UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING AND INFORMATICS

A STUDY OF CLASSIFICATION OF ELECTRICITY CONSUMERS BY ELECTRICITY COMPANIES IN COMPARISON TO DYNAMIC DATA-DRIVEN CLUSTERING BASED ON CONSUMPTION PATTERNS

BY:

GODFREY OKOTH OTIENO

P52/33454/2019

SUPERVISED BY:

DR. WANJIKU NGANGA

A RESEARCH PROJECT REPORT SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS OF THE DEGREE OF MASTERS OF SCIENCE IN COMPUTATIONAL INTELLIGENCE OF THE UNIVERSITY OF NAIROBI

AUGUST 2021

DECLARATION

Student

I declare that the project as represented in this report is my own work and to the best of my knowledge has not been presented to any institution for any academic award.

Signature: Otieno………………………………… **Date:** 15th October 2021………………………………

Name: Godfrey Okoth Otieno **Registration Number:** P52/33454/2019

Supervisor's approval

This project report has been submitted in partial fulfillment of the requirements for the Master of Science in Computational Intelligence of the University of Nairobi with my approval as the Supervisor.

Signature: [illegible]………………………………… **Date:** 15th October 2021………………………………

Dr. Wanjiku Ng'ang'a

Senior Lecturer

School of Computing and Informatics

University of Nairobi

DEDICATION

I dedicate this research project to my parents Mr. and Mrs. Otieno, my siblings, my wife Maurine, my twin daughters Monica and Lucy, and my son Dan.

ACKNOWLEDGEMENT

I thank the Almighty God for the good health and wisdom He accorded me throughout my life and the opportunity to pursue my post-graduate studies. I also thank my beloved family for their unwavering support, love, and encouragement that kept me going throughout the academic years. To my supervisor, Dr. Wanjiku Ng'ang'a for whose advice and guidance without which I wouldn't have completed this project. I am deeply indebted. Special respect to my classmate Astone Owino for his many ideas and support during this research project.

May the Almighty God bless you all abundantly.

DEFINITION OF TERMS AND ABBREVIATIONS

CRISP-DM: Cross Industry Standard Process for Data Mining.

EPRA: Energy & Petroleum Regulatory Authority

KDD: Refers to Knowledge Discovery in Databases

KPLC: Kenya Power and Lighting Company

PCA: Principal Component Analysis

PL/SQL: Procedural Language extension to Structured Query Language (SQL).

SEMMA: The acronym SEMMA stands for Sample, Explore, Modify, Model, Assess, and refers to data mining framework.

ABSTRACT

This research study aimed at determining the impact of dynamic data-driven machine learning clustering technique on billing and revenue by the electricity companies with Kenya Power and Lighting Company (KPLC) as a case study. The study explored the customer classification techniques in use by the electricity distribution company and compared them with the K-Means clustering technique in machine learning. The electricity consumption data used for the research study comprised of 100,000 randomly selected electricity customer accounts for a 12 months consumption period starting from January 2020 to January 2021. The dataset was processed and explored using Python programming language on Jupyter Notebook. The mapping of clusters with the actual tariff classification by the clustering model placed correctly clustered customers at 52% based on their electricity consumption patterns. However, depending on the customers' consumption habits, the 48% getting clustered and mapped to different tariff categories with their actual grouping by the electricity distributor would greatly impact billing and revenue. The study recommends that electricity customer tariff classification by the electricity distribution companies, which informs billing, should be dynamic and data-driven. This, to a great extent, will ensure correct billing and attract more revenue as it is difficult and impossible for electricity distribution companies to detect the change in customer class or lifestyle. The dynamic and data-driven clustering as compared to static once-off customer classification at the point of application and connection to electricity supply will strike a balance between customers' correct and fair billing and the company's optimized revenue flow.

Table of Contents

List of Figures

List of Tables

CHAPTER 1: INTRODUCTION

1.0 Background of the study

In all markets and sectors, prices are determined by costs incurred and are structured to include a recommended return on investment and affordability as approved by the regulators. All carefully thought charges and levies are summed up as revenue requirements by the electricity distributors and sector regulators and then allocated to defined customer tariff categories in the energy sector. Tariff classification reflects the various customer categories within the market. These categories represent a different degree of usage, consumption levels, and prices per electricity unit. These groupings and their subsequent billing depend on the classes derived from various features. The features that are taken into consideration by the electricity distribution companies which inform the set of customers include Customer Features, Property Features, Geographical Features, and Network or Supply Features.

In some cases, electricity customer classification and billing depend on the type of meter installed in the customers' property. The tariff groupings consider the existence of the disadvantaged and poor in the society who are placed on the life-line tariff segment and are supposed to pay less than the average tariff to promote access and affordability principles. In reality, the customer classification often changes because of lifestyle changes, leading to additional new appliances such as fridges, air conditioning, etc. (Mutanen et al., 2011). According to Mutanen, Ruska, Repo, and Jarventausta (2011), it is not easy and in most cases, impossible for electricity suppliers to notice the changes in customer's class or lifestyle, which can only be informed by billing information.

1.1 Problem of Research

The main problem that electricity companies in developing countries face is the loss of revenue caused by inaccurate billing due to errors in electricity tariff classifications of their customers. Even though these companies are trying to make up for the billing inaccuracies to minimize revenue losses using other means like applying stepped tariff, with the rapid growth in customer population coupled with poor planning and boundaries distinctions, it is essential to look at the customers' classification from another angle which is more dynamic and data-driven. Viegas et al.(2016) noted that computational intelligence methods in machine learning are becoming more attractive and are used extensively in getting insights from the electricity grid systems' data. These methods provide decision-makers with predictive models and the ability to extract and use valuable knowledge. The hype around data science, analytics, and business intelligence are at its peak. Almost all companies are aware that data can help them improve their performance somehow, shape, or form. It is often said that data is only as valuable as the insights you will obtain from it. It is against this background that this study is going to explore the existing classification models in use by the electricity companies, establish a dynamic and data-driven clustering technique in machine learning and compare the billing and revenue with the current models in use.

1.2 Objectives of the study

1.2.1 Overall Objective

To study the customer classification techniques currently used by the electricity distribution companies and compare them with the dynamic data-driven clustering techniques in machine learning.

1.2.2 Specific research Objectives

i. To determine the customer's classification techniques in use by the electricity distribution companies.

ii. To determine the dynamic data-driven clustering techniques in machine learning and how they can be applied in electricity customer tariff grouping.

iii. To develop a clustering algorithm, apply it to customer consumption data, and be able to predict customer clusters.

iv. To evaluate and compare the impacts of classification methods in use and the dynamic machine learning clustering techniques on revenue.

1.3 Research Questions

i. What are the existing customer classification techniques in use by the electricity distribution companies?

ii. What are the data-driven clustering techniques in machine learning, and how do they apply to electricity customer tariff grouping?

iii. What impact on revenue does the dynamic data-driven classification techniques have compared to the existing classification techniques used by the electricity distribution companies?

1.4 Scope of the study

The study will take place in Nairobi, Kenya. The data that will be used in this research will be acquired from Indra Sistemas. Indra Sistemas (www.indracompany.com), an IT Company with headquarters in Madrid, Spain, is a leading IT multinational in Europe and Latin America. Indra Sistemas is organized around six vertical markets: Energy and Security; Transport and Traffic; Telecom and Media; Finance and Insurance; and Public Administration and Healthcare. In Africa, Indra Sistemas is the leading supplier of ERP Systems used by most electricity distribution companies. The system serves, among other modules, Customer Management and Billing. Indra Sistemas, through its Software Maintenance Factory located in Nairobi, Kenya, supplies the Customer Management System to major utility companies across Africa, for example, Kenya Power and Lighting Company (KPLC), ZESCO- Zambia, UMEME - Uganda, EDM - Mozambique, ESCOM - Malawi, EGC - Ghana, AEDC - Abuja, Nigeria, etc. For a dataset that qualifies to be an excellent source for this research in electricity consumption data analysis, the study will consider electricity consumption data for active customers from different tariff categories for a given period. For each active customer, there will be electricity usage within the given period of data extraction.

1.5 Significance of the study

Using data by applying machine learning clustering techniques in electricity customer classification and rate it against the existing customer classification techniques in use by the electricity distributors is very important in informing the electricity companies on the best method to adopt to maximize profit minimize loss. Knowing and being able to group electricity customers dynamically at any given point in time, based on their consumption patterns, can help distribution companies bill customers appropriately, optimize their business processes and increase revenue. It also helps in conflict resolution between electricity consumers and their suppliers in cases of overbilling due to wrong customer grouping, for example, by supply features. Data is one of the most valuable resources for any business today. When used correctly can help improve critical decisions in a company that can help position the business for success in today's data-driven world.

CHAPTER 2: LITERATURE REVIEW

2.0 Introduction

The section will explore the techniques that are used in determining customer behavior with a focus on electricity consumption and the theoretical framework used in the study. It will also uncover some of the existing techniques used for customer classification by the electricity distribution companies in Africa. This chapter will then review some of the clustering techniques and applications in electricity customers' classification and how the methods rank against each other.

2.1 Techniques that are used in determining customer behavior.

Customer behavior is a set of actions by the consumers of a certain commodity or service in a marketplace and the underlying motives for those actions. Electricity consumption is one of the aspects of consumer actions that can be used to define customers' behavior. According to Glauner et al.(2017), two fundamental approaches can be applied to determine customers' behavior. The techniques are:

i. Expert and rule-based systems that rely on domain knowledge to make decisions by and large using hand-crafted rules.
ii. Machine learning and Data mining methods that apply statistics to train and learn patterns from given datasets and make decisions for different unobserved data.

In both cases, there are justifications, and none is generally acceptable or more preferred than the other in the field of artificial intelligence. This study focuses on Data mining and Machine learning techniques and will try to establish the consumption behavior of electricity customers based on the consumption patterns as will be revealed by the Machine Learning Clustering algorithms applied.

2.2 Theoretical Framework

Depuru,(2012) noted that significant contributions are being made towards the development and implementation of algorithms for analyzing the electricity consumption patterns of customers. Most methods reviewed in this study use supervised learning. However, identifying electricity consumption patterns and classifying electricity customers is usually very challenging to learn in a supervised way (Glauner et al., 2017). Therefore, it is imperative to use better classification techniques in unsupervised learning to determine customer classes based on their consumption data and other features to inform billing decisions. The clustering algorithms are mainly used to determine the natural and statistical classification of data.

2.3 Clustering

The main objective of clustering is to group individual observations such that the observations from one group are very similar to each other. In addition, we would like them to be very different from the observations in other groups.

Currently, we have two broad categories of clustering, namely, Hierarchical and Flat. Flat clustering is also referred to as partitional aims at creating a set of clusters that are flat and without any explicit structure that relates one cluster to the other. On the other hand, Hierarchical clustering creates a structured order of clusters (Madhulatha, 2012). In this study, we are majorly going to apply K-means, which falls under the Flat clustering category.

The clustering algorithms use the feature space to learn the patterns and group the data in the classes shown below.

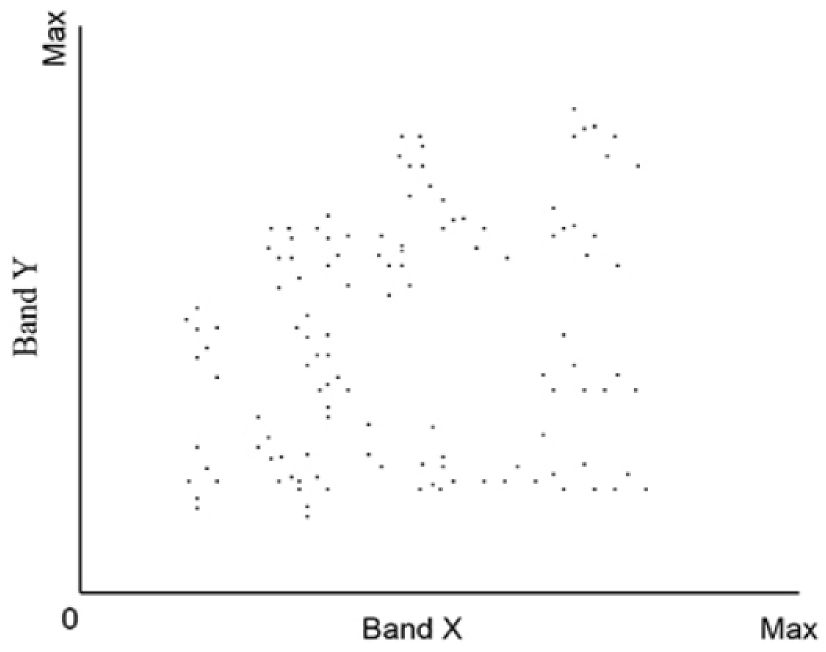


Figure 1: Feature space to learn the patterns

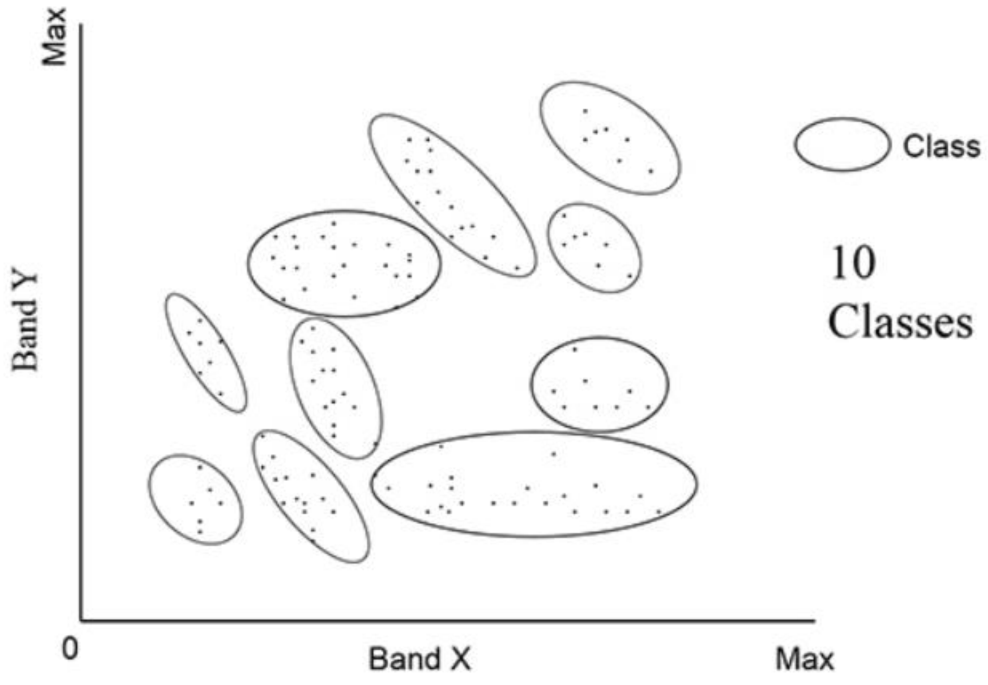


Figure 2: Grouped data in classes

The actual target of the clustering technique is to maximize the similarity of observations inside a cluster and maximize the difference between clusters. This, in another way, can be clearly expressed as minimizing the distance between points in a cluster and maximizing the distance between clusters. This is done with respect to some feature(s), and the clustering goal can be simply achieved as expressed by the steps below;

a. Explore several clustering problems
b. Perform cluster analysis
c. Find the optimal number of clusters.
d. Identify appropriate features
e. Interpret results.

The main tools used in this process are *sklearn* and *pandas* packages in python.

2.4 Clustering Analysis

Originally, cluster analysis was developed by anthropologists aiming to explain the origin of human beings. Later on, it was adopted by psychology, intelligent agencies, and other disciplines (Blashfield & Aldenderfer, 1988).

From the clustering results, we ask ourselves how the clustering solution we are presented with is useful. This question leads us to cluster analysis in the study. Cluster analysis is a multivariate statistical process that groups observations based on some of their features or variables that represent them. Naturally, observations in a dataset can be grouped differently, and doing so may sometimes be very useful in bringing out their features.

We have three different types of analysis, namely; Exploratory, Confirmatory and Explanatory. The exploratory analysis involves getting acquainted with the data, searching for patterns, and determining what methods may be useful to investigate further. Techniques like data visualization, descriptive statistics, and clustering are great ways to get acquainted with the data without exclusively trying to explain anything. There is no specific definition of confirmatory and explanatory analysis, but either way, they are different from an exploratory analysis. They aim to explain a phenomenon, confirm a hypothesis, or validate some previous research. That's why we normally use hypothesis testing and regression analysis (Flora, LaBrish & Chalmers, 2012).

Clustering may be used in the three types of analysis, but most commonly, it is used for exploratory analysis. To elaborate, our electricity customer clustering problem is a prime case of customer segmentation. Clustering can also be used as a confirmation of past beliefs. Maybe we knew the different clusters and just want to assign each observation to a different cluster. This is a rare case, though classification is a much more precise technique to deal with that. The most significant advantage in the case of this study is that as with the customer electricity consumption fundamental changes, a clustering solution may show that the clusters no longer look the same way. Thus the electricity consumption behavior for customers has changed.

2.5 Actual customer classification model by the Electricity Distribution companies.

Currently, most power utility companies use features captured in their systems at the point of processing customers' connection applications and during the connection to group customers. The customer group is what informs the billing. The features that are captured and used in customer tariff classification are listed below.

Table 1: Customer Classification Categories and Features.

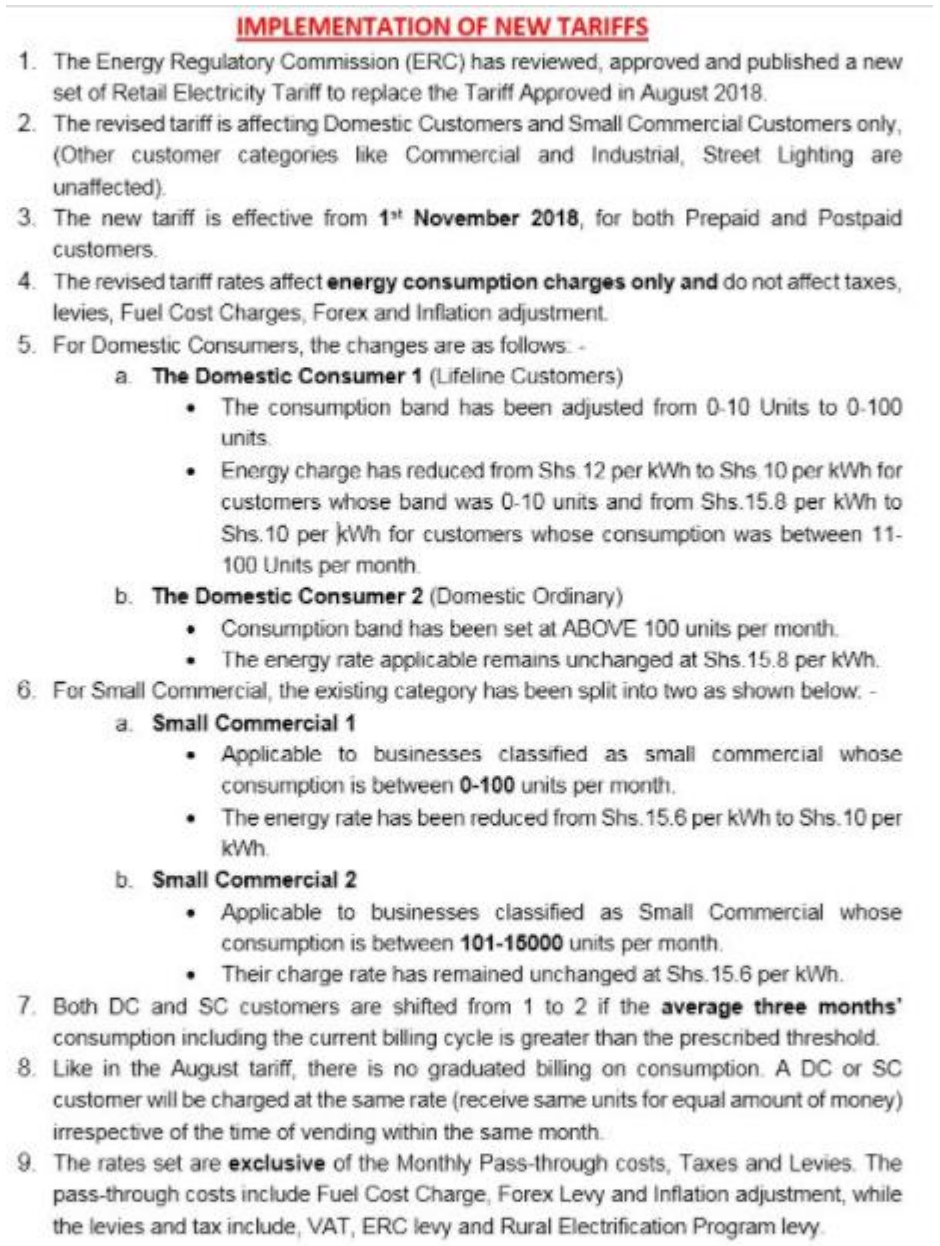
Category	**Features**
Customer	• Names • Identification Document (ID, Passport) • Identification Document Number • Gender • Title

	• Phone Number • Location: Physical Address defined by property number, Landmark feature, street name, etc.
Property	• Type (House, Flat, Bakery, Factory, Church)
Geographical	• Area • Township/Suburb • Town • District • County
Network / Supply	• Feeder details (Capacity) • Transformer details (Capacity) • Phase (Single-phase, Three-phase)

In Kenya, Kenya Power & Lighting Company, under the authority of EPRA, is mandated with the distribution and retail supply of electrical energy to consumers. KPLC proposes the applicable tariff charges based on the customer categories informed by classification features captured from the electricity connection application forms by customers. The proposed rates are then sent to EPRA for approval. The current rates being applied to electricity consumption were proposed by KPLC and approved in August 2018. However, there was a review for Domestic and Small Commercial Customers tariff rates, which was then implemented on 1st November 2018. This study, therefore, uses data based on the reviewed tariff rates as of 1st November 2018.

