FACULTY OF ENGINEERING AND TECHNOLOGY

MASTER IN COMPUTER ENGINEERING

PREDICTING NEXT CHARGE DATE FOR PREPAID ELECTRICITY METERS USING MACHINE LEARNING

Mohammad Abu El Halaweh

Supervised by:

Dr. Ismail Khater

Dr. Ahmad Alsadeh

Date: FEB-2024

MASTER THESIS

Predicting Next Charge Date for Prepaid Electricity Meters Using Machine Learning

Birzeit University

By: Mohammad Abu El Halaweh

This thesis was successfully defended on Thursday, February 15th, 2024.

Supervisors:

Supervisor 1	**Dr. Ismail M Khater**
Supervisor 2	**Dr. Ahmad Alsadeh**

Examiners:

Examiner 1	**Dr. Yazan Abu Farha**
Examiner 2	**Dr. Abualseoud Hanani**

This Master Thesis is prepared by Mohammad Abu El Halaweh as in part fulfillment of the degree requirements for the Master in Computer Engineering.

جامعة بيرزيت
BIRZEIT UNIVERSITY

Declaration of Authorship

I, Mohammad Abu El Halaweh, declare that this thesis titled, Predicting Next Charge Date for Prepaid Electricity Meters Using Machine Learning and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research de-gree at Birzeit University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Birzeit University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed my-self.

Signed: _____

Date: _____

Abstract

Electricity plays a crucial role in the development of the world economy. It is a commodity that needs to be managed efficiently and effectively to ensure its availability to consumers. Unfortunately, electricity production and distribution are complex and expensive processes. Thus, legalizing and rationalizing its use is essential. Consequently, companies in many countries have begun installing prepaid electric meters in customers' homes, offices, and stores, requiring them to purchase electricity on a prepaid basis. Prepaid electricity meters have become increasingly popular as a means of providing consumers with more control over their electricity consumption and expenditure. However, one of the main challenges with prepaid electricity meters is the potential for the meter balance to run out, leading to a cutoff in electricity supply. This can result in customer dissatisfaction due to the sudden interruption of service and its consequent effects, and sometimes it may cause serious disasters. Thus, to mitigate this issue, customers should be alerted when their meter has a low prepaid balance, advising them to recharge it soon. This thesis proposes machine learning-based approaches to genuinely address this issue. We collaborated with the Jerusalem District Electricity Company (JDECO) to collect essential customer data from their prepaid systems. We used the datasets to train various machine learning models to create the Next Charge Date Predictor (NCDP) for customers. Our approach involved using a random forest regressor, XGBoost regressor, linear regressor, polynomial regressor, K-Nearest Neighbors (KNN), decision tree, and recurrent neural network (RNN) deep learning model. Remarkably, our RNN model outperformed the others, achieving a mean absolute error (MAE) of 2.07, effectively predicting customers' next charge dates.

المستخلص

تلعب الكهرباء دورًا حاسمًا في تطوير اقتصاد العالم، حيث إنها سلعة تحتاج إلى إدارة فعالة وكفؤة لضمان توفرها للمستهلكين. للأسف، إنتاج وتوزيع الكهرباء معقد ومكلف. لذلك، من الحكمة تقنين وعقلنة استخدامها. لذلك، بدأت الشركات في العديد من البلدان بتركيب عدادات الكهرباء مسبقة الدفع في منازل العملاء ومكاتبهم ومتاجرهم، حيث يتعين عليهم شراء الكهرباء على أساس مسبق الدفع. أصبحت عدادات الكهرباء مسبقة الدفع شائعة بشكل متزايد كوسيلة لتزويد المستهلكين بمزيد من السيطرة على استهلاكهم للكهرباء. ومع ذلك، أحد التحديات الرئيسية مع عدادات كهرباء مسبقة الدفع هو أن رصيد العداد قد ينفد، مما سيؤدي إلى قطع الكهرباء. إن هذا سيؤدي إلى عدم رضا العملاء عن الانقطاع المفاجئ للخدمة والآثار الناجمة عن ذلك، وأحيانًا قد يسبب كوارث خطيرة. وبالتالي، للتخفيف من هذه المشكلة، يجب تنبيه العميل بأن عداده يحتوي على رصيد منخفض، ويجب عليه شحنه قريبًا. تقترح هذه الأطروحة طرقًا تستند إلى التعلم الآلي للتقليل من حجم هذه المشكلة وتأثيرها السلبي بشكل جاد. قمنا بالتعاون مع شركة كهرباء محافظة القدس (JDECo) لجمع بيانات العملاء الأساسية من عملائهم. استخدمنا مجموعات البيانات لتدريب نماذج التعلم الآلي المختلفة للتنبؤ بتواريخ الشحن الواجبة للعملاء. تضمنت طريقتنا استخدام موديل random forest ، وموديل تعزيز التدرج XGBoost ، وموديل linear regressor ، وموديل polynomial regressor ، وموديل (KNN) ، وموديل decision tree ، وموديل recurrent neural network (RNN) . بشكل لافت، أظهر موديل RNN أفضل أداء، حيث حقق خطأ مطلق متوسط (MAE) قدره 2.07 الذي تنبأ بفعالية لتواريخ الشحن الواجبة للمستهلكين

Contents

List of Figures

List of Tables

List of Abbreviations

AI Artificial Intelligence

ANN Artificial Neural Network

ANOVA Analysis of Variance

CSV Comma-separated Values

DT Decision Tree

GRU Gated Recurrent Units

KNN K-Nearest Neighbors

LSTM Long Short Term Memory

MAE Mean Absolute Error

ML Machine Learning

MSE Mean Squared Error

NCDP Next Charge Date Predictor

PCA Principal Component Analysis

ReLU Rectified Linear Unit

RMSE Root Mean Squared Error

RNN Recurrent Neural Network

STS Standard Transfer Specification

SVM Support Vector Machine

XGB eXtreme Gradient Boosting

Chapter 1

Introduction

Contents

In the modern world, electricity has become an indispensable part of our daily lives. From lighting up our homes and workplaces to powering the electronic devices that we use for communication, entertainment, and work, electricity has transformed the way we live, work, and play. The availability and reliability of electricity have significantly enhanced the quality of life, enabling people to enjoy a range of comforts and conveniences that were once unimaginable. It has become so deeply ingrained in our way of life that it is hard to imagine a world without it. It has also spurred innovation and technological advancements in various sectors such as healthcare, education, transportation, and agriculture, to name a few, yet around 1.2 billion individuals globally lack access to it [3].

Electricity companies face numerous challenges in their operations, including the issue of non-compliance by their customers in paying bills or paying them on time. This problem not only creates financial burdens for these companies but also affects their ability to generate and distribute electricity efficiently. According to a report by the National Energy Assistance, the non-payment of electricity bills by consumers is a significant problem in developing countries, and during the pandemic of coronavirus, the amount of unpaid bills has doubled to over $27 billion [4] and over $40 billion for different utility services [5]. To tackle this issue, many electricity companies have turned to the installation of prepaid meters as a solution which has grown increasingly popular in recent years [6]. Prepaid meters manage consumers to pay for electricity before they consume it, thereby avoiding the accumulation of unpaid bills. The benefits of installing prepaid meters push many countries to install prepaid meters as a solution for unpaid bills, for example, Britain extending their use to utility services such as electricity, gas, and water, with over 31 million meters installed by the end of 2022 [7].

The increasing demand for prepaid meters among customers also has some problems that users face, and the most important of these problems is the constant need to charge the electricity meter to ensure the continuity of the electricity, this issue has made predicting the next charge date a vital issue. The success of prepaid systems is dependent on several factors, where customer satisfaction and acceptance are crucial factors. Machine learning has emerged as a promising technology for predicting consumer behavior, and interest in this field has been rapidly growing. Researchers in various fields, including business administration, have been applying machine learning to prediction problems. In recent years, deep learning algorithms, including recurrent neural networks (RNNs), have shown remarkable performance in various prediction tasks [8]. The domain of customer behavior is an ideal area for applying machine learning for prediction purposes [9].

Existing works of literature have made extended studies to understand consumer behavior and his online purchasing patterns to build a model that can predict their actions based on their behaviors and actions [10] [11]. In contrast to online purchasing, the prepaid environment provides new opportunities to predict customer behavior through the use of machine learning, where the customer charging patterns can be tracked and various features can be extracted to predict customer charges behavior. Customer charges records are sequential records that contain historical data of each charge which could be analyzed to perform prediction tasks in machine learning. Various frameworks have been proposed to

predict consumer behavior using machine learning, such as Bayesian models [12], logistic regression models [13], XGBClassifier [14], and game theory-based approach [15].

1.1 Motivation

The demand for prepaid electricity meters is high as a solution to the lack of commitment of customers to pay their bills on time, or some of them fail to pay them. Because of this, the increasing demand highlights the urgent need for logistical features and facilities for prepaid meter customers to enhance their service experience.

One of the major challenges of prepaid meters is that they are programmed to relay off the power when its balance approaches zero, this is typically what all prepaid systems do. In telecom, internet service providers (ISP), and other service companies, the credit for a customer is registered and controlled in the companies' servers, therefore the company can read and alert the user for low credit before it expires. However, for electricity meters, the balance is registered and controlled inside the meter itself. Consequently, electricity distribution company could not read and alert the customer for low credit unless the customer manually checks his/her meter balance. Hence, when meter balance expires, the electricity will cut off, which could happen at any time, in the morning, evening, at work time, or while traveling. This results in the dissatisfaction of prepaid meter customers with the company's service and often leads to complaints against the company.

The current approach for most prepaid meters is to make a beep sound or a red light [16] when its balance reaches about 30 to 10 kilowatts. Unfortunately, this method has led to dissatisfaction among customers due to the difficulty in periodically checking the meter [17]. Furthermore, in some cases, there may be a physical distance between the meter and the customer, resulting in difficulty in hearing the beep sound of the meter. Consequently, consumers may come close to running out of electricity, leading to inconvenience and disruption in their daily lives.

Unlike online purchasing, customer behavior in a prepaid environment has not been extensively explored, especially in machine learning techniques, therefore, there is a need to explore the effectiveness of using machine learning to build a model that can predict prepaid charges period for customer in prepaid electricity. Therefore, the aim of this study is to explore the potential of using machine-learning algorithms to predict the next charge date since machine learning can learn patterns and relationships from historical data and use

them to make predictions.

Predicting customer behavior in prepaid environments provides new opportunities for machine learning, as charging patterns can be tracked and different features can be extracted and used to predict customer charging behavior. However, the results of this study will have important practical implications for electricity and prepaid service companies, as these results can help companies improve their services provided to prepaid customers and increase their satisfaction and retention.

1.2 Research Questions

After we talked comprehensively about the introduction of the research and the motivation for doing this research, we will now touch on the concrete objectives of this study. The essence of this thesis lies in answering the following research questions, as these questions serve as a guiding beacon for us to reach the ultimate goal of developing an efficient and accurate neural network model to predict the next charging date of electricity meters:

- How machine learning can be leveraged to build a predictive model for prepaid meters?
- How the historical usage patterns analytics can be used to identify trends in usage and determine when customers are likely to need their meters recharged?
- Which of the used machine learning algorithms is the best for building a predictive model for prepaid systems?
- How does the neural network model's performance compare to traditional machine learning models for the sequential data of customers' charging behavior?

1.3 Thesis Contribution

In this thesis, we conducted a comprehensive exploration of employing machine learning and deep learning tools to address the challenges faced by electricity companies for installing prepaid electricity meters. We employed a variety of machine learning algorithms, with a special emphasis on deep learning, aiming to identify the most effective strategies for understanding and predicting customer behavior.

We carried out analyses of customer behaviors and the patterns of their meter recharging activities and historical usage, extending our analyses to investigate connections related to the geographical locations of customers and the behavior of residential areas. We found links and behaviors that lead to improved prediction levels.

This study advances the application of machine learning and deep learning in forecasting the behavior of prepaid electricity meter customers, pinpointing a research gap in existing literature. Unlike most studies focusing on the use of IoT and sensor data for energy predictions, our research capitalizes on customer behavior and meter recharge history. This approach not only addresses a critical void in current studies but also introduces a fresh perspective on optimizing energy resource management through sophisticated analytical techniques. Consequently, our work provides valuable insights for improving the efficiency and dependability of prepaid electricity services.

Overall, this research will add to the growing scientific research that concerns predicting customer behavior using various machine learning techniques. It will also offer practical implications for electricity companies and researchers interested in customer services. Future research in this field may benefit from its findings, which can aid in the development of machine learning algorithms for the prepaid environment.

1.4 Formal Definition of the Problem

This thesis examines the major challenge in the energy sector, which specifically revolves around the management of prepaid electricity meters. The central problem is that the subscriber or the company does not know the balance of the meter due to the nature of the issue, which causes difficulty in anticipating the next charging date for these meters, which is a complicated process due to various influencing factors.

1.4.1 Problem Statement

The primary research question this thesis aims to address is: Given a set of historical charging data for prepaid electricity customers, can we accurately predict the next charge date?

We can define the problem in a formal way as follows:

$$\gamma = f(x_1, x_2, x_3, \ldots, x_n) \tag{1.1}$$

Each Xi represents a specific attribute like payment amount (paymentAMT), previous

charging date (previousChargingDate), kW Amount (KWAmount), kW total price (KW-TotalPrice), debt taken (debtTaken), VAT taken (VATTaken), total fixed for last days (totalFixedForLastDays), total miscellaneous charges (totalMisc), days since last charge (daysSinceLastCharge), payment date (paymentDate), customer number (customerNO). The task is to predict a continuous target variable Y, representing the difference in days to the next charge.

1.4.2 Significance of the Problem

The accurate prediction of the next charge date for prepaid electricity meters is of considerable significance. For electricity companies, these predictions can enhance resource management, improve demand forecasting, and foster proactive customer engagement strategies. For the customers, these predictions can aid in prevent sudden service cut off due to prepaid meter credit, and contribute towards more efficient budget management.

1.4.3 Problem Constraints

The main challenges of this problem involve the precision of past charging data, the inconsistency in electricity usage behaviors, and the resilience and reliability of the prediction model. Additional possible impediments could stem from volatile tariff rates, shifts in customer behavior, or external elements like weather conditions, all of which may not always be accurately foreseen

1.4.4 Scope of the Study

This study is primarily focused on developing a predictive model for prepaid electricity meter recharge dates using historical customer data. The research will be confined to the set of features provided, aiming to establish a reliable predictive model to anticipate the number of days until the next meter recharge.

Subsequent chapters will describe the methodology used, which includes the process of data collection and analysis, selection of predictive modeling techniques, and evaluation metrics for model performance

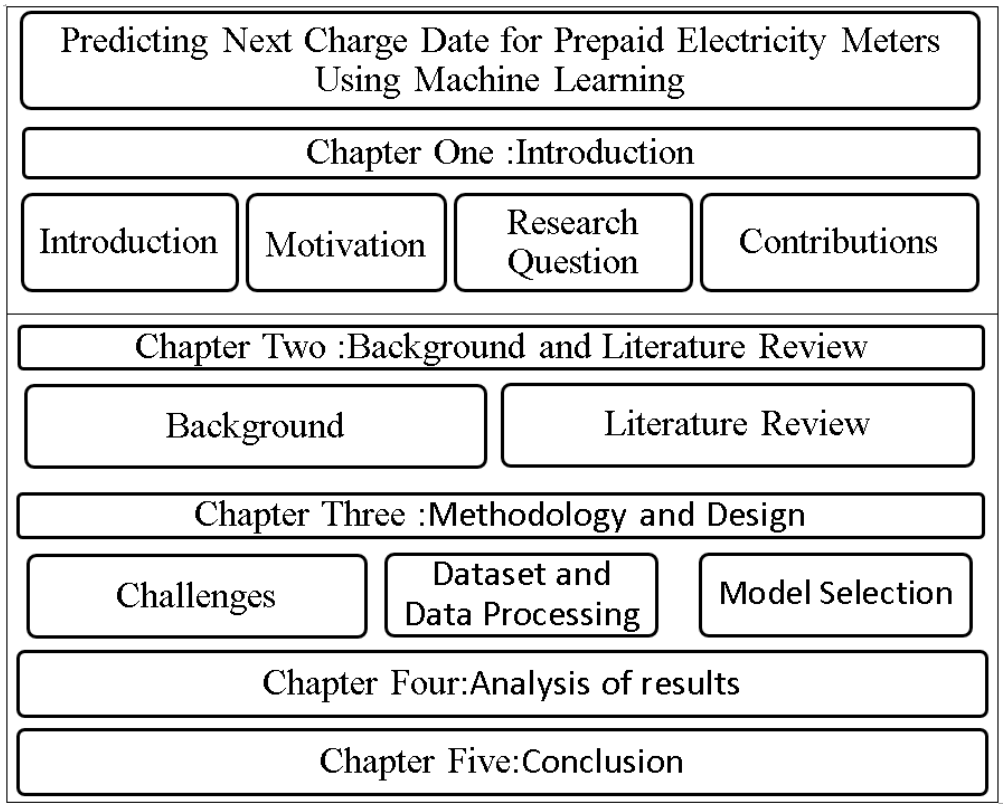

Figure 1.1: Thesis structure

1.5 Thesis Organization

This thesis contains five chapters, chapter 1 is an introduction and related background knowledge, followed by motivation and contribution, chapter 2 is Background and Literature Review, chapter 3 is an introduction contains methodology and system design, chapter 4 is analysis of results, chapter 5 is the conclusion. The chapters of the thesis are organized as Figure1.1

Chapter 2

Background and Related Work

Contents

2.1 Background

2.1.1 JDECo Company

The Jerusalem District Electricity Company (JDECO) is a Palestinian electricity distribution company that was established in 1956. It holds the exclusive rights to supply electricity to consumers in the districts of East Jerusalem, Bethlehem, Ramallah, and Jericho [18]. While JDECO doesn't operate its own power stations, it purchases over 95% of its electricity from the Israel Electric Corporation (IEC) and the remaining portion from the Jordan Electric Power Company (JEPCO) for use in the Jericho district. The company serves electricity for more than 300 thousand customers and its concession area currently covers approximately 25% of the area of the West Bank, equivalent to 366 square kilometers, distributed as follows:

- Jerusalem area: It includes 47 villages and towns and covers an area of 82 square kilometers (not including, of course, occupied Jerusalem in 1948)

- Ramallah area: It includes 72 villages and towns and covers an area of 174 square kilometers
- Bethlehem area: It includes 43 villages and towns and covers an area of 80 square kilometers
- Jericho: It includes 7 places and covers an area of 30 square kilometers

2.1.2 Prepaid Meters Background

There are several types of prepaid meters that are used by service companies, especially electricity companies. These types differ in terms of manufacturing and charging mechanism. One of the ways is to use smart cards to charge the meters, where the subscriber goes to the company or its authorized agents and a special program affiliated with the company that manufactures the meter charges this card and writes on it the current shipment information, then the subscriber inserts this card into the special meter to unload the new charge in it [19].

Another way to charge the meter is by using Standard Transfer Specification (STS) token [20]. To charge the meter using token, the customer purchases an encrypted token with 20 digits from the company, where this token works only for that meter. Then, the customer enters the token to the meter by an electronic keypad. Overall, the various types of prepaid meters share the same objective, which is to allow customers to purchase credit from the company or distributors and use that balance to power their facilities. Once the balance is out, meters will cut off the electricity supply until a new charge is made, ensuring that subscribers only use what they have paid for.

Prepaid Meters Types

Table 2.1 represents several prepaid meter types that are used in the electricity company, which provided us with the data set to carry out this thesis. As the dataset contains information for shipments of several types and systems from these meters mentioned in the table

2.1.3 Machine Learning Background

Machine learning (ML) is a branch of artificial intelligence (AI), it empowers computer systems to learn from data and iteratively improve their performance without being explicitly

Meter Type	Meter Image	Charging Mechanism
Holley Meter (V1 + V2) [21]		Smart Card
Holley Meter 3 [21]		STS Code
Hexing Meter [22]		STS Code
Conlog Meter [23]		STS Code

Table 2.1: Overview of meter types and charging mechanisms

programmed. This transformative technology has revolutionized various industries, ranging from finance and healthcare to transportation and gaming, and its impact is set to grow in the coming years.

The roots of machine learning go back to statistics and computational algorithms, and the rise in it can be attributed to the availability of huge amounts of different types of data, powerful computing systems, and breakthroughs in algorithms and technologies that made their implementing a reality. This has led to a leap in the development of technology and deployment of machine learning models, from simple linear regression and decision trees to sophisticated deep neural networks and reinforcement learning. These models can perform a wide range of tasks, including image and speech recognition, natural language processing and defect detection, and predicting such things as market prices and customer behavior.

Machine learning has been instrumental in fraud detection, risk management, and portfolio optimization in the financial industry [24]. In healthcare, machine learning models have shown promising results in disease diagnosis and drug discovery [25]. Machine learning has also changed the landscape of social media, enabling personalized content recommendation and targeted ads [26].

