**PROJECT REPORT**

**PS04CAST53**

**ON**

**RAINFALL PREDICTION ON ANAND CITY**

**PROJECT GUIDE**

**MR. AGNIVA DAS**

**DEPARTMENT OF STATISTICS**

**SARDAR PATEL UNIVERSITY**

**VALLABH VIDYANAGAR, ANAND-388120**

**2021-2023**

**SUBMITTED BY**

**BHAVESH PATIL**

**PARESH BADGUJAR**

**MSC APPLIED STATISTICS (SEMESTER IV)**

# CERTIFICATE

This is to certify that Mr. Bhavesh Bhagwan Patil, Exam No. 06 , "Master of Science in Applied Statistics", Semester-IV has successfully completed his project entitled "Rainfall Prediction using Machine Learning and Statistical Techniques." for PS04CAST53 in term 2021-2023

Date: /05/2023

Project Guide Head of Department

(Mr. Agniva Das) (Dr. Jyoti M. Divecha)

# CERTIFICATE

This is to certify that Mr. Paresh Suresh Badgujar, Exam No. 11, "Master of Science in Applied Statistics", Semester-IV has successfully completed his project entitled " Rainfall Prediction using Machine Learning and Statistical Techniques." for PS04CAST53 in term 2021-2023

Date: /05/2023

Project Guide Head of Department

(Mr. Agniva Das) (Dr. Jyoti M. Divecha)

# ACKNOWLEDGEMENT

I am really glad for this chance to convey my appreciation to everyone who helped me finish my final year project successfully.

First and foremost, I would want to express my sincere thanks to Mr. Agniva Das Sir, who served as my project guide, for their essential advice, ongoing support, and insightful recommendations. Their expertise and knowledge have helped me navigate through the project, and I could not have done it without them.

I would also want to thank the Department Head, Prof. Divecha Mam for their continual encouragement and support during my academic path. Their steady leadership has been an inspiration to me.

# PROJECT DOMAIN: RAINFALL PREDICTION



# ABSTRACT

Rainfall prediction is a beneficiary one, but it is a challenging task. Machine learning techniques can use computational methods and predict rainfall by retrieving and integrating the hidden knowledge from the linear and non-linear patterns of past weather data.. Predicting the amount of daily rainfall improves agricultural productivity and secures food and water supply to keep citizens healthy. The main objective of this study is to predict the intensity of daily rainfall using machine learning techniques. The dataset was collected from the Meteorological Centre Ahmedabad . Anaconda framework is used, and the coding language used is Python, which is portable and dynamic. Numpy, matplotlib, seaborn, and pandas are the libraries used for the implementation. We used various time series models like ARIMA , SARIMA , Double Exponential Smoothing , Triple Exponential Smoothing , also we used Neural Network models like LSTM and GRU model .

# INDEX

|  |  |  |
| --- | --- | --- |
| **Serial No.** | **Topic** | **Page No.** |
|  | Introduction | 8 |
|  | Objectives | 9 |
|  | Literature Review | 10 |
|  | Data | 11 |
|  | Methodology | 12-16 |
|  | ARIMA Model | 17 |
|  | SARIMA Model | 18 |
|  | Exponential Smoothing | 19 |
|  | Double Exponential Smoothing | 20 |
|  | Triple Exponential Smoothing | 21-22 |
|  | LSTM (Long-Short Term Memory) | 23-29 |
|  | GRU (Gated Recurrent Unit) | 30-33 |
|  | Results and Interpretation | 34-50 |
|  | Conclusion | 51 |
|  | References | 52 |
|  | Appendix | 53 |

# INTRODUCTION

Rainfall forecasting is very important because heavy and irregular rainfall can have many impacts like destruction of crops and farms, damage of property so a better forecasting model is essential for an early. warning ay. This prediction mainly helps farmers and also water resources can be utilized efficiently so that can minimize risks to life and property and also managing the agricultural farms in better way. Rainfall prediction is a challenging task and the results should be accurate. By using machine learning techniques we can produce accurate results. We can just do it by having the historical data analysis of rainfall and can predict the rainfall for future seasons.

Gujarat is a state located in western India, known for its diverse geography and climatic conditions. The state's location near the Arabian Sea makes it susceptible to both the southwest and northeast monsoons, which contribute significantly to the region's annual rainfall. The monsoon season in Gujarat typically spans from June to September, with the heaviest rainfall occurring in July and August. The state's average annual rainfall is around 900 millimeters, with some regions receiving as much as 3,500 millimeters of rain and others as little as 100 millimeters.

The state can be broadly divided into three regions based on rainfall patterns: the coastal region, the central region, and the western arid region. The coastal region, which includes cities like Surat and Bhavnagar, receives the highest amount of rainfall, averaging around 1500 millimeters annually. The central region, which includes cities like Ahmedabad and Vadodara, receives around 700-800 millimeters of rainfall annually. The western arid region, which includes cities like Kutch and Surendranagar, is the driest region in the state, receiving less than 300 millimeters of rainfall annually.

# OBJECTIVE

1. The main objective of this study is to predict the rainfall.
2. To find the robust and accurate model which predict the intensity of rainfall

# LITERATURE REVIEW

In the paper by Sethupathi et al. [1], to predict the rainfall Linear Regression method is used to discover the relationship between dependent variable and independent variable. It will provide reasonable estimate of rainfall for a given time frame but it does not work well on long-time forecasting. Then Random Forest and Logistic Regression models are used to predict whether tomorrow is rain or not. Both models performing well depending on their techniques and evaluated with great accuracy but the accuracy score for the logistic regression algorithm is slightly more efficient than the random forest algorithm.

In the paper by Bari et al. [2], Department of Civil Engineering, Leading University, Sylhet, Bangladesh we learn about the Non-linear rainfall pattern. Various statistical methods are often useful to predict rainfall. Also we learn model estimation techniques to determine the parameters of ARIMA model and checking model diagnostic using ACF and PACF plots.

In the paper by Liyew et al. [3], In this paper, the rainfall was predicted using a machine learning technique. Three machine learning algorithms such as Multivariate Linear Regression (MLR), Random Forest (RF), and gradient descent XGBoost were analyzed which took input variables having moderately and strongly related environmental variables with rainfall. The better machine learning algorithm was identified and reported based on the performance measure using RMSE and MAE

In the paper by Somvanshi et al. [4], we studied ARIMA and ANN Techniques for the prediction of rainfall. here data is taken with the only two attributes like Rainfall (in mm) and Year. ANN is taken with ANN (4,2,1) and ARIMA with ARIMA (4,0,0). Comparing the RMSE of both models ANN gives the better performance

In the paper by Gomathy et al. [5] , there are 19 attributes for 36 sub-divisions. So for the large dataset feature extraction (Principle Component Analysis) is done for extracting necessary variables. Then techniques used are multiple linear regression, support vector machines to predict the intensity of rainfall. Here SVR (Support Vector Regression) is better than MLR. Because MLR can’t catch the non-linearity in a data set and SVR is more helpful in such circumstances. Then measure Mean Absolute Error ((MAE) for both the models to assess execution of the models. And finally conclude that SVR model gives best expectation.

# DATA

The data collected of City Anand from Meteorological Centre , Ahmedabad is from year 1991-2021 . However our station for study have missing data for long periods . in the datafile we just got only one variable which is Rainfall (in mm) .

**The Datafile used :**



The Abbreviations used :

**YEAR :** Year

**MN**  : Month

**DT** : Date

**RF** : Total Rainfall in mm in 24 hours

# METHODOLOGY

Rainfall Dataset

Data Pre-processing

Handling missing data

Feature Extraction

Prediction Model

Testing Data

Training Data

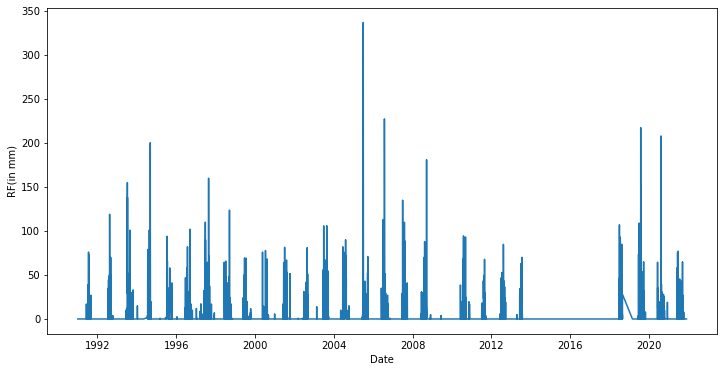
Graphs and Result

Fig.1 Rainfall Prediction Model

PREPROCESSING

1. **Cleaning the data**

Our data contained so much empty values and whitespace characters . to fill those empty values or whitespace characters first we need to fill those empty spaces by replacing with the NaN values .

1. **Missing value imputation**

As we can easily see we have to many missing values from year 2014 to 2018 . we can not fill them directly using imputation methods . we need to make a machine learning model upto year 2013 and based on it we forecast those missing data .

Also there are some missing values in the data upto year 2013 . we can deal with those missing values by some common imputation techniques .

After dealing with the empty spaces , our main critical task is to fill those NaN values using imputation method .

We use linear interpolation to estimate the missing values.

The Linear interpolation formula is given by ,

we estimate the missing values between year 1991 to 2013 .

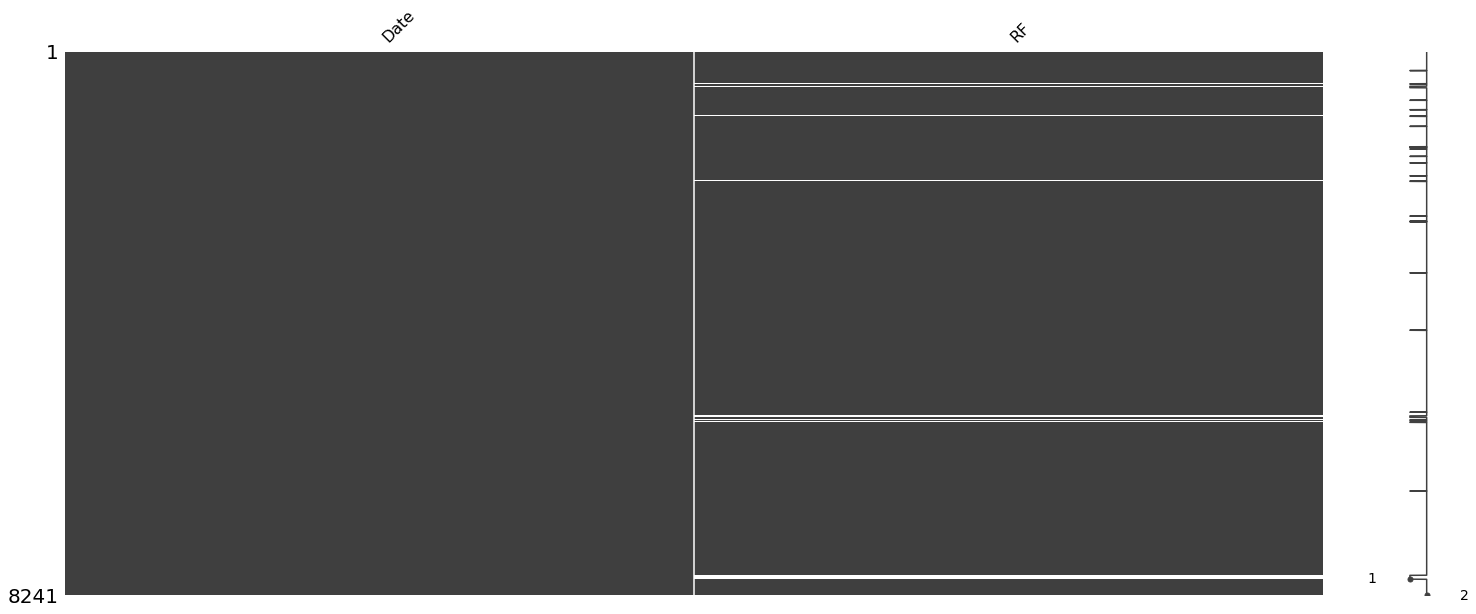
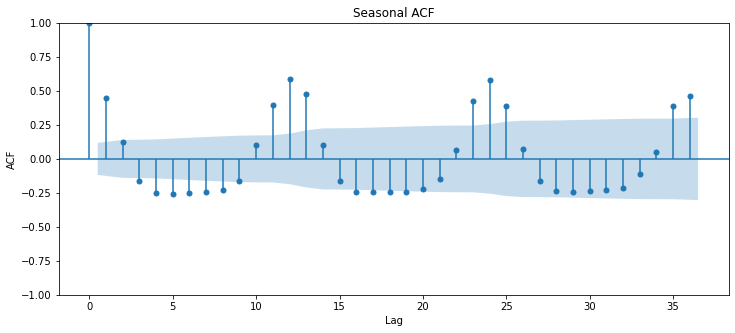
1. **Data Transformation**

Data transformation involves converting raw data into a format that is suitable for analysis or machine learning models**.** Our data contain separate year, month and day column so we convert it into regular date time format . then we give indexing to date column. While working with time series data , it is compulsory to give indexing to date column.

