**An Efficient Deep Learning Framework for Intelligent Energy Management in IoT Networks**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TITLE** | **PAGE NO** |
|  | **ABSTRACT** |  |
|  | **INTRODUCTION** |  |
|  | * 1. General Introduction |  |
|  | * 1. Project Objectives |  |
|  | **SYSTEM** **PROPOSAL** |  |
|  | * 1. Existing System |  |
|  | * + 1. Disadvantages |  |
|  | * 1. Proposed System |  |
|  | 2.2.1 Advantages |  |
|  | * 1. Literature Survey |  |
|  | **SYSTEM DIAGRAMS** |  |
|  | * 1. Architecture Diagram |  |
|  | * 1. Flow Diagram |  |
|  | * 1. UML Diagrams |  |
|  | **IMPLEMENTATION** |  |
|  | * 1. Modules |  |
|  | * 1. Modules Description |  |
|  | **SYSTEM** **REQUIREMENTS** |  |
|  | * 1. Hardware Requirements |  |
|  | * 1. Software Requirements |  |
|  | * 1. Software Description |  |
|  | * 1. Testing of Products |  |
|  | **CONCLUSION** |  |
| **7.** | **FUTURE** **ENHANCEMENT** |  |
| **8.** | **SAMPLE** **CODING** |  |
| **9.** | **SAMPLE** **SCREENSHOT** |  |
| **10.** | **REFERENCES** |  |

**ABSTRACT**

Green energy management is an economical solution for better energy usage, but the employed literature lacks focusing on the potentials of edge intelligence in controllable Internet of Things (IoT). Therefore, in this article, we focus on the requirements of todays’ smart grids, homes, and industries to propose a deep learning based framework for intelligent energy management. We predict future energy consumption for short intervals of time as well as provide an efficient way of communication between energy distributers and consumers. In our system, we have to predict or analyse the household power consumption. Here, we are used household power consumption dataset is taken from dataset repository. Then we have to implement the preprocessing step for avoid wrong prediction. The system is developed the different deep learning algorithms such as long short term memory (LSTM) and gate recurrent unit (GRU) for analysing the energy consumption. Finally, the experimental results shows that some performance metrics such as mean Absolute Error (MAE), mean Squared Error (MSE) and Root Mean Squared Error (RMSE). Then, we have to compare the results and we conclude that GRU is efficient.

**CHAPTER 1**

**INTRODUCTION**

* 1. **General Introduction:**

Energy management at smart grids via automated techniques for future load forecasting is an interesting area of research. Smart grids are the secure and trust-worthy locations to distribute the electric energy among diverse sets of consumers such as smart homes and industries. The electric energy retails chain includes production at power plants, distribution at smart grids, and consumption at residential or commercial buildings and industrial sectors. The amount of energy produced in power plants that is distributed at grids is entirely influenced by its usage at consumer side. Majority of the consumers are non-experts of energy demands from electric grids, resulting financial loss and futile energy expenditure. Similarly, the producers want to minimize the cost and obtain an optimized level of energy generation, farming the need of appropriate scheduling and management strategies.

A proper planning for energy production and consumption ensures its purposeful usage at industries/household and a balanced amount of energy generation at power plants. The channel holding the energy communication stability between producer and consumer is smart grid that is responsible for the equilibrium state of energy for both parties.

Energy forecasting methods are significantly helpful in this regard that predict the future energy of a consumer and demands accordingly from the grids. Miss-prediction of energy leads to additional costs and its wastage. A loss of 10 million pounds per year is reported with an increase of 1% forecasting error in the United Kingdom in 1984 for a residential building.

Electricity load forecasting has gained substantial importance nowadays in the modern electrical power management systems with elements of smart greed technology. A reliable forecast of electrical power consumption represents a starting point in policy development and improvement of energy production and distribution. At the level of individual households, the ability to accurately predict consumption of electricity power significantly reduces prices by appropriate systems for energy storage. Therefore, the energy efficient power networks of the future will require entirely new ways of forecasting demand on the scale of individual households.

**1.2 Objectives:**

The main objective of our project is,

* To predict or to analyze the household power consumption.
* To implement the different deep learning algorithm such as LSTM and GRU.
* To prove the GRU is efficient when compared with long short term memory.
* To enhance the overall performance for classification algorithms.