Table 2: Approved Charge Rates for 2018/19 – EPRA(Approved-Tariffs-Aug-2018.Pdf, n.d.)

Code	Customer Type (Code Name)	Energy Limit kWh/month	Charge Method	Unit	Approved Non Fuel Charge Rates
DC-L	Domestic - Lifeline	0-10	Energy	KShs/ kWh	12.00
DC-O	Domestic -	>11	Energy	KShs/kWh	15.80
SC	Small Commercial	0 – 15,000	Energy	KShs/ kWh	15.60
CI1	Comm./Industrial	>15,000	Energy	KShs/ kWh	12.00
			Demand	KShs/ kVA	800
CI2	Comm./Industrial	No Limit	Energy	KShs/ kWh	10.90
			Demand	KShs/ kVA	520
CI3	Comm./Industrial	No Limit	Energy	KShs/ kWh	10.50
			Demand	KShs/ kVA	270
CI4	Comm./Industrial	No Limit	Energy	KShs/ kWh	10.30
			Demand	KShs/ kVA	220
CI5	Comm./Industrial	No Limit	Energy	KShs/ kWh	10.10
			Demand	KShs/kVA	220
SL	Street Lighting	No Limit	Energy	KShs/kWh	7.50



IMPLEMENTATION OF NEW TARIFFS

1. The Energy Regulatory Commission (ERC) has reviewed, approved and published a new set of Retail Electricity Tariff to replace the Tariff Approved in August 2018.
2. The revised tariff is affecting Domestic Customers and Small Commercial Customers only, (Other customer categories like Commercial and Industrial, Street Lighting are unaffected).
3. The new tariff is effective from **1st November 2018**, for both Prepaid and Postpaid customers.
4. The revised tariff rates affect **energy consumption charges only and** do not affect taxes, levies, Fuel Cost Charges, Forex and Inflation adjustment.
5. For Domestic Consumers, the changes are as follows: -
   a. **The Domestic Consumer 1** (Lifeline Customers)
      - The consumption band has been adjusted from 0-10 Units to 0-100 units.
      - Energy charge has reduced from Shs.12 per kWh to Shs.10 per kWh for customers whose band was 0-10 units and from Shs.15.8 per kWh to Shs.10 per kWh for customers whose consumption was between 11-100 Units per month.

   b. **The Domestic Consumer 2** (Domestic Ordinary)
      - Consumption band has been set at ABOVE 100 units per month.
      - The energy rate applicable remains unchanged at Shs.15.8 per kWh.
6. For Small Commercial, the existing category has been split into two as shown below: -
   a. **Small Commercial 1**
      - Applicable to businesses classified as small commercial whose consumption is between **0-100** units per month.
      - The energy rate has been reduced from Shs.15.6 per kWh to Shs.10 per kWh.

   b. **Small Commercial 2**
      - Applicable to businesses classified as Small Commercial whose consumption is between **101-15000** units per month.
      - Their charge rate has remained unchanged at Shs.15.6 per kWh.
7. Both DC and SC customers are shifted from 1 to 2 if the **average three months'** consumption including the current billing cycle is greater than the prescribed threshold.
8. Like in the August tariff, there is no graduated billing on consumption. A DC or SC customer will be charged at the same rate (receive same units for equal amount of money) irrespective of the time of vending within the same month.
9. The rates set are **exclusive** of the Monthly Pass-through costs, Taxes and Levies. The pass-through costs include Fuel Cost Charge, Forex Levy and Inflation adjustment, while the levies and tax include, VAT, ERC levy and Rural Electrification Program levy.

Figure 3: A review for Domestic and Small Commercial Customers tariff rates(Electricity Cost Tariffs & Schedule of Tariffs 2018 | Kplc. Co.Ke, n.d.)

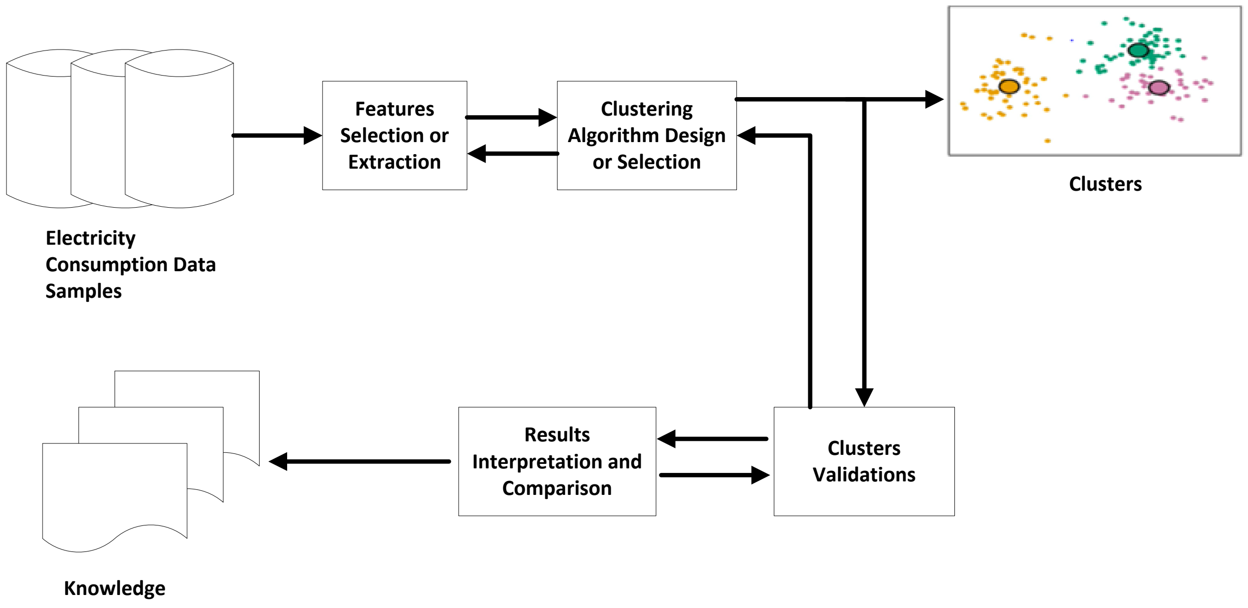
2.6 Data-Driven Clustering Techniques and Benefits.

One of the benefits of using clustering techniques is that they are relatively quick and easy to execute. They also don't require extensive prior knowledge of the domain area, but one must identify and label the groups after the classification. Using clustering techniques on customers' electricity consumption data to dynamically classify customers will use the customers' consumption, derive patterns, and group the customers accordingly irrespective of other features. Ozawa et al., (2016) argue that data-driven Clustering helps identify household's lifestyles and electricity consumption.

This is also an essential take for power utility companies to plan for load-shifting on the electricity grid.

2.7 Conceptual Framework

The conceptual framework of the proposed model is shown below.


Electricity Consumption Data Samples
Features Selection or Extraction
Clustering Algorithm Design or Selection
Clusters
Clusters Validations
Results Interpretation and Comparison
Knowledge

Source: (Xu & Wunsch, 2005)

Figure 4: A clustering procedure

- **K- means clustering process**.

K-means clustering is one of the most extensively applied clustering methods in research projects due to its simplicity. Here is how it works;

i. Choosing the number of clusters. The letter "K" in the name K-means denotes the cluster counts, and that is where this particular clustering technique derives its name from.

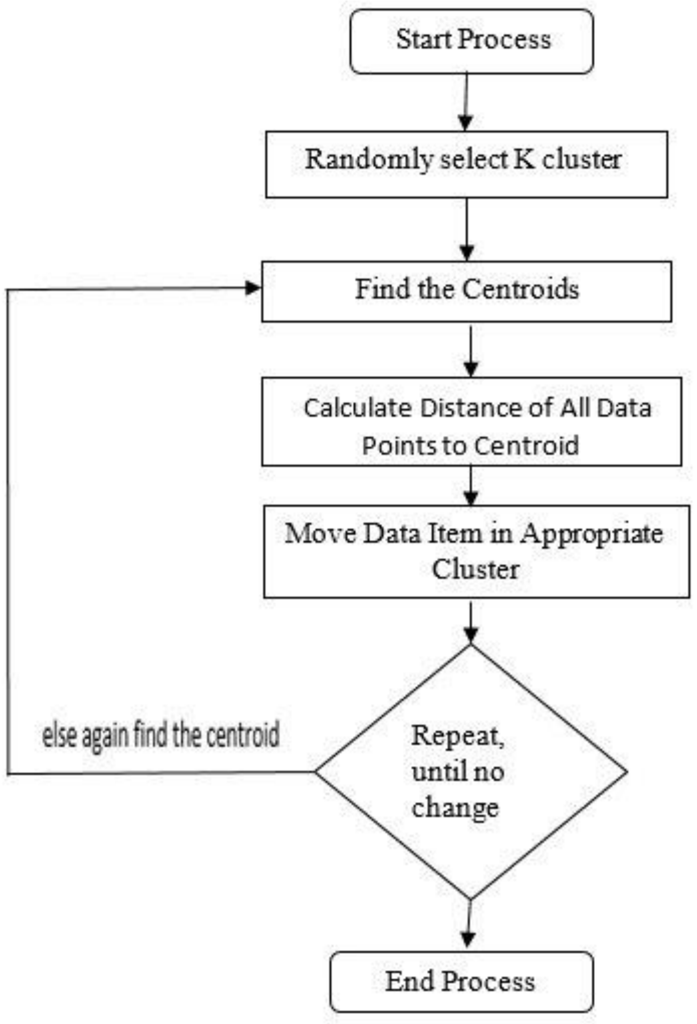
ii. Specifying the seed points. A cluster seed is basically the starting centroid. This point is randomly chosen or specified by the data scientist based on some prior knowledge about the research domain and the dataset.

iii. Assigning each data point in the dataset to a seed point. This is usually achieved by proximity, and each data point is assigned to its closest seed point, where all the data points are assigned to a cluster depending on the squared Euclidean distance from the seed points.

iv. The last step in the process is the calculation of the geometrical center of the clusters, also known as the centroid. Each seed point will then move closer to the data points in each cluster and become their centroid.

v. Repeating the last three steps. This can be done 10, 15, or even 1,000 times until an optimal clustering solution is achieved where the clusters can no longer be adjusted.

Because of the simplicity of the K-means clustering technique, it is prone to some issues. First, the squared Euclidean distance is quite sensitive to outliers. Usually, points that are too far away from the rest will always tend to form a cluster of their own. To overcome this, we can perform the k-median clustering (Madhulatha, 2012). However, it is more computationally expensive to achieve. Another drawback of this clustering method is that the number of clusters has to be specified before commencing the clustering process. This is a problem shared by most clustering algorithms. For the K-Means clustering technique, if the specified value of K is too small, e.g., K = 1, then the centroid will not fall inside the clusters, thus causing a misrepresentation of the data. On the other hand, if the chosen value of K is too large, e.g., K=10, then some of the clusters may end up being split into two. The third challenge with K-means is that it enforces clusters that are of spherical shapes or blobs. The reason being it tries to minimize the distance between each data point and the centroids in a straight line. Therefore, in a situation where clusters are more elongated, K-means will always have trouble separating them. Not to worry, it turns out that for data segmentation, K-means usually yields good results (Madhulatha, 2012). This is great because it is what we are using it for in this study.


Start Process
Randomly select K cluster
Find the Centroids
Calculate Distance of All Data Points to Centroid
Move Data Item in Appropriate Cluster
Repeat, until no change
else again find the centroid
End Process

Source: (Panwar & Gopal, 2016)

Figure 5: K-means Algorithm Process

2.8 Other Authors Related Works and Findings

According to Tureczek et al., (2018), several clustering methods have been used in research, and K-means is the most preferred one. Its derivatives like fuzzy K-Means and adaptive K-Means have also been used. Other clustering algorithms such as random effect mixture and hierarchical clustering models have also been used in most classification problems. Most research studies identified used K-Means as their core clustering technique and then compare other more advanced techniques to this baseline with an indeterminate conclusion on the best method for Clustering.

2.9 Research Gap

Based on the literature review, the notable limitation that comes with a majority of studies identified is that they focus more on comparing the results obtained from using different unsupervised clustering techniques like K-means, hierarchical Clustering, the self-organizing maps (SOM), etc., to categorize customers with similar electricity consumption behavior and not focus on comparing with the existing classification models in use by electricity companies to uncover the billing errors and impact on revenue (Chicco et al., 2006).

CHAPTER 3: RESEARCH DESIGN AND METHODOLOGY

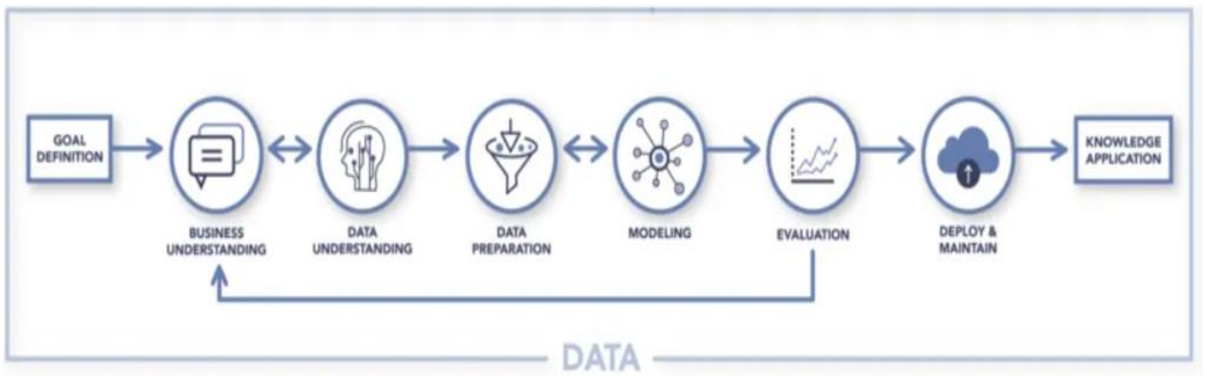
3.0 Introduction

This chapter covers the methodology used to analyze the data from various power utility companies by extracting features used in the existing classification models, establishing the billing model, and noting the resulting revenue as per the current customer classification model.

3.1 Research Design

Being a data science project, the research approach and design is based on a standard data mining methodology. The methodology and the entire process of this study is guided by Cross Industry Standard Process for Data Mining (CRISP-DM). The Cross-Industry Standard Process for Data Mining (CRISP-DM) framework is an extensible, high-level, and most effective process for a data science project (Thurber, 2020).

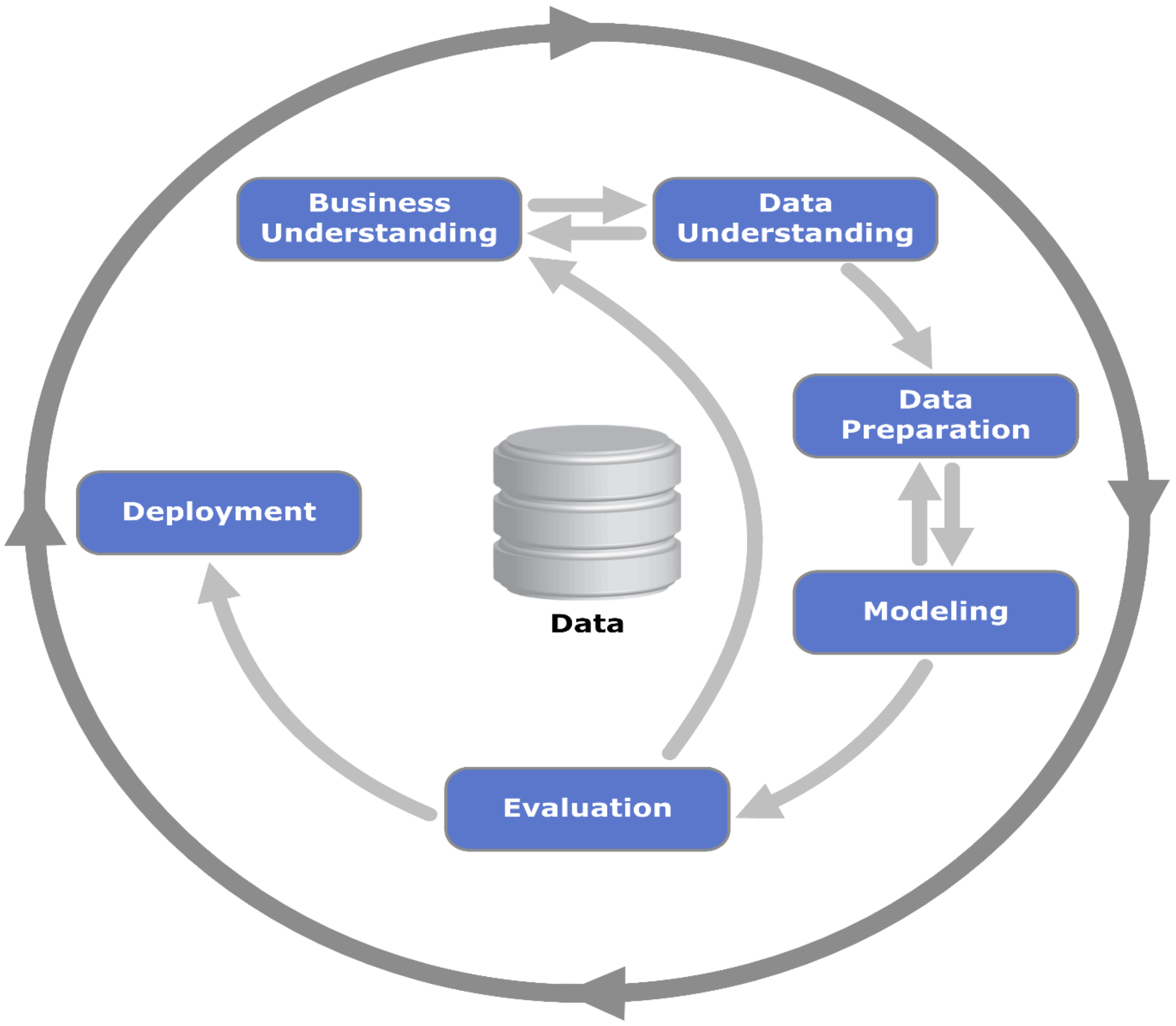
The CRISP-DM methodology is a good choice for this study because its six-phase model provides a straightforward, well-structured approach to planning any data mining project. The methodology is also robust and well-proven. The six phases of the CRISP-DM methodology that will guide this study are; **Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation,** and **Deployment.** *Figure 6* below shows the six main steps of the CRISP-DM process.



Source: (Thurber, 2020)

Figure 6: The CRISP-DM Process Framework

It is worth noting that CRISP-DM, just like the Agile software development methodology, is an iterative process framework even if its stages are outlined in the universal order in which they are executed. This is as depicted by the CRISP-DM life cycle in *Figure 7* below (Jensen, 2012).



Source: (Jensen, 2012)

Figure 7: CRISP-DM Life Cycle

3.1.1 Business Understanding

For any project and more so a project in data science, it is quite significant to have a good understanding of the business process and problems for the project to be a success. This initial phase of the methodology focuses on a proper understanding of the project needs and the objectives from the business perspective. The project requirements are then mapped to the objective, and the information is converted into a well-defined research problem. A project plan is then designed to achieve the outlined objectives (Wirth & Hipp, 2000).

3.1.2 Data Understanding

The data understanding phase of the methodology focuses on assessing the quality and finer details of the data to determine whether they are aligned and are able to support the objectives as outlined in the previous business understanding phase. This second phase usually starts with data collection and focuses on achieving data familiarity by identifying quality features and problems in the data, discovering patterns and generating important insights from the dataset, or detecting interesting features and hidden information to form hypotheses (Wirth & Hipp, 2000). Understanding business and data is important in the formulation of a research problem and helps in understanding the available variables for use.

3.1.3 Data Preparation

The data preparation phase takes into account all activities and processes to construct and transform the dataset from its initial raw form into an acceptable format that can be easily modeled by the machine learning algorithms. Data preparation tasks can be performed repeatedly and don't necessarily have to be performed in any prescribed order. The tasks include features selection, data cleaning, feature engineering of the data, and standardization of data for modeling (Wirth & Hipp, 2000).

3.1.4 Modeling

In the modeling phase, we look into different modeling techniques then select and apply the appropriate ones in line with the research objectives and the data we are dealing with. The parameters are labeled accordingly to optimal values that meet the model's expectations. Usually, there may be several modeling techniques that can be applied differently to the same research problem type. However, some techniques will require specific data formats for modeling. There is always a close link between Data Preparation and Modeling, and in most cases, data problems are realized while modeling and often lead to new ideas for generating new data (Wirth & Hipp, 2000).

3.1.5 Evaluation

This is the stage in the project where we have one or more models and approaches that appear to solve the research problem and have a high ranking quality from the data analysis point of view. Before the model is deployed, it is important to thoroughly evaluate and carefully review competing models to determine which model best addresses issues outlined in the business objectives. The best way to achieve this is to determine if there could be some important business problems that have not been sufficiently addressed. After the model evaluation, a decision on the application and use of the results should be made (Wirth & Hipp, 2000).

3.1.6 Deployment

This last phase of the methodology focuses on how to make the results from the evaluation phase actionable, easy to understand, and applied by the end-users for which the solution was meant. Having a model is generally not the ultimate aim and end of the project. Usually, the knowledge gained from the entire process is sorted, organized, and presented in a certain way for use by the targeted users. Depending on the user requirements and objectives of the project, the deployment phase can be as simple as reporting the findings in a document or as complex as implementing a business information system with embedded artificial intelligence. In most cases, the focus is on the user and not the data analyst who does the deployment. In any case, it is important to understand prior what actions will have to be carried out for the actual implementation and use of the created models. (Wirth & Hipp, 2000).

