Electricity Demand Forecasting and Analysis Using Time-Series Algorithms

CAPSTONE PROJECT - B.Tech. (Information Technology)

By SOUBHIK SINHA (19BIT0303)

Under the guidance of
Dr. (Prof.) Bimal Kumar Ray
Professor (Higher Academic Grade), Dept. of IT, SITE

ABSTRACT

The ability to estimate demand is crucial for the power supply sector. In order to fulfil future demand, power production plants must be scheduled far in advance because electricity cannot be stored. Time series techniques on the historical demand series have been employed for demand prediction up till now in cases when online information about the external circumstances is not accessible. This project proposes the approach of using the appropriate Time-Series algorithm among LSTM (Long Short-Term Memory), ARIMA (Auto-Regressive Integrated Moving Average) and STM (Short Term Memory - RNN) Neural Networks on the basis of error rate (RMSE – Root Mean Squared Error) and R^2-score, as the evaluation metrics, on the dataset acquired from website(s) regulated by the Ministry of Power, Govt. of India. Later we showed the prediction values of power production for 10 blocks of 15 minutes interval for the next day. Afterwards we compare the prediction results to declare the most appropriate one among the algorithms chosen. Also, I showed how error rate can be reduced by taking an extremely rudimentary step towards hybridization.

PROBLEM DEFINITION

The production of electricity has been exponentially growing as compared to past decade – thanks to the rapid growth of industrialization. But nation’s power plants still don’t have an efficient mechanism to predict how much power to produce, say, for the next day. The current mechanism is pretty manual – getting information from the State Govt. (the clients) as the demand order for power. This also causes the problem at times of unknown high-power demand, which eventually stresses power plant equipments – leading to damage and decrease efficiency. Apart from the aforementioned, industries still don’t have batteries / power back-ups big enough to store power equivalent to multiple Mega-Watts.

OBJECTIVE

The aim of this project is to predict the values of power production for the initial 10 blocks (15 mins. Interval each) of the next day. First, prediction will be made on the values of last few blocks of the current day, giving out error rates. Later the models of LSTM, STM(RNN) and ARIMA will be put in a function to predict the electricity production values for the next day. Afterwards, comparison shall be made to conclude the best appropriate model for the issue, as well as considering the average prediction values for comparison.

SCOPE OF THE PROJECT

The commencement of the project shall be done by keeping in mind that the end result should be industrially useful. For the very same reason, I am using a dataset, which is acquired from a government regulated website, which holds information about energy production and consumption – on daily basis. The appropriate data shall be filtered out with necessary attributes. Data transformation and splitting can be carried out for creating training and testing data. Afterwards, model creation can take place and start the training procedure. As the prediction results have to be de-transformed, we should take care by keeping in mind the making of inverse transform function. Meanwhile, we can check for the error rate and positive or negative fitting of the predictions on the test data.

SCOPE OF THE PROJECT

Finally, the trained models will be exported and taken as an input parameter for a function which will provide the output as the final answer to our project. But the aforementioned was all about LSTM and STM. For ARIMA, the procedure deviates after data acquisition. We shall check for stationarity and seasonality and remove any trend that persists. Later, we make create persistence model, AR, MA and the ARIMA model. The model performance can be evaluated on the basis of error rates and fitting level. At last, comparison will be made to figure the best algorithm as well as how average prediction values will be useful. This study will enhance the efficiency of the power plants in terms of cost as well as power production - which will later lead to lowering the equipment failure frequency and demand for coal (as coal prices are growing exponentially - nowadays they are being imported !).

LITERATURE SURVEY

The authors in **[1]** proposed a hybrid long term forecasting method based on data mining technique and Time Series, in order to capture the nonlinear and complex pattern in yearly peak load and energy demand data. First, a support vector regression (SVR)-based forecasting algorithm is created. The Particle Swarm Optimization (PSO) methodology is used to optimise the SVR algorithm's parameters as well as the size of the input samples. A hybrid forecasting method is proposed for long-term annual electric peak load and total electric energy demand in order to reduce forecasting error. The Auto-Regressive Integrated Moving Average (ARIMA), Artificial Neural Network (ANN), and the proposed Support Vector Regression approach are the foundation of the proposed hybrid method. Based on the autocorrelation and partial autocorrelation of the original and differenced time series, the ARIMA method's parameters are established. According to the obtained inaccuracy over the existing data, the suggested hybrid forecasting technique ranks each forecasting method.

LITERATURE SURVEY

The authors of the research **[2]** looked into the advantages of combining data elements to create forecasts for short-term electricity consumption. The complex characteristics and clear seasonal trend that characterise the nature of electricity are typically present. To extract the fluctuation features, adaptive Fourier decomposition is initially applied to its benefit. The sub-series are then conducted to measure and remove the seasonal pattern after the criterion of the linear and stationary sequence is met. The average periodicity length is quantified throughout the seasonal adjustment procedure. Additionally, the sine cosine optimization approach is used to choose the penalty and kernel parameters of the support vector machine in order to achieve the generalisation performance on real electricity demand data. The results of the empirical study demonstrate that the proposed hybrid method's superior properties benefit from the impact of data pre-treatment, and they also demonstrate that this hybrid modelling approach can produce promising prediction results while maintaining a manageable level of computational complexity.

LITERATURE SURVEY

The Holt-Winters exponential smoothing, as discovered by the authors in **[3]**, enables precise periodic series forecasting with a small number of training samples. They provide a hybrid forecasting model to forecast electricity usage based on this technique. The optimal smoothing parameters for the Holt-Winters exponential smoothing are chosen using the fruit fly optimization process. They thoroughly tested the forecasting abilities of the suggested model to other approaches using electricity consumption data from a city in China. The findings show that the suggested model, even with a limited number of training samples, may significantly increase the prediction accuracy of monthly power use. The proposed model's computation time is also the quickest among the benchmark hybrid algorithms that were tested.

LITERATURE SURVEY

Multiple Temporal Aggregation (MTA), according to the authors in **[4]**, has been demonstrated to reduce the model selection issue for low-frequency time series. To better manage the unfavourable effect of seasonality shrinkage that MTA implies and combine it with traditional cross-sectional hierarchical forecasting, they suggest modifying the Multiple Aggregation Prediction Algorithm, a special version of MTA, for high-frequency time series. An empirical evaluation of the effects of temporal aggregate inclusion in hierarchical forecasting is performed using real data from five bank branches. We demonstrate that the suggested MTA methodology exhibits superior accuracy, aggregation consistency, and trustworthy automatic forecasting when used in conjunction with the best reconciliation method.

LITERATURE SURVEY

To forecast the electricity demand at the individual building level and the aggregate level in hourly intervals, the authors of research **[5]** used data from 47 commercial buildings and assessed a number of machine learning techniques. Understanding short-term dynamics necessitates forecasting hourly granularity, while the majority of neighbourhood scale studies are restricted to yearly, monthly, weekly, or daily data resolutions. Two years of data were utilised to train the model, and a third year of untrained data was used to make the forecast. For estimating the power demand of specific buildings and groups of buildings, boosted-tree, random forest, and SVM-linear, quadratic, and cubic were all examined and put to the test. The findings demonstrated that when criteria like computational time and error accuracy are evaluated, boosted-tree, random forest, and ANN delivered the best results for prediction at hourly granularity.

LITERATURE SURVEY

The authors in **[6]** eliminate the redundant influencing elements of building energy use and identify the key factors. They do this by using the rough set theory. These crucial elements were then fed into a deep neural network, which has a "deep" architecture and potent feature extraction skills. The deep neural network's output is the amount of energy used by buildings. For preliminary set reduction, this study gathered data from 100 civic public buildings. Next, for nearly a year, it collected data from a university laboratory building in Dalian to train and test deep neural networks. Both the short- and long-term forecasts for building energy use were tested. Comparisons were made between the predictions made by the deep neural network and those made by the back propagation neural network, the Elman neural network, and the fuzzy neural network. The outcomes demonstrate that the deep neural network and integrated rough set provided the most accurate results. The approach put forward in this study could offer a workable and precise answer for predicting building energy use.

LITERATURE SURVEY

In the study **[7]**, the authors create a machine learning-based cloud platform called Microsoft Azure and a prediction model for energy use. For the algorithm of the prediction model, three methodologies—Support Vector Machine, Artificial Neural Network, and k-Nearest Neighbour—are suggested. The case study of two tenants from a business building in Malaysia focuses on real-world applicability. Before being utilised for model training and testing, the obtained data is analysed and pre-processed. RMSE, NRMSE, and MAPE measures are used to compare the effectiveness of each approach. The test demonstrates that the distribution of energy usage varies depending on the renter.

LITERATURE SURVEY

The authors of the research **[8]** claim that more sophisticated hybrid forecasting methodologies, which use statistical methods and machine learning techniques, are necessary for the analysis of uncertainty in power demand. The research suggests that temperature and day type are the most important characteristics in estimating power consumption. This research is unique in that it uses data on Canadian power consumption to investigate how temperature and day type affect electricity demand using hybrid models. The point forecasts/predictions for the Ontario energy demand data are obtained using the hybrid neural network dynamic regression (NNDR) model. The experimental findings demonstrate the superiority of the proposed NNDR model over the Prophet and Seasonal Autoregressive Integrated Moving Average (DRSARIMA) errors models, two widely used dynamic regression models. Additionally, for the NNDR, DRSARIMA, and Prophet models, long-term point predictions and innovations are utilised to derive two classes of prediction intervals (PIs) utilising data-driven probabilistic innovation distribution and bootstrapping.

LITERATURE SURVEY

The authors of the paper **[9]** make use of ensemble-based machine learning methods. In order to improve data quality for the forecasting job and enable the model to adjust itself in pandemic scenarios, lockdown temporal rules are added to the feature set. The short-term country-level demand forecast model is explored using a number of ensemble-based machine learning methods. Additionally, the quantile random forest regression is carried out in a probabilistic manner. The algorithms are taught to forecast Germany's national demand for case studies. The findings suggest that ensemble models, particularly boosting and bagging-boosting models, are capable of producing precise demand estimates at the national level. Additionally, most of these models are resistant to missing pandemic policy data. The forecasting accuracy during a pandemic crisis is greatly increased by using the pandemic policy data as characteristics, though. For the aforementioned case study, the probabilistic quantile regression also showed great accuracy.

LITERATURE SURVEY

The authors of the study **[10]** assert that artificial neural networks are used to anticipate power consumption using ANN. With historical energy consumption statistics, educational data sets that account for social, economic, technological, and demographic characteristics were built in order to improve accuracy. The developing provinces of Düzce, Turkey, were subjected to the established forecasting methodology. The ANN-based technique was used to predict the region's 15-year electrical energy consumption. The outcomes have undergone a thorough analysis.

PROPOSED METHODOLOGY

- Dataset acquired from the websites : Northern Regional Power Committee (NRPC) and Northern Regional Load Dispatch Centre (NRLDC).
- Platform Chosen : Google Colab
- Algorithms Chosen to be used for analysis and forecasting : LSTM, RNN & ARIMA
- Procedure for LSTM & RNN Model training, testing & forecasting shall be completely different from that of ARIMA (thanks to non-stationarity of dataset).

LSTM & RNN Implementation –

❖ Data transformation by scaling values using MinMaxScaler (inverse transformation function can be used later to acquire original scale values).
❖ Data split → Train : Test = 75% : 25%
❖ Timesteps taken : 10 (10 rows at a time for input, 11th row as output - for learning / training)

PROPOSED METHODOLOGY

- ❖ Extended dataset (by timesteps) : divided into components - x_train, y_train, x_test, y_test
- ❖ Model implementation by importing KERAS, a python library consisting numerous ML model creation tools
- ❖ For RNN, layers have to be set, which includes memory & processing layers, and a hidden layer (to remember previous inputs - initially ZERO), and an output layer
- ❖ LSTM & RNN ingests 3-Dimensional data, hence the existing data has to be converted in the aforementioned structure - consisting batch size, timsteps & input shape
- ❖ Training will be carried out on x_train & y_train components
- ❖ Testing & Prediction of test data will be done on x_test component

💥 Model Evaluation will be based on Error rates, rather than Accuracy & Precision - as Error rates seems efficacious in increasing the efficiency of the power sector industries

PROPOSED METHODOLOGY

- ❖ Evaluation metrics used : RMSE (Root Mean Squared Error) and R2 (R-square) score - for error spreading and model fitting
- ❖ Lastly (for LSTM & RNN), a function will be created to forecast the amount of power to be generated for initial 10 blocks of next day (NOTE : 1 block is equivalent to 15 minutes)

ARIMA Implementation –

- ➢ Implementation of Rolling Statistics, which will examine the data-points for any anomalies , w.r.t. stationarity via moving average.
- ➢ Applying **Augmented Dickey-Fuller (ADF) Test** - a hypothesis based implementation to test stationarity.