Machine Learning Algorithms

Machine learning nowadays are used extensively in our world. Different Machine learning algorithms continually being changing and expanding in the field. Each algorithm has unique strengths and weaknesses that must be carefully considered before use.

For classification tasks, supervised algorithms such as Decision Trees, Naive Bayes, K-Nearest Neighbors, and Support Vector Machines are popular. These algorithms identify patterns in data and use decision boundaries to classify new data points. They are trained on pre-prepared data that includes all categories and classes needed for classification. Regression tasks require different algorithms, such as linear regression, and polynomial regression. These algorithms analyze data using statistical models to predict future outcomes. They can be supervised or unsupervised, depending on the task at hand. They are considered supervised when used to predict a continuous output variable and unsupervised when used for clustering or dimensionality reduction.

There are some algorithms that can be used for both cases of classification or prediction. Where there is an overlap in these algorithms that make them able to take on such tasks. An example of these algorithms is Recurrent Neural Network.

So, it is important to understand the problem domain, analyze and understand the characteristics of the data for the problem in order to correctly choose an algorithm to build an appropriate model.

Artificial Neural Network

The idea of artificial neural networks is an idea inspired by the way the human mind works. Where neural networks consist of several layers, usually known as the data input layer, the hidden layer, and the output layer. Each layer contains nodes, so that each node contains numbers that represent its weight. These numbers are calculated during the learning process Nodes are connected to each other. The value of the nodes passes through the activation function, and it determines whether the result of the next node will be activated or not [27].

Recurrent Neural Network

Recurrent neural networks (RNN) are a type of artificial neural network (ANN) that are characterized by having cycles or temporal links. Recurrent neural networks were built in the late eighties 1986 by Jeffrey Hinton and David Rummelhart, who laid the foundation stone for this technology. Some modifications and updates were added to recurrent neural

networks to overcome some of the problems they were facing. One of the most common problems encountered by recurrent neural networks is the vanishing gradient problem. The problem of vanishing gradient occurs when the values of the error derivatives (gradients) of the deep parameters in the network become too small during the training process. As a result, the update that these parameters receive is very minimal, resulting in slow learning or a complete learning halt. To overcome this problem, Sebastian Hochreiter and Jürgen Schmidhuber proposed a solution to in 1997, by creating what is now known as LSTM [28]. Gated Recurrent Unit (GRU) were developed in 2014 by Kiyojiro Miyazawa, John Scholek, and Felix Gibsat [29], and are a simplification of the LSTM model.

Recurrent neural networks rely on short-term memory to represent temporal and sequential information. At each time step, new information is entered while preserving information from the previous time steps. The hidden state of the network is updated as each new element of the sequence is processed [30]. This allows the network to learn the long-term dependencies between the different elements in the sequence. The main goal of RNN is to learn and process long- and short-term data sequences. It is a type of artificial neural network that features feedback loops, allowing information to be stored across time steps during training and prediction. RNN is designed to handle problems with a temporal dimension, such as sequence prediction, text and speech recognition, and real-time video analysis.

In deep learning problems, data may be composed of long sequences of inputs and outputs. Recurrent neural networks process these sequences by holding a memory of past states and combining it with current inputs to provide contextual knowledge of the data. There is a difference between RNN and other NN, for example, in Feedforward Neural Networks (FFNN), it flows in a single direction, from the input layer through hidden layers (if any) to the output layer [31]. There are no cycles or loops in the connections between the layers, which distinguishes it from recurrent neural networks.

Activation Functions for RNN An activation function determines if a neuron should be activated, essentially deciding the significance of the neuron's input to the network for making predictions through basic mathematical operations [32]. The purpose of the activation function is to generate an output based on a collection of input values provided to a node (or layer). RNNs commonly employ several activation functions to introduce non-linearity into the model and help with learning complex patterns in the data. Figure2.3 shows some of the most common activation functions for RNNs.

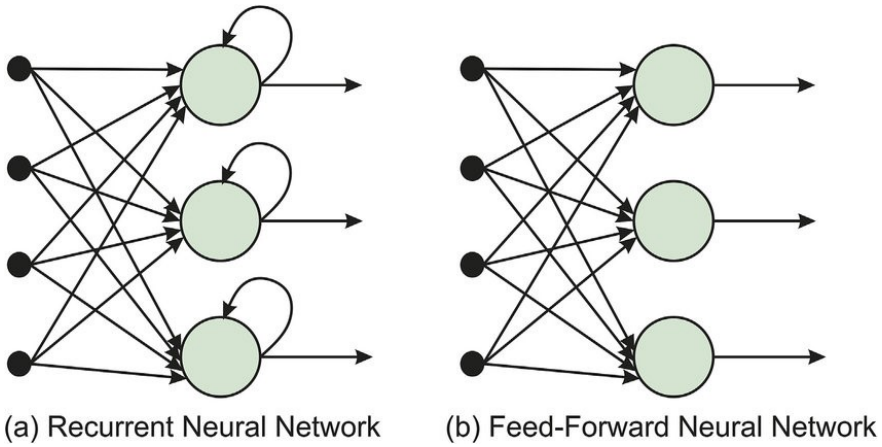

Figure 2.1: The comparison between RNN and FFNN [1].

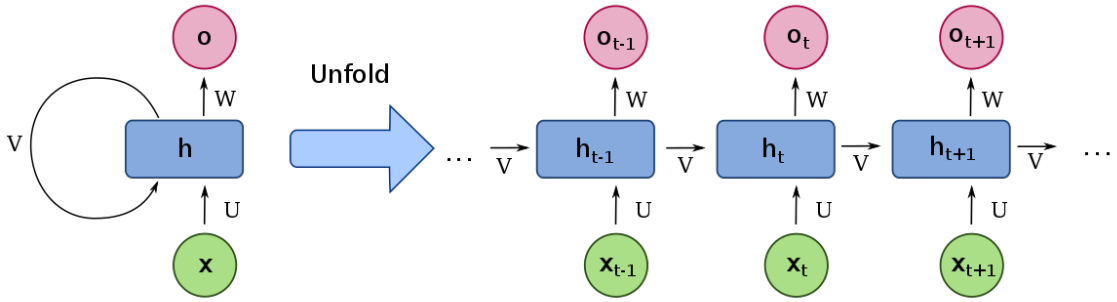

Figure 2.2: A diagram for a one-unit recurrent neural network (RNN). From bottom to top : input state, hidden state, output state. U, V, W are the weights of the network. Compressed diagram on the left and the unfold version of it on the right [2].

Sigmoid (logistic) function

Equation:

$$f(x) = \frac{1}{(1 + e^{-x})} \tag{2.1}$$

Usage: The sigmoid function maps input values to the range $(0, 1)$, making it particularly useful for binary classification problems, output probabilities, or as a gate mechanism in more advanced RNN architectures like LSTMs and GRUs.

Hyperbolic tangent (tanh) function

Equation:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.2}$$

Usage: The tanh function maps input values to the range $(-1, 1)$, providing a better balance of positive and negative values. This makes it a popular choice for hidden layer activation functions in RNNs as it helps with gradient flow and mitigates the vanishing gradient problem to some extent.

Rectified Linear Unit (ReLU) function

Equation:

$$f(x) = \max(0, x) \tag{2.3}$$

Usage: The ReLU function is less computationally expensive and mitigates the vanishing gradient problem effectively. However, it can sometimes result in "dead neurons" in RNNs due to its zero gradient for negative inputs. It is used in RNNs when computational efficiency is prioritized and the risk of dead neurons is deemed acceptable.

SoftPlus function

Equation:

$$f(x) = \log(1 + e^{x}) \tag{2.4}$$

Usage: Softplus is used as an activation function in various types of neural networks, including feedforward, convolutional, and (RNNs). Its smooth nature makes it particularly useful in scenarios where a non-linear, but smooth gradient is beneficial.

Linear function

Equation:

$$f(x) = x \tag{2.5}$$

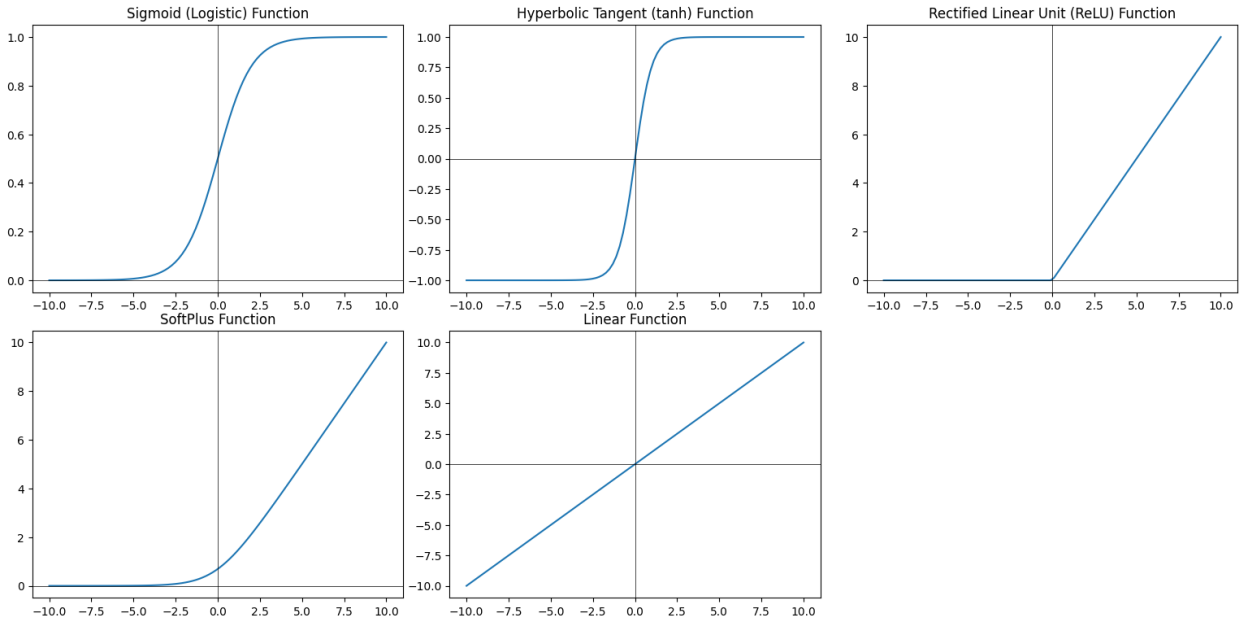

Figure 2.3: Most common activation functions for RNN

Usage: The usage of Linear activation function is when the output is continuous. Typically used for regression problems.

Each activation function has its advantages and limitations, and the choice depends on the specific problem and requirements of the RNN architecture. Usually for regression problem, RNN may use in its hidden layers ReLU or tanh and leaving the final output layer without activation function, while for the classification problems, the final output layer of the neural network should use either sigmoid for binary classification, or softmax for Multi-class classification.

Layer Types for RNN

RNN have different types of layers. Each layer differs in its development and the way of work. The following RNN layers are the common used layers:

- SimpleRNN: Simple RNN is a fully connected RNN which suits for short sequences. One of the problem that SimpleRNN has is gradient vanishing problems, where the gradient become relatively small which affect changing the weights values
- LSTM: LSTM is a Long Short Term Memory layer type, where it is a complex type of layer for RNN. LSTM has a memory in it cells allow it to retain information from

long sequences and short steps

- GRU: GRU is a Gated Recurrent Units, which is similar to LSTM, but simpler. Usually used for less complex problems or smaller datasets. GRU usually needs less computational compared to LSTM, and faster to train in some cases. Both GRU and LSTM are developed to solve the gradient vanishing problem that SimpleRNN has

Metrics for Machine Learning

Performance metrics, also known as error measures, play a crucial role in assessing models across various domains. A performance metric can be described as a logical and mathematical structure designed to evaluate the closeness of actual outcomes to their expected or predicted values for a certain model [33]. In another word, metrics are used to show how well the model is.

There are various types of metrics that could be used for different types of models and problems, and each metric has its own mathematical equations and its value reflects the evaluation differently. Table2.2 shows most common metrics for both classification and regression tasks and the formula for each

Classical Machine Learning Algorithms

There is a wide array of algorithms available for creating models that address various types of problems, including regression and classification issues. Numerous research papers have explored and utilized these algorithms, contributing to the development and refinement of their methodologies. In the following section, we will provide a brief overview of the most significant algorithms that have been employed in other research papers to solve diverse problems.

Linear Regression

Linear regression is one of the oldest and most straightforward machine learning algorithms, serving as a foundation for understanding the behavior and relationship between a dependent variable and one or more independent variables. At its core, linear regression models the relationship by fitting a linear equation to observed data. The equation for a simple linear regression, which involves a single independent variable, is $y = \beta_0 + \beta_1 x + \epsilon$, where y is the dependent variable, x is the independent variable, β_0 is the y-intercept, β_1 is the slope of the line, and ϵ represents the error term [34]. The goal of the algorithm is to

Metric	Task	Description	Formula	Is Higher Value Better
Accuracy	Classification	Proportion of correctly classified instances	$\frac{TP+TN}{TP+FP+TN+FN}$	Yes
Precision		Proportion of true positive predictions out of all positive predictions	$\frac{TP}{TP+FP}$	Yes
F1 Score		Harmonic mean of precision and recall, providing a balanced measure between them	$\frac{2\cdot(Precision\cdot Recall)}{Precision+Recall}$	Yes
Mean Absolute Error (MAE)	Regression	Average of the absolute differences between predicted and actual values	$\frac{1}{n}\sum_{j=1}^{n}\lvert y_i - x_i\rvert$	No
Mean Squared Error (MSE)		Average of the squared differences between predicted and actual values	$\frac{1}{n}\sum_{j=1}^{n} e_j^2$	No
Root Mean Squared Error (RMSE)		Standard deviation of the prediction errors	$\sqrt{\frac{1}{n}\sum_{j=1}^{n} e_j^2}$	No

Table 2.2: Metrics and formulas

find the best-fitting line through the data points that minimizes the sum of the squared differences between the observed values and the values predicted by the line, a method known as Ordinary Least Squares (OLS). Linear regression assumes a linear relationship between the dependent and independent variables, which can be a limitation in real-world scenarios where complex, nonlinear relationships exist. Additionally, it is sensitive to outliers, which can significantly influence the model's predictions

Extreme Gradient Boosting (XGBoost)

Extreme Gradient Boosting, or XGBoost, is an advanced machine learning algorithm that builds upon the foundation of gradient boosting. It is designed to construct a sequence of models in which each subsequent model aims to rectify the errors made by its predecessor. This process is iteratively carried out to improve the overall predictive accuracy of the ensemble model. The objective function of the XGBoost model is determined by combining the loss function with a regularization term. The loss function serves to measure the predictive

power of the model, quantifying the difference between the predicted and actual values. On the other hand, the regularization term acts as a penalty, controlling the complexity of the model by discouraging overfitting. This balance between model performance and complexity helps to prevent the XGBoost model from fitting the training data too closely, which could result in poor generalization to new data. One of the key strengths of XGBoost lies in its versatility and efficiency. It supports various loss functions, making it adaptable to a wide range of predictive modeling tasks, including regression, classification, and ranking. Additionally, XGBoost incorporates several innovative techniques to enhance performance and speed, such as a novel tree learning algorithm and an efficient handling of sparse data. These features, combined with its capability to perform parallel computation and scale effectively across multiple cores, render XGBoost an exceedingly powerful tool for tackling complex machine learning challenges.

Polynomial Regression

Polynomial regression is a type of regression analysis where the relationship between the independent variable(s) and the dependent variable is modeled as an nth degree polynomial. This technique is particularly useful for capturing the non-linear relationship between the dependent and independent variables [35]. By regressing the dependent variable on powers of one or more independent variables, polynomial regression allows for the modeling of more complex, nonlinear dynamics that cannot be captured by simple linear regression. This approach can be instrumental in uncovering underlying patterns in the data, such as curvilinear trends, that are essential for making accurate predictions or understanding the behavior of variables in various scientific and engineering contexts.

Random Forest

Random Forest is a well-established machine learning technique used for a variety of tasks, including regression and classification. As an ensemble learning method, it combines the predictions of numerous simple models, often referred to as "weak learners," to form a more accurate and robust model [36]. The core components of a Random Forest are decision trees, which work by dividing the input space into distinct, as homogeneous as possible, regions.

One significant benefit of the Random Forest algorithm is its ability to assess the importance of features. This is accomplished by measuring the decrease in impurity associated with nodes in the trees that utilize a particular feature, a process that is repeated across all trees in the forest. However, Random Forests can become computationally intensive and require a longer training time as the size of the dataset increases.

Decision Trees

Decision Tree (DT) was built as the name says. In its internal building, each node represents a feature that has a branch, where each branch represents a rule decision that outputs a node output. It has a root node that series of decisions that makes decisions based on the attributes thus leads to prediction or classification, depending on the task [37]. Decision Trees adeptly handle non-linear relationships between variables, making no assumptions about the distribution of the data. They also excel in managing both numerical and categorical data, offering a versatile tool for a wide range of applications. Moreover, their capability to deal with missing values and resist the influence of outliers further underscores their robustness and adaptability in diverse analytical scenarios.

K-Nearest Neighbors (KNN)

The K-Nearest Neighbors (KNN) algorithm is a simple yet versatile method used in machine learning for both classification and regression tasks [38]. It operates on the principle of feature similarity, where predictions for new instances are based on the most similar cases (or nearest neighbors) in the training dataset. This similarity is typically determined by calculating the distance between instances using metrics such as Euclidean or Manhattan distance. The prediction is then made through a majority vote (for classification) or by averaging (for regression) the outcomes of the 'k' nearest neighbors.

Despite its simplicity, KNN faces several challenges. Its computational efficiency drops as the dataset size increases, requiring the calculation of the distance between the query instance and each example in the training set, making it resource-intensive for large datasets [39]. Furthermore, KNN's performance heavily depends on the choice of the 'k' parameter; selecting an unsuitable 'k' can lead to overfitting or underfitting, affecting the algorithm's ability to generalize from training to unseen data. KNN is also sensitive to the scale of the data and the presence of irrelevant features, which can negatively impact accuracy if the data is not properly normalized or if irrelevant features are not excluded. The algorithm struggles in high-dimensional spaces due to the "curse of dimensionality," as the increased dimensions make the concept of "nearest neighbors" less clear due to the increased distance between points. Lastly, unlike model-based approaches, KNN does not provide an explicit model representation, which limits its interpretability and the ability to understand the underlying patterns in the data.

2.2 Literature Review

This section presents an in-depth literature review focused on various research studies related to machine learning and its applications in predicting electricity patterns, customer behaviors, and other interconnected subjects. We delve into a range of academic works, analyzing and synthesizing their methodologies, findings, and the implications they hold for the field of machine learning. We split and categorize the reviews by distinct area of focus for each paper

- **Machine Learning in Energy Consumption Prediction**
  Keytingan et al. [40] provide an overview of energy consumption prediction using machine learning, focusing on techniques like regression analysis, neural networks, decision trees, and SVM. They emphasize the role of machine learning in energy efficiency, particularly in smart buildings. Risul Islam Rasel et al. [41] propose a machine learning approach to predict the electric energy use of a low-energy house using weather and occupancy data. The dataset includes hourly measurements of electric energy consumption, outdoor temperature, and occupancy status for one year. Two machine learning algorithms, support vector regression (SVR) and Artificial Neural Network (ANN), are used to build predictive models. The authors use the root mean square error (RMSE) and the mean absolute error (MAE) as performance metrics to evaluate the models' accuracy. The results show that both SVR and ANN perform well in predicting electric energy use, with ANN outperforming SVR in terms of RMSE and MAE. The paper also includes a feature selection analysis using principal component analysis (PCA) and F-test to determine the most relevant features for predicting energy consumption.

- **Machine Learning in Customer Behavior and Churn Prediction**
  Toderean et al. [42] focus on the use of machine learning algorithms for churn prediction in the prepaid mobile telecommunications industry. They provide a review of various feature selection techniques, including correlation analysis, principal component analysis, and recursive feature elimination. The article presents a case study of churn prediction in the prepaid mobile telecommunications industry using a SVM algorithm. The authors describe the data collection and preprocessing steps, feature selection, model training, and evaluation. They compare the performance of the SVM algorithm to that of other machine learning algorithms and highlighted the importance of feature selection in improving model accuracy. Bart Larivière et al. [43] discuss predicting cus-

tomer retention and profitability using Random Forests and Regression Forests, with a focus on customer behavior and transactional data. Prasad Bhosale et al. [44] propose a machine learning-based model for customer churn prediction in the telecom industry, using data mining methods and feature engineering.

- **Predictive Modeling in Financial Markets**
  Roshan Christy R et al. [45] work were to predict next electricity bill amount. Their work focused on classical machine learning algorithms. They built their models depending on customers socioeconomic and housing characteristics, like how many rooms in the customer house, or does he has a TV or not. The difference between his work and ours is that they tried to predict bills for postpaid meters, but we are trying to predict next charge date for the prepaid customers, also they depend on housing data which couldn't be available for all customers, while we are focusing on the customer historical records to predict his next charge date. C R Karthik et al. [46] compare the performance of DNN and LSTM models in predicting daily variance in financial markets, specifically analyzing the NIFTYIT index.

