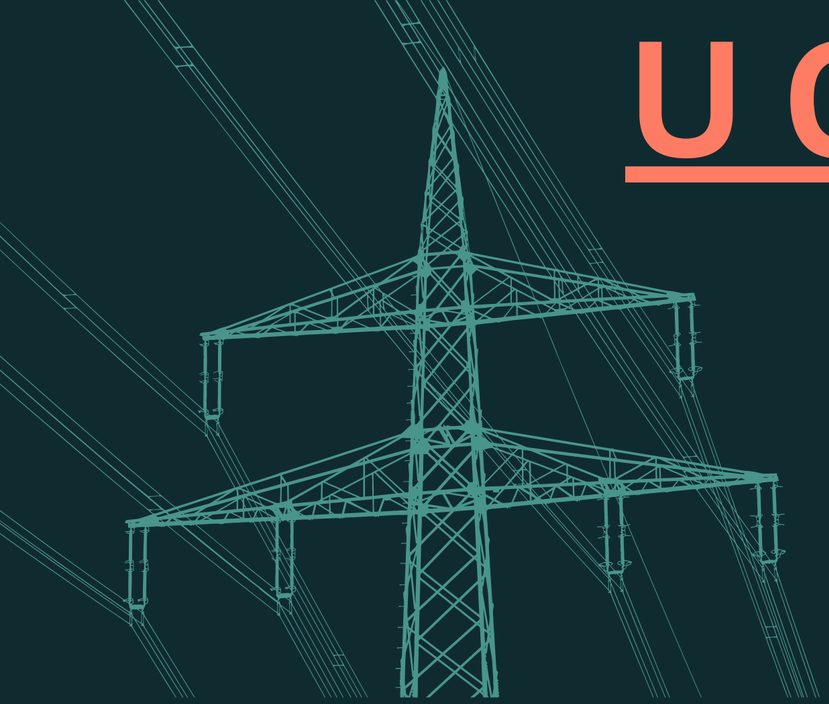
UG PROJECT



Short-term Load Forecasting with Machine Learning

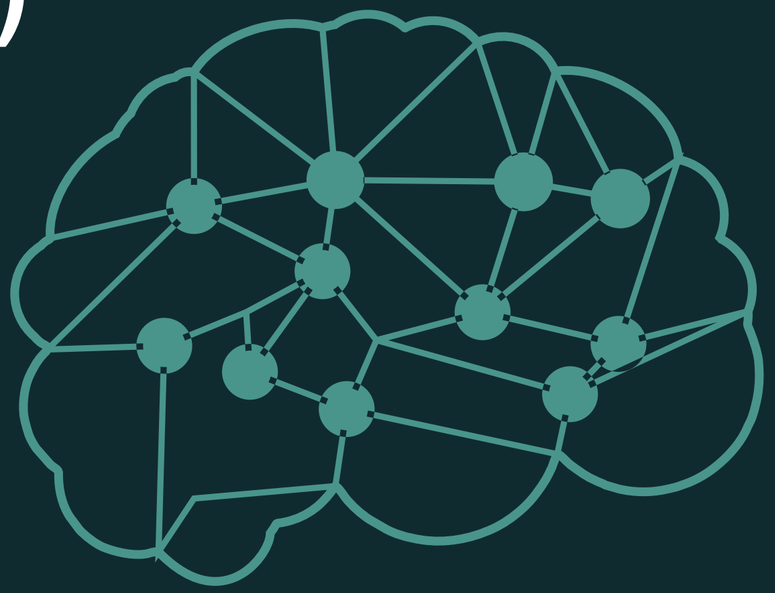
Under Dr. Avirup Maulik,

Department of Electrical Engineering,

Indian Institute of Technology(BHU), Varanasi

Team Members

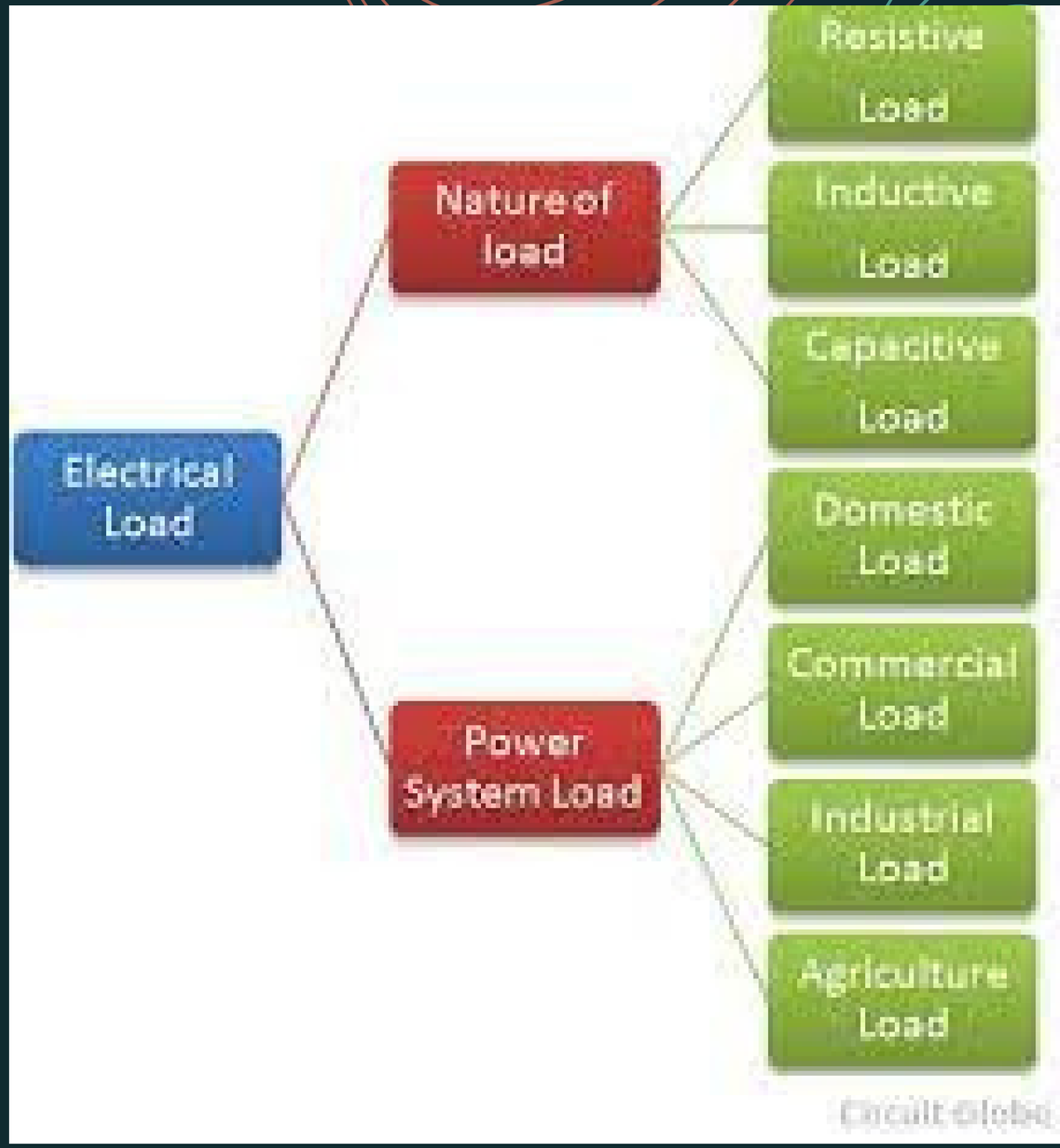
Navapallav Borthakur (20085123)
Vedansh Pandey (20085120)
Vaibhav Joshi (20085118)



INDEX:

What is a Load?

An electrical load is an electrical component or portion of a circuit that consumes (active) electric power, such as electrical appliances and lights inside the home. The term may also refer to the power consumed by a circuit.