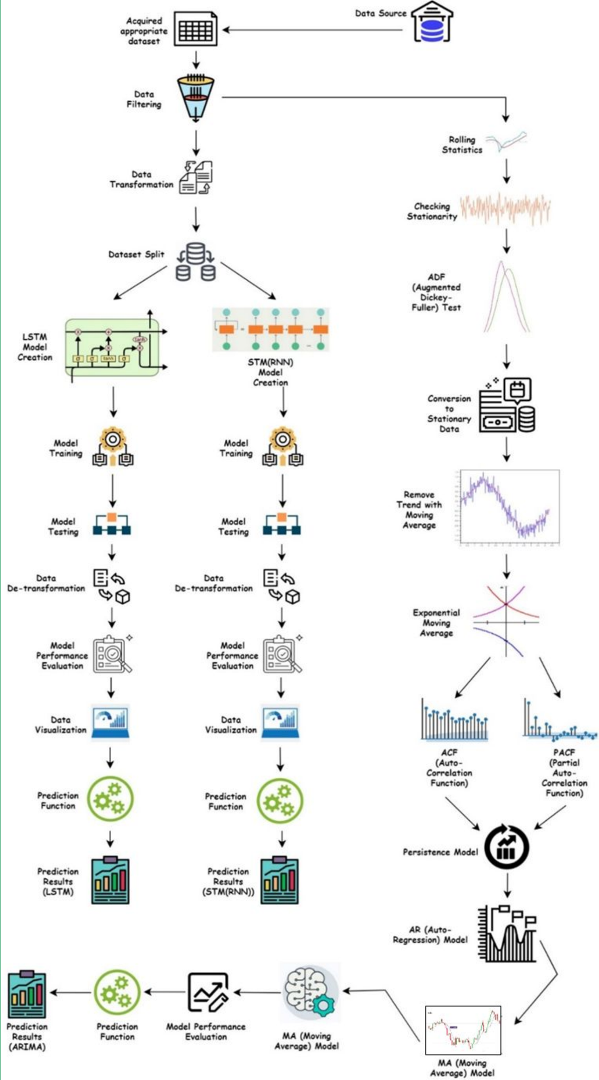
PROPOSED METHODOLOGY

- De-Trending (OR removing stationarity) the dataset : using BoxCox (Logarithmic) transformation, Removing trend by Moving Average & Exponential Decay Transformation
- Applying **ADF** Test to confirm non-stationarity of dataset
- Applying ACF (Autocorrelation Function) & PACF (Partial ACF) to test whether correlation exists among current data-points and a lagged version of the same(i.e., to the previous data-point)
- Before creating **ARIMA** Model, will be creating **Persistence** model - making the assumption that nothing will change between the present and the projected period when calculating the future value of a time series.
- Creating AR (Auto-Regressive) model - a time series model that predicts the value at the following time step using observations from past time steps as the input to a regression equation.

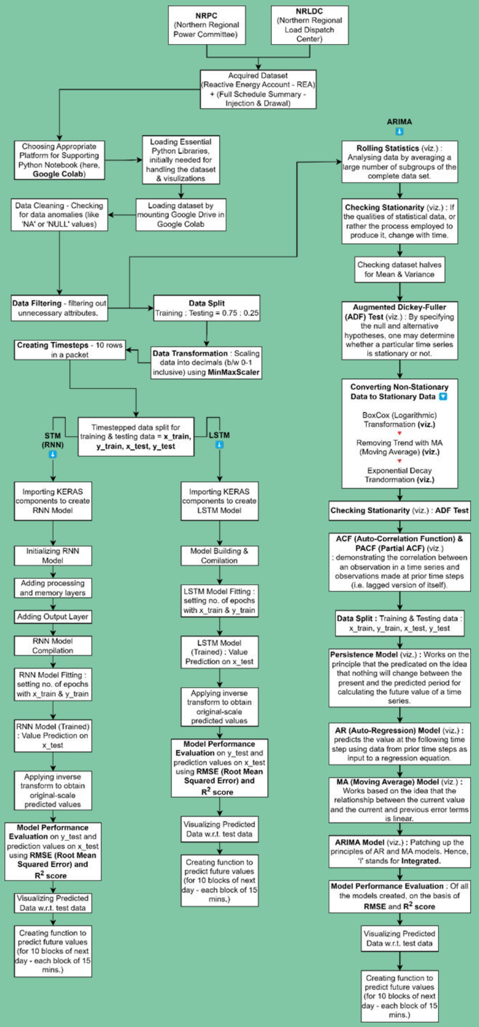
PROPOSED METHODOLOGY

- ➢ Creating MA (Moving Average) Model - a time series average that moves through the series by removing the top items from the most recent averaged group and include the next in each succeeding average.
- ➢ AR and MA Models create the ARIMA Model - hence, ‘I’ stands for integrated - which means that the data is stationary.
- ➢ Model Evaluation will be based on RMSE and R2 score (R-square)
- ➢ Prediction values of the models will be compared to that of original data as well as the average of the predicted values will be compared to check whether the error rate is lowered or not.

SYSTEM ARCHITECTURE


Data Source
Acquired appropriate dataset
Data Filtering
Data Transformation
Dataset Split
LSTM Model Creation
STM(RNN) Model Creation
Model Training
Model Training
Model Testing
Model Testing
Data De-transformation
Data De-transformation
Model Performance Evaluation
Model Performance Evaluation
Data Visualization
Data Visualization
Prediction Function
Prediction Function
Prediction Results (LSTM)
Prediction Results (STM(RNN))
Rolling Statistics
Checking Stationarity
ADF (Augmented Dickey-Fuller) Test
Conversion to Stationary Data
Remove Trend with Moving Average
Exponential Moving Average
ACF (Auto-Correlation Function)
PACF (Partial Auto-Correlation Function)
Persistence Model
AR (Auto-Regression) Model
MA (Moving Average) Model
MA (Moving Average) Model
Model Performance Evaluation
Prediction Function
Prediction Results (ARIMA)

COMPLETE DESIGN


NRPC (Northern Regional Power Committee)
NRLDC (Northern Regional Load Dispatch Center)
Acquired Dataset (Reactive Energy Account - REA) + (Full Schedule Summary - Injection & Drawal)
Choosing Appropriate Platform for Supporting Python Notebook (here, Google Colab)
Loading Essential Python Libraries, initially needed for handling the dataset & visulizations
Loading dataset by mounting Google Drive in Google Colab
Data Cleaning - Checking for data anomalies (like 'NA' or 'NULL' values)
Data Filtering - filtering out unnecessary attributes.
Data Split Training : Testing = 0.75 : 0.25
Data Transformation : Scaling data into decimals (b/w 0-1 inclusive) using MinMaxScaler
Creating Timesteps - 10 rows in a packet
Timestepped data split for training & testing data = x_train, y_train, x_test, y_test
STM (RNN)
LSTM
Importing KERAS components to create RNN Model
Initializing RNN Model
Adding processing and memory layers
Adding Output Layer
RNN Model Compilation
RNN Model Fitting : setting no. of epochs with x_train & y_train
RNN Model (Trained) : Value Prediction on x_test
Applying inverse transform to obtain original-scale predicted values
Model Performance Evaluation on y_test and prediction values on x_test using RMSE (Root Mean Squared Error) and R2 score
Visualizing Predicted Data w.r.t. test data
Creating function to predict future values (for 10 blocks of next day - each block of 15 mins.)
Importing KERAS components to create LSTM Model
Model Building & Comilation
LSTM Model Fitting : setting no. of epochs with x_train & y_train
LSTM Model (Trained) : Value Prediction on x_test
Applying inverse transform to obtain original-scale predicted values
Model Performance Evaluation on y_test and prediction values on x_test using RMSE (Root Mean Squared Error) and R2 score
Visualizing Predicted Data w.r.t. test data
Creating function to predict future values (for 10 blocks of next day - each block of 15 mins.)
ARIMA
Rolling Statistics (viz.) : Analysing data by averaging a large number of subgroups of the complete data set.
Checking Stationarity (viz.) : If the qualities of statistical data, or rather the process employed to produce it, change with time.
Checking dataset halves for Mean & Variance
Augmented Dickey-Fuller (ADF) Test (viz.) : By specifying the null and alternative hypotheses, one may determine whether a particular time series is stationary or not.
Converting Non-Stationary Data to Stationary Data
BoxCox (Logarithmic) Transformation (viz.)
Removing Trend with MA (Moving Average) (viz.)
Exponential Decay Trandormation (viz.)
Checking Stationarity (viz.) : ADF Test
ACF (Auto-Correlation Function) & PACF (Partial ACF) (viz.) : demonstrating the correlation between an observation in a time series and observations made at prior time steps (i.e. lagged version of itself).
Data Split : Training & Testing data : x_train, y_train, x_test, y_test
Persistence Model (viz.) : Works on the principle that the predicated on the idea that nothing will change between the present and the predicted period for calculating the future value of a time series.
AR (Auto-Regression) Model (viz.) : predicts the value at the following time step using data from prior time steps as input to a regression equation.
MA (Moving Average) Model (viz.) : Works based on the idea that the relationship between the current value and the current and previous error terms is linear.
ARIMA Model (viz.) : Patching up the principles of AR and MA models. Hence, 'I' stands for Integrated.
Model Performance Evaluation : Of all the models created, on the basis of RMSE and R2 score
Visualizing Predicted Data w.r.t. test data
Creating function to predict future values (for 10 blocks of next day - each block of 15 mins.)

MODULE DESCRIPTION

TIME-SERIES

- During analysis, when we conclude that the data acquired changes/repeats after certain time-intervals OR the data has a considerable amount of dependence on time – then the data is called – Time Series Data.
- Time-Series Data is based on 5 factors :
    - Level (whether we can find average of data points)
    - Trend
    - Seasonality
    - Cyclic Patterns
    - Noise

MODULE DESCRIPTION

TIME-SERIES ANALYSIS (TSA)

TSA is the foundation for forecasting analysis and prediction, specifically for time-based issue statements like :

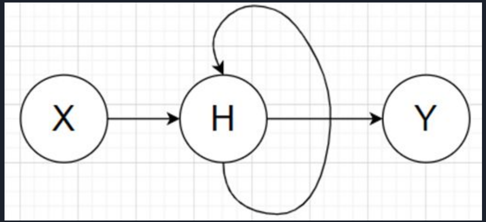
- Examining the trends in the historical dataset
- Recognizing and comparing patterns derived from the previous stage with the current circumstance.
- Recognising the factor(s) impacting a certain variable(s) at various times.

MODULE DESCRIPTION

TIME-SERIES ALGORITHMS

1 SHORT TERM MEMORY (STM) NEURAL NETWORK - RNN

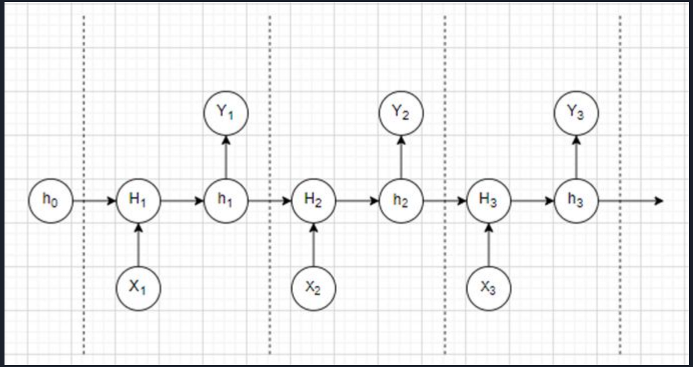
- An artificial neural network that employs sequential data or time series data is known as a recurrent neural network (RNN).



- So, the output that came out from the current step will act as the input for the next step – and it will keep on repeating , depending on the no. of layers included , i.e. , density of the model created.

