WEB TRAFFIC FORECASTING USING ARIMA AND LSTM

## A PROJECT REPORT

***Submitted by***

**ANJANA SRIRAM**

**S.V.NAGESWARI**

**R.NIVEDHA NANDHINI**

***In partial fulfillment for the award of the degree of***

# BACHELOR OF TECHNOLOGY

***in***

## INFORMATION TECHNOLOGY

**EASWARI ENGINEERING COLLEGE, RAMAPURAM**

**ANNA UNIVERSITY: CHENNAI 600 025**

**APRIL 2021**

**ANNA UNIVERSITY: CHENNAI 600 025**

# BONAFIDE CERTIFICATE

Certified that this project report titled **“WEB TRAFFIC FORECASTING USING ARIMA AND LSTM”** is the bonafide work of **“ANJANA SRIRAM (310617205007), S.V.NAGESWARI (310617205045)** and **R.NIVEDHA NANDHINI (310617205054)**” who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

Dr.Anandhi,

## HEAD OF THE DEPARTMENT

Dept. of Information Technology,

Easwari Engineering College,

Bharathi salai, Ramapuram,

Chennai-600 089

Dr.A.K.MARIAPPAN,M.E.,(Ph.D.)

## SUPERVISOR

Professor,

Dept. of Information Technology,

Easwari Engineering College, Bharathi salai, Ramapuram, Chennai-600 089

# VIVA VOCE EXAMINATION

The viva voce examination of the project work, submitted by

## ANJANA SRIRAM, Register Number: 310617205007,

**NAGESWARI.S.V., Register Number: 310617205045** and

**NIVEDHA NANDHINI.R., Register Number: 310617205054 is** held on ……………………………....

## INTERNAL EXAMINER EXTERNAL EXAMINER

**ACKNOWLEDGEMENT**

We express our sincere thanks and gratitude to our Founder Chairman **Dr.T.R.PARIVENDHAR,** and our beloved Chairman **Dr.R.SHIVAKUMAR,** for their support and inspiration. We would like to convey our due respect and regards to our Principal and Head **Dr.K.KATHIRAVAN,** for his constant encouragement and guidance.

With a deep sense of gratitude, we would like to thank **Prof.G.RAMAKRISHNAN,** for his motivation, timely help and valuable suggestions. We would also like to thank our beloved professors **Prof.A.K.MARIAPPAN** and **Dr.D.SIVAKUMAR,** for their profound advice.

We owe our profound gratitude to our project supervisor **Prof.A.K.MARIAPPAN,** Professor, who took keen interest in our project work and guided us all along in bringing out this project in complete shape by providing all the necessary information for developing a good system.

We would like to sincerely thank our project coordinators **Prof.A.K.MARIAPPAN,** Professor and **Prof.A.K.MARIAPPAN,** Professor for their valuable assistance and support.

Finally, we would like to extend our token of appreciation to all the teaching and non-teaching staff of Department of Information Technology for their kind co-operation throughout the project.

# ABSTRACT

Website traffic refers to web users who visit a website. Web traffic is measured in visits, sometimes called "sessions," and is a common way to measure an online business effectiveness at attracting an audience.

When someone visits a website, their computer or other web-connected device communicates with the website's server. Each page on the web is made up of dozens of distinct files. The site's server transmits each file to user browsers where they are assembled and formed into a cumulative piece with graphics and text. Every file sent represents a single “hit”, so a single page viewing can result in numerous hits. This is how the web traffic is recorded.

In today’s world web traffic is one of the serious issues faced by many. Web traffic tends to hinder the smooth user experience and it is also very challenging for the web service providers to maintain a smooth user-server interaction. We are looking to overcome this problem by building a prediction model to forecast the web traffic in advance to avoid all the problems faced. Our model thoroughly studies the previous web traffic data to efficiently predict the web traffic of a particular website at a given point in time.

Forecasting is one of the important goals of mining time-series databases. The efficacy of Time series forecasting has been proved while decision making in various domains. This method is vastly different from the other proposed methods for prediction and analysis. This paper proposes the use of ARIMA and LSTM algorithms to forecast web traffic.

## TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | **v** |
|  | **LIST OF TABLES** | **ix** |
|  | **LIST OF FIGURES** | **x** |
|  | **LIST OF ABBREVIATIONS** | **xi** |
| **1.** | **INTRODUCTION** | **1** |
|  | 1.1 OBJECTIVE | 1 |
|  | 1.2 METHODOLOGY | 2 |
| **2.** | **LITERATURE SURVEY** | **3** |
|  | 2.1 RELATED WORKS | 3 |
|  | 2.2 EXISTING SYSTEM | 6 |
| **3.** | **PROPOSED SYSTEM DESIGN** | **7** |
|  | 3.1 INTRODUCTION | 7 |
|  | 3.2 PROPOSED SYSTEM | 7 |
|  | 3.3 SYSTEM ARCHITECTURE | 8 |
|  | 3.3.1 Multivariate dataset | 8 |
|  | 3.3.2 Data pre-processing | 9 |
|  | 3.3.3 Training and validating using ARIMA | 10 |
|  | 3.3.4 Training and validating using LSTM RNN | 12 |
|  | 3.3.5 Combining ARIMA and LSTM | 13 |
|  | 3.3.6 Deploying the forecasted model | 14 |
| **4.** | **SYSTEM IMPLEMENTATION** | **15** |
|  | 4.1 INTRODUCTION | 15 |
|  | 4.2 HARDWARE AND SOFTWARE SPECIFICATION | 15 |
|  | 4.3 TECHNOLOGIES USED | 16 |
|  | 4.3.1 Python | 16 |
|  | 4.3.2 CUDA | 17 |
|  | 4.3.3 NumPy | 18 |
|  | 4.3.4 Pandas | 19 |
|  | 4.3.5 Statsmodel | 20 |
|  | 4.3.6 Scikit-learn | 20 |
|  | 4.3.7 TensorFlow | 20 |
|  | 4.3.8 Keras | 22 |
|  | 4.3.9 Matplotlib | 22 |
|  | 4.3.10 Streamlit | 23 |
|  | 4.4 MODULES IMPLEMENTATAION | 23 |
|  | 4.4.1 ARIMA | 23 |
|  | 4.4.2 LSTM | 26 |
|  | 4.3.3 Ensemble ARIMA and LSTM | 29 |
|  | 4.5 NON-FUNCTIONAL REQUIREMENT | 31 |
|  |  |  |
|  |  |  |
|  |  |  |
| **5.** | **PERFORMANCE ANALYSIS** | **33** |
|  | 5.1 TESTING | 33 |
|  | 5.2 EXPERIMENTAL RESULTS | 36 |
|  | 5.2.1 Mean Forecast Error (MFE) | 36 |
|  | 5.2.2 Mean Absolute Error (MAE) | 37 |
|  | 5.2.3 Mean Squared Error (MSE) | 38 |
|  | 5.2.4 Root Mean Squared Error (RMSE) | 39 |
|  | 5.3.5 Mean Absolute Percentage Error (MAPE) | 39 |
|  | 5.3 PERFORMANCE EVALUATION | 40 |
|  | 5.4 SUMMARY | 40 |
| **6.** | **CONCLUSION AND FUTURE WORKS** | **41** |
|  | 6.1 CONCLUSION | 41 |
|  | 6.2 FUTURE WORKS | 41 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE NO.** | **TITLE** | **PAGE NO.** |
| 5.1 | Mean Forecast Error Explanation | 36 |
| 5.2 | Error Metrics | 40 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| FIGURE NO. | TITLE | PAGE NO. |
| 3.1 | System Design | 8 |
| 3.2 | Wikipedia Traffic Dataset | 8 |
| 3.3 | Filled missing values | 9 |
| 3.4 | ACF and PACF | 11 |
| 3.5 | ARIMA model | 11 |
| 3.6 | LSTM model | 13 |
| 3.7 | Hybrid model | 14 |
| 3.8 | Website | 14 |
| 4.1 | ARIMA Flow Chart | 26 |
| 4.2 | LSTM Architecture | 29 |
| 4.3 | Ensemble Architecture | 30 |
| 5.1 | MAE Formula | 37 |
| 5.2 | MSE Formula | 38 |
| 5.3 | RMSE Formula | 39 |
| 5.4 | MAPE Formula | 39 |

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| ARIMA | Autoregressive Integrated Moving Average |
| ANN | Artificial Neural Networks |
| LSTM RNN | Long Short Term Memory Recurrent Neural Network |
| DWT | Discrete Wavelet Transform |
| ACF | Autocorrelation Function |
| PACF | Partcial Autocorrelation Function |
| MA | Moving Average |
| AR | Autoregressive |
| ML | Machine Learning |
| DL | Deep Learning |
| RAM | Random Access Memory |
| GPU | Graphics Processing Unit |
| IDE | Integrated Development Environment |
| API | Application Programming Interface |
| CUDA | Computer Unified Device Architecture |
| CuDNN | CUDA Deep Neural Network |
| SciPy | Scientific Python |
| MFE | Mean Forecast Error |
| MAE | Mean Absolute Error |
| MSE | Mean Squared Error |
| RMSE | Root Mean Squared Error |
| MAPE | Mean Absolute Percentage Error |
| CSV | Comma Separated Values |
|  |  |

**CHAPTER 1**

**INTRODUCTION**

In today’s world web traffic is one of the serious issues faced by many. Due to high usage of the internet, the rise in traffic is perceived. Web traffic tends to hinder the smooth user experience and it is also very challenging for the web service providers to maintain a smooth user- server interaction. High web traffic causes websites to experience a lag while loading and the site may even crash at times.

We may have gone over a few internet businesses sites that may crash when the number surpasses the expected limit. which causes a lot of weight for the customers and on account of that it could reduce the customer's evaluations of that website and the customers will prefer and use more stable websites which might result in the decline in one’s trade.

We are looking to overcome this problem by building a prediction model to forecast the web traffic in advance to avoid all the problems faced. This model thoroughly studies the previous web traffic data to efficiently predict the web traffic of a particular website at a given point of time. To achieve a higher level of accuracy in prediction, hybrids of two algorithms are used.

* 1. **OBJECTIVES**
* To predict web traffic in advance using deep learning algorithms.
* Creating a hybrid solution using two different algorithms to produce higher level of accuracy.
* Determining an effective strategy for load balancing of web pages residing in the cloud.
* To provide time series based data for auto scaling the system resources.
* Providing an user interface, where any user can import their data file and obtain predicted value.
* To provide better user experience by predicting web traffic in advance.
* To reduce infrastructural cost when web traffic is predicted to be lower than average.
  1. **METHODOLOGY**

In this methodology we are going to predicts the web traffic ahead of time with the goal that the necessary server can be assigned well ahead of time subsequently forestalling the event of an accident. This model aids in overseeing and limiting accidents adequately which thus forestalls the deficiency of an Organization.