**CHAPTER 2**

**SYSTEM PROPOSAL**

**2.1 EXISTING SYSTEM:**

In existing system, Green energy management is an economical solution for better energy usage, but the employed literature lacks focusing on the potentials of edge intelligence in controllable Internet of Things (IoT). Therefore, in this article, we focus on the requirements of todays’ smart grids, homes, and industries to propose a deep learning based framework for intelligent energy management. We predict future energy consumption for short intervals of time as well as provide an efficient way of communication between energy distributers and consumers. The key contributions include edge devices based real-time energy management via common cloud-based data supervising server, optimal normalization technique selection, and a novel sequence learning based energy forecasting mechanism with reduced time complexity and lowest error rates. In the proposed framework, edge devices relate to a common cloud server in an IoT network that communicates with the associated smart grids to effectively continue the energy demand and response phenomenon. We apply several preprocessing techniques to deal with diverse nature of electricity data, followed by an efficient decision-making algorithm for short-term forecasting and implement it over dependable resource constrained devices. We perform extensive experiments and witness 0.15 and 3.77 units reduced MSE and RMSE for residential and commercial datasets, respectively.

**2.1.1 DISADVANTAGES:**

* It doesn’t efficient for large volume of data’s.
* The prediction is not efficient.
* The performance is low.
* Time consumption is high.
  1. **PROPOSED SYSTEM:**

In this system, the household power consumption dataset is collected from dataset repository. In household power consumption dataset, we have to implement the data pre-processing step. In this step, we have to handle the missing values for avoid wrong prediction, and to drop the unnecessary attributes or columns. Then, we have to split the dataset into test and train. The data is splitting is based on ratio. In train, most of the data’s will be there. In test, smaller portion of the data’s will be there. Training portion is used to evaluate the model and testing portion is used to predicting the model. Then we have to implement the deep learning for predicting the power consumption. Finally, the experimental results shows that the performance metrics such as mean absolute error, mean squared error and root mean squared error.

**2.2.1 ADVANTAGES:**

* It is efficient for large number of datasets.
* Time consumption is low.
* The Process was implemented with removing the unwanted data.

**2.3 LITERATURE SURVEY:**

**2.3.1: Individual Household Electric Power Consumption Forecasting using Machine Learning Algorithms.**

***Year***: September 2019,

***Author***: Aaditi Parate Student, M.Sc. (Big Data Analytics)

***Methodology***: Electricity load forecasting has gained substantial importance nowadays in the modern electrical power management systems with elements of smart greed technology. A reliable forecast of electrical power consumption represents a starting point in policy development and improvement of energy production and distribution. At the level of individual households, the ability to accurately predict consumption of electricity power significantly reduces prices by appropriate systems for energy storage. Therefore, the energy efficient power networks of the future will require entirely new ways of forecasting demand on the scale of individual households.

***Advantage***:

* Many researchers wrote about the ARIMA model, AR model, MA model and also worked with these models on the consumption of electricity.
* I find the ARIMA model easy as compared to the other models and also the ARIMA model gives a better accuracy then the other models.

***Disadvantage***:

* A line plot of the forecast is also created, showing the RMSE in kilowatts for each of the seven lead times of the forecast.
* We can see an interesting pattern. We might expect that earlier lead times are easier to forecast than later lead times, as the error at each successive lead time compounds

**2.3.2 Predicting the Energy Consumption of Residential Buildings for Regional Electricity Supply-Side and Demand-Side Management**

***Year*:** February 25, 2019

***Author***: SHOUPENG SHEN received the B.S. degree

***Methodology***: This paper provides solutions for the energy consumption prediction problem of residential buildings. The research technical route is shown in Figure 1. First, it is important to the data acquired from the databases, which includes the characteristics of the residential buildings, the weather information and the energy consumption, to remove the noise, outliers and missing values from the data. The sensors used for building energy consumption measurement systems are multi-sourced and asynchronous, and the measurement and control system may encounter network fluctuations or network interruption in the long-time operation process, which results in some abnormal and missing values. After the characterization, the features of all the data types are divided into the same standard.

***Advantage***:

* For future work, we are planning to improve the sampling algorithm for imbalance classification and introduce deep learning methods to the research on the classification and prediction of electricity usage in residential buildings.
* This will help increase the accuracy and efficiency. Moreover, we will study the electricity energy scheduling strategy according to the prediction for electricity energy consumption ratings of residential buildings based on an entire region.

***Disadvantage***:

* The comparison results demonstrate that the optimized SVM model based on the SMOTE-ENN improves the classification performance by an average of 24.1% in terms of accuracy, 26.03% in terms of precision, 24.01% in terms of recall and 25.35% in terms of F-measure.

**2.3.3 Household Power Consumption Prediction Method Based on Selective Ensemble Learning**

***Year***: May 21, 2020

***Author***: YIYING ZHANG received the B.E. degree

***Methodology***: In the context power big data, the household power consumption data on the user side has the characteristics of large quantity, wide distribution and many types. Ensemble learning is very excellent in the analysis and mining of power data with large amount of data, strong timeliness and many influencing factors. Based on the data of household power consumption, this paper and predicts the power consumption of some users in a city by using selective ensemble learning technology and combining with meteorological factors. In this paper, K-means algorithm is used to cluster the household power consumption, and then the clustering results are combined with the meteorological information. In the stage of power consumption prediction, a Filter Iterative Optimization Ensemble Strategy (FIOES) is proposed to selectively ensemble the basic learners and get the final prediction model. Experimental results show that the FIOES algorithm has better performance in time cost and prediction accuracy than the traditional ensemble learning algorithm.