MODULE DESCRIPTION

- RNN Actual Model –



tanh(x) function formula -

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$h_t = f(h_{t-1}, X_t)$$

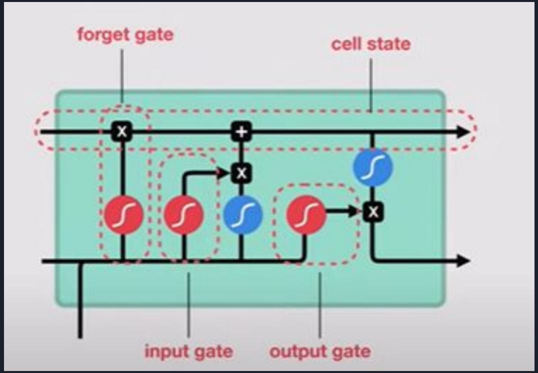
$$h_t = \tanh(W_{hh} \cdot h_{t-1} + W_{Xh} \cdot X_t)$$

$$Y_t = W_{hy} \cdot h_t$$

MODULE DESCRIPTION

2️⃣LONG TERM MEMORY (LSTM) NEURAL NETWORK

- Long short-term memory networks, or LSTMs, are employed in deep learning. Many recurrent neural networks (RNNs) are able to learn long-term dependencies, particularly in tasks involving sequence prediction.
- Below is a pictorial representation of one of the stages in LSTM -



MODULE DESCRIPTION

- LSTM has 3 main gates -
  - FORGET GATE
  - INPUT GATE
  - OUTPUT GATE

MODULE DESCRIPTION

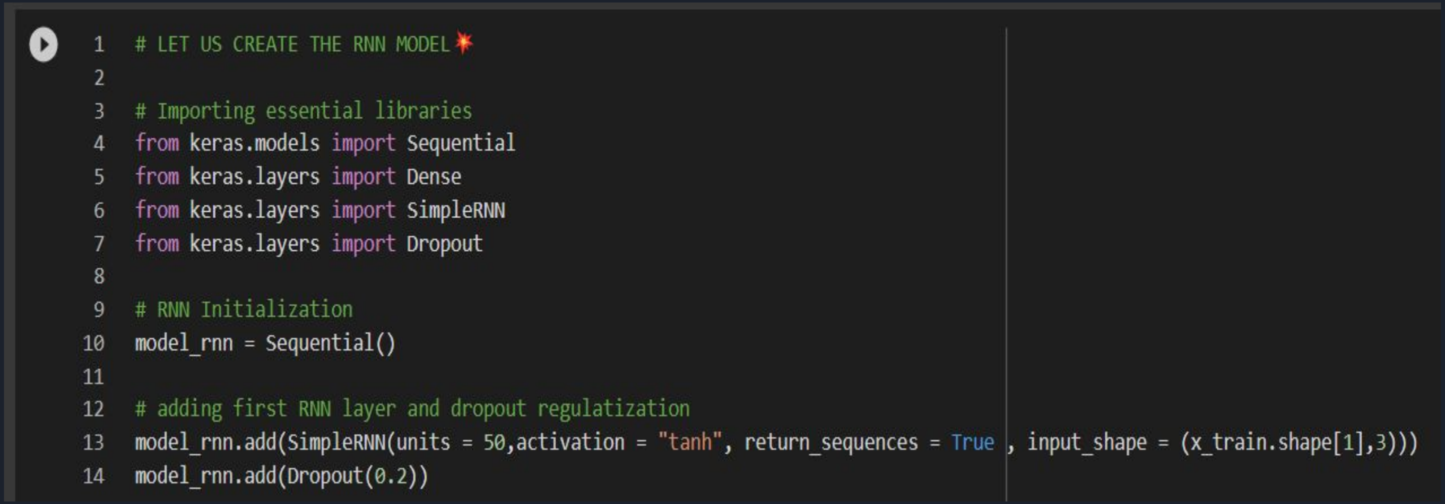
3️⃣AUTO-REGRESSIVE INTEGRATED MOVING AVERAGE (ARIMA)

- A statistical analysis model called ARIMA uses time series data to either forecast future trends or provide a better understanding of the current data set.
- The components of **ARIMA** can be explained as follows :
  - A model known as **autoregression (AR)** depicts a variable that is evolving and regresses on its own lagged, or previous, values.
  - To enable the time series to become stable, **integrated (I)** indicates the differencing of the raw observations (i.e., data values are replaced by the difference between the data values and the previous values).
  - When a moving average model is applied to lagged observations, the **moving average (MA)** incorporates the relationship between an observation and a residual error.

IMPLEMENTATION

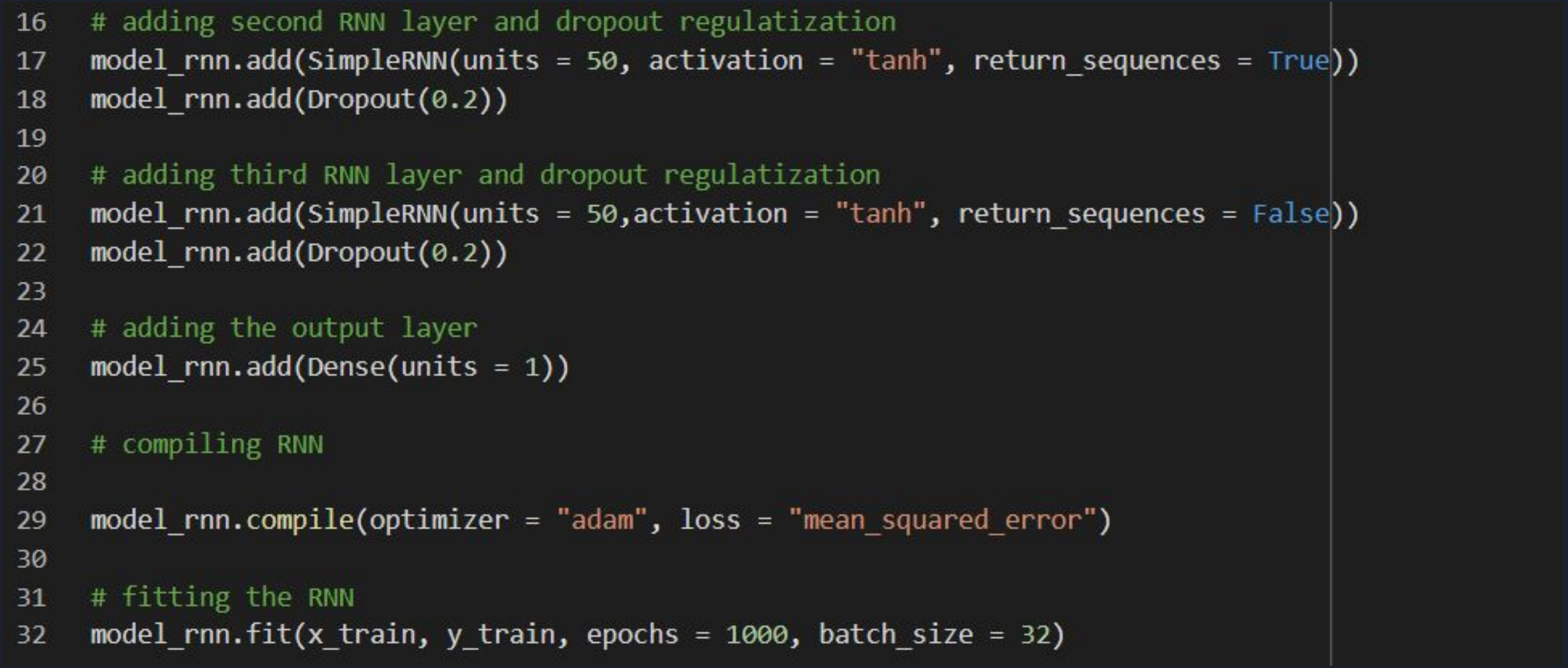
Major implementation steps will be shown on Google Colab. Here, I am including the snapshots of the core steps taken to create and train the models -

RNN



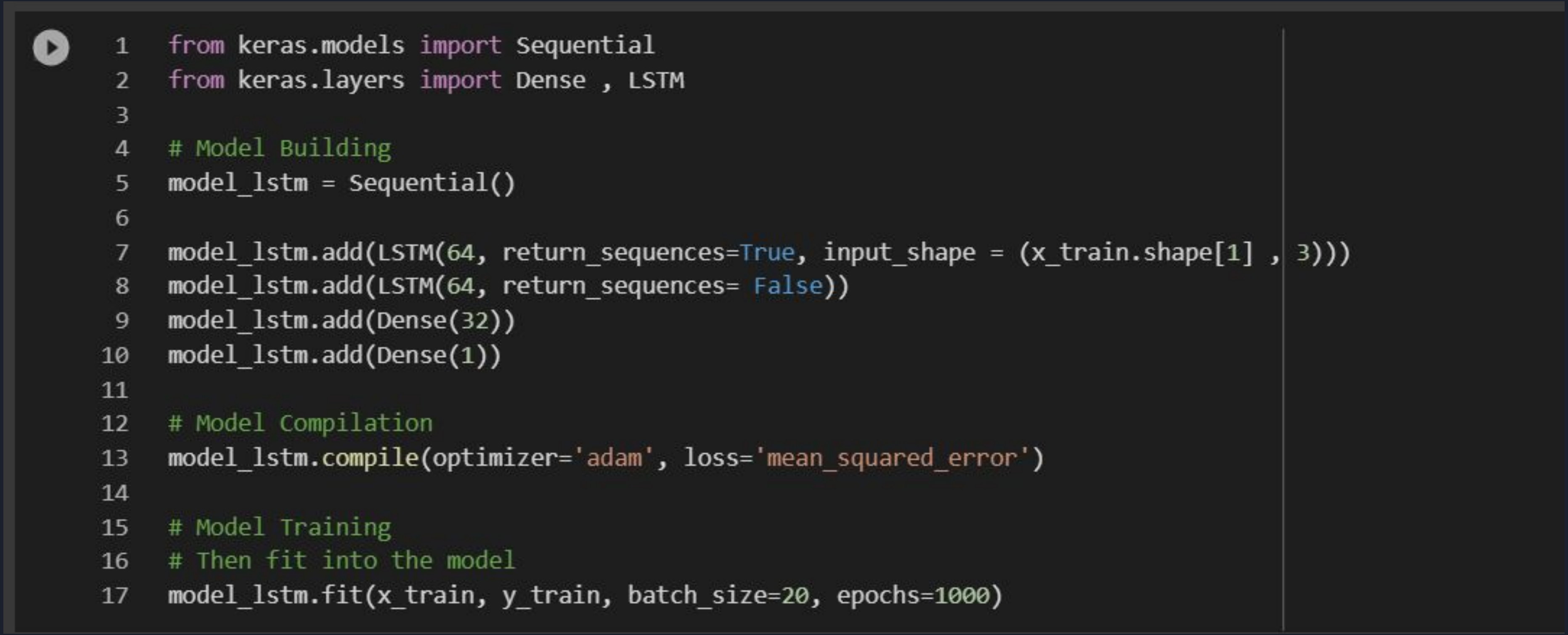
```
# LET US CREATE THE RNN MODEL💥

# Importing essential libraries
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import SimpleRNN
from keras.layers import Dropout

# RNN Initialization
model_rnn = Sequential()

# adding first RNN layer and dropout regulization
model_rnn.add(SimpleRNN(units = 50,activation = "tanh", return_sequences = True , input_shape = (x_train.shape[1],3)))
model_rnn.add(Dropout(0.2))
```

RNN (Contd.)



```python
# adding second RNN layer and dropout regulatization
model_rnn.add(SimpleRNN(units = 50, activation = "tanh", return_sequences = True))
model_rnn.add(Dropout(0.2))

# adding third RNN layer and dropout regulatization
model_rnn.add(SimpleRNN(units = 50,activation = "tanh", return_sequences = False))
model_rnn.add(Dropout(0.2))

# adding the output layer
model_rnn.add(Dense(units = 1))

# compiling RNN

model_rnn.compile(optimizer = "adam", loss = "mean_squared_error")

# fitting the RNN
model_rnn.fit(x_train, y_train, epochs = 1000, batch_size = 32)
```

LSTM



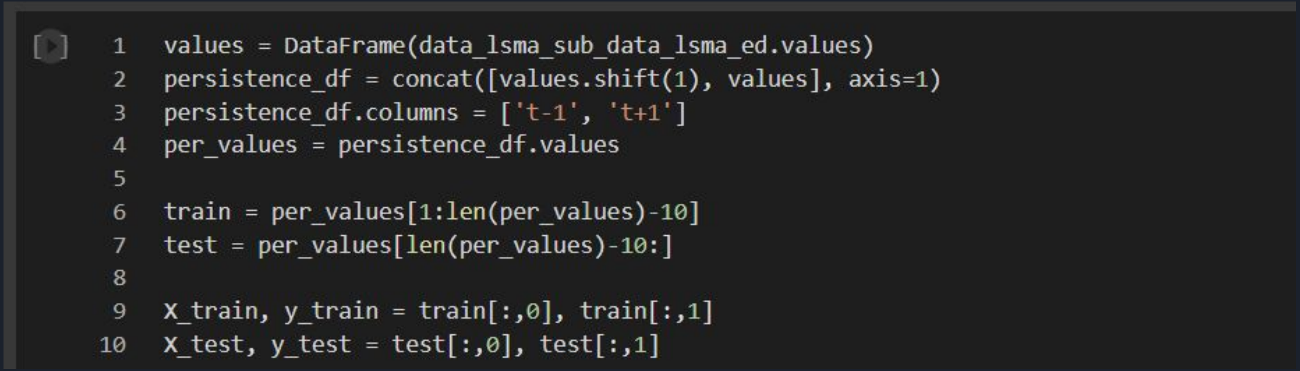
```
from keras.models import Sequential
from keras.layers import Dense , LSTM

# Model Building
model_lstm = Sequential()

model_lstm.add(LSTM(64, return_sequences=True, input_shape = (x_train.shape[1] , 3)))
model_lstm.add(LSTM(64, return_sequences= False))
model_lstm.add(Dense(32))
model_lstm.add(Dense(1))

# Model Compilation
model_lstm.compile(optimizer='adam', loss='mean_squared_error')

# Model Training
# Then fit into the model
model_lstm.fit(x_train, y_train, batch_size=20, epochs=1000)
```