Our Model is a half and half multivariate model as our model is assembled utilizing both ARIMA and LSTM which radically builds the proficiency of individual calculations. ARIMA is best with linear data and LSTM is best with non-linear data. Our model is subsequently acceptable with the two kinds of reports. The yield of the ARIMA is given as input to the LSTM consequently training the dataset twice and thus acquiring better outcomes.

**CHAPTER 2**

**LITRATURE SURVEY**

**2.1 RELATED WORKS**

Rodrigo N. Calheiros et al.[1] ,they provide a Cloud based workload prediction module for SaaS suppliers using the Autoregressive Integrated Moving Average (ARIMA) model. They presented the prediction based on the ARIMA model and estimated its accuracy of future workload prediction by using the real traces of requests to the web servers. They also calculated the impact of the achieved accuracy with respect to the efficiency in resource utilization and QoS. Simulated results show that their model is able to accomplish an average accuracy of up to 91%, which further leads to efficiency in resource usage with minimum impact on the quality of service.

G., P. Zhang .[2], they present a hybrid methodology that consolidates both Autoregressive integrated moving average (ARIMA) and artificial neural networks (ANNs) models. The hybrid method takes advantage of the unique characteristics provided by ARIMA and ANN models in linear and nonlinear modeling. The hybrid methodology considers factors such as sampling variation, model uncertainty, and structure change to provide results thus experimental results with real datasets demonstrates that the consolidated model can be an effective method to improve forecasting accuracy accomplished by either of the models used individually.

Tejas Shelatkar et al.[3] they presented web traffic Time series prediction which is performed using Long Short Term Memory Recurrent Neural Network (LSTM RNN)and Autoregressive integrated moving average(ARIMA) more efficiently and accurately. The system predicts the number of users who will access the website in the future. The system will keep on upgrading and produce accurate results as more user data is fed. The system can be used by any user for improving their web traffic load management and business analysis. LSTM RNN provides more accuracy to the system. The system effectively records seasonal and long-term patterns including information such as holidays, day of week, language, region which will help our model to capture the trends of the data more efficiently.

Saman Feghhi et al.[4]they introduced an attack on the encrypted web traffic that utilizes only the packet timing data on the uplink. This attack is therefore impenetrable to existing packet padding defenses. Likewise, in contrast to existing approaches, this timing-only attack does not need the information on the start or end of the web fetches and so is effective against traffic streams. We exhibit the effectiveness of the attack against the wired and wireless traffic, accomplishing average success rates of 90%. Likewise this timing-only attack serves to emphasize deficiencies in the already present defenses and also to the areas where it would be useful for virtual private network (VPN) designers to concentrate their further attention.

Rishabh Madan at al.[5]they have presented a time series forecasting technique to forecast internet traffic based on prior values. Numerous forecasting techniques such as ARIMA are used for making predictions, but, it is mostly convenient for a time series which is linear. Whereas, neural networks like RNN are capable of predicting time series which are nonlinear. The presented system uses Discrete Wavelet Transform(DWT) and uses a high pass filter and a low pass filter resulting in linear and nonlinear parts for the time series. The proposed technique is more efficient and accurate than ARIMA and RNN individually.

Navyasree Petluri et al.[6]they propose a system where they utilize existing Web Traffic Time Series Forecasting dataset by Google to forecast future traffic of the Wikipedia site. Forecasting web traffic is used to aids website owners to regulate an effective technique for load balancing of web pages present in the cloud, forecasting future patterns based on prior data and comprehend the user behavior. They built a time-series model that uses RNN seq2seq model. They use symmetric mean absolute percentage error (SMAPE) for measuring the complete efficiency and accuracy of the developed model. Finally, evaluating the result of the developed model to the existing ones to determine the effectiveness of the presented method in forecasting future traffic of Wikipedia articles.

Seyyed Meysam et al.[7] they have proposed a system that deals with the issue of detecting DoS and DDoS attacks. Two features number of packets and source IP addresses are utilized as detection metrics that are calculated from network traffic every minute. Thus, a time series based on the number of packets is created using a Box-Cox transformation. An ARIMA model is also used for forecasting the number of packets in every minute. Then, using Lyapunov exponents and categorizing the chaotic behaviour the system differentiates normal traffic and attack traffics from one other. Simulation outcomes present that the proposed system can efficiently differentiate 99.5% of traffic states.

Soheila Mehrmolaei et al[8]In this paper, they have proposed time series forecasting techniques to categorize and place two groups on the basis of forecasting duration. Moreover, a technique is presented by applying an average of estimation error for time series forecasting in ARlMA model. With respect to the outcome the improved ARlMA model is efficient than basic ARlMA model. As the future work, implementation of the application of the proposed approach in multivariate time series data sets can be implemented.

**2.2 EXISTING SYSTEM**

The existing system is observed and inferred from different references. The references uses different algorithm on the given data to predict or forecast the future workload to help the users. The prediction in used for basic traffic forecasting or to defend against attacks. Most system works on a single algorithm to predict the data, where the accuracy of the data is incompetent compared to the hybrid predicting systems. Hybrid systems that utilizes both LSTM and ARIMA uses univariate time series data and the system doesn’t support multivariate time series data.

**DISADVANTAGES**

Existing system deploys the prediction system for single characteristic or attribute. Multivariate dataset is not supported in this system.

**CHAPTER 3**

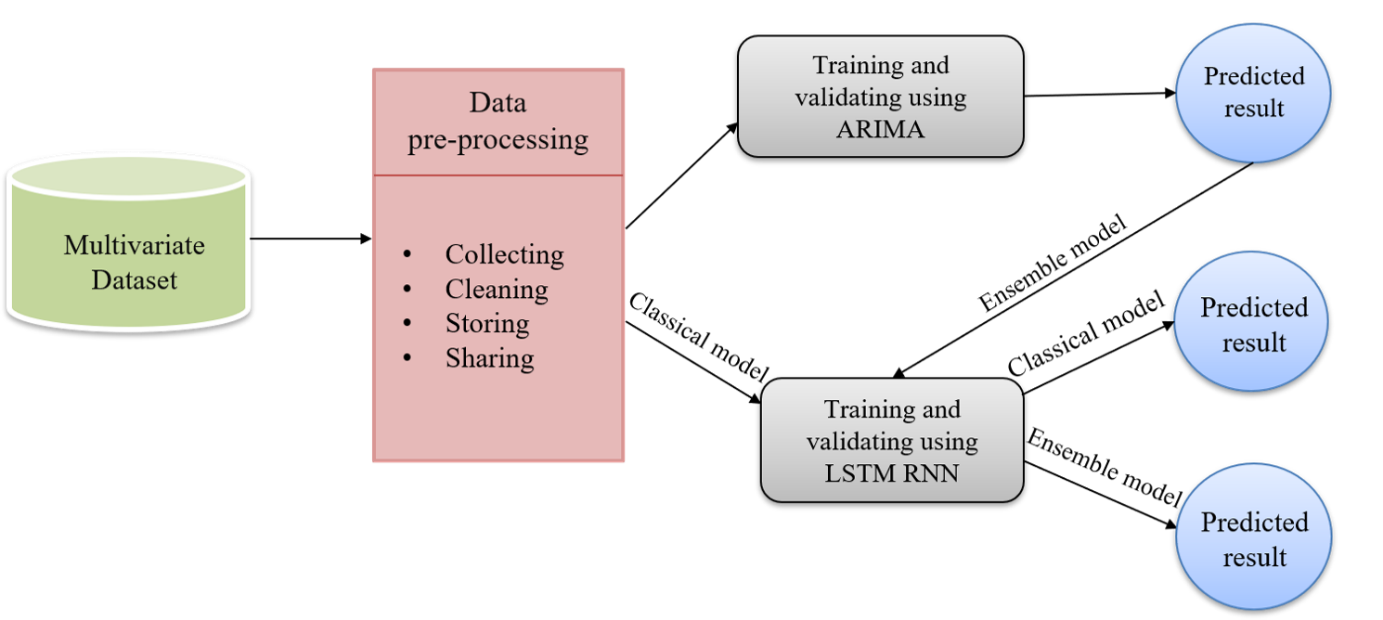
**PROPOSED SYSTEM DESIGN**

**3.1 INTRODUCTION**

Proposed system design describes the entire flow of the system and architecture. It is the conceptual model that defines the structure and behavior. It comprises of system components, the externally properties of the components and their relationships between them. It provides a plan for the product.

At the point when the quantity of hits increments past the limit of a site, it will in general crash consequently making a gigantic misfortune for an organization. To evade this, we have come up with a prediction model which predicts the web traffic ahead of time with the goal that the necessary server can be assigned well ahead of time subsequently forestalling the event of an accident. This model aids in overseeing and limiting accidents adequately which thus forestalls the deficiency of an Organization. Our Model is a half and half multivariate model as our model is assembled utilizing both ARIMA and LSTM which radically builds the proficiency of individual calculations. ARIMA is best with linear data and LSTM is best with non-linear data. Our model is subsequently acceptable with the two kinds of reports. The yield of the ARIMA is given as input to the LSTM consequently training the dataset twice and thus acquiring better outcomes.

**3.2 SYSTEM ARCHITECTURE**

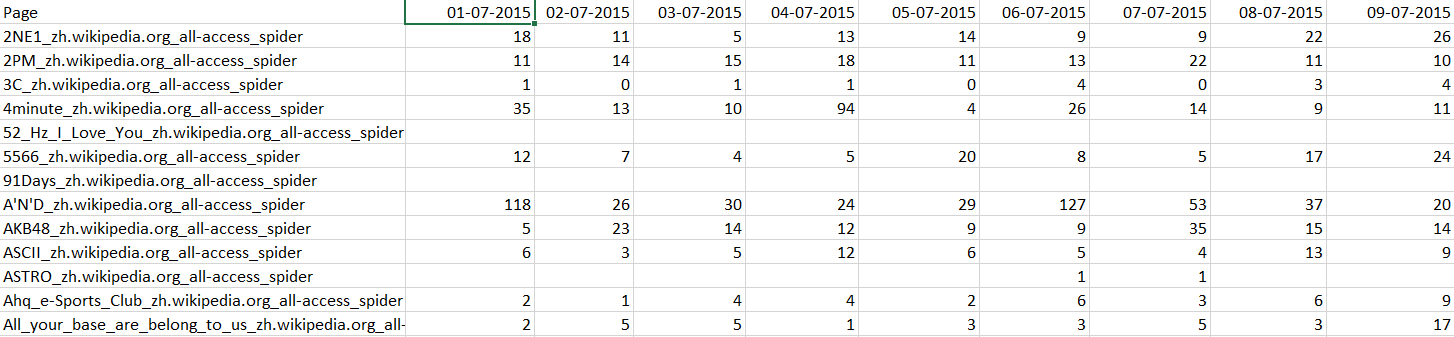


**Figure 3.1 System design**

**3.3 PROPOSED SYSTEM**

**3.3.1 Multivariate dataset**

Multivariate data is the data in which analysis are based on more than two variables per observation. Usually, multivariate data is used for explanatory purposes. The multivariate dataset utilized in this project is Wikipedia Traffic given by Kaggle.