For most data science projects, there are other methodologies that can apply apart from the CRISP-DM process. For example, the Team Data Science Process (TDSP) created by Microsoft, which is also an agile and iterative data science methodology, can be used to deliver predictive analytics solutions and intelligent applications efficiently. Other frameworks that can also be used in the implementation of data science projects include SEMMA and KDD (Dåderman & Rosander, 2018).

3.2 Locality of the Project and Beneficiaries

The data-driven classification models will be applied to Software Factory data owned by Indra Sistemas. The data is acquired by Indra Sistemas from its power utility clients under a maintenance contract signed between the systems supplier and its clients. The main beneficiaries of this project, if adopted and implemented by Indra Sistemas, will be electricity distribution companies in Africa together with their electricity consumers.

3.3 Ethical clearance considerations

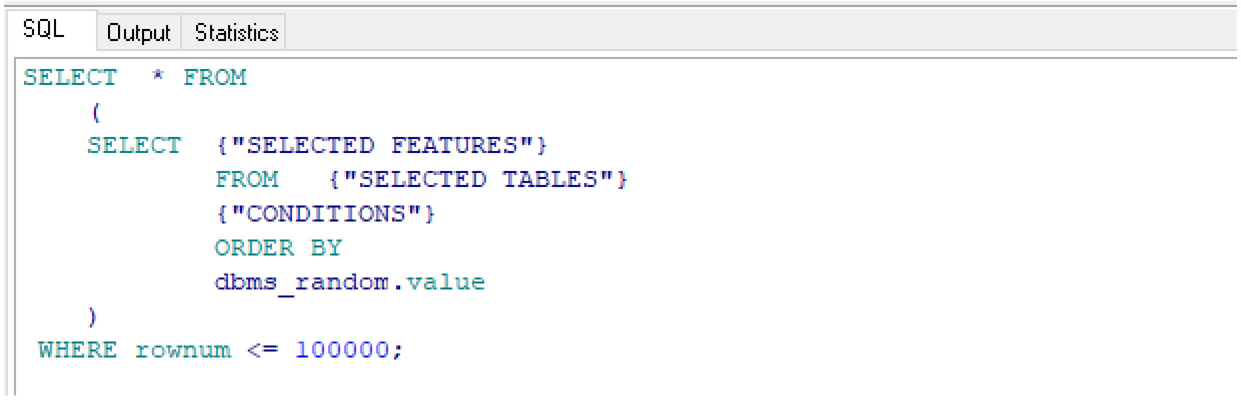
The research will not require any research permission letters from any authorities since authorization has been given by Indra Sistemas for the research to be done entirely on the electricity consumers' data from its clients available at the Software Factory in Nairobi. This authorization has been granted on the condition that the research outcome will be made available and shared with the company.

CHAPTER 4: ANALYSIS, RESULTS, AND DISCUSSION

4.0 Data for the Study

It is always important to know the domain and understand the data used in a research study. For this study, we use electricity customers' consumption data from KPLC. The electricity consumption dataset consists of 100,000 randomly selected electricity customer accounts for 12 months consumption period starting from January 2020 to January 2021. The dataset was extracted from KPLC software maintenance database version Oracle Database 12c, hosted by Indra Sistemas at its software factory in Nairobi.

To ensure a random sample dataset, the extraction query utilized the oracle *DBMS_RANDOM.VALUE* functionality for the 100,000 observations as shown by the PL/SQL query below.



```
SELECT  * FROM
    (
    SELECT  {"SELECTED FEATURES"}
            FROM   {"SELECTED TABLES"}
            {"CONDITIONS"}
            ORDER BY
            dbms_random.value
    )
 WHERE rownum <= 100000;
```

Figure 9: PL/SQL Query template for Data extraction.

The random value *dbms_random.value* generated by the Oracle database makes every row selected from the sample(n) have the same probability of being in the specified 100,000 returned rows (*Database PL/SQL Packages and Types Reference*, n.d.).

This is raw data, and for this study, the dataset will undergo preprocessing by encoding some variables and handling the missing values in the dataset. It is also important to note at this point that the extraction process of the data restricted the volume of the dataset to 100,000 samples and protected the privacy of the customers by omitting sensitive information like the ID and Phone Numbers of the customers.

Data description	Value
Country	Kenya
Type	Electricity consumption data for customers from different classes
Supplier	KPLC through Indra Systemas.
Size	100,000 observations
Length	96 features
Period	12 Months
Start	January 2020
End	January 2021
Referral	This data has not been used for research or referenced before.

Table 3: Initial data description of Customers' Electricity consumption data, comprising 100,000 customers

4.1 Data Exploration and Description

4.1.1 Manual Exploration of the Dataset

It is good practice to start by manually exploring the dataset. I prefer opening the dataset for the first time in MS excel. The main reason is that python is not optimized for eye-balling data, and so it's much more convenient to use a spreadsheet instead. From the manual exploration, we come up with a summary of information in the form of a data dictionary described by the data legends below;

Variable	Data type	Range	Description
Account	numerical	Integer	This is customer's account Number. It is a unique identifier of a customer by the Electricity Distributor.
Service Point	numerical	Integer	This is customer's point of connection to electricity supply.It is a unique identifier of a customer's point of connection by the Electricity Distributor.
Customer	string	varchar	This is customer's Name. Shows the name of a customer as registered by the Electricity Distributor.
Tariff Code	string	varchar	This is customer's consumption category code. Shows the customer's category code as grouped by the Electricity Distributor.
Tariff Desc	string	varchar	This is customer's consumption category code description. Shows the customer's category description as grouped by the Electricity Distributor.
Enrollment Date	date	date	This is the date the customer is connected to electricity supply.
De-Enrollment Date	date	date	This shows the date the electricity supply contract with the customer is terminated.
Account Status	string	varchar	This shows the customer's account status with regards to electricity supply or connection.
Contract Status	string	varchar	This shows the contract status between the customer and the Distribution Company with regards to electricity supply or connection.

Table 4: Customer Information

Variable	Data type	Range	Description
Meter No.	numerical	Integer	This is customer's Meter Number. It is a unique identifier of the meter installed at the customer's premise.
Model Name	string	varchar	This is the name of the Meter installed at the customer's premise.
Meter Type	string	varchar	This is the name of the Meter installed at the customer's premise.
Meter Make	string	varchar	This is the name of the Meter installed at the customer's premise.

Table 5: Meter Information

Variable	Data type	Range	Description
Postal Address	string	varchar	This is the Postal Address of the customer as registered by the Electricity Distributor.
Feeder Name	string	varchar	This is the name of the point linking the substation and the transformer for the supply point of the customer.
Region	string	varchar	This is the regional location of the customer on the grid as registered by the Electricity Distributor.
County	string	varchar	This is the county location of the customer on the grid as registered by the Electricity Distributor.
Branch	string	varchar	This is the administrative location of the customer as identified by the Electricity Distributor.

Table 6: Geographical Information

Variable	Data type	Range	Description
KVA Units	numerical	Real	This is customer's electricity consumption corresponding to the Power component as recorded by the customers meter.
Active Units	numerical	Real	This is customer's electricity consumption corresponding to the Energy component as recorded by the customers meter.
Total Consumption	numerical	Real	This is customer's electricity consumption sum of Power and Energy components taken from the customers meter.
Kva Amnt	numerical	Real	This is customer's electricity consumption charge by the electricity Distributor for the Power component.
Active Amnt	numerical	Real	This is customer's electricity consumption charge by the electricity Distributor for the Energy component.
Total Amount	numerical	Real	This is the sum of charges for Power and Energy components.

Table 7: Consumption and Revenue Information

Data exploration means getting some base information on our dataset. Once we have a good idea of what our dataset contains, we transfer it to Python because it is difficult to understand the information contained in our dataset just by looking or perusing through the file.

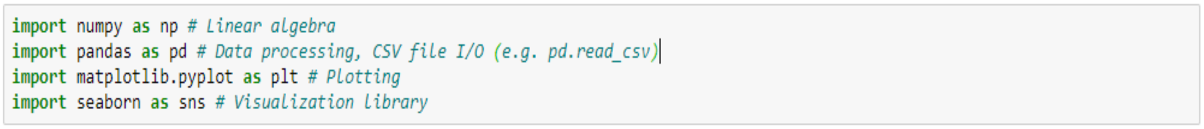
4.1.2 Exploration and description of the Dataset in Python

It is always important to be well informed about the dataset in a research study and know the actual contents. In this study, we start the analysis by exploring each column of the dataset to get a better idea of what each column in the data frame is or means. For this study, the data exploration and description exercise is done in Jupyter Notebook. Jupyter Notebook is an open-source web tool that is interactive and is used in computational processes. This tool helps researchers to bring together the software code, computational inputs and outputs, explanatory texts, and multimedia resources in a single document (Perkel, 2018). "Jupyter" is a loose acronym meaning Julia, Python, and R, all programming languages that were the first target of the Jupyter application (Parkyn, 2019). The notebook technology has since evolved and now supports many other languages. This study focuses only on using Python programming language on Jupyter Notebook. To be able to explore, understand and describe the datasets in Python, we have to make use of libraries, and some of the essential and necessary dependencies this study will use include;

- **pandas** – This is the library that is used to import and manage data sets.
- **NumPy** – This is a library that contains mathematical tools. Basically, this is the library that we need for us to handle any mathematics in our code. Since this study involves machine learning models based on mathematics, we'll need NumPy from time to time to do them but not all the time.
- **Matplotlib** – This is another essential library this study will use. This library has a sub-library called pyplot, which helps in plotting charts. It contains very intuitive and helpful tools which will be used in the visualizations part of this study.

a. Importing Python Libraries

In exploring and describing the dataset, we start by importing all the Python necessary dependencies and useful packages required for data analysis. We begin by importing the most important libraries required, that is, Numpy, Pandas, Matplotlib, and Seaborn. Throughout the study, we can then gradually import the rest as the need arises.

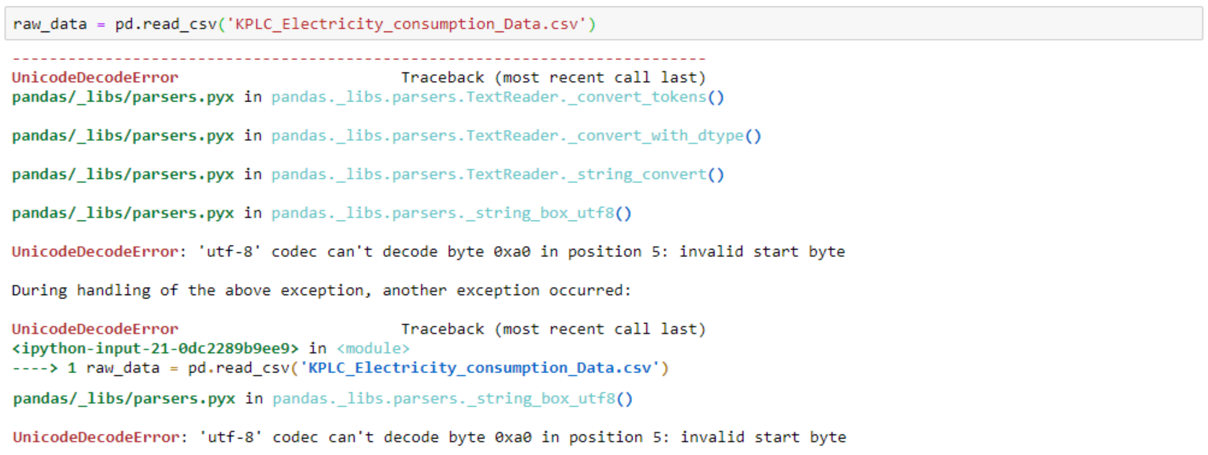


```
import numpy as np # Linear algebra
import pandas as pd # Data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # Plotting
import seaborn as sns # Visualization library
```

Figure 10: Code snippet for Importing Relevant Libraries

b. Load and read the Dataset

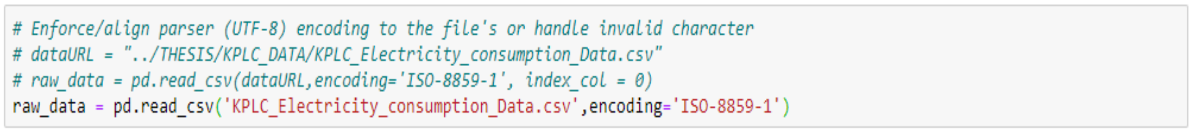
After importing the necessary dependencies, we load the dataset from the .csv file into a new variable called raw_data. Pandas library already imported comes in handy and helps in loading the data into a data frame. Pandas data frame is an object that is very convenient for storing data for all kinds of manipulation and analysis. Unfortunately, in our case, when trying to load and read the electricity consumption data using pandas library the usual way, we run into errors, as shown below.



```
raw_data = pd.read_csv('KPLC_Electricity_consumption_Data.csv')

---------------------------------------------------------------------------
UnicodeDecodeError                        Traceback (most recent call last)
pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._convert_tokens()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._convert_with_dtype()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._string_convert()

pandas/_libs/parsers.pyx in pandas._libs.parsers._string_box_utf8()

UnicodeDecodeError: 'utf-8' codec can't decode byte 0xa0 in position 5: invalid start byte

During handling of the above exception, another exception occurred:

UnicodeDecodeError                        Traceback (most recent call last)
<ipython-input-21-0dc2289b9ee9> in <module>
----> 1 raw_data = pd.read_csv('KPLC_Electricity_consumption_Data.csv')

pandas/_libs/parsers.pyx in pandas._libs.parsers._string_box_utf8()

UnicodeDecodeError: 'utf-8' codec can't decode byte 0xa0 in position 5: invalid start byte
```

Figure 11: Data read errors due to encoding format mismatch

We then proceed to identify the cause of the errors, which is established to be associated with the parser when it is trying to parse data in utf-8 encoding format, but our file seems to be in a different encoding format or contains invalid characters. We resolve the errors simply by telling python the actual encoding format of the file, and in this case, we do it by enforcing the encoding format to Latin Codec (ISO-8859-1), (*Codecs — Codec Registry and Base Classes — Python 3.9.5 Documentation*, n.d.).



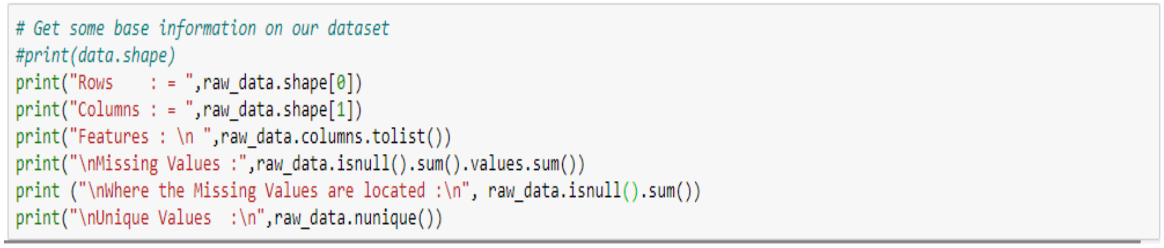
```
# Enforce/align parser (UTF-8) encoding to the file's or handle invalid character
# dataURL = "../THESIS/KPLC_DATA/KPLC_Electricity_consumption_Data.csv"
# raw_data = pd.read_csv(dataURL,encoding='ISO-8859-1', index_col = 0)
raw_data = pd.read_csv('KPLC_Electricity_consumption_Data.csv',encoding='ISO-8859-1')
```

Figure 12: Code snippet for data read errors resolution

c. Explore the Dataset

With the rough idea we already have on what is contained in the dataset, we can validate it by using the *head()* method to display a few rows and columns of the data frame. Data description means getting statistics for the numerical columns contained in our dataset. We use the inbuilt *describe ()* function to display statistical information for the numerical columns of our dataset, such as count, mean, standard deviation, Interquartile ranges, minimum and maximum values from the dataset.

We explore the data further by finding and displaying information such as the number of rows, columns, Features, Missing values, and Unique values from the dataset. To simplify the process, we invoke the inbuilt python functionalities such as *tolist (), isnull (),* and *nunique (),* among others, to compute and display the information we want from the dataset.



```
# Get some base information on our dataset
#print(data.shape)
print("Rows     : = ",raw_data.shape[0])
print("Columns : = ",raw_data.shape[1])
print("Features : \n ",raw_data.columns.tolist())
print("\nMissing Values :",raw_data.isnull().sum().values.sum())
print ("\nWhere the Missing Values are located :\n", raw_data.isnull().sum())
print("\nUnique Values  :\n",raw_data.nunique())
```

Figure 13: code snippet for data Exploration

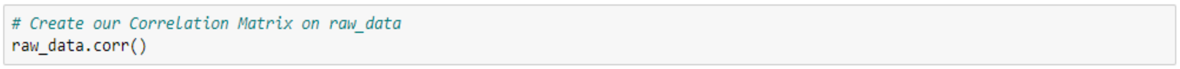
From the output displayed by the code snippet above, it shows that our dataset consists of 100,000 observations and 96 features as expected. The dataset also has missing values but none on the key features like Customer Account, Supply point, and Meter Number.

Filter	Customers	Note
Raw data	100000	Number of Customers in the original dataset
Missing	0	Missing records
Work data	100000	Number of customers in the dataset for clustering

Table 8: Output showing number of observations, features, and missing values

d. Correlation Analysis

A good way to gain an initial understanding of the relationships between the different variables is to explore how they correlate. Depending on different types of data, different techniques can be applied to quantify the correlation. For our initial data exploration purposes, a simple Pearson Correlation will suffice. Pearson is the default approach for most correlation methods in python. We calculate the correlations between our variables using the python *corr()* method applied to our *raw_data* dataset.

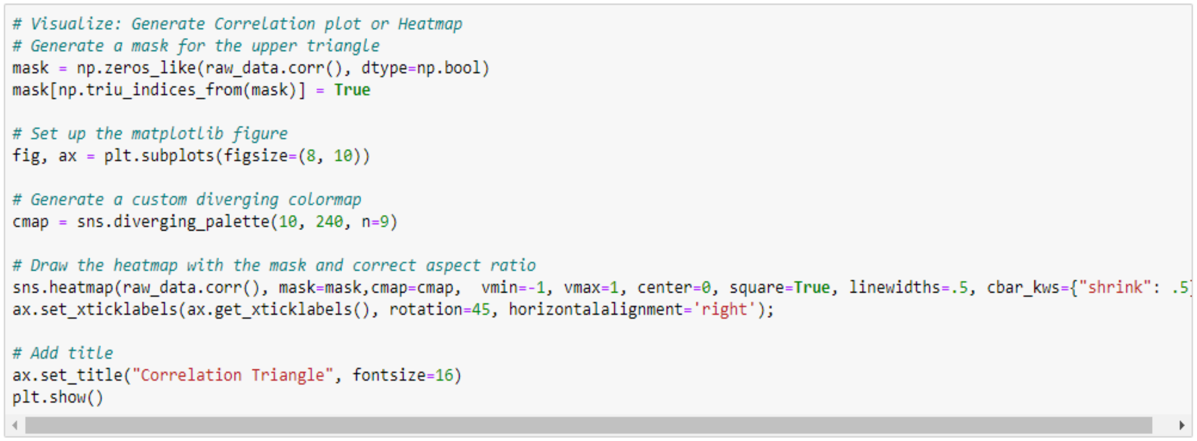


```
# Create our Correlation Matrix on raw_data
raw_data.corr()
```

Figure 14: Code snippet to create a correlation matrix on raw data

Correlation analysis simply means which attributes have correlations in our dataset. For example, one column being correlated to another column. Correlation, on the other hand, is the understanding of the relationship between two variables. Generally, the correlation describes the linear dependency between variables. It ranges from -1 to 1, with 1 indicating a strong positive correlation, and -1 on the other hand, indicating a strong negative correlation. A correlation of zero(0) between two variables means they are not linearly independent. When we consider two variables, we can sometimes say they are correlated. An example in the case of this study is the customer electricity consumption and the amount charged. So when we have a higher electricity consumption value, we would expect to have a corresponding higher amount charged, which translates to higher revenue to the electricity distribution company. In such a case, the attributes are considered to have a high correlation with each other, and we will remove the amount charged on electricity consumption from our analysis or from building the model because it might not affect our model. For this study, we are not going to remove them but instead compute the average from these columns, as we'll see in the next stage of feature engineering. This way, we'll reduce the number of features to which we'll subject to clustering.

A correlation matrix is an extremely useful tool as we are able to see the exact correlation values between variables. Unfortunately, it is hard to get a general overview of the relationships between the features because we are simply looking at numbers. Since we humans are visual creatures, we can often use different data visualization techniques to perceive information more intuitively. A very convenient way for this is a correlation plot or heatmap. The process of generating the correlation plot is shown by the code snippet below.



```
# Visualize: Generate Correlation plot or Heatmap
# Generate a mask for the upper triangle
mask = np.zeros_like(raw_data.corr(), dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
fig, ax = plt.subplots(figsize=(8, 10))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(10, 240, n=9)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(raw_data.corr(), mask=mask,cmap=cmap,  vmin=-1, vmax=1, center=0, square=True, linewidths=.5, cbar_kws={"shrink": .5
ax.set_xticklabels(ax.get_xticklabels(), rotation=45, horizontalalignment='right');

# Add title
ax.set_title("Correlation Triangle", fontsize=16)
plt.show()
```

Figure 15: Code snippet to visualize the correlation on raw data

When the code above is executed, it displays the plot shown below. As we go through the analysis of our data, we tend to find sometimes the variables don't correlate like we would expect them to. This is a situation of outliers, and outliers are really a good way of being able to look into the data and understand why the data doesn't correlate or work in an expected way.

