Appliance Energy Prediction in a Low Energy House using Machine Learning and Deep Learning

Submitted by:

Ahmad Yasin (F2019019041)

Ahmar Amin (F2019019046)

Submitted to:

Sir Khalid Ijaz

Department of Electrical Engineering

School of Engineering (SEN)

University of Management and Technology, Lahore.

Appliance Energy Prediction in a Low Energy House using Machine Learning and Deep Learning

The report is submitted to the faculty of department of electrical engineering of University of Management and Technology, Lahore in partial fulfilment of the

Requirements of the degree of

Bachelors of Science

In

Electrical Engineering

_____

Project Advisor

_____

Director Projects

Department of Electrical Engineering

School of Engineering (SEN)

University of Management and Technology, Lahore.

Declaration

It is declared that the work submitted in this report is completely original and prepared by us, except where explicitly stated otherwise. In addition, this work has not been submitted for any other degree or professional certification attainment.

Signed: _____

Signed: _____

Dated: _____

Appliance Energy Prediction in a Low Energy House using Machine Learning and Deep Learning

Objectives:

To discuss machine learning and deep learning data driven predictive model for the electrical load forecasting of energy appliances in home.

To present better accuracy and performance parameters such as RMSE, R^2, MAPE for load forecasting.

Undertaken by:

It is claimed that the work is completely done by ourselves. The project is accomplished by our skills, efforts, and knowledge.

Supervised by:

Khalid Ijaz

Starting Date:

10-10-2022

Completion Date:

28-05-2023

ACKNOWLEDGEMENT

We would like to express my special thanks of gratitude to my teacher Sir Khalid Ijaz who gave me the golden opportunity to do this wonderful project on Machine Learning and Deep Learning. Your useful advice and suggestions were really helpful to me during the project's completion. In this aspect, we are eternally grateful to you. We would like to acknowledge that this project was completed entirely by us and not by someone else. This project also helped us in doing a lot of Research. Secondly we would also like to thank my parents and friend Faizan Zahid who helped me a lot in finalizing this project within the limited time frame.

Table of Contents:

ABSTRACT

The rising environmental concerns and the necessity for sustainable living have led to a significant increase in the demand for energy-efficient housing. As a result, accurately predicting the energy consumption of appliances has become crucial for optimizing energy usage and reducing costs. This research paper investigates the utilization of machine learning and deep learning methods to forecast energy usage of household appliances in a low-energy residence. The primary aim is to improve energy efficiency by achieving higher prediction accuracy and optimization.

CHAPTER 1:

INTRODUCTION

1.1. Introduction:

Low-energy houses, also known as energy-efficient or green houses, are designed to minimize energy consumption while ensuring a comfortable living environment. Predicting appliance energy consumption is crucial in optimizing energy usage and reducing waste. The project seeks to investigate the use of deep learning and machine learning methods for predicting appliance energy use in low-energy homes.

Significance of Appliance Energy Prediction: Accurate appliance energy prediction provides valuable insights into energy consumption patterns, enabling homeowners to make informed decisions about their energy usage. By understanding when and how appliances consume energy, occupants can adjust their behaviors and take advantage of optimal timeframes for energy-intensive tasks. This knowledge also allows for the implementation of energy management systems that automatically regulate appliance usage, leading to increased efficiency.

Machine Learning for Appliance Energy Prediction: Machine learning algorithms can analyze historical energy consumption data, along with relevant features such as weather conditions, occupancy patterns, and appliance characteristics, to predict future energy consumption. Techniques such as decision trees, regression, and random forests can be employed to build models that capture complex relationships between variables and achieve accurate predictions.

Deep Learning for Appliance Energy Prediction: Deep learning, a subset of machine learning, has gained significant attention for its ability to extract intricate patterns from large datasets. DNN includes recurrent neural networks and CNN, can process sequential information to predict appliance energy consumption. DL models have demonstrated promising results in various domains and enhance the accuracy of energy predictions in low-energy houses.

1.2. Motivation:

The motivation for appliance energy prediction in low-energy houses using machine and deep learning techniques arises from several important factors:

- Energy Efficiency
- Cost Savings
- Integration of Renewable Energy
- Load Balancing and Demand Management
- Promoting Sustainable Living
- Smart Home Integration
- Research and Innovation

Overall, the motivation for appliance energy prediction in low-energy houses using machine and deep learning techniques is driven by the desire to improve energy efficiency, achieve cost savings, integrate renewable energy, manage energy demand, promote sustainable living, enable smart home integration, and foster research and innovation in the field.

1.3. Problem Statement:

The goal of our project is to create a predictive model that uses the ML and DL techniques to forecast the load of a houses and industrial energy system with a one-hour ahead prediction. By accurately predicting the load, the aim is to create a cost-effective energy system for consumers.

1.4. Scope:

The potential scope of our project is broad and encompasses energy optimization, cost savings, demand management, resource planning, environmental impact reduction, anomaly detection, load balancing, behavioral insights, integration with renewable energy, predictive maintenance, and continuous learning.

1.5. Report Organization:

It contains the literature that was reviewed in order to identify the work that has been previously done in this regard and the methods that were adopted to achieve those results. In the third chapter, the methodology that has been chosen by us is reflected and the steps are demonstrated. The fourth chapter contains implementations that were done on software, the

designed algorithms, flow charts. The next chapter consists the results and discussions that were drawn from the project. The last chapter contains conclusion and the future work that can be done.