ARIMA

NOTE : Before the implementation of ARIMA, 3 models were implemented, namely :

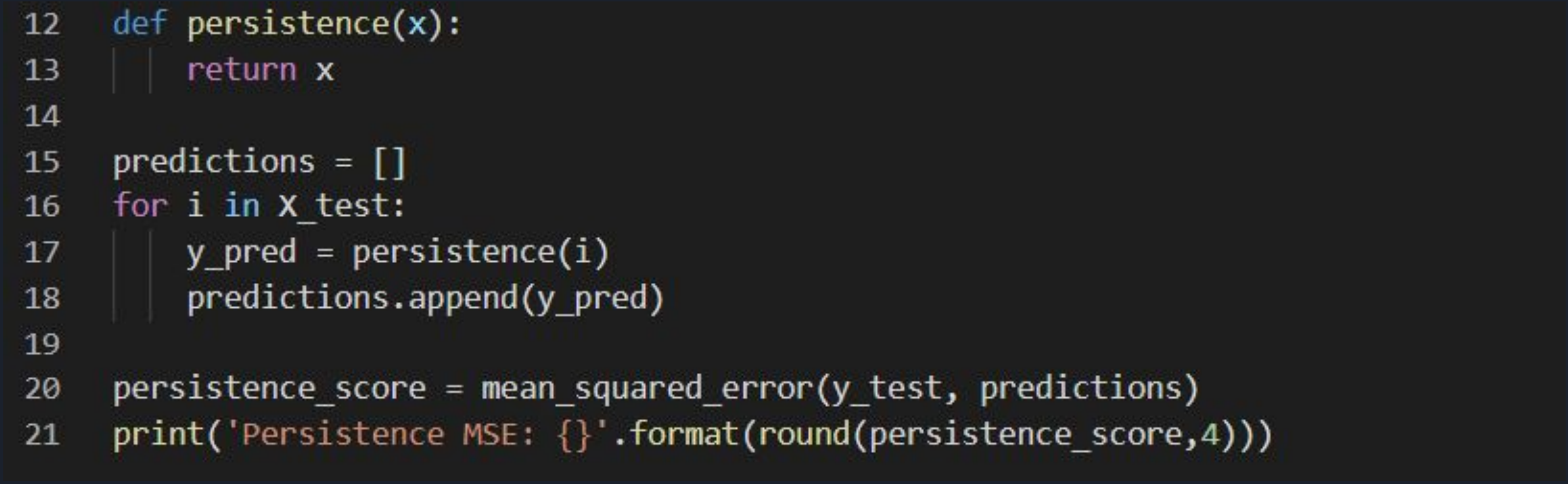
- Persistence Model
- AR (Auto-Regressive) Model
- MA (Moving Average) Model

Persistence Model

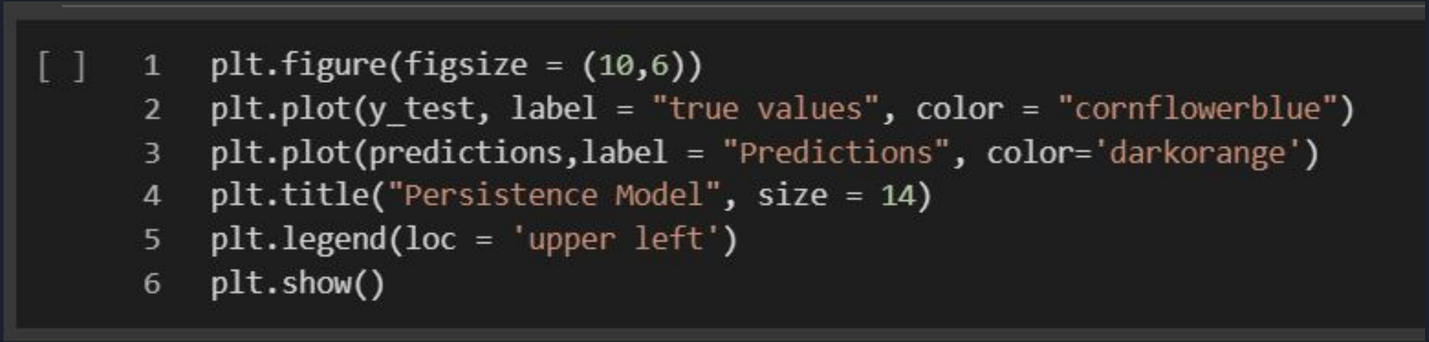


```
values = DataFrame(data_lsma_sub_data_lsma_ed.values)
persistence_df = concat([values.shift(1), values], axis=1)
persistence_df.columns = ['t-1', 't+1']
per_values = persistence_df.values

train = per_values[1:len(per_values)-10]
test = per_values[len(per_values)-10:]

X_train, y_train = train[:,0], train[:,1]
X_test, y_test = test[:,0], test[:,1]
```

Persistence Model (Contd.)

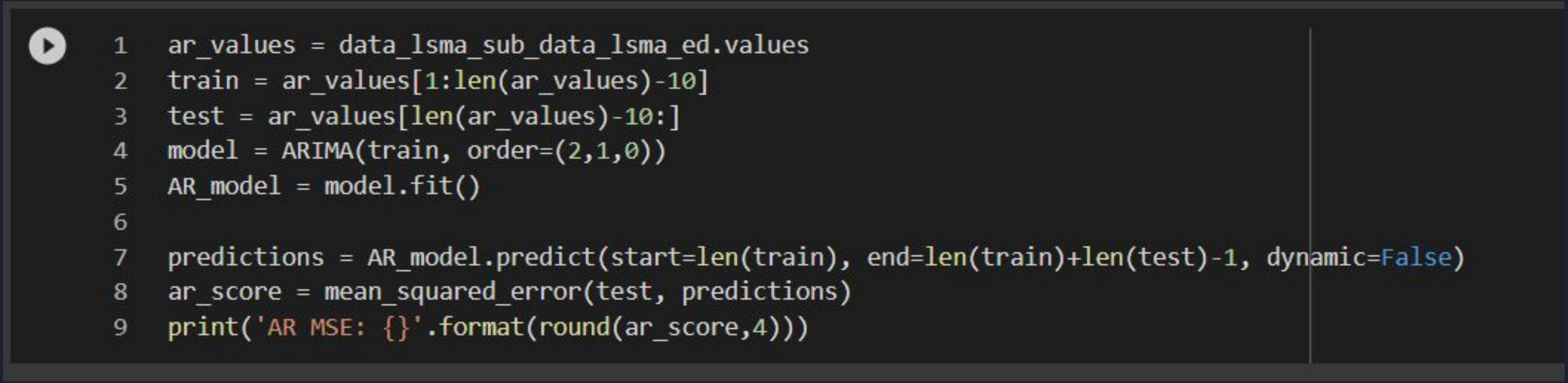


```python
def persistence(x):
    return x

predictions = []
for i in x_test:
    y_pred = persistence(i)
    predictions.append(y_pred)

persistence_score = mean_squared_error(y_test, predictions)
print('Persistence MSE: {}'.format(round(persistence_score,4)))
```

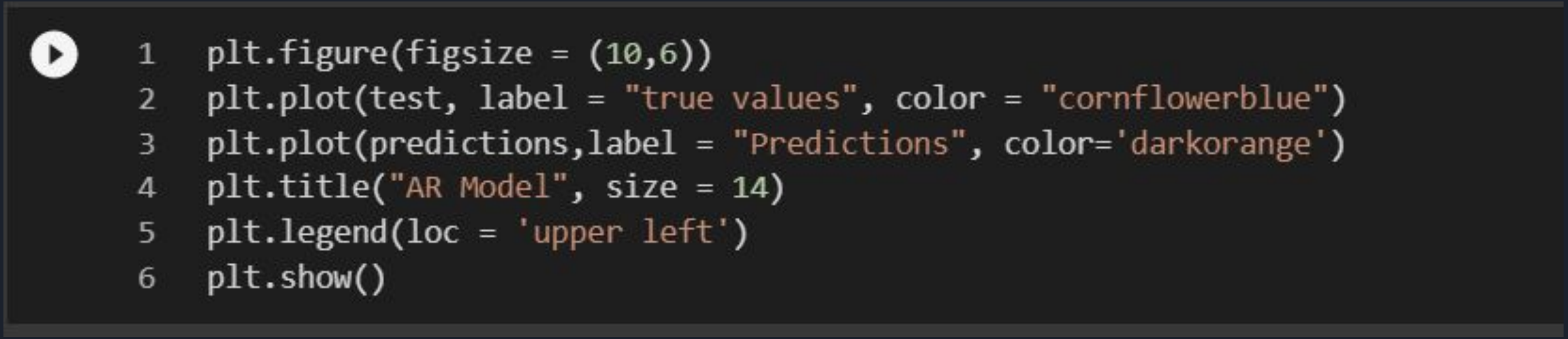


```python
plt.figure(figsize = (10,6))
plt.plot(y_test, label = "true values", color = "cornflowerblue")
plt.plot(predictions,label = "Predictions", color='darkorange')
plt.title("Persistence Model", size = 14)
plt.legend(loc = 'upper left')
plt.show()
```

AR (Auto-Regressive) Model

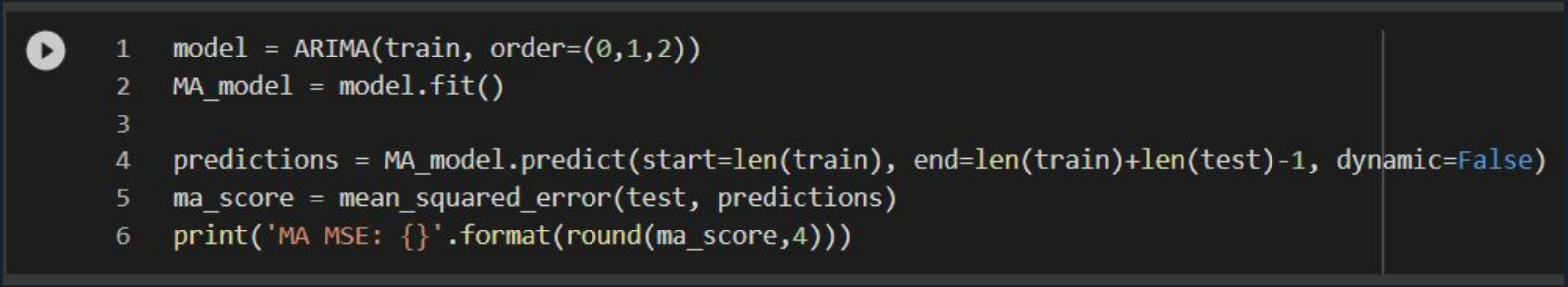


```python
ar_values = data_lsma_sub_data_lsma_ed.values
train = ar_values[1:len(ar_values)-10]
test = ar_values[len(ar_values)-10:]
model = ARIMA(train, order=(2,1,0))
AR_model = model.fit()

predictions = AR_model.predict(start=len(train), end=len(train)+len(test)-1, dynamic=False)
ar_score = mean_squared_error(test, predictions)
print('AR MSE: {}'.format(round(ar_score,4)))
```

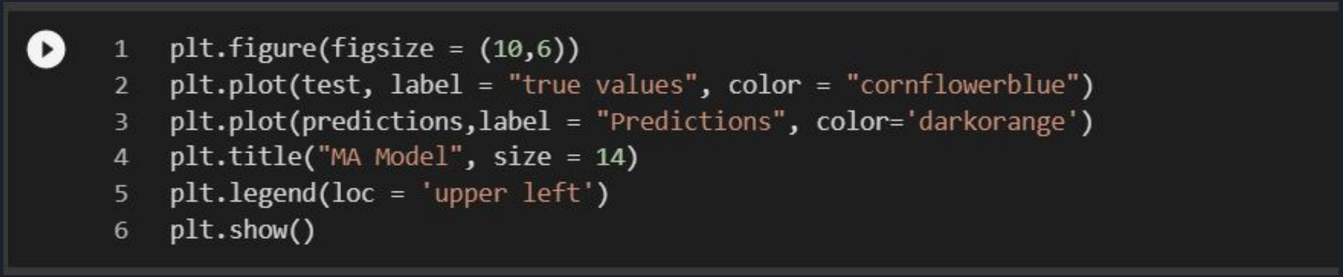


```python
plt.figure(figsize = (10,6))
plt.plot(test, label = "true values", color = "cornflowerblue")
plt.plot(predictions,label = "Predictions", color='darkorange')
plt.title("AR Model", size = 14)
plt.legend(loc = 'upper left')
plt.show()
```