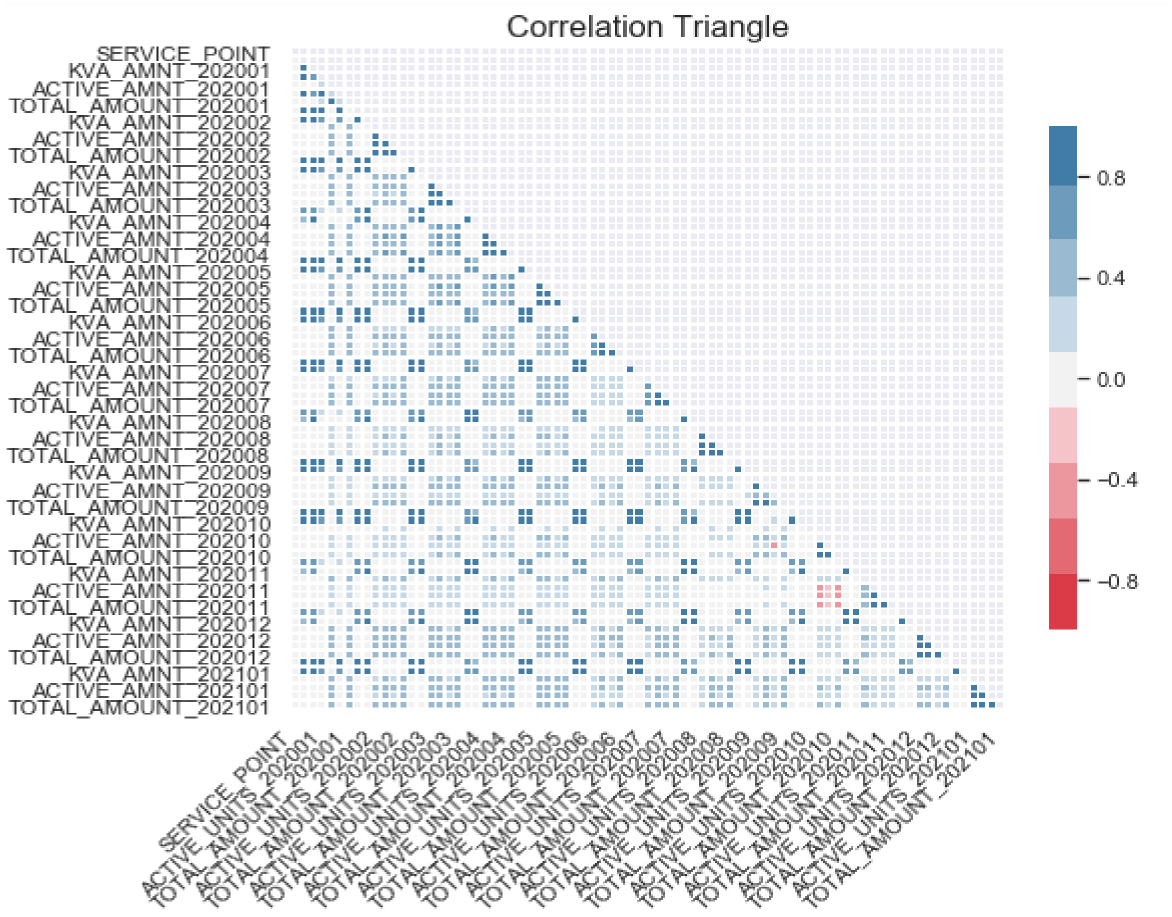
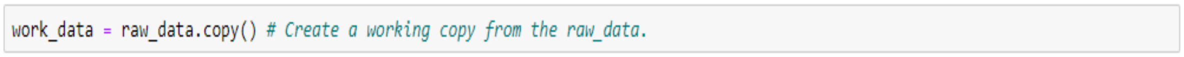


Figure 16: Correlation Triangle

The diagonal values show the correlation of a variable with itself. The values are always 1, and the matrix is symmetrical. So the entries over and under the diagonal are mirror images of themselves, and this is the reason why we have represented the correlation matrix by a triangle depicting the lower part of the correlation matrix table.

4.2 Feature Engineering

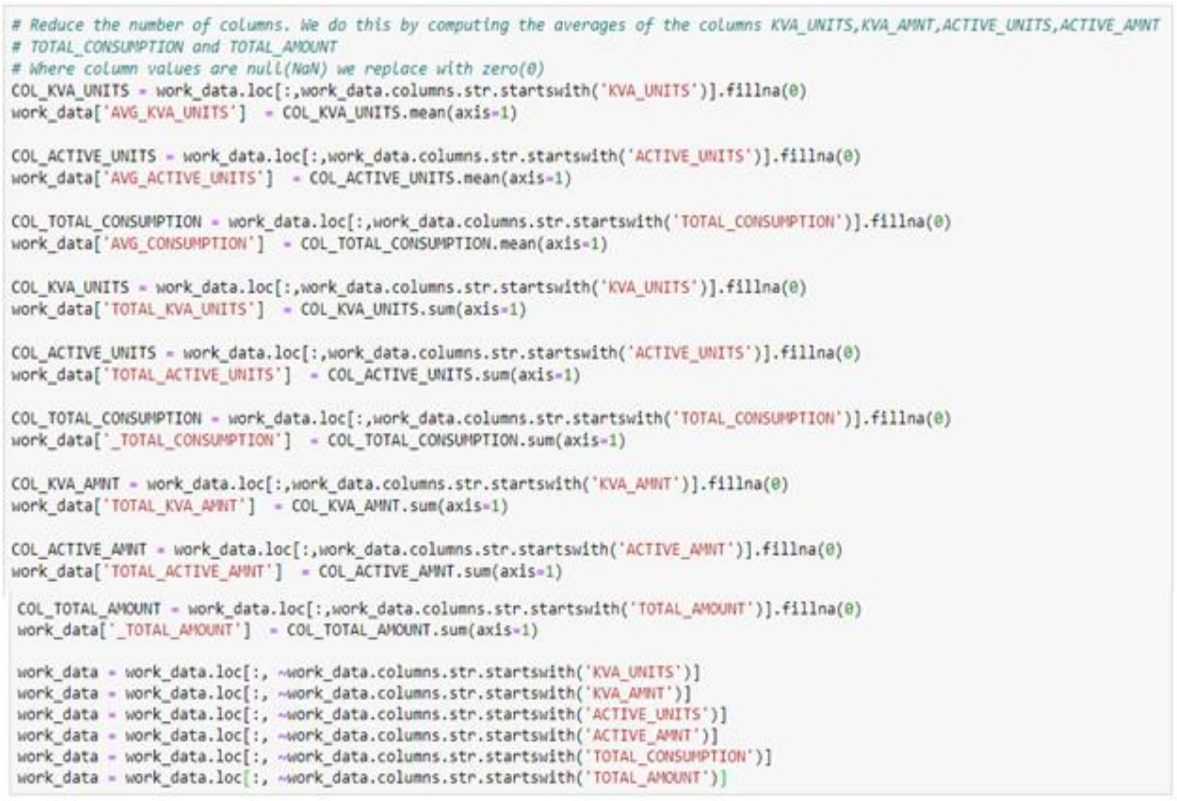
Feature Engineering is the technique of extracting more useful features from existing data. So, in this case, we create a couple of new, more useful columns from the existing ones and remove some that are not useful. We start by creating a working copy(*work_data*) of the original raw data (*raw_data*). It is an important practice to always do this so that at whatever stage of the analysis, we have a clear picture and results based on the dataset in use after the original dataset has undergone transformations. It also helps in keeping track of the data manipulations by maintaining the subsequent data structure even after a series of manipulations.



```
work_data = raw_data.copy() # Create a working copy from the raw_data.
```

Figure 17: Code snippet to create a working copy from a raw dataset

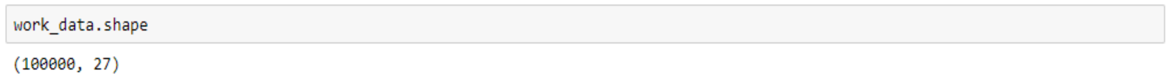
We reduce the number of columns by computing the averages of the consumption columns KVA_UNITS, ACTIVE_UNITS and TOTAL_CONSUMPTION. We are doing this because our dataset was extracted for electricity consumption over a period of 12 months, and so we want to work with monthly average consumption data for each customer. We also compute the total sum of the consumptions and charges for the whole period. For the accurate computation of these values, we ensure that missing data is handled appropriately, and we utilize the python method *fillna()* and replace the missing values by 0 by applying *fillna(0)*. This ensures we compute a standard result even for customers with missing data in some of the columns. Finally, we remove the unnecessary columns from our working dataset.



```
# Reduce the number of columns. We do this by computing the averages of the columns KVA_UNITS,KVA_AMNT,ACTIVE_UNITS,ACTIVE_AMNT
# TOTAL_CONSUMPTION and TOTAL_AMOUNT
# Where column values are null(NaN) we replace with zero(0)
COL_KVA_UNITS = work_data.loc[:,work_data.columns.str.startswith('KVA_UNITS')].fillna(0)
work_data['AVG_KVA_UNITS']  = COL_KVA_UNITS.mean(axis=1)

COL_ACTIVE_UNITS = work_data.loc[:,work_data.columns.str.startswith('ACTIVE_UNITS')].fillna(0)
work_data['AVG_ACTIVE_UNITS']  = COL_ACTIVE_UNITS.mean(axis=1)

COL_TOTAL_CONSUMPTION = work_data.loc[:,work_data.columns.str.startswith('TOTAL_CONSUMPTION')].fillna(0)
work_data['AVG_CONSUMPTION']  = COL_TOTAL_CONSUMPTION.mean(axis=1)

COL_KVA_UNITS = work_data.loc[:,work_data.columns.str.startswith('KVA_UNITS')].fillna(0)
work_data['TOTAL_KVA_UNITS']  = COL_KVA_UNITS.sum(axis=1)

COL_ACTIVE_UNITS = work_data.loc[:,work_data.columns.str.startswith('ACTIVE_UNITS')].fillna(0)
work_data['TOTAL_ACTIVE_UNITS']  = COL_ACTIVE_UNITS.sum(axis=1)

COL_TOTAL_CONSUMPTION = work_data.loc[:,work_data.columns.str.startswith('TOTAL_CONSUMPTION')].fillna(0)
work_data['_TOTAL_CONSUMPTION']  = COL_TOTAL_CONSUMPTION.sum(axis=1)

COL_KVA_AMNT  = work_data.loc[:,work_data.columns.str.startswith('KVA_AMNT')].fillna(0)
work_data['TOTAL_KVA_AMNT']  = COL_KVA_AMNT.sum(axis=1)

COL_ACTIVE_AMNT = work_data.loc[:,work_data.columns.str.startswith('ACTIVE_AMNT')].fillna(0)
work_data['TOTAL_ACTIVE_AMNT']  = COL_ACTIVE_AMNT.sum(axis=1)

COL_TOTAL_AMOUNT = work_data.loc[:,work_data.columns.str.startswith('TOTAL_AMOUNT')].fillna(0)
work_data['_TOTAL_AMOUNT']  = COL_TOTAL_AMOUNT.sum(axis=1)

work_data = work_data.loc[:, ~work_data.columns.str.startswith('KVA_UNITS')]
work_data = work_data.loc[:, ~work_data.columns.str.startswith('KVA_AMNT')]
work_data = work_data.loc[:, ~work_data.columns.str.startswith('ACTIVE_UNITS')]
work_data = work_data.loc[:, ~work_data.columns.str.startswith('ACTIVE_AMNT')]
work_data = work_data.loc[:, ~work_data.columns.str.startswith('TOTAL_CONSUMPTION')]
work_data = work_data.loc[:, ~work_data.columns.str.startswith('TOTAL_AMOUNT')]
```

Figure 18: Code snippet showing Feature Engineering

By doing feature engineering, we manage to compute useful columns and reduce the unnecessary ones but still maintain the number of rows to 100,000, which is the actual number of observations from the start. We now have a working dataset (*work_data*) with 100,000 observations and 27 features.

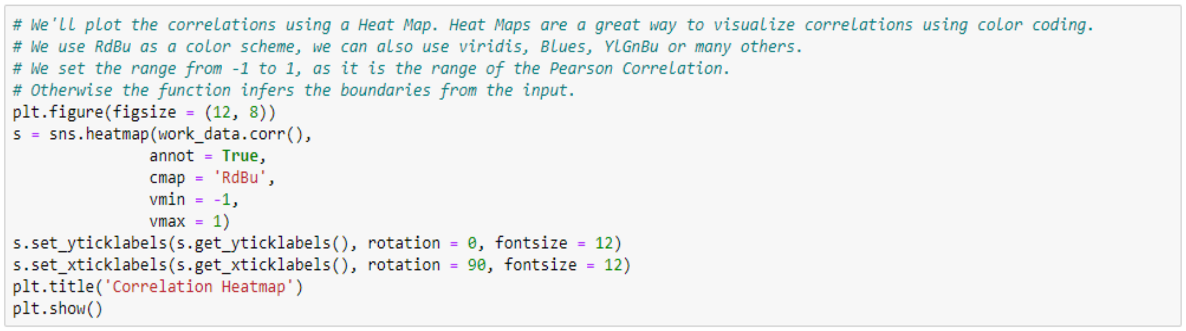


```
work_data.shape
```

```
(100000, 27)
```

Figure 19: Code snippet showing number rows and columns of the engineered dataset

Doing a quick correlation analysis on the working dataset after feature engineering to gauge whether we are still operating on the same dimension of the data as the original dataset is very important.



```
# We'll plot the correlations using a Heat Map. Heat Maps are a great way to visualize correlations using color coding.
# We use RdBu as a color scheme, we can also use viridis, Blues, YlGnBu or many others.
# We set the range from -1 to 1, as it is the range of the Pearson Correlation.
# Otherwise the function infers the boundaries from the input.
plt.figure(figsize = (12, 8))
s = sns.heatmap(work_data.corr(),
               annot = True,
               cmap = 'RdBu',
               vmin = -1,
               vmax = 1)
s.set_yticklabels(s.get_yticklabels(), rotation = 0, fontsize = 12)
s.set_xticklabels(s.get_xticklabels(), rotation = 90, fontsize = 12)
plt.title('Correlation Heatmap')
plt.show()
```

Figure 20: Code snippet to visualize the correlation on engineered data

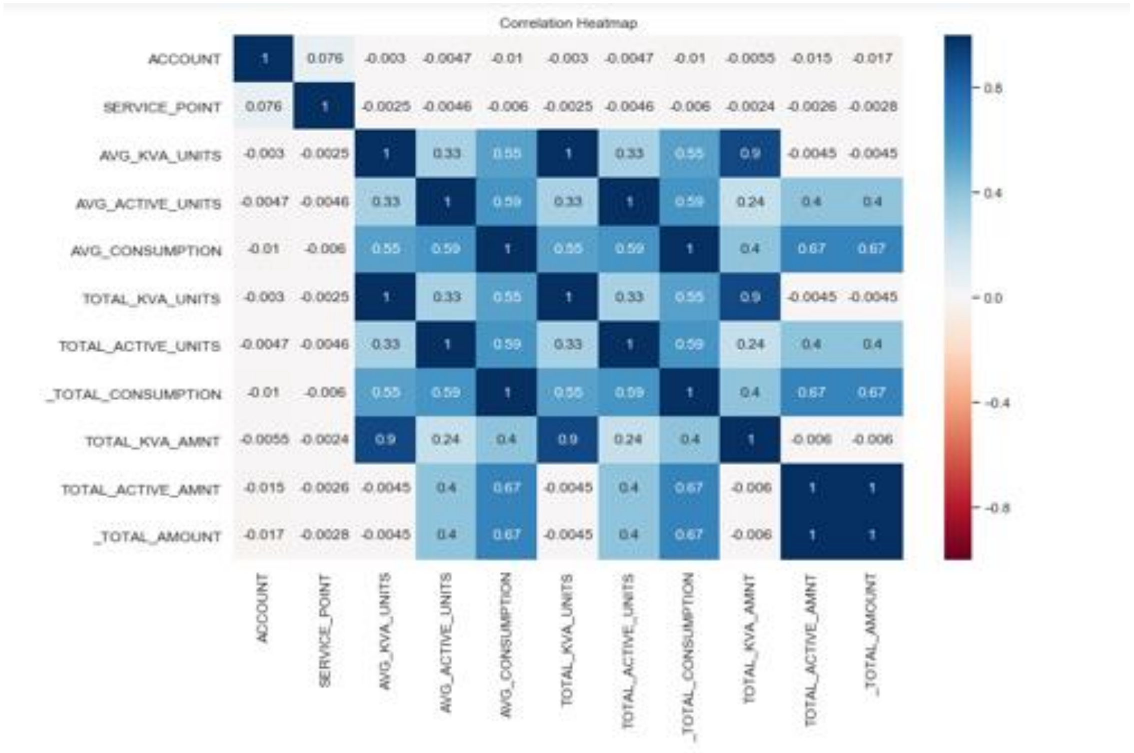
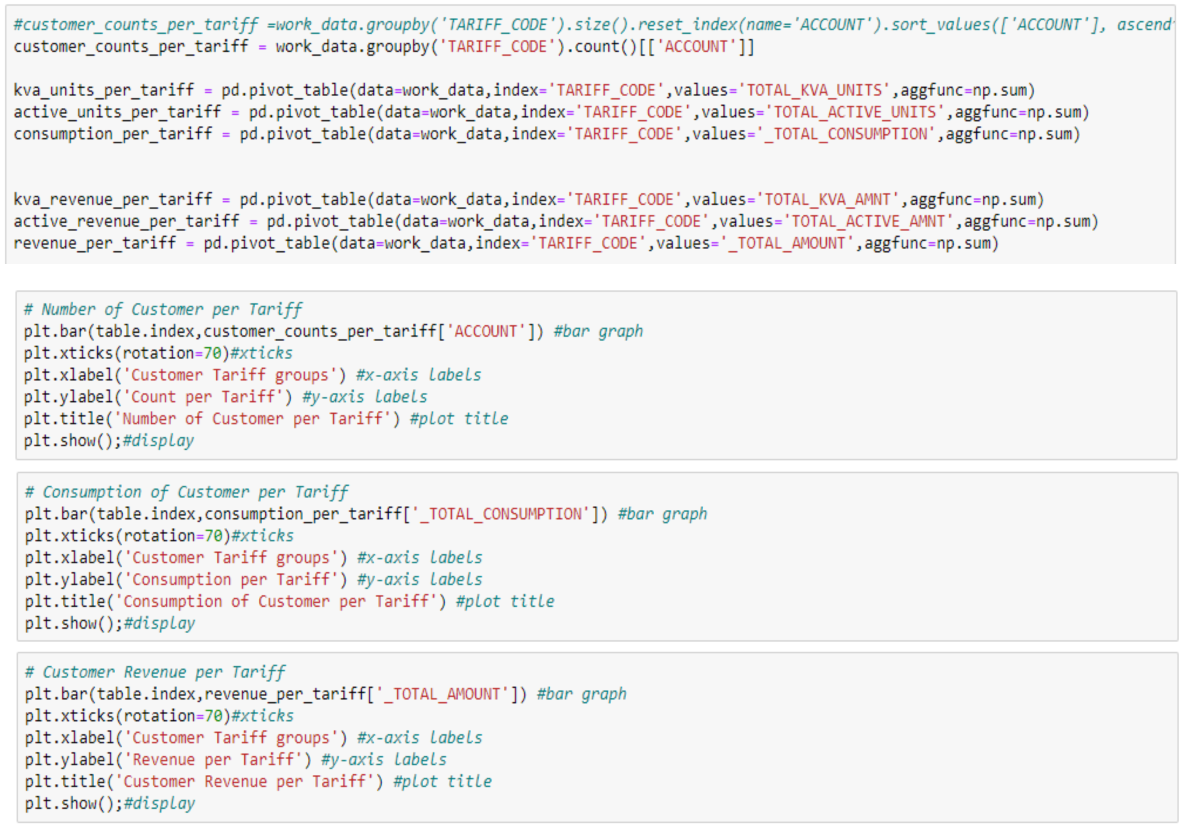


Figure 21: Correlation Heatmap of Engineered data

4.3 Data Analysis and Visualizations

For more exploratory data analysis of the dataset, we can have some plots, which are basically histogram plots for some features in our dataset. We explore the base classification model used by KPLC, which in this case is tariff groupings of the customers. We plot the customer counts, consumption, and revenue per tariff classifications.



```
#customer_counts_per_tariff =work_data.groupby('TARIFF_CODE').size().reset_index(name='ACCOUNT').sort_values(['ACCOUNT'], ascend
customer_counts_per_tariff = work_data.groupby('TARIFF_CODE').count()[['ACCOUNT']]

kva_units_per_tariff = pd.pivot_table(data=work_data,index='TARIFF_CODE',values='TOTAL_KVA_UNITS',aggfunc=np.sum)
active_units_per_tariff = pd.pivot_table(data=work_data,index='TARIFF_CODE',values='TOTAL_ACTIVE_UNITS',aggfunc=np.sum)
consumption_per_tariff = pd.pivot_table(data=work_data,index='TARIFF_CODE',values='_TOTAL_CONSUMPTION',aggfunc=np.sum)


kva_revenue_per_tariff = pd.pivot_table(data=work_data,index='TARIFF_CODE',values='TOTAL_KVA_AMNT',aggfunc=np.sum)
active_revenue_per_tariff = pd.pivot_table(data=work_data,index='TARIFF_CODE',values='TOTAL_ACTIVE_AMNT',aggfunc=np.sum)
revenue_per_tariff = pd.pivot_table(data=work_data,index='TARIFF_CODE',values='_TOTAL_AMOUNT',aggfunc=np.sum)
```

```
# Number of Customer per Tariff
plt.bar(table.index,customer_counts_per_tariff['ACCOUNT']) #bar graph
plt.xticks(rotation=70)#xticks
plt.xlabel('Customer Tariff groups') #x-axis labels
plt.ylabel('Count per Tariff') #y-axis labels
plt.title('Number of Customer per Tariff') #plot title
plt.show();#display
```

```
# Consumption of Customer per Tariff
plt.bar(table.index,consumption_per_tariff['_TOTAL_CONSUMPTION']) #bar graph
plt.xticks(rotation=70)#xticks
plt.xlabel('Customer Tariff groups') #x-axis labels
plt.ylabel('Consumption per Tariff') #y-axis labels
plt.title('Consumption of Customer per Tariff') #plot title
plt.show();#display
```

```
# Customer Revenue per Tariff
plt.bar(table.index,revenue_per_tariff['_TOTAL_AMOUNT']) #bar graph
plt.xticks(rotation=70)#xticks
plt.xlabel('Customer Tariff groups') #x-axis labels
plt.ylabel('Revenue per Tariff') #y-axis labels
plt.title('Customer Revenue per Tariff') #plot title
plt.show();#display
```