CHAPTER 2:

LITERATURE REVIEW

2.1. Literature Review:

These literature review contains the articles bearing the same topic as this report, in which the researcher has made an excellent effort to produce the paper.

Load forecasting plays a critical role in the planning and operation of electric utilities. Its purpose is to accurately predict the magnitude and geographic distribution of electric load across various time periods, typically on an hourly basis, within a given planning horizon. Although the hourly total system load is typically the focus of load forecasting, Grose and Galina have noted that it also includes the forecasting of the system load's daily, weekly, and monthly values as well as its peak load and energy. According to the length of the planning horizon, Srinivsn and Le have further divided load forecasting into three categories: short-term load forecasting, which covers time frames up to one day, medium-term load forecasting, which covers one day to one year, and long-term load forecasting, which covers one to ten years.

Within electric utility firms, accurate load forecasting offers significant potential for cost reductions. In order to optimise operations and decision-making processes, such as dispatch, unit commitment, fuel allocation, and offline network analysis, Bunn and Farmer have emphasised the use of load forecasting. This can result in significant cost reductions. As forecasting errors can significantly affect the economy of operations and control, the accuracy of load projections has a significant impact on power system operations. Both positive and negative forecasting errors might result in higher operating expenses, according to Hada and Mutoo's research. Hobbes et al. have calculated the financial worth of improved short-term load forecasting, showing that even a 1% reduction in the average forecast error can result in savings of hundreds of thousands of dollars.
The distribution of system load is unpredictable and non-stationary, and it depends on a number of variables, including the state of the economy, the time of day, the season, the weather, and random impacts. Numerous functional models and estimating techniques with varied degrees of complexity have been presented in order to increase the accuracy of load forecasting. Early studies on electric load forecasting methods were done by Mathewan and Necholsen, and studies on load demand modelling and forecasting have been done by Aba El-Maged and Senha, Bunn and Farmer, and Griss and Galina. More recently, a thorough study on methods for forecasting electric load was carried out by Mogham and Rahmn.

The paper titled Energy Disaggregation via Discriminative Sparse by Hatmnn et al. explores the application of discriminative sparse coding for the purpose of energy disaggregation,

specifically targeting the disaggregation of energy consumption at the appliance level. The study employs machine learning methods to predict the energy usage of individual appliances using aggregated energy consumption data as input.

Non-Intrusive Appliance Load Monitoring Review and Outlook by Btra et al. provides an overview of techniques for non-intrusive appliance load monitoring, including machine learning approaches, to achieve energy disaggregation. It discusses various algorithms employed for predicting the energy consumption of individual appliances.

Appliance-Level Power Demand Forecasting Using Recurrent Neural Networks by Pason et al., the authors focus on applying RNN, specifically periodic long short term memory networks, for forecasting power demand at the appliance level in residential settings.

Smart Home Energy Management System for Demand Response a page by Ghni ett el., the authors provide an overview of smart home energy management systems and their application in demand response scenarios.

Predictive Energy Management for Smart Homes Using Deep Learning by Kuley ut al, a deep learning framework called Deep Energy is proposed for predicting energy consumption in smart homes. The framework employs deep neural networks to forecast electricity usage at the appliance level, enabling energy optimization and demand response strategies.

Energy Forecasting and Disaggregation Methods in Smart Grids A Review" by Tunas offers a comprehensive review of energy forecasting and disaggregation methods within the context of smart grids. The review covers machine learning techniques, including deep learning algorithms, employed for energy prediction and load disaggregation at the appliance level.

Numerous studies have looked into how to predict household appliance energy use. For instance, Busu proposed a model to forecast household appliance energy consumption for the upcoming hour and the upcoming 24 hours utilising a variety of machine learning methods, such as the decision tree algorithm, the decision table classifier, and the Bayesian network. Their strategy emphasised the significance of choosing the best regression model for a given dataset by incorporating industry-specific knowledge on energy usage and locating an appropriate data structure for the regressor.

Haber used three techniques to anticipate the on/off periods of appliances with a resolution of one hour: the histogram algorithm, the pattern search algorithm, and the Bayesian inference

algorithm. It was discovered that different algorithms performed independently of one another on the same dataset.

Cundaedo offered data-driven prediction methods for home appliance energy utilisation. Repeated cross-validation was used to train and test four statistical models: the support vector machine with radial kernel, the gradient boosting machine, the multiple linear regression, and the random forest. The GBM technique produced the greatest outcomes.

Zheo examined several SVM and artificial neural network-based models for estimating how much energy will be used by buildings. It was discovered to be difficult to choose the best model without comparing each model in the same situation.

Paton suggested artificial neural network- and case-based reasoning-based energy prediction methods. The hourly electricity usage of an institutional building was predicted using these models, with the ANN-based models consistently beating the case-based reasoning models.

Electricity consumption in residential buildings is influenced by factors such as architectural design, occupancy, the count of electric appliances, and the outdoor and indoor environment. Regression models have been employed to analyze the relationships between these factors and gain insights into their impact. Additionally, electricity demands vary between weekdays and weekends due to differences in occupancy patterns. Regression analysis is useful for identifying these dynamic load patterns. Numerous research studies have utilized wireless sensor data to assess environmental conditions and digital meter data to analyze energy use loads, aiming to develop accurate load demand models for predicting future electricity usage in residential buildings through data-driven analysis.