***Advantage*:**

* The main advantage of the Ranking method is that it is simple and fast. There are also advantages in generalization performance and the calculation method of the difference between learners.

***Disadvantage***:

* The proposed FIOES is effective for almost all power consumption prediction, which means that FIOES has good generalization ability.

**2.3.4 Multiple Electric Energy Consumption Forecasting Using a Cluster-Based Strategy for Transfer Learning in Smart Building**

***Year*:** May 7, 2020

***Author*:** [Seungmin Rho](https://www.ncbi.nlm.nih.gov/pubmed/?term=Rho%20S%5BAuthor%5D&cauthor=true&cauthor_uid=32392858)

***Methodology***: Electric energy consumption forecasting is an interesting, challenging, and important issue in energy management and equipment efficiency improvement. Existing approaches are predictive models that have the ability to predict for a specific profile, i.e., a time series of a whole building or an individual household in a smart building. In practice, there are many profiles in each smart building, which leads to time-consuming and expensive system resources. Therefore, this study develops a robust framework for the Multiple Electric Energy Consumption forecasting (MEC) of a smart building using Transfer Learning and Long Short-Term Memory (TLL), the so-called MEC-TLL framework. In this framework, we first employ a k-means clustering algorithm to cluster the daily load demand of many profiles in the training set. In this phase, we also perform Silhouetteanalysis to specify the optimal number of clusters for the experimental datasets.

***Advantage*:**

* There are two main approaches that use Machine Learning to solve the electrical energy consumption forecasting problem in smart buildings, including occupant-centric and energy/devices-centric.
* The first approach considers occupant aspects, including occupancy estimation and identification, human activity recognition, and preference estimation.

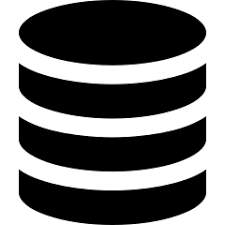
***Disadvantage*:**

* This study develops an effective framework for multiple electric energy consumption forecasting in smart buildings, namely MEC-TLL, which utilizes the concept of transfer learning and a cluster-based strategy for training the LSTM models to reduce the computational time.

**CHAPTER 3**

**SYSTEM DIAGRAMS**

**3.1 ARCHITECTURE DIAGRAM**



**Household power consumption dataset**

Preprocessing

***Checking Missing Values***

***Drop unwanted columns***

Data Splitting

Normalization/Scaling

Classification

Performance Metrics

***GRU***

***LSTM***

**Min Max Scalar**

**INPUT DATA**



* **Mean Absolute Error**
* **Mean Squared Error**
* **Root Mean Squared Error**

***Train***

***Test***

**Compare the results**

FIGURE 3.1: ARCHITECTURE DIAGRAM

**3.2 FLOW DIAGRAM**

Input Data

Preprocessing

Data Splitting

Classification

Performance analysis

FIGURE 3.2: FLOW DIAGRAM

**3.3 UML DIAGRAMS:**

**3.3.1 USE CASE DIAGRAM:**

System

User

FIGURE 3.3.1: USE CASE DIAGRAM

**3.3.2 ACTIVITY DIAGRAM:**

Input Data

Preprocessing

Data splitting

Performance metrics

Classification

FIGURE 3.3.2: ACTIVITY DIAGRAM

**3.3.3 SEQUENCE DIAGRAM:**

Input Data

Preprocessing

Data splitting

Classification

Select data

Missing value

Test and Train

Load data

Data splitting

LSTM and GRU

FIGURE 3.3.3: SEQUENCE DIAGRAM

**3.3.4 ER DIAGRAM:**

Data selection

Preprocessing

Data splitting

Classification

FIGURE 3.3.4: ER DIAGRAM

**3.3.5 CLASS DIAGRAM:**

Select data ()

Load data ()

View data ()

INPUT

Test ()

Data Splitting

MSE ()

MAE ()

RMSE ()

Performance analysis

Preprocessing

Missing values ()

Drop columns ()

Min max scalar ()

LSTM ()

GRU ()

Classification

Train ()

FIGURE 3.3.5: CLASS DIAGRAM

**CHAPTER 4**

**IMPLEMENTATION**

**4.1 MODULES:**

* Data selection
* Preprocessing
* Feature Scaling
* Data splitting
* Classification
* Result generation

**4.2 MODULES DESCRIPTION:**

**4.2.1: DATA SELECTION:**

* The input household power consumption dataset was collected from dataset repository.
* The data selection is the process of analyzing the energy consumption.
* In python, with the help of panda’s package we have to read an input data.
* In our dataset, the dataset contains the information about the Voltage, Global intensity, Global\_active\_power and so on.