Figure 22: Code snippets for histogram plots

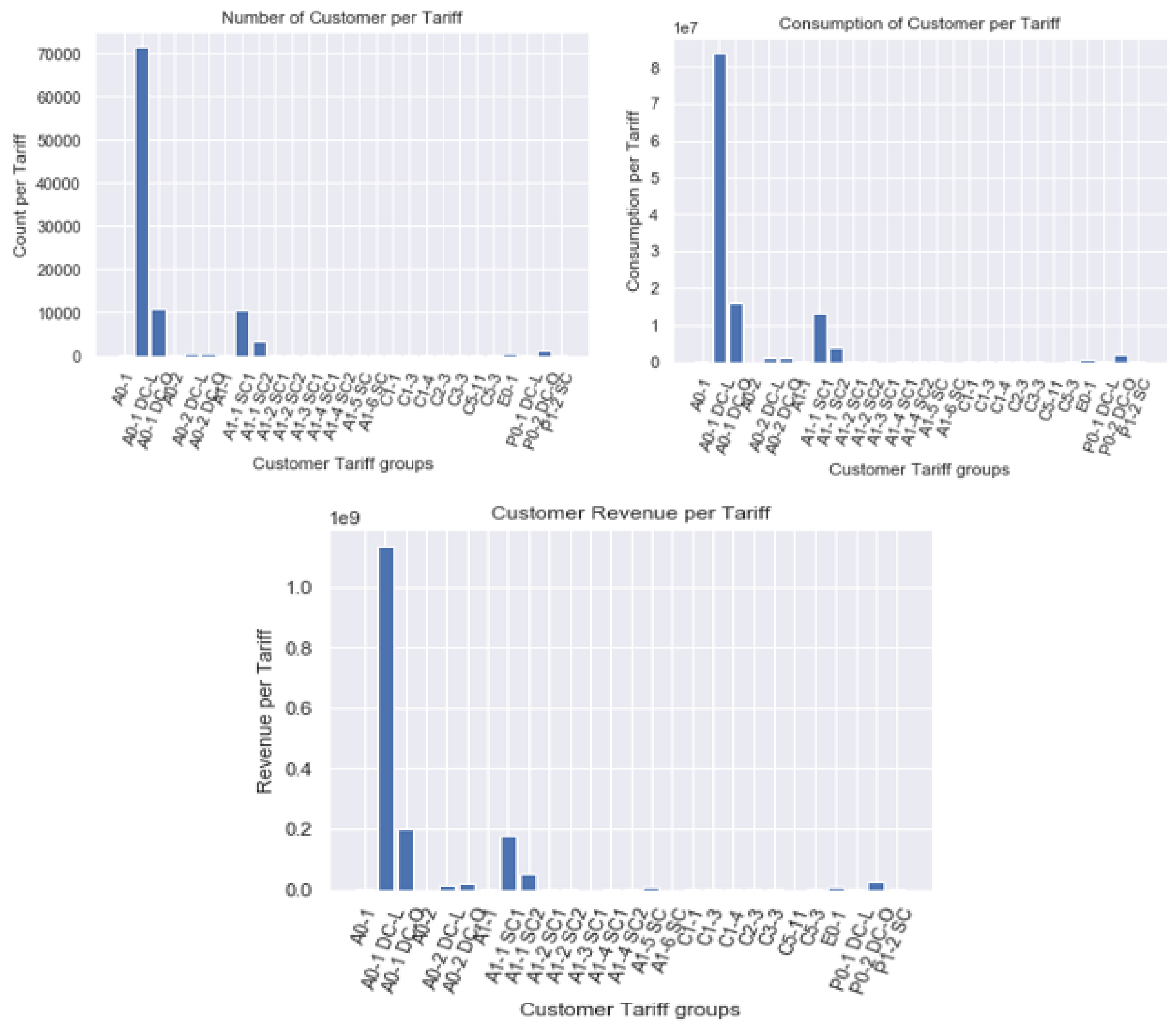
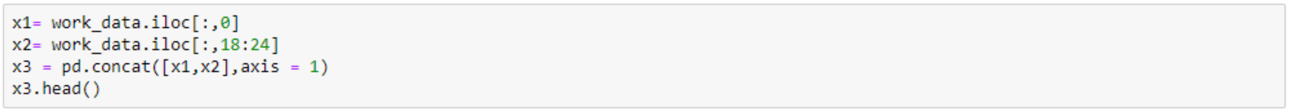


Figure 23: Histogram plots

4.4 Selecting features for Clustering

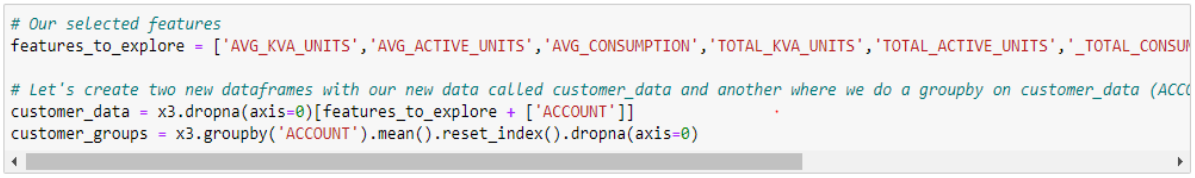
Since we now understand our data much better, we still need to do more in regards to its dimensionality before we can subject it to the clustering process, which is the ultimate goal of the study. However, we have already done feature engineering of our data, and this has greatly reduced the features from 96 to 27. This is still considered a great number to subject to clustering, and to avoid running into dimensionality problems, and we'll consider feature selection. Since we have done the correlation analysis of our dataset, we can apply the results to feature selection. (Chormunge & Jena, 2018). For this study, we'll consider the numerical features we computed during feature engineering coupled with the customers' account numbers for clustering. We apply the pandas method *iloc* to slice our data frame into rows and columns we are interested in. With *iloc*, the first argument indicates the row indices we want to keep from the data frame, while the second argument is the row indices of the columns. In this case, we'll keep all the 100,000 rows(observations), and so we put "**:**" as the first argument. For features, we take the numerical columns we computed and concatenate them with the customer account.



```
x1= work_data.iloc[:,0]
x2= work_data.iloc[:,18:24]
x3 = pd.concat([x1,x2],axis = 1)
x3.head()
```

Figure 24: Code snippet for feature selection

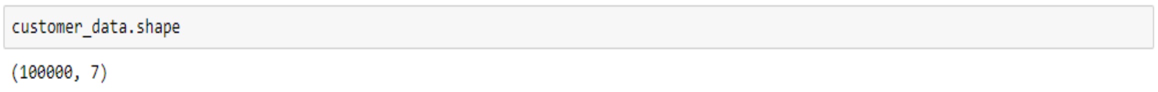
So for clustering, we have selected the features AVG_KVA_UNITS, AVG_ACTIVE_UNITS, AVG_CONSUMPTION, TOTAL_KVA_UNITS, TOTAL_ACTIVE_UNITS, and _TOTAL_CONSUMPTION. We have removed the features TOTAL_KVA_AMNT, TOTAL_ACTIVE_AMNT, and _TOTAL_AMOUNT from clustering and analysis due to their high correlation to the corresponding consumption features and, as such, might not affect our model.



```
# Our selected features
features_to_explore = ['AVG_KVA_UNITS','AVG_ACTIVE_UNITS','AVG_CONSUMPTION','TOTAL_KVA_UNITS','TOTAL_ACTIVE_UNITS','_TOTAL_CONSUM

# Let's create two new dataframes with our new data called customer_data and another where we do a groupby on customer_data (ACCO
customer_data = x3.dropna(axis=0)[features_to_explore + ['ACCOUNT']]
customer_groups = x3.groupby('ACCOUNT').mean().reset_index().dropna(axis=0)
```

Figure 25: Code snippet for data frames with selected features

For clustering, we note that we are clustering customers, and therefore, we need to group our data by ACCOUNT, which in this case represents each customer. We, therefore, create a data frame *customer_groups* meant for this purpose. With the selected features for clustering, the features have been reduced further to 7 from 27. This is now what is going to be subjected to the clustering process.



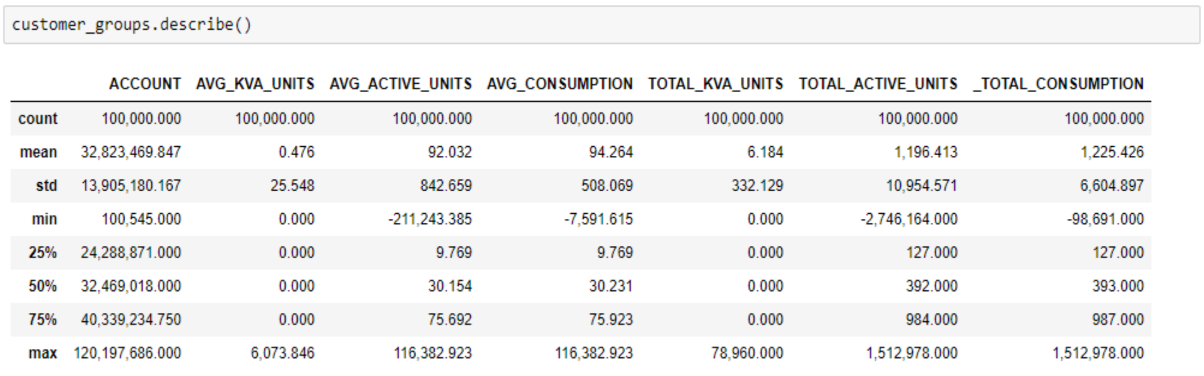
```
customer_data.shape
```

```
(100000, 7)
```

Figure 26: Code snippet and output for a dataset for clustering

4.5 Standardizing the data for Clustering

There is always a big question and controversy around whether to standardize or not to standardize data before applying machine learning algorithms in data science. The ultimate aim of standardization is to reduce the weight of higher numbers and increase that of lower ones. The data should meet assumptions where the variables are not skewed and have the same mean and variance. Standardizing data so that all features have equal weight is important for modeling. (Gal & Rubinfeld, 2019). Otherwise, _TOTAL_CONSUMPTION would be considered much more important than the rest of the features, for instance, in our case. We do not know if this is the case, so we would not like to introduce it to our model. This is what is also referred to as bias. Therefore, for this study, we are going to transform the features in such a way that their values fall within the same numerical range and ensure that the differences between their values are comparable.

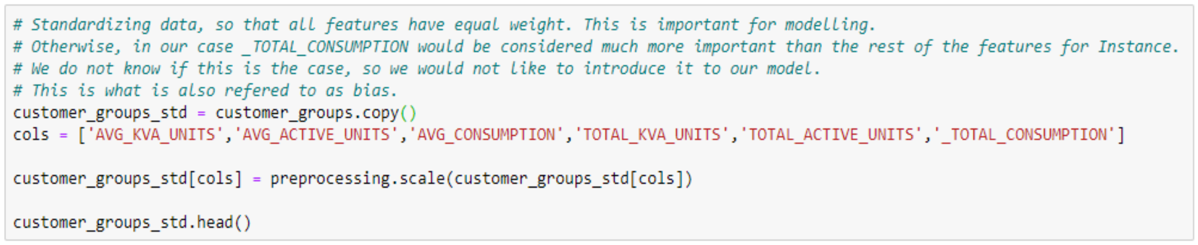


```
customer_groups.describe()
```

	ACCOUNT	AVG_KVA_UNITS	AVG_ACTIVE_UNITS	AVG_CONSUMPTION	TOTAL_KVA_UNITS	TOTAL_ACTIVE_UNITS	_TOTAL_CONSUMPTION
count	100,000.000	100,000.000	100,000.000	100,000.000	100,000.000	100,000.000	100,000.000
mean	32,823,469.847	0.476	92.032	94.264	6.184	1,196.413	1,225.426
std	13,905,180.167	25.548	842.659	508.069	332.129	10,954.571	6,604.897
min	100,545.000	0.000	-211,243.385	-7,591.615	0.000	-2,746,164.000	-98,691.000
25%	24,288,871.000	0.000	9.769	9.769	0.000	127.000	127.000
50%	32,469,018.000	0.000	30.154	30.231	0.000	392.000	393.000
75%	40,339,234.750	0.000	75.692	75.923	0.000	984.000	987.000
max	120,197,686.000	6,073.846	116,382.923	116,382.923	78,960.000	1,512,978.000	1,512,978.000

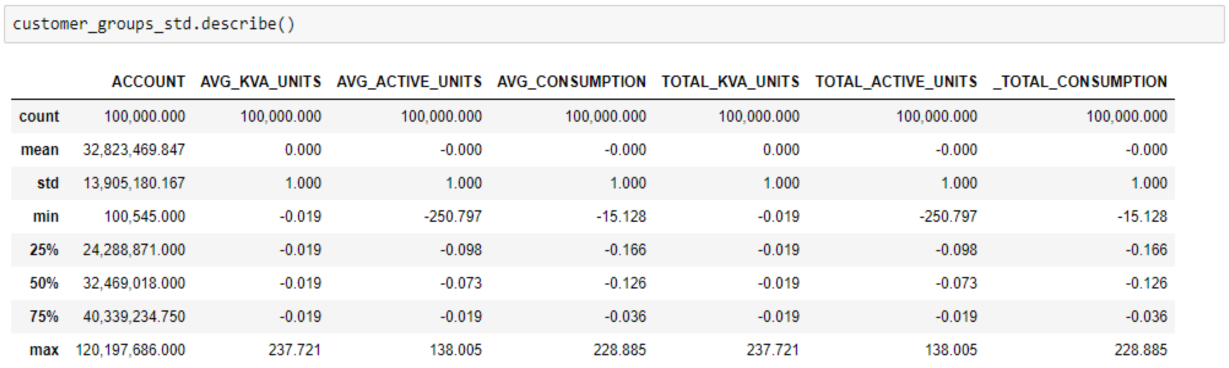
Figure 27: Code snippet and output for unstandardized data

To standardize our data in python, we use the *sklearn* module for *preprocessing* and its *StandardScaler*.



```
# Standardizing data, so that all features have equal weight. This is important for modelling.
# Otherwise, in our case _TOTAL_CONSUMPTION would be considered much more important than the rest of the features for Instance.
# We do not know if this is the case, so we would not like to introduce it to our model.
# This is what is also refered to as bias.
customer_groups_std = customer_groups.copy()
cols = ['AVG_KVA_UNITS','AVG_ACTIVE_UNITS','AVG_CONSUMPTION','TOTAL_KVA_UNITS','TOTAL_ACTIVE_UNITS','_TOTAL_CONSUMPTION']

customer_groups_std[cols] = preprocessing.scale(customer_groups_std[cols])

customer_groups_std.head()
```

Figure 28: Code snippet showing standardization of data



```
customer_groups_std.describe()
```

```
             ACCOUNT  AVG_KVA_UNITS  AVG_ACTIVE_UNITS  AVG_CONSUMPTION  TOTAL_KVA_UNITS  TOTAL_ACTIVE_UNITS  _TOTAL_CONSUMPTION
count    100,000.000    100,000.000       100,000.000      100,000.000      100,000.000         100,000.000         100,000.000
mean  32,823,469.847          0.000            -0.000           -0.000            0.000              -0.000              -0.000
std   13,905,180.167          1.000             1.000            1.000            1.000               1.000               1.000
min      100,545.000         -0.019          -250.797          -15.128           -0.019            -250.797             -15.128
25%   24,288,871.000         -0.019            -0.098           -0.166           -0.019              -0.098              -0.166
50%   32,469,018.000         -0.019            -0.073           -0.126           -0.019              -0.073              -0.126
75%   40,339,234.750         -0.019            -0.019           -0.036           -0.019              -0.019              -0.036
max  120,197,686.000        237.721           138.005          228.885          237.721             138.005             228.885
```

Figure 29: Code snippet and output for standardized data

4.6 Clustering of Customer Data with K-Means.

At this stage of the study, we focus on how we can use the K-means technique to divide our dataset into clusters. We make use of the python *sklearn* library, which contains various clustering functionalities among them, the K-means. We import the *KMeans* package from the *sklearn.cluster* library for use. It is good to take note at this point that K-means doesn't specify the number of clusters K by itself. It only focuses on minimizing the Euclidean norm. Therefore, we should be able to come up with some clusters. To do this, we are going to explore two methods and compare the results to ensure the number of clusters *k* we pick for clustering our data is ideal. We explore the Elbow method and Silhouette Coefficient in determining the ideal number of clusters for our data.

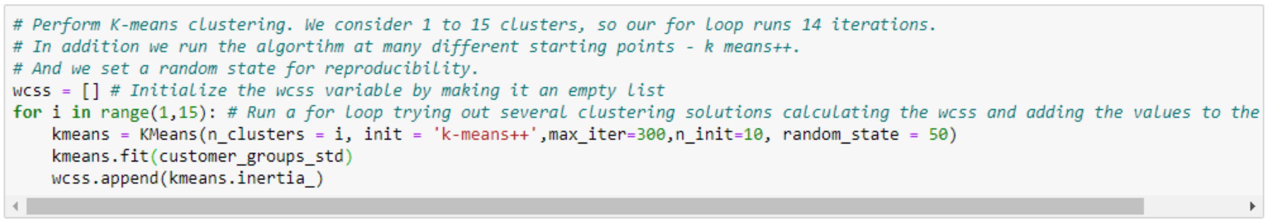
4.6.1 Elbow Method

For the Elbow method, we can start by assuming the value of k = 2 for two clusters and proceed to compute the sum of squared distances between data points inside the cluster. We may also assume the value of k = 3 for three clusters and apply the same technique. We then compare the results in both cases and determine the drastic change. This is exactly what happens in a real situation. Usually, the algorithm runs, say, 10 iterations with 10 different numbers of clusters [k = 1, 2, 3, …, 10]. The within-cluster sum of squares (WCSS) for each of the clustering solutions is calculated for each run. The WCSS is the sum of the variance between the data points in each cluster. It is the measure of the distance separating each data point and the cluster centroid and is taken as the squared difference between the two points. That is how the name within-cluster sum of squares (WCSS) is derived. The mathematical representation of WCSS is given as;

$$WCSS = \sum_{j=1}^{k} \sum_{xi \in Cluster\ j} || Xi \ - \ \bar{X}j \ ||^2,$$

where $\bar{X}j$ is the sample mean of cluster j

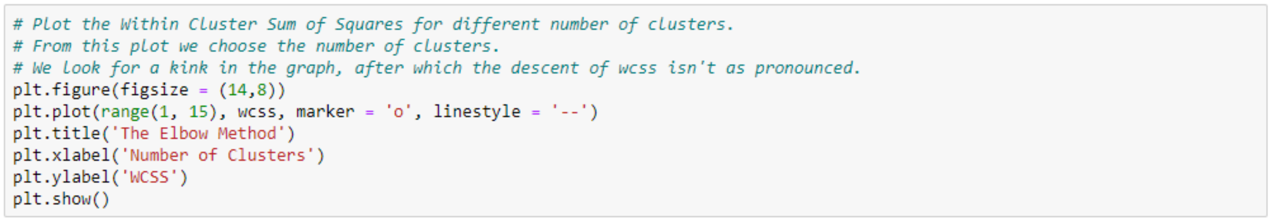
We use the WCSS values in determining the best clustering solution.



```
# Perform K-means clustering. We consider 1 to 15 clusters, so our for loop runs 14 iterations.
# In addition we run the algortihm at many different starting points - k means++.
# And we set a random state for reproducibility.
wcss = [] # Initialize the wcss variable by making it an empty list
for i in range(1,15): # Run a for loop trying out several clustering solutions calculating the wcss and adding the values to the
    kmeans = KMeans(n_clusters = i, init = 'k-means++',max_iter=300,n_init=10, random_state = 50)
    kmeans.fit(customer_groups_std)
    wcss.append(kmeans.inertia_)
```