Moreover, studies have investigated occupant behavior to assess the energy efficiency of appliances used in homes and offices, employing regression analysis check load patterns corresponding to different occupant behaviors. Some studies have also focused on predicting occupancy numbers by analyzing appliance usage patterns.

CHAPTER 3:

METHODOLOGY

3.1. Methodology:

We developed three energy prediction models, consisting of two machine learning-based models and one deep learning model. They are explained below:

3.1.1. LSTM:

The process of constructing a Long Short-Term Memory model, a type of recurrent neural network designed for analyzing sequence data, involves several stages. Below is a general outline of the methodology:

Data preparation: To begin the analysis, obtain the dataset that includes the sequential data to be examined. This data could consist of time series data, text data, or any other form of data with a temporal or sequential nature. Prior to analysis, it is important to perform preprocessing on the data, which involves cleaning, normalizing, and transforming the data as required.

Data representation: To prepare the sequential data for input into the Long Short-Term Memory model, it is necessary to convert it into a suitable format. This process commonly involves representing the data as sequences of fixed-length input vectors or employing techniques such as word embeddings for text data. By converting the data into a compatible format, it becomes feasible to effectively utilize it as input for the LSTM model.

Data splitting: The training, validation, and test sets should be separated into separate portions of the dataset. The LSTM model is trained using the training set so that it can recognise patterns and connections in the data. The validation set is used to adjust hyperparameters and evaluate how well the model performed during training. The test set is saved for the trained LSTM model's final assessment, giving a fair evaluation of how well it performs on unobserved data. It is possible to evaluate the capabilities of the LSTM model effectively and dependably by splitting the dataset in this way.

Model architecture: Design the architecture of LSTM model. Determine the number of LSTM layers to units per layer, and other architectural choices such as the inclusion of additional layers to regularize the model.

Model training: Train the LSTM model using the training set. Throughout training, the model learns to capture temporal dependencies and patterns in the data. The back propagation through time algorithm is typically utilized to update the model's mass and decrease the loss function.

Model evaluation: Access the ability of the trained LSTM model on the validate set. Employ appropriate evaluation metrics based on the specific task, such as mean squared error for regression problems or accuracy, precision, recall score for classification problems.

Hyperparameter tuning: Fine-tune the hyperparameters of the model to optimize its performance. This may involve adjusting learning rates, batch sizes, the number of LSTM units, or exploring different optimization algorithms.

Model testing: To get a fair assessment of the final LSTM model's performance, evaluate it on the test set. In this step, the model's ability to generalise to new data is evaluated.

Model deployment: Once finallize the model's performance, deploy it to make predictions on new, different seque+ntial data.

Implementing an LSTM model can be achieved using deep learning libraries in Python, such as TensorFlow or PyTorch. These libraries offer tools and classes for building, training, and evaluating LSTM models. Additionally, you can leverage prebuilt LSTM architectures or employ transfer learning techniques, if applicable.

3.1.2. Linear Regression:

The process of constructing a linear regression model involves several steps. Here is a general outline of the methodology:

Data collection: Gather the dataset containing the variables you wish to analyze. This includes both the input features and the target variable.

Data preprocessing: Perform necessary preprocessing steps on the dataset, such as handling missing values, addressing outliers, and transforming variables if needed.

Dataset splitting: Divide the dataset into two subsets: the set to be trained and the set to be test. The training set is used to train the linear regression model, while it is used to evaluate performance.

Feature selection: Select relevant features which is likely to have a significant impact on the variable. This step involves exploratory data analysis and statistical techniques to identify the most influential variables.

Model training: Utilize the training set to train the linear regression model. The goal of the model is to identify the line that fits the data the best and minimises the discrepancy between the predicted and actual values of the variable.

Model evaluation: It evaluates the performance of the trained model. Common evaluation metrics for linear regression mean squared error, root mean squared error, and R-squared.

Model improvement: If the model's performance is unsatisfactory, you can iterate and improve the model by refining the feature selection, trying different transformations, or considering interactions between variables.

Model deployment: Once finalize the model's performance, deploy it to make predictions on new, unseen data.

Implementing a linear regression model can be achieved using various libraries in Python, such as scikit-learn, statsmodels, or TensorFlow. These libraries provide user-friendly functions and classes to perform the aforementioned steps.

3.1.3. SVR:

The process of constructing a Support Vector Regression model, a regression model based on Support Vector Machines, involves several steps. Here is a general outline of the methodology:

Data collection: Gather the dataset containing the variables you wish to analyze. This includes both the input features and the target variable.

Data preprocessing: Perform necessary preprocessing steps on the dataset, such as handling missing values, addressing outliers, and scaling the features. Scaling the input features to a similar range is crucial since SVR models are sensitive to feature scale.

Dataset splitting: Divide the training set and the test set in two parts. The trained set is used to train the model, while the test set is used to evaluate performance.

Feature selection/Engineering: Select relevant features which is likely to have a proper impact on the variable. This step involves exploratory data analysis and statistical techniques to identify the most influential variables. You can also create new features or apply transformations to improve the model's performance.

Model training: Train the SVR model using the training set. SVR aims to find the best hyper-plane that fits the training data while minimizing margin violations.