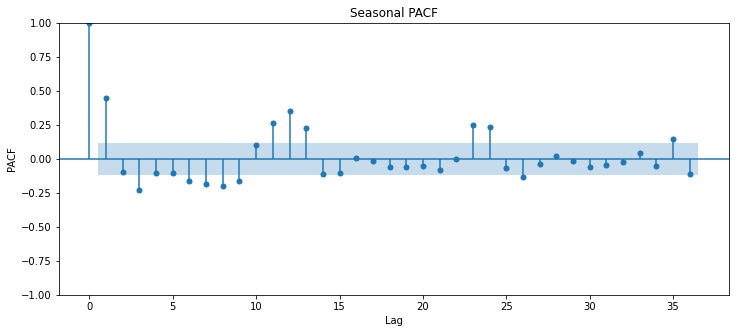
1. **Feature Selection**

Feature selection is an important step in the machine learning pipeline, as it can help improve the accuracy and generalization of models while reducing overfitting and computational complexity. we have 31 irrelavant features containing so many missing values so drop those columns from our dataframe .

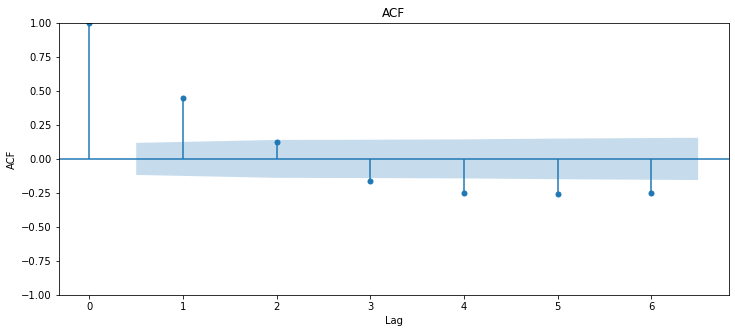
**EDA**

1. **Missing Value Plot**
2. **Auto correlation Plots**
3. ACF (Auto Correlation Plot of Seasonal ARIMA)

The ACF plot is used to identify patterns of autocorrelation in a time series. Specifically, it can be used to determine the order of an autoregressive (AR) model, which is a commonly used time series model.

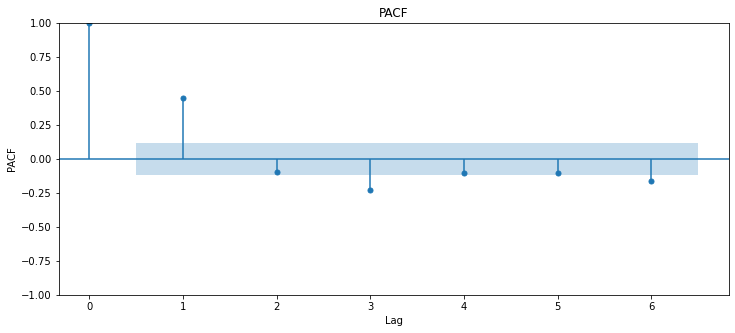
1. PACF (Partial Auto Correlation Plot of Seasonal ARIMA)

The PACF plot (Partial Autocorrelation Function plot) of a seasonal ARIMA model shows the partial autocorrelation coefficients for the seasonal part of the model.

1. ACF (Auto Correlation Plot for ARIMA)

An autocorrelation plot (ACF plot) is a graphical representation of the autocorrelation function of a time series. The autocorrelation function measures the correlation between a time series and its lagged values at different lags.

1. PACF (Partial Auto Correlation Plot for ARIMA )



A partial autocorrelation function plot (PACF plot) is a graphical representation of the partial autocorrelation function of a time series. The partial autocorrelation function measures the correlation between a time series and its lagged values, controlling for the effects of all shorter lags.

# Autoregressive Integrated Moving Average (ARIMA) Process

ARIMA models are an extension of the ARMA model, which combines autoregressive (AR) and moving average (MA) components. The **"I"** in ARIMA stands for "**integrated**," which refers to the differencing of the time series to make it stationary.

An ARIMA model is defined by three parameters: p, d, and q, which correspond to the order of the autoregressive component, the order of differencing, and the order of the moving average component, respectively. The notation for an ARIMA model is ARIMA(p, d, q).

**Auto Regressive (p)**: This parameter represents the order of the autoregressive (AR) component of the model. The AR component models the relationship between the current value of the time series and its past values. A higher value of p indicates a stronger dependence on past values and a more complex model.

**Integrated (d)**: This parameter represents the order of differencing required to make the time series stationary. Differencing involves subtracting each value from its previous value, which can remove trends and seasonality. A higher value of d indicates that more differencing is required to make the series stationary.

**Moving Average (q)**: This parameter represents the order of the moving average (MA) component of the model. The MA component models the relationship between the current value of the time series and past error terms. A higher value of q indicates a stronger dependence on past error terms and a more complex model.

Then we use the ARIMA model with the combinations of AutoRegressive (AR) of order , Integrated (I) of order d and Moving Average of order q .

ARIMA(p,d,q) = AR(p) + I(d) + MA(q)

=

# Seasonal Autoregressive Integrated Moving Average (SARIMA) Process

SARIMA is an extension of the ARIMA model that includes a seasonal component. It is also known as the Box-Jenkins seasonal model. SARIMA models are commonly used in analyzing time series data that exhibit seasonality or periodicity, such as monthly or quarterly data.

The notation for a SARIMA model is SARIMA(p, d, q)(P, D, Q)s, where p, d, and q are the non-seasonal ARIMA model parameters, P, D, and Q are the seasonal ARIMA model parameters, and s is the frequency of the seasonality (i.e., the number of observations in a season).

**Seasonal autoregressive (SAR) component:** This component models the seasonal dependence between the current observation and a specified number of past observations at regular seasonal intervals. The order of this component is denoted by "**P**".

**Seasonal integrated (SI) component:** This component models the seasonal differencing of the time series to make it stationary. The order of seasonal differencing is denoted by "**D**". If the seasonal pattern is not present in the time series, then D=0.

**Seasonal moving average (SMA) component:** This component models the seasonal dependence between the current observation and a specified number of past error terms at regular seasonal intervals. The order of this component is denoted by "**Q**".

# EXPONENTIAL SMOOTHING

What Is Exponential Smoothing?

Exponential smoothing is a time series forecasting method for univariate data. Time series methods like the Box-Jenkins ARIMA family of methods develop a model where the prediction is a weighted linear sum of recent past observations or lags. Exponential smoothing forecasting methods are similar in that a prediction is a weighted sum of past observations, but the model explicitly uses an exponentially decreasing weight for past observations. Specifically, past observations are weighted with a geometrically decreasing ratio. Forecasts produced using exponential smoothing methods are weighted averages of past observations, with the weights decaying exponentially as the observations get older. In other words, the more recent the observation the higher the associated weight.

# DOUBLE EXPONENTIAL SMOOTHING

Double exponential smoothing is sometimes called exponential smoothing with trend. If trend exists, single exponential smoothing may need adjustment. There is a need to add a second smoothing constant to account for trend.

Model =

)

= actual value in time t

α = constant-process smoothing constant

β = trend-smoothing constant

= smoothed constant-process value for period t

= smoothed trend value for period t

t = current time period

A smoothing approach for forecasting such a time series that employs two smoothing constants denoted by and there are two estimates and is the estimate of the level of the time Series constructed in time period t-1 (This is usually called the permanent component).

is the estimate of the growth rate of the Time series constructed in time period t-1 (This is usually called the trend component).

# TRIPLE EXPONENTIAL SMOOTHING

What happens if the data show trend and seasonality? To handle seasonality, we have to add a third parameter

In this case double smoothing will not work. We now introduce a third equation to take care of The resulting set of equation is called the “holt-Winters”(HW) method after the name of the inventors.

The forecasting equation for made after the observation at time t becomes

=(

The Holt-Winters method generalizes Equations and the additive seasonal form of their updating equations for a series {} with period m is

)

The Holt-Winters algorithm with multiplicative seasonal is

=+(1-(

+

Where,

*Y* is the observation

*F* is the smoothing observation

*b* is trend factor

*I* is the seasonal index

*S* is the forecast at m periods ahead

*t* is an index denoting a time period

and ,and are constant that must be estimated in such a way that the MSE of the error is minimized. This is best left to a good software package.

To initialize the HW method we need at least one complete season’s data to determine initial estimates of the seasonal indices

A complete season’s data consists of L periods. And we need to estimate the trend factor from one period to the next. To accomplish this, it is advisable to use two complete seasons; that is, 2L periods.

# LSTM MODEL

Long Short-Term Memory networks, sometimes known as "LSTMs," are a kind of RNN that can learn long-term dependencies. Hochreiter and Schmidhuber (1997) introduced them, and numerous individuals developed and popularized them in subsequent work. They operate extremely effectively on a wide range of issues and are now frequently employed.

An LSTM (Long Short-Term Memory) model is a sort of recurrent neural network (RNN) that can learn long-term relationships in sequential data, such as time series data. The LSTM model's design comprises cells that may selectively store or delete information based on the current input and the prior cell state. This allows the model to recall essential information for lengthy periods of time while avoiding the vanishing gradient problem that is common in standard RNNs

LSTMs are expressly designed to avoid the problem of long-term reliance. Long-term memory is basically their default habit; it is not something they have to work hard to learn!

The usage of LSTM models has been widespread in a number of fields, including time-series forecasting, speech recognition, and natural language processing. An LSTM model may be trained to estimate future values based on historical data in time-series forecasting.

An LSTM model is trained by feeding it a series of inputs, each of which corresponds to a certain time step. The result at each time step is then compared to the predicted output, and the model parameters are changed to minimize the prediction error. This technique is done for numerous epochs until the model converges to a minimal error.