Figure 308: Code snippet to compute the WCSS

The next step is to plot the calculated values for WCSS against the assumed number of clusters.



```
# Plot the Within Cluster Sum of Squares for different number of clusters.
# From this plot we choose the number of clusters.
# We look for a kink in the graph, after which the descent of wcss isn't as pronounced.
plt.figure(figsize = (14,8))
plt.plot(range(1, 15), wcss, marker = 'o', linestyle = '--')
plt.title('The Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```

Figure 31: Code snippet of WCSS plot

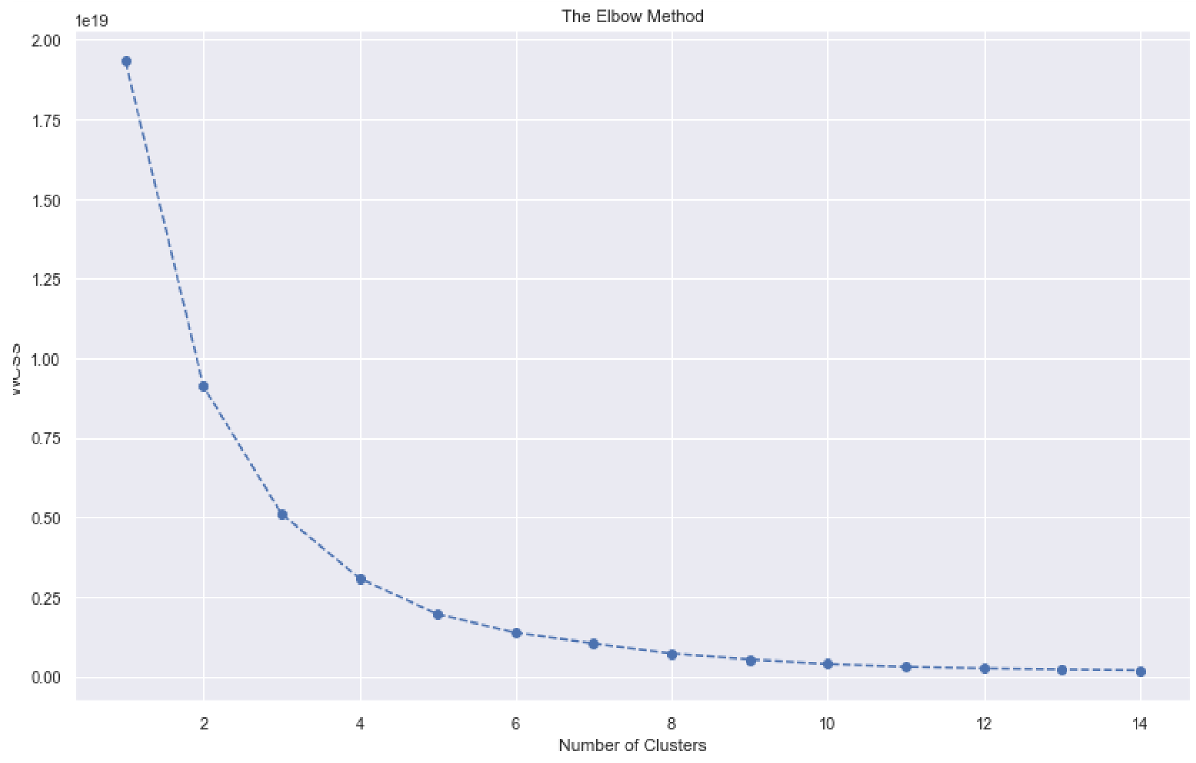


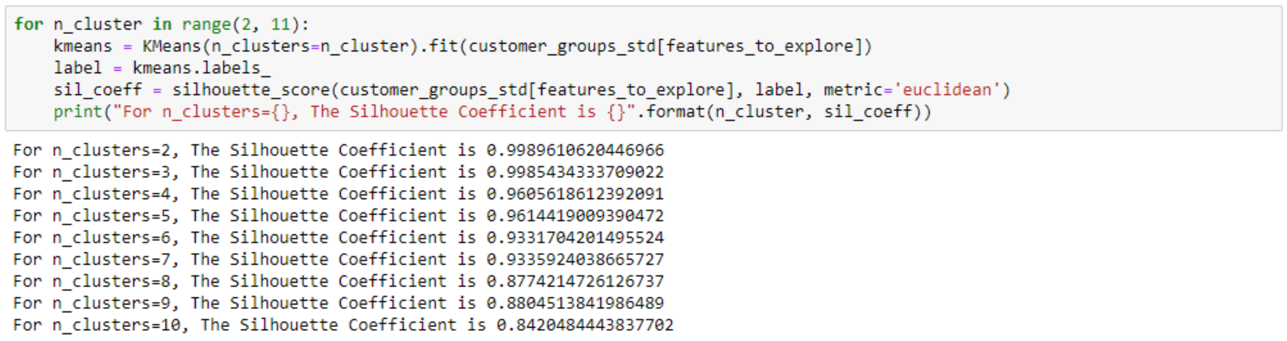
Figure 32: WCSS plot

From the graph above, we can see that the function is steadily decreasing, sometimes sharply declining, and in some instances smoothly. Based on the shape of this curve, we can make some decisions on the optimal number of clusters more precisely by using the elbow method approach. The graph resembles an arm with an elbow, and the aim here is to be able to identify the actual elbow and take the corresponding number as the number of clusters *k*.

In most cases, the portion of the graph below the elbow will be sharply declining, while the portion after the elbow will be characterized by its smooth descent. From the graph, it seems we have some view of the elbow, and in this case, it seems to be at the 3rd mark. When in doubt, it is always good to double-check. To affirm our choice for the number of clusters as k = 3, we further explore the second option, which is the Silhouette Coefficient, and compare the results.

4.6.2 Silhouette Coefficient

Silhouette Coefficients computation in python



```
for n_cluster in range(2, 11):
    kmeans = KMeans(n_clusters=n_cluster).fit(customer_groups_std[features_to_explore])
    label = kmeans.labels_
    sil_coeff = silhouette_score(customer_groups_std[features_to_explore], label, metric='euclidean')
    print("For n_clusters={}, The Silhouette Coefficient is {}".format(n_cluster, sil_coeff))
```

```
For n_clusters=2, The Silhouette Coefficient is 0.9989610620446966
For n_clusters=3, The Silhouette Coefficient is 0.9985434333709022
For n_clusters=4, The Silhouette Coefficient is 0.9605618612392091
For n_clusters=5, The Silhouette Coefficient is 0.9614419009390472
For n_clusters=6, The Silhouette Coefficient is 0.9331704201495524
For n_clusters=7, The Silhouette Coefficient is 0.9335924038665727
For n_clusters=8, The Silhouette Coefficient is 0.8774214726126737
For n_clusters=9, The Silhouette Coefficient is 0.8804513841986489
For n_clusters=10, The Silhouette Coefficient is 0.8420484443837702
```

Figure 33: Silhouette Coefficient computation

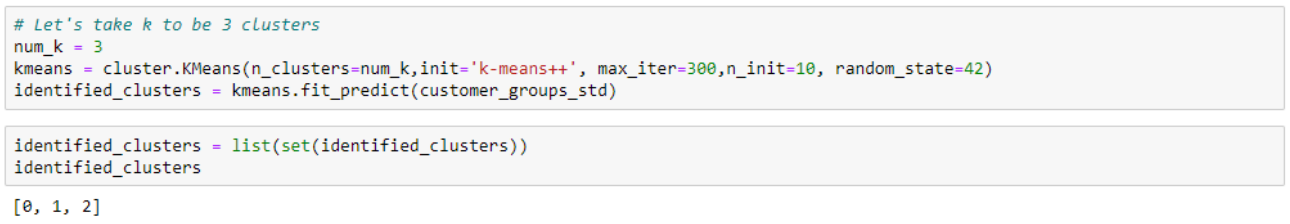
Number of Clusters	Silhouette Coefficient
2	0.9989610620446966
3	0.9985434333709022
4	0.9605618612392091
5	0.9614419009390472
6	0.9331704201495524
7	0.9335924038665727
8	0.8774214726126737
9	0.8804513841986489
10	0.8420484443837702

Table 9: Silhouette Coefficient computation

From the Silhouette Coefficients computed, we have n_clusters = 2, giving the highest score of 0.9989610620446966 and the second highest being n_clusters = 3 with a close score of 0.9985434333709022. Since 2 is too small, and we already discussed the disadvantage of having a small number of clusters, we take 3 as the optimal number of clusters for our model.

4.7 Cluster Results

Now that we have established the number of clusters to use as 3, we perform K-Means clustering and fit the data using the *kmeans.fit()* method on our standardized data.



```
# Let's take k to be 3 clusters
num_k = 3
kmeans = cluster.KMeans(n_clusters=num_k,init='k-means++', max_iter=300,n_init=10, random_state=42)
identified_clusters = kmeans.fit_predict(customer_groups_std)
```

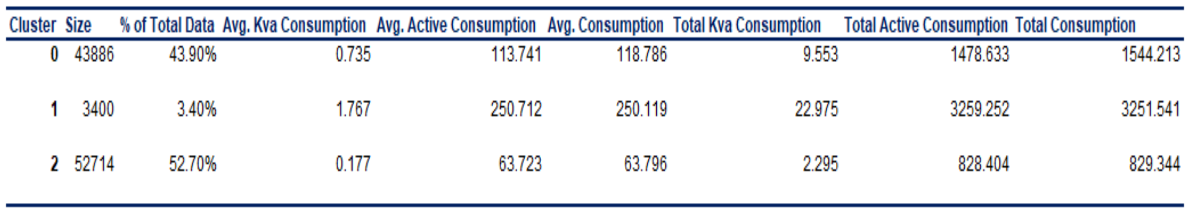
```
identified_clusters = list(set(identified_clusters))
identified_clusters
```

```
[0, 1, 2]
```

Figure 34: Code Snippet showing Identified Clusters

4.8 Cluster Analysis

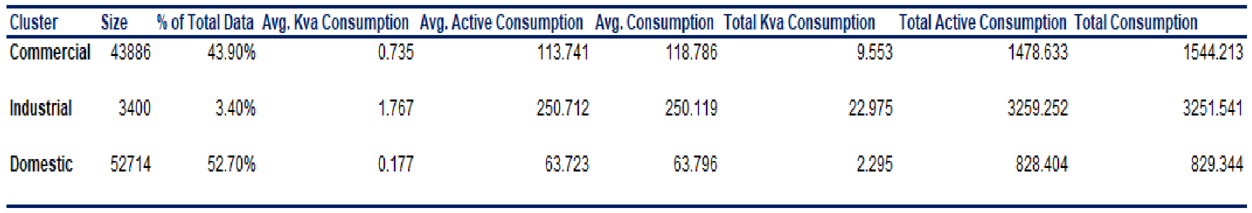
In our cluster solution, we have identified three groups, i.e. [0, 1, 2]. The groups are composed of the customer with different average electricity consumptions. We have cluster 2 (index 1) composed of few customers but with the highest electricity consumption on average. It is followed by cluster 1 (index 0) then cluster 3 (index 2) in descending order. It is crucial to rename our clusters. This enables us to understand the groups and to be able to make different assumptions about them. Even better, other people can easily comprehend what the numbers mean without necessarily having modeling knowledge.



Cluster	Size	% of Total Data	Avg. Kva Consumption	Avg. Active Consumption	Avg. Consumption	Total Kva Consumption	Total Active Consumption	Total Consumption
0	43886	43.90%	0.735	113.741	118.786	9.553	1478.633	1544.213
1	3400	3.40%	1.767	250.712	250.119	22.975	3259.252	3251.541
2	52714	52.70%	0.177	63.723	63.796	2.295	828.404	829.344

Table 10: Identified Clusters

We can name cluster 2 (index 1) as Industrial(Large Power) Customers because they are few and with the highest electricity consumption on average. The group that follows in consumptions is cluster 1 (index 0), and we name it Commercial Customers because of their considerable high Kva consumption. The third and last group, as per our clustering, is cluster 3 (index 2). This group has low average Kva and Active energy consumptions but are the highest in numbers from our sample population. So we name the group Domestic(Standard Power) Customers.



Cluster	Size	% of Total Data	Avg. Kva Consumption	Avg. Active Consumption	Avg. Consumption	Total Kva Consumption	Total Active Consumption	Total Consumption
Commercial	43886	43.90%	0.735	113.741	118.786	9.553	1478.633	1544.213
Industrial	3400	3.40%	1.767	250.712	250.119	22.975	3259.252	3251.541
Domestic	52714	52.70%	0.177	63.723	63.796	2.295	828.404	829.344

Table 11: Cluster Labeling

We have successfully named our clusters. It is good to note that clusters can be named in many different ways but what is important is the use of a clear, descriptive phrase that can be used as a reference.

As the last step, we would like to visualize our clustered data. We start by plotting the raw data rather than the standardized data. The only difference is that this time we have the clusters. We use the data frame *df_clust_kmeans* and plot each individual point and its corresponding cluster. We map the cluster names to the clusters using a python function *.map()* with names as labels. We then create a scatter plot of _TOTAL_CONSUMPTION against ACCOUNT.

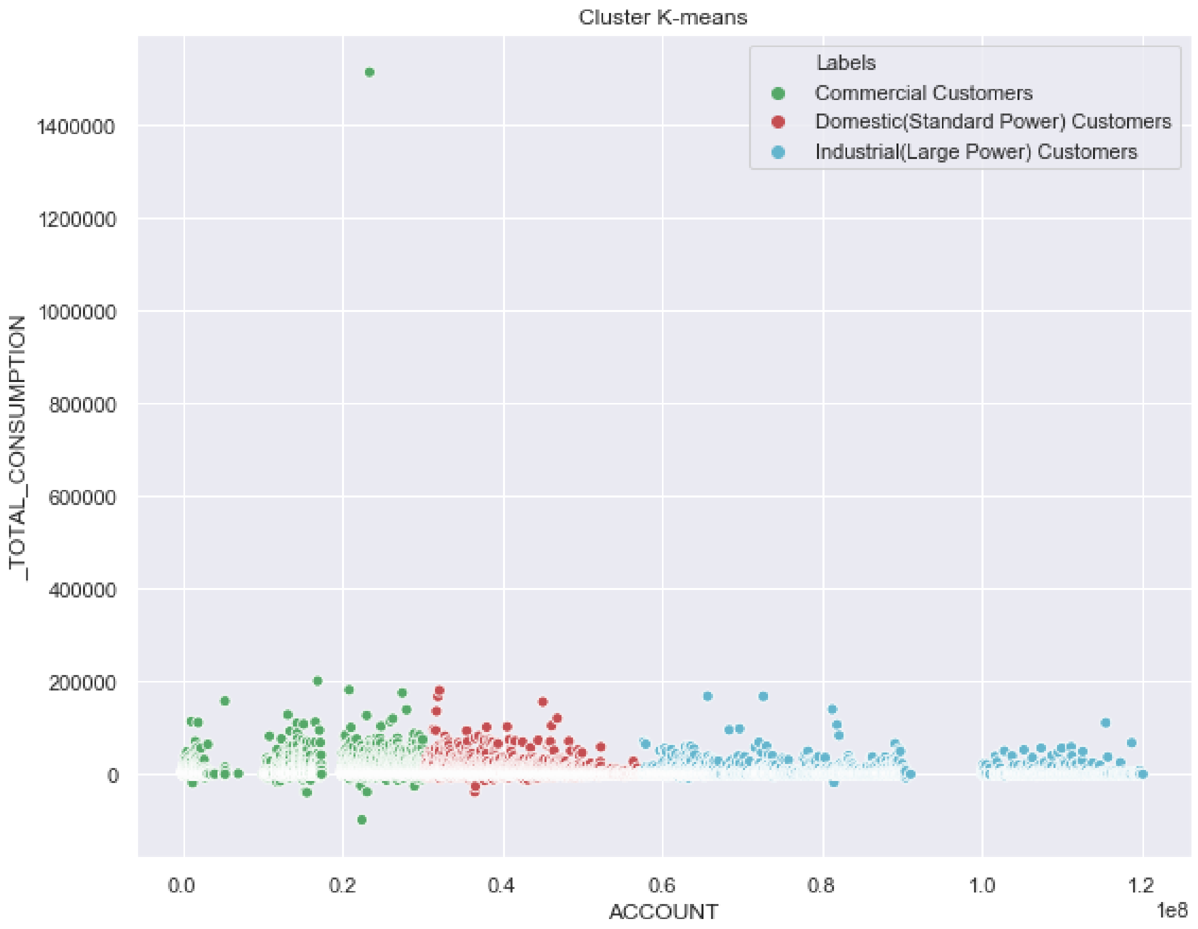


Figure 35: Scatter plot for Total Consumptions for Customers

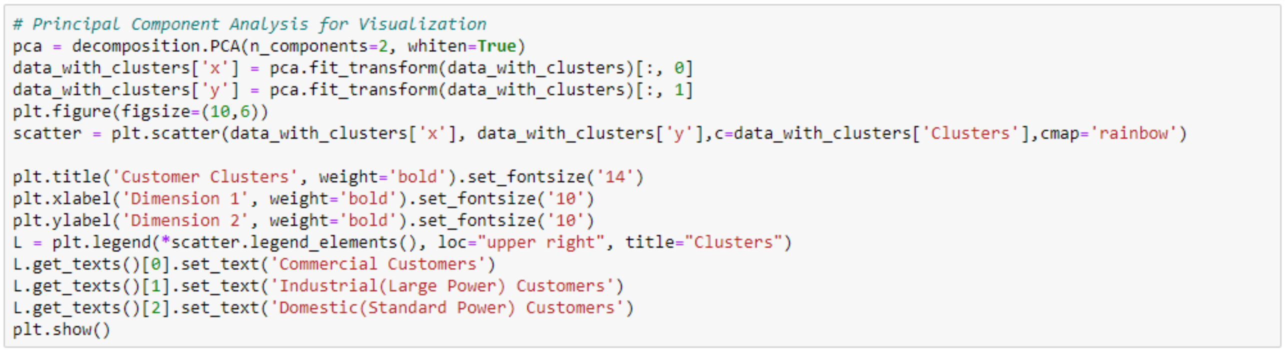
We can see from the graph above that the clusters are grouped together. In this case, it is a bit difficult to get more insight just by looking at the plot. We can, however, conclude that k-means clustering did a decent job of separating our data into clusters. However, the result is far from perfect. So we look further into how we can get even more out of it. Since our data is multi-dimensional, we look at the option of combining K-Means with Principal Component Analysis (PCA) to try and see if it can yield better results.

4.9 K-Means Clustering with PCA

Many times when working with large datasets with many dimensions, things may get too computationally expensive and confusing to keep track of. However, most of the time, when we look at the data itself, we see variables that are strongly correlated and hence possibly redundant, for instance, total consumption and the total amount we have in this study. In most data science problems, there is always the need to reduce the dimensionality of the data and at the same time retain its information. This is what Principal Component Analysis (PCA) is helping to achieve and hence a dimensionality reduction method.

Principal Component Analysis is an algorithm that compresses the dataset's dimension from a higher to a lower dimensionality. It does this based on eigenvectors of the variance in the dataset. PCA is useful in saving processing time through data compression as well as visualization of high dimensional data in 2 or 3 dimensions. It is very helpful in enabling cluster analysis.

For this study, our original dataset had 96 features which translate to 96 dimensions. Through feature engineering, it was reduced to 27 dimensions which were finally reduced to 7 dimensions by feature selection and standardization for clustering. When we visualize the clusters applying PCA to K-means in 2 dimensions, we end up with more pronounced clusters and better results, as shown below.



```
# Principal Component Analysis for Visualization
pca = decomposition.PCA(n_components=2, whiten=True)
data_with_clusters['x'] = pca.fit_transform(data_with_clusters)[:, 0]
data_with_clusters['y'] = pca.fit_transform(data_with_clusters)[:, 1]
plt.figure(figsize=(10,6))
scatter = plt.scatter(data_with_clusters['x'], data_with_clusters['y'],c=data_with_clusters['Clusters'],cmap='rainbow')

