A Project Phase-1 Report on

ADVANCED HOUSEHOLD POWER CONSUMPTION PREDICTION USING LSTM(LONG SHORT TERM MEMORY)

Submitted to the Dept. of Information Technology, SNIST
in the partial fulfillment of the academic requirements for the award of

B.Tech (Information Technology)

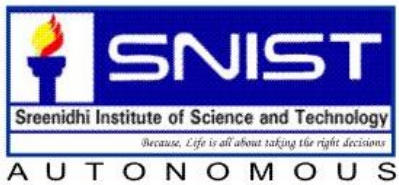
under JNTUH

by

Hanumanth Rao.S(17311A12D7)
M.Bharath Chandra (17311A12D8)
G.Srikar (17311A12D0)

under the guidance of

Dr. K. Kranthi Kumar
Associate professor



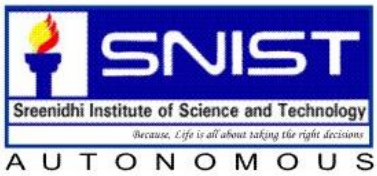
Department of Information Technology
School of Computer Science and Informatics
Sreenidhi Institute of Science and Technology (An Autonomous Institution)
Yamnampet, Ghatkesar Mandal, R. R. Dist., Hyderabad – 501301

affiliated to

Jawaharlal Nehru Technological University Hyderabad

Hyderabad – 500085

2020–21
Department of Information Technology
School of Computer Science and Informatics
Sreenidhi Institute of Science and Technology



Certificate

This is to certify that the Major-Project report on "Advanced Household Power Consumption Prediction using LSTM " is a bonafide work carried out by Hanumanth Rao.S(17311A12D7),M.Bharat Chandra 17311A12D8), Srikar.G17311A 12D0) in the partial fulfillment for the award of B.Tech degree in Information Technology, Sreenidhi Institute of Science and Technology, Hyderabad, affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH), Hyderabad under our guidance and supervision.

The results embodied in the Phase-1 Project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Internal guide	**Project Coordinator**	**Head of the Department**
(Dr. K. Kranthi Kumar)	**(Dr.K.Kranthi Kumar)**	**((Dr. V. V. S. S. S.Balaram)**

Date: **External Examiner**

DECLARATION

We**, Hanumanth Rao. S (17311A12D7), Bharath Chandra.M (17311A12D8) and Srikar. G (17311A12D0),** students of **Sreenidhi Institute of Science and Technology, Yamnampet, Ghatkesar,** studying IV year Istsemester**, Information Technology** solemnly declare that the Major-project work, titled **"Advanced Household Power Consumption Prediction using LSTM**" is submitted to **Sreenidhi Institute of Science and Technology for partial fulfillment** for the award of degree of Bachelor of technology in **Information Technology.**

It is declared to the best of our knowledge that the work reported does not form part of any dissertation submitted to any other University or Institute for award of any degree.

Acknowledgements

We would like to express our immense gratitude and sincere thanks to **Dr. K. Kranthi Kumar**, **Associate Professor** in Information Technology for his guidance, valuable suggestions and encouragement in completing the Major-Project work within the stipulated time.

We would like to express our sincere thanks to **Dr. P. Narasimha Reddy**, Executive Director, **Dr. T. Ch. Siva Reddy**, Principal, **Dr. V. V. S. S. S. Balaram**, Professor & Head of the Department of Information Technology, **Dr. K.Kranthi Kumar**, Associate Professor & Major-Project Work Coordinator of the Department of Information Technology, Sreenidhi Institute of Science and Technology (An Autonomous Institution), Hyderabad for permitting us to do our Group-Project work.

Finally, we would also like to thank the people who have directly or indirectly helped us and parents and friends for their cooperation in completing the Major-Project work.

Hanumanth Rao. S(17311A12D7)
Bharath Chandra.M(17311A12D8)
Srikar.G(17311A12D0)

LIST OF FIGURES

Abstract

Introduction

Electricity power consumption prediction has gained a lot of importance nowadays in the modern electrical power management systems. An authentic prediction of electrical power consumption represents a starting point in policy development and improvement of energy production and distribution. Given the development of electricity meters and the wide adoption of electricity generation technology like solar panels, huge amount of electricity usage data is available.

In our proposed system we implement LSTM techniques to prepare a machine learning model to forecast the electricity consumption in a household so that when a power outage occurs, then the electricity generating companies(non renewable resources) will be in a position to supply the electricity needed by the households when a power outage occurs suddenly. The time series data in our project is the individual household electric power consumption taken from the UCI Machine Learning Repository and this data represents a multivariate time series of power-related variables that in turn could be used to model and forecast future electricity consumption. To explore and understand the dataset we will use various plots and histograms for the data distribution.

Keywords: LSTM(Long short Term Memory).

Index

Chapter-1

INTRODUCTION

Power outage accidents will cause huge economic loss hence it is very important to predict Electricity power consumption prediction has gained a lot of importance nowadays in the modern electrical power management systems. An authentic prediction of electrical power consumption represents a starting point in policy development and improvement of energy production and distribution. Given the development of smart electricity meters and the wide adoption of electricity generation technology like solar panels, huge amount of electricity usage data is available.

1.1 Scope

This System of Predicting the advanced one week Household consumption has a lot of importance to both the organizations and also to the people. Users can minimize their costs and can effectively use their power consumption so that the burden of charges on them decreases which helps in the improving the economic life of the user. Organizations can identify and also they can minimize their wastage of power by predicting it in advance, It also increases the performance and quality of services provided by the organizations to their users. It also helps the organization to view and observe the different segments in which the users are consuming the power.

1.2 Existing System

N. Fume[1] from Regression analysis for prediction of residential energy consumption in a study predicted the residential energy conducted for advanced power consumption prediction and have used general linear regression. This model design is straightforward and eliminates unnecessary variables to improve predictive performance stability. However, the correlation between independent variables used for prediction can lead to multi collinearity problems.

A. Bogomolov [2] attempted to predict energy Consumption using Random forest methods which lead to Over fitting when the complicated correlation of variables becomes complicated or

amount of data increases. When Over fitting occurs its difficult to predict long-term consumption.

1.3 Proposed System

The proposed system performs the task by using the concept of Long Short Term Memory for Household Power Consumption prediction. LSTM(Long Short Term Memory) provides a solution by preserving long term memory by consolidating memory units that can update previous hidden state.

LSTM(Long Short Term Memory) overcomes the problem of Over fitting and Multi Collinearity unlike other models. We calculate the Standard deviation and RMS(Root Mean Square) error of the predicted values of electric power consumption. Thus we are able to predict the electric power consumption values effectively.

Data set description

The data set that we used in this project was collected between December 2006 and November2010 and observations of power consumption within the household was collected every minute. It is a multivariate series that contains the following fields they are

Global active power

It is defined as the total active power consumed by the household. It is measured Kilowatts.

Global reactive power:

It is defined as the total reactive power consumed by the household. It is measured in Kilowatts.

Voltage:

Average voltage that was used by each appliances in the home.

This data represents a multivariate time series of power-related variables that in turn could be used to model and even forecast future electricity consumption. we are using time series data as the data is being recorded every minute so these time series predictions play a major role in the machine learning algorithms we use to predict the required output.

In this project we are using Recurrent Neural Networks (RNN) to predict the data which is Deep Learning Framework. This RNN is mainly based upon the concept of LSTM (Long Short Term Memory).

Recurrent Neural Networks (RNN):

Recurrent Neural Network is mainly used for the sequential data which is mostly used by Apple's Siri and also used in Goggle Search. Recurrent Neural Networks are a type of neural network where the output from previous step are fed as input to the current step and thereby solved. It is the first algorithm that remembers its input, due to an internal memory, which makes it perfectly suited for machine learning problems that involve sequential data. It is one of the algorithms behind the scenes of the amazing achievements seen in deep learning over the past few years.

Like many other deep learning algorithms, recurrent neural networks are very old. They were initially created in the 1980's, In recent years have we seen their true potential. The increase in computational power along with the massive amounts of data that we now have to work with, and the invention of long short-term memory (LSTM) in the 1990s, which plays a major role in the RNN algorithm.

Because of their internal memory, RNN's can remember important things about the input they received, which allows them to be very precise in predicting what's coming next. This is why they're the preferred algorithm for sequential data like time series, speech, text, financial data, audio, video, weather and much more. Recurrent neural networks can form a much deeper understanding of a sequence and its context compared to other algorithms.

Advantages of RNN

1. An RNN remembers each and every information through time, Its very useful in the case of time series prediction only as it has ability to remember previous inputs as well.

2. Recurrent neural network can be even used with the convolutional layers to extend the effective pixel neighborhood.

Disadvantages of RNN

1. Gradient vanishing and exploding problems are there.

2. Training an RNN is a really a hard task.

3. It cannot process too long sequences by using tanh or relu as an activation function.

RNN VS FEED-FORWARD NEURAL NETWORKS

In feed-forward neural network, inputs are fed to the network and transformed into an output. That is when we feed examples, then labels are outputs. They can be useful for classification. For example, when we give an example, it may classify the image accordingly. In a feed-forward neural network, predictions does not depend on the predictions of the previous step. For example, if the network classifies a set of images and the first image is classified as a ship, then the next image classification may not be changed by the former classification. If the second image is a bus, then it is classified as a bus. But the situation is different in recurrent neural networks. The decision the recurrent neural network gets at this moment depends on the decision which the net got at the previous moment.