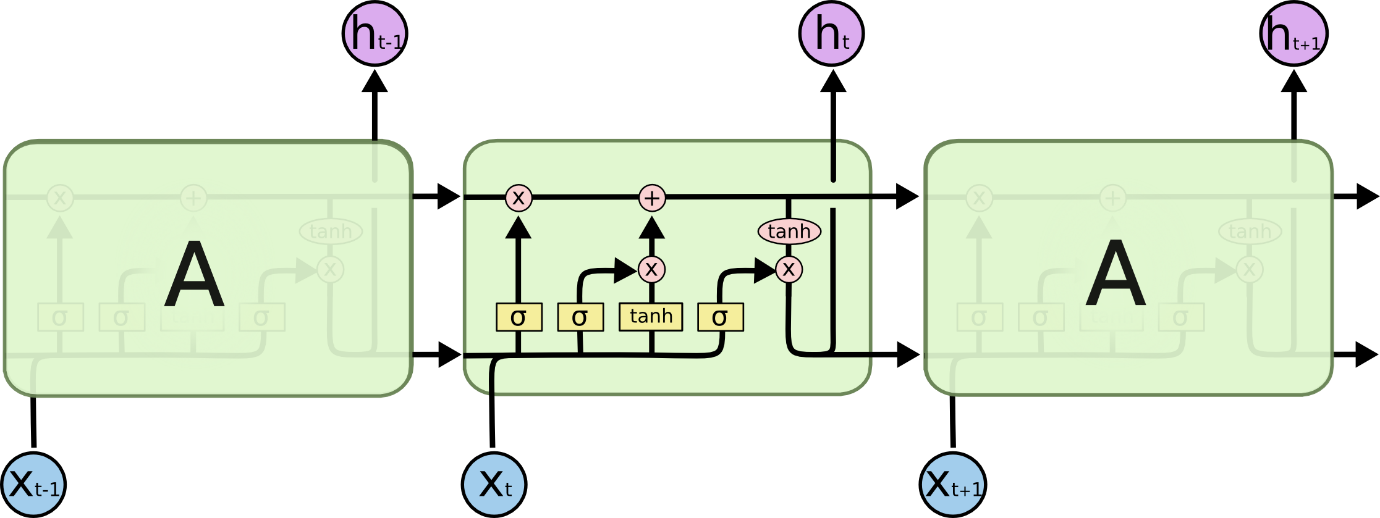
The ability of LSTM models to handle varying sequence lengths and input data types, such as categorical or numerical data, is one of their advantages. They can also detect non-linear correlations in data, making them valuable in a variety of applications.

LSTM models, in summary, are a strong sort of recurrent neural network that may be utilized for time-series forecasting and other sequential data applications. Their capacity to deal with long-term dependencies and non-linear interactions qualifies them for many real-world issues.

Natural language processing, audio recognition, picture captioning, and time-series forecasting have all been effective uses of LSTM networks. They are especially beneficial for activities with long-term dependencies and non-linear data interactions.

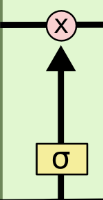
In conclusion, LSTM networks are a strong sort of recurrent neural network capable of modelling sequential data and capturing long-term relationships. They are well-suited for many real-world challenges due to their unique memory cells and gating mechanisms, which allow them to selectively store and retrieve information over.

**Architecture of LSTM Model**

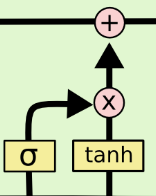


1. **Forget Gate:** The forget gate deletes information that is no longer

helpful in the cell state. Two inputs, x t (at-the-time input) and h t-1 (prior cell output), are supplied into the gate and multiplied using weight matrices before bias is added. The result is sent into an activation function, which produces a binary output. If the output for a certain cell state is 0, the information is lost; if the output is 1, the information is saved for future use.

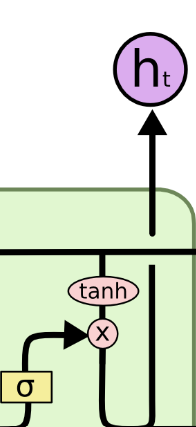


1. **Input gate:** The input gate is responsible for adding important information to the cell state. Initially, the information is controlled using the sigmoid function, and the values to be remembered are filtered using the h t-1 and x t inputs, similar to the forget gate. The tanh function is then used to generate a vector with values ranging from -1 to +1 that encompasses all of the possible values from h t-1 and x t. Finally, the vector and regulated values are multiplied to provide relevant information.



1. **Output gate:** The output gate is in charge of obtaining meaningful

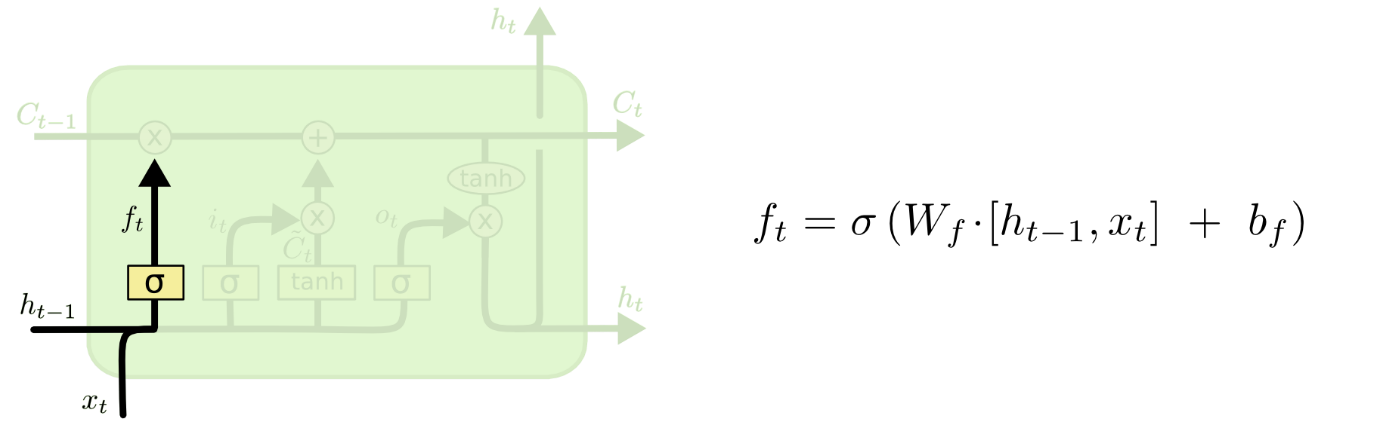
information from the current cell state and presenting it as output. To begin, a vector is created by applying the tanh function on the cell. The information is then controlled using the sigmoid function and filtered by the values to be remembered through h t-1 and x t inputs. Finally, the vector and controlled values are multiplied and provided as output and input to the next cell.



# STEP-BY-STEP LSTM WALK THROUGH

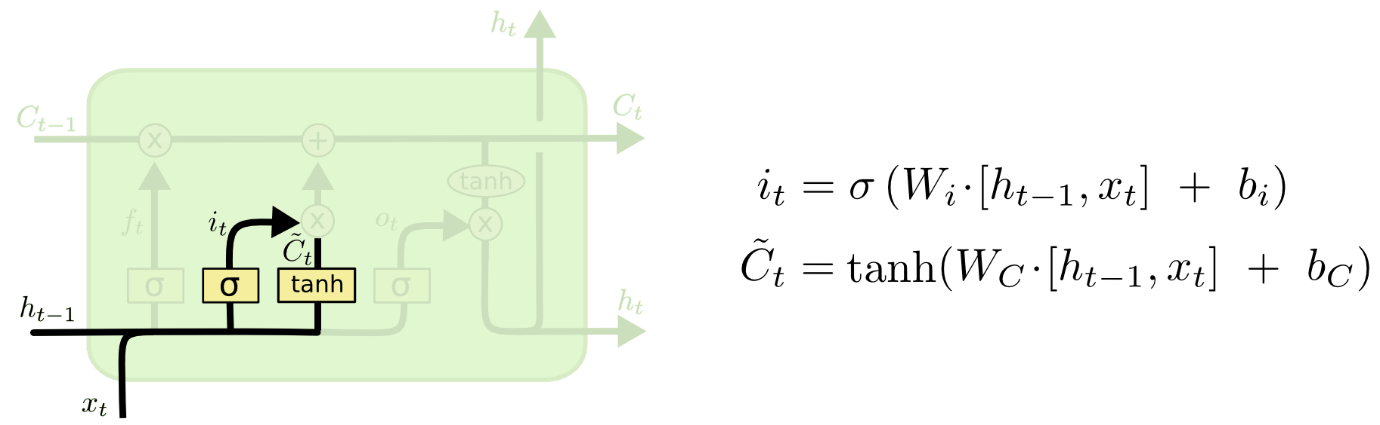
The first step in our LSTM is to decide what information we're going to throw away from the cell state. This decision is made by a sigmoid layer called the "forget gate layer." It looks at and , and outputs a number between 0 and 1 for each number in the cell state . A 1 represents "completely keep this" while a 0 represents "completely get rid of this."

Let's go back to our example of a language model trying to predict the next word based on all the previous ones. In auch a problem, the cell atate might include the gender of the present aubject, ao that the correct pronouns can be used. When we see a new subject, we want to forget the gender of the old subject.



The next step is to decide what new information we're going to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values we'll update. Next, a tanh layer creates a vector of new candidate values, , that could be added to the atate. In the next step, we'll combine these two to create an update to the state.

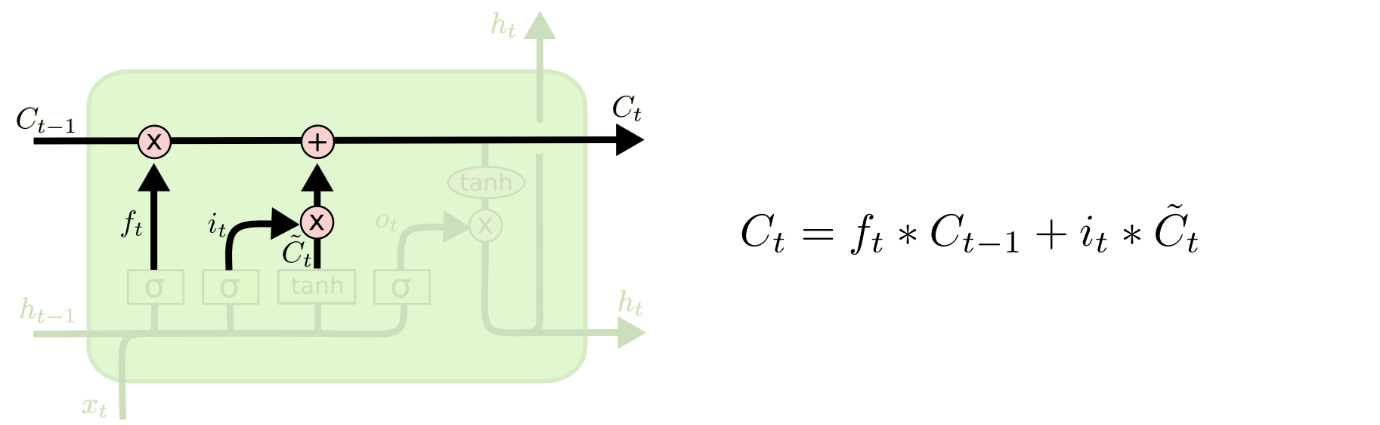
In the example of our language model, we'd want to add the gender of the new subject to the cell state, to replace the old one we're forgetting.