plt.title('Customer Clusters', weight='bold').set_fontsize('14')
plt.xlabel('Dimension 1', weight='bold').set_fontsize('10')
plt.ylabel('Dimension 2', weight='bold').set_fontsize('10')
L = plt.legend(*scatter.legend_elements(), loc="upper right", title="Clusters")
L.get_texts()[0].set_text('Commercial Customers')
L.get_texts()[1].set_text('Industrial(Large Power) Customers')
L.get_texts()[2].set_text('Domestic(Standard Power) Customers')
plt.show()
```

Figure 36: Code snippet for Principal Component Analysis for Visualization

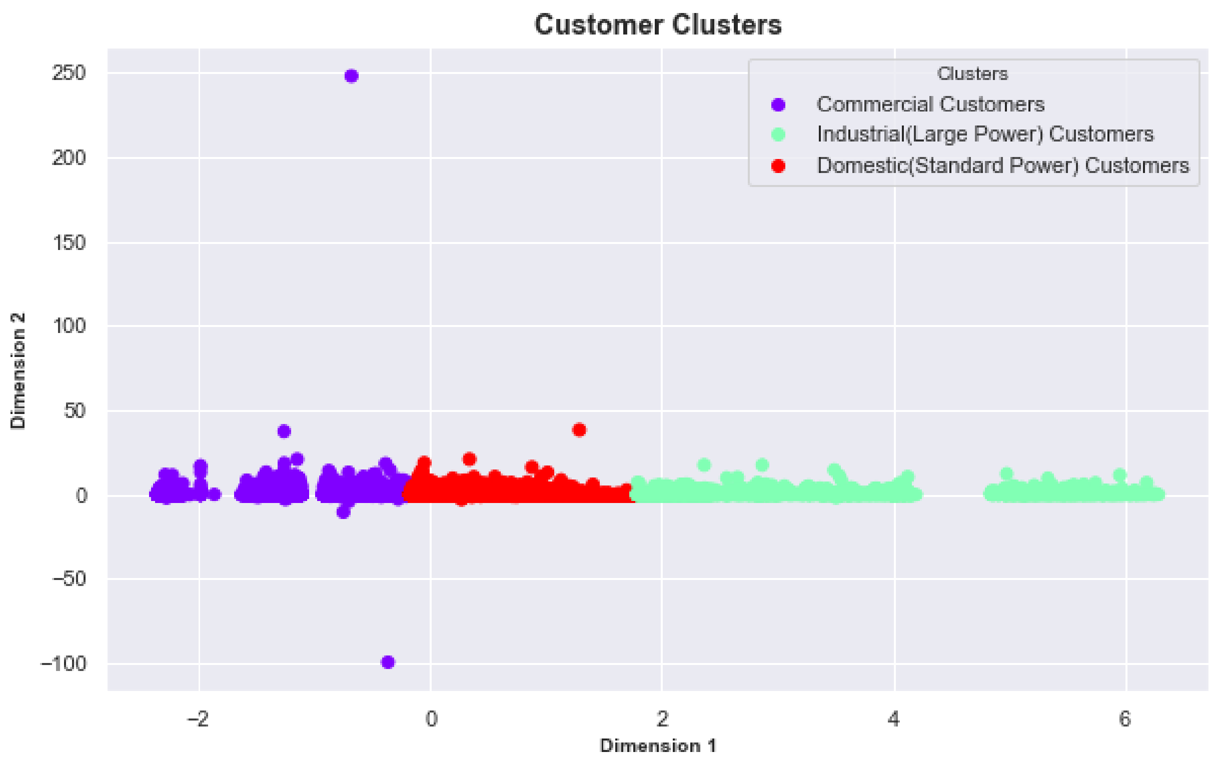


Figure 37: A scatter plot of the identified clusters

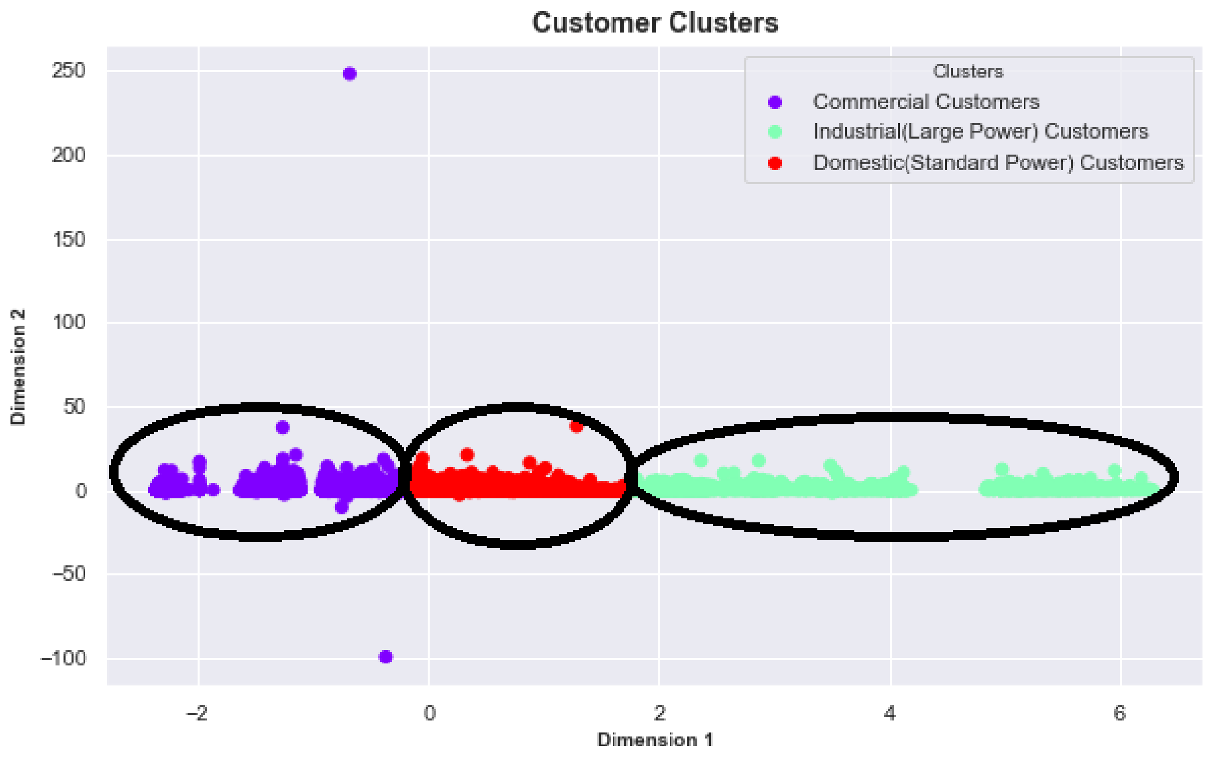


Figure 38: Spherical indications of the identified clusters in a scatter plot

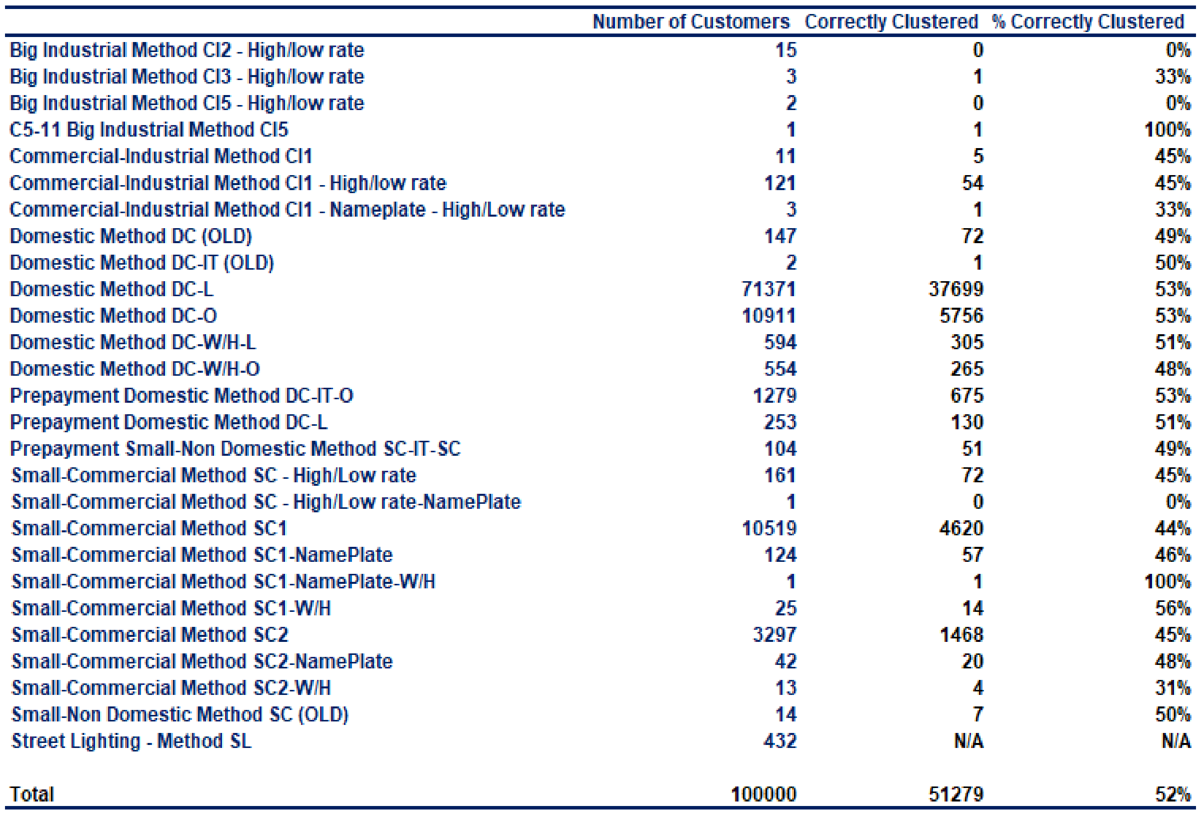
4.10 Mapping the Clusters with Actual Customer Tariff Classification.

After clustering, we need to map our clusters to the actual customer tariff categories by the distribution company and see how correct or wrong our clustering model performed against the actual world data. The mapping objective in the case of this study is to establish the number of customers in each tariff category in each of the identified clusters. To do this, we introduce the kmeans cluster labels for our clustering model into the actual dataset and tag the feature which we would like to use in mapping, which in the case of this study is the customer tariff classification.

	Commercial Customers	Industrial(Large Power) Customers	Domestic(Standard Power) Customers	Number of Customers
Big Industrial Method CI2 - High/low rate	3	0	12	15
Big Industrial Method CI3 - High/low rate	1	1	1	3
Big Industrial Method CI5 - High/low rate	0	0	2	2
C5-11 Big Industrial Method CI5	0	1	0	1
Commercial-Industrial Method CI1	5	0	6	11
Commercial-Industrial Method CI1 - High/low rate	54	9	58	121
Commercial-Industrial Method CI1 - Nameplate - High/Low rate	1	0	2	3
Domestic Method DC (OLD)	71	4	72	147
Domestic Method DC-IT (OLD)	1	0	1	2
Domestic Method DC-L	31242	2430	37699	71371
Domestic Method DC-O	4785	370	5756	10911
Domestic Method DC-W/H-L	273	16	305	594
Domestic Method DC-W/H-O	262	27	265	554
Prepayment Domestic Method DC-IT-O	566	38	675	1279
Prepayment Domestic Method DC-L	113	10	130	253
Prepayment Small-Non Domestic Method SC-IT-SC	51	7	46	104
Small-Commercial Method SC - High/Low rate	72	7	82	161
Small-Commercial Method SC - High/Low rate-NamePlate	0	0	1	1
Small-Commercial Method SC1	4620	334	5565	10519
Small-Commercial Method SC1-NamePlate	57	3	64	124
Small-Commercial Method SC1-NamePlate-W/H	1	0	0	1
Small-Commercial Method SC1-W/H	14	0	11	25
Small-Commercial Method SC2	1468	129	1700	3297
Small-Commercial Method SC2-NamePlate	20	2	20	42
Small-Commercial Method SC2-W/H	4	1	8	13
Small-Non Domestic Method SC (OLD)	7	1	6	14
Street Lighting - Method SL	195	10	227	432
Total	43886	3400	52714	100000

Table 12: Clusters and Tariff Classification Mapping

We can see from the mapping of clusters with the actual tariff classification that the model clustered 52% of the customers correctly as per the actual tariff classification by the electricity distributor. However, we can not clearly say without a doubt that the cases in 48% have been wrongly classified because this research has used the customer consumption data and be able to predict customer's cluster. The electricity consumption patterns by the customer depend on their consumption habits which in this case could have greatly contributed to the 48% being clustered and mapped to different tariff categories with their actual grouping by the electricity distributor. On the other hand, the power utility company would attract more revenue if the customers they have grouped as domestic and being billed on domestic tariff rates but clustered as industrial by the model are billed on industrial tariff rates.



	Number of Customers	Correctly Clustered	% Correctly Clustered
Big Industrial Method CI2 - High/low rate	15	0	0%
Big Industrial Method CI3 - High/low rate	3	1	33%
Big Industrial Method CI5 - High/low rate	2	0	0%
C5-11 Big Industrial Method CI5	1	1	100%
Commercial-Industrial Method CI1	11	5	45%
Commercial-Industrial Method CI1 - High/low rate	121	54	45%
Commercial-Industrial Method CI1 - Nameplate - High/Low rate	3	1	33%
Domestic Method DC (OLD)	147	72	49%
Domestic Method DC-IT (OLD)	2	1	50%
Domestic Method DC-L	71371	37699	53%
Domestic Method DC-O	10911	5756	53%
Domestic Method DC-W/H-L	594	305	51%
Domestic Method DC-W/H-O	554	265	48%
Prepayment Domestic Method DC-IT-O	1279	675	53%
Prepayment Domestic Method DC-L	253	130	51%
Prepayment Small-Non Domestic Method SC-IT-SC	104	51	49%
Small-Commercial Method SC - High/Low rate	161	72	45%
Small-Commercial Method SC - High/Low rate-NamePlate	1	0	0%
Small-Commercial Method SC1	10519	4620	44%
Small-Commercial Method SC1-NamePlate	124	57	46%
Small-Commercial Method SC1-NamePlate-W/H	1	1	100%
Small-Commercial Method SC1-W/H	25	14	56%
Small-Commercial Method SC2	3297	1468	45%
Small-Commercial Method SC2-NamePlate	42	20	48%
Small-Commercial Method SC2-W/H	13	4	31%
Small-Non Domestic Method SC (OLD)	14	7	50%
Street Lighting - Method SL	432	N/A	N/A
Total	100000	51279	52%

Table 13: Number of Customers correctly clustered per Tariff Classification

4.11 Clustering Performance Evaluation

It is good to take note that evaluating any clustering algorithm performance is not just simply taking the counts of errors or the recall and precision as it is in supervised machine learning classification algorithms. In fact, the evaluation metric is not supposed to take into consideration the sheer values of the group labels but should instead define the segmentation of the observations similar to some ground truth of a set of classes by satisfying some assumptions that members in the same group are more similar than members of a different group according to some specified similarity index.

4.11.1 Rand Index

Rand Index, also referred to as adjusted or unadjusted Rand Index, applies where we have some prior information of the actual grouping indicated as *labels_true* and the grouping by the clustering algorithm for the same data points indicated as *labels_pred*. The *labels_true* is used as a reference while the *labels_pred* are cluster labels being evaluated. Rand index function weighs the similarity of the two assignments while ignoring the permutations.

In this study, the *labels_true* for the cluster performance evaluations was taken to be the actual tariff classification by the electricity company, KPLC. If X is the ground truth clusters and is our clustering, then unadjusted Rand Index (RI) is denoted by;

$$\mathbf{RI} = \frac{\boldsymbol{a} + \boldsymbol{b}}{\boldsymbol{X}_{\mathbf{2}}^{\boldsymbol{n\ samples}}}$$

Where;

a is the number of times a pair of data points appear in the same cluster in X and in the same cluster K.

b is the number of times a pair of data points appear in different clusters in X and in different clusters in K,

$\boldsymbol{X}_{\mathbf{2}}^{\boldsymbol{n\ samples}}$ is the total number of possible pairs in a dataset. Whether the computation is done on ordered pairs or unordered pairs doesn't matter here as long as there is consistency.

In python, we use the inbuilt package *metrics.rand_score(labels_true, labels_pred)* to compute the Rand Index. Since the Rand index does not guarantee a value close to 0.0 for a random sample labeling, the adjusted Rand index (ARI) counter this effect and gives a baseline by discounting the expected Random Index (E[RI]). The Adjusted Rand Index (ARI) is denoted by;

$$\mathbf{ARI} = \frac{\mathbf{RI} - \mathbf{E[RI]}}{\mathbf{max(RI)} - \mathbf{E[RI]}}$$

The Adjusted Rand Index (ARI) is automatically computed in python with the help of package *metrics.adjusted_rand_score(labels_true, labels_pred).* As with other clustering metrics, the Adjusted Random score ranges from -1 to 1, with perfect labeling scoring 1.0. The unadjusted Rand Index ranges from 0 to 1. In the case of this study, the computed Adjusted Random Index is **0. 00035539444563387814**.

4.11.2 Silhouette Coefficient

Silhouette Coefficient is one of the ideal evaluation techniques in K-means clustering because it gives a higher value in convex clusters than other clustering techniques like DBSCAN. It is mostly applied where the ground truth labels are not known, and the evaluation is performed by the model itself (Rousseeuw, 1987). The computation is based on the mean distance between a data point and all other points in the same cluster; also, intra-cluster distance (***a)*** and the mean distance between a data point and all other points in the other nearest cluster also referred to as the nearest-cluster distance (***b***). The Silhouette Coefficient **(*s*)** for a given population is thus denoted as;

$$\boldsymbol{s} = \frac{(\boldsymbol{b}-\boldsymbol{a})}{\boldsymbol{max}(\boldsymbol{a},\boldsymbol{b})}$$

In python, the package *sklearn.metrics.silhouette_score* helps with the computation of the Silhouette Coefficient. The score is between -1 and 1, with 1 indicating highly dense clusters being the best and -1 indicating incorrect clustering being the worst-case scenario. The Silhouette Coefficient value of zero(0) or near zero(0) means overlapping clusters. A negative Silhouette score indicates that a data point has been placed in the wrong cluster in a case where a different cluster is deemed more similar (Rousseeuw, 1987). For this study, the computed Silhouette score is **0.550524009530382**. This being on the better half shows that our clustering was correct, not overlapping, and with clustering results not very far from perfect.

4.11.3 Fowlkes-Mallows scores

Fowlkes-Mallows Index is an evaluation technique used in determining the similarity between two clusters obtained from a clustering model. It is also a metric used in measuring confusion matrices. It is mostly applied between two hierarchical clusterings or a clustering technique and a benchmark classification, and in the case of this study is applied based on the latter consideration.

The Fowlkes-Mallows Index (FMI) is mathematically represented as;

$$\boldsymbol{FMI} = \frac{\boldsymbol{TP}}{\sqrt{(\boldsymbol{TP}+\boldsymbol{FP})(\boldsymbol{TP}+\boldsymbol{FN})}}$$

Where;

TP is True Positive and represents the paired number of data points that are in the same clusters in both the true labels and the predicted labels,

FP is False Positive and represents the paired number of data points that are in the same clusters in the true labels and not in the predicted labels,

FN False Negative and represents the paired number of data points that are in the same clusters in the predicted labels and not in the true labels.

If we rewrite the above expression, we can see that Fowlkes-Mallows Index (FMI) is the geometric mean of the product of precision and recall, as shown below.

$$\boldsymbol{FMI} = \sqrt{\left(\frac{\boldsymbol{TP}}{\boldsymbol{TP}+\boldsymbol{FP}}\right).\left(\frac{\boldsymbol{TP}}{\boldsymbol{TP}+\boldsymbol{FN}}\right)}$$

$$\boldsymbol{FMI} = \sqrt{\boldsymbol{Precision}\ X\ \boldsymbol{Recall}}$$

In python, the package *sklearn.metrics.fowlkes_mallows_score* is used to determine the Fowlkes-Mallows index and applies where there is prior information of the ground truth assignments in a dataset. The score ranges from zero(0) to one(1), and a higher value of the Fowlkes–Mallows index indicates greater similarity between the clusters and their baseline classifications. In the case of this study, the computed Fowlkes-Mallows index is **0.501875342731891**. Considering the score range from 0 to 1, this is half and implies a balanced similarity index between our clusters and the actual classifications.

4.11.4 Homogeneity, Completeness and V-measure

Homogeneity and Completeness in clustering also apply where there is some prior information of the ground truth cluster assignments indicated as *labels_true* and the consideration of the actual clustering algorithm assignments of the same data points indicated as *labels_pred.* With the two variables, it is possible to use conditional entropy analysis to define some intuitive metric.

According to (Hirschberg & Rosenberg, 2007), **homogeneity** is a scenario whereby each cluster contains only data points of a single class in a dataset. On the other hand, **completeness** is a situation where all data points of a certain class in a dataset are grouped or put in a single cluster. The two concepts have been turned to scores as completeness_score and homogeneity_score, and both have a lower bound of 0.0 and upper bound of 1.0. The higher the score in both cases, the better the performance of the model. In python the scores are computed by the in-built *sklearn* packages *metrics.homogeneity_score(labels_true,labels_pred)* and *metrics.completeness_score(labels_true, labels_pred)* for homogeneity and completeness respectively.

The harmonic mean of the two scores is called the V-measure and is given by the functions' formula as below.

$$\boldsymbol{v} = \frac{(\mathbf{1} + \boldsymbol{\beta}) \times \boldsymbol{homogeneity} \times \boldsymbol{completeness}}{(\boldsymbol{\beta} \times \boldsymbol{homogeneity} + \boldsymbol{completeness})}$$

It is important to note that the value for beta ($\boldsymbol{\beta}$) defaults to 1.0. Using the value for beta ($\boldsymbol{\beta}$) less than 1 attributes more weight to homogeneity, and using a value greater than 1 attributes more weight to completeness. The V-measure can also be computed by the v_measure_score in python package *metrics.v_measure_score(labels_true, labels_pred).* In python, homogeneity, completeness, and V-measure values can all be calculated using the package *homogeneity_completeness_v_measure* as *metrics.homogeneity_completeness_v_measure(labels_true, labels_pred).*

It is also important to note that ***v_measure_score*** is symmetrical and can as well be used to measure the relationship of two separate clusters emerging from the same dataset. However, this does not apply to homogeneity_score and completeness_score as the two are connected by the relationship *homogeneity_score(a, b) == completeness_score(b, a).*

Score Metric	Value
homogeneity	0.00027523270655764576
completeness	0.0003518353126657282
v-measure	0.00030885512384278945

Table 14: Score metric for homogeneity, completeness, and v-measure

According to homogeneity, completeness and v-measure scoring metrics, the scores indicate that our clustering assignments are not good since it is neither homogeneous nor complete. This is true and very clear from the mapping as not all the data points from of a single tariff classification were assigned to a single cluster by the model to satisfy the completeness criteria, and on the other hand, not only those customers that are members of a single tariff classification were assigned to a single cluster to satisfy our homogeneity criteria.

4.12 Customer Cluster Prediction

As we may recall from the objectives, this research study is aimed at determining and applying data-driven clustering techniques in machine learning to customer consumption data and be able to predict customer clusters. The design and analysis phase of this study has been able to determine the clusters of customers based on the K-Means clustering technique. So our next task and the big question is that, are we able to use the identified clusters from customers' consumption data to determine the cluster of a customer with some given sample consumption pattern? To answer this question, we take the sample test data and apply our model to it see the predictions. In the case of this study, we provide our model with a test sample data with features ACCOUNT, AVG_KVA_UNITS, AVG_ACTIVE_UNITS, AVG_CONSUMPTION, TOTAL_KVA_UNITS, TOTAL_ACTIVE_UNITS, and _TOTAL_CONSUMPTION as they were used in our model. This accurately places the customer in the correct cluster.

CHAPTER 5: CONCLUSION AND RECOMMENDATIONS

5.1. Conclusion

According to most literature reviews, clustering is one of the most prevalent machine learning technique for exploratory analysis (Bolton & Krzanowski, 2003). K-Means, one of the machine clustering technique, was used in this project and to obtain the results reported in this study. Clustering and mapping of customers with the actual tariff classification by the clustering model placed correctly clustered and mapped customers at 52% based on their electricity consumption patterns. However, depending on the customers' consumption habits, 48% were clustered and mapped differently from the actual classification by the electricity distribution company. This revealed the importance and the need to adopt the dynamic data-driven clustering by the electricity companies in grouping their customers for accurate billing and to minimize losses in revenue. The results can also be used for different purposes; for example, the Power Utility inspectorate and fraud detection team can use these results to conduct targeted field inspections on customers clusters with deviations from the actual tariff classification by the electricity company. Based on the results of this research study, it can be concluded that the K-Means clustering technique can be reliably used to categorize electricity consumers based on their consumption patterns for accurate and fair billing aimed at maximizing revenue for electricity distribution companies.

5.2. Research Limitation

The main limitation of this research study was the lack of a standard dataset. The data used for this research has not been used for any other research or referenced before. This is a limitation because research problems use standard datasets that are well-vetted by experts (Marfunin, 1995). A standard dataset is a representation of the actual occurrence of a real-world problem. Another challenge experienced during this research study is that it was not easy to evaluate the model's performance with 100% confidence. This is because the actual tariff classification of customers, which was taken as ground truth label in some cases were marked as "OLD," and this would mean that the assumption that customers in such a group are more similar than customers in another group not marked as "OLD" may not have been satisfied.

5.3 Achievements

The main objective of this research was to study the customer classification techniques currently in use by the electricity distribution companies and compare them with the dynamic data-driven clustering techniques in machine learning. To achieve this, the K-Means clustering model was developed and applied to the electricity consumption data from Kenya Power and Electricity Supply Company (KPLC). The objectives to determine the dynamic data-driven clustering technique, develop an algorithm, and apply it to customers' electricity consumption data were thus achieved. The result of the K-Means clustering model was then mapped to the actual tariff classification by the electricity distribution company, thus achieving the final objective of comparing the classification methods in use and the dynamic machine learning clustering techniques and the impact on revenue.

5.4 Recommendations and Future Work

This research study is a contribution to the body of knowledge and successfully applied K-Means, which is a machine learning clustering technique to the electricity consumption data to cluster, map, and compare the customer clusters with the actual tariff classification by the electricity company. This study presented only one machine learning clustering technique. This research and the results obtained, therefore, provides the basis for further research using some other machine learning techniques and algorithms other than the K-Means. The research recommends future work on standard electricity consumption data using Hierarchical Agglomerative and DBSCAN clustering techniques, which are Hierarchy-based approach and Density-based approach, respectively, as compared to the conducted Partition-based approach. Also, for mapping the model's clusters with the actual tariff classification, the research recommends for further studies the use of Self Organizing Maps (SOM).

REFERENCES

Bolton, R. J., & Krzanowski, W. J. (2003). Projection pursuit clustering for exploratory data analysis. Journal of Computational and Graphical Statistics, 12(1), 121-142.

Chicco, G., Napoli, R., & Piglione, F. (2006). Comparisons among clustering techniques for electricity customer classification. *IEEE Transactions on Power Systems*, *21*(2), 933–940. https://doi.org/10.1109/TPWRS.2006.873122

Chormunge, S., & Jena, S. (2018). Correlation-based feature selection with clustering for high-dimensional data. *Journal of Electrical Systems and Information Technology*, *5*(3), 542–549. https://doi.org/10.1016/j.jesit.2017.06.004

Codecs—Codec registry and base classes—Python 3.9.5 documentation. (n.d.). Retrieved 27th May 2021, from https://docs.python.org/3/library/codecs.html#standard-encodings

Dåderman, A., & Rosander, S. (2018). *Evaluating Frameworks for Implementing Machine Learning in Signal Processing: A Comparative Study of CRISP-DM, SEMMA, and KDD*. http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-235408

Database PL/SQL Packages and Types Reference. (n.d.). Retrieved 30th June, 2021, from https://docs.oracle.com/cd/B19306_01/appdev.102/b14258/d_random.htm#ARPLS040

Depuru, S. S. S. R. (2012). *Modeling, Detection, and Prevention of Electricity Theft for Enhanced Performance and Security of Power Grid*.

Flora, D. B., LaBrish, C., & Chalmers, R. P. (2012). Old and new ideas for data screening and assumption testing for exploratory and confirmatory factor analysis. Frontiers in Psychology, 3, 55.

Gal, M. S., & Rubinfeld, D. L. (2019). Data Standardization. *New York University Law Review*, *94*, 737.

Gantt chart—HandWiki. (n.d.). Retrieved 13th June, 2021, from https://handwiki.org/wiki/Gantt_chart

Glauner, P. O., Boechat, A., Dolberg, L., State, R., Bettinger, F., Rangoni, Y., & Duarte, D. (2017). Large-Scale Detection of Non-Technical Losses in Imbalanced Data Sets. *ArXiv:1602.08350 [Cs]*. http://arxiv.org/abs/1602.08350

Hirschberg, J. B., & Rosenberg, A. (2007). *V-Measure: A conditional entropy-based external cluster evaluation*. https://doi.org/10.7916/D80V8N84

Jensen, K. (2012). *English: A diagram showing the relationship between the different phases of CRISP-DM and illustrates the recursive nature of a data mining project.* Own work based on:ftp://public.dhe.ibm.com/software/analytics/spss/documentation/modeler/18.0/en/ModelerCRISPDM.pdf (Figure 1). https://commons.wikimedia.org/wiki/File:CRISP-DM_Process_Diagram.png#file

Madhulatha, T. S. (2012). An Overview on Clustering Methods. *ArXiv:1205.1117 [Cs]*. http://arxiv.org/abs/1205.1117

Marfunin, A. (1995). Methods and instrumentations =. Berlin: Springer.

Mutanen, A., Ruska, M., Repo, S., & Jarventausta, P. (2011). Customer Classification and Load Profiling Method for Distribution Systems. *IEEE Transactions on Power Delivery*, *26*(3), 1755–1763. https://doi.org/10.1109/TPWRD.2011.2142198

Ozawa, A., Furusato, R., & Yoshida, Y. (2016). Determining the relationship between a household's lifestyle and its electricity consumption in Japan by analyzing measured electric load profiles. *Energy and Buildings*, *119*, 200–210. https://doi.org/10.1016/j.enbuild.2016.03.047

Panwar, P., & Gopal, G. (2016). Image Segmentation using K-means clustering and Thresholding. *Image*, *3*.

Parkyn, N. D. (2019, 30th October). *Realizing OpenCalc using new and emerging computing technology and patterns*. SNAME Maritime Convention. https://onepetro.org/SNAMESMC/proceedings/SMC19/3-SMC19/D033S004R002/3542

Perkel, J. M. (2018). Why Jupyter is data scientists' computational notebook of choice. *Nature*, *563*(7732), 145–147.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, *20*, 53–65. https://doi.org/10.1016/0377-0427(87)90125-7

Thurber, M. (2020, 15th May). A Holistic Framework for Managing Data Analytics Projects. *Elder Research*. https://www.elderresearch.com/blog/a-holistic-framework-for-managing-data-analytics-projects/

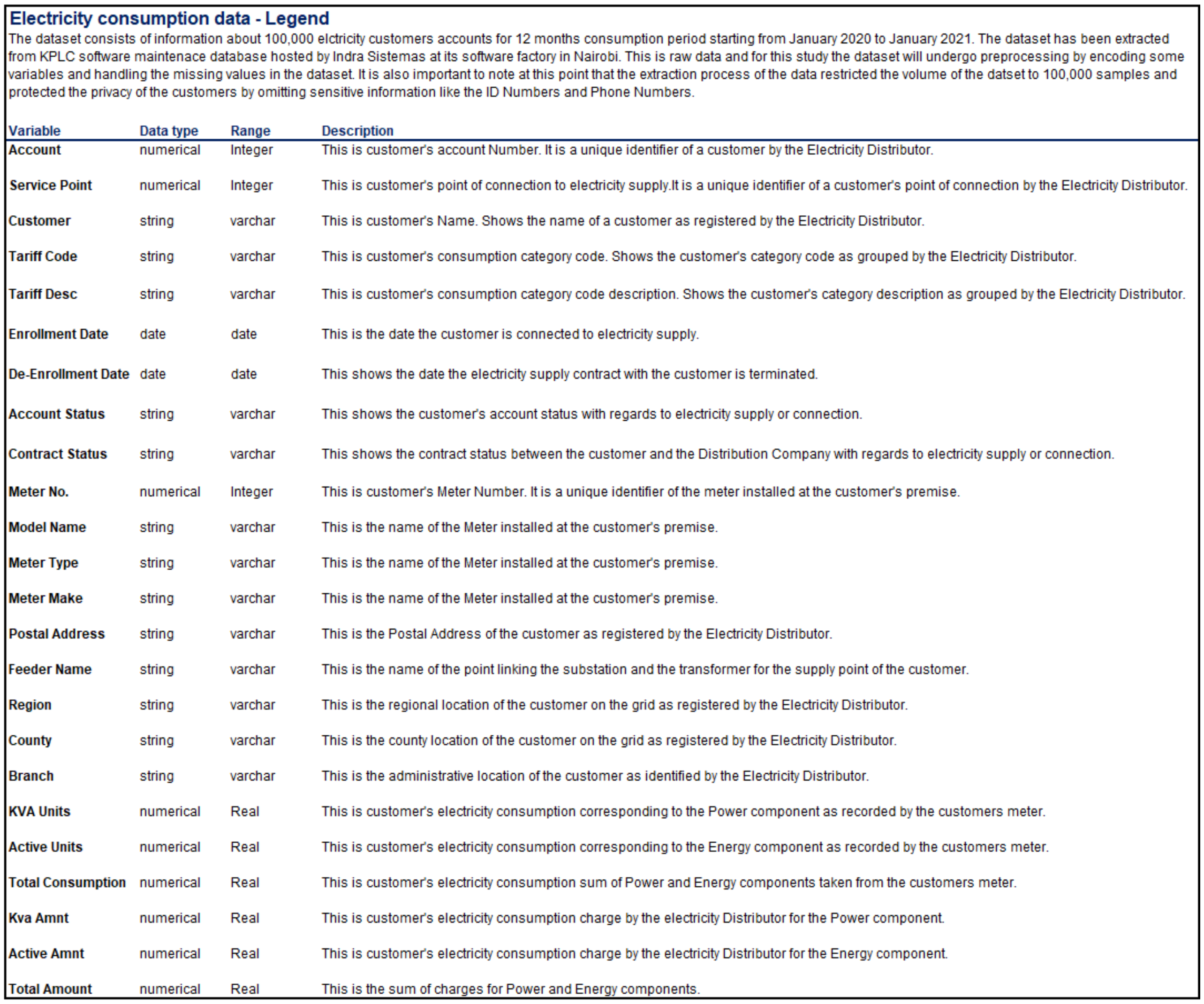
Tureczek, A., Nielsen, P. S., & Madsen, H. (2018). Electricity Consumption Clustering Using Smart Meter Data. *Energies*, *11*(4), 859. https://doi.org/10.3390/en11040859

Wirth, R., & Hipp, J. (2000). *Crisp-DM: Towards a Standard Process Model for Data Mining*. *1*, 11.

Xu, R., & Wunsch, D. (2005). Survey of Clustering Algorithms. *Neural Networks, IEEE Transactions On*, *16*, 645–678. https://doi.org/10.1109/TNN.2005.845141

APPENDICES

APPENDIX 1 – Electricity Consumption Data Legend. A data dictionary describing the features of the dataset.



Electricity consumption data - Legend

The dataset consists of information about 100,000 elctricity customers accounts for 12 months consumption period starting from January 2020 to January 2021. The dataset has been extracted from KPLC software maintenace database hosted by Indra Sistemas at its software factory in Nairobi. This is raw data and for this study the dataset will undergo preprocessing by encoding some variables and handling the missing values in the dataset. It is also important to note at this point that the extraction process of the data restricted the volume of the datset to 100,000 samples and protected the privacy of the customers by omitting sensitive information like the ID Numbers and Phone Numbers.

Variable	Data type	Range	Description
Account	numerical	Integer	This is customer's account Number. It is a unique identifier of a customer by the Electricity Distributor.
Service Point	numerical	Integer	This is customer's point of connection to electricity supply.It is a unique identifier of a customer's point of connection by the Electricity Distributor.
Customer	string	varchar	This is customer's Name. Shows the name of a customer as registered by the Electricity Distributor.
Tariff Code	string	varchar	This is customer's consumption category code. Shows the customer's category code as grouped by the Electricity Distributor.
Tariff Desc	string	varchar	This is customer's consumption category code description. Shows the customer's category description as grouped by the Electricity Distributor.
Enrollment Date	date	date	This is the date the customer is connected to electricity supply.
De-Enrollment Date	date	date	This shows the date the electricity supply contract with the customer is terminated.
Account Status	string	varchar	This shows the customer's account status with regards to electricity supply or connection.
Contract Status	string	varchar	This shows the contract status between the customer and the Distribution Company with regards to electricity supply or connection.
Meter No.	numerical	Integer	This is customer's Meter Number. It is a unique identifier of the meter installed at the customer's premise.
Model Name	string	varchar	This is the name of the Meter installed at the customer's premise.
Meter Type	string	varchar	This is the name of the Meter installed at the customer's premise.
Meter Make	string	varchar	This is the name of the Meter installed at the customer's premise.
Postal Address	string	varchar	This is the Postal Address of the customer as registered by the Electricity Distributor.
Feeder Name	string	varchar	This is the name of the point linking the substation and the transformer for the supply point of the customer.
Region	string	varchar	This is the regional location of the customer on the grid as registered by the Electricity Distributor.
County	string	varchar	This is the county location of the customer on the grid as registered by the Electricity Distributor.
Branch	string	varchar	This is the administrative location of the customer as identified by the Electricity Distributor.
KVA Units	numerical	Real	This is customer's electricity consumption corresponding to the Power component as recorded by the customers meter.
Active Units	numerical	Real	This is customer's electricity consumption corresponding to the Energy component as recorded by the customers meter.
Total Consumption	numerical	Real	This is customer's electricity consumption sum of Power and Energy components taken from the customers meter.
Kva Amnt	numerical	Real	This is customer's electricity consumption charge by the electricity Distributor for the Power component.
Active Amnt	numerical	Real	This is customer's electricity consumption charge by the electricity Distributor for the Energy component.
Total Amount	numerical	Real	This is the sum of charges for Power and Energy components.

APPENDIX 2 – Version control link and Code Snippet.

Github link: https://github.com/godfreykotieno/Clustering-Electricity-Consumption-Data-with-K-Means