Therefore, the current output of a recurrent neural network depends on both the previous output and the current input. In a feed-forward neural network, the information only moves in one direction — from the input layer, through the hidden layers, to the output layer. The information moves straight through the network and never touches a node twice. Feed-forward neural networks have no memory of the input they receive and are bad at predicting what's coming next. Because a feed-forward network only considers the current input, it has no notion of order in time. It simply can't remember anything about what happened in the past except its training but In a RNN the information cycles through a loop. When it makes a decision, it considers the current input and also what it has learned from the inputs it received previously.

Problems of Standard RNN

There are two major obstacles RNN have to deal with. A gradient is a partial derivative with respect to its inputs. a gradient measures how much the output of a function changes if you change the inputs a little bit. The higher the gradient, the steeper the slope and the faster a model can learn. But if the slope is zero, the model stops learning. A gradient simply measures the change in all weights with regard to the change in error.

EXPLODING GRADIENTS

Exploding gradients are when the algorithm, without much reason, assigns a stupidly high importance to the weights. This problem can be easily solved by truncating gradients. In machine Learning, the exploding gradient problem is an issue found in training artificial with gradient-based learning methods and back propagation. An artificial neural network is a learning algorithm, also called neural network or neural net, That uses a network of functions to understand and translate data input into a specific output. This type of learning algorithm is designed to mimic the way neurons function in the human brain. Exploding gradients are a problem when large error gradients accumulate and result in very large updates to neural network model weights during training.

Exploding gradients can cause problems in the training of artificial neural networks. When there are exploding gradients, an unstable network can result and the learning cannot be completed. The values of the weights can also become so large as to overflow and result in something called NaN values. NaN values, which stands for not a number, are values that represent an undefined or unrepresentable values. It is useful to know how to identify exploding gradients in order to correct the training.

VANISHING GRADIENTS

Vanishing gradients occur when the values of a gradient are too small and the model stops learning or takes way too long as a result. This was a major problem in the 1990s and much harder to solve than the exploding gradients. Fortunately, it was solved through the concept of LSTM by Sepp Hochreiter and Juergen Schmidhuber.

The vanishing gradient problem is an issue that sometimes arises when training machine learning algorithms through gradient descent. This most often occurs in neural networks that have several neuronal layers such as in a deep learning system, but also occurs in recurrent neural networks. The key point is that the calculated partial derivatives used to compute the gradient as one goes deeper into the network. Since the gradients control how much the network learns during training, if the gradients are very small or zero, then little to no training can take place, leading to poor predictive performance.

LONG SHORT-TERM MEMORY (LSTM)

Long short-term memory networks are an extension for recurrent neural networks, which basically extends the memory. Therefore it is well suited to learn from important experiences that have very long time lags in between. The units of an LSTM are used as building units for the layers of a RNN, often called an LSTM network.

LSTM enables RNN to remember inputs over a long period of time. This is because LSTMs contain information in a memory, much like the memory of a computer. The LSTM can read, write and delete information from its memory. Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This is a behavior required in complex problem domains like machine translation, speech recognition, and more. LSTMs are a complex area of deep learning. It can be hard to get your hands around what LSTMs are, and how terms like bidirectional and sequence-to-sequence relate to the field. This memory can be observed as a gated cell, with gated meaning the cell decides whether or not to store or delete information, It is based on importance it assigns to the information. The assigning of importance happens through weights, which are also learned by the algorithm. LSTMs on the other hand, make small modifications to the information by

multiplications and additions. With LSTMs, the information flows through a mechanism known as cell states. This way, LSTMs can selectively remember or forget things. The information at a particular cell state has three different dependencies.

In LSTM you have three gates they are input, forget and output gate. These gates determine whether or not to let new input in input gate, delete the information because it isn't important in forget gate, or let it impact the output at the current time step in output gate.

Why use a Deep Learning Algorithm?

With the data volume increasing enormously day by day we shouldn't restrict ourselves to only the standard ML algorithms. Deep learning algorithms also help us to handle large volumes of knowledge and without leaving the key insights and by tuning the model within the proper way gives us the maximal yield i.e in our cause maximum accuracy. The model also determines if our prediction is best or worse from its own neural architecture.

Tensor flow

Tensor Flow is an interface for expressing machine learning algorithms and an implementation for executing such algorithms. A computation expressed using Tensor Flow can be executed with little or no change on a wide variety of heterogeneous systems, ranging from mobile devices such as phones and tablets up to large-scale distributed systems of hundreds of machines and thousands of computational devices such as GPU cards. The system is flexible and can be used to express a wide variety of algorithms, including training and inference algorithms for deep neural network models, and it has been used for conducting research and for deploying machine learning systems into production across more than a dozen areas of computer science and other fields, including speech recognition, computer vision, robotics, information retrieval, natural language processing, geographic information extraction, and computational drug discovery. The Tensor Flow API and a reference implementation were released as an open-source package under the Apache 2.0 license in November, 2015 and are available at www.tensorflow.org.TensorFlow is a machine learning system that operates at large scale and in heterogeneous environments. Tensor Flow uses dataflow graphs to represent computation, shared

state, and the operations that mutate that state. It maps the nodes of a dataflow graph across many machines in a cluster, and within a machine across multiple computational devices, including multi core CPUs, general purpose GPUs, and custom-designed ASICs known as Tensor Processing Units (TPUs). This architecture gives flexibility to the application developer: whereas in previous "parameter server" designs the management of shared state is built into the system, Tensor Flow enables developers to experiment with novel optimizations and training algorithms. Tensor Flow supports a variety of applications, with a focus on training and inference on deep neural networks.

Tensor Flow is an open-source software library. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

Sequential Model

Sequence models are the machine learning models that input or output sequences of data. Sequential data includes text streams, audio clips, video clips, time-series data and etc. Recurrent Neural Networks (RNNs) is a popular algorithm used in sequence models. The Sequential model API is a way of creating deep learning models where an instance of the Sequential class is created and model layers are created and added to it.

Dense Layer

Dense layer is the regular deeply connected neural network layer. It is most common and frequently used layer. Dense layer does the below operation on the input and return the output. The "Deep" in deep-learning comes from the notion of increased complexity resulting by stacking several consecutive (hidden) non-linear layers. Dense layers add an interesting non-linearity property, thus they can model any mathematical function.

Scikit Learn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python. It was originally called scikits.learn and was initially developed by David Cournapeau as a Google summer of code project in 2007. Later, in 2010, Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, and Vincent Michel, from FIRCA (French Institute for Research in Computer Science and Automation), took this project at another level and made the first public release (v0.1 beta) on 1st Feb. 2010.

Scikit-learn is a Python module integrating a wide range of state-of-the-art machine learning algorithms for medium-scale supervised and unsupervised problems. This package focuses on bringing machine learning to non-specialists using a general-purpose high-level language. Emphasis is put on ease of use, performance, documentation, and API consistency. It has minimal dependencies and is distributed under the simplified BSD license, encouraging its use in both academic and commercial settingsScikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.

Scikit-learn library is focused on modeling the data. Some of the most popular groups of modelsprovided by Sklearn are as follows –

Supervised Learning algorithms − Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are the part of scikit-learn.

Unsupervised Learning algorithms − On the other hand, it also has all the popular unsupervised learning algorithms from clustering, factor analysis, PCA (Principal Component Analysis) to unsupervised neural networks.

Clustering − This model is used for grouping unlabeled data.

Cross Validation − It is used to check the accuracy of supervised models on unseen data.

Dimensionality Reduction − It is used for reducing the number of attributes in data which can be further used for summarisation, Visualisation and feature selection.

Ensemble methods − As name suggest, it is used for combining the predictions of multiple supervised models.

Feature extraction − It is used to extract the features from data to define the attributes in image and text data.

Feature selection − It is used to identify useful attributes to create supervised models.

Open Source − It is open source library and also commercially usable under BSD license

Mean Squared Error

Mean Squared Error is a model evaluation metric often used with regression models. The mean squared error of a model with respect to a test set is the mean of the squared prediction errors over all instances in the test set. The prediction error is the difference between the true value and the predicted value for an instance. 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 (about the origin) of the error, and thus incorporates both the variance of the estimator (how widely spread the estimates are from one data sample to another) and its bias (how far off the average estimated value is from the true value). For an 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. The MSE either assesses the quality of a predictor (i.e., a function mapping arbitrary inputs to a sample of values of some random variable), or of an estimator. The definition of an MSE differs according to whether one is describing a predictor or an estimator.

MinMaxScaler

MinMaxScaler scales all the data features in the range [0, 1] or else in the range [-1, 1] if there are negative values in the dataset. This scaling compresses all the inliers in the narrow range [0, 0.005].In the presence of outliers, Standard Scaler does not guarantee balanced feature scales, due to the influence of the outliers while computing the empirical mean and standard deviation. This

leads to the shrinkage in the range of the feature values. The main idea behind normalization/standardization is always the same. Variables that are measured at different scales do not contribute equally to the model fitting & model learned function and might end up creating a bias. Thus, to deal with this potential problem feature-wise normalization such as MinMax Scaling is usually used prior to model fitting.

Numpy:

NumPy is the fundamental package for scientific computing in Python. It's a Python library which provides a multidimensional array object, several derived objects (such as masked arrays and matrices), and a combination of routines for quick operations on arrays, including mathematical, logical and shape manipulation, sorting, selecting, I/O, basic linear algebra, basic statistical operations, random simulation and a lot more.

At the core of the NumPy package, is the ndarray object. This binds n-dimensional arrays of homogeneous data types, with a lot of operations being performed in the compiled code for good performance.

