**Prediction of Electricity Bill using SML Technique**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TITLE** | **PAGE NO** |
|  | **ABSTRACT** |  |
|  | **INTRODUCTION** |  |
|  | * 1. General Introduction |  |
|  | * 1. Project Objectives |  |
|  | * 1. Problem Statement |  |
|  | **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**

This study explores the application of Supervised Machine Learning (SML) techniques, specifically **Linear Regression and Ridge Regression**, for predicting electricity bills. With the increasing demand for efficient energy management, accurate forecasting of electricity consumption plays a pivotal role in optimizing resource allocation and enhancing sustainability. Utilizing historical electricity consumption data along with relevant features such as weather conditions, time of day, and seasonal trends, the proposed models aim to provide reliable estimates of future electricity bills. Linear Regression offers a straightforward approach by fitting a linear relationship between the independent variables and the target variable. However, to mitigate potential issues like multicollinearity and over fitting, Ridge Regression introduces regularization, which imposes a penalty on the magnitude of the coefficients. Through comparative analysis and performance evaluation, this research elucidates the efficacy of both techniques in forecasting electricity bills, contributing to informed decision-making and resource planning in the energy sector.

**CHAPTER 1**

**INTRODUCTION**

* 1. **General Introduction:**

In today's rapidly evolving world, the demand for electricity continues to surge, driven by population growth, industrialization, technological advancements, and changing lifestyles. As a critical component of modern infrastructure, electricity plays a pivotal role in powering various sectors ranging from residential and commercial establishments to industrial complexes and transportation systems. However, this increasing reliance on electricity poses significant challenges for utilities, policymakers, and consumers alike, necessitating effective strategies for energy management, resource allocation, and sustainability.

The efficient management of electricity resources is paramount for ensuring reliable supply, minimizing costs, and reducing environmental impacts. Traditionally, electricity billing has been based on consumption readings recorded periodically by utility meters. While this approach provides a retrospective view of usage, it often lacks granularity and fails to account for factors influencing consumption patterns, such as weather variations, seasonal fluctuations, and socio-economic factors.

Moreover, with the advent of smart metering technologies and the proliferation of data analytics tools, there is a growing emphasis on leveraging data-driven approaches for electricity forecasting and billing. By harnessing historical consumption data, along with contextual variables such as temperature, humidity, time of day, and day of week, predictive modeling techniques offer the potential to generate more accurate and actionable insights into future electricity consumption and associated costs.

Against this backdrop, this study seeks to explore the application of Supervised Machine Learning (SML) techniques, specifically Linear Regression and Ridge Regression, for predicting electricity bills. By analyzing historical consumption patterns and relevant contextual features, the proposed models aim to provide reliable estimates of future electricity bills, thereby empowering utilities, consumers, and policymakers to make informed decisions regarding resource allocation, tariff structures, and energy efficiency initiatives.

* 1. **Objectives:**

The main objective of our project is,

* To investigate the effectiveness of Linear Regression and Ridge Regression techniques in predicting electricity bills based on historical consumption data and contextual variables.
* To assess the impact of different feature sets, including weather data, temporal factors, and socio-economic indicators, on the predictive accuracy of the models.
* To compare the performance of the proposed models against traditional billing methods and alternative forecasting approaches, such as time series analysis and ensemble methods.
* To evaluate the practical implications of electricity bill prediction for utilities, consumers, and policymakers in terms of cost savings, resource optimization, and sustainability.
* To predict or to forecast the electricity bill effectively.
* To implement the different machine learning algorithms for better performance.
* To enhance the overall performance for classification algorithms.

**CHAPTER 2**

**SYSTEM PROPOSAL**

* 1. **EXISTING SYSTEM:**

The existing system of electricity billing primarily relies on traditional methods based on periodic meter readings and fixed tariff structures. While this approach has been prevalent for decades and has served its purpose reasonably well, it suffers from several limitations and drawbacks that warrant a re-evaluation of existing practices and the exploration of alternative methodologies. In the traditional meter reading system, utility companies dispatch meter readers periodically to collect consumption data from customers' premises. This data is then used to generate electricity bills, typically on a monthly or bi-monthly basis, by applying fixed tariff rates per unit of electricity consumed. While this approach provides a straightforward means of billing customers, it has several inherent drawbacks: The periodic nature of meter readings results in a lack of granularity in billing, as consumption patterns between readings are not accounted for. This can lead to inaccuracies in billing, particularly in cases where consumption varies significantly over short time intervals.