**4.2.2: DATA PREPROCESSING:**

* Data pre-processing is the process of removing the unwanted data from the dataset.
* Pre-processing data transformation operations are used to transform the dataset into a structure suitable for machine learning.
* This step also includes cleaning the dataset by removing irrelevant or corrupted data that can affect the accuracy of the dataset, which makes it more efficient.
* Missing data removal
* ***Missing data removal***: In this process, the null values such as missing values and Nan values are replaced by 0.
* Missing and duplicate values were removed and data was cleaned of any abnormalities.

**4.2.3 FEATURE SCALING:**

* In our process, we have to implement the feature scaling such as min mas scalar.
* Feature scaling is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step.
* The **min-max scalar** form of normalization uses the mean and standard deviation to box all the data into a range lying between a certain min and max value. For most purposes, the range is set between 0 and 1.

**4.2.4: DATA SPLITTING:**

* During the machine learning process, data are needed so that learning can take place.
* In addition to the data required for training, test data are needed to evaluate the performance of the algorithm in order to see how well it works.
* In our process, we considered 70% of the input dataset to be the training data and the remaining 30% to be the testing data.
* Data splitting is the act of partitioning available data into two portions, usually for cross-validator purposes.
* One Portion of the data is used to develop a predictive model and the other to evaluate the model's performance.
* Separating data into training and testing sets is an important part of evaluating data mining models.
* Typically, when you separate a data set into a training set and testing set, most of the data is used for training, and a smaller portion of the data is used for testing.

**4.2.5: CLASSIFICATION:**

* In our process, we have to implement the deep learning algorithm such as GRU and LSTM.
* A ***gated recurrent unit (GRU)*** is part of a specific model of recurrent neural network that intends to use connections through a sequence of nodes to perform machine learning tasks associated with memory and clustering, for instance, in speech recognition.
* ***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.
* The key difference between GRU and LSTM is that GRU's bag has two gates that are reset and update while LSTM has three gates that are input, output, forget.
* GRU is less complex than LSTM because it has less number of gates. If the dataset is small then GRU is preferred otherwise LSTM for the larger dataset.

**4.2.6: RESULT GENERATION:**

The Final Result will get generated based on the overall classification and prediction. The performance of this proposed approach is evaluated using some measures like,

* **MAE:** In statistics, the **mean absolute error** (MAE) is a way to measure the accuracy of a given model. It is calculated as:

**MAE = (1/n) \* Σ|yi – xi|**

Where:

* **Σ:** A Greek symbol that means “sum”
* **yi:** The observed value for the ith observation
* **xi:** The predicted value for the ith observation
* **n:** The total number of observations
* **MSE:** The mean squared error (MSE) is a common way to measure the prediction accuracy of a model. It is calculated as:

**MSE**= (1/n) \* Σ (actual – prediction) 2

Where:

* **Σ** – a fancy symbol that means “sum”
* **n** – sample size
* **actual** – the actual data value
* **forecast** – the predicted data value

**CHAPTER 5**

**SYSTEM REQUIREMENTS**

**5.1 HARDWARE REQUIREMENTS:**

* System : Pentium IV 2.4 GHz
* Hard Disk : 200 GB
* Mouse : Logitech.
* Keyboard : 110 keys enhanced
* Ram : 4GB

**5.2 SOFTWARE REQUIREMENTS:**

* O/S : Windows 7.
* Language : Python
* Front End : Anaconda Navigator – Spyder

**5.3 SOFTWARE DESCRIPTION:**

**5.3.1 Python**

Python is one of those rare languages which can claim to be both *simple* and powerful. You will find yourself pleasantly surprised to see how easy it is to concentrate on the solution to the problem rather than the syntax and structure of the language you are programming in. The official introduction to Python is Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. I will discuss most of these features in more detail in the next section.

## **5.3.2 Features of Python**

### **Simple**

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself.

### **Easy to Learn**

As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.

### **Free and Open Source**

Python is an example of a FLOSS (Free/Libré and Open Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.

### **High-level Language**

When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, etc.

### **Portable**

Due to its open-source nature, Python has been ported to (i.e. changed to make it work on) many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features.

You can use Python on GNU/Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE and PocketPC!

You can even use a platform like [Kivy](http://kivy.org) to create games for your computer and for iPhone, iPad, and Android.

### **Interpreted**

This requires a bit of explanation.