- **Machine Learning for Purchase Behavior Analysis**
  Evans [14] presents a study on predicting customers' next purchase day using algorithms like decision trees, random forests, and SVM, with a focus on feature selection and model training. Jing Li et al. [47] and Zeng et al. [48] propose methods for predicting online purchasing behavior, emphasizing the role of SVM and logistic regression classifiers in understanding customer demographics and shopping behaviors. Qian Liu et al. [49] introduce deep learning models for consumer behavior analysis, discussing the effectiveness of rDNN and KmDNN models compared to traditional models

Table2.3 and Table2.4 present a comparison of various related studies. It includes the algorithms utilized in each study and the highest accuracy values reported for each algorithm.

Table 2.3: Literature review summary - Part 1

Work	**Research About**	**Publish Date**
Keytingan et al. [40]	Energy consumption prediction by using machine learning for smart building: Case study in Malaysia	2021

Continued on next page

Table 2.3 – continued from previous page

Work	Research About	Publish Date
Toderean et al. [42]	Methods for Churn Prediction in the Pre-Paid Mobile Telecommunications Industry	2016
Evan [14]	Predict customers Next Purchase Day for Grocery	2021
Risul Islam Rasel et al. [41]	Predicting Electric Energy Use of a Low Energy	2019
Bart Larivière et al. [43]	Predicting customer retention and profitability by using random forests and regression forests techniques	2005
Jing Li et al. [47]	Using Support Vector Machine for Online Purchase Predication	2017
Zeng et al. [48]	User behavior modeling, recommendations, and purchase prediction during shopping festivals	2019
Qian Liu et al [49]	Deep Learning-Based Consumer Behavior Analysis and Application Research	2022
Prasad Bhosale et al [44]	A Dynamic Churn Prediction Model using Machine Learning Approach	2021
C R Karthik et al [46]	Forecasting variance of NiftyIT index with RNN and DNN	2022
Eyden Samunderu et al . [50]	Predicting customer purpose of travel in a low-cost travel environment—A Machine Learning Approach	2022
Roshan Christy R et al. [45]	Prediction of Electricity Bill using Supervised Machine Learning Technique	2022

Table 2.4: Literature review summary - Part 2

Work	Algorithms	Metrics	Value	Records	Features
[40]	k-NN	MAPE	0.94%	+10K	-
	SVM		0.67%		
	ANN		1.84%		

Continued on next page

Table 2.4 – continued from previous page

Work	Algorithms	Metrics	Value	Records	Features
[42]	SVM	Accuracy	99.70%	+3K	21
	BN		99.10%		
	MLP		99.55%		
[14]	XGBClassifier	Accuracy, F1-Score	92%, 0.85	+1M	8
	LogisticRegression		90%, 0.79		
[41]	SVR	MSE, RMSE	1.32%, 11.49%	-	-
	ANN		0.14%, 0.18%		
[43]	Random Forests	AUC	0.714	100K	-
	Logistic Regression		0.695		
[47]	SVM	F1-Score	0.046	10K	7
[48]	Logistic Regression	Accuracy	74%	+500K	-
[49]	DNN	AUC	0.7893	+ 2M	159
	rDNN		0.8322		
	KmDNN		0.8064		
[44]	Random Forest	Accuracy	95%	+5K	-
	DT		89%		
	BaggingClassifier		94%		
	KNN		81%		
[46]	DNN	MSE	9.6571	+25K	6
	RNN		8.233		
[50]	Random Forests	Precision	0.90	+67K	161
[45]	SVR	R2 Score	-0.56%		10
	Decision Tree		72.40%		
	Random Forest		84.97%		
	Linear Regression		88.09%		
	Ridge Regression		88.52%		

Based on previous literature reviews, it is evident that related work has focused either

on predicting prepaid churn or on customers' next purchase of a specific item in grocery or online shopping. However, in our model, we take a different approach by predicting the next charge date for prepaid electricity meters. We believe our model can be applied to any prepaid meter in utility services, providing a practical solution for utility companies looking to improve customer service.

Chapter 3

Methodology and Design

Contents

To construct a machine learning model, several preliminary steps must be completed. Figure 3.1 illustrates the common steps required to build such a model. This process outlines

the essential steps for developing, deploying, and maintaining our predictive model. Our methodology unfolds as follows:

1. Define the Problem: We start by understanding the problem to correctly identify how to solve it. A clear understanding of the problem guides us in selecting suitable regression algorithms for predicting numerical outcomes due to their appropriateness.

2. Data Collection: We obtain data exclusively from prepaid meters provided by the JDECo company.

3. Data Preprocessing and Exploration: Before modeling, we perform the preprocessing stage. This includes visualizing the data to identify patterns and insights, conducting statistical analysis to understand its distribution and relationships, and cleaning the data to correct inconsistencies or remove outliers.

4. Baseline Model Training: Initially, we train our model using the raw dataset without any feature engineering. This baseline model serves as a reference point, allowing us to measure the impact of further feature selection, engineering, and optimizations on model performance.

5. Feature Selection and Engineering: We then extract and engineer new features that are likely to enhance the model's predictive accuracy.

6. Model Selection and Training: We proceed to select and train a more sophisticated machine learning model. This step involves experimenting with various algorithms and configurations to identify the most effective solution for our prediction task.

7. Model Evaluation: We evaluate the model using metrics suitable for regression tasks, such as mean absolute error (MAE) or root mean squared error (RMSE), to determine the model's accuracy.

8. Model Fine-tuning and Optimization: Based on the evaluation results, we engage in fine-tuning and optimization to enhance the model's accuracy. This may include adjusting model parameters, employing more advanced feature engineering techniques, or exploring alternative modeling approaches.

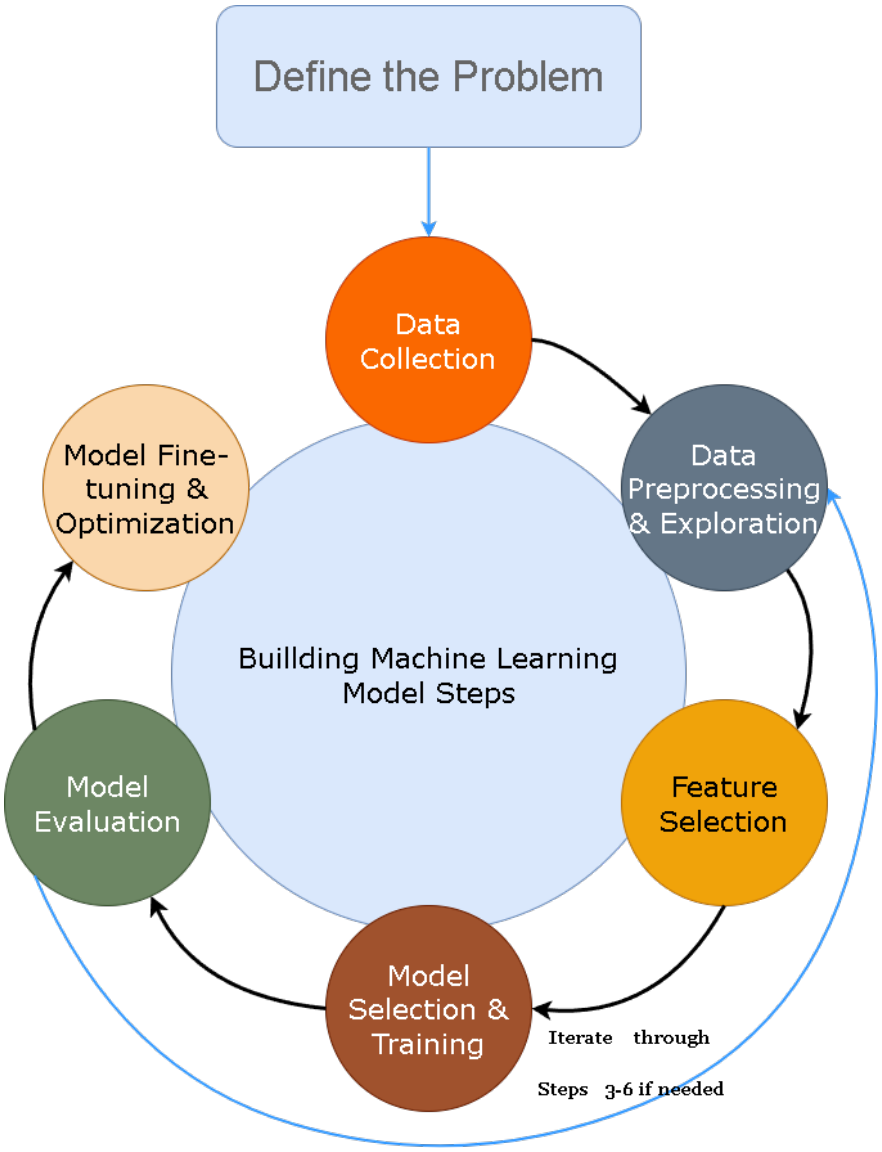

Figure 3.1: Building ML steps

3.1 Challenges

Modeling a problem to be solved in the machine learning context can face several challenging issues in general. Specifically, in our NCDP, we face the following issues:

1. Data Quality: Data quality can dramatically affect performance if left unprocessed or processed incorrectly. Therefore, we first perform data visualization and analysis to understand the relationships between the features, their correlations, and data statistics. This step aids us in cleaning the data by identifying null values and handling them. It also assists us in performing feature selection and feature engineering.

2. Selecting Features: In the area of feature selection, our dataset presents a unique set of challenges. It includes many features, some of which are relevant to our predictive task, while others are not. Selecting features is an important step in building an effective machine learning model. This process is particularly vital in our context because it directly affects the accuracy of predicting the next charging date for prepaid meters. To address this challenge, we implement a three-method approach. First, we use the "SelectKBest" method from the Scikit-learn library [51], a powerful feature selection tool that aids in identifying the most significant features based on statistical tests. This method enables us to isolate features that have the greatest predictive power with respect to the target variable. Using this technique illustrates the importance of the features we extract and engineer, and allows us to enhance our model prediction. Second, we apply Principal Component Analysis (PCA). The third method involves grouping features into relative categories and running tests over these categories separately and combined to see their effect on the model.

3. Model Selection: Choosing the right machine learning model that fits the nature of our data is a challenge. Different models have different strengths and weaknesses. Additionally, tuning hyperparameters for optimal performance requires a careful balance between model complexity and generalization ability to avoid overfitting or underfitting. So address this challenge, we will use different classical machine learning algorithms as a baseline, then we will build a Neural Network Model approach using RNN. For finding hyper parameters for RNN, we will conduct various methods to search for optimal hyper parameters, focusing on a Hyper Search mechanism using Random Search by Keras, which will loop through previously setup parameters with steps to find optimal hyperparameters. Also, we will build our Model using different layers

types to find the best layer types that suits our problem.

4. Scalability and Efficiency: As data volumes grow, ensuring that our algorithms can scale effectively without significant loss in performance becomes a challenge. This includes considerations for computational resources and algorithm efficiency. In this step, we are building our model using different record counts to observe the effect of dataset size and the performance of the models in relation to this.

3.2 Dataset and Data Processing

3.2.1 Data Collection

The dataset for prepaid meters used in this thesis was obtained from the Jerusalem District Electricity Company (JDECo). The dataset encompasses charges made across several types of prepaid meter systems. Although these systems have files with different names and schemas, they ultimately contain the same information necessary for this study. We have consolidated the data from various system types into three flat CSV files. The primary dataset file contains detailed charge information for all customers within a specific period, while the other files include customer service information, such as meter amp class, among other data. The company operates 10 different prepaid systems for various meter types, and data from all these meter types have been collected. The dataset spans from January 2021 to December 2021 and comprises approximately 850,000 records with 39 features. Table 3.1 illustrates some of the features in the dataset.

Feature	**Description**
`meterNo`	A unique identifier for each prepaid meter
`customerNO`	A unique identifier for each customer, it also means service number
`paymentDate`	The date of specific charge record
`KWTotalPrice`	The amount of money the customer paid for watt for that specific record
`KWAmount`	The amount of KW the customer purchased for that specific record
`previousChargingDate`	The date of the previous charge for specific charge
`accumulatedKWConsumed`	The total amount of electricity used by the customer
`tariffID`	The tariff ID that represents the KW price and fixed amount per day
`branchID`	The branchID represents a classification number for the Governorate of each customer

Table 3.1: Selected features from the charging dataset

Feature	Description
customerNO	Foreign key customer number
facilityType	The type of the facility that the meter serves
phasesAMPClass	A category class that represents the number of meter phases. It also represents how many amperes this meter can supply
areaID	The area id where the meter (the service) is located. For example, if the meter is located in Jerusalem Bait-Hanina, the branchID would be 8, and the area ID 12
subBranch	The city id where the meter (the service) is located.

Table 3.2: Features within the service information dataset

Feature	Description
facilityType	A number represents the type of the facility.
facilityGroupID	A number represents the group that the facility refers to, like Commercial, Residential.

Table 3.3: Features within the facility dataset

Cust. NO	Pay AMT	Pay. Date	P. Charging Date	Tarrif	KW AMT	Date Diff.	Branch ID
3507073150	200	12/7/2021	12/3/2021	71	332	NULL	35
3507073150	200	12/13/2021	12/7/2021	71	348	NULL	35
3507073150	200	12/19/2021	12/13/2021	71	346	NULL	35
3507073150	200	12/24/2021	12/19/2021	71	346	NULL	35
7047102790	300	1/26/2021	12/3/2020	21	534	NULL	70
7047102790	150	2/9/2021	1/26/2021	21	269	NULL	70
7047102790	220	2/23/2021	2/9/2021	21	391	NULL	70

Table 3.4: Sample of charges dataset (raw dataset) records/features

3.2.2 Data Privacy and Protection

In this study, we have taken several measures to uphold the highest standards of data privacy and protection, in line with ethical guidelines.

Anonymization of personal data The dataset includes historical data on prepaid meter charges and consumption patterns. To protect the privacy of individuals involved, all personally identifiable information (PII) associated with customer accounts has been deleted. This includes excluding names, phone numbers and full addresses from the data set. By removing these elements, we ensure that the data cannot be directly linked to any individual, significantly reducing the risk of privacy breaches.

Masking of Customer Numbers In addition to omitting direct identifiers, we have employed a masking technique on customer numbers, which could potentially be used to

infer the identity of the customers. Masking involves replacing the original customer numbers with pseudonyms or non-identifiable codes. This process ensures that each customer's data remains confidential and secure, as the masked identifiers do not retain any linkage to the customers' actual identities or any other personal information.

Data Access and Handling Protocols Access to the anonymized and masked dataset is strictly controlled and limited to the research team involved in this study.

3.2.3 Target Feature Creation

Our target feature is a numerical value representing the number of days it takes a customer to recharge his/her meter. In the dataset we obtained, this target feature does not exist as a standalone feature, and therefore, it needed to be created. We accomplished this through the following steps: First, we sorted the dataframe by two columns: 'customerNO' and 'paymentDate', arranging the charges from oldest to newest. Next, we calculated the difference in days until the next payment for each customer. This was achieved by grouping the data by 'customerNO' and then applying the 'shift(-1)' method to shift the data upwards in the 'paymentDate' column. The result of this subtraction provides the number of days until the next charge, which serves as our target feature. Subsequently, we converted this time difference into a numeric format representing days. It is expected that a null value for our target feature will be generated for every customer's last record, as the next charge date for the last entry is unknown. This record will be dropped during data cleaning. We named the target feature 'daysDiffToNextCharge'.

Customer NO	Payment Date	Days Diff to Next Charge
3507073150	12/7/2021	6
3507073150	12/13/2021	6
3507073150	12/19/2021	5
3507073150	12/24/2021	NULL
7047102790	1/26/2021	14
7047102790	2/9/2021	18
7047102790	2/27/2021	NULL

Table 3.5: Demonstration of target feature

3.2.4 Data Preprocessing

Generally, real-world data, which is also known as raw data, may be characterized by incompleteness, noise, and inconsistency. Incomplete data refers to the absence of specific attribute values, while noisy data encompasses errors, missing values, and outliers. Inconsistent data, on the other hand, involves discrepancies in codes or names [52]. Therefore, we need to perform data preprocessing which involves in cleaning data, visualization, feature engineering and statistical analysis

Data Visualization and Statistics

Data visualization helps us to understand the relation and correlation between features by looking to data exploration through visualization and statistical analysis. Visualization and statistical analysis helps to understand the underlying patterns, trends, and correlations between the features [53]. By examining the data, we can gain insights into the underlying patterns, trends, and correlations between features, which will help make informed decisions when selecting features and building machine learning models. We can start by showing statistical data about important features like "paymentAMT", and our target feature "daysDiffToNextCharge". Understanding these statistics can be very useful in our model, since it can give an initial understanding of data distribution. Also the Min, Max, and Interquartile Ranges (25%, 50%, 75%) can help identify outliers in the data

Statistics	**Value**	**Description of Statistics**
Count	851,971.0	Number of non-null records
Mean	129.429	Mean of the values
Standard Deviation	121.043	Standard deviation of the records
Minimum	0.0	Minimum value for a payment
25% Quartile	50.0	First quartile (25th percentile)
Median (50% Quartile)	100.0	Second quartile (Median, 50th percentile)
75% Quartile	150.0	Third quartile (75th percentile)
Maximum	2,000.0	Maximum value

Table 3.6: Statistical table for payment amount feature

Data Analytics and Analysis

Table 3.6 is a summary which tells us that most payments are relatively small (below 150.0), as the median (50% percentile) is 100.0. However, there are some larger payments as

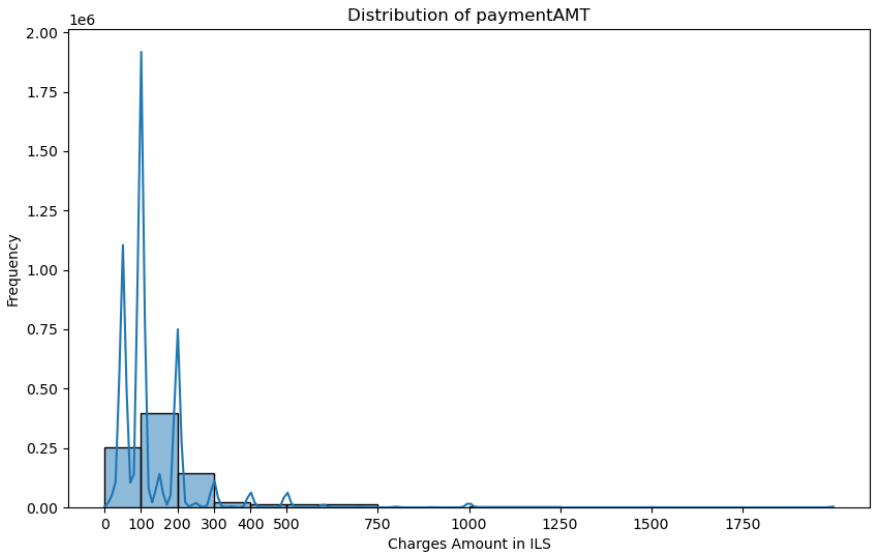

Figure 3.2: Histogram for payment amount feature

well, up to 2000.0. The relatively large standard deviation indicates that there's a significant variation in payment amounts. It also shows that there are some free or other type of charges which could be outliers.

We can also draw a histogram distribution for this feature. Figure 3.2 is a histogram for payment amount feature which shows the histogram distribution for payment amount field. Histogram distribution shows how the data is distributed as frequency term [54]. It also draws a kernel density estimate Line "KDE", which is a smooth curve that represents a probability density function. KDE models the 'probability' of the data distribution. Its curve can exceed the actual frequencies in the histogram because it's showing densities, not frequencies.

The target feature, as summarized in Table 3.7, showcases its statistical characteristics, highlighting a broad range in the number of days until the next charge, spanning from the same day to nearly 10 months later. The average time to the next charge extends just beyond two weeks, yet it is accompanied by a relatively large standard deviation. This suggests that the data is widely dispersed around the mean. Such dispersion indicates the presence of outlier records which could potentially impact model performance if not addressed. Therefore, it is imperative to clean these outliers from the dataset prior to training the model to ensure more accurate and reliable predictions.