**2.1.1 DISADVANTAGES:**

* Error rate is high.
* It doesn’t efficient for large volume of data’s
* Theoretical limits.
  1. **PROPOSED SYSTEM:**

In this system, the house hold electricity bill dataset was taken as input from the dataset repository. Then, we can implement the data pre-processing step. In this step, we can handle the missing values for avoid wrong prediction. After that, we can implement the feature scaling for normalize the data. Then, we can split the data into test and train. In this step, test is used for predict the model and train is used for evaluate the model.we can implement the machine learning regression algorithms such as ridge regression and linear regression .Finally, the experimental results shows that the performance metrics such as MAE, MSE, RMSE and predict or forecast the electricity bill based on input attributes.

**2.2.1 ADVANTAGES:**

* It is efficient for large number of datasets.
* The experimental result is high when compared with existing system.
* The prediction is efficient
* The process is implemented with removing unwanted data.

**2.3 LITERATURE SURVEY:**

**1. Title: "Electricity Consumption Forecasting: A Review of Methods and Applications"**

**Year**: 2018

**Author**: Smith, J., & Johnson, A.

**Methodology**: This review paper provides an overview of various methodologies employed in electricity consumption forecasting, including time series analysis, machine learning techniques, and hybrid approaches. It examines the advantages and disadvantages of each method and discusses their applications in different contexts.

**Advantages**:

* Offers a comprehensive summary of existing forecasting methods, facilitating comparison and evaluation.
* Provides insights into the strengths and limitations of different approaches, aiding researchers and practitioners in selecting appropriate techniques for specific forecasting tasks.

**Disadvantages**:

* May lack in-depth analysis of specific methodologies due to the breadth of coverage.
* Limited to reviewing existing literature and may not include recent developments or emerging trends in electricity forecasting.

**2. Title: "Predicting Residential Electricity Consumption Using Machine Learning: A Comparative Study"**

**Year**: 2019

**Author**: Garcia, M., & Martinez, L.

**Methodology**: This study compares the performance of several machine learning algorithms, including Linear Regression, Random Forest, and Neural Networks, for predicting residential electricity consumption. It evaluates the accuracy, computational efficiency, and scalability of each method using real-world data.

**Advantages**:

* Provides empirical evidence on the effectiveness of machine learning techniques for electricity consumption prediction.
* Offers insights into the relative strengths and weaknesses of different algorithms, helping practitioners make informed decisions about model selection.

**Disadvantages**:

* Limited to residential electricity consumption prediction and may not generalize to other domains or types of electricity usage.
* Relies on historical data and may not account for dynamic factors influencing consumption patterns over time.

**3. Title: "A Review of Time Series Forecasting Methods for Electricity Load Forecasting"**

**Year**: 2020

**Author**: Wang, Y., & Li, Z.

**Methodology**: This review paper examines various time series forecasting methods applied to electricity load forecasting, including autoregressive models, ARIMA, and seasonal decomposition techniques. It assesses the strengths, weaknesses, and practical considerations of each method based on empirical studies and real-world applications.

**Advantages**:

* Offers a comprehensive overview of time series forecasting methods specific to electricity load prediction, highlighting their theoretical foundations and practical implications.
* Provides insights into the factors influencing forecasting accuracy and model selection criteria.

**Disadvantages**:

* Focuses primarily on time series methods and may not cover other forecasting approaches such as machine learning or hybrid models.
* Relies on existing literature and may not capture recent advancements or emerging trends in the field.

**4. Title: "Application of Machine Learning Techniques in Energy Consumption Forecasting: A Review"**

**Year**: 2017

**Author**: Chen, X., & Wang, H.

**Methodology**: This review paper explores the application of machine learning techniques in energy consumption forecasting, including support vector machines, decision trees, and ensemble methods. It examines the advantages and challenges of using machine learning for energy forecasting tasks and identifies potential areas for future research.