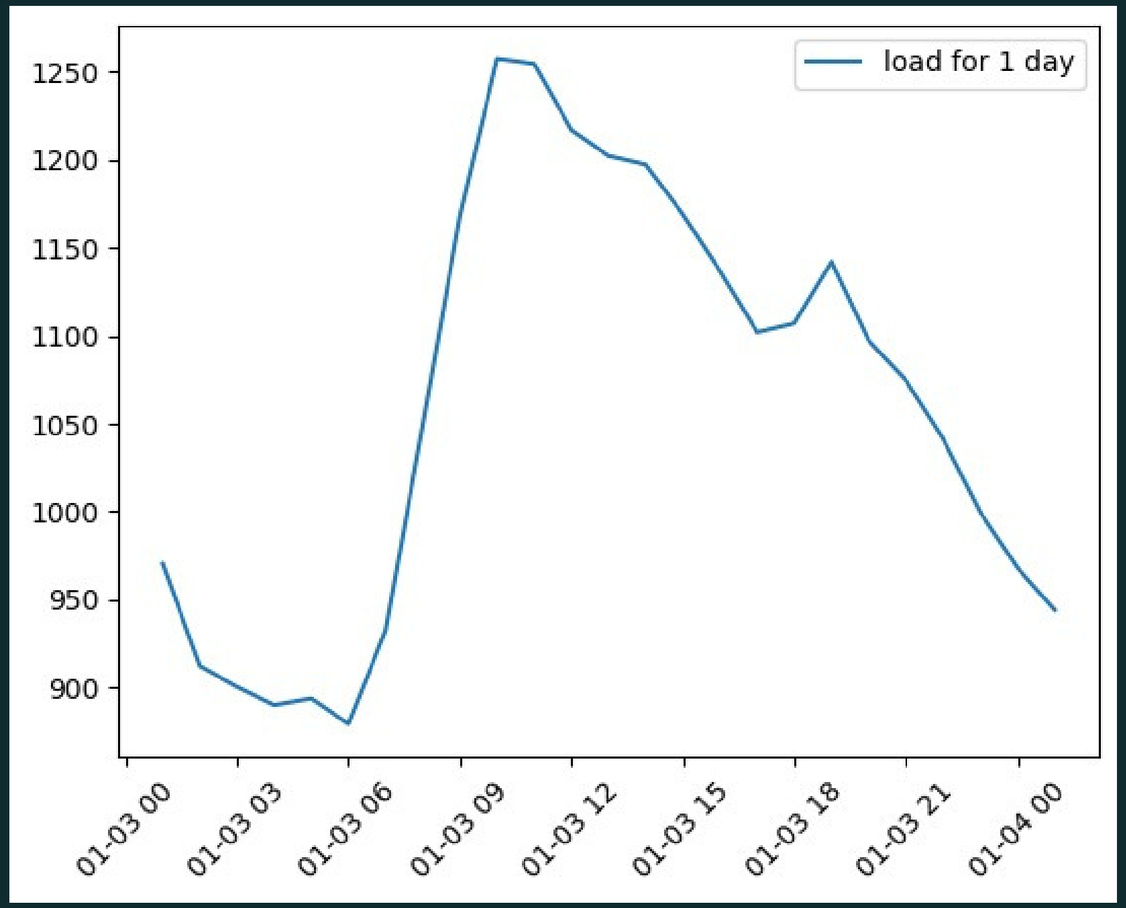


Load Curves:

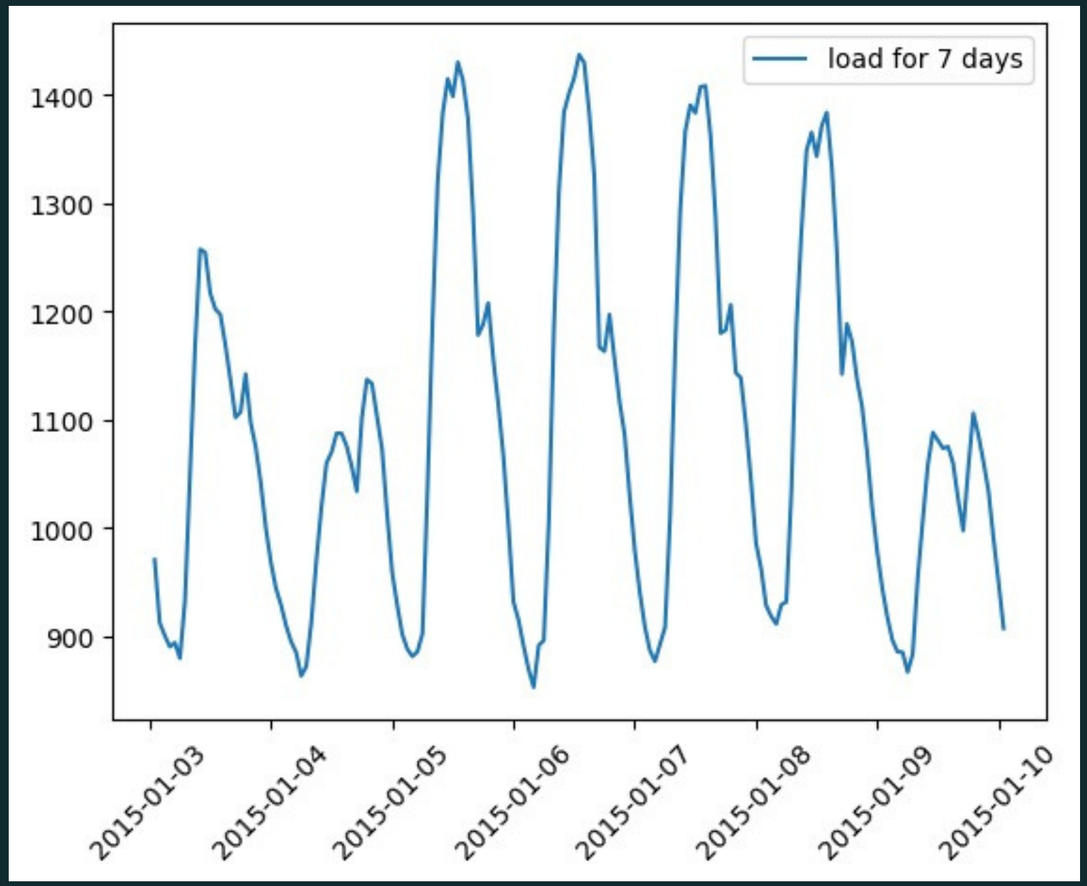
- Variations in load on power station from time to time.
  - Daily load curves.
  - Monthly load curves.
  - Annual load curves.
- Load curves give –
  - Variation of load during different times.
  - Total no. of units generated.
  - Maximum demand.
  - Average load on a power station.
  - Load Factor.

LOAD CURVES:

FOR 1 DAY:



FOR 7 DAYS:



LOAD FORECASTING:

• Forecasting is the problem of predicting future values of a time series based on current and previous values.
• Load forecasting is the process of estimating future consumptions based available data and information about consumer behaviour.
• Short-term load forecasting can assist in estimating load flows and making decisions to avoid overloading. Implementing such decisions on time improves network reliability and reduces the occurrence of equipment failures or blackouts.

Role of load forecasting:

- **For Proper planning of power system:**
  - To determine the potential need of additional generating facilities
  - To determine the size of the plants.
  - On correct values of demands, load forecasting will prevent over designing of conductor size.
- **For proper planning of Transmission and Distribution facilities**
  - Wastage bacause of unplanned purchase of equipments can be avoided.
- **For proper financing**
- **For proper Grid Formation**

Factors affecting Load Forecasting

Time Factors-
- Hours of the day
- Day of the week
- Time of the year

Weather Conditions:
- Temperature
- Humidity

Types of costumers:
- Residential
- Commercial
- Industrial
- Agriculture

Special Events:
- Public Holidays

DATA USED:

datetime	temp_2M	humidity_2M	precipitation_2M	wind_speed_2M	holiday	net_demand
1/3/2015 1:00	25.86525879	0.018576382	0.016174316	21.85054582	0	970.345
1/3/2015 2:00	25.89925537	0.018653292	0.016418457	22.16694428	0	912.1755
1/3/2015 3:00	25.93728027	0.01876786	0.015480042	22.45491088	0	900.2688
1/3/2015 4:00	25.95754395	0.018890057	0.016273499	22.11048087	0	889.9538
1/3/2015 5:00	25.97384033	0.018981151	0.017280579	21.18608852	0	893.6865
1/3/2015 6:00	26.03414307	0.019079996	0.014541626	20.06203782	0	879.2323
1/3/2015 7:00	26.6914917	0.019331587	0.006645203	21.62349567	0	932.4876
1/3/2015 8:00	27.67406616	0.019369703	0.00686264	23.775317	0	1048.972
1/3/2015 9:00	28.76040039	0.019171301	0.010231018	24.6361515	0	1167.9074
1/3/2015 10:00	29.76665649	0.018759238	0.009017944	25.86267136	0	1257.5069
1/3/2015 11:00	30.52376709	0.018408278	0.004322052	26.82808151	0	1254.583
1/3/2015 12:00	30.98180542	0.018133579	0.00279808	26.65431045	0	1216.9004

MACHINE LEARNING MODELS USED:

1.RANDOM FOREST MODEL

2.LINEAR REGRESSION MODEL

3.XGBoost(Extreme Gradient Boosting)Model

4.ARIMA(Autoregressive Integrated Moving Average)

5.Neural Network-LSTM Model(long short-term memory networks)

Random Forest Model:

1. Random forest regression is a method that combines multiple decision trees to make a prediction.

2. Each decision tree is trained on a subset of the training data and with a subset of the features.

3.Can handle large datasets with missing data and outliers well.

4.Easy to implement and requires minimal tuning of hyperparameters.