Additionally, Figure 3.4 displays a bar chart that illustrates the relationship between various types of facilities and their recharge frequency. The aim of this plot is to uncover

Statistics	Value	Description of Statistics
Count	851,971	Total number of non-null records
Mean	13.074	Mean of the values
Standard Deviation	11.429	Standard deviation of the records
Minimum	0.0	Same day (minimum time to the next charge)
25% Quartile	5.0	First quartile (25% of charges occur within 5 days)
Median (50% Quartile)	10.0	Median number of days to the next charge
75% Quartile	18.0	Third quartile (75% of charges occur within 18 days)
Maximum	296.0	Longest time to the next charge (nearly 10 months)

Table 3.7: Target feature statistics

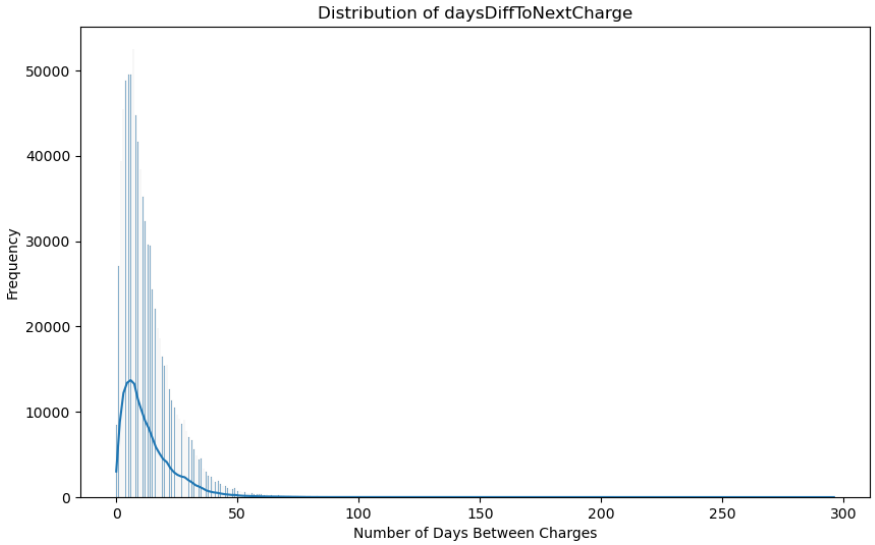

Figure 3.3: Histogram distribution for days to recharge

patterns associated with facility types and their corresponding recharge behaviors. By analyzing this data, our goal is to determine whether there are any new features that can be extracted to improve our model's performance. Identifying such features could provide valuable insights into how different facility types influence recharge behavior, potentially leading to more accurate predictions of recharge events.

From Figure 3.4, we observe significant variations in the charging cycles among different types of facilities. For example, facilities like print shops, epicenters, groceries, supermarkets, and swimming pools exhibit more frequent recharge activities, typically every 2 to 5 days, compared to facilities such as newspapers, elevators, petrol stations, and others. This

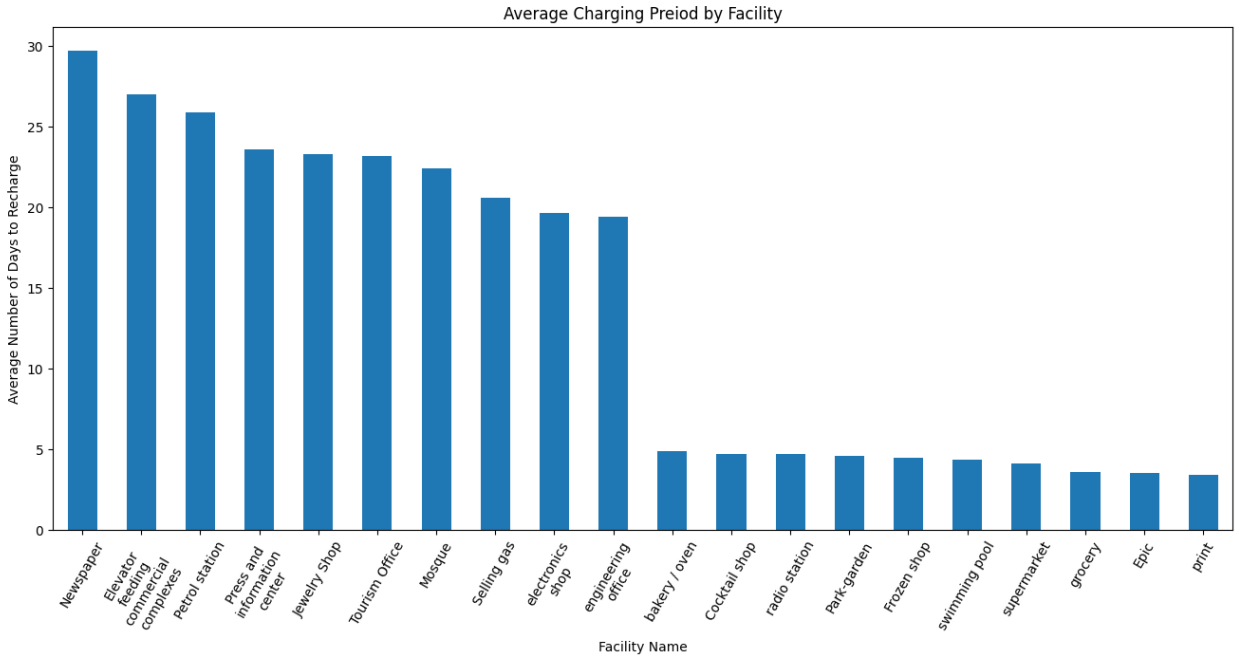

Figure 3.4: Highest/Least 10 facilities average recharge days

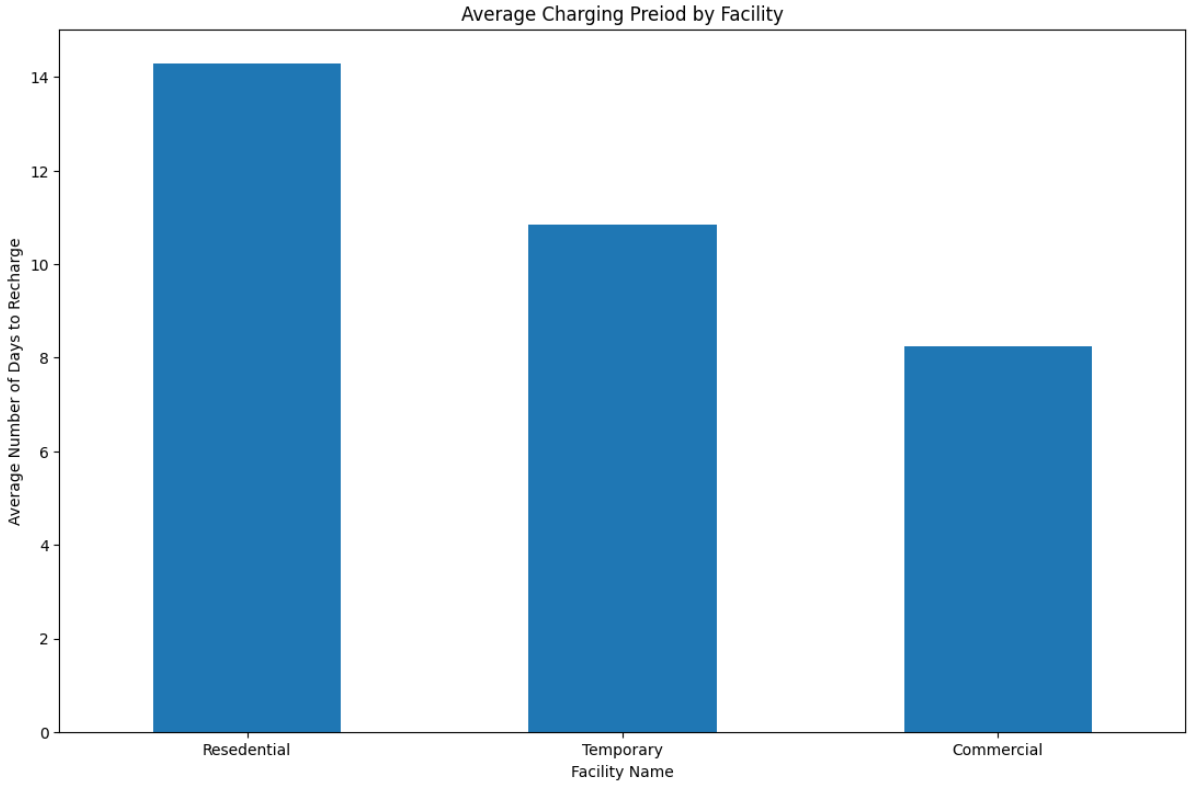

Figure 3.5: Facilities recharge days grouped

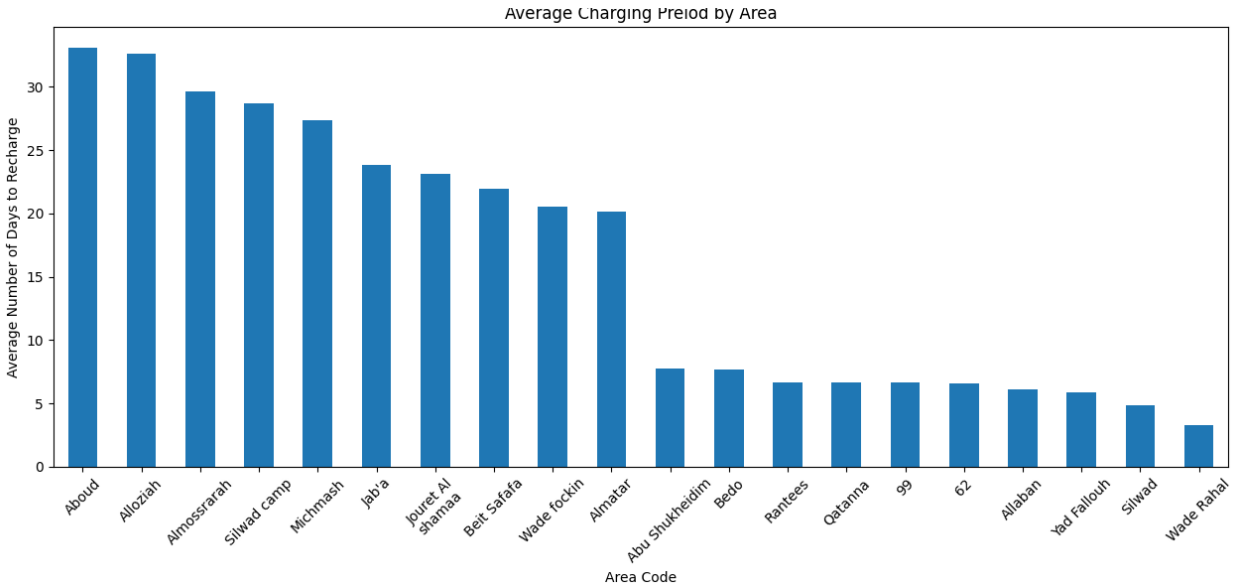

Figure 3.6: Highest/Least 10 area's average recharge days

discrepancy provides valuable insights into the potential benefits of incorporating facility type into the original dataset or extracting new features in subsequent analysis steps. Furthermore, the facilities have been categorized into distinct groups, revealing three primary types of facilities. Figure 3.5 demonstrates a clear differentiation between commercial and residential facilities, with the latter less likely to engage in frequent recharging compared to their commercial counterparts. This categorization and the observed recharge behaviors can significantly influence the development and refinement of predictive models, offering a nuanced understanding of how facility type affects recharge frequency.

Additionally, Figure 3.6 showcases a bar chart visualization that illustrates the recharging patterns of the top and bottom 10 areas, shedding light on their relationship with the frequency of meter recharging across different locations. This visualization reveals notable disparities in the recharging cycles among the areas, which could reflect differences in usage behavior or consumption patterns. For instance, areas with shorter recharge intervals, like Wade Rahal, might consume energy at a faster rate, leading to quicker depletion of their prepaid credit and necessitating more frequent recharging. Such analysis could offer critical insights into the importance of including geographical location in the dataset, potentially aiding in the discovery of new features that could enhance predictive modeling efforts by accounting for regional variations in energy consumption and recharging habits.

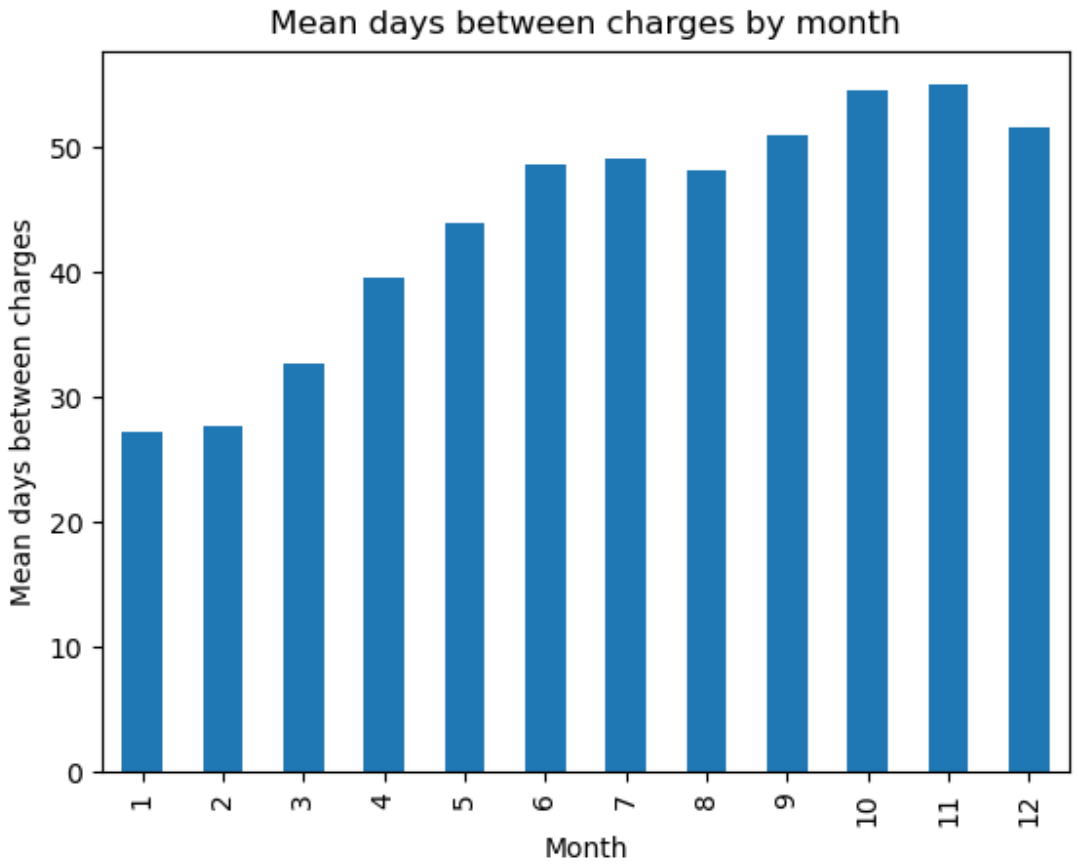

Figure 3.7: Mean days over months

Figure 3.7 presents a bar chart for month vs recharging frequency. We can notice that recharging frequency increased in winter and summer, and decreases in other seasons. This illustrates that there is a relation between the month of the year and meter recharging frequency. The figure shows that in January and February months, customers used to recharge their meters more frequently, which could be related to the low temperature in winter. Also, in summer months, the figure shows that charging frequency less than winter, and these months have almost the same charging frequency. This can give us insights about extracting new features based on the month of the charge

Data Correlation

For our raw dataset, we used the Seaborn framework [55] to generate a correlation heatmap, as shown in Figure 3.8. A correlation heatmap is a method used to display data that rep-

resents the correlation between numerical features in the dataset. This type of heatmap visually displays data using colors to represent values on a 2D grid, where cells show the correlation coefficient between pairs of numerical features. These coefficients reveal the strength and direction of their linear relationships, ranging from -1 to 1. Values close to 1 or -1 indicate strong positive or negative relationships, respectively, while values near 0 show weak or no linear relationships. Negative values mean that as one variable increases, the other tends to decrease, and vice versa. The analysis of the heatmap results aids in understanding the dataset's relationships, assisting in the feature selection process to build a predictive model. The heatmap represents the correlation between all features in the original dataset. From this, we can deduce that there is a strong correlation between pairs of values like "paymentAMT and KWAmount," "paymentAMT and KWTotalPrice," among others, while there is negative or weak correlation between the target feature "daysDiffBetweenLastTwoCharges" and all other features. Consequently, we cannot rely solely on the raw dataset collected from the company and need to employ feature engineering techniques to extract new features. These features will assist in building a predictive model to forecast the next charge date or period for customers

Data Cleaning

Data cleaning is an essential step in the machine learning process that involves identifying, correcting, or removing any inaccuracies, inconsistencies, or errors present in raw data [56]. Data cleaning aims to make the data better and more usable for machine learning algorithms and improve the reliability and quality of the data which will affect positivity and improve the performance and accuracy for a certain machine learning model that built on this data. To perform data cleaning on our dataset, we need to perform several tasks, such as handling missing values, removing duplicates, identifying and handling outliers, feature scaling and normalization. As shown in the heatmap figure 3.8, there are several features that either have null values or where all the values in the column are null. Therefore, we will check how many rows have null values for these features and decide whether to drop them or handle the null values. Typically, several aspects should be considered when handling missing values or missing data. These considerations are summarized as follows:

1. Impact on analysis: If the missing data values will impact the analysis of the data, so it is preferred to impute the missing values rather than dropping it

2. Data Imputation: If we have a reliable way to estimate the missing values, it is preferred

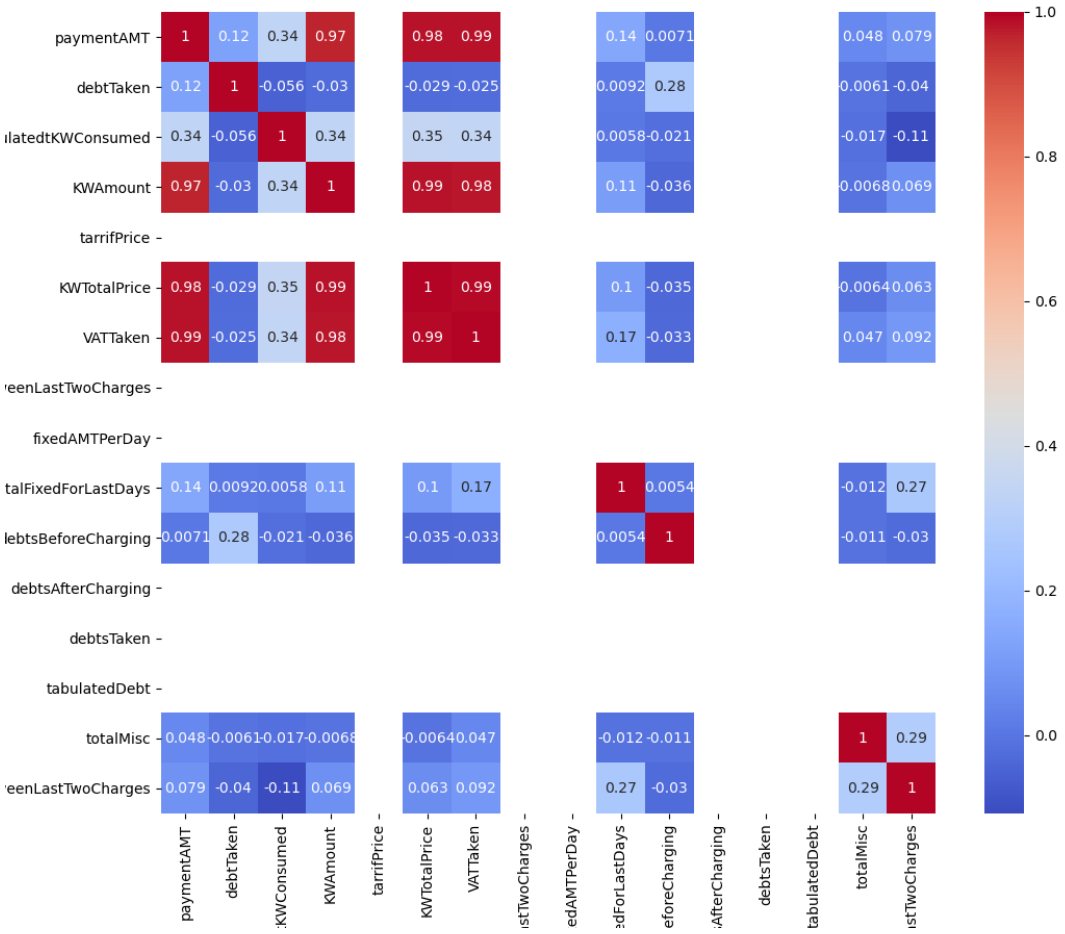

Figure 3.8: Heatmap feature correlation for the charges dataset

to impute the value. These ways could be any method which depend on the data itself, like mean value, median value, regression value or others

3. Nature of the data: If the data is sequential or time-series data, dropping data might create inconsistency in the sequence

4. Random data: If the missing data values are for random data, so dropping it certainly won't affect the model

Based on the previous points, and as shown in Table 3.8, we can process our missing values as follows:

1. Irrelevant data: Datasets may contain irrelevant data columns, such as sequence numbers or indexes representing information from other systems, or random numbers like "Payment Number". Features like "Payment NO", "Voucher NO", "Government Payment NO", and "Misc Payment NO" are sequence numbers that do not describe the target feature and are not related to it; therefore, they can be safely dropped without affecting the dataset's consistency.