MA (Moving Average) Model

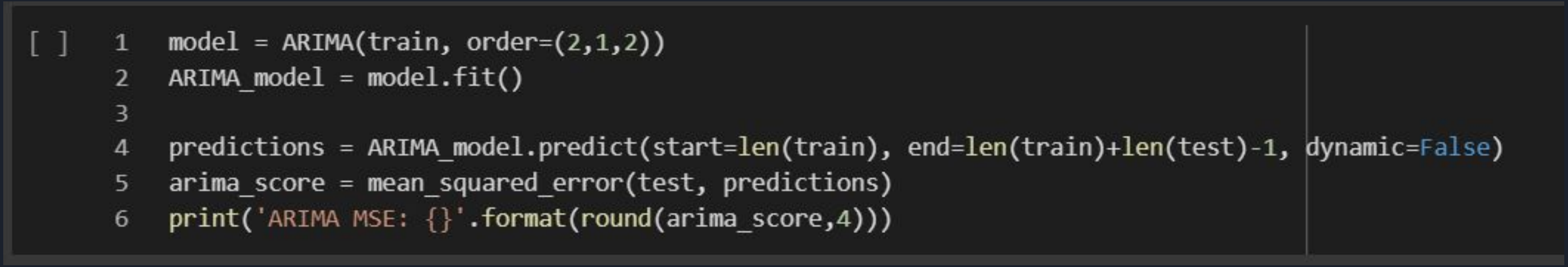


```
model = ARIMA(train, order=(0,1,2))
MA_model = model.fit()

predictions = MA_model.predict(start=len(train), end=len(train)+len(test)-1, dynamic=False)
ma_score = mean_squared_error(test, predictions)
print('MA MSE: {}'.format(round(ma_score,4)))
```

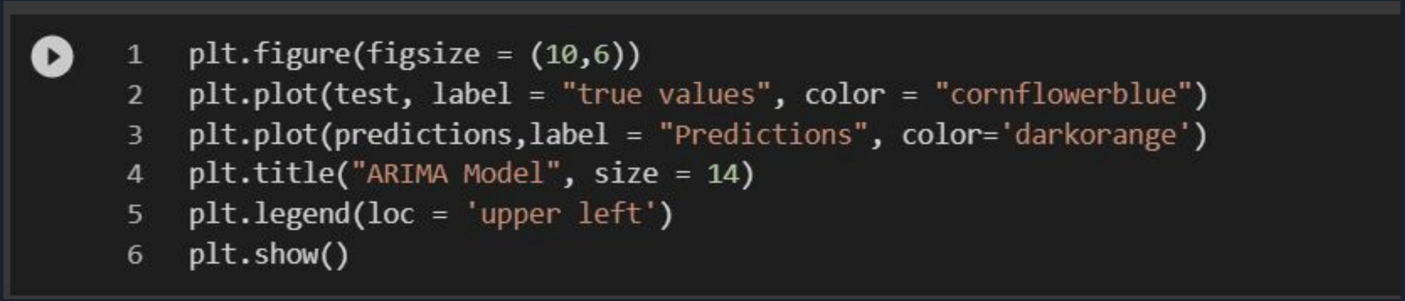


```
plt.figure(figsize = (10,6))
plt.plot(test, label = "true values", color = "cornflowerblue")
plt.plot(predictions,label = "Predictions", color='darkorange')
plt.title("MA Model", size = 14)
plt.legend(loc = 'upper left')
plt.show()
```

ARIMA Model



```
[ ]
model = ARIMA(train, order=(2,1,2))
ARIMA_model = model.fit()

predictions = ARIMA_model.predict(start=len(train), end=len(train)+len(test)-1, dynamic=False)
arima_score = mean_squared_error(test, predictions)
print('ARIMA MSE: {}'.format(round(arima_score,4)))
```



```
plt.figure(figsize = (10,6))
plt.plot(test, label = "true values", color = "cornflowerblue")
plt.plot(predictions,label = "Predictions", color='darkorange')
plt.title("ARIMA Model", size = 14)
plt.legend(loc = 'upper left')
plt.show()
```

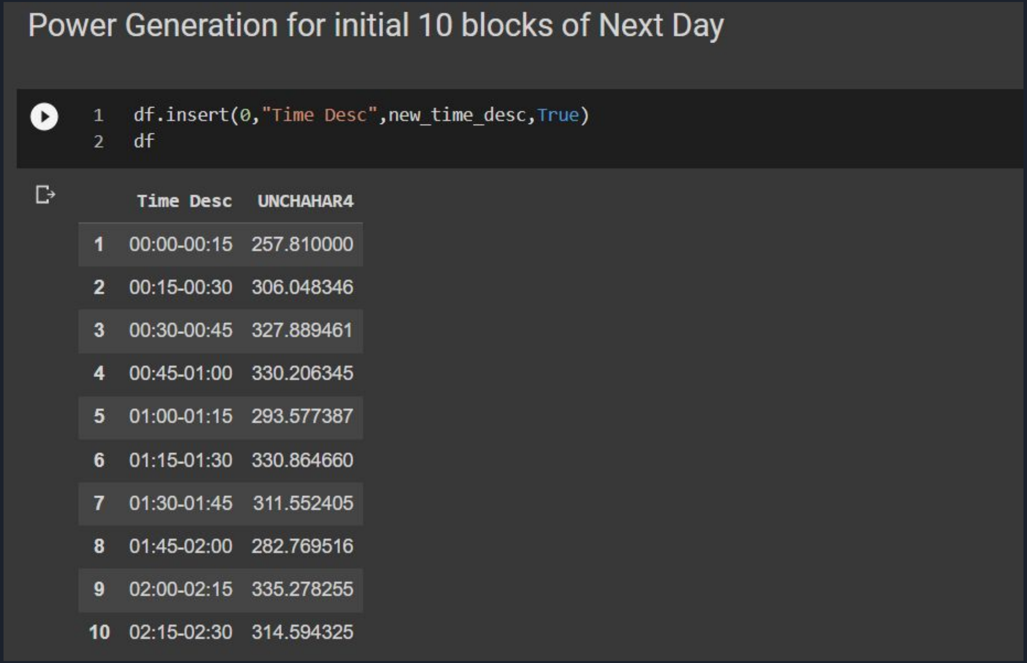
TESTING & PERFORMANCE METRICS

Testing will be based on x_test component of data. For evaluation metrics, I will be using RMSE (Root Mean Square Error) and R2 (R-square) score.

Algo. ➡️ Metrics ⬇️	RNN	LSTM	ARIMA Model
RMSE	`7791.47`	`7754.76`	`0.33`
R2 (R-square) score	`-30.10`	`-29.81`	`-11.50`

The end result for each algorithms' implementation is to forecast the power production for initial 10 blocks of next day (1 block = 15 minutes)

For RNN –

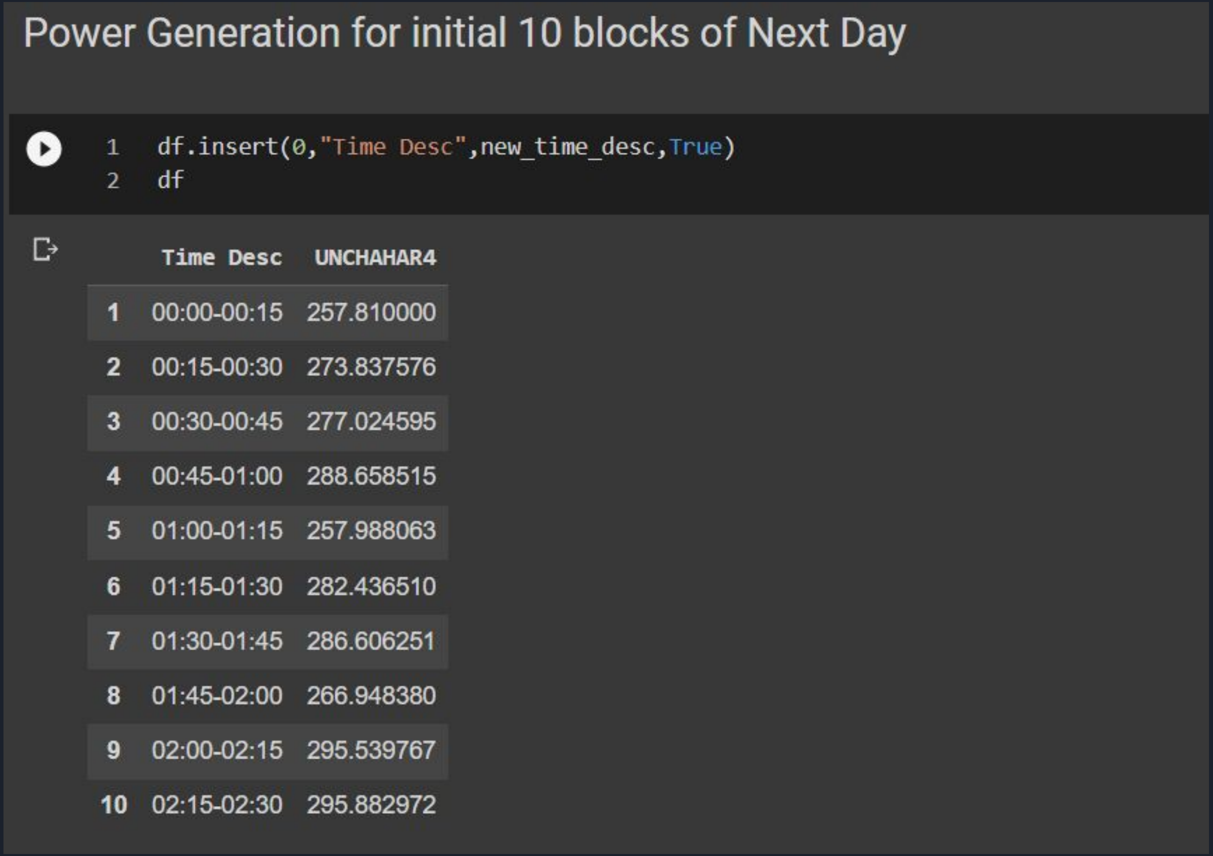


Power Generation for initial 10 blocks of Next Day

```
df.insert(0,"Time Desc",new_time_desc,True)
df
```

	Time Desc	UNCHAHAR4
1	00:00-00:15	257.810000
2	00:15-00:30	306.048346
3	00:30-00:45	327.889461
4	00:45-01:00	330.206345
5	01:00-01:15	293.577387
6	01:15-01:30	330.864660
7	01:30-01:45	311.552405
8	01:45-02:00	282.769516
9	02:00-02:15	335.278255
10	02:15-02:30	314.594325

For LSTM –

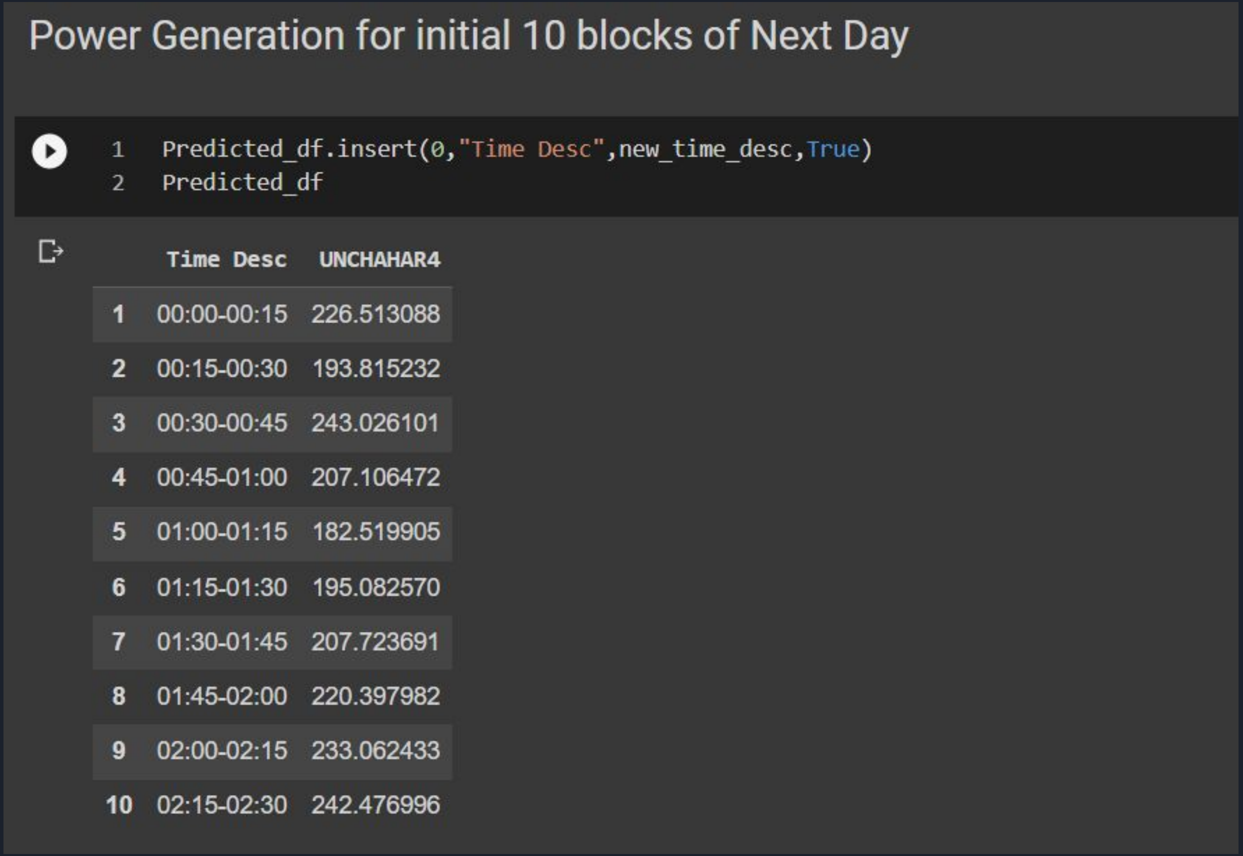


Power Generation for initial 10 blocks of Next Day

```
df.insert(0,"Time Desc",new_time_desc,True)
df
```