It's now time to update the old cell state, , into the new cell state - The previous ateps already decided what to do, we just need to actually do it.

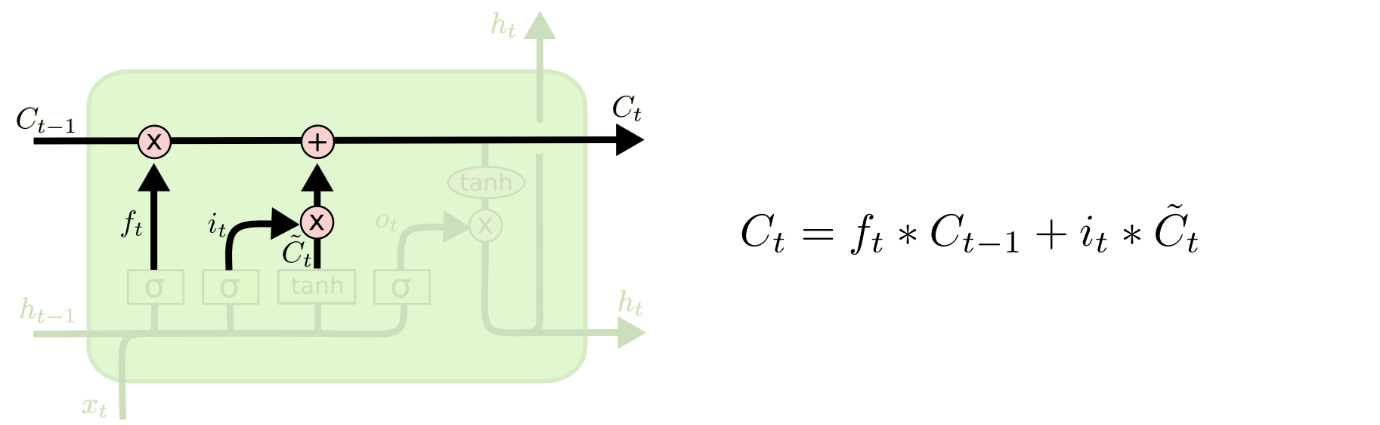
We multiply the old state by , forgetting the things we decided to forget earlier. Then we add . This is the new candidate values, acaled by how much we decided to update each state value.

In the case of the language model, this is where we'd actually drop the information about the old subject's gender and add the new information, as we decided in the previous steps.



Finally, we need to decide what we're going to output. This output will be based on our cell state, but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we're going to output. Then, we put the cell state through tanh (to push the values to be between -1 and 1 ) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

For the language model example, since it just saw a subject, it might want to output information relevant to a verb, in case that's what is coming next. For example, it might output whether the subject is singular or plural, so that we know what form a verb should be conjugated into if that's what follows next.



# GRU MODEL

The Gated Recurrent Unit (GRU) model is a form of Recurrent Neural Network (RNN) architecture that is frequently used for modelling sequential data, such as time series, natural language, and voice.

The GRU model is comparable to the more well-known Long Short-Term Memory (LSTM) model, although it has less parameters and may be trained more quickly. It does this by employing two gates—an update gate and a reset gate—to regulate the information flow through the network.

The update gate decides how much of the previous state to retain and how much of the incoming input to incorporate into the current state, whereas the reset gate decides how much of the prior state to discard. The GRU model can capture complex relationships in incoming data and create predictions based on prior inputs by dynamically updating and resetting its state.

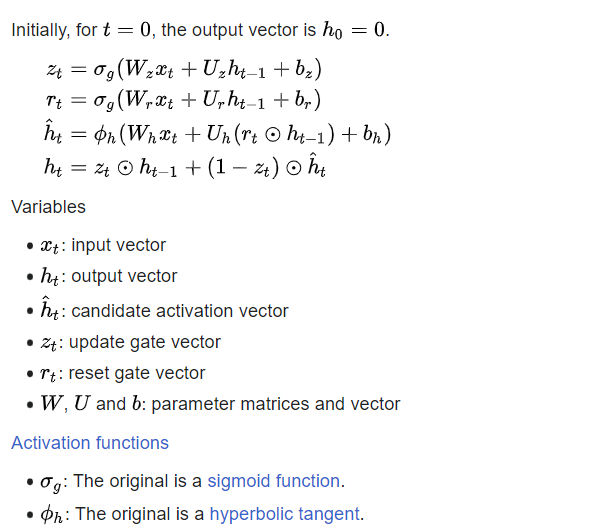
The GRU model has been applied in a wide range of applications, including machine translation, sentiment analysis, and speech recognition. It is a very strong tool for representing sequential data and has made major contributions to developments in natural language processing and related domains.

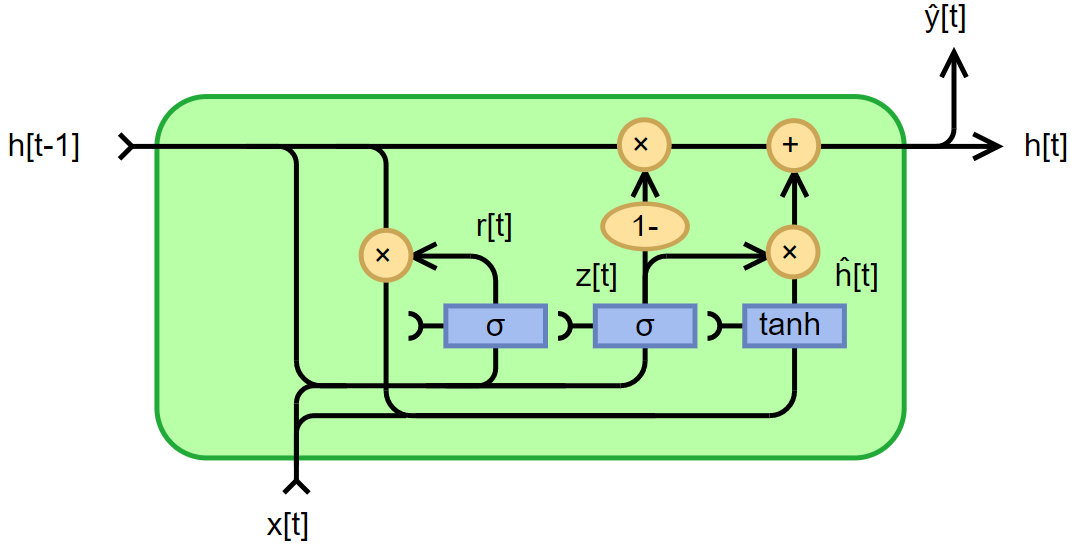
**Architecture**

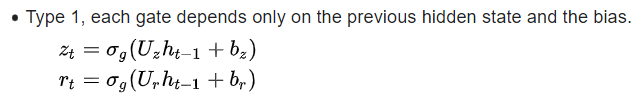
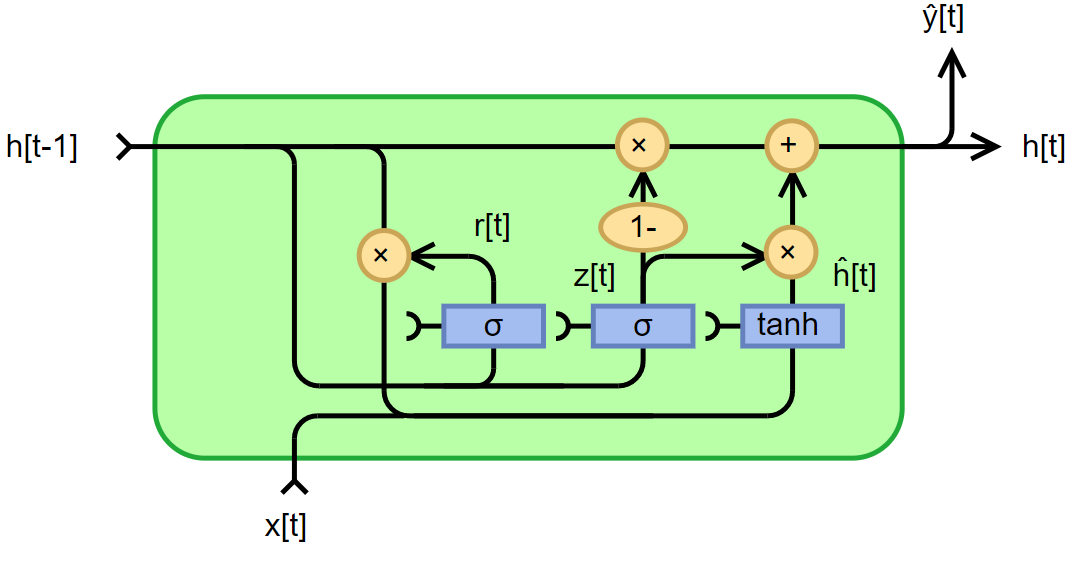
There are several variations on the full gated unit, with gating done using the previous hidden state and the bias in various combinations, and a simplified form called minimal gated unit.