There are many crucialdifferences between NumPy arrays and the standard Python sequences:

• NumPy arrays have a fixed size at creation, unlike Python lists (which can be grown dynamically). Modifying the size of an ndarray will create a new array and delete the original one.

• The elements in a NumPy array are all ensured to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of variable sized elements.

• NumPy arrays facilitate advanced mathematical and different types of operations on huge numbers of data. Typically, such operations are executed very efficiently and with few lines of code than is possible using Python’s built-in sequences.

ndarray.ndim

The number of dimensions of the array. In the Python , the number of dimensions is referred to as rank.

ndaray.shape

The dimensions of the array. This is a tuple of integers representing the size of the array in each dimension. For a matrix with n rows and m columns, shape will be (n,m). The length of the shape tuple is thereby the rank, or the number of dimensions, ndim.

ndarray.size

The total number of elements of the array. This is equal to the product of the elements of shape.

ndarray.dtype

An object describing the type of the elements in the array. One can create or specify dtype's using the standard Python types. Additionally NumPy provides types of its own. Like numpy.int32, numpy.int16, and numpy.float64, are some examples.

ndarray.itemsize

The size in bytes of each element of an array. For example, an array of elements of type float64has itemsize 8,while one of type complex32 has itemsize 4. It is equivalent to ndarray.dtype.itemsize.

Types of Plots

We can create various kinds of plots using matplotlib:

1) Histogram

2) Scatter Plot

3) Pie Chart

Histograms

Histograms are the graphical representations of a probability distribution. The histogram is a kind of bar chart. Using matplotlib and its bar chart function, one can create histogram charts.

A histogram chart has several advantages.

1) It displays the number of values within a specific interval.

2) It is more suitable for large data sets as they can be grouped within the intervals.

Scatter Plots

A scatter plot is used so as to graphically display the relationship between the variables. A basic plot can be created by using the plot method. However, if we need more control of a plot, it iss recommended that we use the scatter() method provided by matplotlib. It has several advantages:

1) It shows the correlation between variables.

2) It is more suitable for large data sets.

3) It is very easy to find clusters.

4) It is possible to represent each piece of data as a point on the plot.

Pie Chart

Pie charts are generally used to show percentage or proportional data. Usually, the percentage represented by each category is provided next to the corresponding slice of the pie. matplotlib provides the pie method to make pie charts. It has several advantages:

1) It summarizes a large data set in visual form.

2) It displays the relative proportions of multiple classes of data.

3) The size of the circle is made proportional to the total quantity.

Activation function

An activation function is a function that is added into an ANN in order to help the network learn the various complex patterns in the data. It takes in output signal from the previous cell and converts it into some other form that can be taken as an input to the next cell.

Desirable features of an activation function:

Vanishing Gradient problem: Neural Networks are trained using the process gradient descent. The gradient descent consists of the backward propagation step which is the chain rule to get the change in the weights so as to reduce the loss after every epoch.

Zero-Centered: Output of the activation function should be symmetrical at zero so that the gradients do not shift to a particular direction.

Types of Activation Function:

Sigmoid:

It is widely used non-linear activation function. Sigmoid transforms the values between the range 0 and 1. It is the mathematical expression for sigmoid-

f(x) = 1/(1+e^-x)

ReLU:

The ReLU function is non-linear activation function that gained popularity in the deep learning domain. ReLU stands for Rectified Linear Unit. The main advantage of using the ReLU function over other activation functions is that it does not activate all neurons at the same time.

Chapter-2

SYSTEM ANALYSIS

This System Analysis is very closely related to requirements analysis. It is also "an explicit formal inquiry carried out to help someone identify a better course of action and make a better decision than he might otherwise have made." This step has breaking down the system into different pieces to analyze the situation, getting project goals, breaking down what needs to be created and trying to engage users so that definite requirements can be defined.

2.1 Performance Requirements

Performance is measured in terms of the output provided by the application. Requirement specification plays a crucial part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which fit into specified environment. It rests largely with the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements should be known during the initial stages so that the system can be designed according to the requirements of the problem. It is very hard to modify the system once it has been designed and on the other hand designing a system, which does not adhere to the requirements of the user, it is of no use.

The requirement specification for any system can be broadly stated as given:

- The system should be able to interface with the existing system
- The system should be accurate
- The system should be better than the existing system

The existing system is fully dependent on the user to perform all the duties.

2.2 Software Requirements

- **Operating System**: Microsoft Windows 10.
- **Technology**: Recurrent Neural Networks(RNN), Long Short Term Memory(LSTM)
- **Programming Language**: Python.
- **Modules**: Tensorflow, Keras, Pandas, Numpy, Matplotlib, Scikit learn.
- **Platform**: Google Colab

- **Data Set**: Individual Household Electric Power Consumption Dataset(UCI Machine Learning Repository)

2.3 Hardware Requirements

Processor : Intel P-IV based system

RAM : Min. 512 MB

2.4 Household Power Consumption Prediction System

Power Outage accidents cause huge economic loss to the social economy. Our day to day activities and households go into darkness, our study and industrial activities becomes harder to do, Hence it is very important to predict household power consumption.

We need to download the dataset form UCI Machine Learning Repository which has 2075259 values and 9 attributes which is collected over 4 years. The attributes are global active power, Global reactive power, Voltage, Global intensity, Sub_metering1,Sub_metering2 and sub_metering_3 and our final target is to predict the global active power. We have to import various python libraries like Numpy for handling the array data, pandas to convert the given data into dataframes and to handle those Dataframes, Matplotlib to draw plots like line plot, scatter plot and bar plot among various features or attributes of given data.NaN is to be imported from numpy library to handle the missing or null values. We need to use a function called pd.read_csv() to the load the dataset into our python notebook by keeping the separator as ";", sort the dates by using parse_dates and assign false value to low memory because our data is very huge containing 137 MB of data. A simple way to store datasets is to use CSV files (comma separated files).CSV files contains plain text and is a well know format that can be read by everyone including Pandas. By default, when you print a DataFrame, you will only get the first 5 rows, and the last 5 rows. Next, we merge the date and time attributes by using str.cat() function and then we set the date_time feature as index by using set_index() function.Pandas set_index() is a method to set a List, Series or Data frame as index of a Data Frame. Index column can be set while making a data frame too. But sometimes a data frame is made out of two or more data frames and hence later index can be changed using this method. As there are many missing

values or null values in the form of '?' in the data which we have , we need to replace those particular null values of data in the form of '?' with nan which we have imported from the python library Numpy by using he function data.replace(). NaN handles the missing values of data in a systematic manner.numpy.nan is IEEE 754 floating point representation of Not a Number (NaN), which is of Python build-in numeric type float, then by using the data.info() function we can get all the details of datatypes of each and every attribute of our data. Afterwards we need to convert the datatype of each attribute of our data into float for uniformity of the datatype of our data using the data.astype() function.astype() method is used to cast a pandas object to a specified dtype. astype() function also provides the capability to convert any suitable existing column to categorical type. We need to get the idea of how any null values are present in our data using np.isnan(data).sum() function. So, when we apply this function and count the number of empty values of data we get it as 25979 in each feature of our data.

Afterwards, we write a python function to fill the missing values in our data and this step can be called Data Preprocessing. As we have the data recorded for each minute for four years we divide the data according to minutes and fill the missing value of household power with the value of previous year's household power consumed by the particular household and this is best possible way of filling the missing values of our data because we can't fill the empty values with mean or median value of household power consumption. After filling the missing values of our data, we confirm that the missing values in our data are 0 by applying the function np.isnan(data).sum() which gives 0.We have our data without any missing or null values in it at this stage. Now we need to do merge the rows of data and arrange the data in such a way that only the household power consumption for each day is only present in our data by using the function resample('D') in which d means the data is sorted according to the date of household power consumption. We define a function in python to plot various features i.e 7 attributes of our dataset in different years to visualize them and to draw insights from them. In the first plot, we plot the 7 features of our data at different years i.e singe feature at a time. The parameter called loc in the plt.title() function specifies the location of the title in the plot.yticks() Function. The annotate() function in pyplot module of matplotlib library is used to get and set the current tick locations and labels of the y-axis. There are some fluctuations in voltage from 2010 onwards and this is due to the power fluctuations occured in those particular years, the meter reading for those particular years is 0 and there is no household power consumption at that point of time. The plot

of every feature is flat in the months of august and September of 2008 and this means that there may be an occurence of power outage or family may have gone to any tour leaving home. In the plot of global active power we use the same above function to plot the global active power for each year i.e 2007, 2008, 2009 and 2010 as we are more interested towards global active power of our dataset. From this plot we can conclude that the household power consumption at the starting of the year and at the ending of the year are high and the household power consumption is lower in the middle of the year. As the data is from USA, people might be using heater systems and other devices used to warm the households so, the power consumption is high at starting and ending of each year.

There are also many fluctuations of power each year and they might be attributed to the different styles of work by people as people work more on weekdays and less on weekends. Here in this plot we plot the global active power in each year in the form of a histogram by using a function called hist() by binning the data and setting the size of each bin to 200.Matplotlib can be used to create histograms. A histogram shows the frequency on the vertical axis and the horizontal axis is another dimension. Usually it has bins, where every bin has a minimum and maximum value. Each bin also has a frequency between x and infinite. In the year of 2007, the power consumption is concentrated between 1.3 KW and 1.5 KW, in 2008 the power consumption is concentrated between 1.5 KW and 1.6 KW and in the years 2009,2010 the power consumption is concentrated around 1.5 KW approximately. After all these we're going to forecast the household power consumption for the next week using the data of previous week's household power consumption. Simply, from the data of week1 we predict the power consumption of week 2, similarly we predict and forecast the household power consumption of week 3 and week 4 using the data of week 2 and week 3.