A program written in a compiled language like C or C++ is converted from the source language i.e. C or C++ into a language that is spoken by your computer (binary code i.e. 0s and 1s) using a compiler with various flags and options. When you run the program, the linker/loader software copies the program from hard disk to memory and starts running it.

Python, on the other hand, does not need compilation to binary. You just run the program directly from the source code. Internally, Python converts the source code into an intermediate form called bytecodes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc. This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!

### **Object Oriented**

Python supports procedure-oriented programming as well as object-oriented programming. In procedure-oriented languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In object-oriented languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.

### **Extensible**

If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your program in C or C++ and then use it from your Python program.

### **Embeddable**

You can embed Python within your C/C++ programs to give scripting capabilities for your program's users.

### **Extensive Libraries**

The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, FTP, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces), and other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the Batteries Included philosophy of Python.

Besides the standard library, there are various other high-quality libraries which you can find at the Python Package Index.

**5.4 TESTING PRODUCTS:**

System testing is the stage of implementation, which aimed at ensuring that system works accurately and efficiently before the live operation commence. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an error. A successful test is one that answers a yet undiscovered error.

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. . A series of tests are performed before the system is ready for the user acceptance testing. Any engineered product can be tested in one of the following ways. Knowing the specified function that a product has been designed to from, test can be conducted to demonstrate each function is fully operational. Knowing the internal working of a product, tests can be conducted to ensure that “al gears mesh”, that is the internal operation of the product performs according to the specification and all internal components have been adequately exercised.

**5.4.1 UNIT TESTING:**

Unit testing is the testing of each module and the integration of the overall system is done. Unit testing becomes verification efforts on the smallest unit of software design in the module. This is also known as ‘module testing’.

The modules of the system are tested separately. This testing is carried out during the programming itself. In this testing step, each model is found to be working satisfactorily as regard to the expected output from the module. There are some validation checks for the fields. For example, the validation check is done for verifying the data given by the user where both format and validity of the data entered is included. It is very easy to find error and debug the system.

**5.4.2 INTEGRATION TESTING:**

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined, may not produce the desired major function. Integrated testing is systematic testing that can be done with sample data. The need for the integrated test is to find the overall system performance. There are two types of integration testing. They are:

i) Top-down integration testing. ii) Bottom-up integration testing.

**5.4.3 TESTING TECHNIQUES/STRATEGIES:**

* **WHITE BOX TESTING:**

White Box testing is a test case design method that uses the control structure of the procedural design to drive cases. Using the white box testing methods, we

Derived test cases that guarantee that all independent paths within a module have been exercised at least once.

* **BLACK BOX TESTING:**

1. Black box testing is done to find incorrect or missing function
2. Interface error
3. Errors in external database access
4. Performance errors.
5. Initialization and termination errors

In ‘functional testing’, is performed to validate an application conforms to its specifications of correctly performs all its required functions. So this testing is also called ‘black box testing’. It tests the external behaviour of the system. Here the engineered product can be tested knowing the specified function that a product has been designed to perform, tests can be conducted to demonstrate that each function is fully operational.

**5.4.4 SOFTWARE TESTING STRATEGIES**

**VALIDATION TESTING:**

After the culmination of black box testing, software is completed assembly as a package, interfacing errors have been uncovered and corrected and final series of software validation tests begin validation testing can be defined as many,

But a single definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the customer

**USER ACCEPTANCE TESTING:**

User acceptance of the system is the key factor for the success of the system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system at the time of developing changes whenever required.

**OUTPUT TESTING**:

After performing the validation testing, the next step is output asking the user about the format required testing of the proposed system, since no system could be useful if it does not produce the required output in the specific format. The output displayed or generated by the system under consideration. Here the output format is considered in two ways. One is screen and the other is printed format. The output format on the screen is found to be correct as the format was designed in the system phase according to the user needs. For the hard copy also output comes out as the specified requirements by the user. Hence the output testing does not result in any connection in the system.

**CHAPTER 6**

**CONCLUSION**

We conclude that, the household power consumption dataset was taken from dataset repository as input. We are developed the two deep learning algorithms such as long short term memory and gate recurrent unit. Finally, the result shows that some performance metrics such as mae, mse and rmse.

**CHAPTER 7**

**FUTURE ENHANCEMENT**

In the future, we should like to hybrid the two different machine learning. In future, it is possible to provide extensions or modifications to the proposed clustering and classification algorithms to achieve further increased performance. Apart from the experimented combination of data mining techniques, further combinations and other clustering algorithms can be used to improve the performance.