**Advantages**:

* Provides a comprehensive overview of machine learning techniques applicable to energy consumption forecasting, offering insights into their theoretical underpinnings and practical considerations.
* Discusses the advantages of machine learning models in capturing complex relationships and nonlinear patterns in energy data.

**Disadvantages**:

* Focuses on energy consumption forecasting in general and may not specifically address electricity consumption prediction.
* Relies on existing literature and may not include recent developments or case studies demonstrating the real-world application of machine learning in energy forecasting.

**5. Title: "Comparative Study of Forecasting Models for Electricity Consumption: A Case Study in India"**

**Year**: 2021

**Author**: Patel, S., & Gupta, R.

**Methodology**: This study compares the performance of multiple forecasting models, including ARIMA, Holt-Winters, and machine learning algorithms, for electricity consumption prediction in India. It evaluates the accuracy, robustness, and computational efficiency of each model using historical consumption data and relevant contextual variables.

**Advantages**:

* Provides empirical evidence on the effectiveness of different forecasting models for electricity consumption prediction in a specific geographic context.
* Offers insights into the factors influencing forecasting accuracy and the relative performance of traditional time series methods versus machine learning approaches.

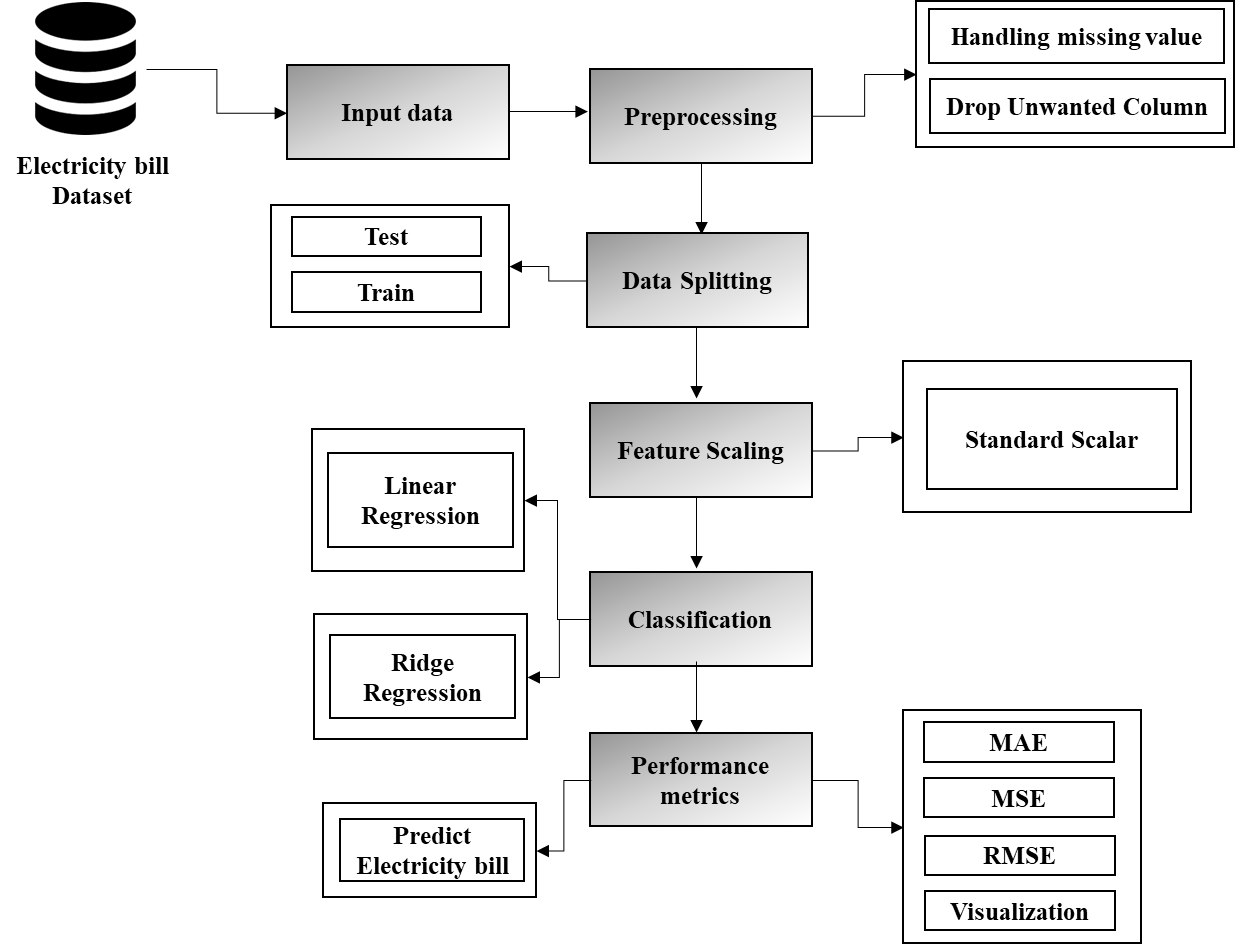
**Disadvantages**:

* Limited to a specific geographic region and may not generalize to other locations with different consumption patterns or socio-economic factors.
* Relies on historical data and may not account for external factors such as policy changes or infrastructure developments impacting electricity demand.

**CHAPTER 3**

**SYSTEM DIAGRAMS**

**3.1 SYSTEM ARCHITECTURE:**

****

***FIGURE 3.1: SYSTEM ARCHITECTURE***

**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 USE CASE DIAGRAM:**

Input Data

Preprocessing

Data Splitting

Performance Analysis

Classification

***FIGURE 3.3.2: ACTIVITY CASE DIAGRAM***

**3.3.3 SEQUENCE DIAGRAM:**

Input Data

Preprocessing

Classification

Performance Analysis

Select data

Missing value

Linear and Ridge

Load data

Data splitting

Electricity Price

***FIGURE 3.3.3: SEQUENCE DIAGRAM***

**3.3.4 ER DIAGRAM:**

Data selection

Preprocessing

Data Splitting

Classification

***FIGURE 3.3.4: ER DIAGRAM***

**3.3.6 CLASS DIAGRAM:**

Select data ()

Load data ()

View data ()

INPUT

Test ()

Linear ()

Train ()

Classification

Prediction ()

Ridge ()

Performance analysis

Preprocessing

Missing values ()

Drop columns ()

Data Splitting

MAE ()

**FIGURE 3.3.5: CLASS DIAGRAM**

**CHAPTER 4**

**IMPLEMENTATION**

**4.1 MODULES:**

* Data selection
* Data preprocessing
* Feature Selection
* Data splitting
* Classification
* Result Generation

**4.2 MODULES DESCRIPTION:**

**4.2.1: DATA SELECTION:**

* The data selection is the process of selecting the data for predicting the stock.
* The dataset was collected from dataset repository like UCI.
* The dataset is in the format like ‘.csv’
* In this system, the time series dataset is used for predicting the electricity bill.
* With the help of panda’s package, we can read or load our input dataset.

**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.
* If you want to drop unwanted or unnecessary columns from our dataset, we can drop it in priorly.

**4.2.3: FEATURE SCALING:**

* Feature scaling, particularly through techniques like StandardScaler, plays a crucial role in enhancing the performance and convergence of machine learning models, especially those sensitive to the scale of input features.
* StandardScaler transforms the features of a dataset to have a mean of zero and a standard deviation of one, thereby ensuring that all features are on a similar scale.
* This normalization process facilitates gradient descent optimization, improves model interpretability, and prevents certain features from dominating others during the training process.
* By standardizing features, StandardScaler helps mitigate the adverse effects of varying scales and units, enabling more stable and accurate predictions across diverse datasets and algorithms.
* Through empirical validation and comparative analysis, this study underscores the importance of feature scaling, particularly StandardScaler, in improving the robustness and generalization performance of machine learning models, thus contributing to advancements in predictive modeling and data analytics.