We use the data of household power consumption from 2007 to 2010 as training data and we use the data of household power consumption in the year 2010 as the testing data. We divide the whole data according to our wish by using the function data.loc() function. Pandas provide a unique method to retrieve rows from a Data frame. DataFrame.loc[] method is a method that takes only index labels and returns row or dataframe if the index label exists in the caller data frame. Finally we extract the column of Global_Active_Power as our aim is to predict that global_active_power. While preparing training data we divide the whole training data into

X_train[] and y_train[] in such a way that X_train[] stores the data of previous week and the y_train[] stores the data of coming week. Afterwards we convert these data into array data using the function np.array().NumPy is used to work with arrays. The array object in NumPy is called ndarray.We can create a NumPy ndarray object by using the array() function. Finally we the data in the form of X_train and Y_train which have 1098 rows and 7 columns. We convert the array data into dataframe with the help of python pandas library and the function pd.Dataframe() for the systematic handling of the data.Pandas DataFrame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns).

A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas Data Frame consists of three principal components, the data, rows, and columns. First row of data consists of power consumption data of 1st week and in the next row next seven values are present which represent the data of the next week, like this arrangement of data is done perfectly and this all is due to conversion of data into data frames. We scale the data which we have from 0 to 1 by using MinMaxScaler() which is a best method used to scale the values of data. The main and prime reason behind the scaling of data is that our neural network can't take values in thousands and even though if it takes those values then it leads to inaccurate and inefficient output which is not at all desired. We need to reshape our data from 2-D data to 3-D data using reshape() function because neural network can't take 2 dimensional data as input and it can only take higher dimensional data like 3-D data as input which leads to better output. We finally build our regression model using LSTM neural network. We include Sequential model in it because the layers of neural network will be arranged in a sequence by Sequential Model.

A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor. In the LSTM network we give 200 units, choose relu activation function because the rectified linear activation function overcomes the vanishing gradient problem, allowing models to learn faster and perform better and also the rectified linear activation is the default activation function when developing multilayer perceptron and in the input shape 7 is the number of time steps and 1 is number of features we are going to predict. We add 7 dense layers to the neural network model as we are predicting 7 steps in the future. Dense layer is the regular deeply connected neural network layer. It is most common and frequently

used layer. Dense layers learn a weight matrix, where the first dimension of the matrix is the dimension of the input data, and the second dimension is the dimension of the output data. We fit the data i.e X_train, y_train to the regression model by using the fit() function with 100 epochs for highly efficient and quality training. The fit() method takes the training data as arguments, which can be one array in the case of unsupervised learning, or two arrays in the case of supervised learning. The model is fitted using X and Y , but the object holds no reference to X and y .We prepare the test data taking the values of household power consumption of 2010 into consideration. We prepare the test data by following the same steps while preparing the training data Finally we predict the household power consumption for the next week using the data of previous week and we predict the values by using the predict() function and we fit the X_test into the predict() function. Python predict() function enables us to predict the labels of the data values on the basis of the trained model. The predict() function accepts only a single argument which is usually the data to be tested. It returns the labels of the data passed as argument based upon the learned or trained data obtained from the model. Thus, the predict() function works on top of the trained model and makes use of the learned label to map and predict the labels for the data to be tested. When we predict the values of household power consumption we get the values in form of thousands but we need to apply inverse transform on that data by using the function y_scaler.inverse_transform() in order to obtain the original values. So, after obtaining these values they might not be same as the original values but they may be nearer to the original values and the amount of nearness to the original values shows the efficiency of our neural network model. In the process of evaluation of model, we define a function to calculate the root mean square and the mean squared error of our data and we pass the values of y_true which are original values and y_pred i.e predicted values of household power consumption and calculate the respective scores. When we calculate the root mean square error of our predicted values we get the value as 598 approximately and when we calculate the total standard deviation of our whole data we get the value as 710 which means that our model is highly efficient ad the root mean square error of our predicted values are far less than the standard deviation of our data and while using regression to predict the output this is the only way to evaluate the performance of the model unlike classification.

Advantages

1. An advanced (one week) prediction can help power companies regulate their supply and also, the consumer can use this information to make better decisions both financially and environment-consciously.
2. With the help of this model we can also predict the total wasted power by doing some modifications.
3. With the help of this model we can also predict the total wasted power by doing modifications.
4. With the recent increase in smart meters across the residential sectors, we have large publicly available datasets.
5. With such data, the power consumption of individual households can be tracked in almost real-time.
6. Using LSTM network for modeling gives highly accurate output which helps power regulating companies to keep track of power consumption and operate seamlessly.

Chapter-3

SYSTEM IMPLEMENTATION

The implementation stage of any project is a true display of the defining moments that make a project a success or a failure. The implementation stage is defined as the system or system modifications being installed and made operational in a production environment. The phase is initiated after the system has been tested and accepted by the user. This phase continues until the system is operating in production in accordance with the defined user requirements.

Algorithm for Power usage Prediction

1. Import all Packages.
2. Load the Data set
3. Identify the Null values in Dataset and replace with NaN.
4. Fill the Missing values.
5. Plot the Graphs.
    5.1. Identify the variation of the Global active Power using various plots.
6. Prepare the data for Training and Testing.
7. Reshape the Training data into array by using numpy.
    7.1. Storing the alternate weeks data into lists X_train and Y_train.
    7.2. Convert lists into dataframes and perform MinMaxNormalization.
    7.3. Convert the Training and Test data into 3-Dimensional Data.
8. Train the model and and calculate loss. Increase epochs to minimize losses.
9. Evaluate the Model.
    9.1. Calculate Root Mean Square Error.
    9.2. Calculate the Total Score.

9.3. Calculate the standard Deviation.

Psuedocode for Training the data set

If(data shape!=3-Dimensional)

{

1. Convert the Model into sequential Model.
2. Add LSTM Network on the Model.
3. Add Dense Layers
4. Compile the Model.

}

Else

{

1. Reshape the Model using reshape() function.
2. Convert the Model into sequential Model.
3. Add LSTM Network on the Model.
4. Add Dense Layers
5. Compile the Model.

}

Algorithm for Normalization and Inverse Normalization

1. Convert the Test and Training data into array using Numpy.

2. Scale the Data by using fit_transform() and MinMax scalar. .

3. Reshape the Data to 3-Dimensional Data.

4. Predict the values by using the build Model.

5. Apply inverse_transform() on the output dataframe and obtain the orginal values.

Chapter-4

SYSTEM DESIGN

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. Object-oriented analysis and design methods are becoming the most widely used methods for computer systems design.

4.1 Architectural Design

In Fig 4.1 depicts architectural design of our proposed system

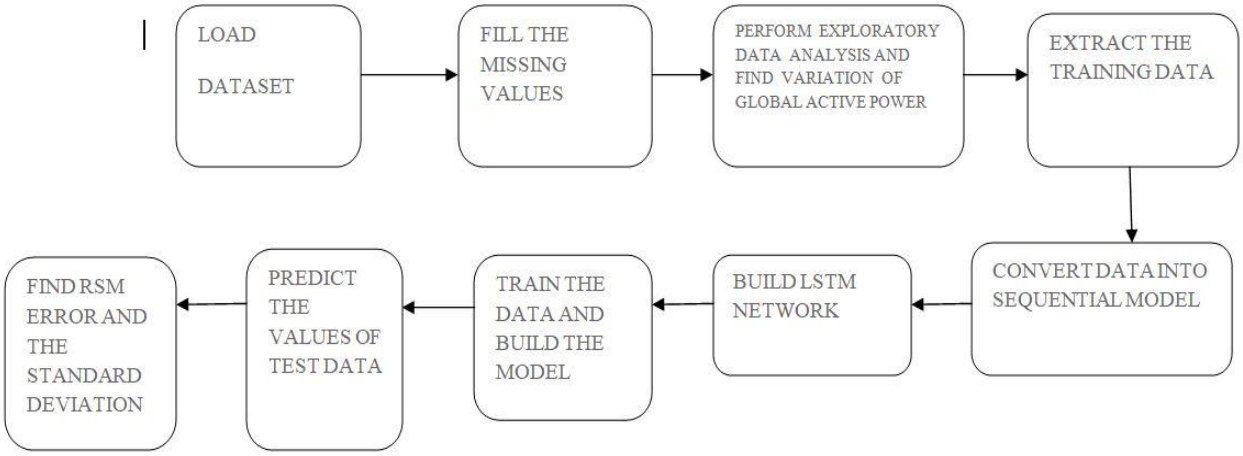


Fig 4.1 Architectural Design

The main concept of this LSTM model is initially we select the input data i.e. training data and perform the operations over 100 epochs and calculate the mean squared errors and total standard deviation of the whole data. We initially perform the training operations of the given model and the model gets trained and classify the given dataset into separate divisions and perform the test operations and come to a conclusion by predicting the household electric power consumption in one week advance. We compare the mean squared errors of predicted, true values of test data and standard deviation of the whole data, if the above mean squared error is less than the standard deviation of total data then the model is said to be efficient.

LOAD DATASET

In this step we download the data from UCI Machine Learning Repository and load the dataset into Google Colab from the local storage of our system using upload() function. After this we read the dataset using read_csv() function.