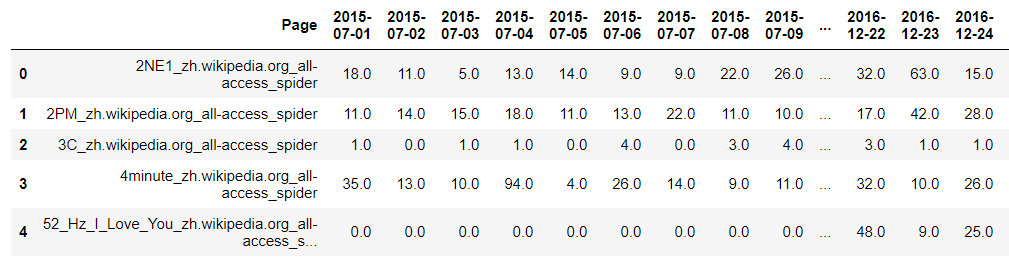
**Figure 3.2 Wikipedia Traffic dataset**

The above dataset comprises of some missing values so it goes through the data pre-processing step prior to training and validating.

**3.3.2 Data pre-processing**

Data pre-processing is an information mining procedure that includes changing crude information into a reasonable configuration. Raw data is regularly deficient, conflicting, and additionally ailing in specific practices or drifts, and is probably going to contain numerous mistakes. Data pre-processing includes four steps, collecting, cleaning, storing and sharing.

As the first step, the dataset adopted for this project is daily views of Wikipedia articles provided by Kaggle comprising roughly 145,000 records. The dataset involves two fields, date and page. Page field comprises more than 1 lakh Wikipedia articles and the date field shows the number of hits day by day. The next step involves data cleaning, the gathered Wikipedia dataset may contain some missing values and this process fills the missing values with zero and organizes the raw data for the following steps. When the dataset is cleaned and stacked, ensure that the dataset stored is right, prior to continuing further.



**Figure 3.3 Filled missing values**

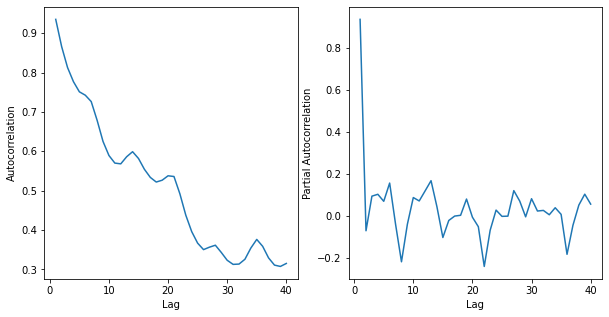
The refined dataset in its raw state cannot be supplied to the deep learning model which could lead to misleading prediction or an accuracy deficient model. To overcome these data-oriented challenges, the data set has to be segregated into a coarse-grained variant of itself which not only reduces the complexity of the dataset but also provides a clear and segregated view of data. This stage is called as data framing and in this stage, the dataset used to train the model is transposed from its existing state and classified based on the linguistic preferences of the user (hits based on user’s language preferences). Whence the dataset is segregated, the model is fed with the framed structure which initiates the training process. The model plots the graphical representation of the segregated dataset which serves as the precursor for the whole prediction model which will be later put forth on the task of forecasting the future state of plausible user hits the target site / server could get at the predicted time frame.

**3.3.3 Training and validating using ARIMA**

ARIMA (Auto-Regressive Integrated Moving Average) is really a class of models that 'clarifies' a given time arrangement dependent on its own previous values, that is, its own lags and the lagged forecast mistakes, so condition can be utilized to conjecture future values. ARIMA is isolated into three sections:

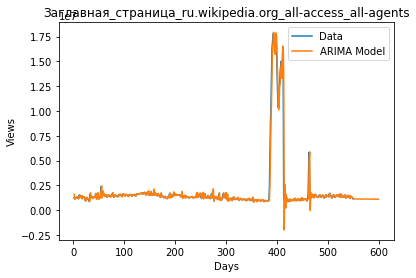
* Autoregressive (AR) forecasts future outcome from the past value.
* Integrated (I), it has to do with the distinction in time arrangement.
* Moving average (MA) model doesn't utilize the previous estimates to anticipate the future qualities though it utilizes the blunders from the past result.

Before training the data frames ARIMA model goes through different steps and one of them is its plot autocorrelation function (ACF) and partial autocorrelation (PACF) to distinguish the potential MA and AR model.



**Figure 3.4 ACF and PACF**

Based on the ACF and PACF values, the best ARIMA fir model is found for training and validating. In this manner, the forecast outcome is generated and plotted with the best model.



**Figure 3.5 ARIMA model**

ARIMA model at times can anticipate the week-by-week base of the sign, which is acceptable. In different cases, it appears to simply give a direct fit. This is conceivably valuable.

Be that as it may, on the off chance that we just aimlessly apply the ARIMA model to the entire dataset, the outcomes are not close to the same as utilizing the basic models. It actually appears to make them interesting properties, so perhaps we can consolidate this with another model to improve results.

The performance of ARIMA is evaluated by Mean Absolute Percentage Error (MAPE) is 15.704892.

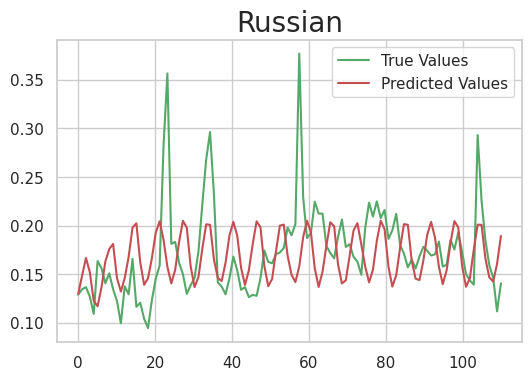
**3.3.4 Training and validating using LSTM RNN**

Recurrent neural networks (RNNs) can predict the next value(s) in a sequence or classify it. A sequence is stored as a matrix, where each row is a feature vector that describes it. Naturally, the order of the rows in the matrix is important.

RNNs are a really good fit for solving Natural Language Processing (NLP) tasks where the words in a text form sequences and their position matters. That said, cutting edge NLP uses [the Transformer](https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)) for most (if not all) tasks.

LSTM represents Long short-term memory is a self-supervised learning method, it is appropriate for both univariate and multivariate dataset. In this project, the multivariate dataset is utilized.Amultivariate dataset implies where there is more than one field to forecast.

Subsequent to pre-processing, the dataset is split for training and testing. Converting the split dataset into NumPy array and reshaped the array (3D) to which the LSTM model accepts. At that point construct the LSTM design. Fabricated model train and test the dataset for evaluating the performance.

****

**Figure 3.6 LSTM model**

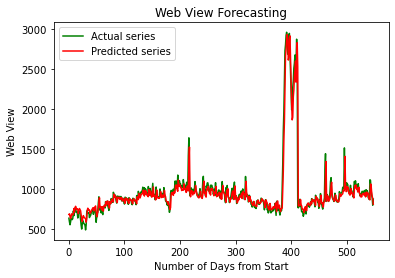
The performance of LSTM is evaluated by Mean Absolute Percentage Error (MAPE) is 35.686939.

**3.3.5 Combining ARIMA and LSTM**

Ensemble learning methods are widely used nowadays for its predictive performance improvement. Ensemble learning combines multiple predictions (forecasts) from one or multiple methods to overcome accuracy of simple prediction and to avoid possible overfit.

The over two models are fitted depending on their best. The ARIMA model suits well for the linear dataset and LSTM works out positively for the non-linear dataset. As an individual outcome, both show their anticipated worth with some flaws. For better precision and result, the two models are ensemble together.

As the blend of both the models, from the outset ARIMA model's yield is given as the contribution of the LSTM model. Along these lines the dataset is trained twice. With this mix, the precision level expanded and the rate blunder decreased. MAPE score for the ensembled model is 7.862425, lower than ARIMA and LSTM MAPE score exclusively.

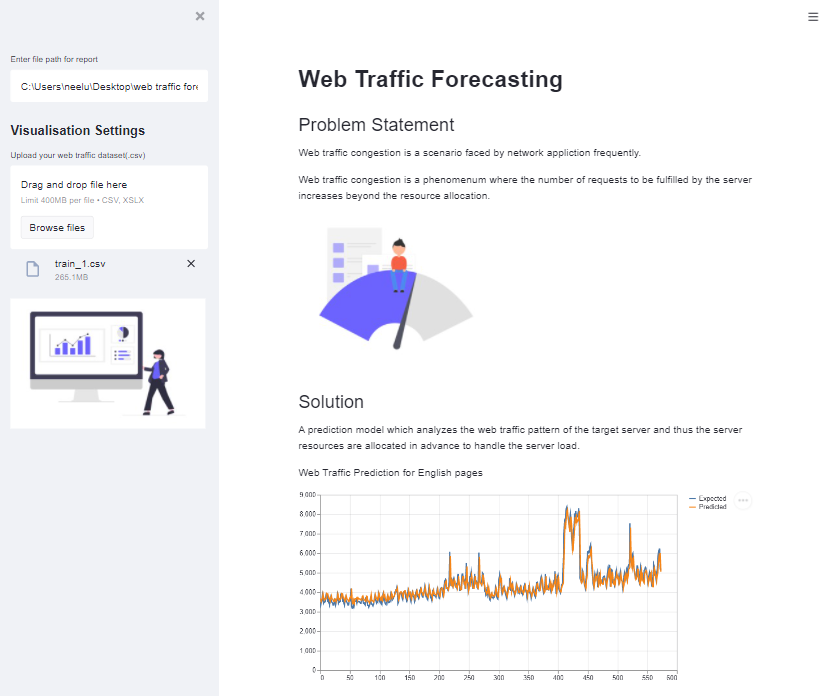


**Figure 3.7 Hybrid model**

**3.3.6 Deploying the forecasted model**

When the models are prepared and approved with ARIMA and LSTM, the forecasted model is incorporated with a website. The application is created with streamlit which is an open-source structure for deploying ML models.

The customer can discover their site traffic by uploading the past hits (ensure it is a multivariate dataset). The customer can see their traffic in graphical portrayal and can download the traffic record as a text file.



**Figure 3.8 Website**

**CHAPTER 4**

**SYSTEM IMPLEMENTATION**

**4.1 INTRODUCTION**

Implementation is the carrying out, execution, or practice of a plan, a method, or any design, idea, model, specification, standard or policy for doing something. As such, implementation is the action that must follow any preliminary thinking in order for something to actually happen.

The various modules, specifications required and technologies used are listed below. Each of the functionalities describes the usage and properties of the modules. This helps in a clearer view of the working of the model.