Model evaluation: Utilising the test set, assess the trained SVR model's performance. The effectiveness of the model can be evaluated using common metrics for regression models including mean squared error, root mean squared error, and R^2

Model improvement: If the performance is unsatisfactory, you can iterate and improve the model by adjusting hyperparameters, exploring different kernels, or applying regularization techniques. Cross-validation can also be employed to effectively tune the hyperparameters.

Model deployment: If it is ok use it to make predictions on different, unseen data.

Implementing an SVR model can be achieved using various Python libraries, such as scikit-learn. Scikit-learn provides a brief tools for machine learning, including the SVR class for building SVR models. It offers flexibility in terms of kernel selection, hyperparameter tuning, and customization of the model.

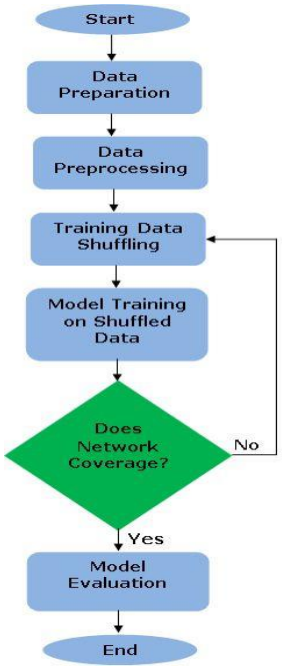


Figure 1: Model Implementation

CHAPTER 4

SOFTWARE IMPLEMENTATION

4.1. Software Implementation:

Jupyter Notebook is an open-source web application that facilitates the creation and sharing of documents incorporating live code, equations, visualizations, and narrative text. It is widely employed for various tasks, including data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and numerous other applications. With its versatility and interactive nature, Jupyter Notebook serves as a valuable tool for data analysis and computational workflows in diverse fields.

To install Jupyter Notebook, you can follow these steps:

Install Python: Begin by installing Python, as Jupyter Notebook relies on it as its underlying programming language.

Install Jupyter Notebook: After installing Python, proceed to install Jupyter Notebook itself. You can accomplish this by using package managers such as pip or conda, which are commonly used for Python package installations. By the command "**pip install jupyter**" will download and install Jupyter notebook.

Launch Jupyter Notebook: Once the installation is complete, you can launch Jupyter Notebook by executing a command in your terminal or command prompt "**Jupyter notebook**". This will start a local web server and open a new browser window or tab, displaying the Jupyter Notebook interface.

By following these steps, you will be able to install and launch Jupyter Notebook, enabling you to create interactive documents with code, visualizations, and explanations.

Pandas is a Python library that is open-source and offers efficient tools for data manipulation and analysis. It is extensively utilized for tasks like data cleaning, transformation, exploration, and visualization. Pandas is built on top of NumPy, a widely-used numerical computing library in Python, and provides additional functionalities that are specifically designed for working with structured data.

To install pandas, you can use "**pip install pandas**"

NumPy also known as Numerical Python, is an essential library for numerical computing in the Python programming language. It offers robust array objects and a comprehensive collection of mathematical functions, enabling efficient manipulation of large multidimensional arrays and matrices. To install this library simple use "**pip install numpy**"

TensorFlow is an open-source deep learning workplace. Its purpose is to simplify the development and implementation of machine learning models, particularly those that rely on neural networks. TensorFlow provides a versatile architecture that enables users to define and train a wide range of machine learning models, including both basic linear regression models and intricate deep neural networks. Use "**pip install tensorflow**" to install and download the library of CPU version.

Keras is a Python-based open-source deep learning framework that offers a user-friendly and intuitive interface. It provides a high-level API for designing, constructing, training, and evaluating deep learning models. Keras prioritizes ease of use, modularity, and extensibility, making it suitable for both novice users and experienced practitioners in the field of deep learning. By leveraging Keras, developers can efficiently work with deep learning models, regardless of their level of expertise.

Scikit-learn as sklearn, is a widely used open-source machine learning library for Python. It offers an extensive tools and algorithms for various tasks including classification, regression, clustering, and dimensionality reduction. Scikit-learn is built on other popular scientific computing libraries in Python ensuring seamless integration within the Python data science ecosystem.

matplotlib.pyplot often referred to as plt, is a module within the widely used matplotlib library for Python. It offers a comprehensive set of functions and utilities for generating diverse types of plots, charts, and visualizations.

4.2. Algorithm Explanation:

In [4]:
```
from keras.layers import BatchNormalization
from keras.layers import Input
from keras.layers.convolutional import Conv1D, MaxPooling1D,MaxPooling2D
model = models.Sequential()
model.add(Conv1D(filters=128, kernel_size=1, activation='relu',input_shape=(11,1)))
#model.add(TimeDistributed(MaxPooling1D(pool_size=1,strides=1, padding='valid')))
#model.add(TimeDistributed(MaxPooling1D(pool_size=1,strides=1, padding='valid')))
#model.add(Dropout(0.1))
model.add(MaxPooling1D(pool_size=1,strides=1, padding='valid'))
#model.add(Dropout(0.1))
#model.add(TimeDistributed(MaxPooling1D(pool_size=2,strides=1, padding='valid')))
#model.add(Dropout(0.1))
#model.add(Flatten())
#model.add(RepeatVector(5629))
model.add((LSTM(8,activation='linear')))
#model.add(Flatten())
#model.add(GRU(300))
#model.add(Dense(150,activation='linear'))
#model.add(Dense(75,activation='linear'))
#model.add(Dropout(0.1))
#model.add(Bidirectional(LSTM(300)))
model.add(Dense(1,activation='linear'))
```