FILLING THE MISSING VALUES

This is the step of Data Pre-processing and in this step we identify the missing values which are present in the form of '?' and replace those missing values with electric power consumption values of the previous year. This is about filling the missing values.

PERFORM EXPLORATORY DATA ANALYSIS

In this step we perform exploratory data analysis by visualizing the data with the help of bar plots, line graphs and histograms and study the variation of global active power in various years along with months of the corresponding year.

EXTRACTING THE TRAINING DATA

We split the entire dataset into 2 parts i.e we take the data of household electric power consumption from 2007 to 2009 as training data and take the data 2010's household electric power consumption as testing data and we extract the training data from the dataset by using loc[] function.

CONVERTING THE DATA INTO SEQUENTIAL MODEL

We build the Sequential Model which is in the form linear stack of layers i.e one layer over the other and we give the data to that model.

BUILD THE LSTM NETWORK

We insert the LSTM network into the sequential model, use Relu activation function to learn complex patterns of data and add dense layers to the LSTM network.

4.2 UML Diagrams

UML diagrams are the unified modeling diagrams used to describe the sequential flow and the operations and the main functionalities of the project in a pictorial way. There are different types UML diagrams present they are

1. Use case Diagrams
2. Sequence Diagrams
3. Flow Chart
4. State diagram

4.2.1 Use Case Diagram

In Fig 4.2 it depicts the Use case diagram of the project

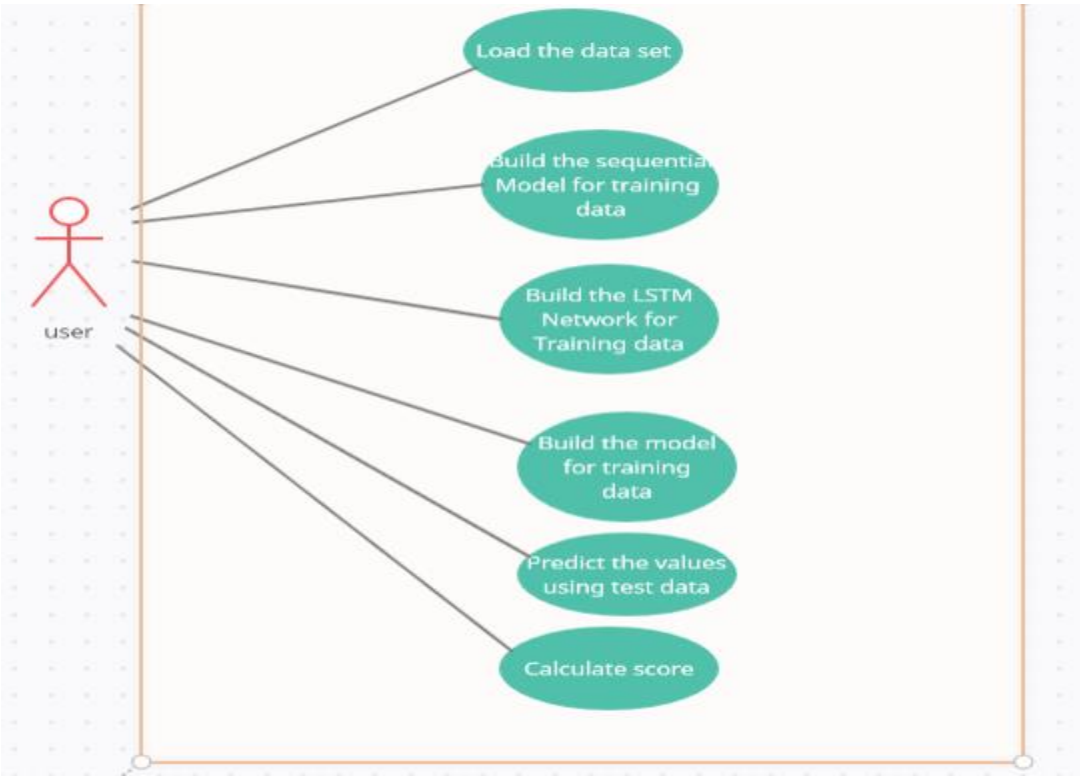


Fig 4.2.1 Use Case Diagram of the Household Power Prediction

This use case diagram as shown in the figure depicts the functionality and the process flow of this LSTM model. First we give the data of household power consumption of a particular week as input and try to find out the household electric power consumption for them succeeding week

of the previously mentioned week. The operation are exploratory data analysis, normalization of the original values of the household electric power consumption and it compares with the already trained model and it predicts the household electric power consumption for the coming week. This is how it predicts the household electric power consumption for the succeeding week of the previous week and view all the plots of various features while performing the exploratory data analysis.

4.2.2 Sequence Diagram

In Fig 4.2.2 it shows the Sequence Diagram of the project

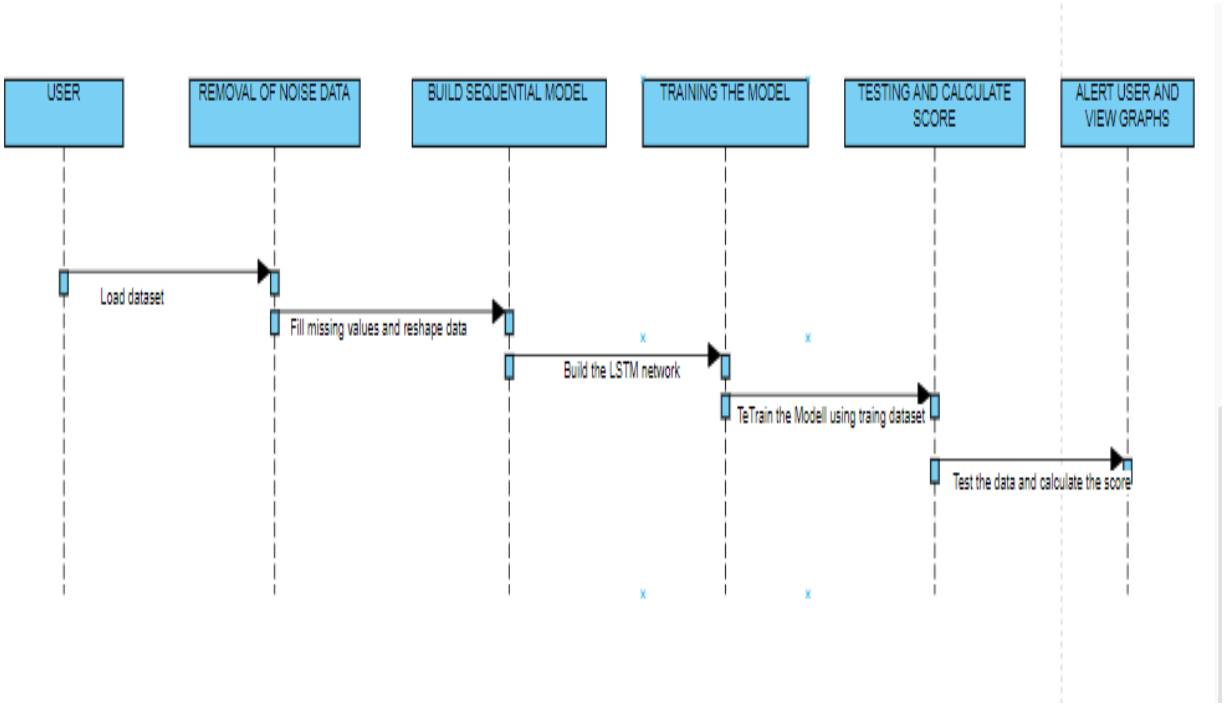


Fig 4.2.2 Sequence Diagram of proposed project implementation

The above diagram as shown in the figure shows the sequence of steps occurred during the process in a sequential manner. First we enter the values of household electric power consumption prediction and our model reads those values and it performs the training of the model, validation tests and estimates the accuracy of model i.e. by comparing the mean squared errors of predicted, true values of household electric power consumption and the standard deviation of the whole data. Finally it predicts the values of household electric power consumption for given test data values and we compare them with the original values. The

accuracy of our model lies in the nearness of the predicted values to the original values of household electric power consumption.