**4.2 HARDWARE AND SOFTWARE SPECIFICATIONS**

# HARDWARE SYSTEM CONFIGURATION:

* Processor - Intel
* Speed - 1.8 GHz
* RAM - 8 GB
* Hard Disk - 260 GB
* GPU - NVIDIA

# SOFTWARE SYSTEM CONFIGURATION:

* Operating System - Windows 8/10
* IDE - PyCharm
* Distributor - Anaconda
* Notebook - Jupyter
* Browser - Chrome/Firefox/Edge

**4.3 TECHNOLOGIES USED**

**4.3.1 Python**

**What is Python?**

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple

approach very effective.

**Why python is used for deep learning?**

Python offers concise and readable code. While complex algorithms and versatile workflows stand behind machine learning and AI, Python’s simplicity allows developers to write reliable systems. Developers get to put all their effort into solving an DL problem instead of focusing on the technical nuances of the language.

Additionally, Python is appealing to many developers as it’s easy to learn. Python code is understandable by humans, which makes it easier to build models for deep learning.

Many programmers say that Python is more intuitive than other programming languages. Others point out the many frameworks, libraries, and extensions that simplify the implementation of different functionalities. It’s generally accepted that Python is suitable for collaborative implementation when multiple developers are involved. Since Python is a general-purpose language, it can do a set of complex deep learning tasks and enable you to build prototypes quickly that allow you to test your product for deep learning purposes.

**4.3.2 CUDA**

[CUDA](https://developer.nvidia.com/about-cuda)is a parallel computing platform and programming model developed by Nvidia for general computing on its own GPUs (graphics processing units). CUDA enables developers to speed up compute-intensive applications by harnessing the power of GPUs for the parallelizable part of the computation.

**CUDA in Deep Learning**

[Deep learning](https://www.infoworld.com/article/3163130/artificial-intelligence/what-deep-learning-really-means.html) has an outsized need for computing speed. For example, [to train the models for Google Translate in 2016](https://www.nytimes.com/2016/12/14/magazine/the-great-ai-awakening.html), the Google Brain and Google Translate teams did hundreds of one-week [TensorFlow](https://www.infoworld.com/article/3278008/tensorflow/what-is-tensorflow-the-machine-learning-library-explained.html)runs using GPUs; they had bought 2,000 server-grade GPUs from Nvidia for the purpose. Without GPUs, those training runs would have taken months rather than a week to converge. For production deployment of those TensorFlow translation models, Google used a new custom processing chip, the TPU (tensor processing unit).

In addition to TensorFlow, many other DL frameworks rely on CUDA for their GPU support, including Caffe2, CNTK, Databricks, H2O.ai, Keras, MXNet, PyTorch, Theano, and Torch. In most cases they use the [cuDNN](https://developer.nvidia.com/cudnn)library for the deep neural network computations. That library is so important to the training of the deep learning frameworks that all of the frameworks using a given version of cuDNN have essentially the same performance numbers for equivalent use cases. When CUDA and cuDNN improve from version to version, all of the deep learning frameworks that update to the new version see the performance gains. Where the performance tends to differ from framework to framework is in how well they scale to multiple GPUs and multiple nodes.

**4.3.3 NumPy**

NumPy is a library for the [Python programming language](https://en.wikipedia.org/wiki/Python_(programming_language)), adding support for large, multi-dimensional [arrays](https://en.wikipedia.org/wiki/Array_data_structure) and [matrices](https://en.wikipedia.org/wiki/Matrix_(math)), along with a large collection of [high-level](https://en.wikipedia.org/wiki/High-level_programming_language) [mathematical](https://en.wikipedia.org/wiki/Mathematics) [functions](https://en.wikipedia.org/wiki/Function_(mathematics)) to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by [Jim Hugunin](https://en.wikipedia.org/wiki/Jim_Hugunin) with contributions from several other developers. In 2005, [Travis Oliphant](https://en.wikipedia.org/wiki/Travis_Oliphant) created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is [open-source software](https://en.wikipedia.org/wiki/Open-source_software) and has many contributors.

Using NumPy in Python gives functionality comparable to [MATLAB](https://en.wikipedia.org/wiki/MATLAB) since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of [scalars](https://en.wikipedia.org/wiki/Scalar_(computing)). In comparison, MATLAB boasts a large number of additional toolboxes, notably [Simulink](https://en.wikipedia.org/wiki/Simulink), whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; [SciPy](https://en.wikipedia.org/wiki/SciPy) is a library that adds more MATLAB-like functionality and [Matplotlib](https://en.wikipedia.org/wiki/Matplotlib) is a plotting package that provides MATLAB-like plotting functionality. Internally, both MATLAB and NumPy rely on [BLAS](https://en.wikipedia.org/wiki/Basic_Linear_Algebra_Subprograms) and [LAPACK](https://en.wikipedia.org/wiki/LAPACK) for efficient linear algebra computations.

**4.3.4 Pandas**

In [computer programming](https://en.wikipedia.org/wiki/Computer_programming), pandas is a [software library](https://en.wikipedia.org/wiki/Software_library) written for the [Python programming language](https://en.wikipedia.org/wiki/Python_(programming_language)) for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and [time series](https://en.wikipedia.org/wiki/Time_series). It is [free software](https://en.wikipedia.org/wiki/Free_software) released under the [three-clause BSD license](https://en.wikipedia.org/wiki/3-clause_BSD_license). The name is derived from the term "[panel data](https://en.wikipedia.org/wiki/Panel_data)", an [econometrics](https://en.wikipedia.org/wiki/Econometrics) term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself. [Wes McKinney](https://en.wikipedia.org/wiki/Wes_McKinney) started building what would become pandas at [AQR Capital](https://en.wikipedia.org/wiki/AQR_Capital) while he was a researcher there from 2007 to 2010.

Pandas is mainly used for [data analysis](https://en.wikipedia.org/wiki/Data_analysis). Pandas allows importing data from various file formats such as [comma-separated values](https://en.wikipedia.org/wiki/Comma-separated_values), [JSON](https://en.wikipedia.org/wiki/JSON), [SQL](https://en.wikipedia.org/wiki/SQL), [Microsoft Excel](https://en.wikipedia.org/wiki/Microsoft_Excel). Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as [data cleaning](https://en.wikipedia.org/wiki/Data_cleaning), and [data wrangling](https://en.wikipedia.org/wiki/Data_wrangling) features.

**4.3.5 Statsmodel**

Statsmodels is a [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) package that allows users to explore data, estimate [statistical models](https://en.wikipedia.org/wiki/Statistical_models), and perform [statistical tests](https://en.wikipedia.org/wiki/Statistical_tests). An extensive list of descriptive statistics, statistical tests, plotting functions, and result statistics are available for different types of data and each estimator. It complements [SciPy](https://en.wikipedia.org/wiki/SciPy)'s stats module.

Statsmodels is part of the Python scientific stack that is oriented towards [data analysis](https://en.wikipedia.org/wiki/Data_analysis), [data science](https://en.wikipedia.org/wiki/Data_science) and [statistics](https://en.wikipedia.org/wiki/Statistics). Statsmodels is built on top of the numerical libraries [NumPy](https://en.wikipedia.org/wiki/NumPy) and SciPy, integrates with [Pandas](https://en.wikipedia.org/wiki/Pandas_(software)) for data handling, and uses Patsy for an [R](https://en.wikipedia.org/wiki/R_(programming_language))-like formula interface. Graphical functions are based on the [Matplotlib](https://en.wikipedia.org/wiki/Matplotlib) library. Statsmodels provides the statistical backend for other Python libraries. Statmodels is [free software](https://en.wikipedia.org/wiki/Free_software) released under the [Modified BSD (3-clause) license](https://en.wikipedia.org/wiki/BSD_licenses).

**4.3.6 Scikit-learn**

Scikit-learn is a [free software](https://en.wikipedia.org/wiki/Free_software) [machine learning](https://en.wikipedia.org/wiki/Machine_learning) [library](https://en.wikipedia.org/wiki/Library_(computing)) for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) [programming language](https://en.wikipedia.org/wiki/Programming_language). It features various [classification](https://en.wikipedia.org/wiki/Statistical_classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis)  and [clustering](https://en.wikipedia.org/wiki/Cluster_analysis) algorithms including [support vector machines](https://en.wikipedia.org/wiki/Support_vector_machine), [random forests](https://en.wikipedia.org/wiki/Random_forests), [gradient boosting](https://en.wikipedia.org/wiki/Gradient_boosting), [*k*-means](https://en.wikipedia.org/wiki/K-means_clustering) and [DBSCAN](https://en.wikipedia.org/wiki/DBSCAN), and is designed to interoperate with the Python numerical and scientific libraries [NumPy](https://en.wikipedia.org/wiki/NumPy) and [SciPy](https://en.wikipedia.org/wiki/SciPy).

**4.3.7 TensorFlow**

TensorFlow is an open-source library developed by Google primarily for deep learning applications. It also supports traditional machine learning. TensorFlow was originally developed for large numerical computations without keeping deep learning in mind. However, it proved to be very useful for deep learning development as well, and therefore Google open-sourced it.

TensorFlow accepts data in the form of multi-dimensional arrays of higher dimensions called tensors. Multi-dimensional arrays are very handy in handling large amounts of data.

TensorFlow works on the basis of data flow graphs that have nodes and edges. As the execution mechanism is in the form of graphs, it is much easier to execute TensorFlow code in a distributed manner across a cluster of computers while using GPUs.

**Why TensorFlow is popular for Deep Learning?**

Deep learning applications are very complicated, with the training process requiring a lot of computation. It takes a long time because of the large data size, and it involves several iterative processes, mathematical calculations, matrix multiplications, and so on. If you perform these activities on a normal Central Processing Unit (CPU), typically it would take much longer.

Graphical Processing Units (GPUs) are popular in the context of games, where you need the screen and image to be of high resolution. GPUs were originally designed for this purpose. However, they are being used for developing deep learning applications as well. One of the major advantages of TensorFlow is that it supports GPUs, as well as CPUs. It also has a faster compilation time than other deep learning libraries, like Keras and Torch.

**4.3.8 Keras**

Keras is a minimalist Python library for deep learning that can run on top of Theano or TensorFlow. It was developed to make implementing deep learning models as fast and easy as possible for research and development. It runs on Python 2.7 or 3.5 and can seamlessly execute on GPUs and CPUs given the underlying frameworks. It is released under the permissive MIT license.

Keras was developed and maintained by [François Chollet](https://www.linkedin.com/in/fchollet), a Google engineer using four guiding principles:

* Modularity: A model can be understood as a sequence or a graph alone. All the concerns of a deep learning model are discrete components that can be combined in arbitrary ways.
* Minimalism: The library provides just enough to achieve an outcome, no frills and maximizing readability.
* Extensibility: New components are intentionally easy to add and use within the framework, intended for researchers to trial and explore new ideas.
* Python: No separate model files with custom file formats. Everything is native Python.

