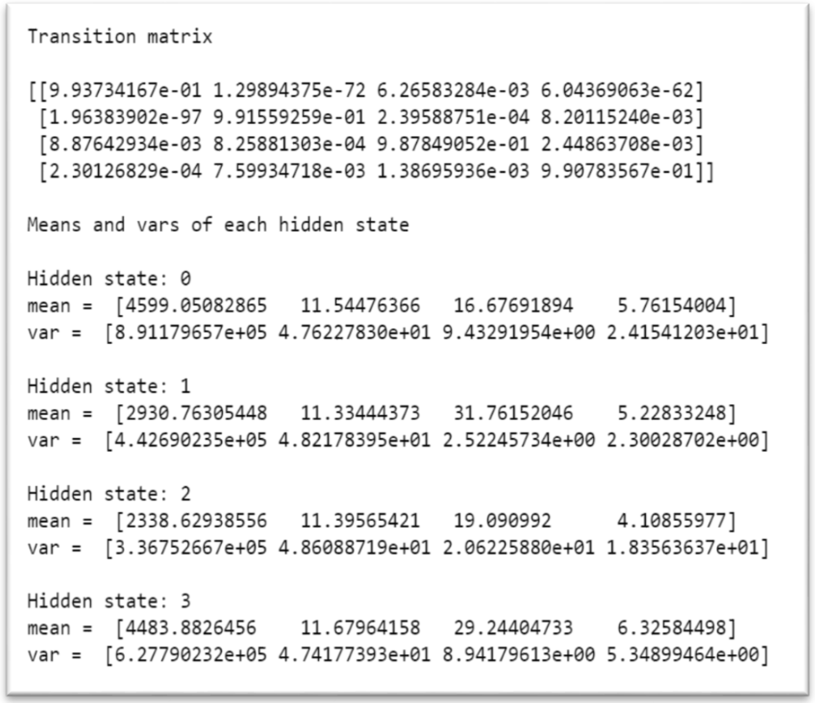


```
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima_model import ARIMA
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()

rolling_mean = dfstat.rolling(window = 12).mean()
rolling_std = dfstat.rolling(window = 12).std()
plt.plot(dfstat, color = 'blue', label = 'Original')
plt.plot(rolling_mean, color = 'red', label = 'Rolling Mean')
plt.plot(rolling_std, color = 'black', label = 'Rolling Std')
plt.legend(loc = 'best')
plt.title('Rolling Mean & Rolling Standard Deviation')
plt.show()
```



Obtained State transition matrix



```
Transition matrix

[[9.93734167e-01 1.29894375e-72 6.26583284e-03 6.04369063e-62]
 [1.96383902e-97 9.91559259e-01 2.39588751e-04 8.20115240e-03]
 [8.87642934e-03 8.25881303e-04 9.87849052e-01 2.44863708e-03]
 [2.30126829e-04 7.59934718e-03 1.38695936e-03 9.90783567e-01]]

Means and vars of each hidden state

Hidden state: 0
mean =  [4599.05082865    11.54476366    16.67691894     5.76154004]
var =  [8.91179657e+05 4.76227830e+01 9.43291954e+00 2.41541203e+01]

Hidden state: 1
mean =  [2930.76305448    11.33444373    31.76152046     5.22833248]
var =  [4.42690235e+05 4.82178395e+01 2.52245734e+00 2.30028702e+00]

Hidden state: 2
mean =  [2338.62938556    11.39565421    19.090992       4.10855977]
var =  [3.36752667e+05 4.86088719e+01 2.06225880e+01 1.83563637e+01]

Hidden state: 3
mean =  [4483.8826456     11.67964158    29.24404733     6.32584498]
var =  [6.27790232e+05 4.74177393e+01 8.94179613e+00 5.34899464e+00]
```

Hidden Markov Model with MCMC

- HMM is modified for better performance using Bayesian Inference. Markov Chain Monte Carlo (MCMC) method is used to obtain the parameters of the model in place of the commonly used Baum-Welch algorithm for better performance.
- The prediction is done by employing modified Hidden Markov Model (HMM) considering electricity demand of short span as stochastic and non-stationary time series.
- The applicability of HMM to the present work is based on the hypothesis that it is a suitable model with limited amount of data.

Results

- LSTM model gave the best accuracy of 97.765 percent and then modified HMM performed with the accuracy of 97.446. HMM has comparable error metrics to LSTM with a higher MAE and lowest RMSE.
- The LSTM is known to produce accurate forecasts and perform well but took way greater amount of computation time and processing power as compared to HMM and modified HMM proposed model. Given the size of our training data and the system upon which the program was run, 100 iterations of LSTM took 30-40 minutes to train while 100 MCMC steps took 8 seconds per iteration making 13 minutes in total.