4.2.3 Flow Chart

In Fig 4.3 it shows the Flow chart of the project

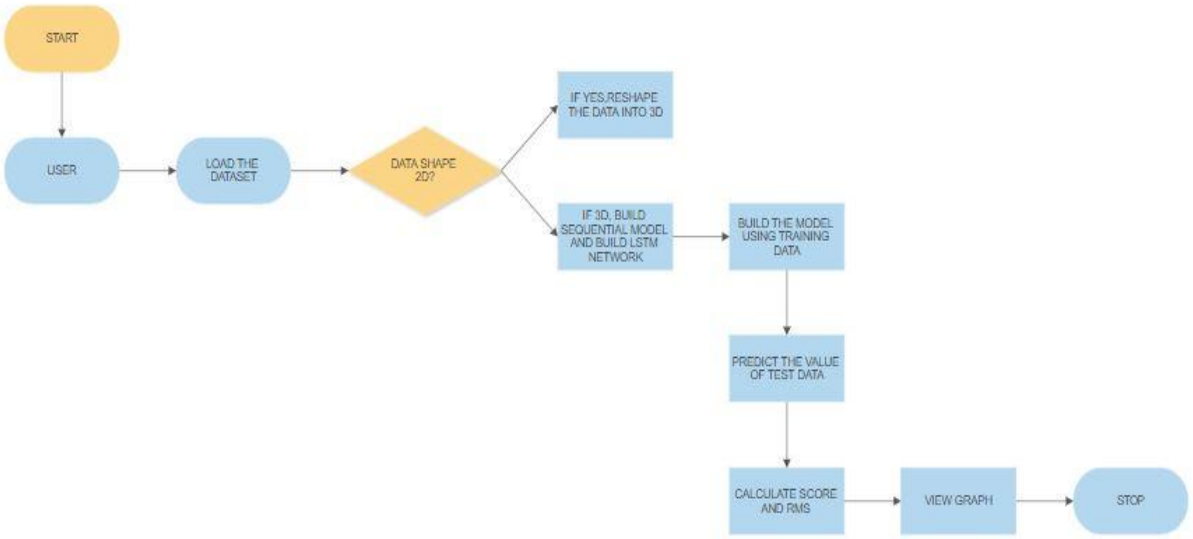


Fig 4.2.3 Flow chart of the project

The above diagram as shown in the figure shows the flow chart. It shows different activities that are performed during the process. The flow chart starts with start and ends with stop. Firstly we need to load the dataset and check the shape of dataset. If the shape of dataset is 3-dimensional then it goes to the next step or otherwise it is sent to reshaping. Afterwards, the model is build with LSTM network, training of the data is done to the model and then the prediction of values of household electric power consumption is performed. If the data is in the normalized form then the inverse transform is applied to the predicted values and then the original values are obtained. Finally mean squared error is calculated for predicted, true values of household electric power consumption and then it is compared with standard deviation of whole data and ensured that standard deviation of data is greater than the mean squared error.

4.2.4 State diagram

In Fig 4.4 it shows the State diagram of the project

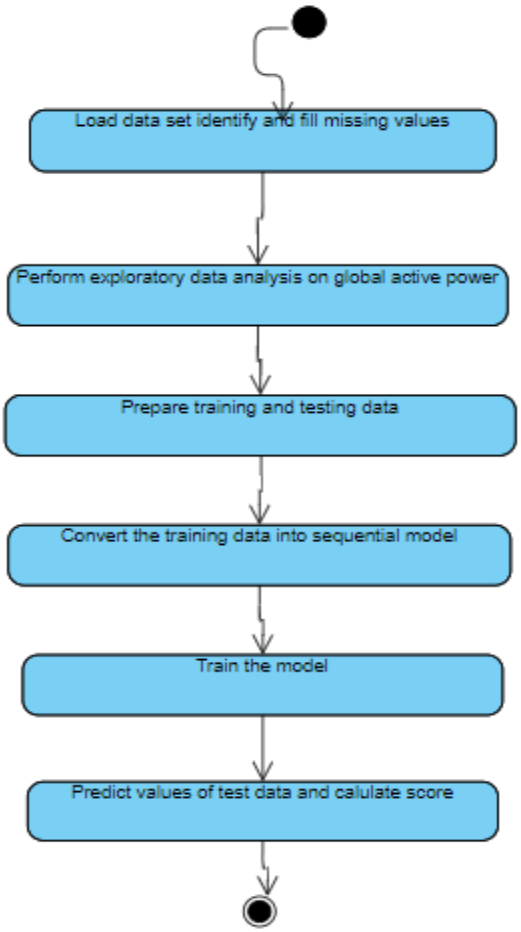


Fig 4.2.4 State diagram of the project

The above diagram as shown in the figure shows the flow chart. It shows various steps the process control flow passes through. First state is the state of loading the complete individual household electric power consumption dataset into our python notebook and after this we identify and fill the missing values with the values of household electric power consumption of the previous year. Secondly, the coming state is the process of exploratory data analysis. Afterwards we prepare the training and testing data by considering a part of whole dataset to training data and the remaining data to testing data and subsequently we feed the training data to the LSTM model and perform the process.

Chapter-5

OUTPUT SCREENS

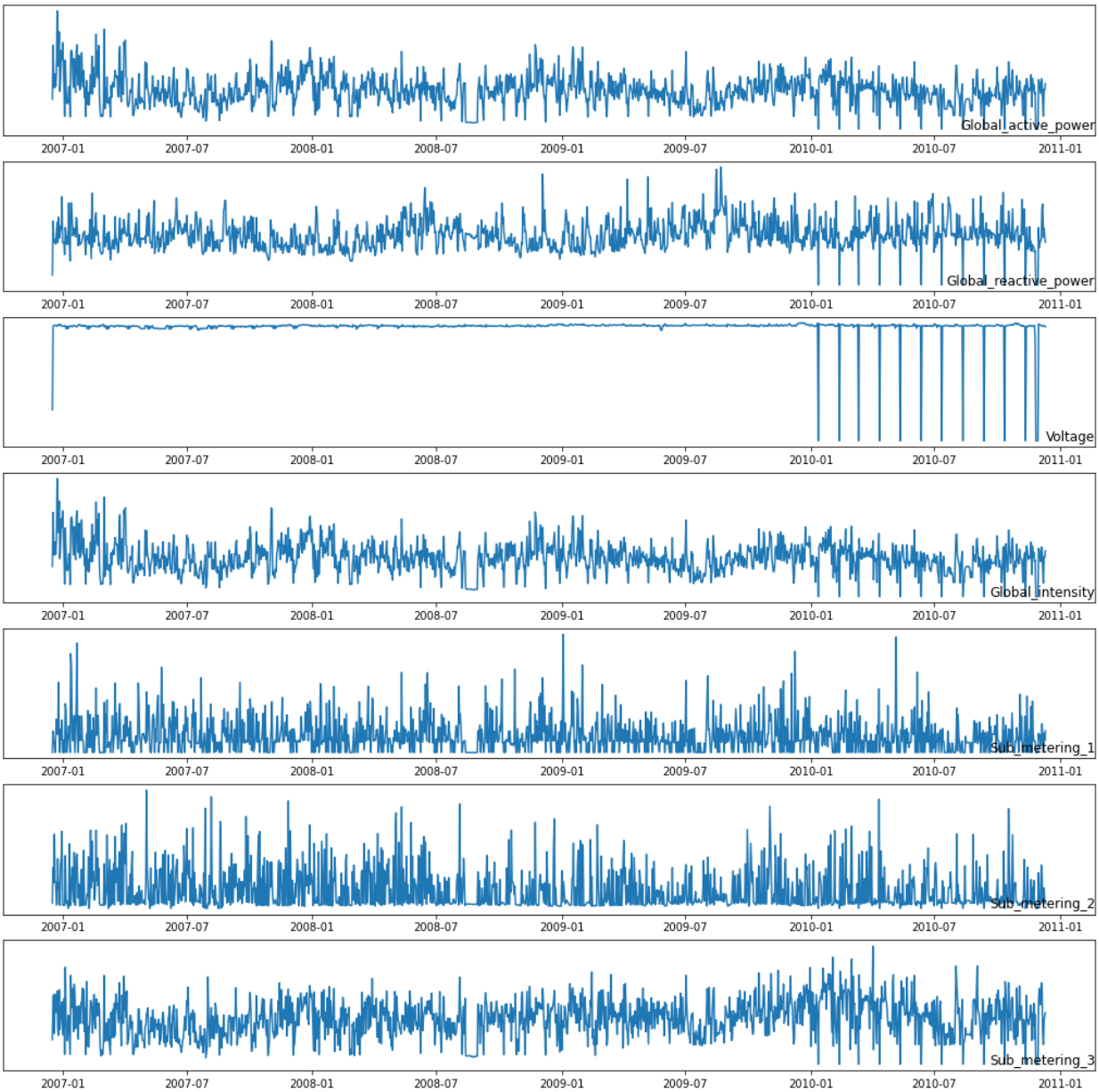


Fig 5.1 Variation of all features.

In the above fig 5.1 is the graph for studying the variation of global_active_power, global_reactive_power, voltage, global intensity, sub_metering1, sub_metering2, sub_metering3 over the years 2007,2008,2009 and 2010.The canvas for this plot is created using subplot()

function and yticks() function is also used here which sets the y-axis tick values, which are the locations along the y-axis where the tick marks appear. From this plot we can study the variation of the features of dataset over multiple years and in the month of September 2008 in this plot the plot is flat and from this we can conclude that the members in the household may have gone on a holiday or there may be a power outage at that time and we can also conclude that voltage is constant from this plot.