**4.3.9 Matplotlib**

Matplotlib is a [plotting](https://en.wikipedia.org/wiki/Plotter) [library](https://en.wikipedia.org/wiki/Library_(computer_science)) for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) programming language and its numerical mathematics extension [NumPy](https://en.wikipedia.org/wiki/NumPy). It provides an [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) [API](https://en.wikipedia.org/wiki/API) for embedding plots into applications using general-purpose [GUI toolkits](https://en.wikipedia.org/wiki/GUI_toolkit) like [Tkinter](https://en.wikipedia.org/wiki/Tkinter), [wxPython](https://en.wikipedia.org/wiki/WxPython), [Qt](https://en.wikipedia.org/wiki/Qt_(software)), or [GTK](https://en.wikipedia.org/wiki/GTK). There is also a [procedural](https://en.wikipedia.org/wiki/Procedural_programming) "pylab" interface based on a [state machine](https://en.wikipedia.org/wiki/State_machine) (like [OpenGL](https://en.wikipedia.org/wiki/OpenGL)), designed to closely resemble that of [MATLAB](https://en.wikipedia.org/wiki/MATLAB), though its use is discouraged. [SciPy](https://en.wikipedia.org/wiki/SciPy) makes use of Matplotlib.

Matplotlib was originally written by [John D. Hunter](https://en.wikipedia.org/wiki/John_D._Hunter). Since then, it has an active development community and is distributed under a [BSD-style license](https://en.wikipedia.org/wiki/BSD_licenses). Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012 and was further joined by Thomas Caswell.

Matplotlib 2.0.x supports Python versions 2.7 through 3.6. Python 3 support started with Matplotlib 1.2. Matplotlib 1.4 is the last version to support Python 2.6. Matplotlib has pledged not to support Python 2 past 2020 by signing the Python 3 Statement.

**4.3.10 Streamlit**

[Streamlit](https://streamlit.io/) is an open-source Python library that makes it easy to create and share beautiful, custom web apps for machine learning and data science. It is an awesome new tool that allows engineers to quickly build highly interactive web applications around their data, machine learning models, and pretty much anything. The best thing about Streamlit is it doesn't require any knowledge of web development. Install Streamlit using [PIP](https://pip.pypa.io/en/stable/installing/) and run the app:

***Pip install streamlit***

***Streamlit hello***

**4.4 MODULES IMPLEMENTATAION**

**4.4.1 ARIMA**

**What is ARIMA?**

An autoregressive integrated moving average, or ARIMA, is a statistical analysis model that uses [time series data](https://www.investopedia.com/terms/t/timeseries.asp) to either better understand the data set or to predict future trends.

An autoregressive integrated moving average model is a form of [regression analysis](https://www.investopedia.com/terms/r/regression.asp) that gauges the strength of one dependent variable relative to other changing variables. The model's goal is to predict future securities or financial market moves by examining the differences between values in the series instead of through actual values.

An ARIMA model can be understood by outlining each of its components as follows:

* [*Autoregression (AR)*](https://www.investopedia.com/terms/a/autoregressive.asp)refers to a model that shows a changing variable that regresses on its own lagged, or prior, values.
* *Integrated (I)*represents the differencing of raw observations to allow for the time series to become stationary, i.e., data values are replaced by the difference between the data values and the previous values.
* [*Moving average (MA)*](https://www.investopedia.com/terms/m/movingaverage.asp)incorporates the dependency between an observation and a residual error from a moving average model applied to lagged observations.

Each component functions as a parameter with a standard notation. For ARIMA models, a standard notation would be ARIMA with p, d, and q, where integer values substitute for the parameters to indicate the type of ARIMA model used. The parameters can be defined as:

* *p*: the number of lag observations in the model; also known as the lag order.
* *d*: the number of times that the raw observations are differenced; also known as the degree of differencing.
* q: the size of the moving average window; also known as the order of the moving average.

**How ARIMA used for time series forecasting?**

A popular and widely used statistical method for time series forecasting is the ARIMA model. Exponential smoothing and ARIMA models are the two most widely used approaches to time series forecasting and provide complementary approaches to the problem. While exponential smoothing models are based on a description of the trend and seasonality in the data, ARIMA models aim to describe the autocorrelations in the data.

A stationary time series data is one whose properties do not depend on the time, that is why time series with trends, or with seasonality, are not stationary. the trend and seasonality will affect the value of the time series at different times, On the other hand for stationarity it does not matter when you observe it, it should look much the same at any point in time. In general, a stationary time series will have no predictable patterns in the long-term.

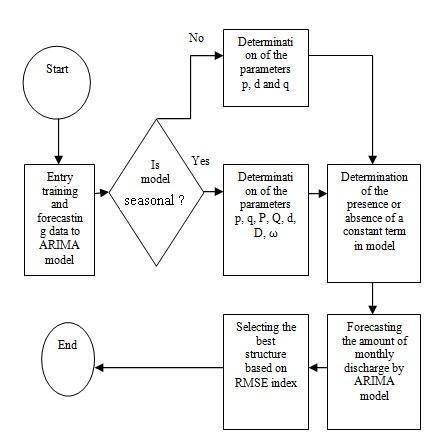
**STEPS**

1. Visualize the Time Series Data

2. Identify if the date is stationary

3. Plot the Correlation and Auto Correlation Charts

4. Construct the ARIMA Model or Seasonal ARIMA based on the data



**Figure 4.1 ARIMA Flow Chart**

**4.4.2 LSTM**

**What is LSTM?**

Long short-term memory (LSTM) is an artificial [recurrent neural network](https://en.wikipedia.org/wiki/Recurrent_neural_network) (RNN) architecture used in the field of [deep learning](https://en.wikipedia.org/wiki/Deep_learning). Unlike standard [feedforward neural networks](https://en.wikipedia.org/wiki/Feedforward_neural_network), LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected [handwriting recognition](https://en.wikipedia.org/wiki/Handwriting_recognition), [speech recognition](https://en.wikipedia.org/wiki/Speech_recognition) and anomaly detection in network traffic or IDSs (intrusion detection systems).

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three *gates* regulate the flow of information into and out of the cell. LSTM networks are well-suited to [classifying](https://en.wikipedia.org/wiki/Classification_in_machine_learning), [processing](https://en.wikipedia.org/wiki/Computer_data_processing) and [making predictions](https://en.wikipedia.org/wiki/Predict) based on [time series](https://en.wikipedia.org/wiki/Time_series) data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the [vanishing gradient problem](https://en.wikipedia.org/wiki/Vanishing_gradient_problem) that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, [hidden Markov models](https://en.wikipedia.org/wiki/Hidden_Markov_models) and other sequence learning methods in numerous applications.

A RNN using LSTM units can be trained in a supervised fashion, on a set of training sequences, using an optimization algorithm, like [gradient descent](https://en.wikipedia.org/wiki/Gradient_descent), combined with [backpropagation through time](https://en.wikipedia.org/wiki/Backpropagation_through_time) to compute the gradients needed during the optimization process, in order to change each weight of the LSTM network in proportion to the derivative of the error (at the output layer of the LSTM network) with respect to corresponding weight.

A problem with using [gradient descent](https://en.wikipedia.org/wiki/Gradient_descent) for standard RNNs is that error gradients [vanish](https://en.wikipedia.org/wiki/Vanishing_gradient_problem) exponentially quickly with the size of the time lag between important events. This is due to if the [spectral radius](https://en.wikipedia.org/wiki/Spectral_radius) of  is smaller than 1. However, with LSTM units, when error values are back-propagated from the output layer, the error remains in the LSTM unit's cell. This "error carousel" continuously feeds error back to each of the LSTM unit's gates, until they learn to cut off the value.

**How LSTM used for time series forecasting?**

LSTM models are able to store information over a period of time. In order words, they have a memory capacity. Remember that LSTM stands for Long Short-Term Memory Model. This characteristic is extremely useful when we deal with Time-Series or Sequential Data.

In Sequence-to-Sequence Learning, an RNN model is trained to map an input sequence to an output sequence. The input and output need not necessarily be of the same length. The seq2seq model contains two RNNs, e.g., LSTMs. They can be treated as an encoder and decoder. The encoder part converts the given input sequence to a fixed-length vector, which acts as a summary of the input sequence.

This fixed-length vector is called the context vector. The context vector is given as input to the decoder and the final encoder state as an initial decoder state to predict the output sequence. Sequence to Sequence learning is used in language translation, speech recognition, time series  
forecasting, etc.

We will use the sequence-to-sequence learning for time series forecasting. We can use this architecture to easily make a multistep forecast. we will add two layers, a repeat vector layer and time distributed dense layer in the architecture.

A repeat vector layer is used to repeat the context vector we get from the encoder to pass it as an input to the decoder. We will repeat it for n-steps (*n* is the no of future steps you want to forecast). The output received from the decoder with respect to each time step is mixed. The time distributed densely will apply a fully connected dense layer on each time step and separates the output for each timestep. The time distributed densely is a wrapper that allows applying a layer to every temporal slice of an input.

We will stack additional layers on the encoder part and the decoder part of the sequence-to-sequence model. By stacking LSTM’s, it may increase the ability of our model to understand more complex representation of our time-series data in hidden layers, by capturing information at different levels.

**STEPS**

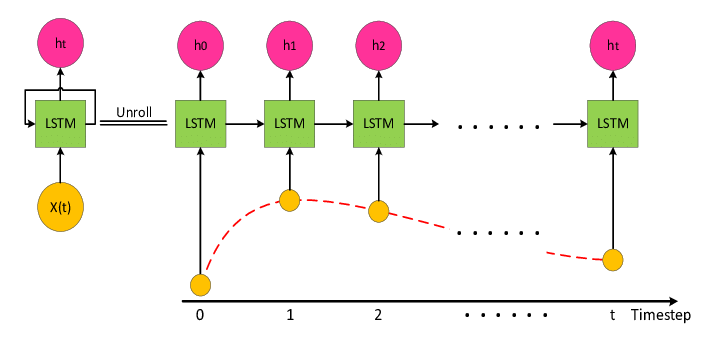
## Define Network

## Compile Network

## Fit Network

## Evaluate Network

## Make Predictions



**Figure 4.2 LSTM Architecture**

**4.4.3 Ensemble ARIMA and LSTM**

Ensemble learning combines the predictions from multiple neural network models to reduce the variance of predictions and reduce generalization error. Techniques for ensemble learning can be grouped by the element that is varied, such as training data, the model, and how predictions are combined.

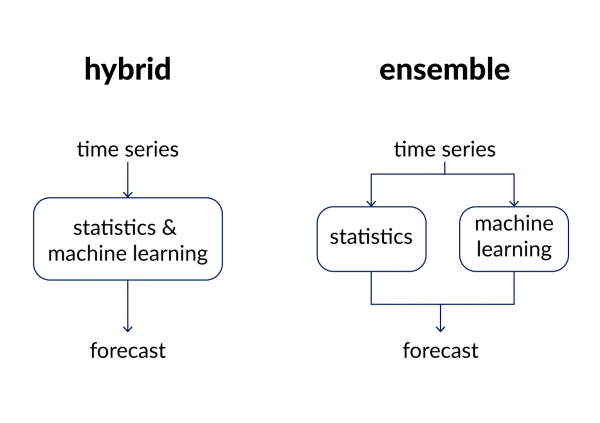
# Why Use Ensemble Learning?