**CHAPTER 8**

**SAMPLE CODING**

#==================== IMPORT REQUIRED LIBRARIES =============================

import pandas as pd

from tkinter.filedialog import askopenfilename

from keras.models import Sequential

from keras.layers import Dense, LSTM, Dropout, GRU

from keras.optimizers import SGD

from sklearn.preprocessing import MinMaxScaler

import numpy as np

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

#========================= DATA SELECTION ==================================

filename = askopenfilename()

dataframe=pd.read\_csv(filename)

print("==========================================")

print("-------------- Input Data ----------------")

print("==========================================")

print()

print(dataframe.head(20))

#============================= PREPROCESSING ==============================

#==== DROP UNWANTED COLUMNS ====

dataframe=dataframe.drop('datetime',axis=1)

#==== CHECKING MISSING VALUES ====

print("====================================================")

print("--------- Before Checking Missing Values ----------")

print("===================================================")

print()

print(dataframe.isnull().sum())

print("====================================================")

print("--------- After Checking Missing Values ----------")

print("===================================================")

print()

dataframe=dataframe.fillna(0)

print(dataframe.isnull().sum())

#=========================== DATA SPLITTING =======================

#=== TEST AND TRAIN ===

x=dataframe.drop('Global\_active\_power',axis=1)

y=dataframe.Global\_active\_power

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size = 0.2,random\_state = 42)

print("===============================================")

print("-------------- Data Splitting ----------------")

print("===============================================")

print()

print("Total Data size is :",dataframe.shape[0])

print("Total test Data size is :",x\_test.shape[0])

print("Total train Data size is :",x\_train.shape[0])

#=========================== MIN MAX SCALAR =======================

scaler = MinMaxScaler()

scaler.fit(x\_train)

data\_scaler=scaler.transform(x\_test)

#============================= CLASSIFICATION ==============================

#=== LSTM ===

X=np.expand\_dims(x\_train, axis=2)

Y=np.expand\_dims(y\_train,axis=1)

#=== lstm architecture ===

model = Sequential()

#=== lstm layers ===

model.add(LSTM(input\_shape=(x\_train.shape[0],1), kernel\_initializer="uniform", return\_sequences=True, stateful=False, units=50))

model.add(Dropout(0.2))

model.add(LSTM(5, kernel\_initializer="uniform", activation='relu',return\_sequences=False))

model.add(Dropout(0.2))

model.add(Dense(3,kernel\_initializer="uniform",activation='relu'))

model.add(Dense(1, activation='linear'))

#=== compile the model ===

model.compile(loss="mae", optimizer='adam',metrics=['mae','mse'])

model.summary()

print()

print("==================================================")

print("--------- Long Short Term Memory ----------------")

print("==================================================")

print()

#=== fitting the model ===

His=model.fit(X[0:10000],Y[0:10000], epochs = 5, batch\_size=2000, verbose = 2)

Xx=np.expand\_dims(x\_test, axis=2)

Yy=np.expand\_dims(y\_test, axis=1)

#=== evaluate the model ===

mae\_lstm1 =model.evaluate(X[0:10000],Y[0:10000], verbose=2)[1]

#=== predict the model ===

pred\_lstm = model.predict(Xx[0:10000])

print("--------- Performance Analysis for LSTM ----------------")

print()

from sklearn import metrics

mae\_lstm = metrics.mean\_absolute\_error(pred\_lstm,Yy[0:10000])

print("1.Mean Absolute Error :",mae\_lstm1)

print()

mse\_lstm = metrics.mean\_squared\_error(pred\_lstm,Yy[0:10000])

print("2.Mean Squared Error :",mse\_lstm)

print()

from math import sqrt

rmse\_lstm=sqrt(mse\_lstm)

print("3.Root Mean Squared Error :",rmse\_lstm)

# mape = metrics.mean\_absolute\_percentage\_error(pred\_lstm,Yy[0:10000])

# from sklearn.metrics import mean\_absolute\_percentage\_error

#==== GRU =====

# The GRU architecture

regressorGRU = Sequential()

# First GRU layer with Dropout regularisation

regressorGRU.add(GRU(units=50, return\_sequences=True, input\_shape=(x\_train.shape[0],1), activation='tanh'))

regressorGRU.add(Dropout(0.2))

# Second GRU layer

regressorGRU.add(GRU(units=50, return\_sequences=True, input\_shape=(x\_train.shape[0],1), activation='tanh'))

regressorGRU.add(Dropout(0.2))

# Third GRU layer

regressorGRU.add(GRU(units=50, return\_sequences=True, input\_shape=(x\_train.shape[0],1), activation='tanh'))

regressorGRU.add(Dropout(0.2))

# Fourth GRU layer

regressorGRU.add(GRU(units=50, activation='tanh'))

regressorGRU.add(Dropout(0.2))

# The output layer

regressorGRU.add(Dense(units=1))