	Time Desc	UNCHAHAR4
1	00:00-00:15	257.810000
2	00:15-00:30	273.837576
3	00:30-00:45	277.024595
4	00:45-01:00	288.658515
5	01:00-01:15	257.988063
6	01:15-01:30	282.436510
7	01:30-01:45	286.606251
8	01:45-02:00	266.948380
9	02:00-02:15	295.539767
10	02:15-02:30	295.882972

For ARIMA –



Power Generation for initial 10 blocks of Next Day

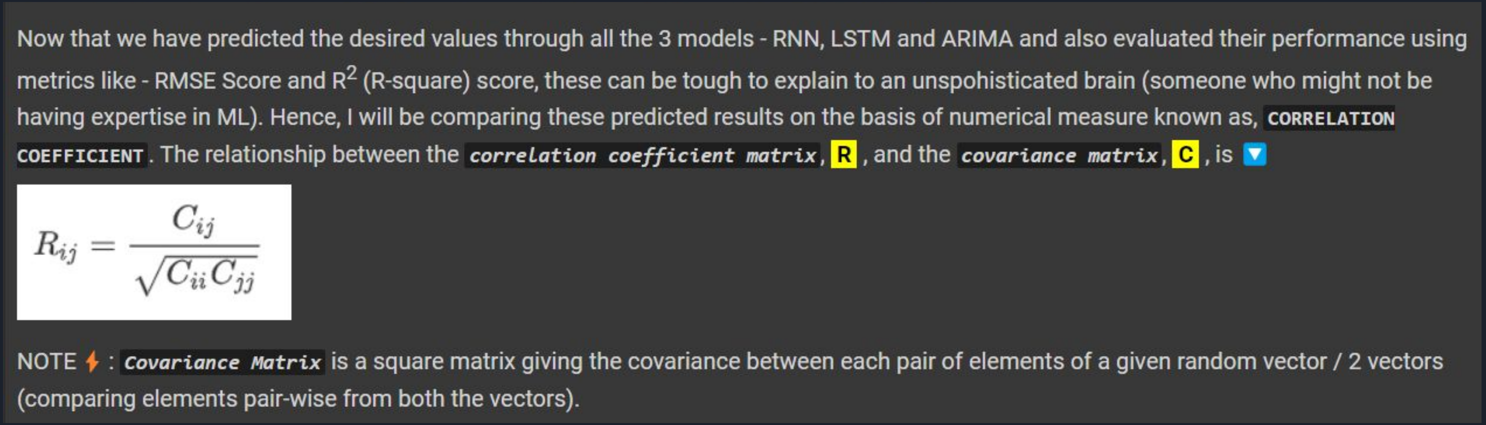
```
Predicted_df.insert(0,"Time Desc",new_time_desc,True)
Predicted_df
```

	Time Desc	UNCHAHAR4
1	00:00-00:15	226.513088
2	00:15-00:30	193.815232
3	00:30-00:45	243.026101
4	00:45-01:00	207.106472
5	01:00-01:15	182.519905
6	01:15-01:30	195.082570
7	01:30-01:45	207.723691
8	01:45-02:00	220.397982
9	02:00-02:15	233.062433
10	02:15-02:30	242.476996

RESULTS

- Results obtained on the basis of evaluation metrics cannot cater the needs of unsophisticated brains (say, a business analyst who can be a neophyte towards the concepts of Machine Learning).
- Best possible way : provide a pictorial representation of your analysis.
- The prediction values of RNN, LSTM and ARIMA will be compared with that of original power production data of the next day (from the data source).
- Relation among the data will be acquired on the basis of **RMSE, R-Square and Correlation Coefficient** (This is an inbuilt function in the Python's Numpy Library)

RESULTS

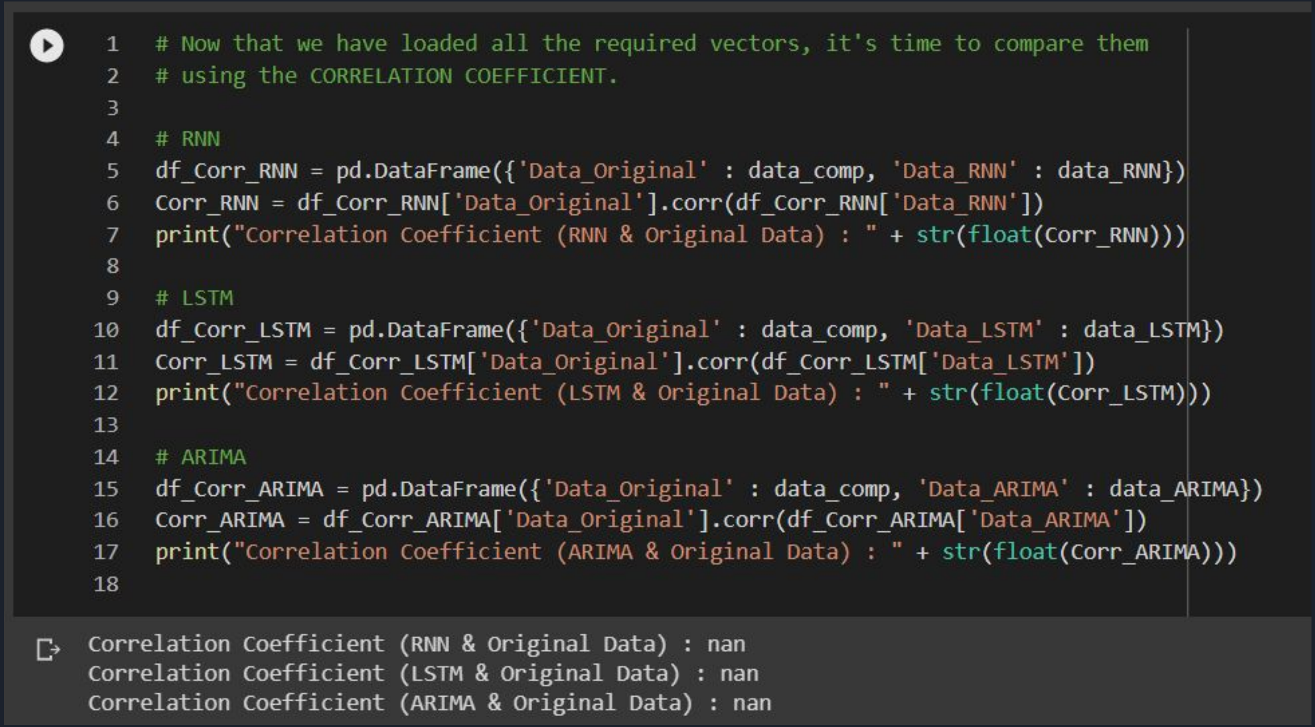


Now that we have predicted the desired values through all the 3 models - RNN, LSTM and ARIMA and also evaluated their performance using metrics like - RMSE Score and R^2 (R-square) score, these can be tough to explain to an unspohisticated brain (someone who might not be having expertise in ML). Hence, I will be comparing these predicted results on the basis of numerical measure known as, `CORRELATION COEFFICIENT`. The relationship between the ***`correlation coefficient matrix`***, **R**, and the ***`covariance matrix`***, **C**, is 🔽

$$R_{ij} = \frac{C_{ij}}{\sqrt{C_{ii} C_{jj}}}$$

NOTE ⚡ : ***`Covariance Matrix`*** is a square matrix giving the covariance between each pair of elements of a given random vector / 2 vectors (comparing elements pair-wise from both the vectors).

RESULTS



```python
# Now that we have loaded all the required vectors, it's time to compare them
# using the CORRELATION COEFFICIENT.

# RNN
df_Corr_RNN = pd.DataFrame({'Data_Original' : data_comp, 'Data_RNN' : data_RNN})
Corr_RNN = df_Corr_RNN['Data_Original'].corr(df_Corr_RNN['Data_RNN'])
print("Correlation Coefficient (RNN & Original Data) : " + str(float(Corr_RNN)))

# LSTM
df_Corr_LSTM = pd.DataFrame({'Data_Original' : data_comp, 'Data_LSTM' : data_LSTM})
Corr_LSTM = df_Corr_LSTM['Data_Original'].corr(df_Corr_LSTM['Data_LSTM'])
print("Correlation Coefficient (LSTM & Original Data) : " + str(float(Corr_LSTM)))

# ARIMA
df_Corr_ARIMA = pd.DataFrame({'Data_Original' : data_comp, 'Data_ARIMA' : data_ARIMA})
Corr_ARIMA = df_Corr_ARIMA['Data_Original'].corr(df_Corr_ARIMA['Data_ARIMA'])
print("Correlation Coefficient (ARIMA & Original Data) : " + str(float(Corr_ARIMA)))

```

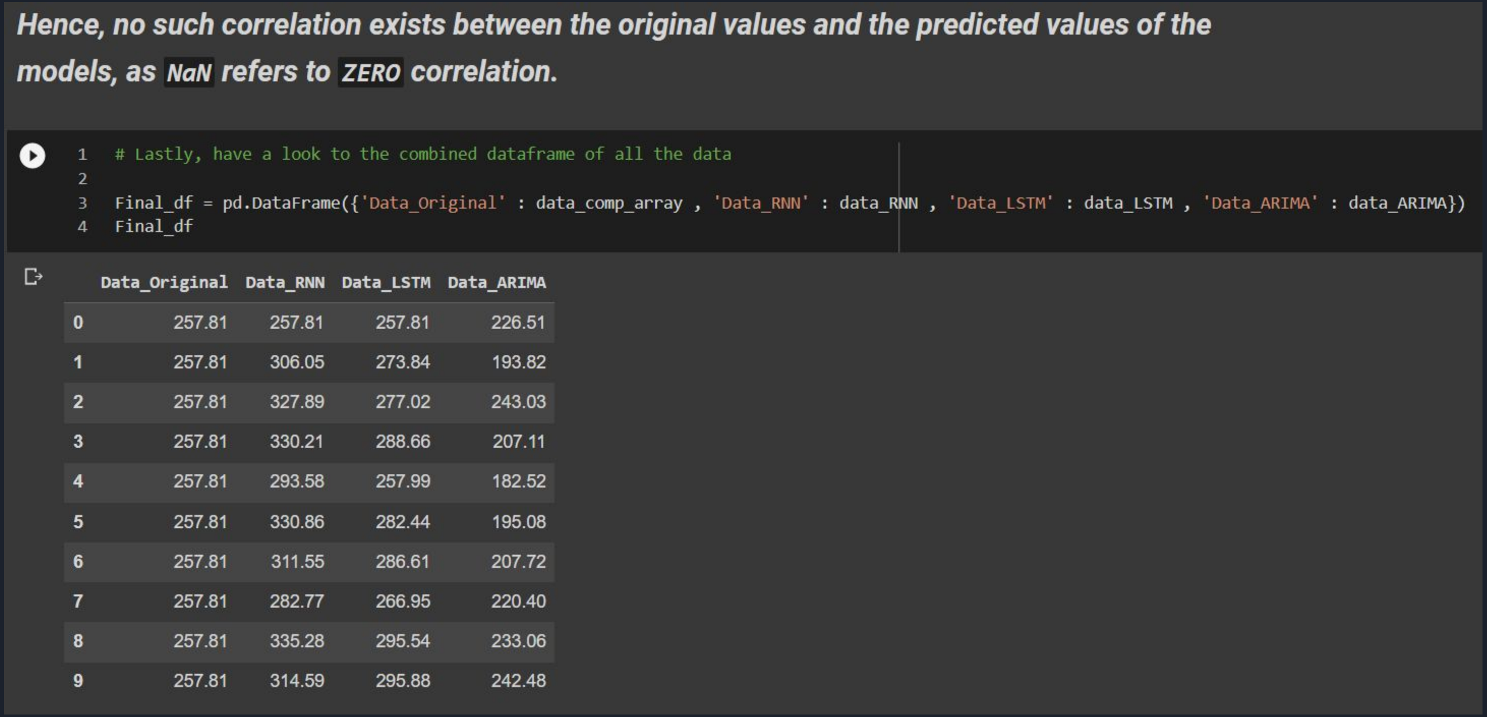
```
Correlation Coefficient (RNN & Original Data) : nan
Correlation Coefficient (LSTM & Original Data) : nan
Correlation Coefficient (ARIMA & Original Data) : nan
```