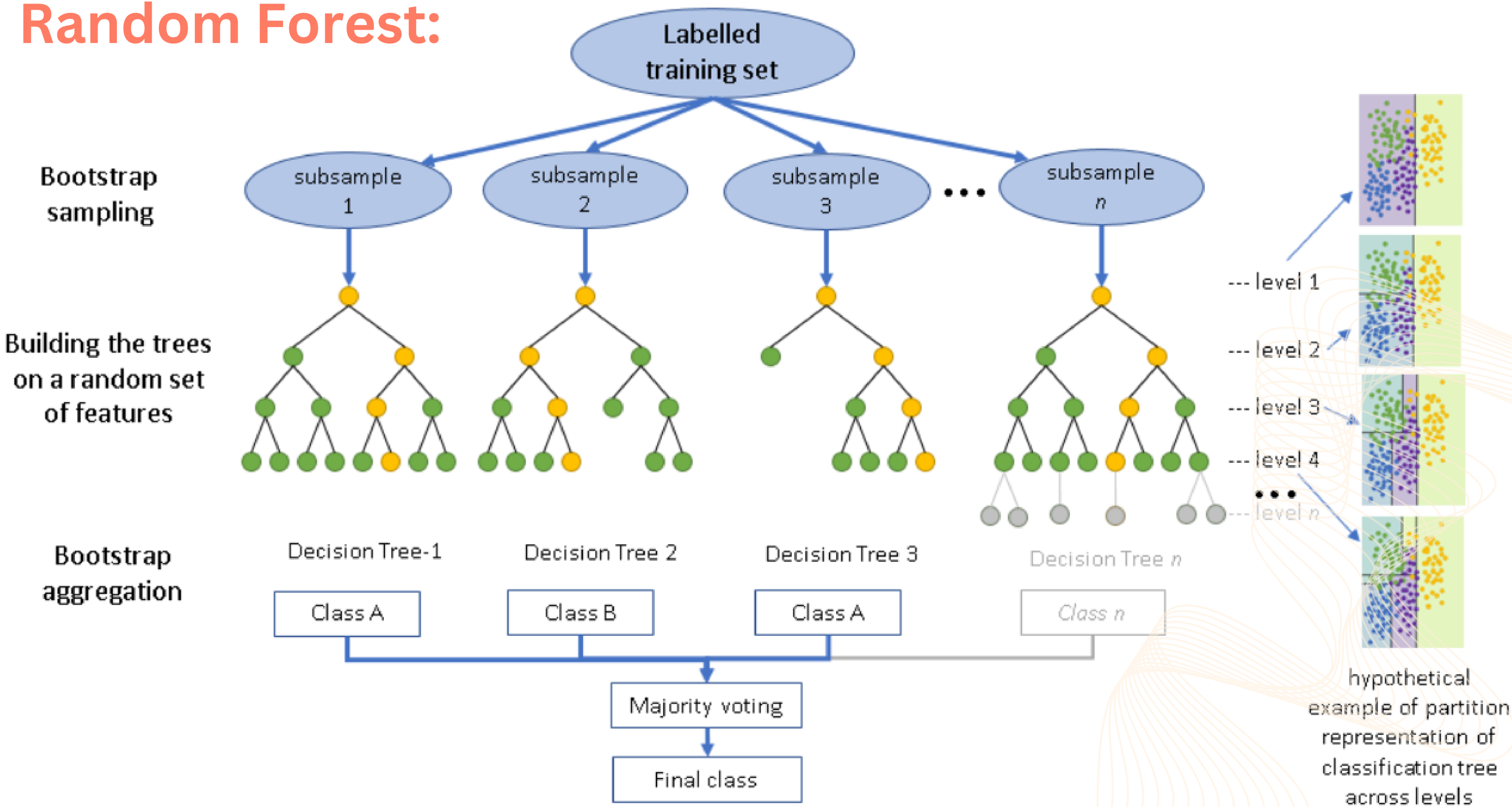
5.can be used for both continuous and categorical variables

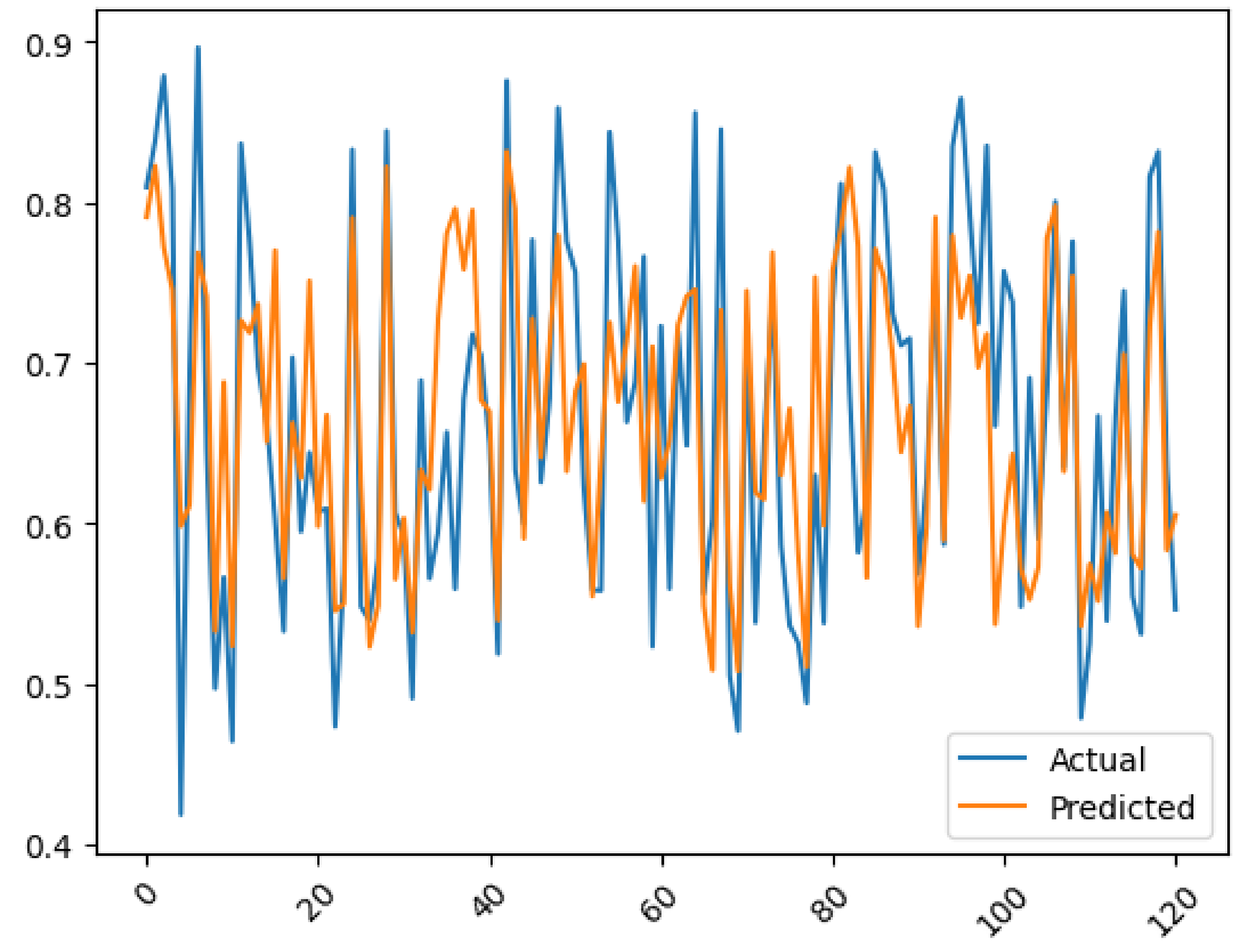
6.Disadvantage: can be slower to train and difficult to interpret.

```
#Import the Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor
#Create a Random Forest regressor object from Random Forest Regressor class
RFReg = RandomForestRegressor(n_estimators = 200, random_state = 0)
#Fit the random forest regressor with training data represented by X_train and y_
RFReg.fit(X_train, y_train)
```

```
RandomForestRegressor
domForestRegressor(n_estimators=200, random_state=0)
```

Random Forest:





Random Forest Regression Output:

Linear Regression Model:

1.models the relationship between a dependent variable and two or more independent variables.
2.The goal is to find the best fitting line that describes the relationship between the dependent variable and the independent variables.
3.The line is determined by estimating the coefficients (or weights) of each independent variable that minimize the sum of squared errors
4.assumes that effect of each independent variable on the dependent variable is independent of the other variables.
5. Easiest algorithm to implement, Takes less time to process

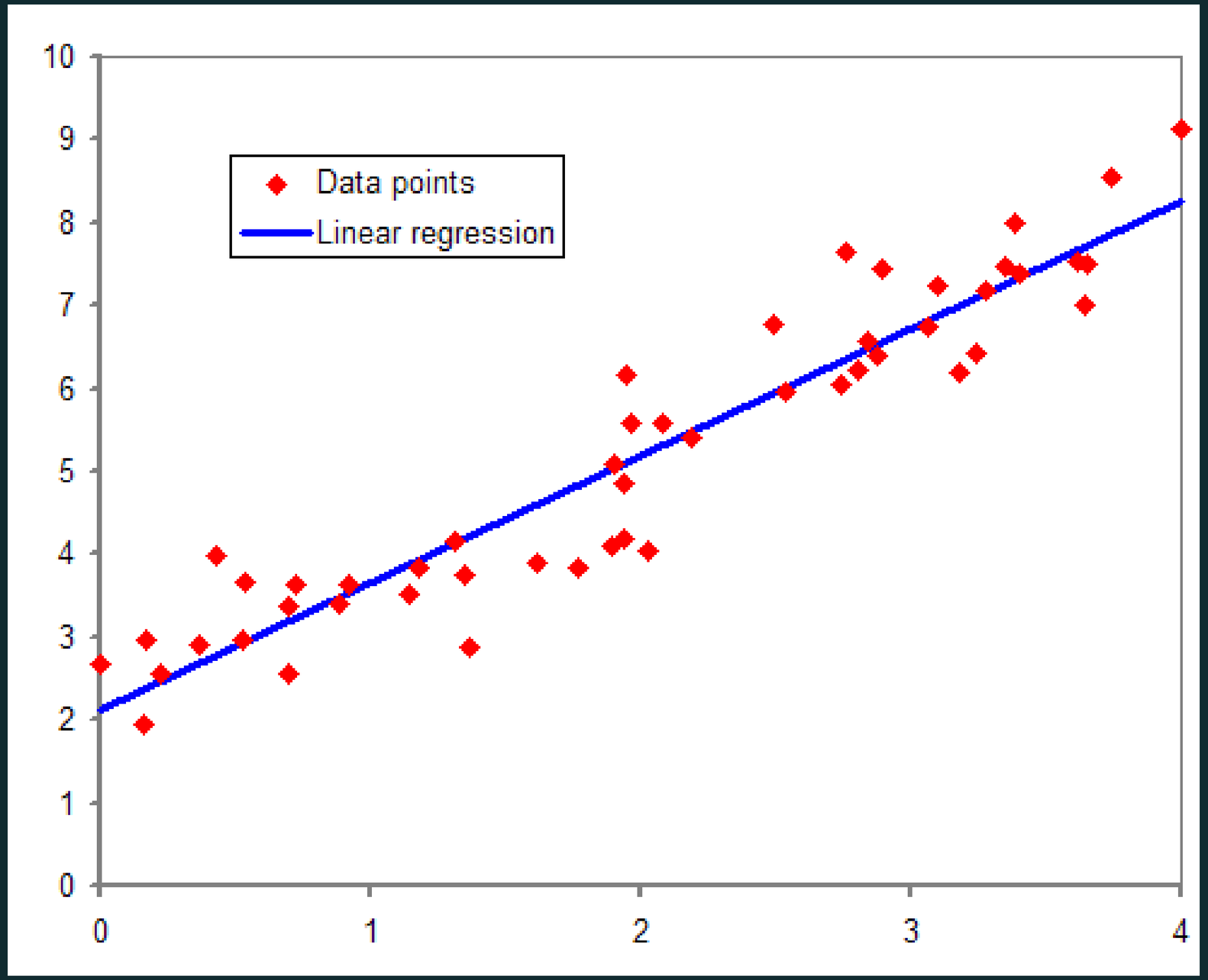
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

```
[ ] y_pred = regressor.predict(X_test)
    np.set_printoptions(precision=4)
```