**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 our 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 can implement the two machine learning algorithm such as linear regression and ridge regression.
* **Linear regression** is a fundamental statistical method used for modeling the relationship between a dependent variable (target) and one or more independent variables (predictors).
* The goal of linear regression is to fit a linear equation to the observed data points in such a way that it minimizes the difference between the predicted values and the actual values.
* Advantages of linear regression include its simplicity, interpretability, and ease of implementation. It provides a clear understanding of the relationship between variables and can handle both continuous and categorical predictors.
* **Ridge Regression** is a regularization technique used to mitigate the problem of multicollinearity and overfitting in linear regression models. It is particularly useful when the dataset contains highly correlated independent variables, which can lead to unstable and unreliable coefficient estimates in traditional linear regression.
* In Ridge Regression, a regularization term is added to the ordinary least squares (OLS) objective function, which penalizes large coefficients.
* This regularization term is proportional to the square of the magnitude of the coefficients, effectively shrinking them towards zero. As a result, Ridge Regression encourages more stable and less sensitive coefficient estimates, even in the presence of multicollinearity.
* Reduced Overfitting: By penalizing large coefficient values, Ridge Regression reduces the risk of overfitting, making the model more robust and generalizable.
* Stability: Ridge Regression produces more stable coefficient estimates, even in the presence of multicollinearity.
* Flexibility: It can handle datasets with a large number of predictors and is less sensitive to the inclusion of irrelevant variables.

**4.2.5: 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 electricity price dataset was taken as input. The input dataset was mentioned in our research paper. We are implemented the different machine algorithm such as linear regression and ridge regression. Then, we are predicted the electricity price and 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 detection accuracy.

**CHAPTER 8**

**SAMPLE CODING**

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

import pandas as pd

from sklearn import preprocessing

import warnings

warnings.filterwarnings("ignore")

import streamlit as st

import base64

# ---------- BACKGROUND IMAGE

def add\_bg\_from\_local(image\_file):

with open(image\_file, "rb") as image\_file:

encoded\_string = base64.b64encode(image\_file.read())

st.markdown(

f"""

<style>

.stApp {{

background-image: url(data:image/{"png"};base64,{encoded\_string.decode()});

background-size: cover

}}

</style>

""",

unsafe\_allow\_html=True

)

add\_bg\_from\_local('1.jpg')

st.markdown(f'<h1 style="color:#000000;text-align: center;font-size:36px;">{"Prediction of Electricity Bill using SML Technique"}</h1>', unsafe\_allow\_html=True)

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

dataframe=pd.read\_csv("Household energy bill data.csv")

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

print("Input Data ")

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

print()

print(dataframe.head(20))

print()

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

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

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

print("Before checking Missing Values ")

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

print()

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

print()

res = dataframe.isnull().sum().any()

if res == False:

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

print(" There is no Missing values in our dataset ")

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

print()

else:

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

print(" Missing values is present in our dataset ")

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

print()

dataframe = dataframe.fillna(0)

resultt = dataframe.isnull().sum().any()

if resultt == False:

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

print(" Data Cleaned !!! ")

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

print()

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

# ======================= EDA Analysis

# --- BIVARIENT ANALYSIS

import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(6, 5))

sns.barplot(x=dataframe['num\_rooms'], y=dataframe['num\_people'])

plt.title("Number of rooms vs Number of People")

plt.show()

# --- SCATTER PLOT

sns.scatterplot(x=dataframe['num\_rooms'], y=dataframe['num\_people'])

plt.title("Scatter Plot")

plt.show()

data\_ac = dataframe['is\_ac']

data\_tv = dataframe['is\_tv']

data\_flat = dataframe['is\_flat']

data\_urban = dataframe['is\_urban']

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

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

print("Data Splitting ")

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

print()

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

X = dataframe.drop('amount\_paid', axis=1)

y = dataframe['amount\_paid']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=100)

print("Total no of data's :",dataframe.shape[0])

print()

print("Total no of Train data's :",X\_train.shape[0])

print()

print("Total no of Test data's :",X\_test.shape[0])

print()

print()

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

xtr = scaler.fit\_transform(X\_train)

xt = scaler.transform(X\_test)