RESULTS

- If you observe, the correlation coefficient for all the models' prediction values comes out to be **NaN (Not–a-Number)** value, which is equivalent to **ZERO.**

- Hence, there is **no correlation among the predicted data and the original data.**

RESULTS

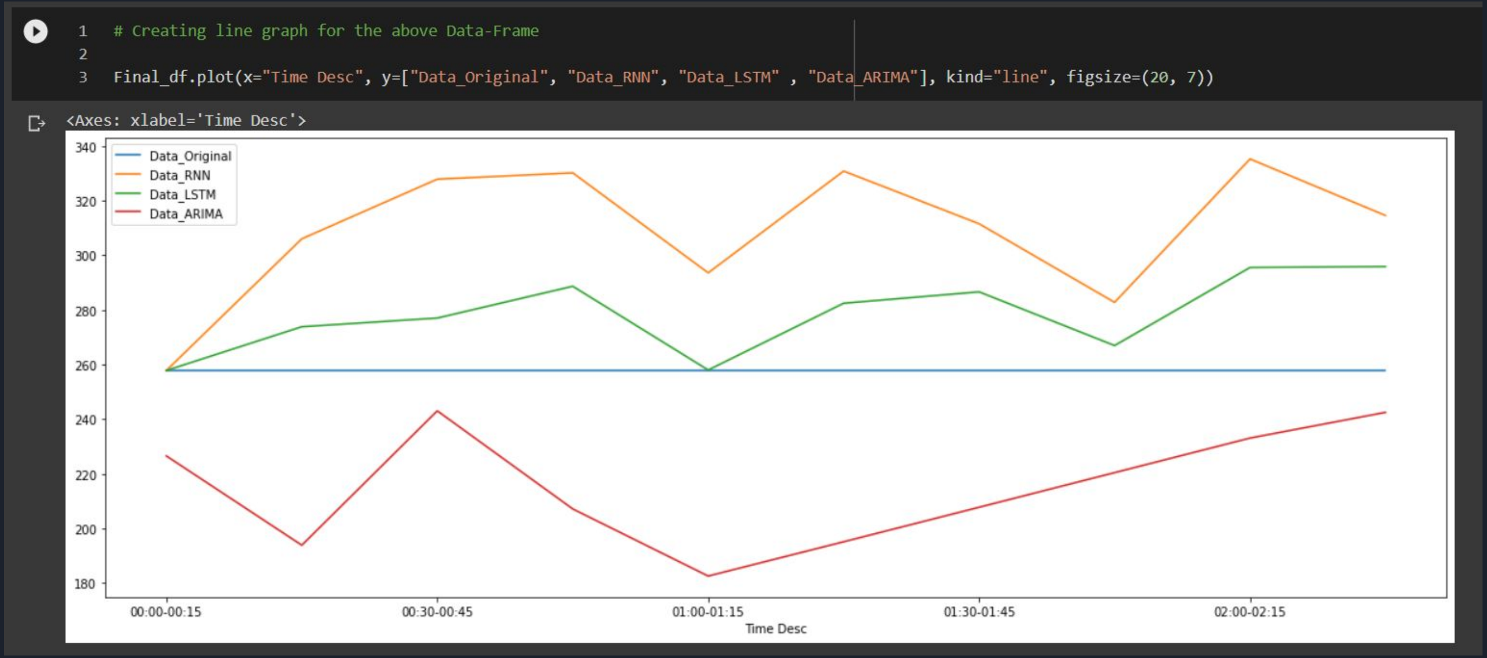


Hence, no such correlation exists between the original values and the predicted values of the models, as `NaN` refers to `ZERO` correlation.

```
# Lastly, have a look to the combined dataframe of all the data

Final_df = pd.DataFrame({'Data_Original' : data_comp_array , 'Data_RNN' : data_RNN , 'Data_LSTM' : data_LSTM , 'Data_ARIMA' : data_ARIMA})
Final_df
```

	Data_Original	Data_RNN	Data_LSTM	Data_ARIMA
0	257.81	257.81	257.81	226.51
1	257.81	306.05	273.84	193.82
2	257.81	327.89	277.02	243.03
3	257.81	330.21	288.66	207.11
4	257.81	293.58	257.99	182.52
5	257.81	330.86	282.44	195.08
6	257.81	311.55	286.61	207.72
7	257.81	282.77	266.95	220.40
8	257.81	335.28	295.54	233.06
9	257.81	314.59	295.88	242.48

RESULTS



```
# Creating line graph for the above Data-Frame

Final_df.plot(x="Time Desc", y=["Data_Original", "Data_RNN", "Data_LSTM" , "Data_ARIMA"], kind="line", figsize=(20, 7))
```

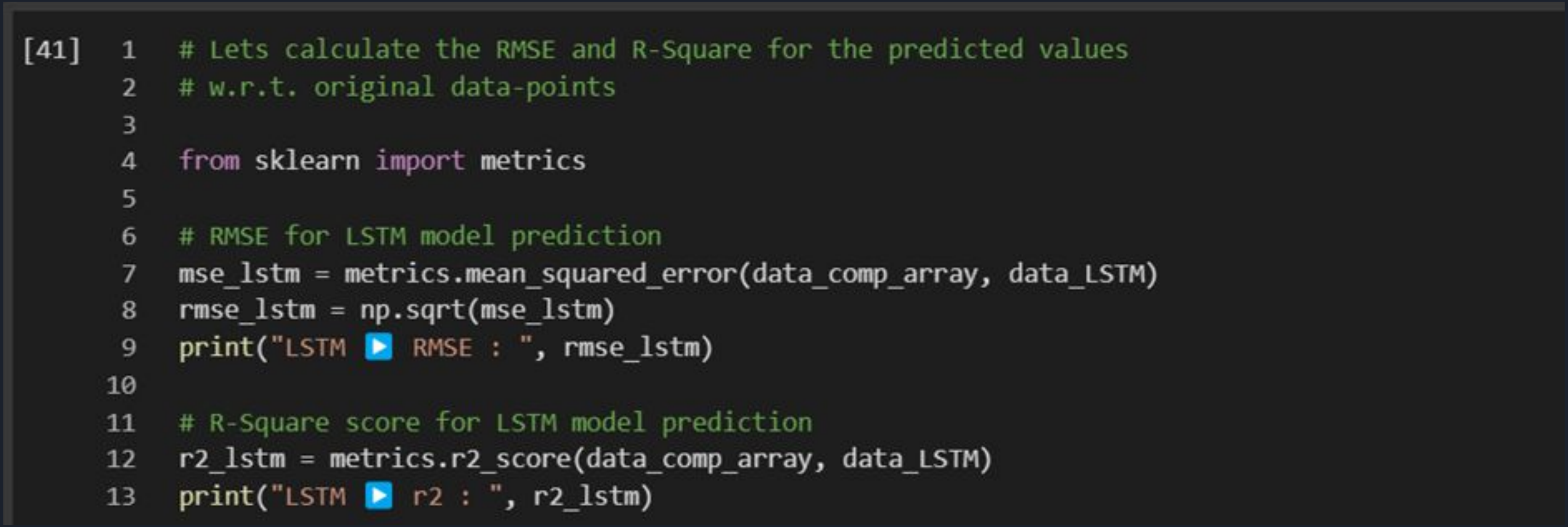
```
<Axes: xlabel='Time Desc'>
```

RESULTS

Possible explanation for the plot :

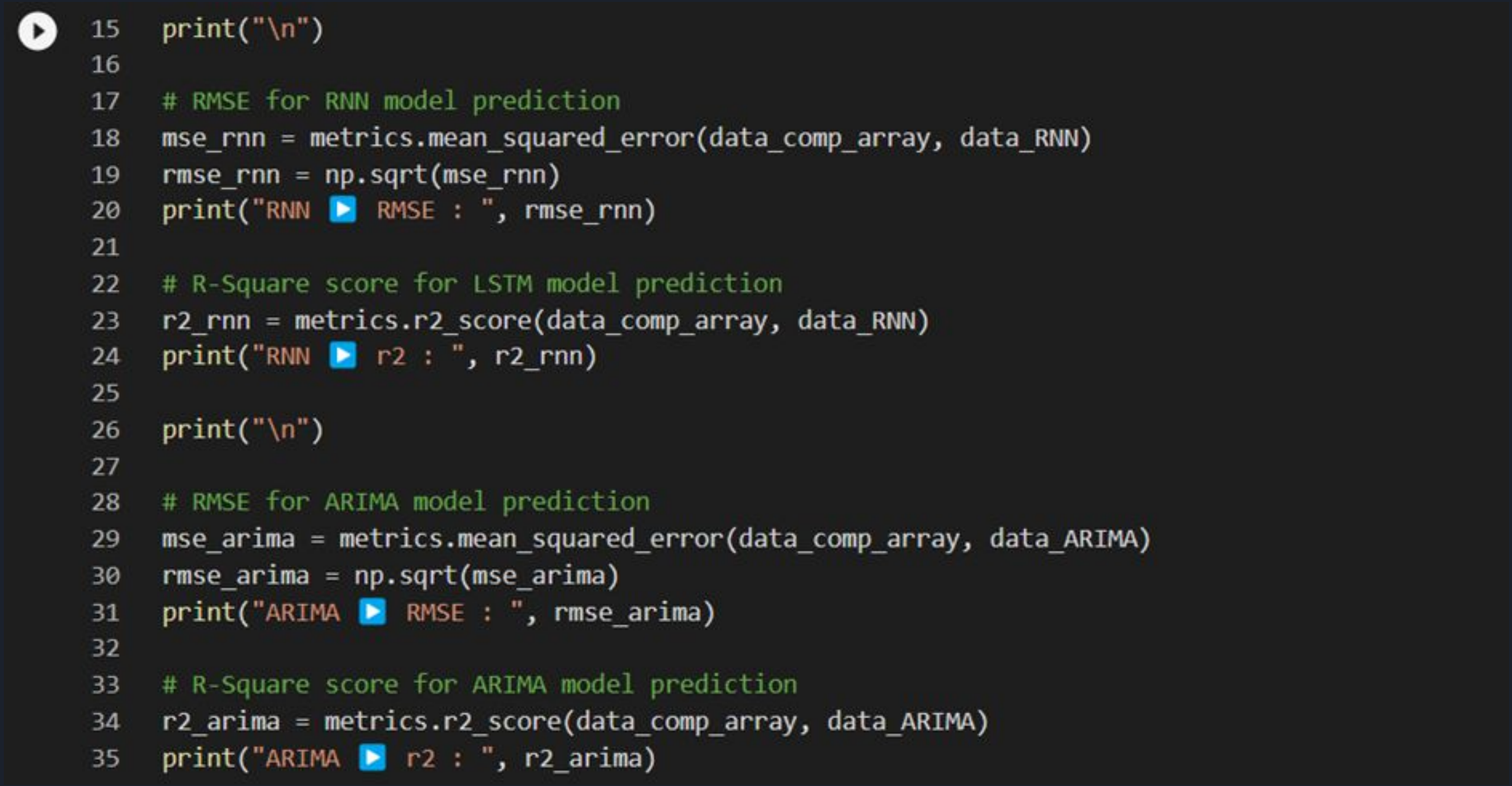
- ARIMA was fed with stationary (or “De-Trended” data); till the component of **Training-n-Testing** procedure it showed promising results. But when going to the step of prediction, the results were pretty abrupt, showing huge deviation from the original data (i.e., error rate is high). The possible cause is that ARIMA cannot handle small datasets - it needs extremely large datasets (something extracted from data-warehouses) to work efficiently.
- Though RNN and LSTM dominates, RNN has the problem of vanishing gradient even for smaller no. of data points. Hence, LSTM dominated among all and wins the game.