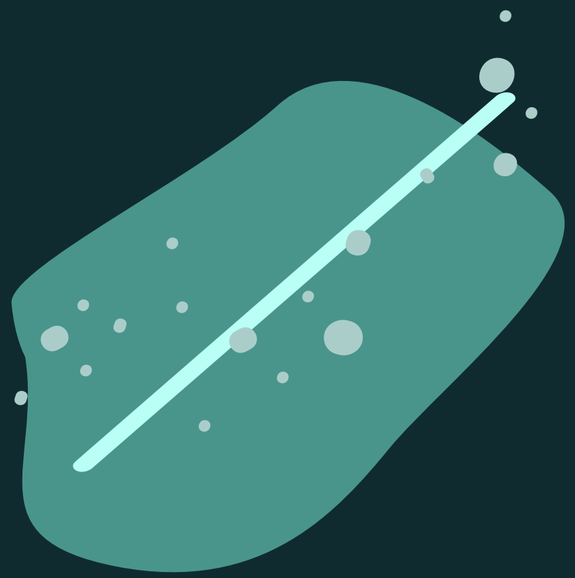
```
[ ] print(y_pred)
```

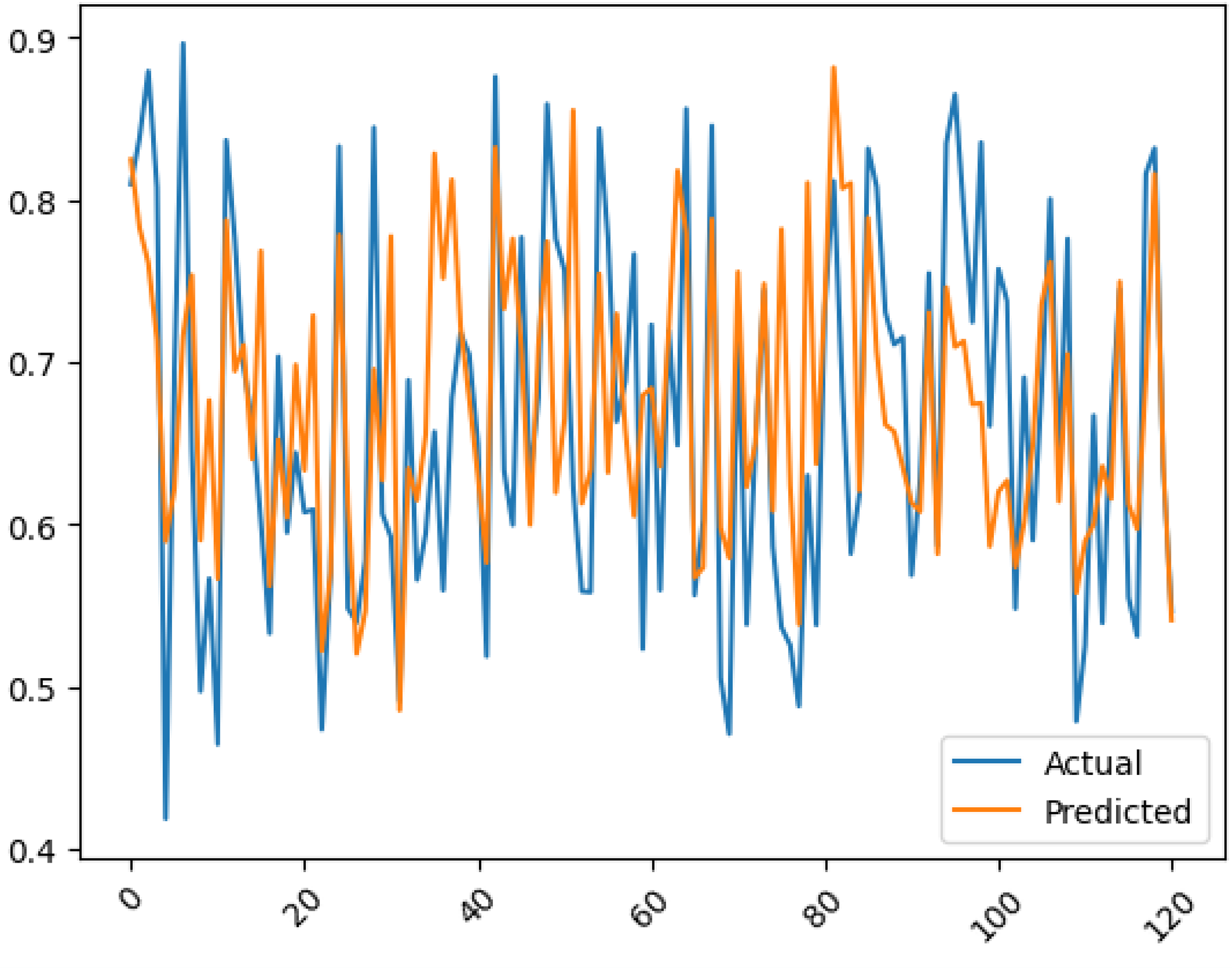
```
    [1461.9109 1391.2678 1355.4106 ... 1034.3933 1103.382  1129.5615]
```

Linear Regression:



Linear Regression Output:

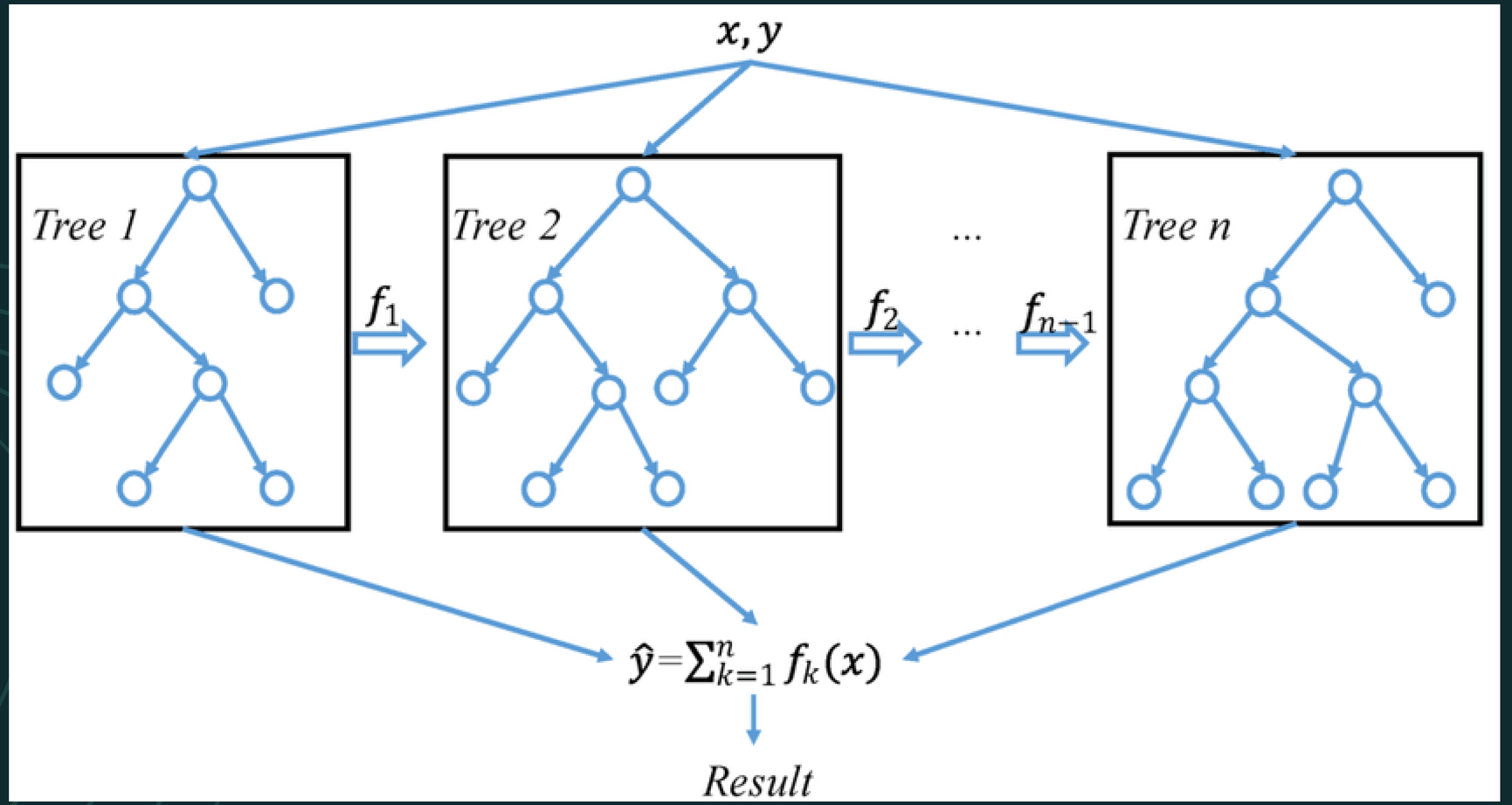




XGBoost model

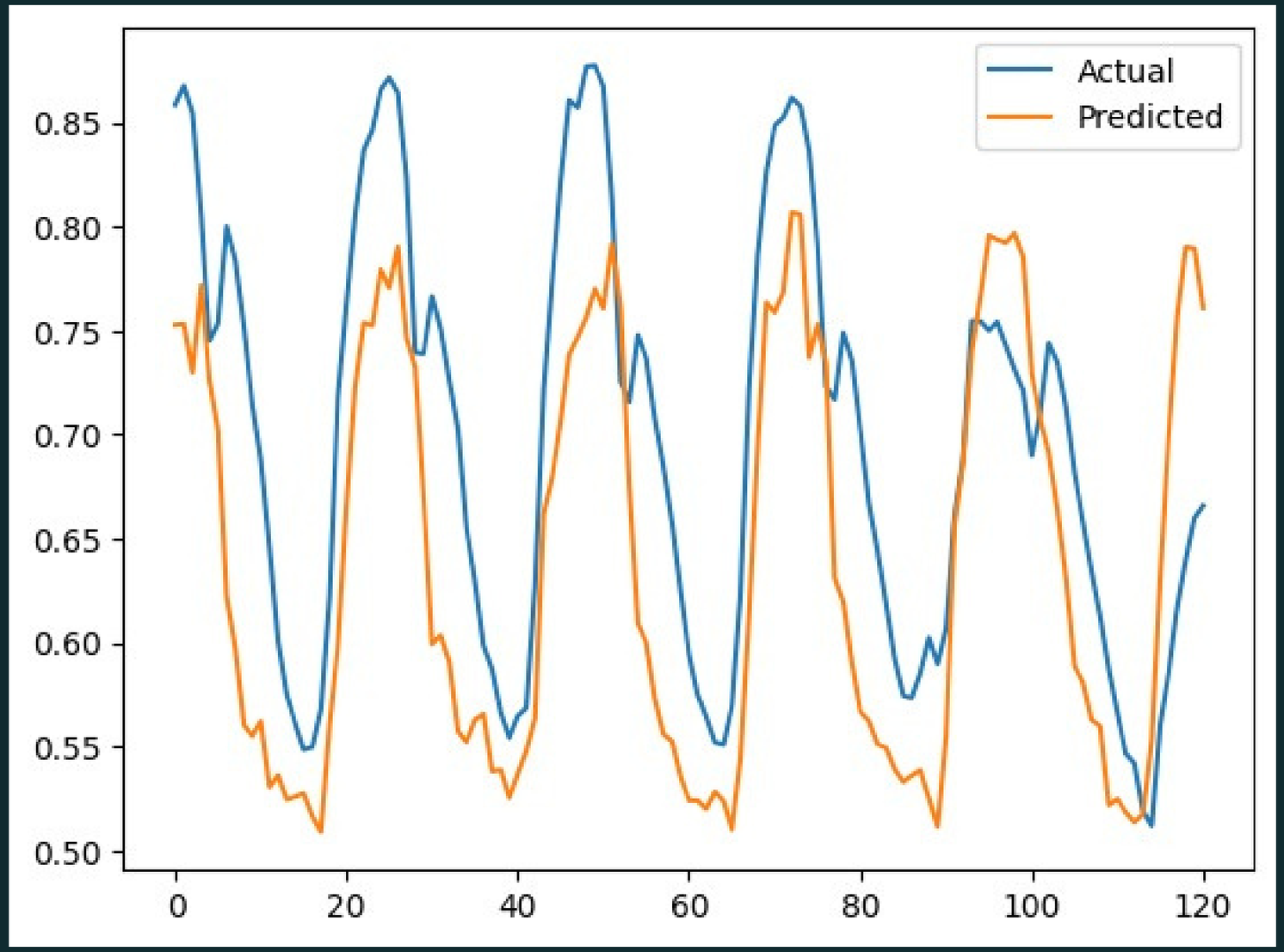
- XGBoost (Extreme Gradient Boosting) is a popular machine learning algorithm that is used for both regression and classification problems.
- It is an implementation of gradient boosting, which is an ensemble learning method that combines several weak learners (e.g., decision trees) to create a strong learner.
- XGBoost is known for its speed and scalability, making it useful for large datasets and real-time applications.
- XGBoost uses a technique called "gradient boosting" to iteratively add new trees to the model, where each new tree tries to correct the errors made by the previous trees.
- XGBoost provides several regularization techniques to prevent overfitting, which can help improve the accuracy of load prediction models.
- XGBoost is known for its speed and scalability, making it useful for large datasets and real-time applications.

XGBoost Model




```
from xgboost import XGBRegressor

Regressor=XGBRegressor(n_estimators =200,random_state = 0)
Regressor.fit(train_X,train_y)
y_pred=Regressor.predict(test_X)
```

XGBoost model output



Time Series

WHAT IS TIME SERIES?

A dataset where magnitude of quantity is measured at different time intervals

- A very common example is the price of a stock over a span of % days.

Day	1	2	3	4	5
Stock Price	200	195	210	215	?

Predicting Stock Value on Day 5

- Remains Same 215
- Average of all (200+195+210+215)/4
- Average of Last 2 values (210+215)/2
- More weight of recent value (215*0.8 + 210*0.2)

Time Series

DIFFERENT FROM TRADITIONAL MACHINE LEARNING ALOGRITHMS

OTHER ML ALGORITHMS (Property Price Prediction)	TIME SERIES (Stock Price Prediction)
Many independant variables, 1 Dependent Variable	No notion of dependent or independent variables, Single Variable
Output depends on independent variables	Output depends on previous values
No relation between to different data points	New data points depand on previous ones.

Time Series

TRAITS OF TIME SERIES

Trend

Seasonality

Unexpected Events

PRE-REQUISITES FOR APPLYING TIME SERIES ALGORITHMS

- Mean of the Series should be constant
- Variance should be constant

ARIMA(Autoregressive Integrated Moving Average):