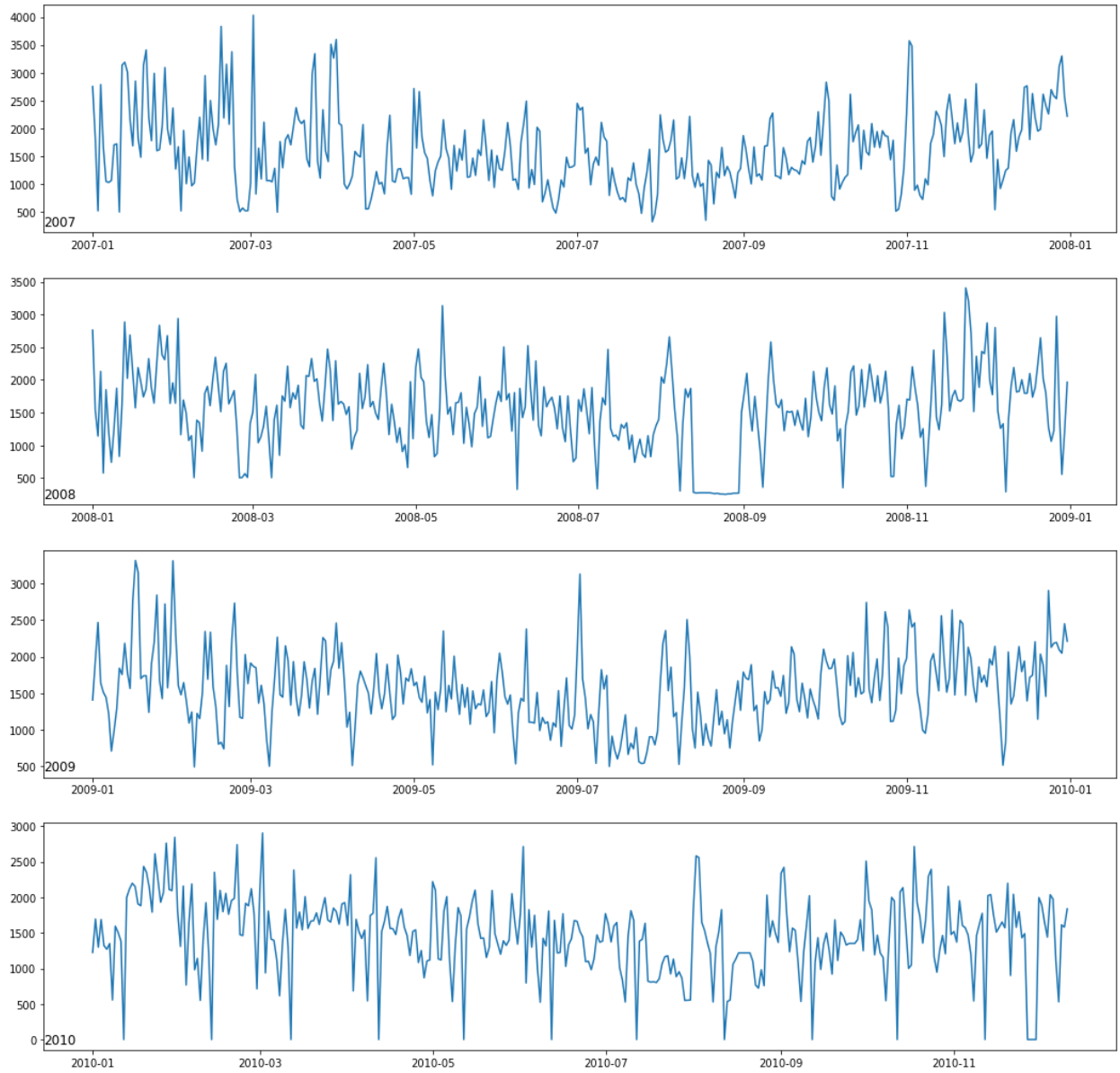


Fig 5.2 Variation of Global Active Power

In the above fig 5.2 it is the graph for studying the variation of global_active_power over the years 2007,2008, 2009 and 2010.The global_active_power is extracted from the existing features using the plt.plot( ) function and from this plot we can derive an insight that the household electric power consumption is high at the starting and ending of the above mentioned years because it'll be winter at the starting and ending of years and the electric power consumption is also high in the winter season because heating systems in winter and the electric power consumption is less from April to September in each of the four years. There is also a significant amount of fluctuation of electric power consumption and this may be attributed to the usage of high power in the weekends and usage of less power in week days because people tend to invest more time for themselves and their family rather in the week days because they work in week days.

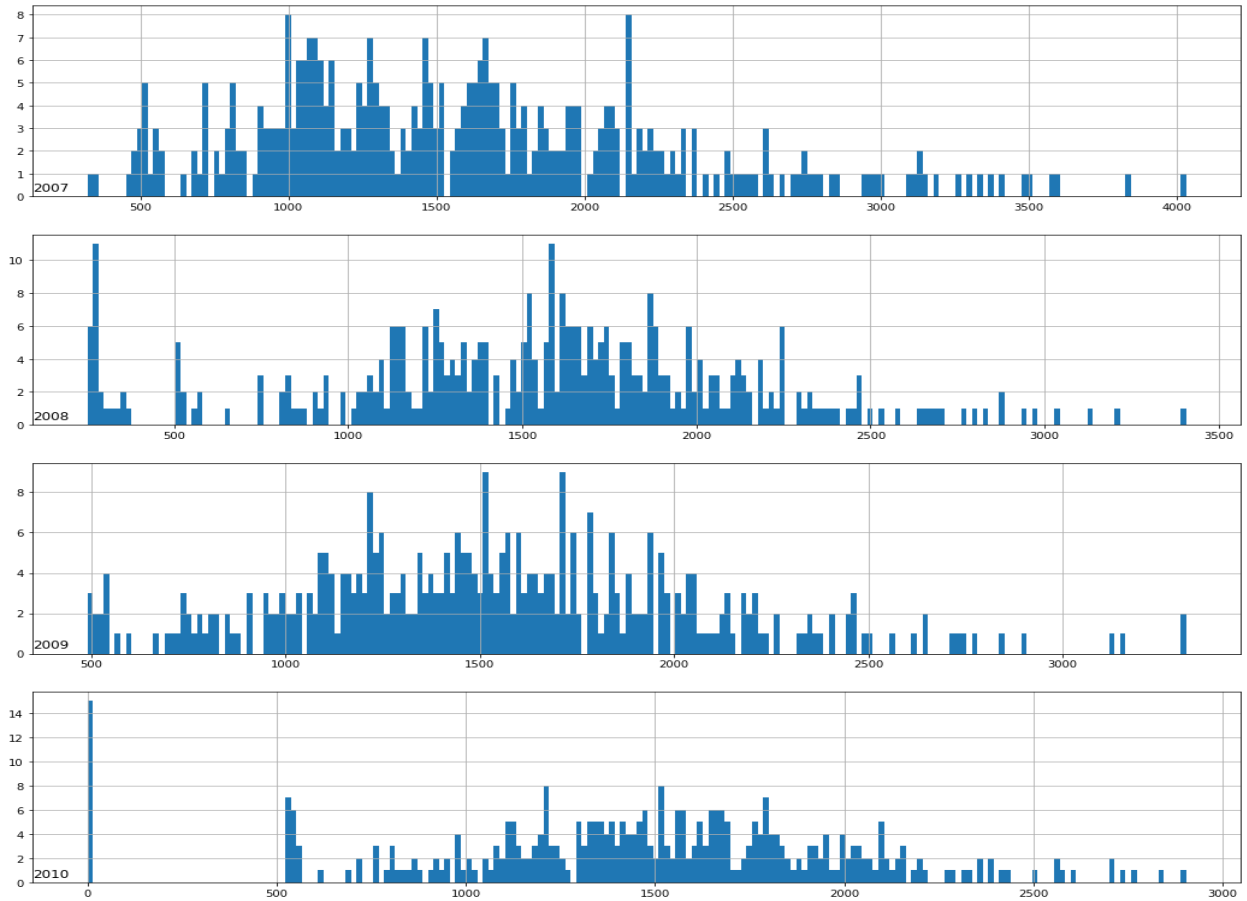


Fig 5.3 Histogram representation of Electric power Consumption

In the above fig 5.3 the plot is to study the consumption of household electric power consumption using a histogram from the years 2007 to 2010 and this histogram is created using hist() function. In the year 2007 the consumption of household electric power is concentrated around 1.3KW and 1.5KW, in the year 2008 the household electric power consumption is concentrated around 1.5KW and 1.6KW, in the year 2009 the consumption of electric power is concentrated around 1.4KW and 1.5KW and in the year 2010 the household electric power consumption is concentrated around 1.5KW.These are the insights we can derive from the above histogram plot.