Figure 2: Machine Learning Algorithm

In [40]:
```
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras import models
from keras.applications import DenseNet121
from keras.layers.core import Dense, Activation, Flatten
from keras.layers import LSTM,GRU
```

In [41]:
```
from keras.layers import BatchNormalization
from keras.layers import Input
from keras.layers.convolutional import Conv1D, MaxPooling1D,MaxPooling2D
model = models.Sequential()
model.add(LSTM(21,return_sequences=True, dropout=1,
input_shape=(train_X.shape[1], train_X.shape[-1])))

#model.add(Flatten())
#model.add(GRU(300))
#model.add(Dense(150,activation='linear'))
#model.add(Dense(75,activation='linear'))
#model.add(Dropout(0.1))
#model.add(Bidirectional(LSTM(300)))
model.add(Dense(1,activation='linear'))
```

In [42]:
```
from keras import optimizers

sgd = optimizers.Adam(lr=0.003)
model.compile(sgd,loss='mse')

model.summary()
```

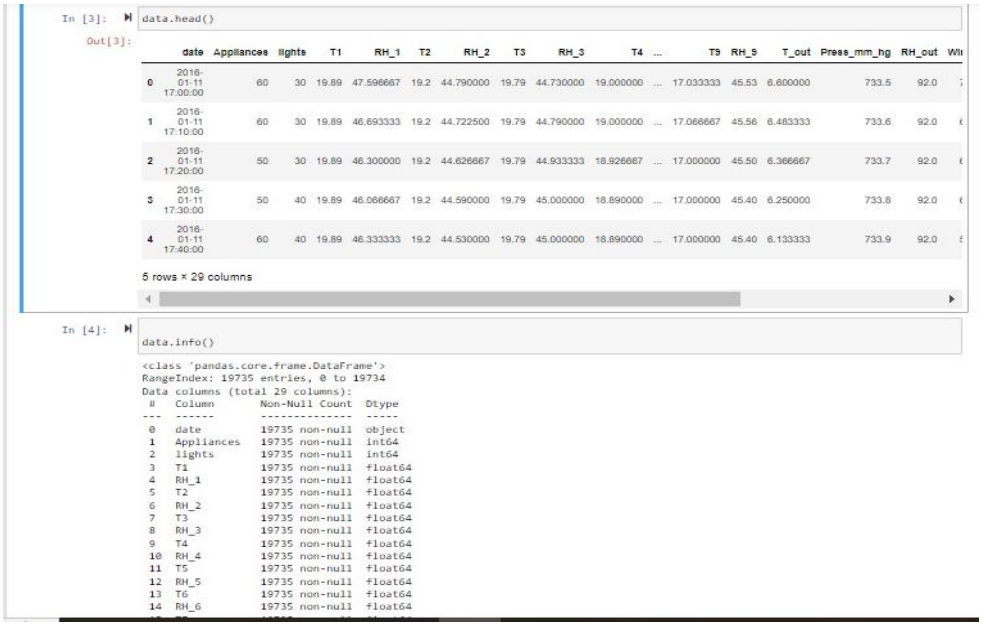
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 21, 21)            1932