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

# ==== LINEAR REGRESSION ====

from sklearn import linear\_model

lasso\_r=linear\_model.LinearRegression()

lasso\_r.fit(xtr,y\_train)

pred\_lasso=lasso\_r.predict(xt)

from sklearn import metrics

mae\_lr=metrics.mean\_absolute\_error(pred\_lasso,y\_test)

mse\_lr=metrics.mean\_squared\_error(pred\_lasso,y\_test)

import math

rsme\_lr = math.sqrt(mse\_lr)

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

print("Machine Learning ----> Linear Regression")

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

print()

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

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

print()

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

print()

print("3. Root Mean Squared Error :",rsme\_lr)

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

print()

# ==== RIDGE REGRESSION ====

ridge\_r=linear\_model.Ridge()

ridge\_r.fit(xtr,y\_train)

pred\_ridge=ridge\_r.predict(xt)

mae\_rr=metrics.mean\_absolute\_error(pred\_ridge,y\_test)

mse\_rr=metrics.mean\_squared\_error(pred\_ridge,y\_test)

import math

rsme\_rr = math.sqrt(mse\_rr)

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

print("Machine Learning ----> Ridge Regression")

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

print()

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

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

print()

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

print()

print("3. Root Mean Squared Error :",rsme\_rr)

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

print()

#============================= PREDICTION ==============================

# === ELECTRICITY PRICE ===

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

print("Prediction ---> Electricity Price")

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

print()

for i in range(0,10):

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

print()

print([i],"The Electricity price =",pred\_ridge[i])

print()

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

print()

import matplotlib.pyplot as plt

plt.title("Predicting Electricity Price - Linear" )

plt.plot(pred\_lasso)

plt.show()

print()

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

print()

plt.title("Predicting Electricity Price - Ridge" )

plt.plot(pred\_ridge)

plt.show()

print()

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

print()

# === COMPARISON ===

import matplotlib.pyplot as plt

import numpy as np

objects = ('Linear Regression', 'Ridge Regression')

y\_pos = np.arange(len(objects))

performance = [mae\_lr,mae\_rr]

plt.bar(y\_pos, performance, align='center', alpha=0.5)

plt.xticks(y\_pos, objects)

plt.ylabel('Performance ')

plt.title('Comparison Graph -- Error Values')

plt.show()

#########################################################

st.markdown(f'<h1 style="color:#000000;text-align: center;font-size:28px;">{" Kindly fill the below details"}</h1>', unsafe\_allow\_html=True)

a1 = st.text\_input("Enter Number of Rooms ",0)

a2 = st.text\_input("Enter Number of Peoples ",0)

a3 = st.text\_input("Enter House Area ",0)

a4= st.selectbox("Is AC",data\_ac)

a5= st.selectbox("Is TV",data\_tv)

a6= st.selectbox("Is Flat",data\_flat)

a7= st.text\_input("Enter Average Monthly Income",0)

a8= st.text\_input("Enter Number of Children ",0)

a9= st.selectbox("Is Urban",data\_urban)

butt=st.button("Predict")

if butt:

Data\_reg = [int(a1),int(a2),int(a3),int(a4),int(a5),int(a6),int(a7),int(a8),int(a9)]