Ensembles are predictive models that combine predictions from two or more other models.

Ensemble learning methods are popular and the go-to technique when the best performance on a predictive modelling project is the most important outcome.

Nevertheless, they are not always the most appropriate technique to use and beginners the field of applied machine learning have the expectation that ensembles or a specific ensemble method are always the best method to use.

Ensembles offer two specific benefits on a predictive modelling project, and it is important to know what these benefits are and how to measure them to ensure that using an ensemble is the right decision.



**Figure 4.3 Ensemble Architecture**

**4.5 NON-FUNCTIONAL REQUIREMENTS**

**FEASIBILITY STUDY:**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

**TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY:**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**CHAPTER 5**

**PERFORMANCE ANALYSIS**

**5.1 TESTING**

**INTRODUCTION**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**DEVELOPING METHODOLOGIES**

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used.

The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

**TYPES OF TESTS**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input: Identified classes of valid input must be accepted.

Invalid Input: Identified classes of invalid input must be rejected.

Functions: Identified functions must be exercised.

Output: Identified classes of application outputs must be exercised.

Systems/Procedures: Interfacing systems or procedures must be invoked.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**Performance Test**

The Performance test ensures that the output is produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

**Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**5.2 EXPERIMENTAL RESULTS**

The performance evaluation for the time series forecasting model is calculated with the underneath methods.

**5.2.1 Mean Forecast Error (MFE)**

Mean forecast error shows the deviation of a forecast from actual demand. This is the mean of the differences per period between a number of period forecasts and the actual demand for the corresponding periods.

**Methods for Calculation**

Mean forecast error can be calculated in three ways, as follows:

**Exponential Smoothing**

ME(i + 1) = ((i) \* (D(i) - F(i)) + (1 - ((i)) \* ME(i)

**Average Forecast Error**

ME(i + 1) = ((D(i) - F(i)) + ....... + (D(i - (n - 1)) - F(i - (n - 1))) / n

**Explanation**

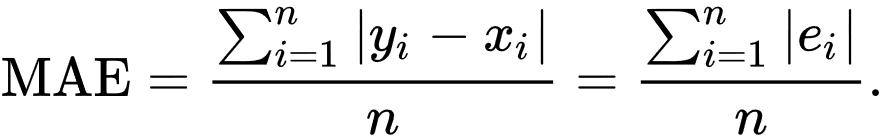
|  |  |  |
| --- | --- | --- |
| ME(i) | = | Mean forecast error for period (i) |
| ((i) | = | Smoothing constant for exponential smoothing in period (i) |
| D(i) | = | Base demand during period (i) |
| F(i) | = | Base forecast for period (i) |
| I | = | Period number |
| N | = | Number of periods included in calculating the mean |

**Table 5.1 Mean Forecast Error Explanation**

Base demand and base forecast represent demand and forecast, respectively, for one period adjusted for seasonal variations and the effects of a varying number of workdays per period.

**5.2.2 Mean Absolute Error (MAE)**

In [statistics](https://en.wikipedia.org/wiki/Statistics), mean absolute error (MAE) is a measure of [errors](https://en.wikipedia.org/wiki/Error_(statistics)) between paired observations expressing the same phenomenon. Examples of *Y* versus *X* include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement. MAE is calculated as:



**Figure 5.1 MAE Formula**

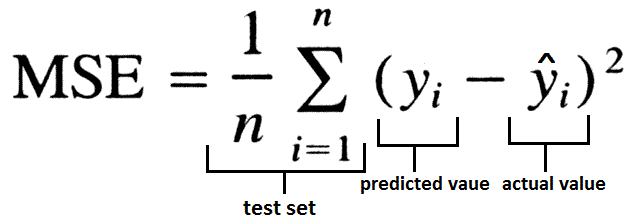
It is thus an arithmetic average of the absolute errors , |ei| = |yi - xi| where yi is the prediction and xi the true value. Note that alternative formulations may include relative frequencies as weight factors. The mean absolute error uses the same scale as the data being measured. This is known as a scale-dependent accuracy measure and therefore cannot be used to make comparisons between series using different scales. The mean absolute error is a common measure of [forecast error](https://en.wikipedia.org/wiki/Forecast_error) in [time series analysis](https://en.wikipedia.org/wiki/Time_series_analysis), sometimes used in confusion with the more standard definition of [mean absolute deviation](https://en.wikipedia.org/wiki/Mean_absolute_deviation). The same confusion exists more generally.

**5.2.3 Mean Squared Error (MSE)**

In [statistics](https://en.wikipedia.org/wiki/Statistics), the mean squared error (MSE) or mean squared deviation (MSD) of an [estimator](https://en.wikipedia.org/wiki/Estimator) (of a procedure for estimating an unobserved quantity) measures the [average](https://en.wikipedia.org/wiki/Expected_value) of the squares of the [errors](https://en.wikipedia.org/wiki/Error_(statistics))—that is, the average squared difference between the estimated values and the actual value. MSE is a [risk function](https://en.wikipedia.org/wiki/Risk_function), corresponding to the [expected value](https://en.wikipedia.org/wiki/Expected_value) of the squared error loss. The fact that MSE is almost always strictly positive (and not zero) is because of [randomness](https://en.wikipedia.org/wiki/Randomness) or because the estimator [does not account for information](https://en.wikipedia.org/wiki/Omitted-variable_bias) that could produce a more accurate estimate.

The MSE is a measure of the quality of an estimator—it is always non-negative, and values closer to zero are better.

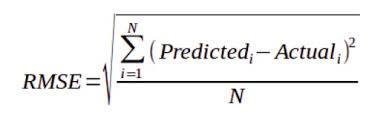
The MSE is the second [moment](https://en.wikipedia.org/wiki/Moment_(mathematics)) (about the origin) of the error, and thus incorporates both the [variance](https://en.wikipedia.org/wiki/Variance) of the estimator (how widely spread the estimates are from one [data sample](https://en.wikipedia.org/wiki/Data_sample) to another) and its [bias](https://en.wikipedia.org/wiki/Bias_of_an_estimator) (how far off the average estimated value is from the true value). For an [unbiased estimator](https://en.wikipedia.org/wiki/Unbiased_estimator), the MSE is the variance of the estimator. Like the variance, MSE has the same units of measurement as the square of the quantity being estimated. In an analogy to [standard deviation](https://en.wikipedia.org/wiki/Standard_deviation), taking the square root of MSE yields the root-mean-square error or [root-mean-square deviation](https://en.wikipedia.org/wiki/Root-mean-square_deviation) (RMSE or RMSD), which has the same units as the quantity being estimated; for an unbiased estimator, the RMSE is the square root of the [variance](https://en.wikipedia.org/wiki/Variance), known as the [standard error](https://en.wikipedia.org/wiki/Standard_error).



**Figure 5.2 MSE Formula**

**5.2.4 Root Mean Squared Error (RMSE)**

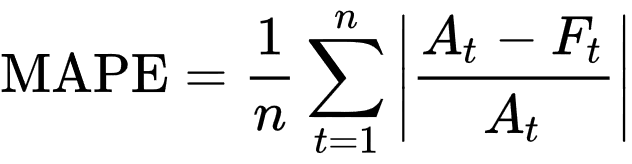
Root Mean Square Error (RMSE) is the [standard deviation](https://www.statisticshowto.com/probability-and-statistics/standard-deviation/) of the [residuals](https://www.statisticshowto.com/residual/) ([prediction errors](https://www.statisticshowto.com/prediction-error-definition/)). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the [line of best fit](https://www.statisticshowto.com/line-of-best-fit/). Root mean square error is commonly used in climatology, forecasting, and [regression analysis](https://www.statisticshowto.com/probability-and-statistics/regression-analysis/) to verify experimental results.



**Figure 5.3 RMSE Formula**

**5.2.5 Mean Absolute Percentage Error (MAPE)**

The mean absolute percentage error (MAPE), also known as mean absolute percentage deviation (MAPD), is a measure of prediction accuracy of a forecasting method in [statistics](https://en.wikipedia.org/wiki/Statistics), for example in [trend estimation](https://en.wikipedia.org/wiki/Trend_estimation), also used as a [loss function](https://en.wikipedia.org/wiki/Loss_function) for regression problems in [machine learning](https://en.wikipedia.org/wiki/Machine_learning). It usually expresses the accuracy as a ratio defined by the formula:



**Figure 5.4 MAPE Formula**

where *At* is the actual value and *Ft* is the forecast value. The MAPE is also sometimes reported as a percentage, which is the above equation multiplied by 100. The difference between *At* and *Ft* is divided by the actual value *At* again. The absolute value in this calculation is summed for every forecasted point in time and divided by the number of fitted points *n*.

**5.3 PERFORMANCE EVALUATION**

Web traffic forecasting using ARIMA and LSTM model’s performance metrics are tabulated below,

|  |  |  |  |
| --- | --- | --- | --- |
| **Methods** | **ARIMA** | **LSTM** | **Hybrid** |
| MFE | 0.000204 | 0.01451 | 0.00563 |
| MAE | 0.188291 | 0.09685 | 0.08114 |
| MSE | 0.054247 | 0.02006 | 0.02240 |
| RMSE | 0.232909 | 0.14164 | 0.11493 |
| MAPE | 15.704892 | 35.6869 | 7.86242 |

**Table 5.2 Error Metrics**

Where, MFE - Mean Forecast Error

MAE - Mean Absolute Error

MSE - Mean Squared Error

RMSE - Root Mean Squared Error

MAPE - Mean Absolute Percentage Error

**5.4 SUMMARY**

As the blend of both the models, from the outset ARIMA model's yield is given as the contribution of the LSTM model. Along these lines the dataset is trained twice. With this mix, the precision level expanded and the rate blunder decreased. MAPE score for the ensembled model is 7.862425, lower than ARIMA (15.704892) and LSTM (35.6869) MAPE score exclusively.

**CHAPTER 6**

**CONCLUSION AND FUTURE WORKS**

**6.1 CONCLUSION**

A web traffic forecasting model is fabricated utilizing ARIMA and LSTM, which proficiently predicts the web traffic in advance, and thereby the server can be allocated based on the requirement and numerous issues identified with web traffic can be addressed.

ARIMA model has a huge advantage in univariate time series forecasting. ARIMA model attempts to describe the trends and seasonality in time series as a function of lagged val- ues(Auto Regressive parameter) and Averages changing over time intervals( Moving Averages).

LSTM networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems.The LSTM model LSTM models are able to store information over a period of time. In order words, they have a memory capacity. Remember that LSTM stands for Long Short-Term Memory Model. This characteristic is extremely useful when we deal with Time-Series or Sequential Data.

To get the best of both worlds we have built a hybrid model consisting of both ARIMA and LSTM. By combining the two models , both linear and nonlinear datas are dealt with ease. hence increasing the efficiency drastically when compared with both the models individually.