# Compiling the RNN

regressorGRU.compile(optimizer=SGD(lr=0.01, decay=1e-7, momentum=0.9, nesterov=False),loss='mae',metrics=['mae','mse'])

print("==================================================")

print("-------------------- GRU -------------------------")

print("==================================================")

print()

# Fitting to the training set

regressorGRU.fit(X[0:10000],Y[0:10000],epochs=10,batch\_size=150, verbose=2)

pred\_gru = regressorGRU.predict(Xx[0:10000])

mae\_lstm =regressorGRU.evaluate(X[0:10000],Y[0:10000], verbose=2)[1]

print("--------- Performance Analysis for GRU ----------------")

print()

from sklearn import metrics

mae\_gru = metrics.mean\_absolute\_error(pred\_gru,Yy[0:10000])

print("1.Mean Absolute Error :",mae\_gru)

print()

mse\_gru = metrics.mean\_squared\_error(pred\_gru,Yy[0:10000])

print("2.Mean Squared Error :",mse\_gru)

print()

from math import sqrt

rmse\_gru=sqrt(mse\_gru)

print("3.Root Mean Squared Error :",rmse\_gru)

#============================= COMPARISION ==============================

if mae\_lstm1<mae\_gru:

print("=========================")

print("LSTM is efficeint")

print("=========================")

else:

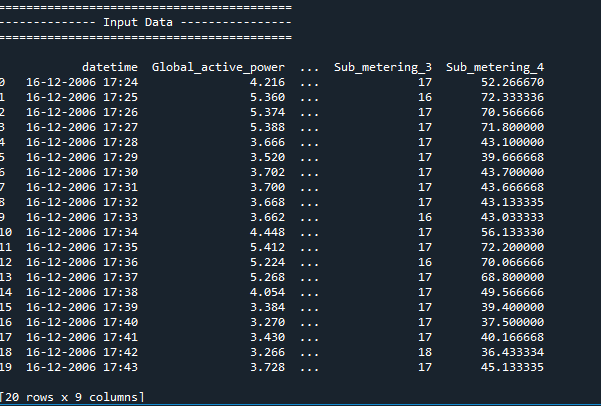
print("=========================")

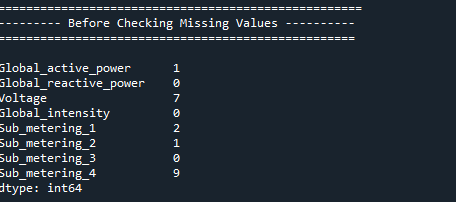
print("GRU is efficeint")

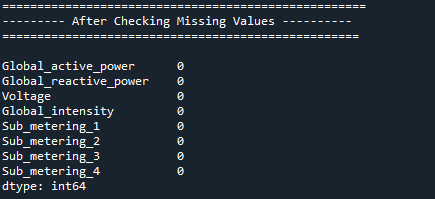
print("=========================")