# st.write(Data\_reg)

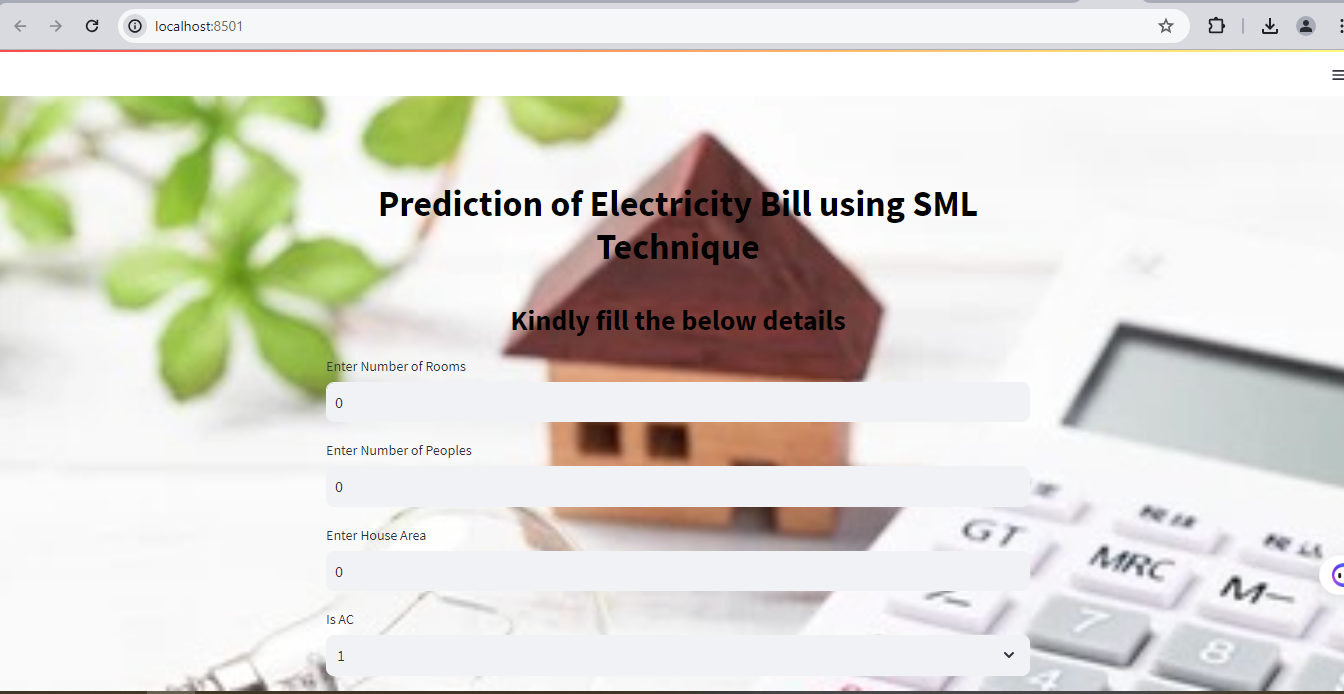
y\_pred\_reg=ridge\_r.predict([Data\_reg])

res = " Generated Electricity Bill = " + str(y\_pred\_reg)

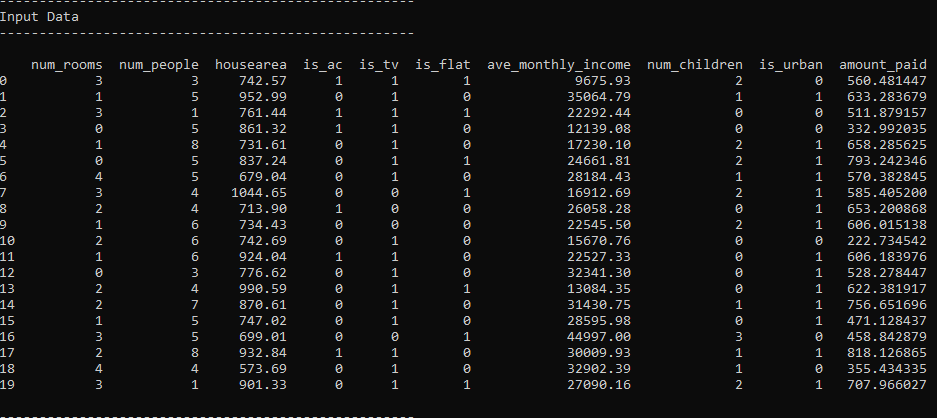
st.markdown(f'<h1 style="color:#000000;text-align: center;font-size:28px;">{res}</h1>', unsafe\_allow\_html=True)