**6.2 FUTURE WORK**

We have created a website whereby uploading the previously obtained traffic as a CSV file, the website predicts the web traffic. Future work would be to implement this as a plug-in, by using it, the service provider can get the predicted traffic in an instant.

**APPENDIX**

**CODE SNIPPET**

**ARIMA**

import warnings

warnings.filterwarnings('ignore')

import numpy as np

import pandas as pd

import re

import matplotlib.pyplot as plt

from pandas.plotting import autocorrelation\_plot

from statsmodels.tsa.arima\_model import ARIMA

url = " https://media.githubusercontent.com/media/Olivia-bot-eng/dataset/master/train\_1.csv"

train\_df = pd.read\_csv(url).fillna(0)

train\_df.head()

def find\_language(url):

res = re.search('[a-z][a-z].wikipedia.org',url)

if res:

return res[0][0:2]

return 'na'

train\_df['lang'] = train\_df.Page.map(find\_language)

lang\_sets = {}

lang\_sets['en'] = train\_df[train\_df.lang=='en'].iloc[:,0:-1]

lang\_sets['ja'] = train\_df[train\_df.lang=='ja'].iloc[:,0:-1]

lang\_sets['de'] = train\_df[train\_df.lang=='de'].iloc[:,0:-1]

lang\_sets['na'] = train\_df[train\_df.lang=='na'].iloc[:,0:-1]

lang\_sets['fr'] = train\_df[train\_df.lang=='fr'].iloc[:,0:-1]

lang\_sets['zh'] = train\_df[train\_df.lang=='zh'].iloc[:,0:-1]

lang\_sets['ru'] = train\_df[train\_df.lang=='ru'].iloc[:,0:-1]

lang\_sets['es'] = train\_df[train\_df.lang=='es'].iloc[:,0:-1]

sums = {}

for key in lang\_sets:

sums[key] = lang\_sets[key].iloc[:,1:].sum(axis=0) / lang\_sets[key].shape[0]

days = [r for r in range(sums['en'].shape[0])]

fig = plt.figure(1,figsize=[10,10])

plt.ylabel('Views per Page')

plt.xlabel('Day')

plt.title('Pages in Different Languages')

labels={'en':'English','ja':'Japanese','de':'German',

'na':'Media','fr':'French','zh':'Chinese',

'ru':'Russian','es':'Spanish'

}

for key in sums:

plt.plot(days,sums[key],label = labels[key] )

plt.legend()

plt.show()

from statsmodels.tsa.stattools import pacf

from statsmodels.tsa.stattools import acf

for key in sums:

fig = plt.figure(1,figsize=[10,5])

ax1 = fig.add\_subplot(121)

ax2 = fig.add\_subplot(122)

data = np.array(sums[key])

autocorr = acf(data)

pac = pacf(data)

x = [x for x in range(len(pac))]

ax1.plot(x[1:],autocorr[1:])

ax2.plot(x[1:],pac[1:])

ax1.set\_xlabel('Lag')

ax1.set\_ylabel('Autocorrelation')

ax2.set\_xlabel('Lag')

ax2.set\_ylabel('Partial Autocorrelation')

print(key)

plt.show()

warnings.filterwarnings('ignore')

params = {'en': [4,1,0], 'ja': [7,1,1], 'de': [7,1,1], 'na': [4,1,0], 'fr': [4,1,0], 'zh': [7,1,1], 'ru': [4,1,0], 'es': [7,1,1]}

for key in sums:

print(key)

data = np.array(sums[key])

result = None

arima = ARIMA(data,params[key])

result = arima.fit(disp=False)

pred = result.predict(2,599,typ='levels')

x = [i for i in range(600)]

i=0

plt.plot(x[2:len(data)],data[2:] ,label='Expected')

plt.plot(x[2:],pred,label='Predicted')

plt.xlabel('Days')

plt.ylabel('Views')

plt.legend()

plt.show()

warnings.filterwarnings('ignore')

from sklearn.metrics import mean\_absolute\_error

from sklearn.metrics import mean\_squared\_error

from math import sqrt

from statistics import mean

from sklearn.metrics import accuracy\_score

for key in sums:

data = np.array(sums[key])

result = None

arima = ARIMA(data,params[key])

result = arima.fit(disp=False)

pred = result.predict(2,599,typ='levels')

expected = []

predictions = []

for i in range(len(data)):

predictions.append(pred[i]/1000)

expected.append(data[i]/1000)

bias = []

mae = []

mse = []

rmseArr = []

mape = []

forecast\_errors = [expected[i]-predictions[i] for i in range(len(expected))]

bias.append(sum(forecast\_errors) \* 1.0/len(expected))

mae.append(mean\_absolute\_error(expected, predictions))

mse.append(mean\_squared\_error(expected, predictions))

mse\_err = mean\_squared\_error(expected, predictions)

rmseArr.append(sqrt(mse\_err))

y\_true, y\_pred = np.array(expected), np.array(predictions)

mape.append(np.mean(np.abs((y\_true - y\_pred) / y\_true)) \* 100)

print('Bias: %f' % mean(bias))

print('MAE: %f' % mean(mae))

print('MSE: %f' % mean(mse))

print('RMSE: %f' % mean(rmseArr))

print('MAPE: %f' % mean(mape))

**LSTM**

import numpy as np

import pandas as pd

import re

import matplotlib.pyplot as plt

import seaborn as sns

sns.set(context='notebook',

        style='whitegrid',

        palette='deep',

        font='sans-serif',

        font\_scale=1,

        color\_codes=True,

        rc=None)

from tensorflow import keras

from tensorflow.keras import layers

import tensorflow as tf

from keras.preprocessing.sequence import TimeseriesGenerator

from statsmodels.tsa.seasonal import seasonal\_decompose

import statsmodels.graphics.tsaplots as sgt

import statsmodels.tsa.stattools as sts

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import LSTM

import datetime, os

from keras.preprocessing.sequence import TimeseriesGenerator

url = "https://media.githubusercontent.com/media/Olivia-bot-eng/dataset/master/train\_1.csv"

train\_data = pd.read\_csv(url)

train = train\_data

train.head(3)

%%time

train = train.fillna(method='ffill', downcast='infer')

train.tail(3)

%%time

for cols in train.columns[1:]:

    train[cols] = pd.to\_numeric(train[cols], downcast='integer')

df = pd.DataFrame(train.iloc[:,1:].values.T,

            columns=train.Page.values, index=train.columns[1:])

df.index = pd.to\_datetime(df.index, errors='ignore',

                                            dayfirst=False,

                                            yearfirst=False,

                                            utc=None,

                                            format="%Y/%m/%d",

                                            exact=False,

                                            unit=None,

                                            infer\_datetime\_format=True,

                                            origin='unix',

                                            cache=True)

df.head(3)

wikipedia = (df.filter(like='wikipedia'))

wikipedia

wikipedia.iloc[:,0:10].plot(figsize=(20,10))

plt.show()

def get\_language(page):

    res = re.search('[a-z][a-z].wikipedia.org',page)

    if res:

        return res[0][0:2]

    return 'other'

(wikipedia.columns.map(get\_language)).unique()

len((wikipedia.columns.map(get\_language)).unique())

languages = list((wikipedia.columns.map(get\_language)).unique())

languages.remove('other')

languages

for lang in (languages):

    locals()['lang\_'+str(lang)] = wikipedia.loc[:, wikipedia.columns.str.contains('\_'+str(lang)+'.wiki')]

for lang in (languages):

    locals()['hits\_'+str(lang)] = np.array(locals()['lang\_'+str(lang)].iloc[:,:].sum(axis=1))

for lang in (languages):

    print((locals()['hits\_'+str(lang)]).shape)

keys = languages

values = ['Chinese', 'French', 'English', 'Russian', 'German', 'Japanese', 'Spanish']

d = dict(zip(keys,values))

index = wikipedia.index

hits = pd.DataFrame(index=index, columns=list(d.values()))

hits = hits.fillna(0)

for key, value in d.items():

    hits[value] = locals()['hits\_'+str(key)]

hits.plot(figsize=(25,8), title ='Hits on Wikipedia pages per Language', fontsize=15)

plt.legend(loc='upper left')

plt.show()

plt.rcParams["figure.dpi"] = 100

hits.iloc[:,0:1].plot(figsize=(20,4))

sgt.plot\_acf(np.array(hits.iloc[:,0:1]),

            ax=None,

            lags=None,

            alpha=0.05,

            use\_vlines=True,

            unbiased=False,

            fft=False,

            title='Autocorrelation',

            zero=False,  # Not including the 1st term as its acf w.r.t. itself will always be 1.

            vlines\_kwargs=None)

plt.show()

plt.rcParams["figure.dpi"] = 100

hits.iloc[:,0:1].plot(figsize=(20,4))

sgt.plot\_pacf(np.array(hits.iloc[:,0:1]),

            ax=None,

            lags=None,

            alpha=0.05,

            method='ols',

            use\_vlines=True,

            title='Partial Autocorrelation',

            zero=False,    # Not including the 1st term as its pacf w.r.t. itself will always be 1.

            vlines\_kwargs=None)

plt.show()

brk = 0.8

data\_split = int(len(hits)\*brk)

data\_split

X, y = hits.iloc[:data\_split,:], hits.iloc[data\_split:,:]

scaler = MinMaxScaler()

scaler.fit(X)

scaled\_X = scaler.transform(X)

scaled\_y = scaler.transform(y)

X\_df = (pd.DataFrame(scaled\_X))

y\_df = (pd.DataFrame(scaled\_y))

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(20,8), dpi=100)

plt.suptitle('Train-Test Split', fontsize=20)

X\_df.plot(ax=axes[0], title='Train Data')

y\_df.plot(ax=axes[1], title='Test Data')

plt.show()

pd.DataFrame(scaled\_y[3:13,:]).plot(figsize=(15,5), title='Periodicity')

plt.show()

print(scaled\_X.shape)

print(scaled\_y.shape)

print('No. of features = '+str(scaled\_X.shape[1]))

print('No. of train instances = '+str(scaled\_X.shape[0]))

print('No. of test instances = '+str(scaled\_y.shape[0]))

length = 7

batch = 1

n\_features = scaled\_X.shape[1]

n\_features

generator = TimeseriesGenerator(data = scaled\_X,

                                targets = scaled\_X,

                                length = length,

                                sampling\_rate=1,

                                stride=1,

                                start\_index=0,

                                end\_index=None,

                                shuffle=False,

                                reverse=False,

                                batch\_size=batch)