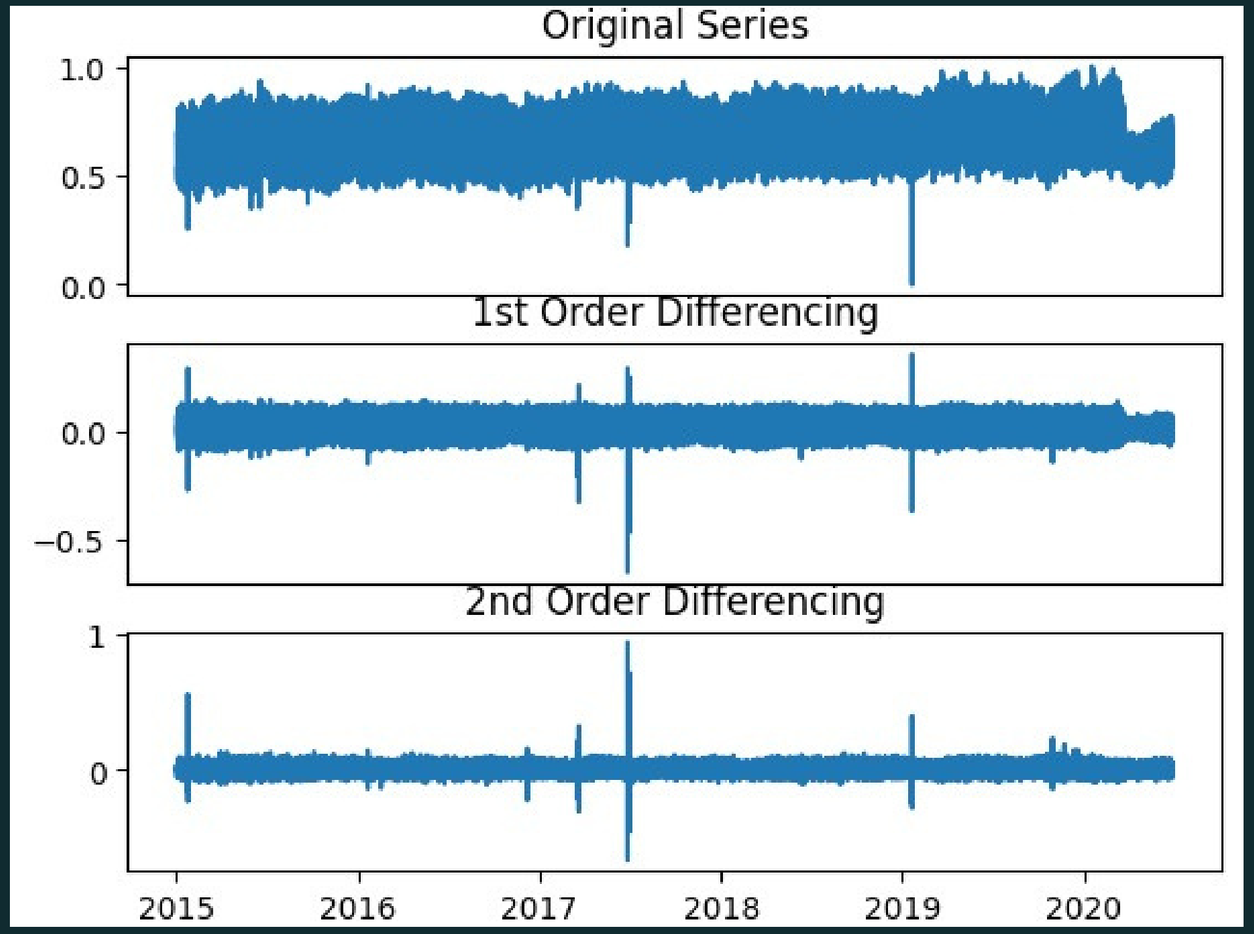
- For I(d)=>

Assume that, the time series is stationary, if not then we can perform transformation and/or differencing of the series to convert the series into a stationary process

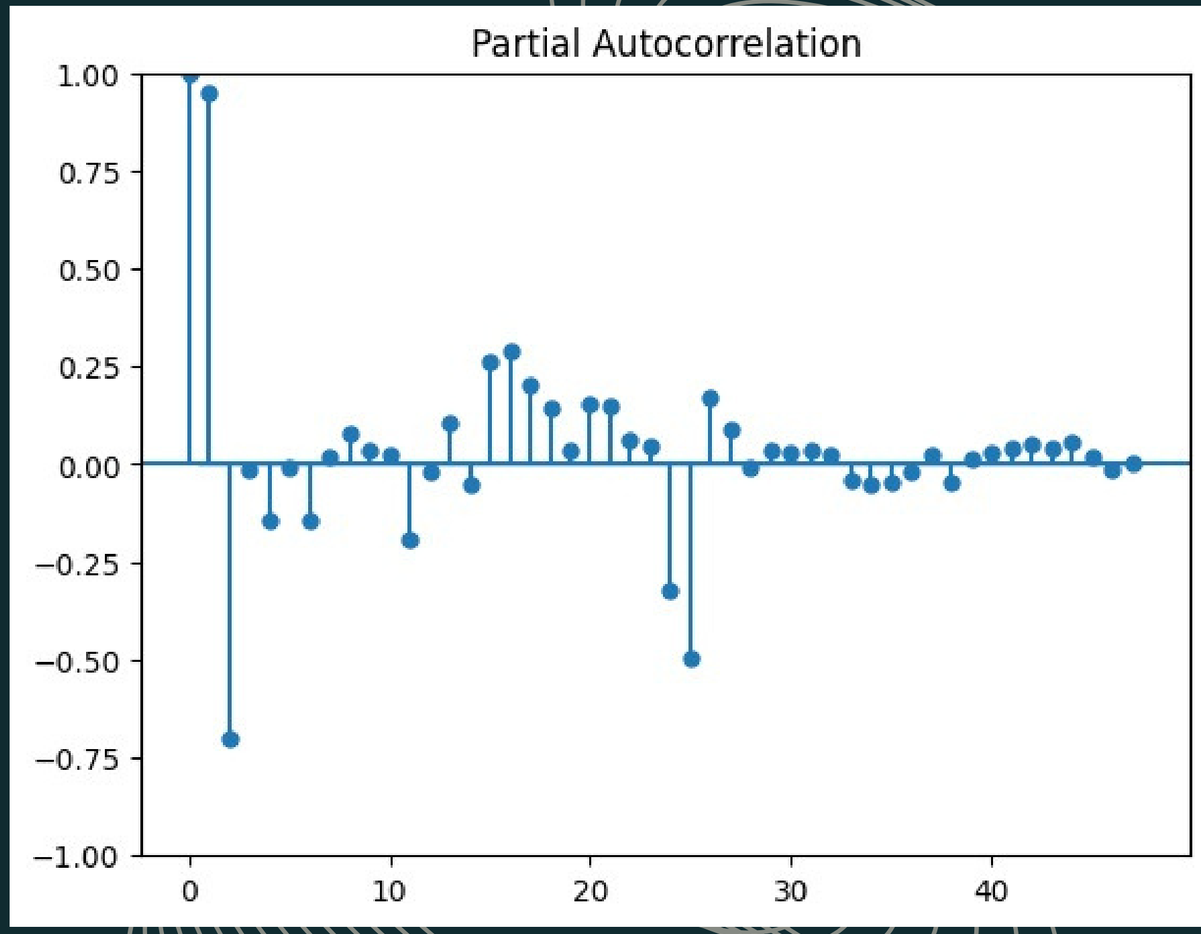
- For AR n MA

Once the series is stabilized, we can plot the ACF and PACF(Partial auto-correlation Function) plots to identify the orders of AR and MA terms in the ARMA model. Though ACF and PACF do not directly dictate the order of the ARMA model, the plots can facilitate understanding the order and provide an idea of which model can be a good fit for the time-series data.

I(d) in ARIMA:



PACF in ARIMA:



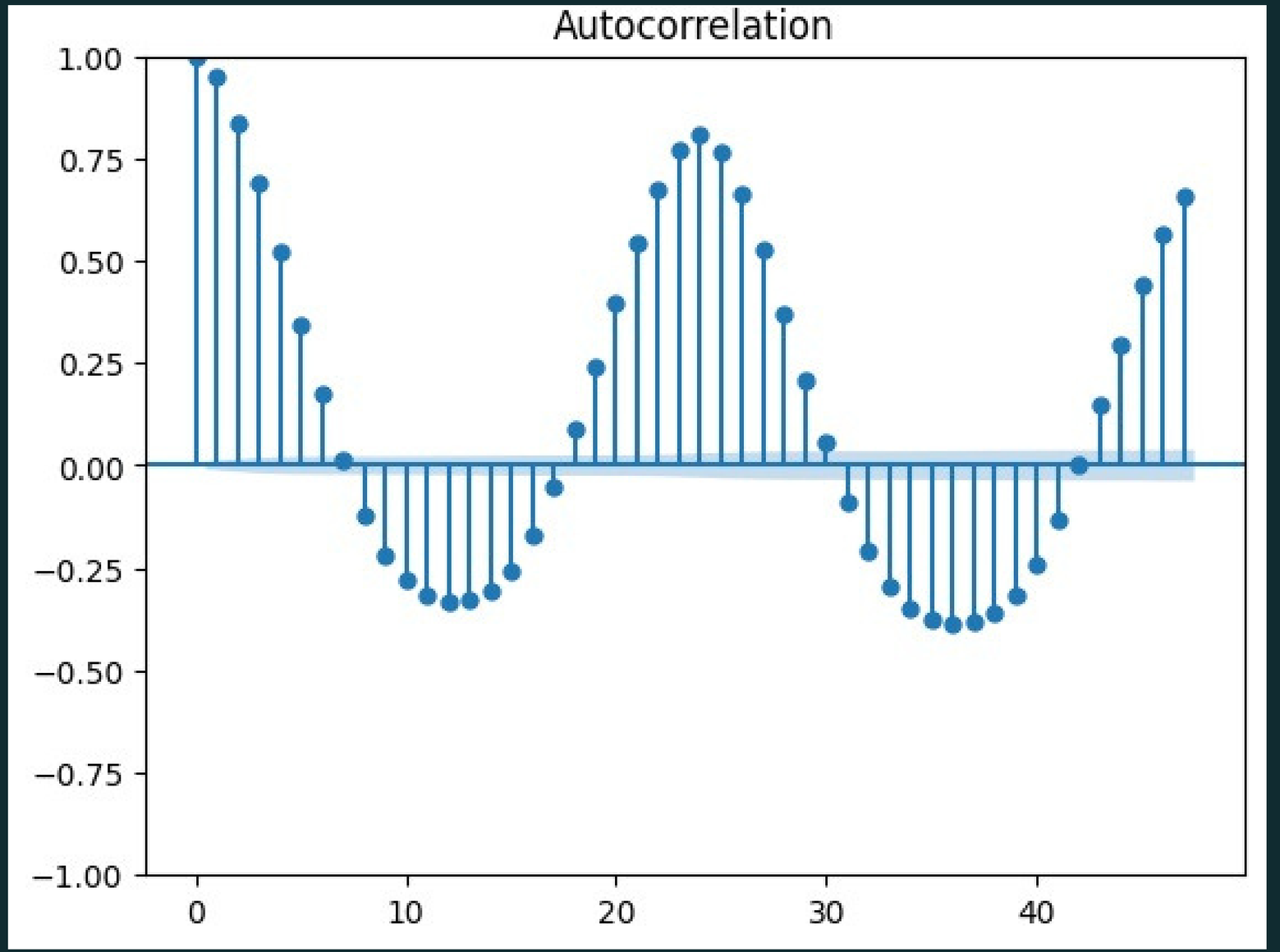
ACF:

- bar chart of coefficients of correlation between a time series and it lagged values
- ACF explains how the present value of a given time series is correlated with the past (1-unit past, 2-unit past, ..., n-unit past) values.
- the y-axis expresses the correlation coefficient whereas the x-axis mentions the number of lags
- If y(t-1), y(t), y(t-1),....y(t-n) are values of a time series at time t, t-1,...,t-n, then the lag-1 value is the correlation coefficient between y(t) and y(t-1), lag-2 is the correlation coefficient between y(t) and y(t-2) and so on.