2. "Meter Reading Date", "KW Total Price", "VAT", "Total Fixed for Last Days", "Debts Before Charging", and "Total Misc" features: These features contain data that cannot be imputed by mean, median, or other methods. Furthermore, they have a small number of missing values, which do not exceed 0.001% of the dataset. Thus, dropping the rows with null values for these features will not negatively impact our model.

3. "Tariff Price" feature: This column is entirely null and could be imputed by another column. However, since we have another column, "tariff id", which can reflect the same meaning or have the same effect on the data, we choose to drop the "Tariff Price" feature.

4. "Date Difference Between Last Two Charges" feature: This feature is entirely null, but it can be imputed from other features, as we have the "previous charging date" and "payment date" for each charge record. Therefore, we can impute its value by subtracting "Previous Charging Date" from "Payment Date", which will give number representing the days' difference between those two features.

Column Name	Number of Null Values	Percent of Null Values
Meter Reading	4,298	0.004%
Meter Reading Date	152	~0.0%
Tariff Price	891,134	100%
KW Total Price	120	~0.0%
VAT	120	~0.0%
Fixed Amount Per Day	891,134	100%
Total Fixed For Last Days	120	~0.0%
Debts Before Charging	120	~0.0%
Debts After Charging	891,134	100%
Debts Taken	891,134	100%
Payment Description	891,134	100%
Payment NO	1,597	0.001%
Voucher NO	1,343	0.001%
Government Payment NO	891,134	100%
Misc Payment NO	891,134	100%
Tabulated Debt	891,134	100%
Social Campaign	891,134	100%
Total Misc	120	~0.0%
Date Difference Between Last Two Charges	891,134	100%

Table 3.8: Null value analysis of the dataset columns

Feature Insights and Engineering Opportunities

In this section, we analyze the results from Section 3.2.4 on Data Visualization and Statistics, exploring the potential for creating new features based on insights gained from visualization and statistical analysis. Although these features have not yet been created, our goal is to discuss the opportunities for their development and how they might be used in subsequent steps.

Masoud Nikravesh et al. [57] published a book about feature selection, describing techniques for feature selection and highlighting its crucial role in enhancing the generalization capabilities of learning machines. They emphasize the importance of data representation, which we addressed in Section 3.2.4. This led us to visualize the relationship and correlation

between various features. This step can involve selecting a fixed number of features, whether binary, categorical, or continuous.

Facility Type Analysis

By analyzing the results of the previous section, and as shown in Figure 3.4, we notice a relationship between the facility type and the number of days to recharge. This observation leads us to consider creating one of the following:

1. Continuous Feature: In the NCDP, a continuous feature is the average number of days to recharge based on the facility type. The advantage of using a continuous feature is that it aids in collecting more accurate data. This helps our model learn from this feature and improves prediction performance.

2. Categorical Feature: A categorical feature can be either binary or a classifier for the facility that categorizes how often a facility type recharges. Creating a categorical feature can sometimes simplify the model and make it easier to interpret. It can also help the model avoid overfitting during the training process.

Several methods exist for selecting the best features to apply. One such method is using Analysis of Variance (ANOVA), a statistical method used to test differences between two or more means. We applied ANOVA to the features mentioned above, resulting in an F-value of 614.80. This value suggests that "facilityType" significantly affects our target feature. Additionally, considering that machine learning is an empirical process, and given that we do not have a large number of new features to test, we have decided to create both features. Subsequently, we will conduct tests to measure the model's performance after applying both features.

Facility Groups Analysis

Using the same methodology as before, our dataset includes various groups corresponding to each facility, broadly categorized into three major groups: "Residential", "Temporary", and "Commercial". To investigate the relationship between the facility group and our target feature, we graphically represent this potential correlation by examining the variance between the facility group and the mean of our target feature. As observed in Figure 3.5, there is a noticeable discrepancy in the data for each group of facilities. This difference may lead us to create new features to capture this distinction more accurately.

Geographical Analysis

The geographical region or residential area impacts the nature of charging operations, as hotter areas typically see an increase in charging rates during summer days, unlike colder areas. Therefore, we have conducted an investigation by graphing the residential area against the average number of days for recharge. The results, as shown in Figure 3.6, suggest a relationship between the geographical area and the frequency of recharging. This relationship prompts the creation of new features that may help improve the performance of the proposed model.

Season Analysis

As demonstrated in Figure 3.7, there appears to be a variation in the number of days between charges and the frequency of recharging across different months. This variation underscores the potential for the creation of new features that could enhance the performance of our proposed model. Therefore, we create a categorical feature that categorizes the month into high, low, and mid. This categorization represents the frequency with which customers recharge in a given month. We have created a categorical feature because the mean value between months is not significant.

3.3 Model Implementation and Selection

In this section, we explain the process of implementing our machine learning models to create the NCDP for customers based on their previous charging behavior and other relevant features. The choice of a machine learning model significantly affects the prediction outcome. The selection of the model depends on several factors, which are pertinent to our thesis and are mentioned as follows:

1. Nature of the problem: usually for machine learning problems, the nature of the problem is either classification, regression or clustering. For each problem, there are several machine learning models and algorithms that can handle the problem. Each algorithm differs in its approach, complexity, and the type of data it handles best. Some algorithms, like neural networks, are highly flexible and can be used for various types of problems but require large amounts of data and computational power. Others, like decision trees, are simpler and more interpretable but might not capture complex patterns as effectively.

In the NCDP, the problem is identified as a regression problem that can also be approached as a time series problem, since it involves predicting future events based on past data, and the sequence of the data is significant.

2. Data characteristics: the type, quantity and quality of the data also influence the model selection process. Deep learning models are considered as data-hungry models, so they need a large amount of data for better learning and better performance and consequently generalization [58].

As a result of the considerations outlined above, and given the variety of models that can be applied, we face the challenge of finding the model that offers the optimal balance between performance and complexity. At the beginning of this thesis, through the previous sections, we have clearly defined our problem and gained an understanding of the data we have, including the relationships between its features. In the current step, we are creating and selecting models capable of addressing the problem. Initially, we are developing a basic models using simple algorithms like linear regression, XGB and others, as this provides us with a benchmark to measure the performance of more complex models. Subsequently, we will experiment with a more complex model, such as a RNN, and compare the performance between the basic and complex models.

3.3.1 Selection of Tools and Environment

Programming Language and Libraries:

- Python 3.9 was chosen for its extensive support in data science and machine learning, offering robust libraries and a vast community for troubleshooting and support.
- The utilization of libraries like NumPy and Pandas facilitated efficient data manipulation and analysis.
- For model building and neural network implementation, Keras and TensorFlow were selected. Keras, running on top of TensorFlow, offered a user-friendly interface for constructing and training neural networks.
- The LSTM (Long Short-Term Memory) layer from TensorFlow's Keras was a key component in the RNN architecture, chosen for its effectiveness in handling sequential

data and its ability to overcome the vanishing gradient problem common in standard RNNs.

- Regularizers, EarlyStopping, and Adam optimizer from Keras played important roles in enhancing model performance and preventing overfitting. Regularizers helped to impose penalties on layer parameters or layer activity, EarlyStopping was used to halt the training process at the right time, and the Adam optimizer was utilized for its adaptive learning rate capabilities.

- TimeseriesGenerator from TensorFlow to create sequences for RNN, which facilitates the generation of data for training and testing time series models.

Hardware Specifications:

The computational experiments were conducted on a desktop PC equipped with the following hardware components:

- Processor (CPU): Intel(R) Core(TM) i7-4700MQ CPU @ 2.40GHz, 4 cores.
- Memory (RAM): 16 GB DDR3 at 1600 MHz.
- Storage: 256 GB SSD
- Graphics Card (GPU): NVIDIA Quadro K3100M, Intel(R) HD Graphics 4600.
- Operating System: Windows 10 Pro, Build 19045

3.3.2 Non Machine Learning Approach

For predicting the next charge date, several non-machine learning methods can be employed, with one of the simplest being the calculation of the average number of days between charges. This method, while not accounting for individual customer patterns or historical trends, offers a straightforward approach to the task. Utilizing this method, we calculated the average number of days between two consecutive charges across the entire dataset and individually per customer, applying it to the test set we had previously segmented. The results showed an average of 8.10 days for the dataset as a whole and 5.07 days for individual customer averages. Despite these averages appearing somewhat high, they will serve as a foundational baseline for further analysis.

3.3.3 Classical Machine Learning Algorithms

In this section we will build our classical machine learning algorithms models like linear regression, polynomial regression, XGB, forest tree and decision tree algorithms. Classical machine learning algorithms have some advantages compared with deep learning algorithms. Usually, they have very fast training time, needless data size for training, and their algorithms are more simplicity and interpretability. So, in this step, we will first build the model using only the charging dataset file, then we will add features related to service area etc. from the other datasets files we have. Also, we will extract and add new features to the models based on statistics founded in the previous sections, then we will compare the results for each step.

Establishing a Baseline Model

In this experiment we will train our model based on all charges in raw charges dataset file only, without adding or extraction any new feature from service dataset file.

Goal of this experiment: The goal of this experiment is to be considered as a baseline for further experiments, which can give us insights about further steps

Algorithm used	MAE value	Time elapsed to train in seconds
Linear	7.22	< 1
Polynomial	6.93	2.89
XGB	6.46	3.05
Decision Trees	6.59	2.58
Random Forest	6.45	211.91
KNN	6.89	3.615

Table 3.9: Classical ML algorithms results on raw dataset

Table 3.9 shows the performance of classical machine learning algorithms that was built using only charging dataset file, without extracting new features, or adding further features like area, tariff... etc. The models were built using all records exists in the dataset with cleaning rows with null values as mentioned in section 3.2.4. We noticed that the algorithms have very fast training time, with an average performance of about 6.7 on MAE metrics.

Analyzing the Impact of Record Counts on Model

In this experiment, we will train our model on different records count. First we will split our data into train and test samples. Then we will train our models using different record

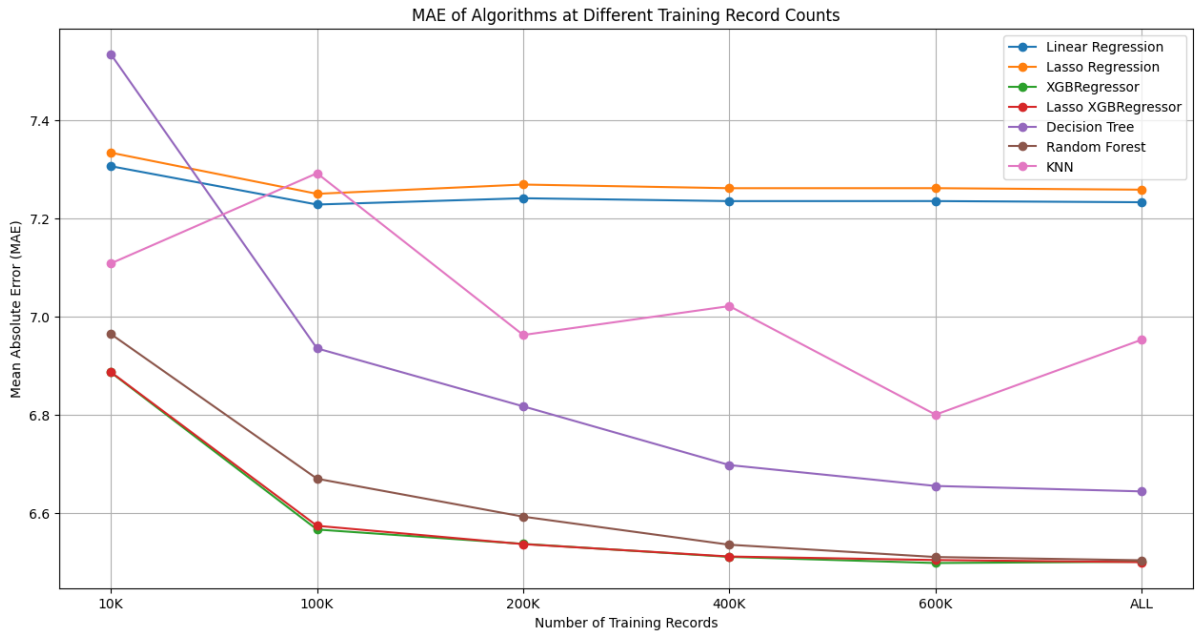

Figure 3.9: MAE for classical algorithms on raw charges dataset and different records count

counts from the train set and test them on the separated test set.

Goal of this experiment: The goal of this experiment is to examine how well each algorithm can handle dataset size. This can help us determine the minimum amount of data needed to reach an acceptable accuracy, while saving computational time.

Figure 3.9 shows the result as a line graph for classical machine learning performance among different records count when building those algorithms on raw dataset. Each step we increased the records count for the training set to check the performance of each algorithm and how its performance gets affected with the dataset size. Linear regression showed relatively higher MAE values across all record counts, indicating less precise predictions, while increasing the record count did not improve the performance of the model, which might suggest a limitation in capturing the underlying pattern. In contrast, as the record count increases, certain models such as the XGB, decision tree and random forest exhibit a relatively stable or slightly improving MAE. KNN, initially exhibited a decrease in MAE, indicating an improvement in performance. However, as the data size increased, the performance began to fluctuate, and a consistent decrease in MAE was not observed. This test gives us insights that XGB and random forest may handle larger dataset in a better way among other algorithms. Overall, the trend of performance for all algorithms are close to each other, and it needs further data processing and feature engineering

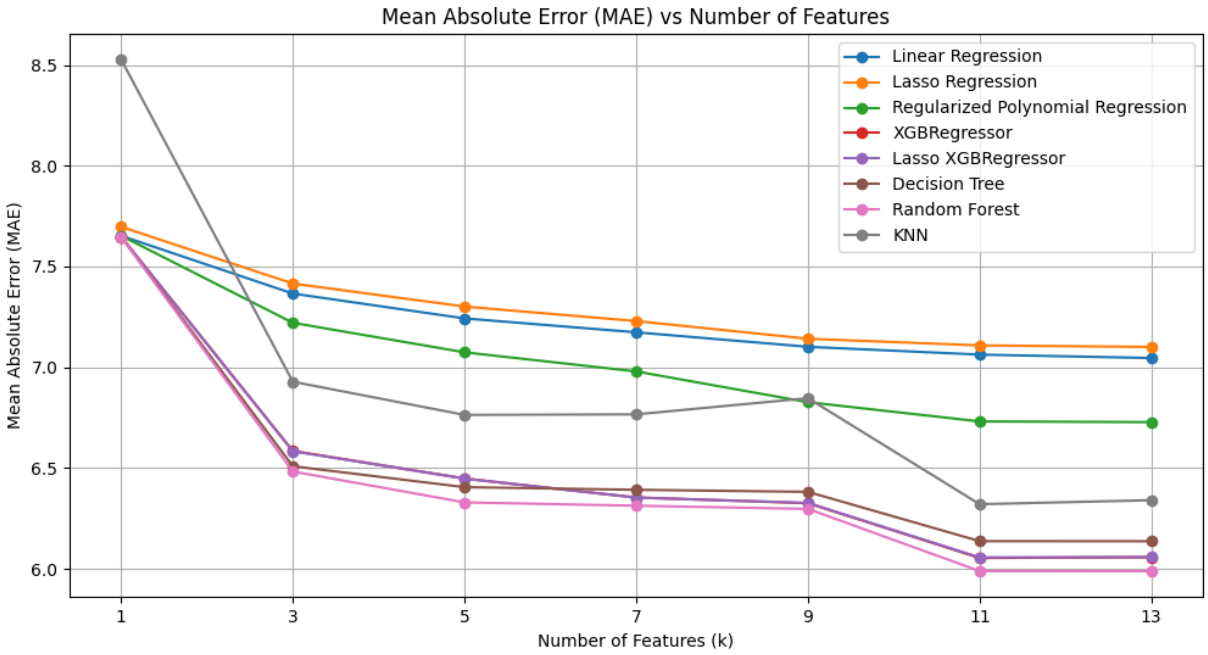

Figure 3.10: MAE for different features selected by SelectKBest

Enhancing Model Accuracy with Service-Related Feature Integration

In this experiment, we will merge data from additional files that contain customer service-related information, such as area, meter amp phase class, among others. These features represent informative features about the meter and the customer service information. The features are facilityType, subBranch, phasesAMPClass, areaID, facilityGroupID. Also, we will build our classical models by selecting different number of features. We will select these features by using selectKBest method from sklearn [51]. Our approach involves executing a loop that iterates from 1 to the total number of available features. In each iteration, we utilize the SelectKBest method to identify and select the most impactful features with the highest relevance to our target variable. For each iteration, we will monitor the MAE value to determine the optimal set of features for building our model

Goal of this experiment: The goal of this test to see how each model will perform when adding more informative features like merging service related data

From Figure 3.10, we can notice the following observations:

- Impact of Feature Selection: As the number of informative features increases, there is a general trend of decreasing MAE for most algorithms, indicating that incorporating more informative features may "but not necessary" improves model accuracy.

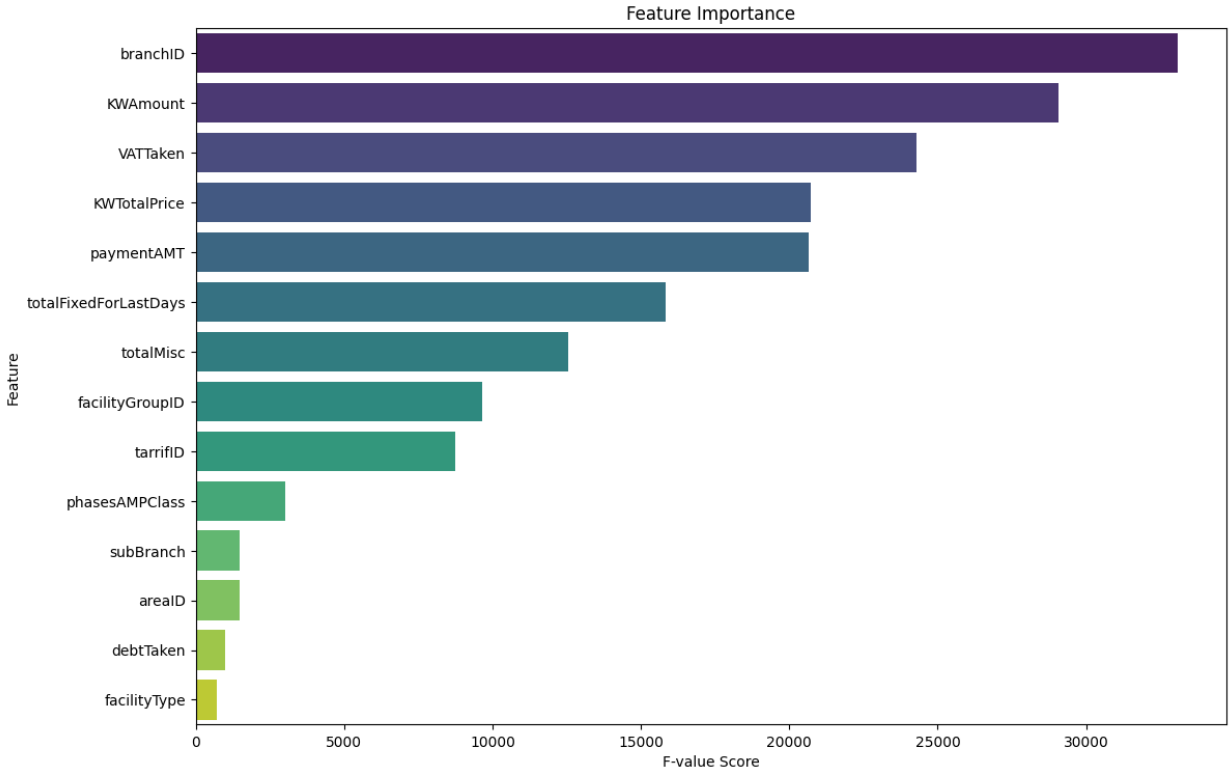

Figure 3.11: Feature importance following the integration of service-related features

Improving the performance or decreasing it depends on the added features and the used algorithm.

- Algorithm Sensitivity to Feature Selection: Methods such as Random Forest and XGBoost show a particularly strong response to increasing feature consistently reducing the MAE as more features are taken into account. This reflects its power and ability to leverage a wide range of features effectively. On the other hand, linear models, including linear regression and lasso regression, show consistent but more moderate improvements. It is worth noting that the performance of the K-Nearest Neighbors (KNN) algorithm initially fluctuates but stabilizes and improves later, indicating a threshold effect where a certain number of features is necessary for the algorithm to efficiently capture underlying patterns. Similarity in performance trends between XGBoost and Lasso-XGBoost across the board the k values are particularly interesting, indicating that Lasso regularization did not significantly affect feature selection in XGBoost for this dataset.