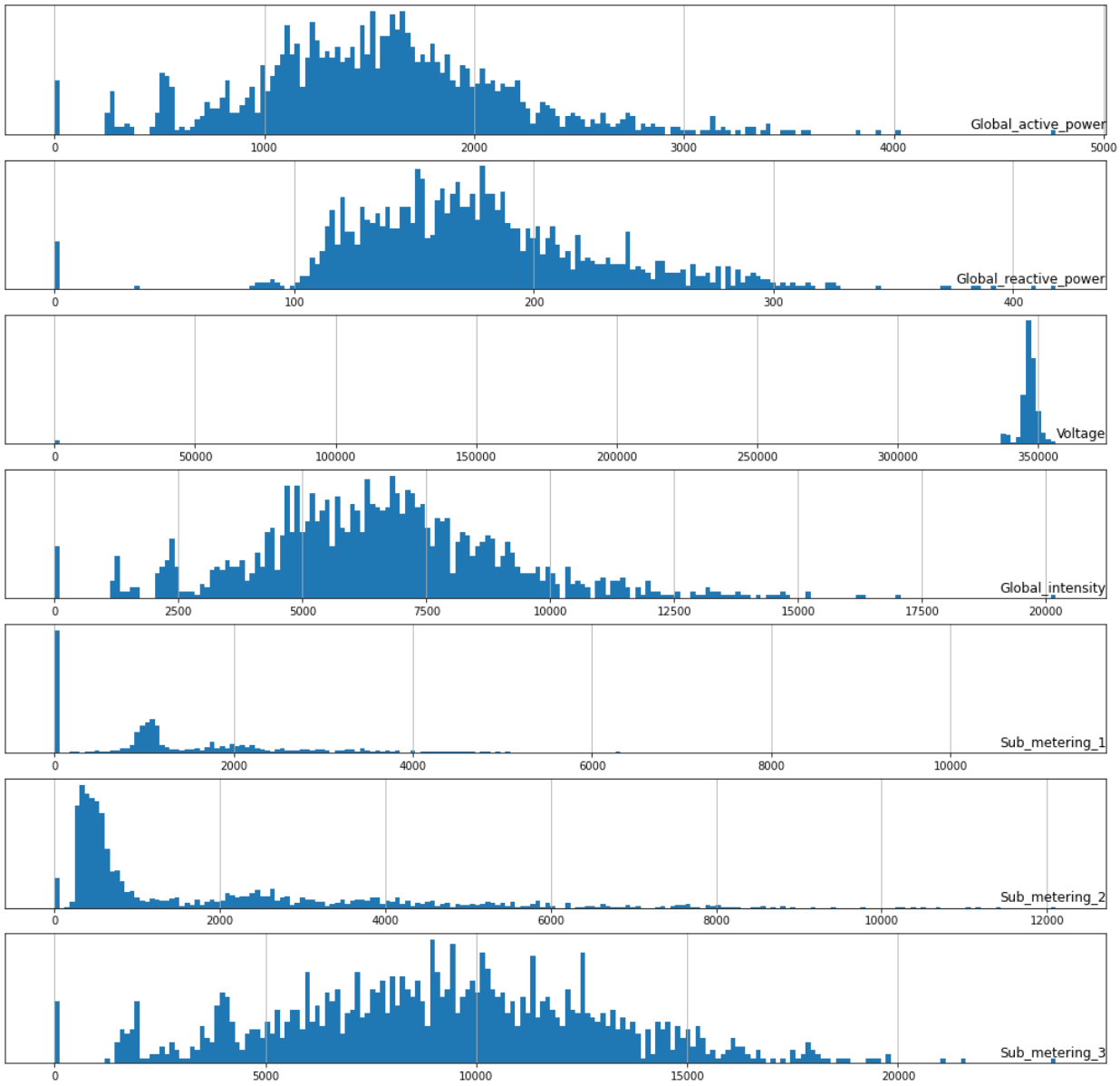


Fig 5.4 Histogram representation of all features

In the above fig 5.4 this plot is used to study the variation of all features of the dataset with the help of a histogram and this histogram is created using hist() function and canvas for this histogram is created using plt.subplot() function.The global_active_power is concentrated around 1.5KW and we're only interested to study only about global_active_power and we also finally predict the household electric power consumption.

```
[ ] evaluate_model(y_true, y_pred)

(578.5965060654481,
 [586.1500892435465,
  592.709746223213,
  573.4769385258126,
  565.5656782440522,
  583.0747895718465,
  575.418102365468,
  572.4228822468815])

[ ] #standard deviation

    np.std(y_true[0])

    710.0253857243853
```

Fig 5.5 predicted values

In the above fig 5.5 the model prediction is obtained by following the below mentioned steps.

1. Divide the testing data into X_test and Y_test.

2. By using X_test, predict the value of y_pred whose values are stored in an array.

3. Evaluate the model by using evaluate_model function which takes y_true and y_predicted as parameters.

4. Calculate Root Mean Squared Error(RMSE) and Standard Deviation.

We extract the testing data from the dataset and assign the data from 2007 to 2009 to training data and assign the data in 2010 to the testing data and again we divide the testing data into

X_test and y_test and store the values of the present week's household electric power consumption into y_test and store the values of corresponding previous week's houshold electric power consumption into X_test and finally we convert them ito array data.We apply scaler transform function onto the testing data which scales the data from 0 to 1.We feed the X_test's data to our model and make the model to predict the values of household electric power consumption and store those values in the variable y_pred and again we apply inverse transform to convert them into original values.Afterwards we compare the values of y_pred and y_test, calculate the RMS value, standard deviation of y_pred values and y_test values and after calculation of those scores we observe that the standard deviation of y_pred(578) is less than the standard deviation of the y_test(710) values which we named it as y_true.So, we can conclude that our model is accurate from the above results.

Chapter-6

FUTURE SCOPE

The analysis and projections lead to a number of conclusions and recommendations that can be considered by governments and people, engaged in different aspects of electricity system planning, regulation and policy.

In the further improvement of the model, we can work on the data collected from the different regions and the dependencies that are related to the electric power consumption and we can also use other resources such as Renewable resources and we can gain more insights from data which makes the model diverse.

CONCLUSION

The decision maker in the energy sector needs accurate forecasts of electricity consumption since most decisions are necessarily based on anticipating future demands. Short-term load forecasting is based on mathematical model and different input variables. Due to the problem complexity, the model structure and parameters are determined based on available data. In this project, various model based on RNN is implemented. Moreover, a growing number of hybrid forecasting techniques are being applied to increase the model's accuracy, usually by introducing wavelet transformation or/and evolutionary algorithm. The methods done in this project show certain ability to predict household electric power, ultimately leading to a reduction in the operating costs of the power system and increasing its efficiency and reliability.

I find the LSTM model easy as compared to the other models and also the LSTM model gives a better accuracy then the other models. Therefore in this project we used the LSTM model on the individual household electricity consumption dataset. Then, chose the suitable forecasting method and identified the most suitable forecasting period by considering the smallest values of RMSE. In this dataset the consumption of electricity is more in the starting and ending months of the year and regular during the other time period of the year. The results showed that the LSTM model represent the most suitable forecasting periods in weekly basis.

Appendix-A

Google Colab

Google Colaboratory is a free online cloud-based Jupyter notebook environment that allows us to train our machine learning and deep learning models on CPUs, GPUs, and TPUs.

LONG SHORT-TERM MEMORY (LSTM)

Long short-term memory networks are an extension for recurrent neural networks, which basically extends the memory. Therefore it is well suited to learn from important experiences that have very long time lags in between. The units of an LSTM are used as building units for the layers of a RNN, often called an LSTM network.

Tensor flow

Tensor Flow is an interface for expressing machine learning algorithms and an implementation for executing such algorithms. A computation expressed using Tensor Flow can be executed with little or no change on a wide variety of heterogeneous systems, ranging from mobile devices such as phones and tablets up to large-scale distributed systems of hundreds of machines and thousands of computational devices such as GPU cards.

Scikit Learn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python. It was originally called scikits.learn and was initially developed by David Cournapeau as a Google summer of code project in 2007. Later, in 2010, Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, and Vincent Michel, from FIRCA (French Institute for Research in Computer Science and Automation), took this project at another level and made the first public release (v0.1 beta) on 1st Feb. 2010.

Numpy

NumPy is the fundamental package for scientific computing in Python. It's a Python library which provides a multidimensional array object, several derived objects (such as masked arrays and matrices), and a combination of routines for quick operations on arrays, including mathematical, logical and shape manipulation, sorting, selecting, I/O, basic linear algebra, basic statistical operations, random simulation and a lot more.

Activation function

An activation function is a function that is added into an ANN in order to help the network learn the various complex patterns in the data. It takes in output signal from the previous cell and converts it into some other form that can be taken as an input to the next cell.

Desirable features of an activation function:

Vanishing Gradient problem

Neural Networks are trained using the process gradient descent. The gradient descent consists of the backward propagation step which is the chain rule to get the change in the weights so as to reduce the loss after every epoch.

Zero-Centered

Output of the activation function should be symmetrical at zero so that the gradients do not shift to a particular direction.

REFERENCES

[1] Fumo N, Biswas MR. Regression analysis for prediction of residential energy consumption. Renew Sustain Energy Rev 2015;47:332e43.

[2] Bogomolov A, Lepri B, Larcher R, Antonelli F, Pianesi F, Pentland A. Energy consumption prediction using people dynamics derived from cellular network data. EPJ Data Science 2016;5(1):1e13 .

[3] Shen M, Cui Q, Fu L. Personality traits and energy conservation. Energy Policy. 2015;85:322-34.

[4] Milfont TL, Sibley CG. The big five personality traits and environmental engagement: Associations at the individual and societal level. J Environ Psychol. 2012 32(2):187-95.

[5] He HZ, Kua HW. Lessons for integrated household energy conservation policy from Singapore's southwest Eco-living Program. Energy Policy. 2013;55:105-16.

[6] Gosling SD, Rentfrow PJ, Swann WB. A very brief measure of the Big-Five personality domains. J Res Pers. 2003;37(6):504-28.

[7] Shen M, Young R, Cui Q. The normative feedback approach for energy conservation behaviour in the military community. Energy Policy. 2016; 98:19-32.

[8] Shen M, Cui Q. Behaviour Driven Energy Efficiency: A Customized Feedback Approach. Energy Procedia. 2015;78:2112-7.

[9] Shen M, Cui Q, Fu L. Personality traits and energy conservation. Energy Policy. 2015;85:322-34.

[10] Milfont TL, Sibley CG. The big five personality traits and environmental engagement: Associations at the individual and societal level. J Environ Psychol. 2012 32(2):187-95.

[11] He HZ, Kua HW. Lessons for integrated household energy conservation policy from Singapore's southwest Eco-living Program. Energy Policy. 2013;55:105-16.

[12] Gosling SD, Rentfrow PJ, Swann WB. A very brief measure of the Big-Five personality domains. J Res Pers. 2003;37(6):504-28.

[13] c Kamunda A Study on Efficient Energy Use for Household Appliances in Malaw

[14] Naser Farag Abed1 and Milan M.Milosavljevic 1,2 1 Singidunum University Belgrade, 11000, Serbia 2 School of Electrical Engineering, Belgrade University, Belgrade, 11000, Serbia“Single Home Electricity Power Consumption Forecast Using Neural Networks Model”IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 3 Issue 1, January 2016. ISSN 2348 – 7968

[15]https://machinelearningmastery.com/how-to-develop.