Figure: Modified HMM Forecasted Values

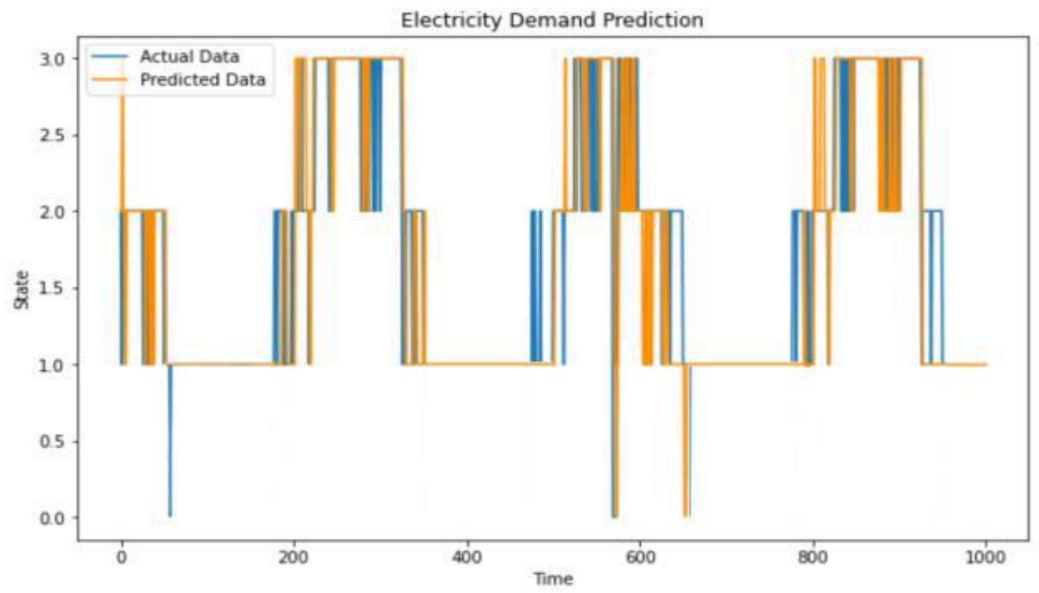


Table 1. Comparative Analysis

	AR	ARIMA	EMA	LSTM	HMM	Mod. HMM
MAE	2.557	1.76	1.357	0.177	1.016	0.616
RMSE	3.094	2.849	1.2	0.815	0.371	0.165
% Accuracy	86.675	91.138	93.108	97.765	96.411	97.446

Conclusion

In this work, modified HMM model has been proposed for forecasting short term electricity demand of Delhi city, India. HMM has been modified using Bayesian inference and MCMC sampling to obtain the model parameters. From the results, it can been concluded that the overall performance of HMM model and its modification is better in predicting the future electricity forecast values than all other statistical methods if accuracy, computation time and data set size are considered together. Both methods using modified HMM and LSTM have better error metrics but LSTM is found to be more time consuming, computationally complex as compared to the HMM model. Although, MCMC sampling takes considerable computational time as compared to Baum-Welch HMM, but it is suggested to use the proposed modified HMM in case of limited amount of available training data to have less complexity and overall better performance.

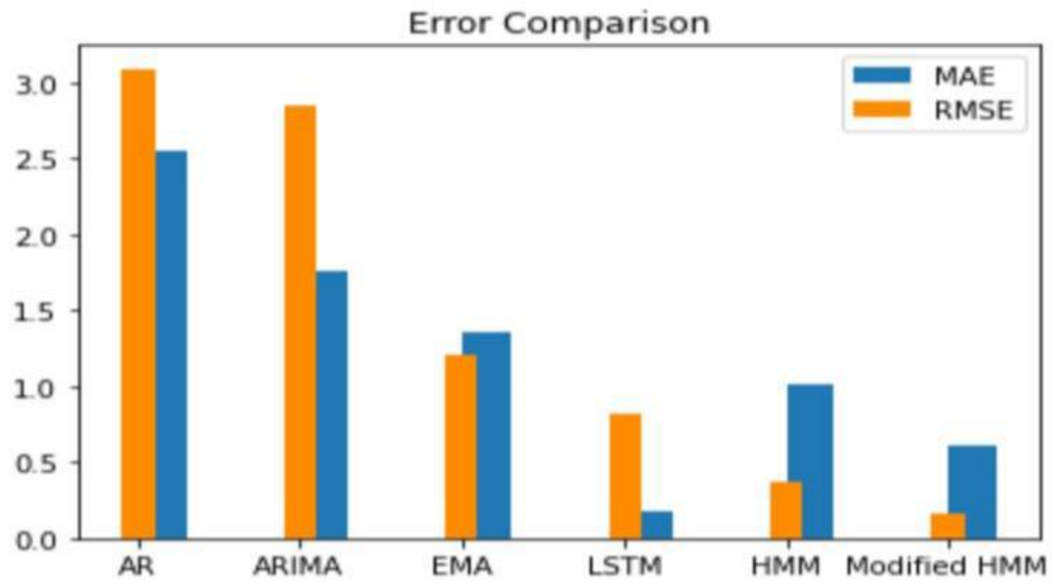


Figure: MAE and RMSE Comparison