Advancing Model Predictions through Feature Extraction and Engineering

In this experiment, we will apply feature extraction and engineering to extract new features depending on our observations as mentioned in section 3.2.4

1. Feature Extraction: We introduced four new features — "dayOfPre", "monthOfPre", "month", and "dayOfWeek" — derived from the "previousChargingDate" and "paymentDate" fields, as these are date fields that cannot be directly utilized in the model. These features represent the day and month of the previous charge, the month of the current payment, and the day of the week, respectively. Additionally, we attempted to encode these date fields into Unix timestamps, transforming them into a numerical format that represents the number of seconds since January 1, 1970. However, this encoding approach led to a decrease in the model's performance, indicating that the direct conversion of date fields into Unix timestamps might not capture the temporal relationships and patterns effectively for our specific modeling task.

2. Following insights from section 3.2.4, we developed additional features including:

   (a) previousChargeColdnessRate, coldnessRate: Based on Figure 3.7 that illustrates the mean average for customers recharging over months. We divided the months into three categorical months based on the plot.

(b) customerAVG: this feature represents the average number of days it takes for each customer to recharge. It's calculated as a mean value of the total days between recharges for each customer, offering a personalized measure of recharge frequency.

(c) branch_area_avg_category: this feature is a categorical representation based on the average recharge frequency, grouped by customer location, which includes branch, sub-branch, and area levels. The creation of this feature involved several steps:

(d) First, we computed two statistical measures for each geographical area defined by combinations of branchID, subBranchID, and areaID: the trimmed mean and the interquartile range (IQR) of 'customerAVG'. The IQR was chosen for its robustness in calculating an average that mitigates the influence of outliers. ii. We then categorized each area into predefined bins based on its trimmed mean recharge frequency, creating a categorical feature that reflects the regional variation in recharge habits

(e) RechargeIntervalFactor (RIF): this feature is designed to capture variations in recharge behavior by considering both the amount recharged and the time interval between recharges. The feature is calculated by taking the total quantity of purchased electricity (in kilowatt-hours, kWh), dividing it by the quantity of electricity for the previous charge, and then multiplying by the number of days since the last charge (daysSinceLastCharge) as per the following equation:

$$RIF = \left(\frac{KW_i}{KW_{i-1}}\right) \times \text{daysSinceLastCharge} \tag{3.1}$$

After performing feature selection and engineering, we build the models again and the results were as shown in Table3.10:

Algorithm	**Raw Data**	**After Merging Service Info**	**After Merging and Engineering**
Linear	7.22	6.99	4.64
Polynomial	7.00	6.65	4.20
XGB	6.49	5.93	3.65
Decision Trees	6.63	5.98	4.88
Random Forest	6.50	5.81	3.74
KNN	7.26	6.12	4.12

Table 3.10: MAE results for classical ML on various features sets

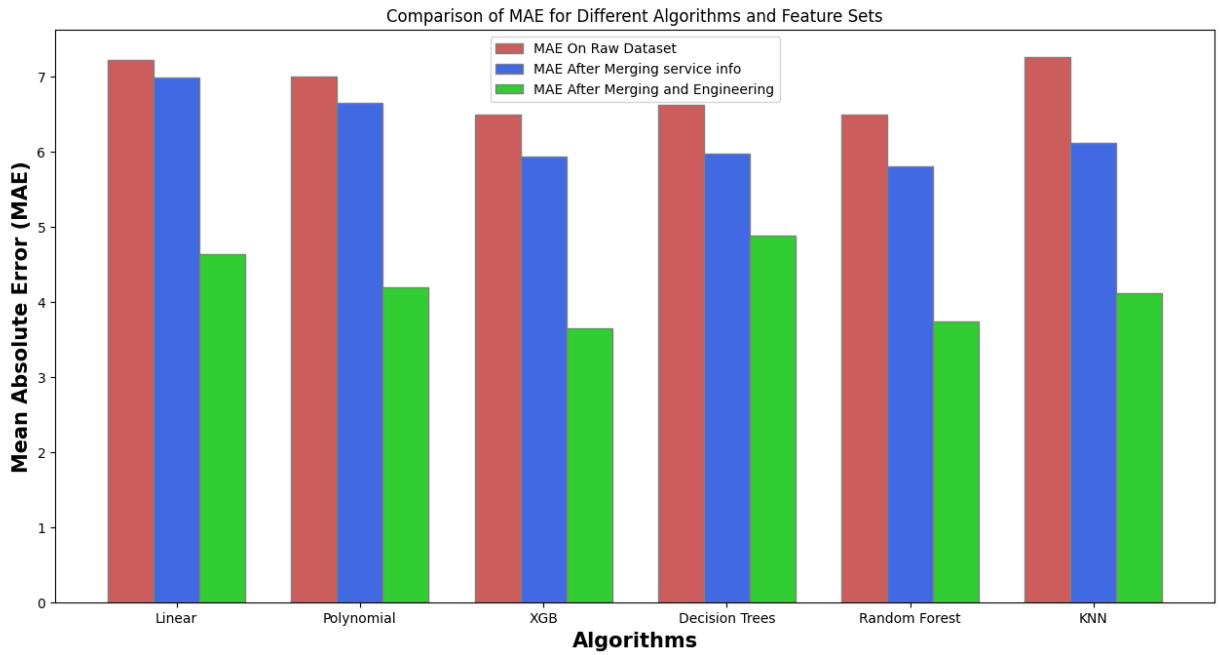

Figure 3.12: Classical ML MAE results on various features set

Figure 3.12 visually compares the Mean Absolute Error (MAE) of various machine learning algorithms across three different stages of data processing: using the raw dataset, after merging service information, and after performing merging and feature engineering as described previously. The results confirm that across all algorithms, there is a noticeable reduction in MAE as we move from using the raw dataset to applying additional data processing steps. This trend highlights the significant impact of feature engineering on model performance. Overall, each algorithm shows improvement in MAE after merging service related information like service location, meter amp phase class and other, and also further improvement after additional feature engineering features. This suggests that the added data and engineered features are providing valuable information that helps the models make more accurate predictions

Model Performance through Strategic Training Data Selection

The dataset we have is sorted by the customer number and payment date. The customer number in the dataset represents the location (geographical region) for the service. So, splitting the data without shuffling may affect the performance of the model. Hence, we run our classical algorithms for the same dataset but once in a sorted dataset, and the other is

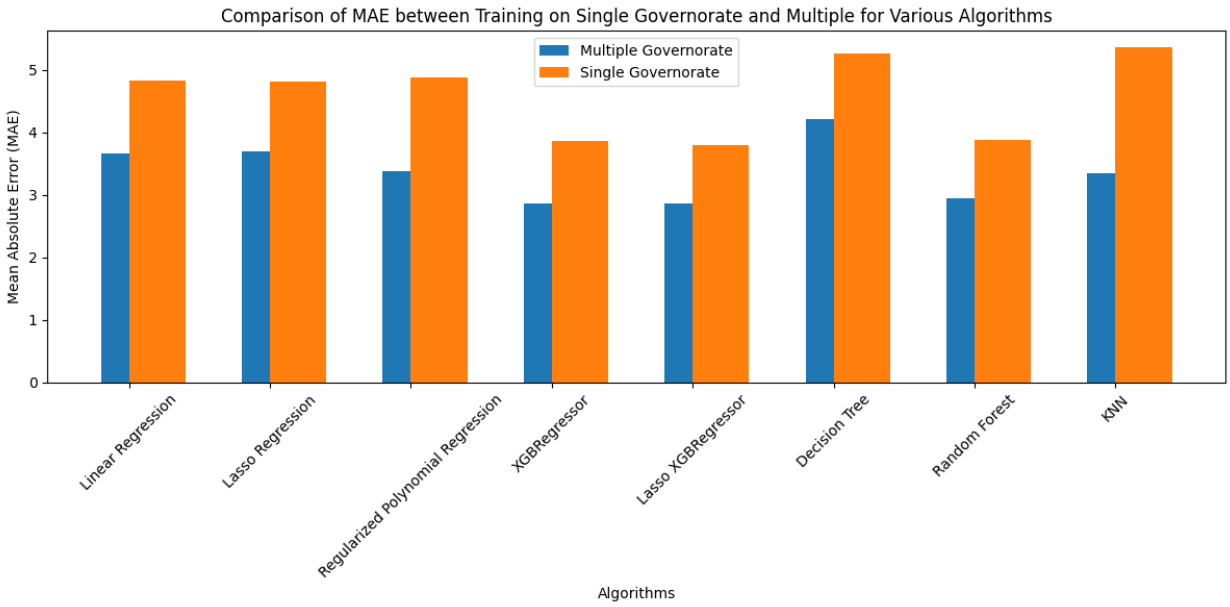

Figure 3.13: Classical machine learning MAE comparing selecting single governorate in training VS all governorates

by shuffling the data.

Goal of this experiment: the goal of this experiment is to examine the importance diversity and representation in training predictive models, and to examine if taking random data from same governorate is enough or should consider taking data from all governorates

As Figure 3.13 shows, all algorithms perform better when including data from multiple governorates rather than on the governorate-specific training set. This suggests that the governorate-specific model may not generalize well across different governorates, indicating significant regional variations in electricity charging patterns that are not captured when the model is trained on data from a single governorate. It also suggests the importance of considering geographical (or regional) features as part of the model. It implies that models can benefit from understanding regional differences, which could be encoded explicitly as features or captured through more complex, regionally stratified model training approaches

3.3.4 Deep Learning: Emphasizing Recurrent Neural Networks

In this section, we will develop our model using RNNs. RNNs are a category of deep learning algorithms that offer certain advantages over classical machine learning algorithms, especially for processing sequential data. This makes them particularly well-suited for time-

series applications. Unlike classical models, RNNs possess a form of memory that enables them to retain information from previous inputs. This feature is beneficial for making predictions based on historical data, which is relevant to our problem and dataset.

Our approach integrates methodologies from classical machine learning models, adapted for use with recurrent neural networks (RNNs). Initially, we will develop the model using only the charging dataset. We plan to subsequently enhance the model by incorporating additional features related to the service area and other relevant information from supplementary files, as described in the section 3.3.3. These features, similar to those used in classical machine learning, will be incorporated into RNN models. After incorporating these features, we will perform a comparative analysis to assess the model's performance at various stages. This iterative process enables us to evaluate the influence of additional data on the model's predictive accuracy, thereby allowing us to refine our strategy for optimal results.

Furthermore, the performance of RNN models is influenced by various hyperparameters, such as the selection of window size (time step), the loss function used in the training process, model architecture, among others. Therefore, we will conduct experiments to search for the optimal values for these hyperparameters to achieve the best performance. This involves systematically adjusting and evaluating the impact of different hyperparameters on the model's accuracy and efficiency, allowing us to fine-tune the RNN to our specific requirements and data characteristics.

Evaluating Recurrent Layer Architectures

Goal of this experiment: The goal for this experiment is to find the layer type for our predictive model.

In this round, we will build the RNN model architecture that will be used to handle our problem. RNN has multiple layer types that differ in their implementation and purpose. There are three types (SimpleRNN, LSTM, GRU) that can handle regression problems.

We constructed the model employing various types of recurrent neural networks and discovered that the SimpleRNN was marginally quicker to train, taking approximately 3 hours and 50 minutes, in contrast to LSTM and GRU models. However, it was less accurate. Meanwhile, the performance outcomes of LSTM and GRU models were nearly identical, with both requiring about 4 hours and 8 minutes to train. This indicates that while SimpleRNN offers a time advantage, LSTM and GRU provide a better balance between training duration and accuracy, making them potentially more suitable for applications where model precision is critical. And, since LSTM can capture more complex patterns, and the payment of the

customer can be affected with long-term trends, season, weather, or customer social life, we will focus on using LSTM as our recurrent layer type.

Hyperparameter Optimization

RNN has multiple hyper parameters that are passed when building the models. Each layer can have different hyper parameters with different values. These hyper parameters affect the performance of the model dramatically since it affects how the model runs and learn from the training data. Finding hyper parameters is a crucial process that needs a lot of experiments, so there are multiple ways to accomplish it. First method is by manually trying different hyper parameters for each layer. This method is a time consuming and subject to human error. The second method involves using grid search, by defining a list of values for each hyper parameter, then iterating over every single value to log the model's performance results for each configuration built with those values. The third method, in principle, is similar to the second method, but it simplifies the process by avoiding the need for detailed coding and configurations. This is achieved through the use of a library named "RandomSearch," which provides a mechanism similar to the grid search approach but selects hyperparameters at random to evaluate the model's performance. This is done by define the range of values for each parameter, and the library will build and log the results for each run, and gives us the best hyper parameters achieved from the results. In our implementation, we experimented with all these methods and concluded that "RandomSearch" is the most effective approach. It offers time efficiency and a lower likelihood of errors. We utilized RandomSearch to explore various parameters, including the number of units in each layer, the ratios of L1 and L2 regularization, the dropout ratio for each dropout layer, and the learning rate. This method allowed us to efficiently identify optimal configurations without the exhaustive process required by grid search. So, after utilizing the Random Search approach to determine the RNN model architecture, the following RNN model architecture was obtained as best suited our problem:

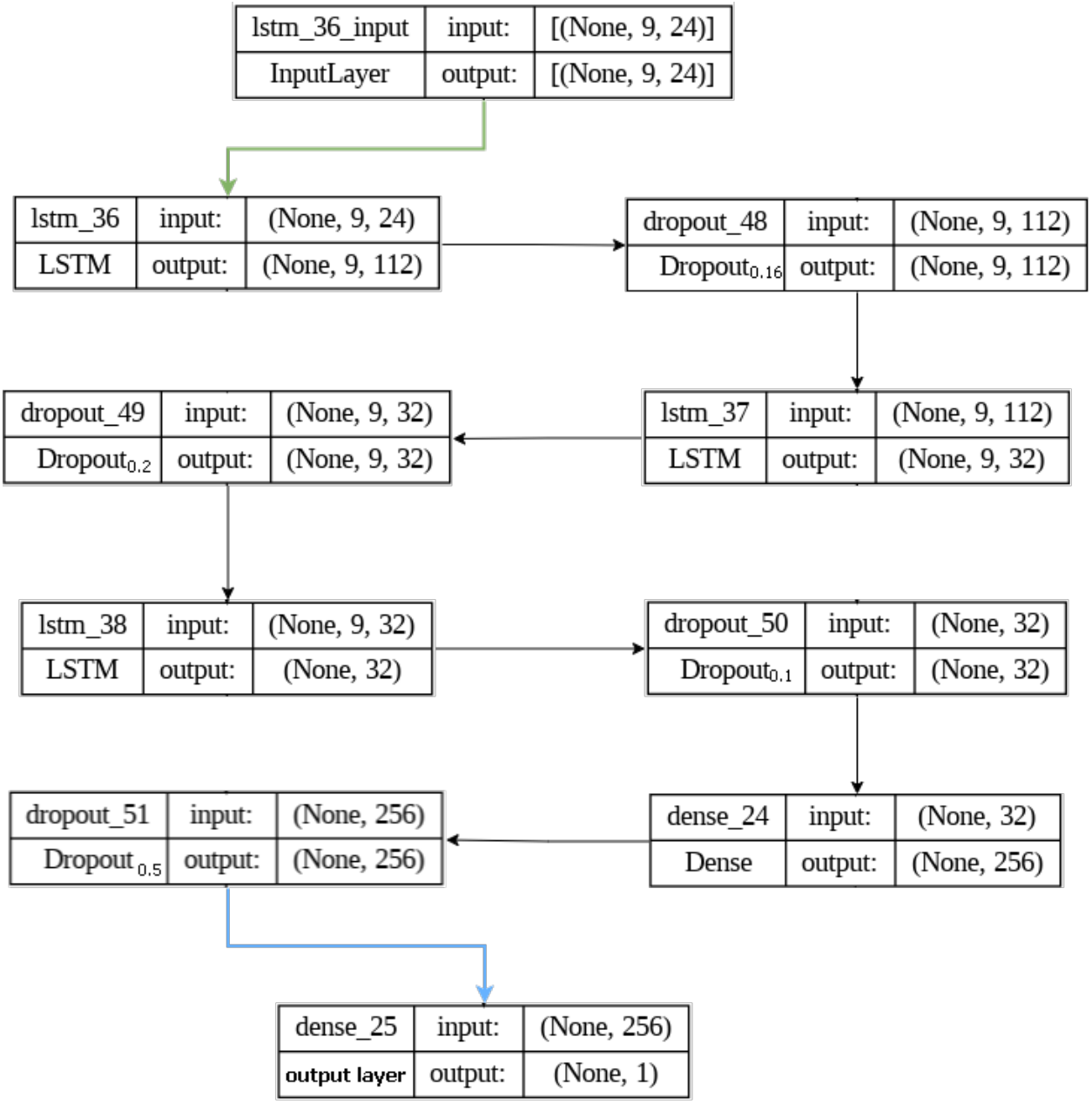

Figure 3.14: RNN architecture after hyperparameters optimization

The model consists of the following layers and components:

- Sequential Model: The model is a sequential model, which means that the layers are stacked linearly.
- Input Layers:
  - The Masking layer here acts as the input layer. The masking layer purpose is to handle zero padding records for records who doesn't have the required window size
  - units=112: This LSTM layer has 112 units. Each unit is a cell in the LSTM layer.
  - input_shape=(9, 24): This specifies the shape of the input data. Since it's an LSTM layer, the input shape would be (time steps, features). This input shape could be changed for better performance. Samples of data are listed in Table 3.4.
- Dropout Layer:
  - Dropout (0.16): This layer randomly sets a fraction (16% here) of the input units to zero at each update during training, which helps prevent over fitting.
- Additional LSTM Layers:
  - Two more LSTM layers with 32 units each.
  - Each LSTM layer is followed by a Dropout layer (20% and 10% dropout rates, respectively) for regularizing the model.
- Dense Layers for Processing:
  - Dense(units=256): This is a fully connected layer with 256 units. It is used for further processing and pattern recognition after the sequence data has been processed by the LSTM layers.
- Output Layer:
  - Dense(units=1): The final layer is a Dense layer with a single unit. This configuration because our model is designed for regression

RNN configuration:

In addition to the RNN structure, the following configuration were obtained by performing Random Search for hyperparameters:

- kernel_regularizer=regularizers.l1(l1=0.0001): This adds L1 regularization to the first LSTM layer, which can help prevent overfitting by adding a penalty to the loss function based on the magnitude of the weights
- Adam optimizer with a learning rate of 0.0001.
- The loss function is Mean Squared Error (MAE). We used both MSE and MAE for most tests
- EarlyStopping(monitor='val_loss', patience=5): It will stop the training process if the validation loss does not improve for 5 consecutive epochs, helping to prevent overfitting and may also save training time.

Data input:
Initially, the dataset is divided into two distinct subsets: 80% of the data is allocated for training purposes, with 20% of this training set further reserved for validation purposes, and the remaining 20% is set aside for testing. Following this, the input features (X) are standardized using StandardScaler. Standardization is an important preprocessing step, especially for RNNs, as it transforms the features to have a mean of zero and a unit variance. After that, RNN models requires data to be in a shape of (time steps, features) as shown in Table 3.4. Time steps is a variable represents how much records the model will look back to predict the future. This variable could be tuned in building the model to enhance the performance and may affect the performance directly. Features represent how many features that will be fed to the model to be trained. In the next rounds, we will experiment different features and time steps to select those how give best performance.

Impact of Record Volume on Model Efficacy

In this experiment, we will try to train our model based on different number of dataset sizes. We split our dataset into train and test set by loading the whole dataset, and then train our model by slicing the train set based on the record count.

Goal of this experiment: The goal of this experiment is to examine the importance of having more training data for the model. Determining the minimum amount of data required for the model to reach an acceptable level of accuracy. It is particularly useful when data collection (or training) is expensive or time-consuming.