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import LSTM

import datetime, os

model = Sequential(layers=None, name="LSTM\_Model")

model.add(LSTM( units = 400,

                activation='tanh',

                input\_shape=( length, n\_features),

                recurrent\_activation='sigmoid',

                use\_bias=True,

                kernel\_initializer='glorot\_uniform',

                recurrent\_initializer='orthogonal',

                bias\_initializer='zeros',

                unit\_forget\_bias=True,

                kernel\_regularizer=None,

                recurrent\_regularizer=None,

                bias\_regularizer=None,

                activity\_regularizer=None,

                kernel\_constraint=None,

                recurrent\_constraint=None,

                bias\_constraint=None,

                dropout=0.0,

                recurrent\_dropout=0.0,

                implementation=2,

                return\_sequences=True,

                return\_state=False,

                go\_backwards=False,

                stateful=False,

                time\_major=False,

                unroll=False

            ) )

model.add(LSTM(units = 500, return\_sequences=True))

model.add(LSTM(units = 500, return\_sequences=False))

model.add(Dense(700, activation="relu", name="layer1"))

model.add(Dense(100, activation="relu", name="layer2"))

model.add(Dense( units = n\_features,

                activation='relu',

                use\_bias=True,

                kernel\_initializer='glorot\_uniform',

                bias\_initializer='zeros',

                kernel\_regularizer=None,

                bias\_regularizer=None,

                activity\_regularizer=None,

                kernel\_constraint=None,

                bias\_constraint=None))

model.compile(optimizer='adam', loss='mse')

from sklearn.metrics import mean\_absolute\_error

from sklearn.metrics import mean\_squared\_error

from math import sqrt

from statistics import mean

from sklearn.metrics import accuracy\_score

plt.figure(dpi=100)

plt.plot(np.linspace(0,t\_l,t\_l), scaled\_y[:,0:1] , label='True Values',c='g')

plt.plot(np.linspace(0,t\_l,t\_l), np.array(test\_predictions)[:,0:1], label='Predicted Values',c='r')

plt.title(hits.columns[0], fontsize=20)

plt.legend()

plt.show()

expected = np.reshape(scaled\_y, (np.product(scaled\_y.shape),))

predictions = []

for i in range(len(test\_predictions)):

  for j in range(len(test\_predictions[i])):

    predictions.append(test\_predictions[i][j])

bias = []

mae = []

mse = []

rmseArr = []

mape = []

forecast\_errors = [expected[i]-predictions[i] for i in range(len(expected))]

bias.append(sum(forecast\_errors) \* 1.0/len(expected))

mae.append(mean\_absolute\_error(expected, predictions))

mse.append(mean\_squared\_error(expected, predictions))

mse\_err = mean\_squared\_error(expected, predictions)

rmseArr.append(sqrt(mse\_err))

y\_true, y\_pred = np.array(expected), np.array(predictions)

mape.append(np.mean(np.abs((y\_true - y\_pred) / y\_true)) \* 100)

print('Bias: %f' % mean(bias))

print('MAE: %f' % mean(mae))

print('MSE: %f' % mean(mse))

print('RMSE: %f' % mean(rmseArr))

print('MAPE: %f' % mean(mape))

**Ensemble Model**

import streamlit as st

import pandas as pd

import datetime

import numpy as np

import re

import matplotlib.pyplot as plt

import math

import time

import warnings

import base64

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import LSTM

from statsmodels.tsa.arima\_model import ARIMA

from statsmodels.tsa.stattools import pacf

from statsmodels.tsa.stattools import acf

from sklearn.metrics import mean\_absolute\_error

from sklearn.metrics import mean\_squared\_error

from math import sqrt

warnings.filterwarnings('ignore')

image = '.\\undraw\_fast\_loading\_0lbh.png'

st.title("Web Traffic Forecasting")

st.header("Problem Statement")

st.write("Web traffic congestion is a scenario faced by network appliction frequently.")

st.write("Web traffic congestion is a phenomenum where the number of requests to be fulfilled by the server increases beyond the resource allocation.")

st.image(image, caption=None, width=300, use\_column\_width=None)

st.header("Solution")

st.write("A prediction model which analyzes the web traffic pattern of the target server and thus the server resources are allocated in advance to handle the server load.")

path\_name = st.sidebar.text\_input("Enter file path for report")

st.sidebar.header("Visualisation Settings")

uploaded\_file = st.sidebar.file\_uploader(label="Upload your web traffic dataset(.csv)", type=['csv','xslx'])

st.sidebar.image("undraw\_Data\_trends\_re\_2cdy.png",caption=None, width=300, use\_column\_width=None)

global df

if uploaded\_file is not None:

try:

df = pd.read\_csv(uploaded\_file).fillna(0)

list\_of\_column\_names = list(df.columns)

list\_of\_column\_names.remove('Page')

j = 0

for i in range(1,len(list\_of\_column\_names)):

if i % 100 == 0:

list\_of\_column\_names[j] = list\_of\_column\_names[i]

j = j + 1

x = [datetime.date(2015, 7, 1)] \* 550

def find\_language(url):

res = re.search('[a-z][a-z].wikipedia.org', url)

if res:

return res[0][0:2]

return 'na'

df['lang'] = df.Page.map(find\_language)

lang\_sets = {}

lang\_sets['en'] = df[df.lang == 'en'].iloc[:, 0:-1]

lang\_sets['ja'] = df[df.lang == 'ja'].iloc[:, 0:-1]

lang\_sets['de'] = df[df.lang == 'de'].iloc[:, 0:-1]

lang\_sets['na'] = df[df.lang == 'na'].iloc[:, 0:-1]

lang\_sets['fr'] = df[df.lang == 'fr'].iloc[:, 0:-1]

lang\_sets['zh'] = df[df.lang == 'zh'].iloc[:, 0:-1]

lang\_sets['ru'] = df[df.lang == 'ru'].iloc[:, 0:-1]

lang\_sets['es'] = df[df.lang == 'es'].iloc[:, 0:-1]

sums = {}

for key in lang\_sets:

sums[key] = lang\_sets[key].iloc[:, 1:].sum(axis=0) / lang\_sets[key].shape[0]

days = [r for r in range(sums['en'].shape[0])]

fig = plt.figure(1, figsize=[10, 10])

fig, ax = plt.subplots()

plt.ylabel('Views per Page')

plt.xlabel('Day')

plt.title('Pages in Different Languages')

labels = {'en': 'English', 'ja': 'Japanese', 'de': 'German',

'na': 'Media', 'fr': 'French', 'zh': 'Chinese',

'ru': 'Russian', 'es': 'Spanish'

}

for key in sums:

plt.plot(days, sums[key], label=labels[key])

for key in sums:

fig = plt.figure(1, figsize=[10, 5])

ax1 = fig.add\_subplot(121)

ax2 = fig.add\_subplot(122)

data = np.array(sums[key])

autocorr = acf(data)

pac = pacf(data)

x = [x for x in range(len(pac))]

ax1.plot(x[1:], autocorr[1:])

ax2.plot(x[1:], pac[1:])

ax1.set\_xlabel('Lag')

ax1.set\_ylabel('Autocorrelation')

ax2.set\_xlabel('Lag')

ax2.set\_ylabel('Partial Autocorrelation')

warnings.filterwarnings('ignore')

if path\_name:

file\_name = path\_name + "\\hits.txt"

else:

file\_name = "..\\hits.txt"

f = open(file\_name, "w")

list\_of\_column\_names = list(df.columns)

list\_of\_column\_names.remove('Page')

params = {'en': [4, 1, 0], 'ja': [7, 1, 1], 'de': [7, 1, 1], 'na': [4, 1, 0], 'fr': [4, 1, 0], 'zh': [7, 1, 1],

'ru': [4, 1, 0], 'es': [7, 1, 1]}

lang\_obj = {'en': 'English', 'ja': 'Japanese', 'de': 'German', 'na': 'Media', 'fr': 'French', 'zh': 'Chinese',

'ru': 'Russian', 'es': 'Spanish'}

for key in sums:

for keys in lang\_obj:

if (keys == key):

f.write(lang\_obj[keys])

f.write("\n")

data = np.array(sums[key])

f.write("Date Expected Predicted Error")

f.write("\n")

result = None

arima = ARIMA(data, params[key])

result = arima.fit(disp=False)

pred = result.predict(2, 599, typ='levels')

x = [i for i in range(600)]

i = 0

for i in range(len(data)):

f.write(str(list\_of\_column\_names[i]) + " " + str(math.ceil(data[i])) + " " + str(

math.ceil(pred[i])) + " " + str(

np.sqrt(np.mean((math.ceil(data[i]) - math.ceil(pred[i])) \*\* 2))))

f.write("\n")

warnings.filterwarnings('ignore')

train\_df = df.drop('Page', axis=1)

global chart

bias = []

mae = []

mse = []

rmseArr = []

mape = []

for key in sums:

row = [0] \* sums[key].shape[0]

for i in range(sums[key].shape[0]):

row[i] = sums[key][i]

X = row[0:549]

y = row[1:550]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=0)

sc = MinMaxScaler()

X\_train = np.reshape(X\_train, (-1, 1))

y\_train = np.reshape(y\_train, (-1, 1))

X\_train = sc.fit\_transform(X\_train)

y\_train = sc.fit\_transform(y\_train)

X\_train = np.reshape(X\_train, (384, 1, 1))

regressor = Sequential()

regressor.add(LSTM(units=8, activation='relu', input\_shape=(None, 1)))

regressor.add(Dense(units=1))

regressor.compile(optimizer='rmsprop', loss='mean\_squared\_error')

regressor.fit(X\_train, y\_train, batch\_size=10, epochs=100, verbose=0)

inputs = X

inputs = np.reshape(inputs, (-1, 1))

inputs = sc.transform(inputs)

inputs = np.reshape(inputs, (549, 1, 1))

y\_pred = regressor.predict(inputs)

y\_pred = sc.inverse\_transform(y\_pred)

b = np.reshape(y\_pred, (np.product(y\_pred.shape),))

expected = []

predictions = []

for i in range(len(y)):

predictions.append(b[i]/1000)

expected.append(y[i]/1000)

forecast\_errors = [expected[i]-predictions[i] for i in range(len(expected))]

bias.append(sum(forecast\_errors) \* 1.0/len(expected))

mae.append(mean\_absolute\_error(expected, predictions))

mse.append(mean\_squared\_error(expected, predictions))

mse\_err = mean\_squared\_error(expected, predictions)

rmseArr.append(sqrt(mse\_err))

y\_true, y\_pred = np.array(expected), np.array(predictions)

mape.append(np.mean(np.abs((y\_true - y\_pred) / y\_true)) \* 100)

d = {'Expected': y, 'Predicted': b}

chart = pd.DataFrame(data=d)

for keys in lang\_obj:

if keys == key:

st.write("Web Traffic Prediction for " + lang\_obj[keys] + " pages")

progress\_bar = st.sidebar.progress(0)

chart\_new = st.line\_chart(chart[:25])

j = 12

for i in range(0, len(chart), 25):

progress\_bar.progress(j + 4)

end\_index = i + 25

new\_rows = chart[i:end\_index]

chart\_new.add\_rows(new\_rows)

time.sleep(0.4)

j = j + 4

progress\_bar.empty()

print('Bias: %f' % mean(bias))

print('MAE: %f' % mean(mae))

print('MSE: %f' % mean(mse))

print('RMSE: %f' % mean(rmseArr))

print('MAPE: %f' % mean(mape))

except Exception as e:

print(e)

df = pd.read\_excel(uploaded\_file)