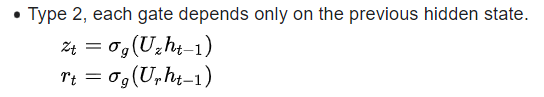
Types of gated units

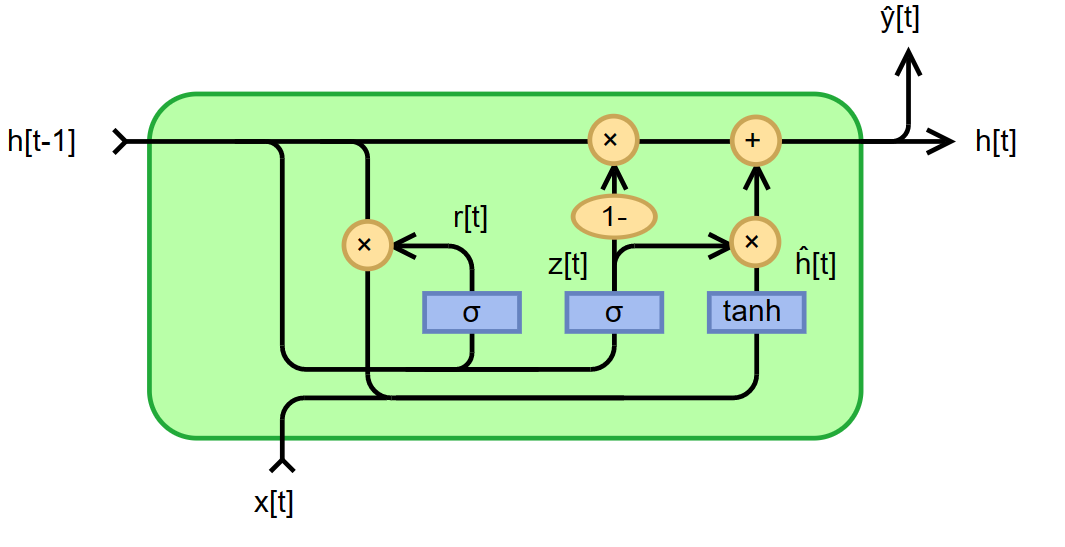
1) full gated units

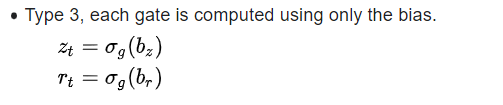
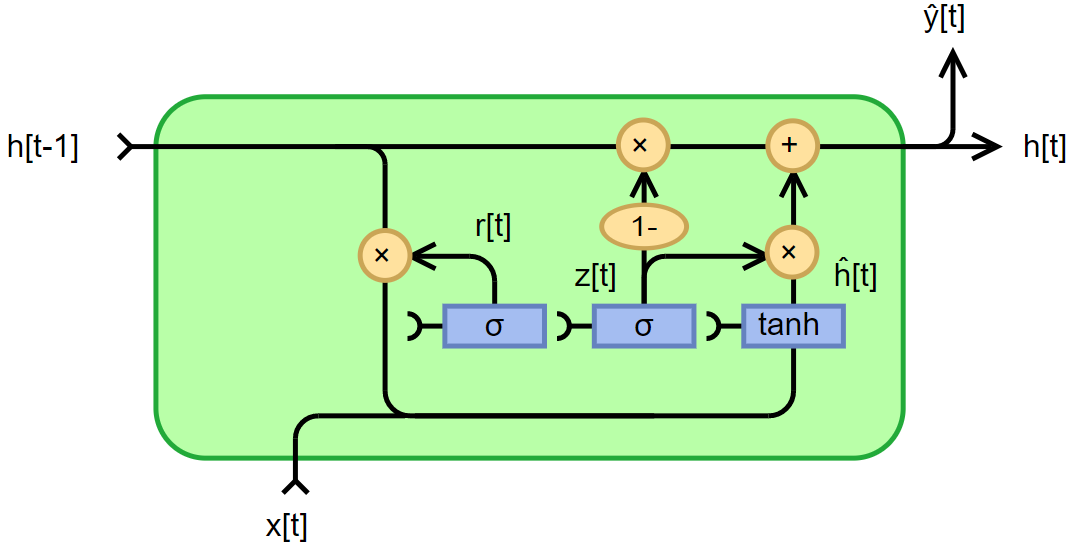












# Results & Interpretation

1. Stationarity Check

We use ADF (Augmented Dickey Fuller Test) to check the Stationarity of the data. The test is conducted by adding the lagged values of the dependent variable

= + +

where is a pure white noise error term and

where = ( − ), = (− ), etc.

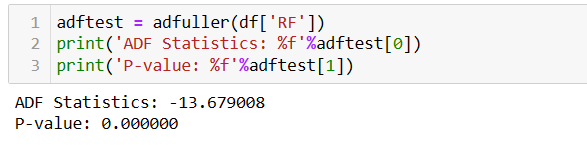
Test the hypothesis:

H0: The time series data has a unit root, indicating it is non-stationary

H1: The time series data does not have a unit root, indicating it is stationary.

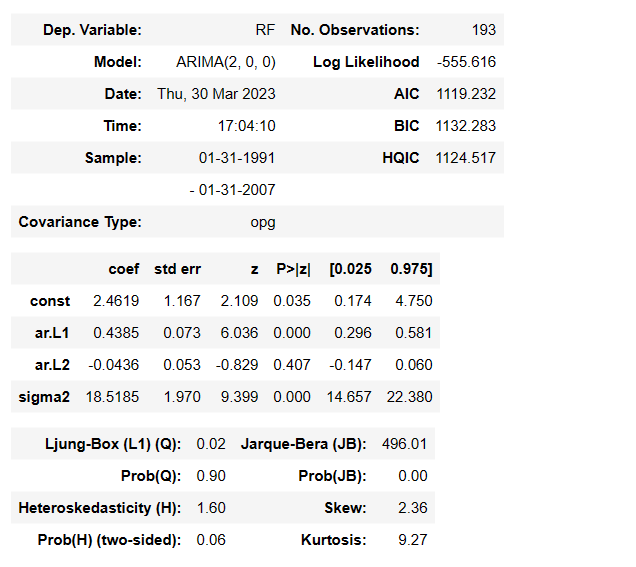
Decision:

If the estimated coefficient is significantly different from zero and the test statistic exceeds a critical value at a chosen level of significance, then the null hypothesis of a unit root (non-stationarity) is rejected in favour of the alternative hypothesis of stationarity.

We test it for our variable of concern i.e **“Rainfall”**

We reject the null hypothesis because our P-value (0.000000) which is very less than the level of significance (0.05). which states that our data is stationary.

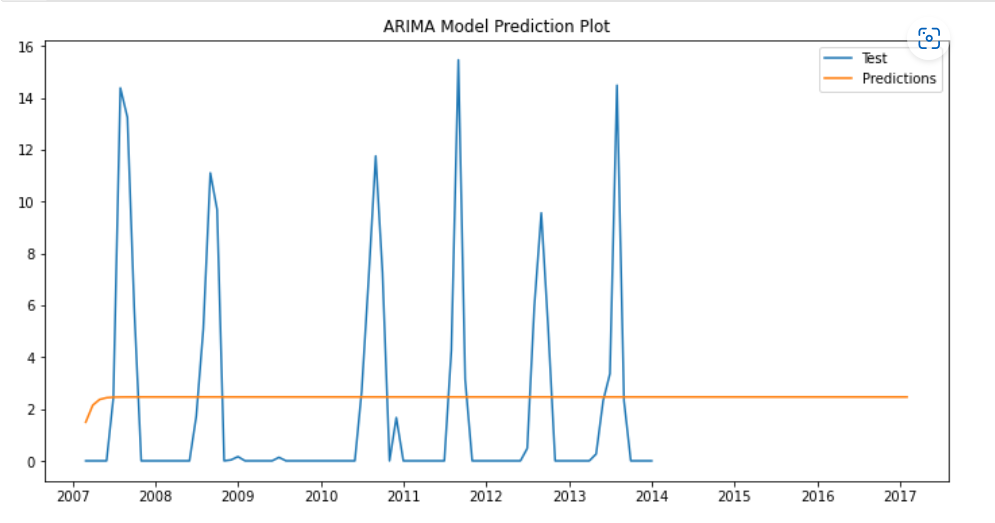
1. Fitting ARIMA model:



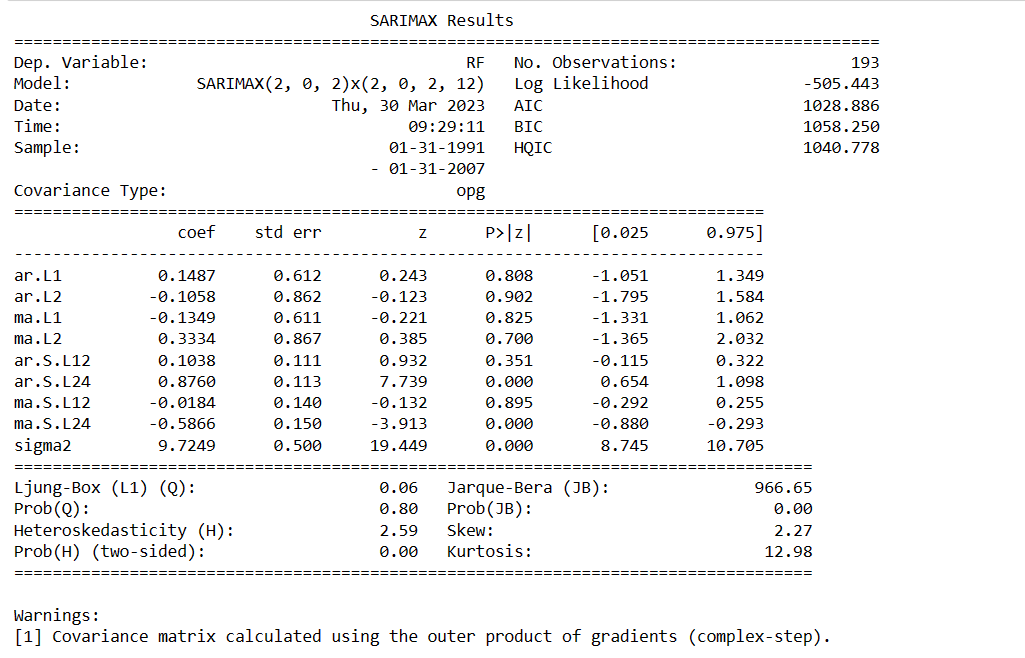
Interpretation:

1. The ARIMA model is,
2. From the model coefficients, we can see only the coefficient of L1 is statistically significant as its p-values is less than the level of significance(0.05).
3. The Ljung-Box (L1) (Q) is the LBQ test statistic at lag 1 is, the Prob(Q) is 0.02, and the p-value is 0.90. Since the probability is above 0.05, we can’t reject the null that the errors are white noise.
4. Heteroscedasticity tests that the error residuals are homoscedastic or have the same variance. The summary performs White’s test. Our summary statistics show a test statistic of 1.60 and a p-value of 0.066, which means we reject the null hypothesis and our residuals show variance.

Visualization:



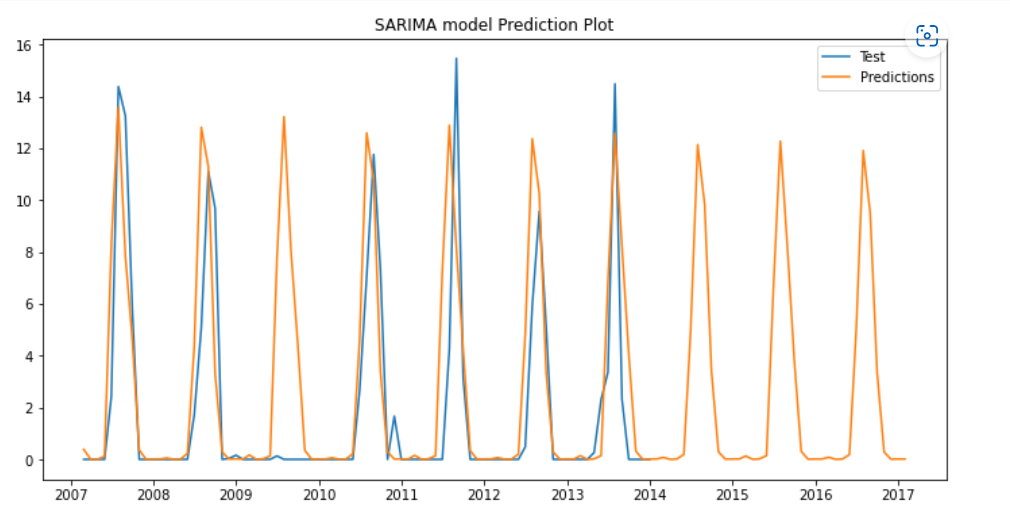
The above plot is of Prediction plot of ARIMA taking **Year** on X-axis and **amount of monthly rainfall** on Y-axis. We take testing data where it conclude that the model is not well trained on the training data and also do not show accurate prediction over a testing data. Also the model forecast the constant values. Its because may be ARIMA model does not capture seasonality.