Figure 3.15 shows that the model is learning to predict more accurate predictions when having more training samples. This means that more data generally improves the perfor-

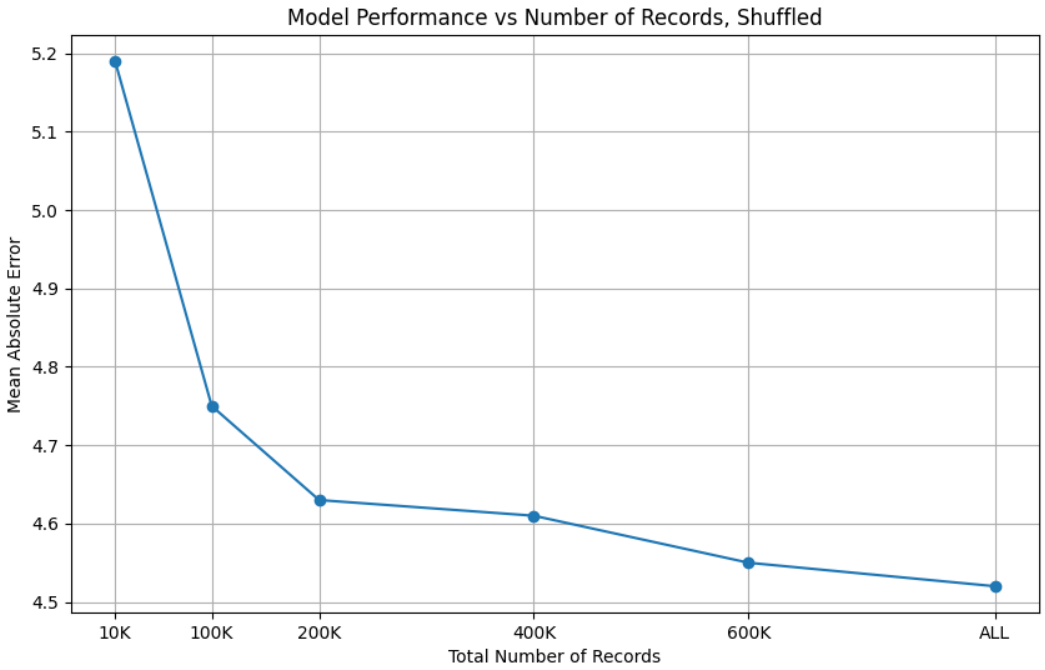

Figure 3.15: MAE performance of RNN For shuffled data over different record count on raw features

mance, since our model seems to benefit from larger datasets, as indicated by the decreasing MAE values. However, the rate of improvement decreases as the dataset size increases. The results also show that we reached "Point of Diminishing Returns", which means that the model improves slightly smaller enhancements by adding more records. This could give us that having 200K or 400K of records could be enough for training the data if we are having a limitation in computational power or struggling with a long time for the training process. Despite that, the model continued to improve even with slight improvement.

Feature Selection and Engineering for Model Improvement

In this experiment, we will build our model by using only one feature as a baseline. After that, we will add the features related to customer service, and see how they affect the performance of the model. Then, we will also add engineered features and see their effects in the overall performance of the model. Each set of features will be examined alone and together. Finally, we will apply PCA technique to try to enhance the results.

Goal of this experiment: The goal of this experiment is to try to figure out the suitable sets of features and how the model gets affected by adding each set of features.

	Raw Dataset	**After Merging Service Info**	**After Merging and Engineering**
RNN LSTM	4.52	4.67	3.09

Table 3.11: MAE performance for RNN LSTM after merging service data and feature engineering

Table 3.11 shows how the model performance performed after merging service-related information. The results from our model demonstrate the significant impact of data preprocessing and feature engineering on model performance. Initially, when the model was trained on the raw dataset, the MAE was recorded at 4.52. This initial performance serves as a baseline for understanding the model's capability with raw charging data. After merging the customer service info features, there was a slight increase in the error, rising to 4.67. This decrease of performance shows that RNN model doesn't improve for service information as classical machine learning algorithms do. This could be related to the nature of data, because RNN are particularly well-suited for sequential data where the order and context matter significantly, such as time series. Meanwhile, the table also shows a noticeable enhancement when adding engineered features since most of the new features represent sequential data.

We did further investigation to see the effect of features by dividing the features into

informative groups and make test for each set of features individually and together as the following:

- Single Feature (baseline): From the dataset, we extracted a feature named "daysSinceLastCharge," which represents the number of days between the current charge date and the previous charge date. This feature demonstrates a sequential pattern in customer behavior, as it indicates the number of days between each charge and the preceding one. Given that our problem is sequential problem, we conducted experiments focusing solely on the "daysSinceLastCharge" feature to assess the model's performance based on this single input. This approach serves as a baseline for our feature experimentation. We trained our model using the "daysSinceLastCharge" feature as input and our designated target feature as output, utilizing all available records in the dataset. The MAE for this test was 3.9, providing an initial indication of the predictive accuracy of the model.

- Sequential features: This test is to check the influence of sequential features in the RNN. These features represent the number of days between previous and current charge, the day and the month for both previous charge and current charge. Adding temporal features related to dates ('dayOfPreviousCharge', 'monthOfPreviousCharge', etc.). decreases the MAE to 3.4. This suggests that incorporating more detailed time-based context helps the model make more accurate predictions.

- Sequential and statistical features: We added statistical features that represent the mean average for the customer to recharge, a categorical to his mean average, a categorical for the month of the current and previous charge. Incorporating statistical features (like 'coldnessRate', 'previousChargeColdnessRate', etc.) further reduces the MAE to 3.2. It indicates that these features provide information that helps the model understand the patterns and behavior of customer recharging better.

- Sequential, statistical and geo features: We added features that demonstrate the customer meter location. Adding geographical features (like 'branchID', 'subBranch', 'areaID') slightly increases the MAE to 3.1. This minor increase suggests that while the area features might provide some contextual information, they might not be as predictive or might introduce some noise compared to the previous set of features.

- Sequential, statistical, area and financial features: We added features that demonstrate the financial details of the customer charges. Finally, including financial features ('pay-

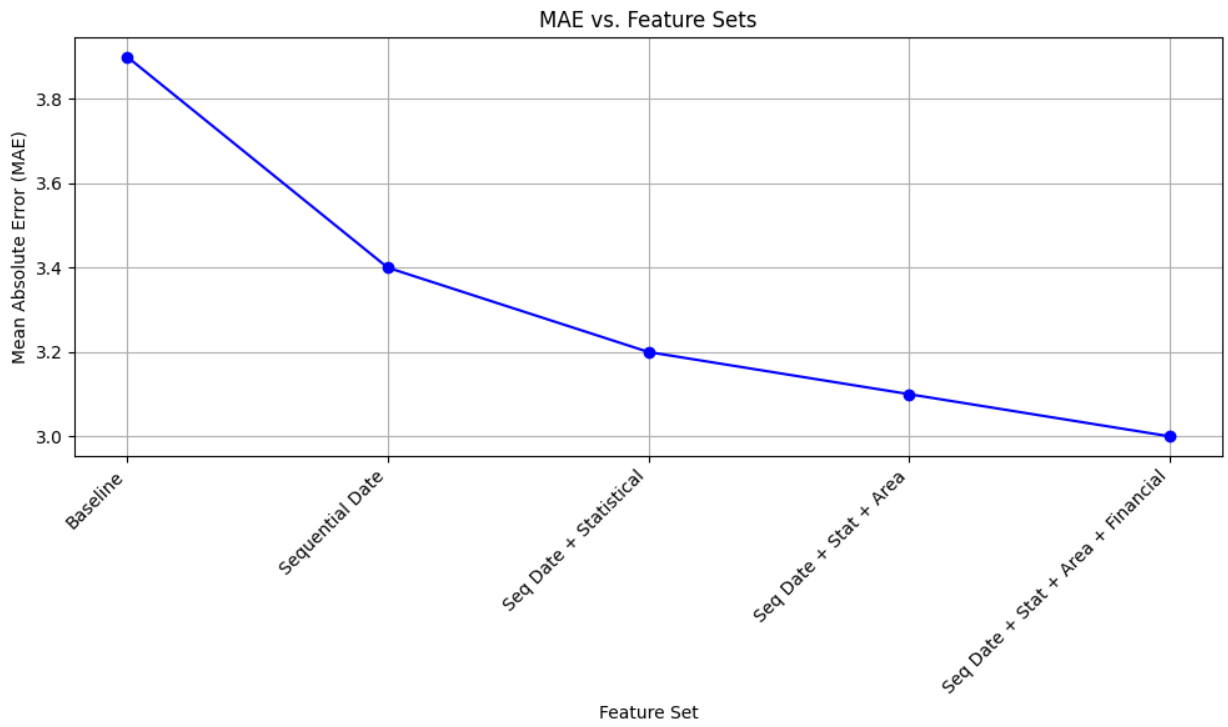

Figure 3.16: RNN performance for different features set

mentAMT', 'KWAmount', 'tarrifID') reduces the MAE to 3.0. This is the lowest MAE observed, indicating that financial features, when combined with the other types, provide more information that contributes to making the most accurate predictions.

Figure 3.16 shows the results for running these tests. It shows that RNN performance improved by incorporating more informative features, but if we compare it with classical machine learning, the performance is relatively small for some set of features, which shows that RNN is less sensitive to certain types of features.

We also implemented PCA to attempt to improve our model's performance by reducing the dimension of the input features. We employed two approaches: the first by selecting components that account for 95% of the total variance, while the second approach involved selecting a fixed number of principle components. Table 3.12 shows the results of running PCA which indicates top result for retrieving 95% of variance which is a close result to use all features from previous test.

PCA	Number of Components			
	95% of total variance	5 Components	10 Components	15 Components
MAE Value	3.15	3.44	3.27	3.18

Table 3.12: Impact of PCA components on MAE value

Loss Function Analysis

In this experiment, we will use different types of loss functions in the training process. The loss function helps the model to evaluate its performance and try to enhance its performance by adjusting its weights and units depending on the result from the loss function.

Goal of this experiment: The goal of this experiment is to examine the performance of the model by using different loss functions and to find the most suitable loss function for our model.

We trained our model using five different loss functions using 100K of the dataset for faster training. The first experiment was built using mean absolute error (MAE), which is less sensitive to outliers. The second is by using mean squared error (MSE), which emphasizes larger errors by squaring them. Third is by using root mean square error (RMSE) which is similar to MSE but used when large errors are not desired. Fourth experiment was built using Huber Loss, which is a combination of MSE and MAE. Huber is smooth for small errors and linear for large ones. Fifth is by using Log-Cosh Loss, which offers a balance between MSE and MAE.

Loss function	MAE	MSE	RMSE	Huber	Log-Cosh
MAE Value	3.21	3.28	3.30	3.19	3.20

Table 3.13: Comparison of MAE values for different loss functions

By running all the mentioned loss functions, we noticed that the model has close test results, where the MAE falls between 3.28 for MSE to 3.19 for Huber as shown in Table 3.13

Activation Functions and the Role of Non-Scaled Features

Goal of this experiment: The goal for this experiment is to evaluate the effect of different activation functions on all dense layers.

Our model consists of two dense layers. The first layer serves as the basic input for the feature transformation and learning process, using the "ReLU" activation function. Meanwhile, the output layer uses a "linear" activation function, as used in all previous tests. We defined an array with the activation functions we need to test and performed a loop through it to test various types of activation functions for both layers. Also, we examined both dense layers and activation functions while scaling target feature and keeping it in raw form. The results indicated a decrease in model performance, as shown in Table 3.14 for using linear activation in the output layer, and nonlinear for first dense layer:

Output Activation Function	**Activation Function for Dense Layer 1**		
	ReLU	**SoftPlus**	**Linear**
ReLU	5.22	5.21	5.17
SoftPlus	5.17	5.20	5.19
Linear	3.34	3.28	3.32

Table 3.14: RNN performance for different activation functions, target feature scaled

Additionally, we performed the same tests to assess the impact of retaining features in their "raw form." The first test involved using our target feature as is, while the second test involved leaving all features (X and Y) in raw form. We executed both tests using various activation functions: Linear, ReLU, and SoftPlus. The results indicated that keeping the target feature in raw format results in less sensitive to activation function type. Table 3.15 illustrates the results

Output Activation Function	**Activation Function for Dense Layer 1**		
	ReLU	**SoftPlus**	**Linear**
ReLU	3.24	3.26	3.32
SoftPlus	3.27	3.29	3.30
Linear	3.23	3.26	3.27

Table 3.15: RNN performance for different activation functions, target feature raw form (not scaled)

Optimizing Temporal Dependency Capture through Window Size Tuning

RNN LSTM shape consists of time steps and feature count. Time steps are a number that helps the model to capture temporal dependencies. It tells the model how many steps it

should look back to predict the future. The following figure illustrates how window size works. Each box in Figure 3.17, illustrates records for the same customer with window size of 3 steps (3 window size).

This parameter can change the model performance dramatically. Finding the best window size is an empirical process. So, we conducted a loop mechanism to evaluate the performance of the model for different values of window size.

First, we need to take into consideration that our problem depends on customer behavior and patterns, and the dataset consists of multiple records for the same customer. So, it is necessary to maintain the sequence integrity for each customer during creating the time steps. Also, we must sort the sequence by customer and payment date to ensure that the sequence steps go from older charges to newer charges.

Furthermore, we address the issue of insufficient records for some customers relative to the chosen window size. To handle this, we adopt a padding mechanism to pad customer's charges that have less than window size plus one. We did that through applying two key modifications:

1. Adding zeros at the start of each customer sequence (as shown in Table 3.16). This process can ensure that all customers are included in the training and testing process, despite their records counting in the dataset.

2. Adding a masking layer with a mask value of zero, which can handle the zero padding in the first step, so it lets the model handle these zeros and won't get affected by the training or testing

Original Customer Charge Data	**Adjusted Charge Data with Padding**
[159, 50, 3]	[0, 0, 0]
[159, 70, 6]	[0, 0, 0]
	[159, 50, 3]
	[159, 70, 6]

Table 3.16: Customer charges before and after padding for window size 3

From Figure 3.18, it is noticeable that increasing the window size from 1 to 35 shows a trend of decreasing error, indicating that providing more historical data points (time steps) can enhance the model's performance. Based on the results, we continued to expand the window size to see if the model's performance would continue to improve. However, as shown

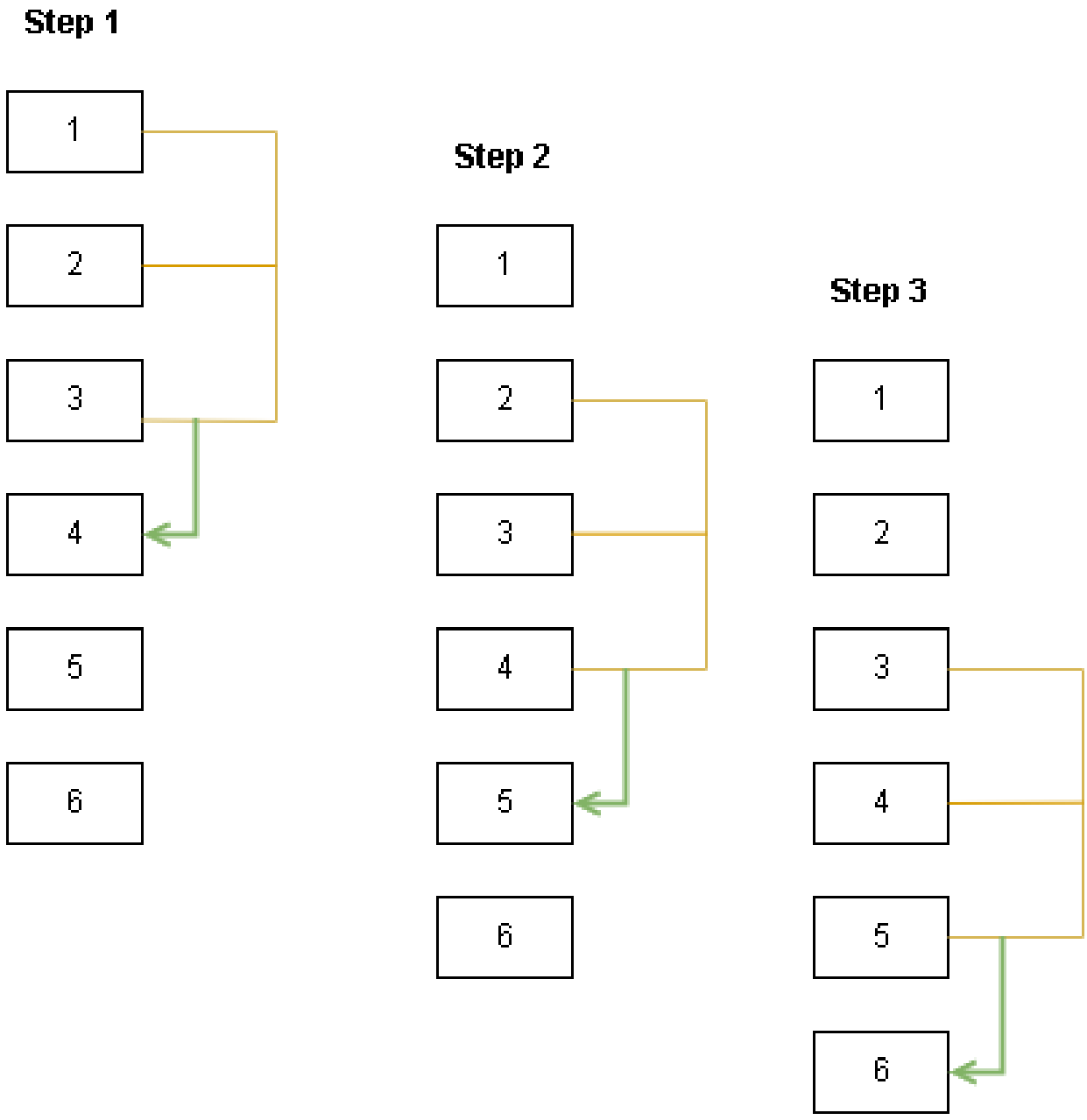

Figure 3.17: RNN window size of 3 demonstration

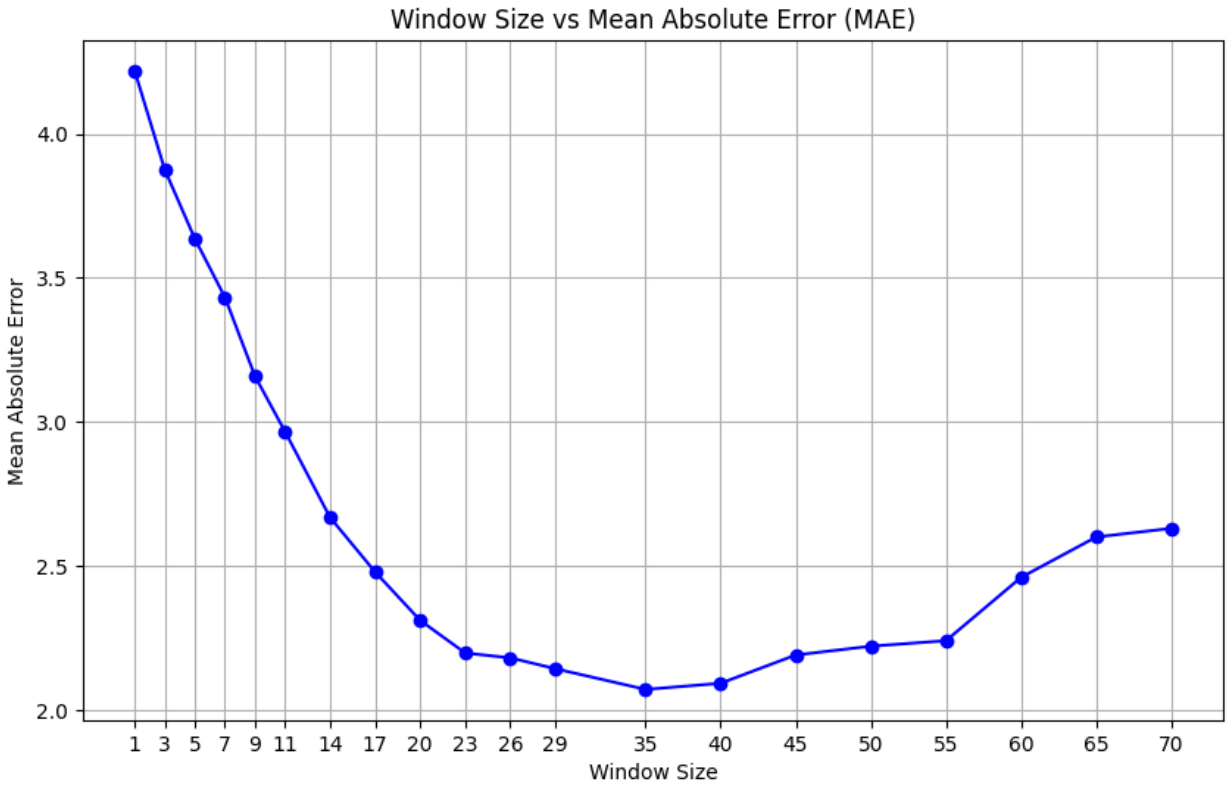

Figure 3.18: MAE RNN performance for different window size

in the figure, beyond a certain window size, the model's performance ceased to improve and began to decline. This decline could be a sign of introducing noise or feeding irrelevant information to the model due to the lengthy sequence. On the other hand, selecting large window size will take higher computational power and memory which could be an issue for certain situations.

Efficacy of Segmented Model Training Versus Generalized Approaches: A Comparative Study

In this experiment, we conduct tests on the training datasets using various data splitting strategies to explore the impact of segmentation on model performance. Initially, models will be trained using data segmented by specific criteria, such as facility type and geographical location, and then tested on same segments criteria. This approach allows for a direct comparison between models trained on homogeneous versus heterogeneous data sets. Subsequently, the entire dataset will be shuffled to eliminate any segmentation, and the model will be retrained and tested using the same evaluation criteria as before.