PACF:

- at explains the partial correlation between the series and lags itself.
- PACF can be explained using a linear regression where we predict y(t) from y(t-1), y(t-2), and y(t-3)
- In PACF we correlate the “parts” of y(t) and y(t-3) that are not predicted by y(t-1) and y(t-2).

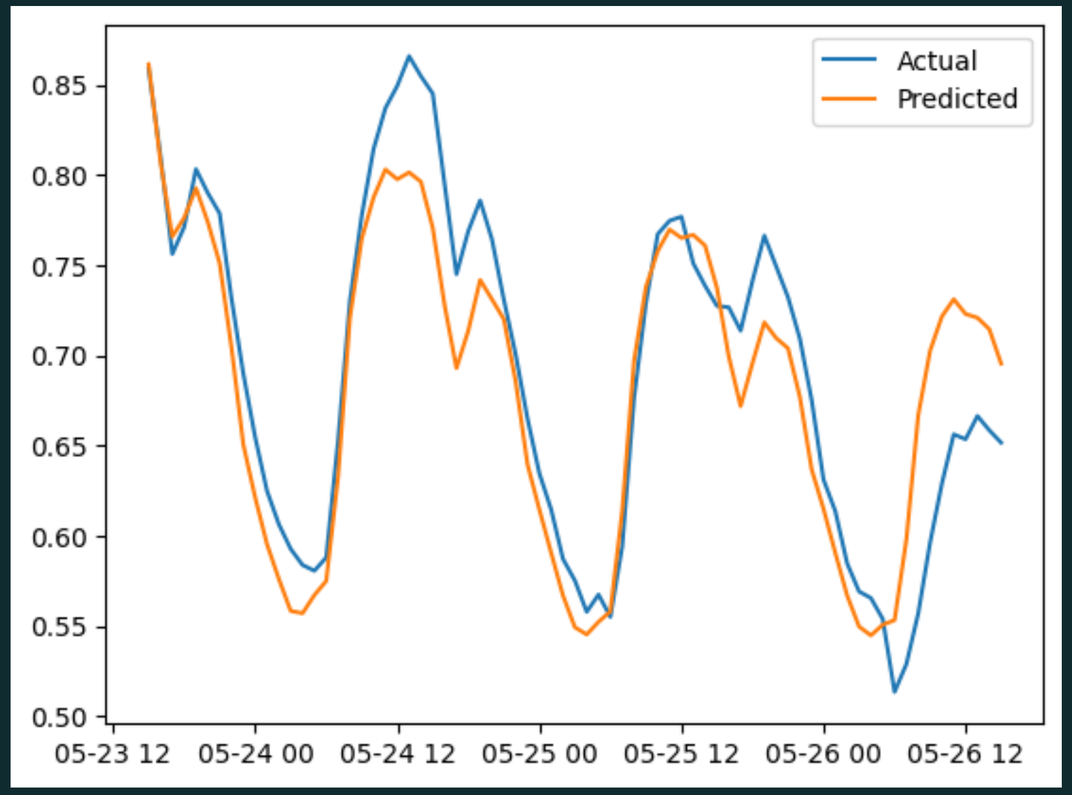
ACF in ARIMA:


Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
10
20
30
40

CODE ARIMA:

```
model = ARIMA(endog=train_y, exog=train_X, order=(24, 0, 24))
model_fit = model.fit()
# Make predictions
predictions = model_fit.predict(start=len(train_y),
                                end=len(train_y)+len(test_y)-1, exog=test_X)
# Plot predicted vs actual values
plt.plot(test_y[0:73], label='Actual')
plt.plot(predictions[0:73], label='Predicted')
plt.legend()
plt.show()
```

OUTPUT ARIMA:

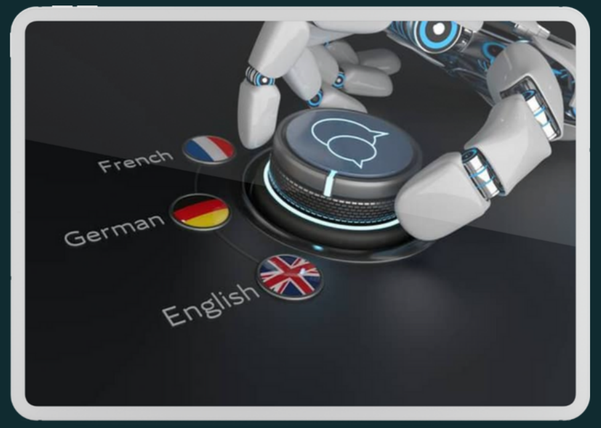


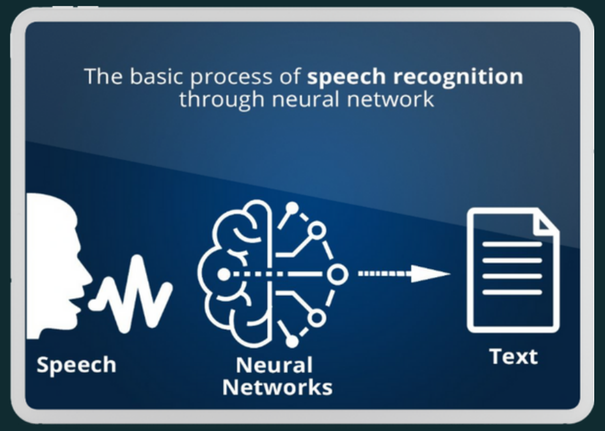
RNN (Recurrent Neural Network)

APPLICATIONS

Most of the advanced Machine Learning Features use RNN

- Autocomplete feature of Google
- Language Translation





- Speech Recognition
- Music Generation

- Hand-Writing Recognition
- Captioning Images

RNN (Recurrent Neural Network)

HOW ARE THEY DIFFERENT FROM SIMPLE NEURAL NETWORK

Simple Neural Network	Recurrent Neural Netwoks (RNN)
Feedforward Architecture, Information flows in one direction only.	Recurrent Architecture, Information retention from previous data values.
Used for tasks that involve static, non-sequential data.	RNNs are designed to handle sequential or time-dependent data.
Lack the ability to retain information about past inputs or context, making them less suitable for tasks that involve sequential data.	RNNs can operate on sequences of varying lengths and are well-suited for tasks such as natural language processing, speech recognition, machine translation, and time series analysis.

LSTM as a Modification of RNN

L I M I T A T I O N S O F R N N

Problem of Vanishing Gradients

- Vanishing gradients refers to a problem that can occur during the training of recurrent neural networks (RNNs),
- The gradients used to update the network's parameters become extremely small over time.
- This phenomenon can hinder the RNN's ability to capture long-term data points in sequential data.

Example

Mr. Vedansh likes to eat Dosa and idli hence his favourite cuisine is South Indian

Here the RNN would easily predict the Cuisine as the weights of Dosa and Idli are still prominent.

Mr. Vedansh likes to eat Dosa and idli though, giving health a priority he has started cycling and is consuming more fruits and vegetables, His favourite cuisine is ??

Here the RNN would not be able to predict the cuisine as the weights of Dosa and idli have vanished over time.

This is where LSTM comes in. It can store data for longer durations and even decide the data point that should be stored and the ones that should be discarded

LSTM(Long short-term memory):