1. Fitting SARIMA Model

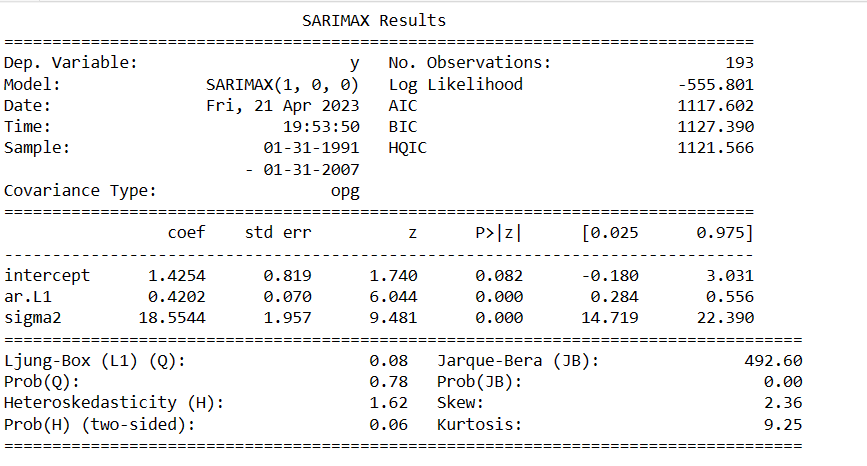
Interpretation:

1. The seasonal ARIMA model we get ,
2. From the model coefficients , we can see only the coefficient of AR L24 and MA L24 are statistically significant as its p-values are less than the level of significance(0.05).
3. The Ljung-Box (L1) (Q) is the LBQ test statistic at lag 1 is, the Prob(Q) is 0.06, and the p-value is 0.80. Since the probability is above 0.05, we can’t reject the null that the errors are white noise.
4. Heteroscedasticity tests that the error residuals are homoscedastic or have the same variance. The summary performs White’s test. Our summary statistics show a test statistic of 2.59 and a p-value of 0.00, which means we reject the null hypothesis and our residuals show variance.

Visualization:



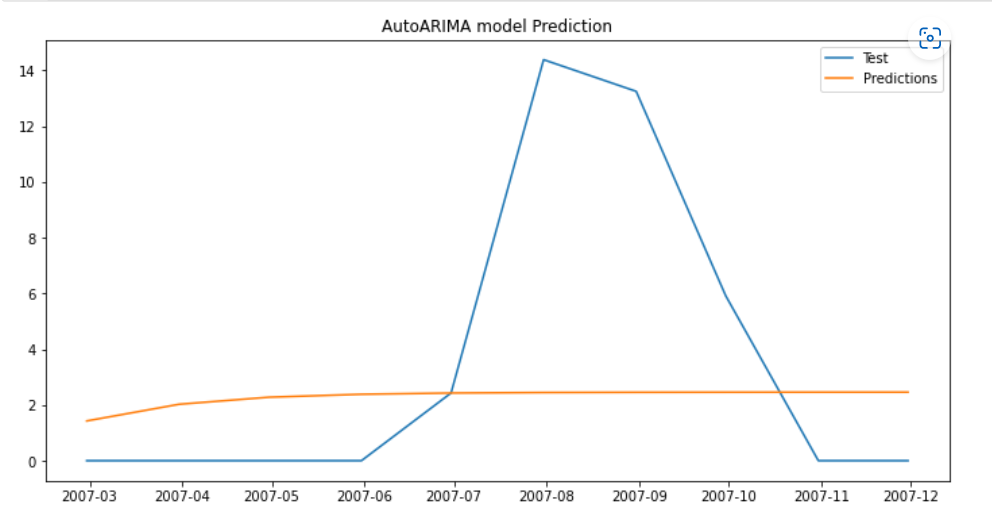
The above plot is of Prediction plot of Seasonal ARIMA taking **Year** on X-axis and **amount of monthly rainfall** on Y-axis. We take training and testing data where it conclude that the model is well trained on the training data and also shows good prediction over a testing data. Also the model forecast the next future values. The SARIMA model captures the seasonality of the data.

1. Fitting Auto ARIMA Model

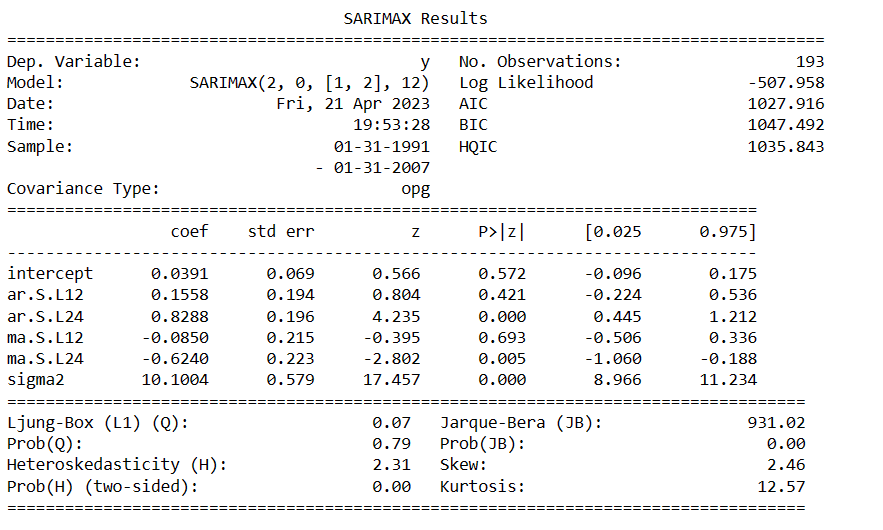
Interpretation:

1. The Auto ARIMA model we get,
2. From the model coefficients, we can see coefficient of AR L1 is statistically significant as its p-values is very less than the level of significance(0.05).
3. The Ljung-Box (L1) (Q) is the LBQ test statistic at lag 1 is, the Prob(Q) is 0.08, and the p-value is 0.78 Since the probability is above 0.05, we can’t reject the null that the errors are white noise.
4. Heteroscedasticity tests that the error residuals are homoscedastic or have the same variance. The summary performs White’s test. Our summary statistics show a test statistic of 1.62 and a p-value of 0.06, which means we reject the null hypothesis and our residuals show variance.

Visualization:



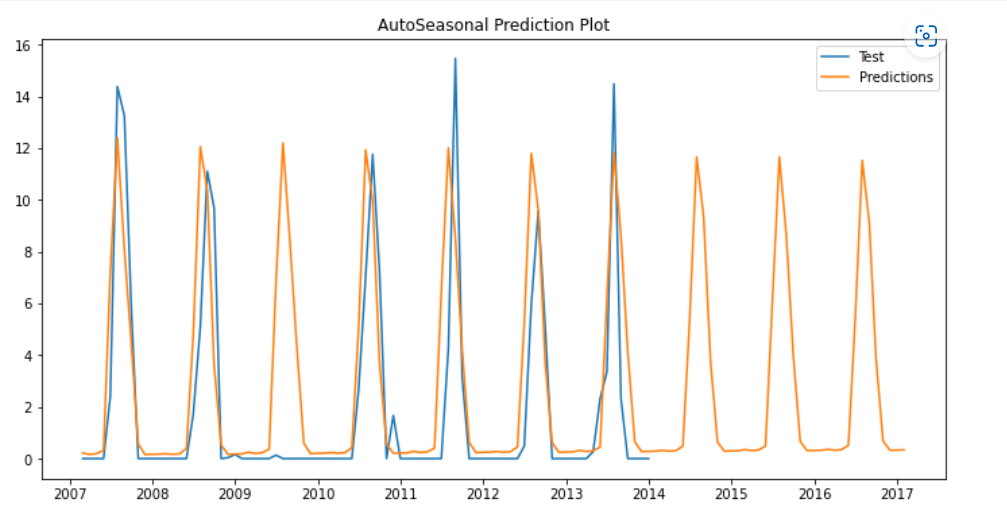
The above plot is of Prediction plot of Auto ARIMA taking **Year** on X-axis and **amount of monthly rainfall** on Y-axis. We take testing data where it conclude that the model is not well trained on the training data and also do not show accurate prediction over a testing data. Also the model forecast the constant values. Its because may be Auto ARIMA model also does not capture seasonality.

1. Fitting Auto Seasonal ARIMA Model

Interpretation:

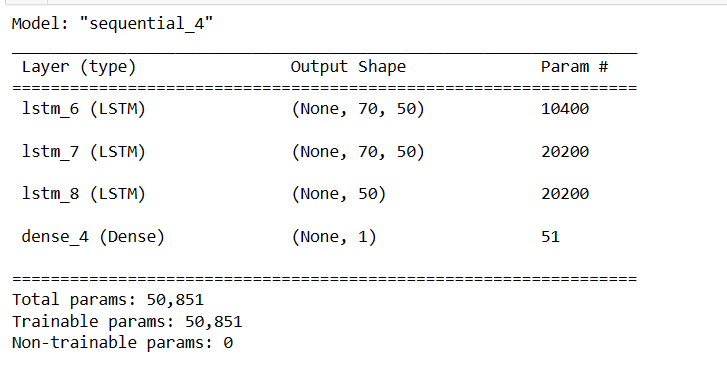
1. The seasonal ARIMA model we get ,
2. From the model coefficients , we can see only the coefficient of AR L24 and MA L24 are statistically significant as its p-values are less than the level of significance(0.05).
3. The Ljung-Box (L1) (Q) is the LBQ test statistic at lag 1 is, the Prob(Q) is 0.07, and the p-value is 0.79. Since the probability is above 0.05, we can’t reject the null that the errors are white noise.
4. Heteroscedasticity tests that the error residuals are homoscedastic or have the same variance. The summary performs White’s test. Our summary statistics show a test statistic of 2.31 and a p-value of 0.00, which means we reject the null hypothesis and our residuals show variance.

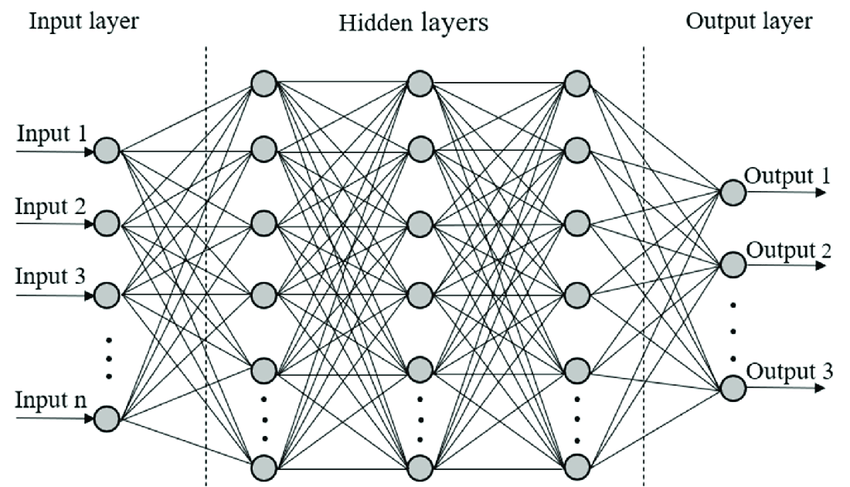
Visualization:



The above plot is of Prediction plot of Seasonal ARIMA taking **Year** on X-axis and **amount of monthly rainfall** on Y-axis. We take training and testing data where it conclude that the model is well trained on the training data and also shows good prediction over a testing data. Also the model forecast the next future values. The SARIMA model captures the seasonality of the data.