 dense (Dense)               (None, 21, 1)             22

=================================================================
```

Figure 3: Deep Learning Algorithm

CHAPTER 5

RESULTS AND DISCUSSION

5.1. Results:



In [3]:
```
data.head()
```

Out[3]:

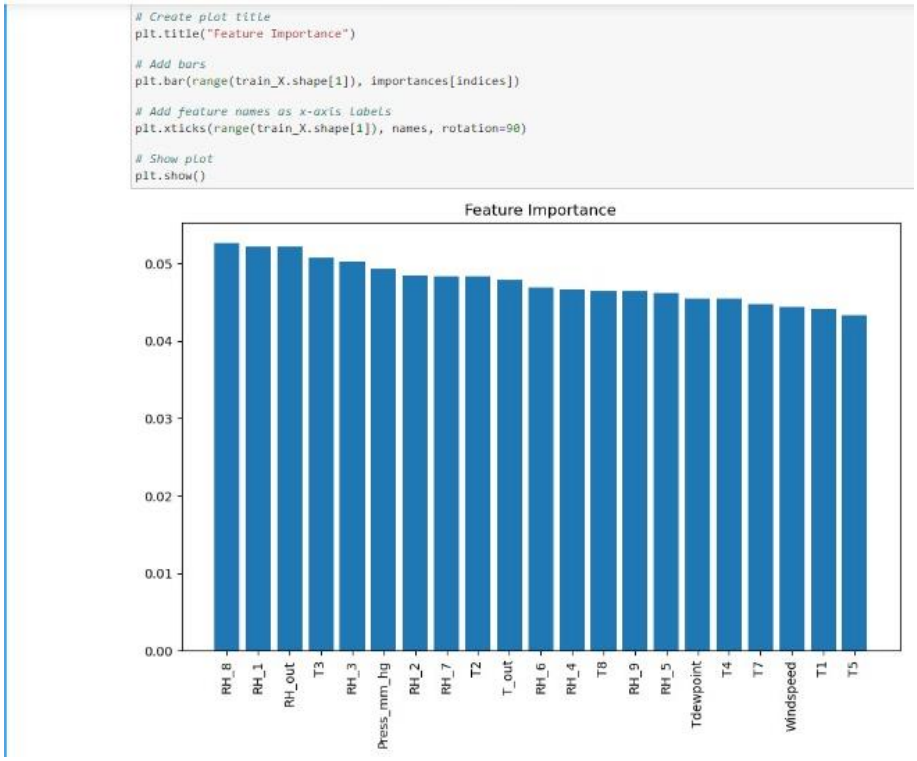
	date	Appliances	lights	T1	RH_1	T2	RH_2	T3	RH_3	T4	...	T9	RH_9	T_out	Press_mm_hg	RH_out	Wi
0	2016-01-11 17:00:00	60	30	19.89	47.596667	19.2	44.790000	19.79	44.730000	19.000000	...	17.033333	45.53	6.600000	733.5	92.0	7
1	2016-01-11 17:10:00	60	30	19.89	46.693333	19.2	44.722500	19.79	44.790000	19.000000	...	17.066667	45.56	6.483333	733.6	92.0	6
2	2016-01-11 17:20:00	50	30	19.89	46.300000	19.2	44.626667	19.79	44.933333	18.926667	...	17.000000	45.50	6.366667	733.7	92.0	6
3	2016-01-11 17:30:00	50	40	19.89	46.066667	19.2	44.590000	19.79	45.000000	18.890000	...	17.000000	45.40	6.250000	733.8	92.0	6
4	2016-01-11 17:40:00	60	40	19.89	46.333333	19.2	44.530000	19.79	45.000000	18.890000	...	17.000000	45.40	6.133333	733.9	92.0	5

5 rows × 29 columns

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19735 entries, 0 to 19734
Data columns (total 29 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   date         19735 non-null  object
 1   Appliances   19735 non-null  int64
 2   lights       19735 non-null  int64
 3   T1           19735 non-null  float64
 4   RH_1         19735 non-null  float64
 5   T2           19735 non-null  float64
 6   RH_2         19735 non-null  float64
 7   T3           19735 non-null  float64
 8   RH_3         19735 non-null  float64
 9   T4           19735 non-null  float64
 10  RH_4         19735 non-null  float64
 11  T5           19735 non-null  float64
 12  RH_5         19735 non-null  float64
 13  T6           19735 non-null  float64
 14  RH_6         19735 non-null  float64
```

Figure 4: Preparation of data



```
# Create plot title
plt.title("Feature Importance")

# Add bars
plt.bar(range(train_X.shape[1]), importances[indices])

# Add feature names as x-axis labels
plt.xticks(range(train_X.shape[1]), names, rotation=90)

# Show plot
plt.show()
```


Figure 5: Important Features of Machine Learning Models

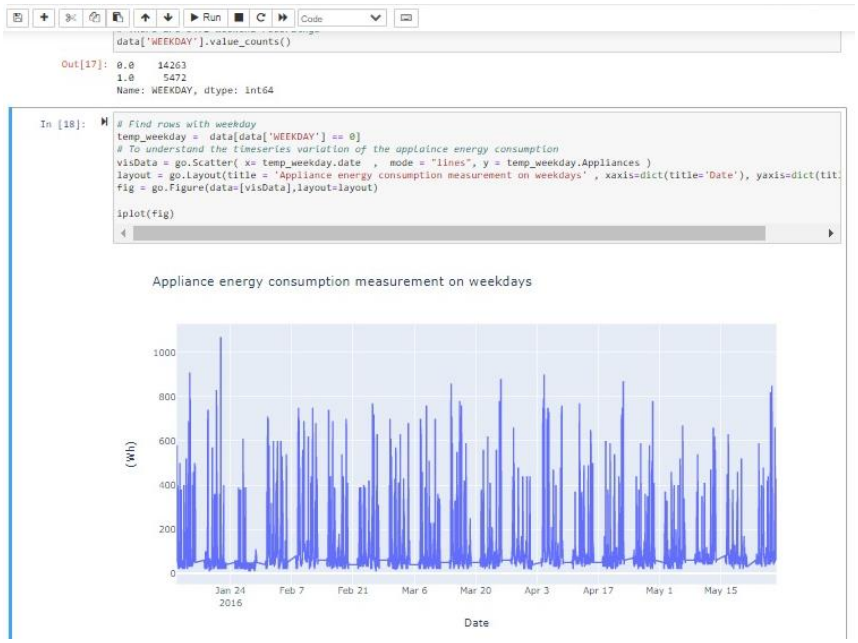


Figure 6: Iplot of Appliance Energy Consumption Measurement for Deep Learning Model