Goal of this experiment: The goal is to examine whether building separate models for each group of facilities or each governorate can enhance model performance, compared to training a single model on data aggregated from all segments. This comparative analysis seeks to identify the benefits, if any, of targeted model training on segmented datasets in improving the accuracy and effectiveness of predictive outcomes.

Training Segment	Testing Segment	Mean Absolute Error
Residential	Residential	2.51
Commercial	Commercial	1.46
Temporary	Temporary	1.94
Specific Governorate ID 35	Specific Governorate ID 35	2.77
Specific Governorate ID 70	Specific Governorate ID 70	2.12
All (Shuffled)	Residential	2.56
All (Shuffled)	Commercial	1.33
All (Shuffled)	Temporary	1.62
All (Shuffled)	Specific Governorate ID 35	2.63
All (Shuffled)	Specific Governorate ID 70	2.08

Table 3.17: Comparison of Model Performance: Segregated vs. Aggregated Training

Table 3.17 illustrate a slight improvement achieved by segmenting the training model by facility type for residential facilities and testing it on the same criteria. Meanwhile, all other

tests demonstrated better or equivalent performance when the model was trained using an aggregated approach.

3.4 Evaluation of Prediction Models using the Differential Penalty Score (DPS)

3.4.1 Purpose of the DPS Metric

The Differential Penalty Score (DPS) is a custom evaluation metric designed to assess the performance of NCDP, since accurately forecasting future events is critical to avoiding negative outcomes. In the context of predicting the next charge date for prepaid electricity meters, the DPS metric aims to quantify the effectiveness of different algorithms in forecasting the time until a customer's next charge with an emphasis on penalizing prediction errors asymmetrically. This asymmetric penalty approach is motivated by the operational necessity to minimize the risk of electricity cutoffs due to meter credit running out, which can result from underestimating the time until the next recharge.

3.4.2 DPS Formula

The DPS is calculated using the following formula, which applies different weights to overestimations and underestimations:

$$P(y_i, \hat{y}_i) = \begin{cases} a \times (\hat{y}_i - y_i) & \text{if } \hat{y}_i > y_i, \\ b \times (y_i - \hat{y}_i) & \text{if } \hat{y}_i \leq y_i. \end{cases} \tag{3.2}$$

$$DPS = \frac{1}{N} \sum_{i=1}^{N} P(y_i, \hat{y}_i). \tag{3.3}$$

where:

- y_i is the actual number of days until the next charge for the i-th observation.
- $\hat{y}_i$ is the predicted number of days until the next charge for the i-th observation.
- a is the penalty weight for overestimation.
- b is the penalty weight for underestimation, typically greater than a to reflect the higher cost of underestimations.
- N is the total number of observations.

The term 1/N in DPS formula serves to normalize the total penalty across all observations, making the DPS an average penalty per observation. This step is important to make the DPS becomes comparable across datasets of different sizes. Without normalization, larger datasets would naturally tend to have larger total penalties simply because they have more observations, not necessarily because they have worse predictive performance.

3.4.3 Comparison of Model Performances

The DPS was calculated for all predictive models used in NCDP to evaluate their performance in estimating the next charge date for prepaid electricity meters.

Table 3.18 indicates that RNN and XGBRegressor models exhibit the lowest DPS values, signifying their superior performance in predicting the days until the next charge with minimal risk of underestimating, this reduces the possibility of power outages due to meter credit running out.

Model	Differential Penalty Score (DPS)
Linear Regression	6.82
Lasso Regression	6.87
Polynomial Regression	6.43
Decision Tree	8.03
Random Forest	5.91
XGBRegressor	5.37
Lasso XGBRegressor	5.37
Deep Learning RNN	3.56

Table 3.18: Comparison of DPS across different predictive models.

Chapter 4

Analysis of result and Discussion

This thesis aimed to examine using classical machine learning and deep learning models in predicting the next charge date for prepaid meters, while focusing on prepaid electricity meters. We focused on analyzing customer historical patterns and service geographical location (i.e., spatio-temporal data) to extract specific and common patterns among the customers to perform our task. As a start to our research, we implemented multiple classical machine learning algorithms to perform our task. We started by training the algorithms based on raw charges dataset as a baseline to our models, as described in section 3.3.3. The results show all algorithms have relatively close scores, with an average of 6.8 of MAE. This result illustrates that building our task using only raw charges dataset will yield to high range of error value since 6.8 MAE can be very high for our predictive task. So, we merged service related information to the dataset by merging it to the charges dataset. The new merged data contains geographical location information for the service like "city, area", and information about the meter class like number of phases and amperes. Merging these features resulted in a slightly better performance for the algorithms with an enhancement about 0.7 MAE. These results illustrate that we need further features engineering to reach an accepted score. For that, we performed feature extraction and engineering to extract new features. This step led to a significant improvement in the model's performance, as illustrated in Table 13. The results showed a notable reduction in error metrics, with the XGBoost (XGB) model achieving the lowest scores: a Mean Absolute Error (MAE) of 3.65, Mean Squared Error (MSE) of 53.32, and Root Mean Square Error (RMSE) of 7.30. Following closely was the Random Forest model, which recorded an MAE of 3.74, MSE of 55.37, and RMSE of 7.44. These results highlight the effectiveness of refining our approach and the impact of feature selection on enhancing predictive accuracy.

Second, we implemented a deep learning approach using RNN to build our predictive task. Initially, we followed the same steps in classical machine learning, as we started with raw charges dataset to be a baseline for building the model. The result shows better performance comparing it with classical machine learning with an MAE of 4.5. Moreover, we performed training the model based on single governorate and test on all governorates. We found that training on single governorate will yield to unstable performance as shown in Figure 20, and when training by multiple governorates, the model will have a stable robust performance. In addition, we also performed merging and feature extraction and engineering. Our finding for this test showed that RNN performance enhances with more informative features, but it improves less than classical machine learning compared to raw charges features. Moreover, RNN are highly sensitive to time steps (window size) parameter. Altering this parameter changes the performance dramatically as shown in Figure 23, while it shows an improvement of performance by more than 50% between using 1 window size to 35 window size, with a result of improvements from 4.2 to 2.0 MAE. This test illustrates that using bigger window size can lead to more effective performance and accurate prediction. Also, we have experimented the effect of using different types of activation functions for the dense layers in the model, and found that using "relu" for first layer, and "linear" for output dense layer gives best performance, while using any other activation function for the output layer gives worse result. Also, scaling the target feature plays important role in affecting the performance of the model. We also run several experiments on scaled and non-scaled target feature, along with selecting different activation function, and the results shows that scaling the target feature needs to carefully select the dense layers activation functions, while keeping the target feature non scaled, reduce the importance of the activation function type.

These experiments collectively indicate that model performance is sensitive to dataset size, feature selection, data sorting methods, window size for time steps, and activation functions in layers. The best outcomes were achieved with larger datasets, comprehensive feature sets, shuffling techniques, and particular activation function configurations.

In addition, for the evaluation of predictive models using the newly introduced DPS metric function, the RNN method demonstrates superior performance over classical machine learning algorithms. As illustrated in Table 3.18, the RNN achieves the lowest DPS value of 3.56, indicating a significantly enhanced ability to predict the next charge date for prepaid electricity meters with minimal risk of underestimation.

As a summary for classical machine learning compared to deep learning approach using RNN, we can find that RNN performance has outperformed classical machine learning in all experiments. Also, RNN can give more accurate predictions by increasing time step (window size), where this parameter doesn't exist in classical machine learning algorithms. Meanwhile, classical machine learning algorithms are simpler, and need much less time for training and testing. Figure 4.1 shows a bar chart to compare the performance of the classical machine learning algorithms compared to RNN. The results are by applying final features set, and using time step of 35 for RNN.

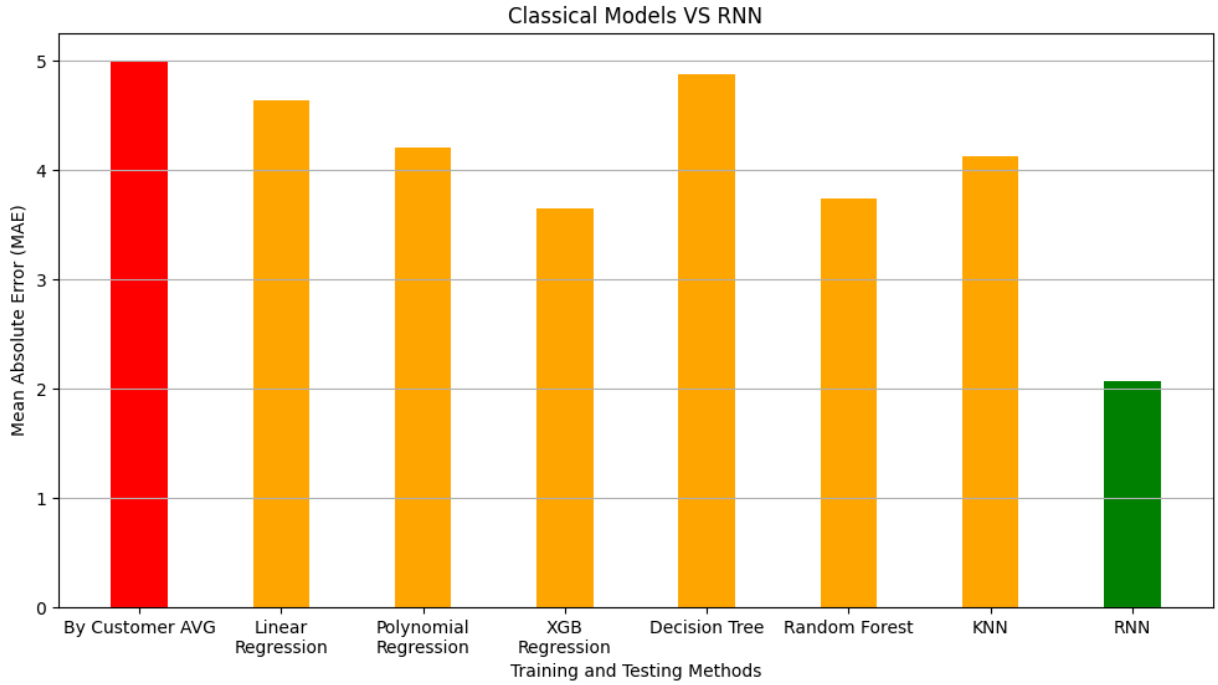

Figure 4.1: RNN VS classical machine learning algorithms performance

Chapter 5

Conclusion

This thesis has successfully demonstrated the application of both classical machine learning and advanced deep learning techniques in predicting the next charge date for prepaid electricity meters. The experiments conducted revealed key insights: larger datasets significantly improve the model's accuracy, and feature engineering plays a crucial role in enhancing prediction capabilities. The tests with different splitting methods for training data are important, feature subsets, and window sizes for time steps provided valuable insights into optimizing the model's performance.

The deep learning approach, especially with tuned hyperparameters and activation functions, showed promising results, outperforming classical models in certain configurations. However, classical machine learning models still hold value due to their simplicity and effectiveness with smaller or less complex datasets. In the other hand, while classical machine learning results were promising, RNN can work with fewer features and still give an accepted performance, while classical machine learning algorithms needs more features to perform less accurate predictions compared to RNN.

In conclusion, this thesis has demonstrated the utility of machine learning for predicting recharge times for prepaid meters, answering key questions about the application of these technologies. It was found that historical usage patterns, service-related data and the geographical location for the customer can significantly enhance model accuracy, highlighting the importance of feature engineering. Among tested algorithms, deep learning, particularly RNNs, showed notable performance, even with minimal features, compared to traditional machine learning models.

5.1 Future Work

Future work can explore the integration of these models into real-world applications, further enhancing the operational efficiency and customer experience in the utility sector. It also can include building and testing the model using datasets from different utility services such water or gas services, exploring different features such as weather, global or country major situation. Also, an experiment with newer or more complex deep learning architectures to improve prediction accuracy may be consider as a challenge for future works.

Bibliography

[1] A. Eliasy and J. Przychodzen, "The role of ai in capital structure to enhance corporate funding strategies," *Array*, vol. 6, p. 100017, 2020.

[2] W. Commons, "File:recurrent neural network unfold.svg — wikimedia commons, the free media repository," 2022. [Online; accessed 17-February-2023].

[3] International Energy Agency, *Energy Access Outlook 2020: From Poverty to Prosperity*. Paris: IEA, 2020.

[4] S. Horsley, "Millions at risk of losing power over unpaid bills," mar 2021. [Accessed 09 Apr 2023].

[5] H. K. Trabish, "Utility customers owe up to $40b in covid-19 debt, but who will pay it?," *Utility Dive*, Dec. 2020. [Accessed 9 Apr 2023].

[6] IMARC Group, "Prepaid electricity metering market: Global industry trends, share, size, growth, opportunity and forecast 2022-2027," 2022.

[7] J. S. Jones, "Over 31.3 million smart meters in gb," Mar. 2023. [Accessed 07 04 2023].

[8] M. Gocken, M. ÖZÇALICI, A. B. İpek, and A. Dosdoğru, "Integrating metaheuristics and artificial neural networks for improved stock price prediction," *Expert Systems with Applications*, vol. 44, 2015.

[9] K. Coussement and K. D. Bock, "Customer churn prediction in the online gambling industry: The beneficial effect of ensemble learning," *Journal of Business Research*, vol. 66, p. 1629–1636, 2013.

[10] P. R. Hoban and B. a. R. E., "Effects of internet display advertising in the purchase funnel: Model-based insights from a randomized field experiment," *Journal of Marketing Research*, vol. 52, 2015.

[11] Vakratsas, T. D., and Ambler, "How advertising works: What do we really know?," *Journal of Marketing*, 1999.

[12] H. a. K. Li and PK, "Attributing conversions in a multichannel online marketing environment: An empirical model and a field experiment," *Journal of marketing research*, vol. 51, pp. 40–56, 2014.

[13] X. Shao and Lexin, "Data-driven multi-touch attribution models," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 258–264, 2011.

[14] E. D. Ocansey", title = "Using Machine Learning to Predict Customers' Next Purchase Day, "Using machine learning to predict customers' next purchase day," 2021. [Online; accessed 17-February-2023].

[15] R. Berman, "Beyond the last touch: Attribution in online advertising," *Marketing Science*, 2018.

[16] T. W. Klug, A. D. Beyene, T. H. Meles, M. A. Toman, S. Hassen, M. Hou, B. Klooss, A. Mekonnen, and M. Jeuland, "Pre-paid meters and household electricity use behaviours: evidence from addis ababa, ethiopia," *Energy Policy*, p. 113226, Jan. 2022.

[17] M. AbuBaker, "Data mining applications in understanding electricity consumers' behavior: A case study of tulkarm district, palestine," *Energies*, vol. 12, 2019.

[18] Wikipedia contributors, "Jerusalem district electricity company — Wikipedia, the free encyclopedia," 2022. [Online; accessed 17-February-2023].

[19] A. Casarin and L. Nicollier, "Prepaid meters in electricity. a cost-benefit analysis," 2008.

[20] L. Franek, L. Šastný, and P. Fiedler, "Prepaid meters in electricity. a cost-benefit analysis," *IFAC Proceedings Volumes*, vol. 46, no. 1474-6670, pp. 428–433, 2013.

[21] H. Tech, "Utility meter," apr 2023. [Accessed 09 Apr 2023].

[22] Hexing, "Utility meters," 2023. [Accessed 09 Apr 2023].

[23] Conlog, "Conlog tech," 2023. [Accessed 10 Apr 2023].

[24] G. Ian, Y. Bengio, and C. Aaron, *Deep learning*. MIT press, 2016.

[25] R. Miotto, L. Li, B. A. Kidd, and J. T. Dudley, "Deep patient: An unsupervised representation to predict the future of patients from the electronic health records," *Nature Publishing Group*, vol. 8, pp. 1–10, 2018.

[26] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, 2015.

[27] L. Hardesty, "Explained: Neural networks." MIT News Office, 2017.

[28] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, 2020.

[29] K. Cho, B. v. Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *Proceedings of the Association for Computational Linguistics*, pp. 1724–1734, 2014.

[30] A. Gaikwad, *The Fundamentals of Machine Learning*. LAP Lambert Academic Publishing, 2023.

[31] P. Vasant, G. Weber, J. Thomas, J. Marmolejo-Saucedo, and R. Rodriguez-Aguilar, *Artificial Intelligence for Renewable Energy and Climate Change*. Wiley, 2022.

[32] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[33] A. Botchkarev, "A new typology design of performance metrics to measure errors in machine learning regression algorithms," *Interdisciplinary Journal of Information, Knowledge, and Management (IJIKM)*, vol. 14, pp. 045–076, 2019.

[34] V. N. Gudivada and C. Rao, *Chapter 8 - Machine Learning*, pp. 197–228. Elsevier, 2018.

[35] E. Ostertagova, "Modelling using polynomial regression," *Procedia Engineering*, vol. 48, p. 500–506, 12 2012.

[36] P. D. Reddy and L. R. Parvathy, "Prediction analysis using random forest algorithms to forecast the air pollution level in a particular location," in *2022 3rd International Conference on Smart Electronics and Communication (ICOSEC)*, pp. 1585–1589, 2022.

[37] L. Rokach and O. Maimon, "Decision trees," *The Data Mining and Knowledge Discovery Handbook*, vol. 6, pp. 165–192, 2005.

[38] S. Gu, "Identifying network connection: Benign vs. dos attack - a comparative analysis of binary classifiers," *Applied and Computational Engineering*, vol. 34, pp. 147–152, 01 2024.

[39] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* New York: Springer Series in Statistics, 2nd ed., 2009.

[40] M. Shapi, N. A. Ramli, and L. Awalin, "Energy consumption prediction by using machine learning for smart building: Case study in malaysia," *Developments in the Built Environment*, vol. 5, p. 100037, 2021.

[41] R. I. Rasel, N. Sultana, S. Akther, and A. Haroon, "Predicting electric energy use of a low energy," 2019.

[42] G. Toderean, I. Brânduşoiu, and H. Beleiu, "Methods for churn prediction in the pre-paid mobile telecommunications industry," in *2016 International Conference on Communications (COMM)*, pp. 97–100, 2016.

[43] B. Larivière and D. V. d. Poel, "Predicting customer retention and profitability by using random forests and regression forests techniques," *Expert Systems with Applications*, vol. 29, no. 2, pp. 472–484, 2005.

[44] P. Bhosale, G. Jadhav, A. Dhane, and N. Pise, "A dynamic churn prediction model using machine learning approach," *Expert Systems with Applications*, vol. 10, Sept. 2021.

[45] R. R, V. G, P. Jeyanthi, S. Revathy, L. Gladance, and V. Mary, "Prediction of electricity bill using supervised machine learning technique," in *International Conference on Recent Advances in Engineering, Technology and Science*, pp. 1232–1236, Apr. 2022.

[46] C. R. Karthik, Raghunandan, B. A. Rao, and N. V. S. Reddy, "Forecasting variance of niftyit index with rnn and dnn," *Journal of Physics: Conference Series*, 2022.

[47] X. Liu and J. Li, "Using support vector machine for online purchase prediction," in *International Conference on Electronic Commerce and Contemporary Economic Development*, pp. 1–6, 2016.

[48] M. Zeng, H. Cao, M. Chen, and Y. Li, "User behaviour modeling, recommendations, and purchase prediction during shopping festivals," June 2019.

[49] Y. Zhang, A. Wang, and W. Hu, "Deep learning-based consumer behavior analysis and application research," *Wireless Communications and Mobile Computing*, vol. 2022, p. Article ID 9147812, Apr. 2022.

[50] E. Samunderu and M. Farrugia, "Predicting customer purpose of travel in a low-cost travel environment—a machine learning approach," *Machine Learning with Applications*, vol. 9, no. 2666-8270, p. 100379, 2022.

[51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[52] C. Bhatt, "Data visualization and visual data mining," *CSI Communications*, pp. 12–13, 2014.

[53] T. Sandle, "Digital data #4: Looking for data trends and patterns with visualization," vol. 3, pp. 1–5, 07 2022.

[54] B. Silverman, *Density Estimation for Statistics and Data Analysis*. Chapman and Hall/CRC.

[55] M. L. Waskom, "seaborn: statistical data visualization," *Journal of Open Source Software*, vol. 6, p. 3021, 2021.

[56] R. K. Kumar and R. Chadrasekaran, "Attribute correction-data cleaning using association rule and clustering methods," *International Journal of Data Mining & Knowledge Management Process*, vol. 1, pp. 22–32, 2011.

[57] I. Guyon, M. Nikravesh, S. Gunn, and L. A. Zadeh, *Feature Extraction*. Springer Berlin Heidelberg, 2006.

[58] T. van der Ploeg, P. C. Austin, and E. W. Steyerberg, "Modern modelling techniques are data hungry: a simulation study for predicting dichotomous endpoints," *BMC Medical Research Methodology*, vol. 14, p. 137, 2014.