1. Fitting LSTM Model



Interpretation:

276 datapoints in the input layer of shape (70,1)

LSTM\_8

LSTM\_7

LSTM\_6

# How the number of trainable parameters for the LSTM model are calculated?

1. **First LSTM layer:**

* Input shape: (None, 70, 1) (70 time steps, 1 feature)
* Output shape: (None, 70, 50) (50 units)
* Number of parameters: 4 \* ((input dim + 1) \* units + units^2) (for input weights, recurrent weights, and biases)
* Substituting the values: 4 \* ((1 + 1) \* 50 + 50^2) = 10,400 parameters

1. **Second LSTM layer**:

* Input shape: (None, 70, 50) (70 time steps, 50 units)
* Output shape: (None, 70, 50) (50 units)
* Number of parameters: 4 \* ((input dim + 1) \* units + units^2) (for input weights, recurrent weights, and biases)
* Substituting the values: 4 \* ((50 + 1) \* 50 + 50^2) = 20,200 parameters

1. **Third LSTM layer:**

* Input shape: (None, 70, 50) (70 time steps, 50 units)
* Output shape: (None, 50) (50 units)
* Number of parameters: 4 \* ((input dim + 1) \* units + units^2) (for input weights, recurrent weights, and biases)
* Substituting the values: 4 \* ((50 + 1) \* 50 + 50^2) = 20,200 parameters

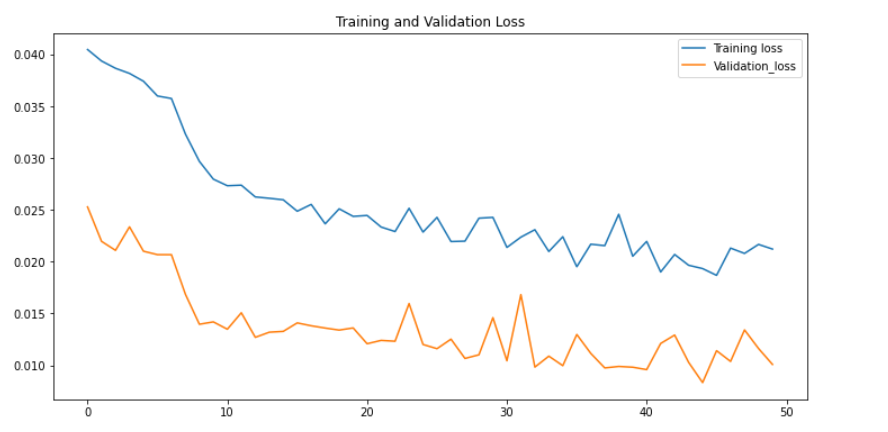
1. **Dense output layer:**

* Input shape: (None, 50) (50 units)
* Output shape: (None, 1) (1 unit)
* Number of parameters: (input dim + 1) \* units (for input weights and biases)
* Substituting the values: (1 + 1) \* 50 + 1 = 51 parameters

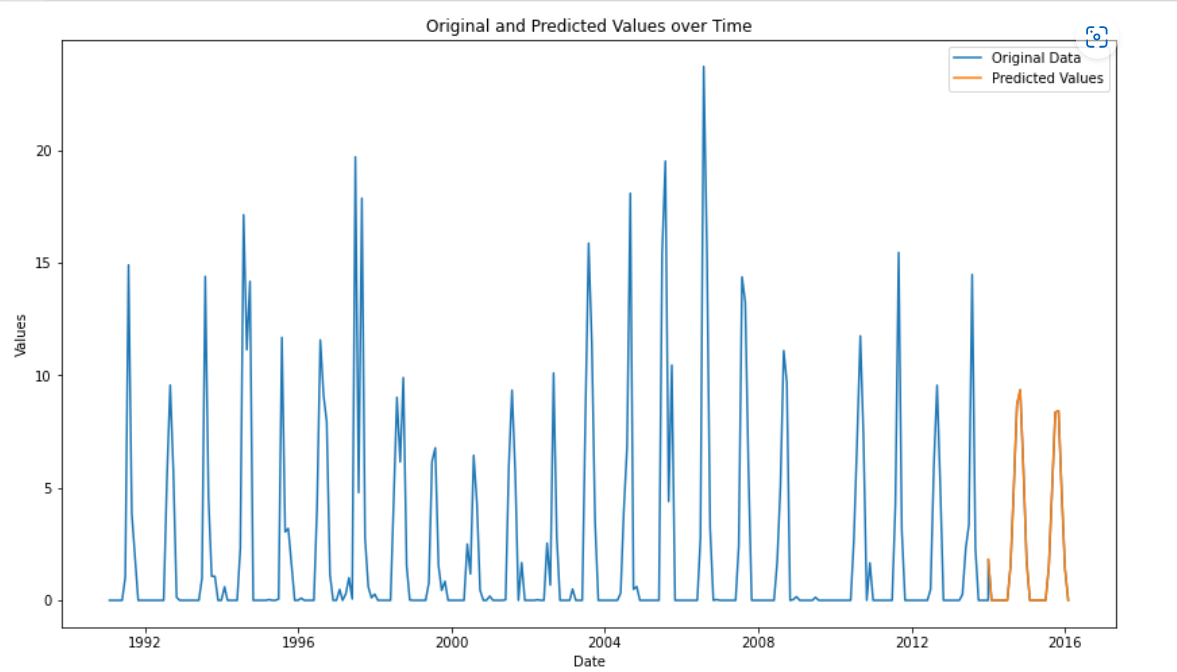
Total trainable parameters for the LSTM model = Parameters for all layers = 10,400 + 20,200 + 20,200 + 51 = 50,851

So, the correct number of trainable parameters for the LSTM model you provided is 50,851.

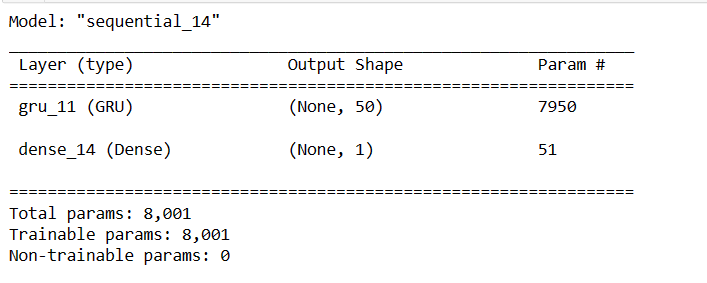
Visualization:



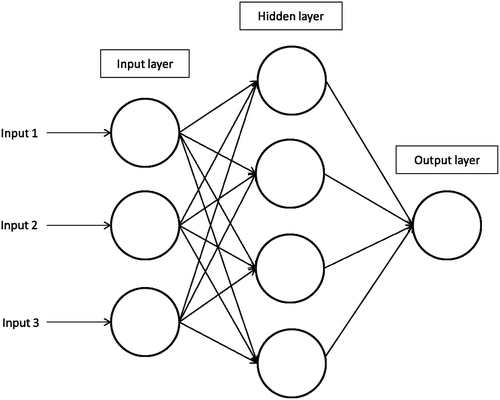
During the training process, the model learns from the training data and tries to minimize the loss function. On the other hand the purpose of the validation set is to evaluate the model's performance on unseen data. The validation loss gives an indication of how well the model is likely to perform on new, unseen data. However, small differences between the training and validation loss are generally expected. As in our case training loss is very close validation loss.



The above plot is of Prediction plot of LSTM taking **Year** on X-axis and **amount of monthly rainfall** on Y-axis. Also the model forecast the next future values.

1. Fitting GRU Model

Interpretation:



1

276 datapoints in the input layer of shape (70,1)

# How the number of trainable parameters for the GRU model are calculated?

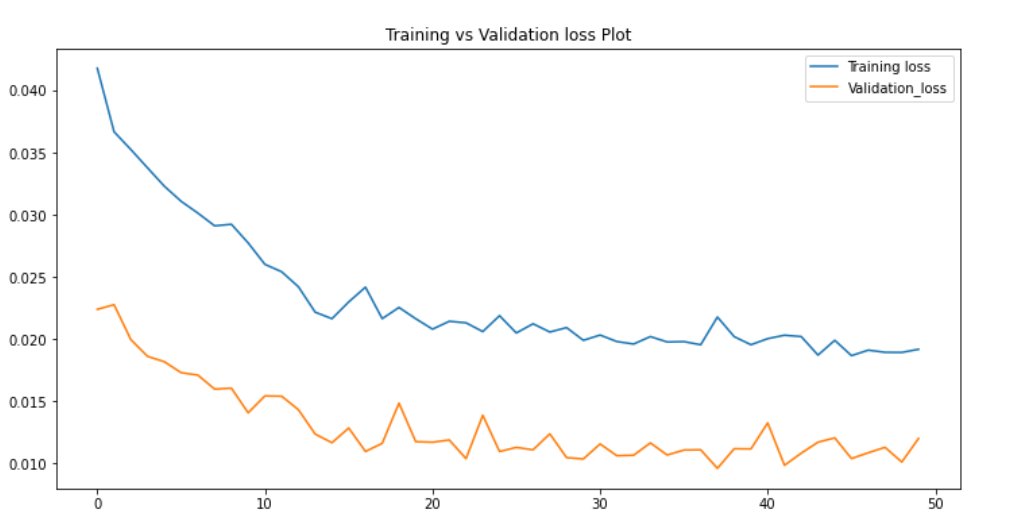
The number of parameters in a GRU layer is determined by the number of input features, the number of GRU units, and the number of biases. In this case, we have:

* Input shape: (batch size, time steps, input features) **=** (70, 1, 1)
* Number of GRU units: 50
* Number of biases: 3 (one for the reset gate, one for the update gate, and one for the candidate activation)

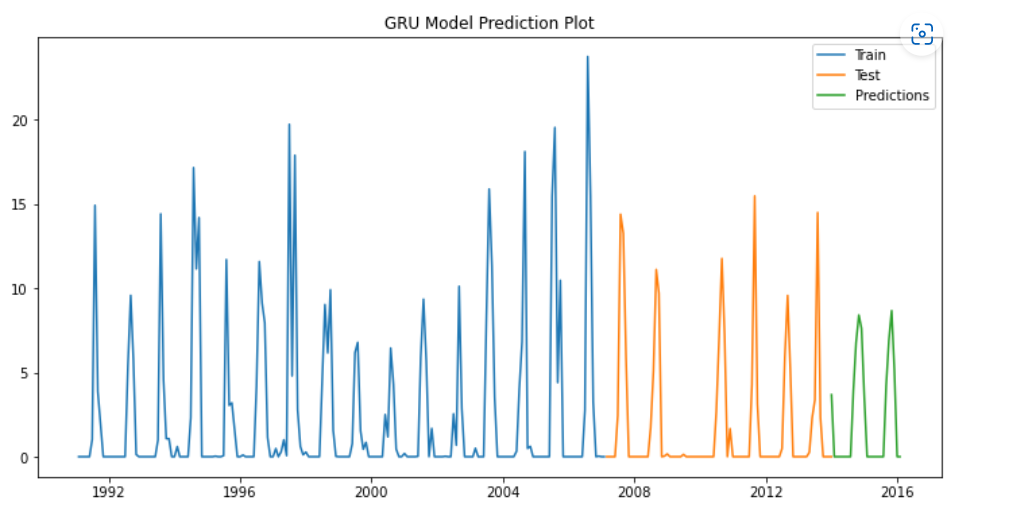
Using the formula (number of input features + number of GRU units + 1) x number of GRU units x 3, we can calculate the number of parameters as (1 + 50 + 1) x 50 x 3 = 7950. Therefore, we have 7950 trainable parameters in the GRU layer.

In the given GRU model summary, the total parameters are calculated by adding up the trainable parameters in each layer. In this case, there is only one GRU layer with 7,950 trainable parameters and one dense layer with 51 trainable parameters. Therefore, the total trainable parameters in this model are 7,950 + 51 = 8,001.