RESULTS



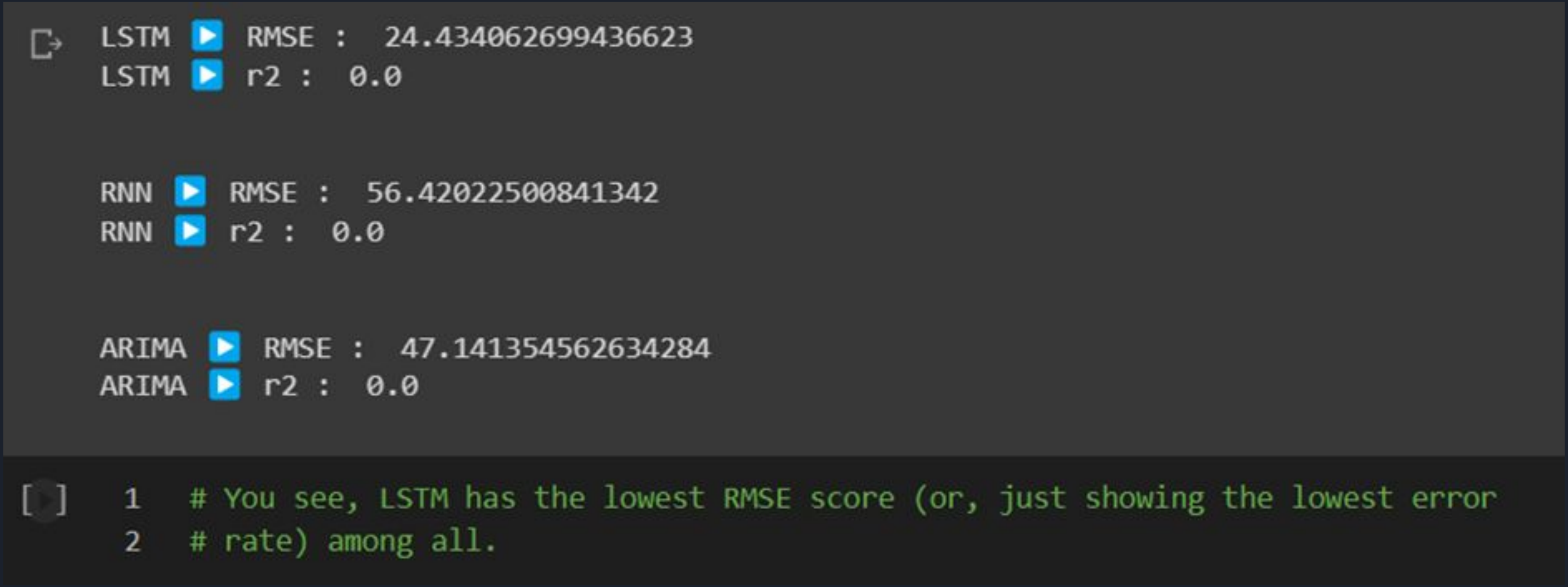
```
[41]
# Lets calculate the RMSE and R-Square for the predicted values
# w.r.t. original data-points

from sklearn import metrics

# RMSE for LSTM model prediction
mse_lstm = metrics.mean_squared_error(data_comp_array, data_LSTM)
rmse_lstm = np.sqrt(mse_lstm)
print("LSTM ▶ RMSE : ", rmse_lstm)

# R-Square score for LSTM model prediction
r2_lstm = metrics.r2_score(data_comp_array, data_LSTM)
print("LSTM ▶ r2 : ", r2_lstm)
```

RESULTS



```
print("\n")

# RMSE for RNN model prediction
mse_rnn = metrics.mean_squared_error(data_comp_array, data_RNN)
rmse_rnn = np.sqrt(mse_rnn)
print("RNN ▶ RMSE : ", rmse_rnn)

# R-Square score for LSTM model prediction
r2_rnn = metrics.r2_score(data_comp_array, data_RNN)
print("RNN ▶ r2 : ", r2_rnn)

print("\n")

# RMSE for ARIMA model prediction
mse_arima = metrics.mean_squared_error(data_comp_array, data_ARIMA)
rmse_arima = np.sqrt(mse_arima)
print("ARIMA ▶ RMSE : ", rmse_arima)

# R-Square score for ARIMA model prediction
r2_arima = metrics.r2_score(data_comp_array, data_ARIMA)
print("ARIMA ▶ r2 : ", r2_arima)
```

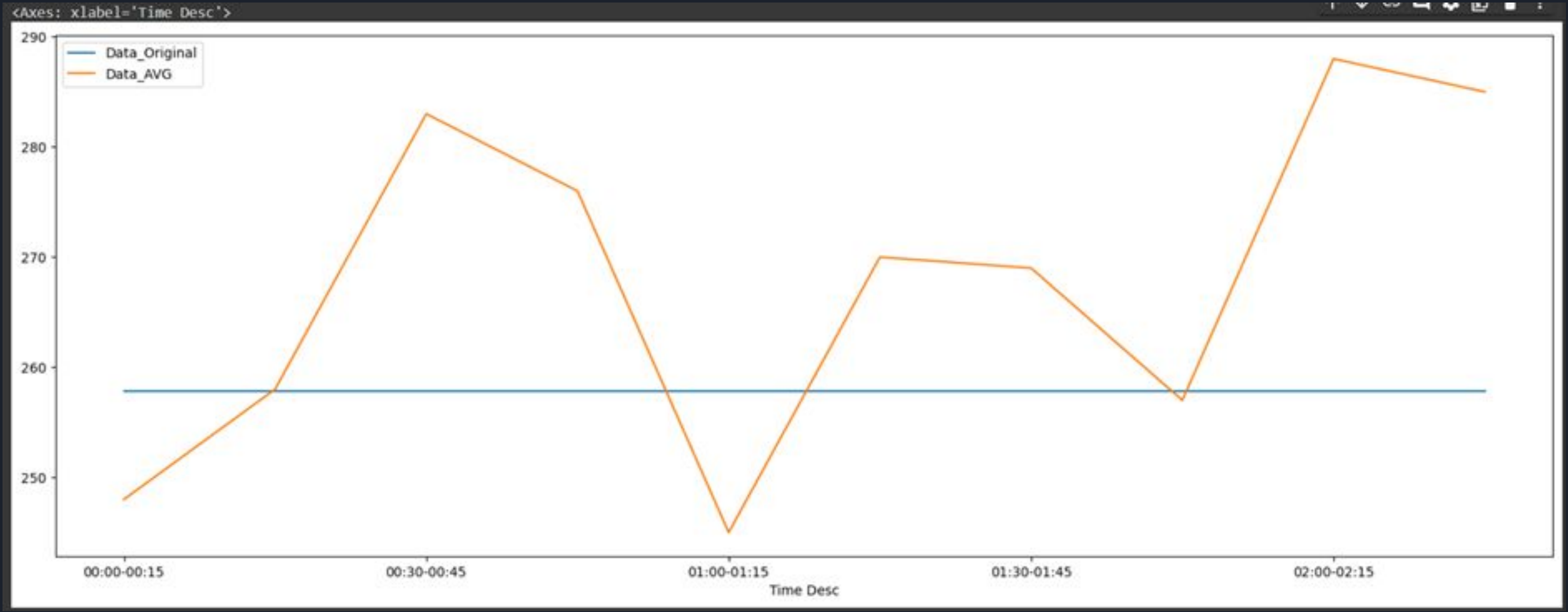
RESULTS



```
LSTM ▶ RMSE :  24.434062699436623
LSTM ▶ r2 :  0.0


RNN ▶ RMSE :  56.42022500841342
RNN ▶ r2 :  0.0


ARIMA ▶ RMSE :  47.141354562634284
ARIMA ▶ r2 :  0.0
```

```
# You see, LSTM has the lowest RMSE score (or, just showing the lowest error
# rate) among all.
```

RESULTS


```
<Axes: xlabel='Time Desc'>
```

Data_Original
Data_AVG
290
280
270
260
250
00:00-00:15
00:30-00:45
01:00-01:15
01:30-01:45
02:00-02:15
Time Desc

CONCLUSION

- Efficient energy production is important to produce electricity at an affordable cost (cost that not just include money, but time and effort).
- Inculcated three “Time-Series” algorithms - RNN, LSTM and ARIMA to analyse and predict power production data.
- ARIMA can only deal successfully non-stationary data, hence numerous procedure were employed to “De-Trend” the acquired data.
- Evaluation Metrics considered : RMSE (Root Mean Squared Error) and R-square score - for denoting error rate and model fitting.

CONCLUSION

- The below table shows the performance score according to the evaluation metrics taken :

Algo. ➡️ Metrics ⬇️	RNN	LSTM	ARIMA Model
RMSE	`7791.47`	`7754.76`	`0.33`
R2 (R-square) score	`-30.10`	`-29.81`	`-11.50`

CONCLUSION

- At the time of predicting fresh values , ARIMA showed it’s disadvantages of unable to predict even after training with stationary data. The prime cause for the same is ARIMA cannot efficiently deal with daily OR hourly data (i.e, data of smaller time-slabs), but rather won’t produce much error rate with monthly or yearly data. But industries are more concerned about “How much to produce the very next day”.
- Even though RNN and LSTM dominates, because of the fact that RNN has the vanishing gradient problem, which causes it to reduce the memory efficiency, which eventually won’t try to memorize very old / initially fed data, LSTM wins the game. But the project doesn’t end here; I took the average of the predicted values of all the models and compared that to the original data, which turned out to be offering lower error rate.

PROJECT OUTCOME

- Beneficiaries : The power sector industries, especially the nationalised power companies.
- The project :
  - Will increase the power production output
  - Minimize the cost of equipment failure
  - Reduce the excessive coal buying errors
  - Will enhance overall safety

KEY TAKE-AWAY

- Above-rudimentary-level understanding on “Time-Series” concept and algorithms.
- The kind of data to deal with.
- An exploration ability to find appropriate data source
- Correct usage of python libraries
- Data analysis and visualization skills (concentrating on matplotlib.pyplot)

- Storytelling

FUTURE WORK

- Implementation of an API, consisting of the desired trained model and inculcating the API in websites (or any other software OR web-based services).
- Including the API in power sector organization's websites along with the presence of appropriate data to directly feed according to the parameters needed / asked for.
- Analysing streaming data along with batch data (if historical data is included for long term analysis). For streaming data algorithms like RNN shall perform better, but eventually activity shall be passed on to LSTM to produce minimum loss value and minimum error rates. If historical stationary data (like of yearly or monthly) is considered, ARIMA may dominate.

REFERENCES

[1] Mohammad-Rasool Kazemzadeh, Ali Amjadian, Turaj Amraee. (2020). “A hybrid data mining driven algorithm for long term electric peak load and energy demand forecasting”. Energy, Volume 204, 117948, ISSN 0360-5442, https://doi.org/10.1016/j.energy.2020.117948.

[2] Ranran Li, Ping Jiang, Hufang Yang, Chen Li. (2020). “A novel hybrid forecasting scheme for electricity demand time series”. Sustainable Cities and Society, Volume 55, 102036, ISSN 2210-6707, https://doi.org/10.1016/j.scs.2020.102036.

[3] Weiheng Jiang, Xiaogang Wu, Yi Gong, Wanxin Yu, Xinhui Zhong. (2020). “Holt–Winters smoothing enhanced by fruit fly optimization algorithm to forecast monthly electricity consumption”. Energy, Volume 193, 116779, ISSN 0360-5442, https://doi.org/10.1016/j.energy.2019.116779.

[4] Evangelos Spiliotis, Fotios Petropoulos, Nikolaos Kourentzes, Vassilios Assimakopoulos. (2020). “Cross-temporal aggregation: Improving the forecast accuracy of hierarchical electricity consumption”. Applied Energy, Volume 261, 114339, ISSN 0306-2619, https://doi.org/10.1016/j.apenergy.2019.114339.

[5] Shalika Walker, Waqas Khan, Katarina Katic, Wim Maassen, Wim Zeiler. (2020). “Accuracy of different machine learning algorithms and added-value of predicting aggregated-level energy performance of commercial buildings”. Energy and Buildings, Volume 209, 109705, ISSN 0378-7788, https://doi.org/10.1016/j.enbuild.2019.109705.

REFERENCES

[6] Lei Lei, Wei Chen, Bing Wu, Chao Chen, Wei Liu. (2021). “A building energy consumption prediction model based on rough set theory and deep learning algorithms”. Energy and Buildings, Volume 240, 110886, ISSN 0378-7788, https://doi.org/10.1016/j.enbuild.2021.110886.

[7] Mel Keytingan M. Shapi, Nor Azuana Ramli, Lilik J. Awalin. (2021). “Energy consumption prediction by using machine learning for smart building: Case study in Malaysia”. Developments in the Built Environment, Volume 5, 100037, ISSN 2666-1659, https://doi.org/10.1016/j.dibe.2020.100037.

[8] S. Bowala, M. Makhan, Y. Liang, A. Thavaneswaran and S. S. Appadoo. (2022). "Superiority of the Neural Network Dynamic Regression Models for Ontario Electricity Demand Forecasting". IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), pp. 182-187, doi: 10.1109/CCECE49351.2022.9918212.

[9] A. Arjomandi-Nezhad, A. Ahmadi, S. Taheri, M. Fotuhi-Firuzabad, M. Moeini-Aghtaie and M. Lehtonen. (2022). "Pandemic-Aware Day-Ahead Demand Forecasting Using Ensemble Learning". IEEE Access, vol. 10, pp. 7098-7106, 2022, doi: 10.1109/ACCESS.2022.3142351.

[10] Y. E. UNUTMAZ, A. DEMİRCİ, S. M. Tercan and R. Yumurtaci. (2021). "Electrical Energy Demand Forecasting Using Artificial Neural Network". 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), 2021, pp. 1-6, doi: 10.1109/HORA52670.2021.9461186.

Thank You