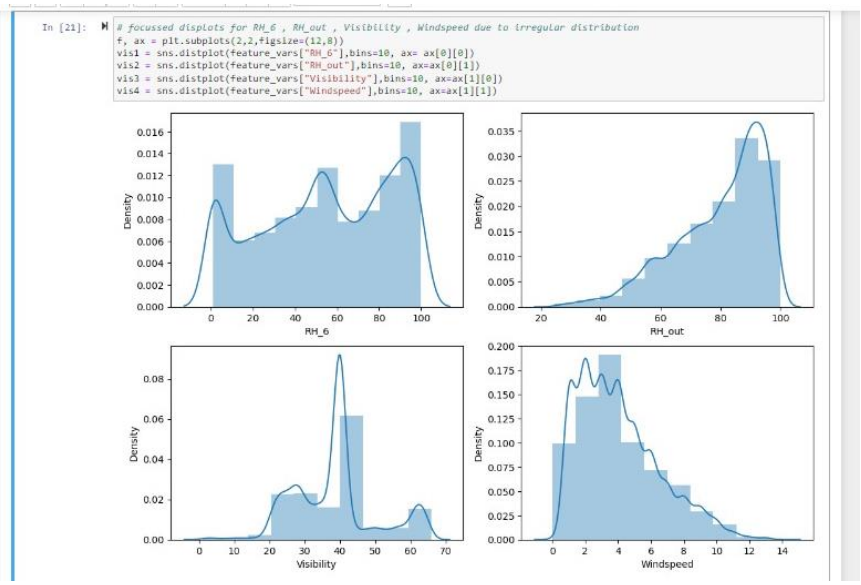


Figure 7: Features Selection

In [41]:
```
from keras.layers import BatchNormalization
from keras.layers import Input
from keras.layers.convolutional import Conv1D, MaxPooling1D,MaxPooling2D
model = models.Sequential()
model.add(LSTM(21,return_sequences=True, dropout=1,
input_shape=(train_X.shape[1], train_X.shape[-1])))

#model.add(Flatten())
#model.add(GRU(300))
#model.add(Dense(150,activation='linear'))
#model.add(Dense(75,activation='linear'))
#model.add(Dropout(0.1))
#model.add(Bidirectional(LSTM(300)))
model.add(Dense(1,activation='linear'))
```

In [42]:
```
from keras import optimizers

sgd = optimizers.Adam(lr=0.003)
model.compile(sgd,loss='mse')

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 21, 21)            1932

 dense (Dense)               (None, 21, 1)             22

=================================================================
Total params: 1,954
Trainable params: 1,954
Non-trainable params: 0
_________________________________________________________________
```

In [43]:
```
from keras.callbacks import EarlyStopping, ModelCheckpoint

earlystop = EarlyStopping(monitor ='loss', min_delta =0 ,patience = 50)
```

In [44]:
```
import time

start=time.time()

history= model.fit(train_X, train_y, validation_split=0.2, batch_size= 500, epochs= 200, verbose= 1, callbacks= [earlystop],
```

Figure 8: Model Implementation

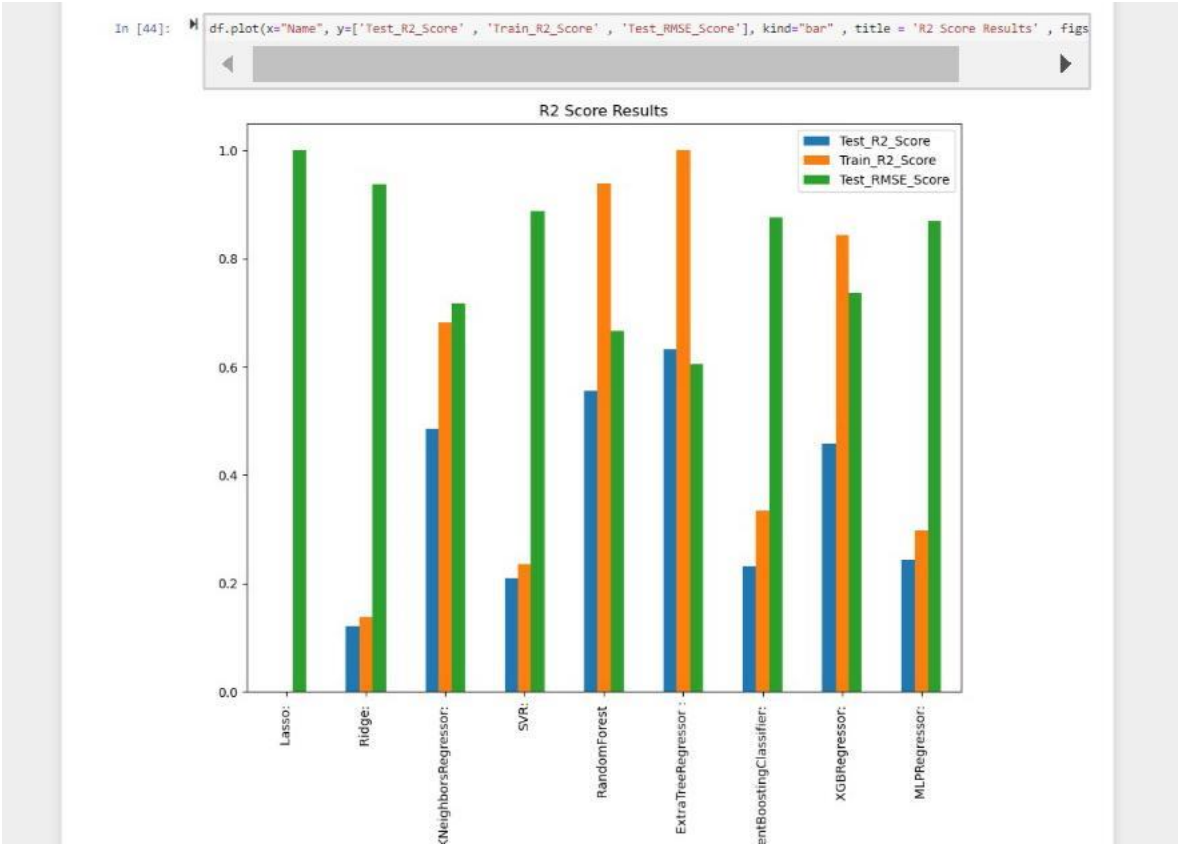


Figure 9: Parameters Calculation

Figure 10: Final Result

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1. Summary:

Data Collection: Data is collected from various sensors and smart meters installed in the low-energy house. These sensors measure energy consumption, temperature, humidity, occupancy, and other relevant variables. Data is collected at regular intervals, such as every few minutes or hourly.

Feature Engineering: The collected data undergoes preprocessing and transformation to extract meaningful features for the machine learning and deep learning models. This process includes tasks such as data cleaning, normalization, handling missing values, and extracting relevant features from raw data.

Model Selection: Various machine learning and deep learning algorithms are explored to check the most value able model for predicting the appliance energy consumption. Commonly considered algorithms.

Training and Validation: In the chosen approach, the selected model is trained using a specific portion of the collected data referred to the training set. During the training process, the model learns the underlying patterns and relationships that exist between the input features and appliance energy consumption. To assess the working of the trained model and ensure its power to generalize to unseen data, a separate portion of the data known as the validation set is employed. The model is evaluated on the validation set, allowing for the measurement of its predictive capabilities and potential improvements. By utilizing distinct training and validation sets, it becomes possible to optimize and validate the model's performance before applying it to unseen data.

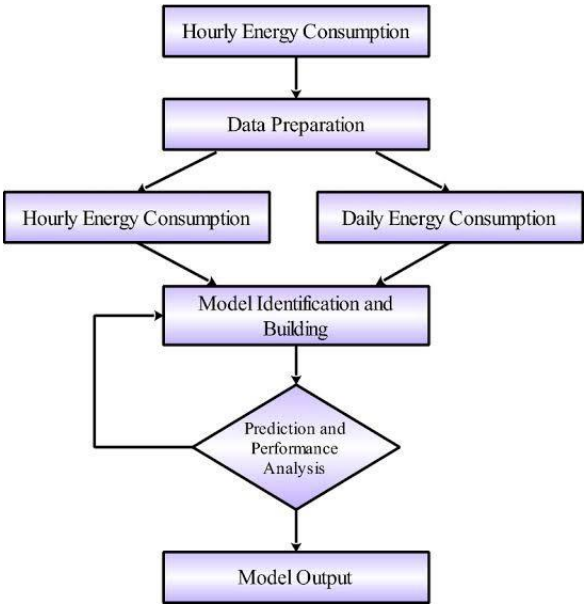
Model Evaluation: The performance of the trained model is assessed using evaluation metrics. Those metrics provide insights into the accuracy and reliability of the energy predictions made by the model.

Fine-tuning and Optimization: To optimize the performance of the model, it may undergo additional iterations of training, validation, and hyper-parameter tuning. Techniques such as cross-validation and grid search can be utilized in this process to determine the optimal hyperparameters for the model. Cross-validation involves partitioning the training data into multiple subsets and performing training and validation on different combinations of these subsets.

Deployment and Real-Time Prediction: Once the model is finalized and evaluated, it can be deployed in the low-energy house environment for real-time appliance energy prediction. The model takes input from the sensors and provides accurate predictions of future energy consumption, enabling residents to make informed decisions regarding energy usage and conservation.

Monitoring and Iterative Improvements: The deployed model is continuously monitored to ensure its ongoing performance. If any discrepancies or inaccuracies are observed, further improvements or updates to the model may be implemented.

By accurately predicting appliance energy consumption, this project aims to empower residents to optimize their energy usage, reduce waste, and contribute to the overall energy efficiency of the low-energy house.



6.2. Conclusion:

Leveraging machine learning] and deep learning techniques to predict appliance energy consumption in low-energy houses holds great promise for optimizing energy efficiency. Accurate forecasting of appliance energy consumption empowers homeowners to actively manage their energy usage, resulting in cost reduction and a positive impact on sustainability efforts. Continued research and development in this domain will facilitate the creation of intelligent energy management systems that adapt to occupants' needs and environmental conditions, leading to further improvements in energy efficiency for residential buildings.

6.3. Reference:

B. Yildiz et al. Recent advances in the analysis of residential electricity consumption and applications of smart meter data Appl. Energy(2017)

H.Z. Wang et al. Deep learning based ensemble approach for probabilistic wind power forecasting Appl. Energy (2017)

Y. Sun et al. A review of the-state-of-the-art in data-driven approaches for building energy prediction Energy Build. (2020)

M. Gilanifar et al. Multitask Bayesian spatiotemporal Gaussian processes for short-term load forecasting IEEE Trans. Ind. Electron. (June 2020)

U.S. Energy Information Administration, “Frequently Asked Questions”, Accessed: July. 2021, [online]. Available:

F.L. Quilumba et al. Using smart meter data to improve the accuracy of intraday load forecasting considering customer behavior similarities