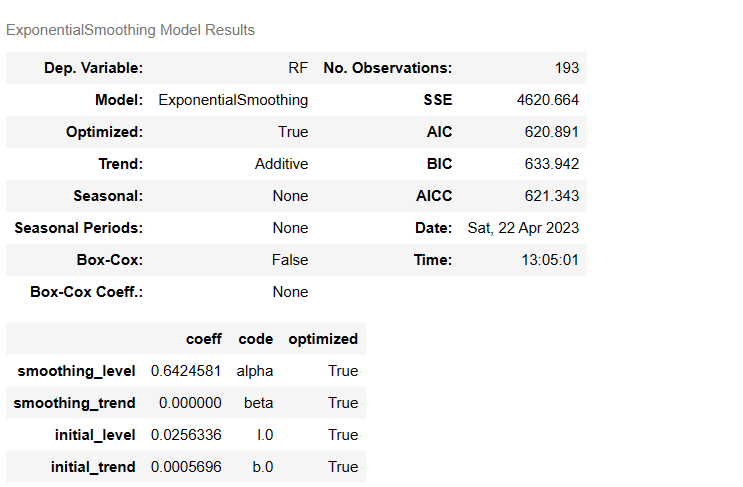
Visualization:



However, small differences between the training and validation loss are generally expected. As in our case there is very small difference in training and validation loss as they tends towards same direction.



The above plot is of Prediction plot of GRU model taking **Year** on X-axis and **amount of monthly rainfall** on Y-axis. We take training and testing data where it conclude that the model is well trained on the training data so we forecast the next future values using testing data, Also the model forecast the next future values.

1. Fitting Double Exponential Smoothing model

Interpretation:

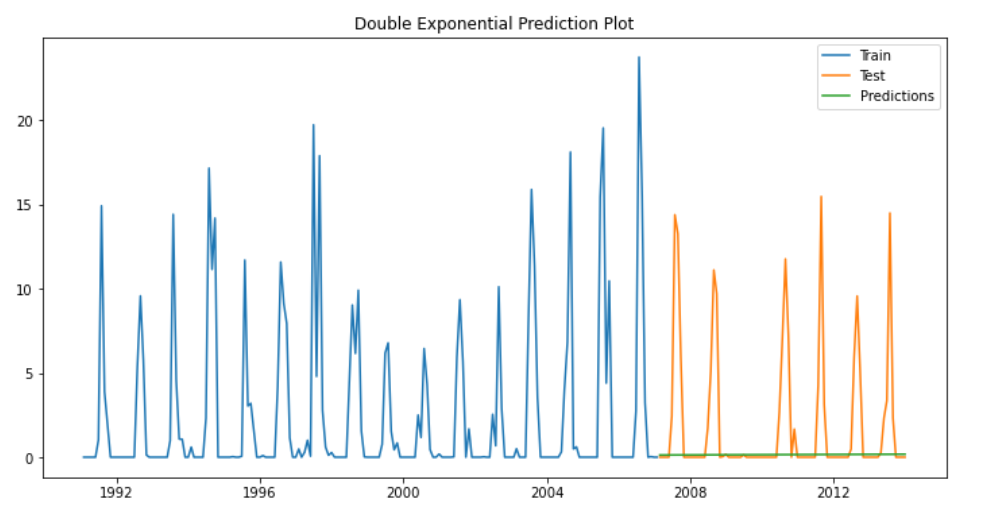
1. The Double Exponential model we get is,

=

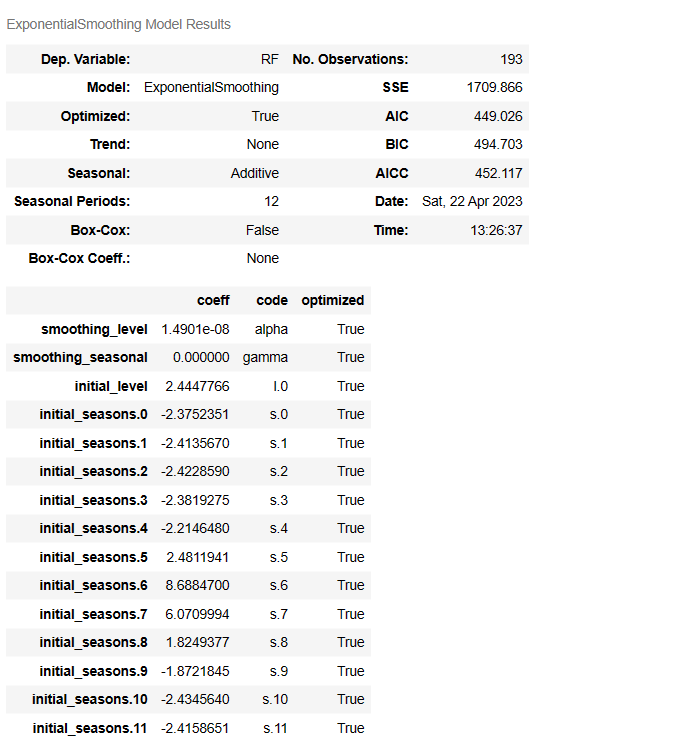
represents a forecast for time period t using the Exponential Smoothing method with a trend component.

1. The term 0.6424581y\_t represents the forecasted value of the time series variable () at time period t, multiplied by the smoothing level (0.6424581). The smoothing level is a parameter that determines the weight given to the most recent observation in the forecast.

Visualization:



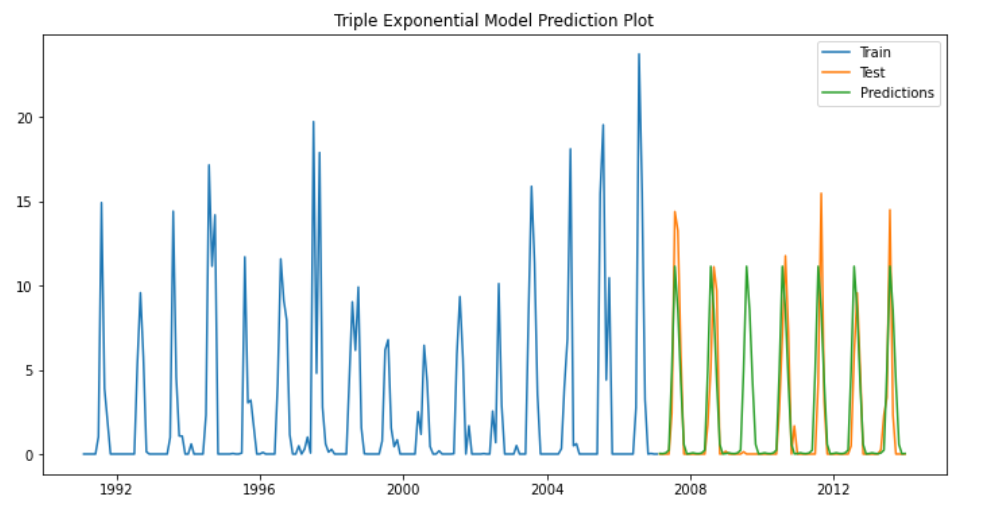
The above plot is of Prediction plot of Double Exponential smoothing model taking **Year** on X-axis and **amount of monthly rainfall** on Y-axis. We take training and testing data where it conclude that the model is not well trained on the training data also it does not give accurate results.

1. Fitting Triple Exponential Smoothing model

Interpretation:

1. The weight given to the current observed value is 0.6424581, while the weight given to the previous predicted value is 1-0.6424581. Additionally, there is a constant term of (0.0256336+0.0005696) included in the equation.

Visualization:



The above plot is of Prediction plot of Triple Exponential smoothing model taking **Year** on X-axis and **amount of monthly rainfall** on Y-axis. We take training and testing data where it conclude that the model is well trained on the training data so we forecast the next future values using testing data, Also our model forecast the next future values.

# CONCLUSION

Rainfall Prediction is the application area of data science and machine learning to predict

the state of the atmosphere.

The existing methods for rainfall forecasting fail in the most complicated situations because they cannot forecast the hidden patterns present, which are yet to be understood to perform an accurate prediction. There are other existing factors that causes in the accurate rainfall prediction. Here we use various approaches like some statistical models and recurrent neural networks.

|  |  |  |  |
| --- | --- | --- | --- |
| **Models** | **MAE** | **MSE** | **RMSE** |
| 1. Auto ARIMA | 3.9205 | 29.99 | 5.4766 |
| 2. ARIMA | 3.0396 | 15.3798 | 3.9217 |
| 3. Auto SARIMA | 1.4995 | 5.6504 | 2.3770 |
| 4. SARIMA | 1.6604 | 10.3901 | 3.2233 |
| 5. LSTM | 0.0604 | 0.0089 | 0.0944 |
| 6. GRU | 0.05 | 0.010 | 0.1040 |
| 1. Double Exponential | 1.99 | 18.38 | 4.29 |
| 8. Triple Exponential | 1.55 | 7.88 | 2.81 |

Our rainfall data has monthly seasonal pattern. We need different time series models to capture this seasonality. As from the above table, Auto ARIMA and ARIMA model does not capture the seasonality from the data. While Seasonal ARIMA model take this seasonality into account for future prediction.

Other than that exponential models take prediction as weighted averages of past observation. As mean squared error value of Triple Exponential model is very less than double exponential model. Because Triple Exponential model works better, when there is a sudden changes in the seasonality.

Over all on the basis of above accuracy table, LSTM is the best model for predicting the rainfall. Because it has very less Mean Squared Value of 0.0089 but if we consider the Mean Absolute Error value of GRU model, which is less as comparative to LSTM model then it shows that the GRU model is robust model. Overall, the ability of LSTMs to capture long-term dependencies and handle variable-length input sequences makes them a good choice for predicting rainfall. It proves that more complex methods are sometimes better than a simple methods.

# REFERENCES

|  |  |
| --- | --- |
| [1] | Y. S. G. M. M. A. Gowtham Sethupati M, “Efficient Rainfall Prediction and Analysis using Machine Learning Techniques,” *Turkish Journal of Computer and Mathematics Education,* pp. 144-150, 2013. |
| [2] | M. R. M. H. S. R. S H BARI, "Forecasting Monthly Precipitation in Sylhet City Using ARIMA Model," *Civil and Environmental Research,* vol. 7, pp. 69-78, 2015. |
| [3] | C. M. L. a. H. A. Melese, "Machine Learning techniques to predict daily rainfall amount," *Liyew and Melese ﻿Journal of Big Data,* pp. 01-11, 2021. |
| [4] | O. P. N. M. P. R. C. V.K.Somvanshi, "Modelling and Prediction of rainfall using artificial neural network and ARIMA techniques," *ResearchGate,* vol. 10, pp. 141-151, 2006. |
| [5] | A. B. N. R. A. P. K. A. L. Dr. C K Gomathy, "A STUDY ON RAINFALL PREDICTION TECHNIQUES," *International Journal of Scientific Research in Engineering and Management (IJSREM),* vol. 05, no. 10, pp. 01-15, 2021. |
| [6] | N. Gruber, D. R. Jockisch and F. M. Salem, 20 January 2017. [Online]. Available: https://en.wikipedia.org/wiki/Gated\_recurrent\_unit. [Accessed 28 March 2023]. |
| [7] | C. Olah, 27 August 2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed 02 February 2023]. |
| [8] | C. Chatfield, The Analysis of Time Series-An Introduction, Florida: CRC Press, 2004. |
| [9] | D. C. P. Damodar N. Gujarati, Basic Econometrics, New York: McGraw-Hill/Irwin, a business unit of The McGraw-Hill Companies, 2009. |

APPENDEX

All the project code files are stored on G**itHub** depository.

Link for code files: <https://github.com/bhava534/Msc-Final-Year-Project---Rainfall-Prediction-in-Anand-City>