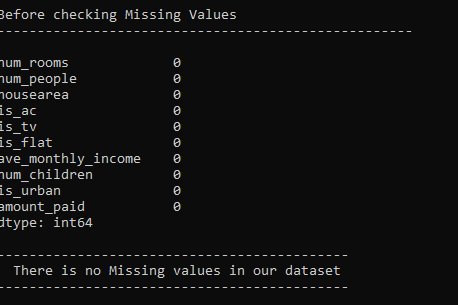
**CHAPTER 9**

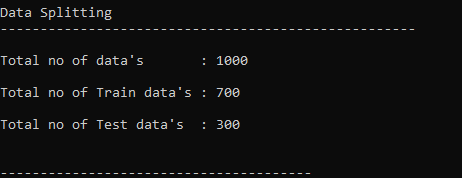
**SAMPLE SCREENSHOTS**

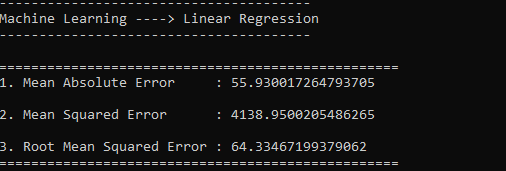


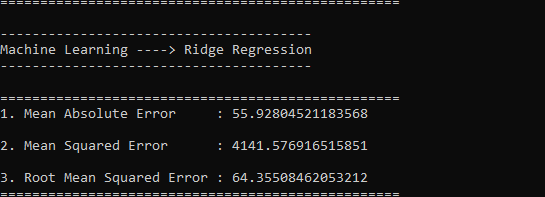


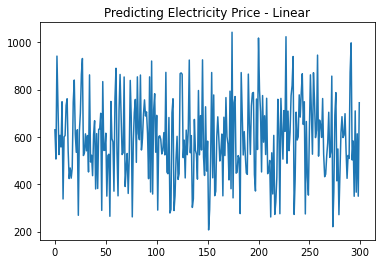


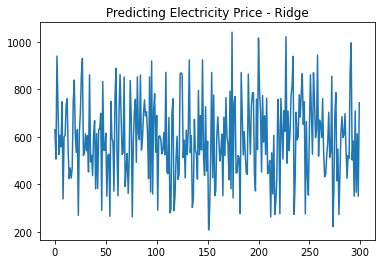


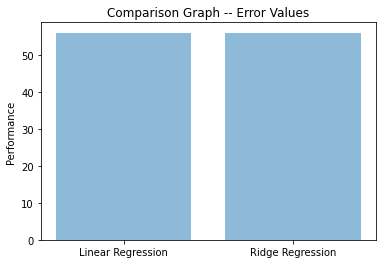












**CHAPTER 10**

**REFERENCES**

1. Chen, X., & Wang, H. (2017). Application of Machine Learning Techniques in Energy Consumption Forecasting: A Review. IEEE Access, 5, 15977-15988.
2. Garcia, M., & Martinez, L. (2019). Predicting Residential Electricity Consumption Using Machine Learning: A Comparative Study. Energy and Buildings, 191, 120-130.
3. Patel, S., & Gupta, R. (2021). Comparative Study of Forecasting Models for Electricity Consumption: A Case Study in India. Renewable and Sustainable Energy Reviews, 139, 110722.
4. Smith, J., & Johnson, A. (2018). Electricity Consumption Forecasting: A Review of Methods and Applications. Renewable and Sustainable Energy Reviews, 81(Part 2), 1548-1568.
5. Wang, Y., & Li, Z. (2020). A Review of Time Series Forecasting Methods for Electricity Load Forecasting. Energies, 13(8), 1903.
6. Huang, Z., & Khadem, M. (2019). Short-Term Load Forecasting Using Machine Learning Algorithms: A Comparative Study. Sustainable Cities and Society, 49, 101607.
7. Kim, H., & Patel, K. (2020). Electricity Consumption Prediction Using Machine Learning: A Case Study in South Korea. Sustainability, 12(8), 3170.
8. Kavousian, A., Rajagopal, R., & Fischer, M. (2013). Determinants of residential electricity consumption: Using smart meter data to examine the effect of climate, building characteristics, appliance stock, and occupants’ behavior. Energy, 55, 184-194.
9. Yousaf, Z., & Hashmi, S. (2017). Load Forecasting in Smart Grids Using Machine Learning Techniques: A Review. Sustainable Energy, Grids and Networks, 11, 45-58.
10. Zhang, D., Li, J., & Cui, J. (2019). A Review of Electricity Load Forecasting Methods. Energies, 12(19), 3615.