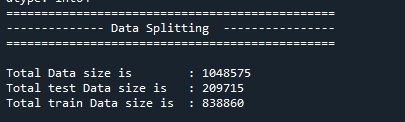
**CHAPTER 9**

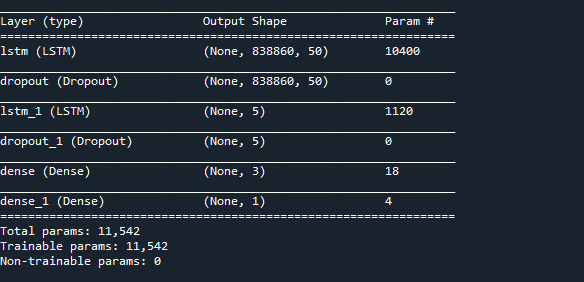
**SAMPLE SCREENSHOTS**

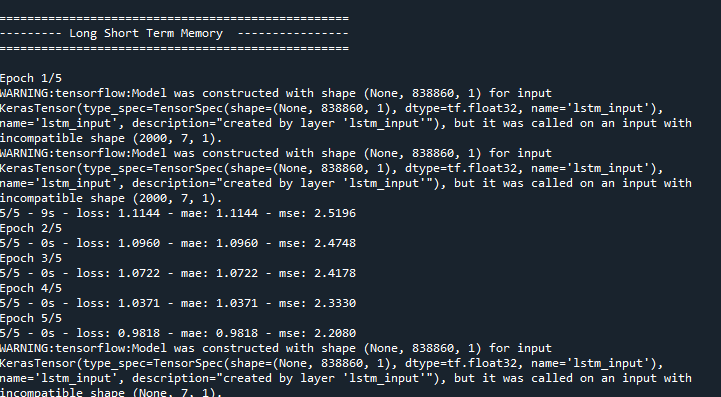


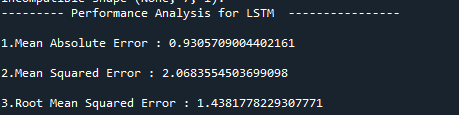


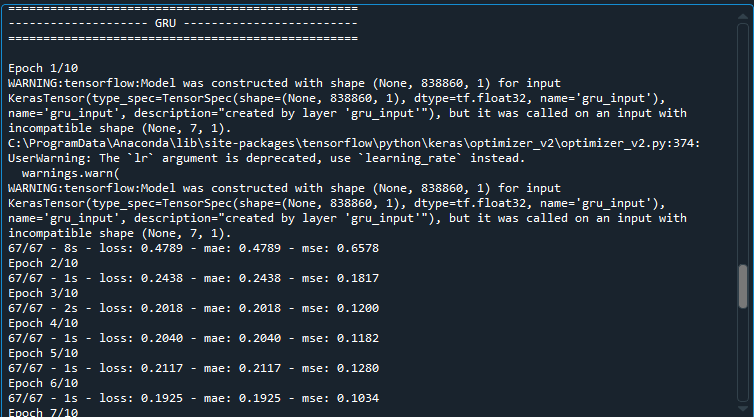


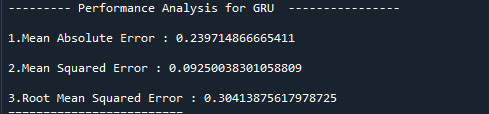














**CHAPTER 10**

**REFERENCES**

[1] Yeboah, Samuel Asuamah, Manu Ohene, and T. B. Wereko. ”Forecasting aggregate and disaggregate energy consumption using arima models: a literature survey.” Journal of Statistical and Econometric

Methods 1.2 (2012): 71-79.

[2] Miao, Junwei. ”The Energy Consumption Forecasting in China Based on ARIMA Model.” (2015).

[3] Pao, Hsiao-Tien, and Chung-Ming Tsai. ”Modeling and forecasting the CO 2 emissions, energy consumption, and economic growth in Brazil.” Energy 36.5 (2011): 2450-2458.Pao, Hsiao-Tien, and Chung-Ming Tsai. ”Modeling and forecasting the CO 2 emissions, energy consumption, and economic growth in Brazil.” Energy 36.5 (2011): 2450-2458.

[4]Crompton, Paul, and Yanrui Wu. ”Energy consumption in China: past trends and future directions.”Energy economics 27.1 (2005): 195-208.

[5]Yazdan, Gudarzi Farahani, Varmazyari Behzad, and Moshtaridoust Shiva. ”Energy consumption inIran: past trends and future directions.” Procedia-Social and Behavioral Sciences 62 (2012): 12-17.

[6]zer, Mustafa, and Charles Mensah. ”A Bivariate Causality between Energy Consumption and Economic Growth in Ghana.”

[7]Owda, Hasan MH, et al. ”Using Artiﬁcial Neural Network Techniques for Prediction of ElectricEnergy Consumption.” arXiv preprint arXiv:1412.2186(2014) Tamizharasi, G., S. Kathiresan, and K. S. Sreenivasan. ”Energy forecasting using artiﬁcial neural networks.” International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering 3.3 (2014): 7568-7576.

[8]Kandananond, Karin. ”Forecasting electricity demand in Thailand with an artiﬁcial neural network approach.” Energies 4.8 (2011): 1246-1257.

[9]Wang, Xiping, and Ming Meng. ”A Hybrid Neural Network and ARIMA Model for Energy Consumption Forcasting.” Journal of computers 7.5 (2012): 1184-1190.

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

[11] Naser Farag Abed1 and Milan M.Milosavljevic 1,2 1 Singidunum University Belgrade, 11000, Serbia 2 Schoolof 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 – 79

[12] Y. Wang, Q. Chen, M. Sun, C. Kang, and Q. Xia, "An ensemble forecasting method for the aggregated load with subprofiles," IEEE Transactions on Smart Grid, vol. 9, no. 4, pp. 3906-3908, 2018.

[13] Y.-F. Zhang and H.-D. Chiang, "Enhanced ELITELoad: A Novel CMPSOATT Methodology Constructing Short-Term Load Forecasting Model for Industrial Applications," IEEE Transactions on Industrial Informatics, 2019.

[14] P. Zhuang and H. Liang, "Hierarchical and decentralized stochastic energy management for smart distribution systems with high BESS penetration," IEEE Transactions on Smart Grid, vol. 10, no. 6, pp. 6516-6527, 2019.

[15] G. Černe, D. Dovžan, and I. J. I. T. o. I. E. Škrjanc, "Short-term load forecasting by separating daily profiles and using a single fuzzy model across the entire domain," vol. 65, no. 9, pp. 7406-7415, 2018.