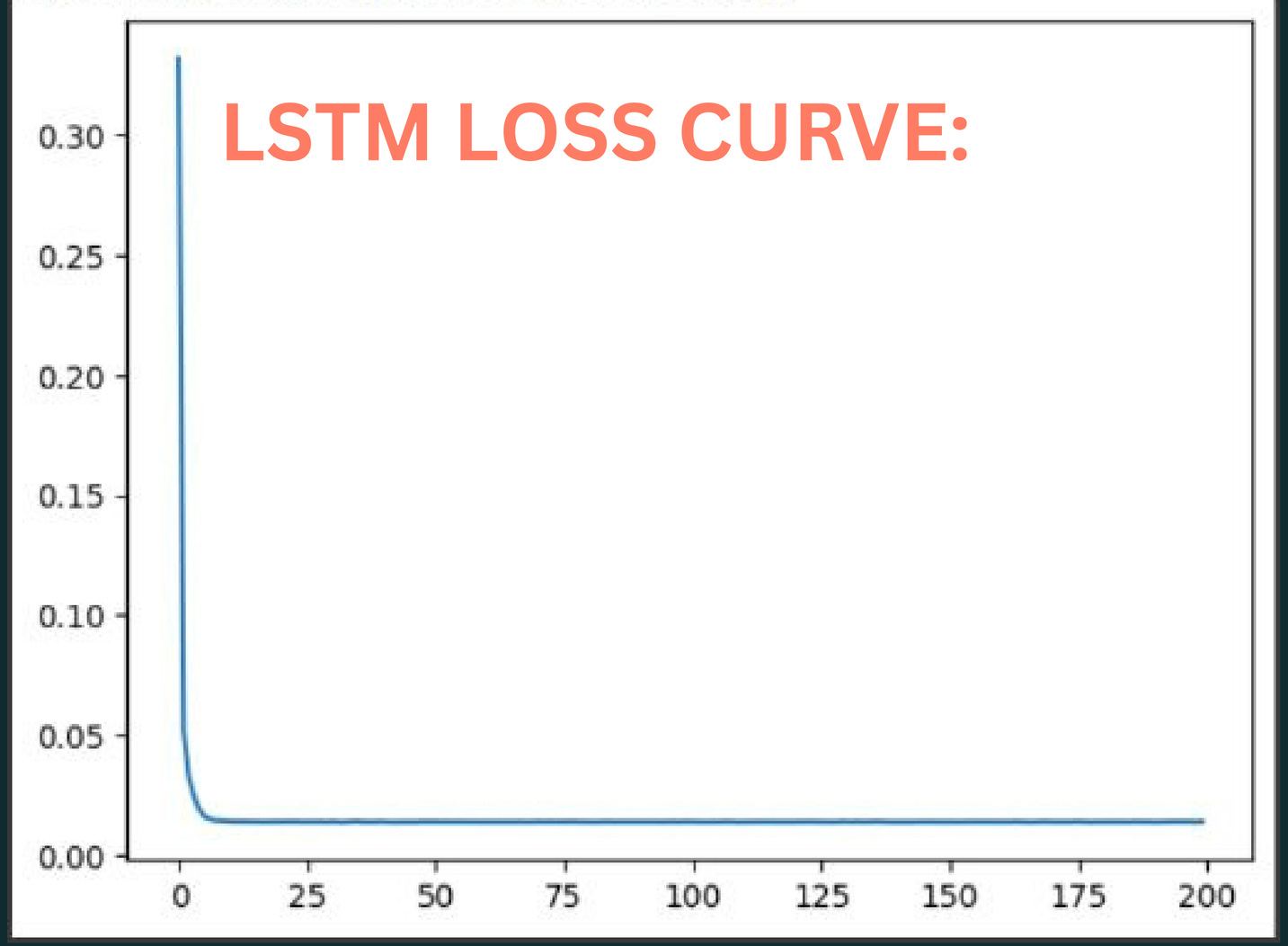
- LSTM is a type of recurrent neural network that is designed to handle long-term dependencies in sequential data.
- It uses memory cells and gates to selectively forget or remember information from the past, allowing it to capture long-term patterns.
- The architecture includes an input gate, a forget gate, an output gate, and a memory cell.
- LSTM is widely used in tasks such as speech recognition, natural language processing, and time series prediction.
- While powerful, LSTM can be computationally expensive and requires a large amount of training data to perform well.
- There are many variations of LSTM, including variants with peephole connections, stacked LSTM layers, and bidirectional LSTM

LSTM CODE:

+ Code + Text

```
for epoch in tqdm(range(num_epochs)):
    epoch_loss = 0.0

    model.train()

    for idx, batch in enumerate((train_dataloader)):
        batch_loss = 0.0
        optimizer.zero_grad()
        if(idx == 86):
          h0 = torch.randn(16, 2, 1)
          c0 = torch.randn(16, 2, 1)
        else:
          h0 = torch.randn(16, batch_size, 1)
          c0 = torch.randn(16, batch_size, 1)
        # h0 = torch.randn(8, batch_size, 1)
        # c0 = torch.randn(8, batch_size, 1)
        batch_output, (hn, cn) = model(batch[0].permute(1, 0, 2).float(), (h0, c0))
        batch_output = batch_output.float()
        batch_target = batch[1].float()

        batch_loss = loss_fn(batch_output.squeeze(), batch_target.permute(1, 0, 2).squeeze())
        epoch_loss += batch_loss.item()
        batch_loss.backward()
        optimizer.step()
    epoch_loss /= 87
    losslis.append(epoch_loss)
    print(f"Epoch {epoch+1}: Loss = {epoch_loss:.6f}")

    # scheduler.step(epoch_loss)
```

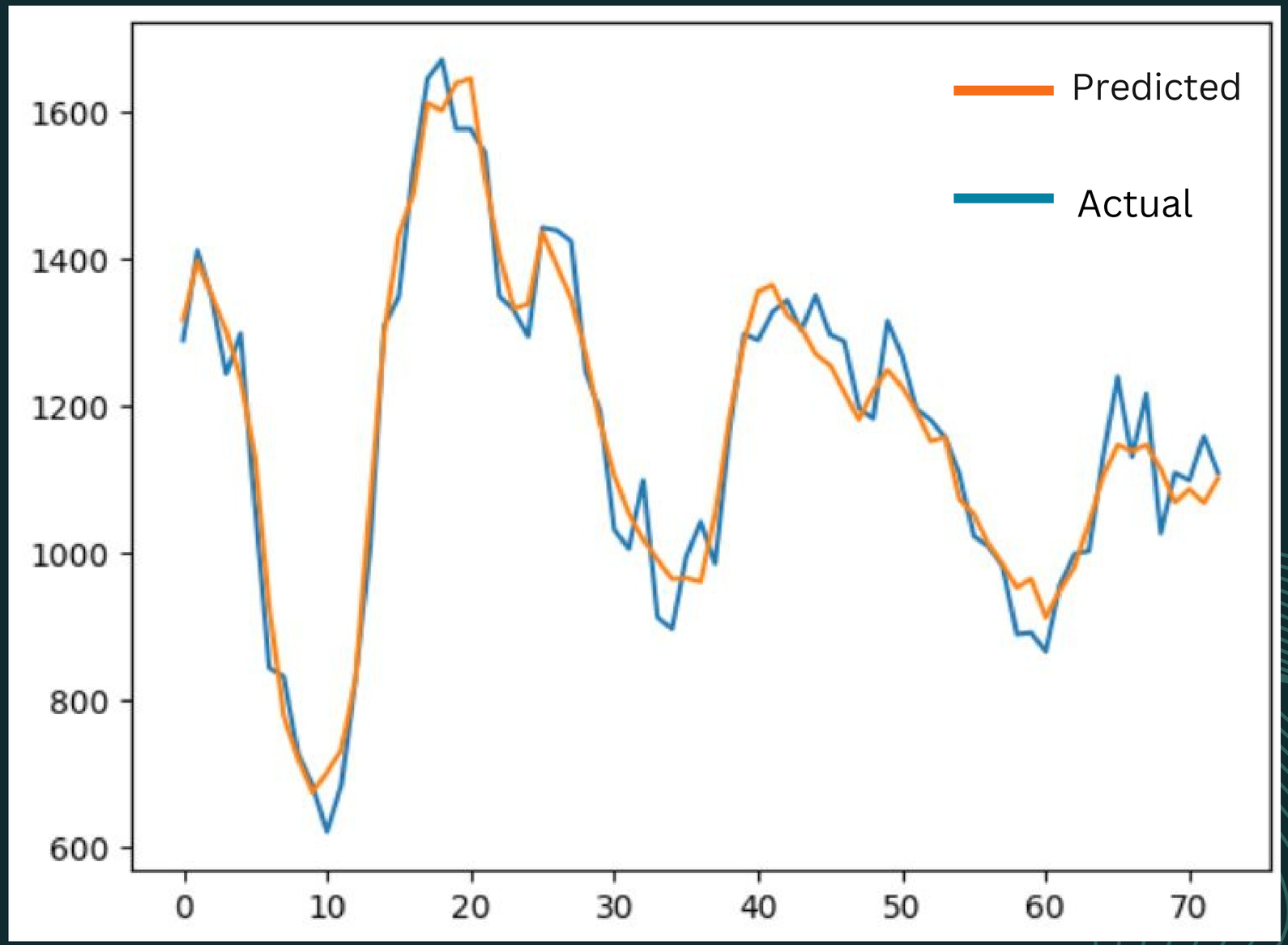
LSTM LOSS CURVE:



The above curve shows that the first iteration gives a larger value of losses but they drastically reduces with subsequent iterations.

```
100%
Epoch 1: Loss = 0.332067
Epoch 2: Loss = 0.051387
Epoch 3: Loss = 0.032670
Epoch 4: Loss = 0.024561
Epoch 5: Loss = 0.019087
Epoch 6: Loss = 0.015993
Epoch 7: Loss = 0.014923
Epoch 8: Loss = 0.014496
Epoch 9: Loss = 0.014295
Epoch 10: Loss = 0.014237
Epoch 11: Loss = 0.013981
Epoch 12: Loss = 0.013906
Epoch 13: Loss = 0.013921
Epoch 14: Loss = 0.013950
Epoch 15: Loss = 0.013846
Epoch 16: Loss = 0.013825
Epoch 17: Loss = 0.013832
Epoch 18: Loss = 0.013827
Epoch 19: Loss = 0.013811
Epoch 20: Loss = 0.013958
```

LSTM OUTPUT:


Predicted
Actual
1600
1400
1200
1000
800
600
0
10
20
30
40
50
60
70

COMPARISON

MODELS	MEAN ABSOLUTE ERROR	ACCURACY
MULTIPLE LINEAR REGRESSION	0.07	89.22
RANDOM FOREST REGRESSION	0.06	90.92
XGBOOST REGRESSION	0.07	89.24
ARIMA TIME-SERIES	0.08	88.12
LSTM TIME-SERIES	0.04	93.24


THANK YOU !!!