```
#!/usr/bin/env python
# coding: utf-8

# ## Godfrey Okoth Otieno
# ## MSc. Computational Intelligence
# ### P52/33454/2019
# ## Supervisor: Dr. Wanjiku Nganga
#

# ### 1. Import the relevant libraries

import numpy as np # Linear algebra
import pandas as pd # Data processing, CSV file I/O (e.g. pd.read_csv)

#These are the visualization libraries. Matplotlib is standard and is what most people use.
#Seaborn works on top of matplotlib
import matplotlib. pyplot as plt # Plotting
import seaborn as sns # Visualization library
import warnings
sns.set() # Set the style of all graphs to seaborn one.This overrides the default matplotlib look with
the seaborn one

warnings.filterwarnings('ignore') # Turn off warnings
pd.options.display.float_format = "{:,.3f}".format # Format float values to 3 decimal places in a
dataframe

# Load clustering libraries
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler #For standardizing features. We'll use the
StandardScaler module.
from sklearn.preprocessing import LabelEncoder #For Encoding categorical variables.
import sklearn.metrics as metrics
from sklearn.metrics import silhouette_score, homogeneity_score #For computing the Silhouette
Coefficient.
from sklearn import cluster, decomposition
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

# ### 2. Load the data

# Enforce/align parser (UTF-8) encoding to the file's or handle invalid character
# dataURL = "../THESIS/KPLC_DATA/KPLC_Electricity_consumption_Data.csv"
# raw_data = pd.read_csv(dataURL,encoding='ISO-8859-1', index_col = 0)
raw_data = pd.read_csv('KPLC_Electricity_consumption_Data.csv',encoding='ISO-8859-1')

# ### 3. Data Exploration & Description
raw_data.head() # Display few rows and columns of the data frame
raw_data.describe() # Display statistical information for the numerical columns of the dataset
```

```
# Get some base information on our dataset
#print(data.shape)
print("Rows    : = ",raw_data.shape[0])
print("Columns : = ",raw_data.shape[1])
print("Features : \n ",raw_data.columns.tolist())
print("\nMissing Values :",raw_data.isnull().sum().values.sum())
print ("\nWhere the Missing Values are located :\n", raw_data.isnull().sum())
print("\nUnique Values  :\n",raw_data.nunique())

raw_data.dtypes



# ### 4. Correlation Analysis
# Create our Correlation Matrix on raw_data
raw_data.corr()
# Visualize: Generate Correlation plot or Heatmap
# Generate a mask for the upper triangle
mask = np.zeros_like(raw_data.corr(), dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
# Set up the matplotlib figure
fig, ax = plt.subplots(figsize=(12, 8))
# Generate a custom diverging colormap
cmap = sns.diverging_palette(10, 240, n=9)
# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(raw_data.corr(), mask=mask,cmap=cmap,  vmin=-1, vmax=1, center=0,
square=True, linewidths=.5, cbar_kws={"shrink": .5})
ax.set_xticklabels(ax.get_xticklabels(), rotation=45, horizontalalignment='right');
# Add title
ax.set_title("Correlation Triangle", fontsize=16)
plt.show()

# ### 5. Feature Engineering
work_data = raw_data.copy() # Create a working copy from the raw_data.
# Meter Numbers are being displayed as float. To correct this convert the METER_NO field as
below
work_data['METER_NO'] = work_data['METER_NO'].astype(str).replace('\.0', '', regex=True)
pd.get_dummies(work_data, columns=["TARIFF_CODE"]).head()
# Reduce the number of columns. We do this by computing the averages of the columns
KVA_UNITS,KVA_AMNT,ACTIVE_UNITS,ACTIVE_AMNT
# TOTAL_CONSUMPTION and TOTAL_AMOUNT
# Where column values are null(NaN) we replace with zero(0)
COL_KVA_UNITS = work_data.loc[:,work_data.columns.str.startswith('KVA_UNITS')].fillna(0)
work_data['AVG_KVA_UNITS']  = COL_KVA_UNITS.mean(axis=1)

COL_ACTIVE_UNITS =
work_data.loc[:,work_data.columns.str.startswith('ACTIVE_UNITS')].fillna(0)
work_data['AVG_ACTIVE_UNITS']  = COL_ACTIVE_UNITS.mean(axis=1)
```

```
COL_TOTAL_CONSUMPTION =
work_data.loc[:,work_data.columns.str.startswith('TOTAL_CONSUMPTION')].fillna(0)
work_data['AVG_CONSUMPTION']  = COL_TOTAL_CONSUMPTION.mean(axis=1)

COL_KVA_UNITS = work_data.loc[:,work_data.columns.str.startswith('KVA_UNITS')].fillna(0)
work_data['TOTAL_KVA_UNITS']  = COL_KVA_UNITS.sum(axis=1)

COL_ACTIVE_UNITS =
work_data.loc[:,work_data.columns.str.startswith('ACTIVE_UNITS')].fillna(0)
work_data['TOTAL_ACTIVE_UNITS']  = COL_ACTIVE_UNITS.sum(axis=1)

COL_TOTAL_CONSUMPTION =
work_data.loc[:,work_data.columns.str.startswith('TOTAL_CONSUMPTION')].fillna(0)
work_data['_TOTAL_CONSUMPTION']  = COL_TOTAL_CONSUMPTION.sum(axis=1)

COL_KVA_AMNT = work_data.loc[:,work_data.columns.str.startswith('KVA_AMNT')].fillna(0)
work_data['TOTAL_KVA_AMNT']  = COL_KVA_AMNT.sum(axis=1)

COL_ACTIVE_AMNT =
work_data.loc[:,work_data.columns.str.startswith('ACTIVE_AMNT')].fillna(0)
work_data['TOTAL_ACTIVE_AMNT']  = COL_ACTIVE_AMNT.sum(axis=1)

COL_TOTAL_AMOUNT =
work_data.loc[:,work_data.columns.str.startswith('TOTAL_AMOUNT')].fillna(0)
work_data['_TOTAL_AMOUNT']  = COL_TOTAL_AMOUNT.sum(axis=1)

work_data = work_data.loc[:, ~work_data.columns.str.startswith('KVA_UNITS')]
work_data = work_data.loc[:, ~work_data.columns.str.startswith('KVA_AMNT')]
work_data = work_data.loc[:, ~work_data.columns.str.startswith('ACTIVE_UNITS')]
work_data = work_data.loc[:, ~work_data.columns.str.startswith('ACTIVE_AMNT')]
work_data = work_data.loc[:, ~work_data.columns.str.startswith('TOTAL_CONSUMPTION')]
work_data = work_data.loc[:, ~work_data.columns.str.startswith('TOTAL_AMOUNT')]

work_data.shape
# Create our Correlation Matrix on work_data
work_data.corr()

# We'll plot the correlations using a Heat Map. Heat Maps are a great way to visualize correlations
using color coding.
# We use RdBu as a color scheme, we can also use viridis, Blues, YlGnBu or many others.
# We set the range from -1 to 1, as it is the range of the Pearson Correlation.
# Otherwise the function infers the boundaries from the input.
plt.figure(figsize = (12, 8))
s = sns.heatmap(work_data.corr(),
            annot = True,
            cmap = 'RdBu',
            vmin = -1,
            vmax = 1)
s.set_yticklabels(s.get_yticklabels(), rotation = 0, fontsize = 12)
s.set_xticklabels(s.get_xticklabels(), rotation = 90, fontsize = 12)
plt.title('Correlation Heatmap')
plt.show()


# ### 6. Plot the data
#customer_counts_per_tariff
=work_data.groupby('TARIFF_CODE').size().reset_index(name='ACCOUNT').sort_values(['ACC
OUNT'], ascending=True)
customer_counts_per_tariff = work_data.groupby('TARIFF_CODE').count()[['ACCOUNT']]

kva_units_per_tariff =
pd.pivot_table(data=work_data,index='TARIFF_CODE',values='TOTAL_KVA_UNITS',aggfunc=n
p.sum)
active_units_per_tariff =
pd.pivot_table(data=work_data,index='TARIFF_CODE',values='TOTAL_ACTIVE_UNITS',aggfun
c=np.sum)
consumption_per_tariff =
pd.pivot_table(data=work_data,index='TARIFF_CODE',values='_TOTAL_CONSUMPTION',aggfu
nc=np.sum)


kva_revenue_per_tariff =
pd.pivot_table(data=work_data,index='TARIFF_CODE',values='TOTAL_KVA_AMNT',aggfunc=n
p.sum)
active_revenue_per_tariff =
pd.pivot_table(data=work_data,index='TARIFF_CODE',values='TOTAL_ACTIVE_AMNT',aggfunc
=np.sum)
revenue_per_tariff =
pd.pivot_table(data=work_data,index='TARIFF_CODE',values='_TOTAL_AMOUNT',aggfunc=np.
sum)

table =
pd.concat([customer_counts_per_tariff,kva_units_per_tariff,active_units_per_tariff,consumption_p
er_tariff,
              kva_revenue_per_tariff,active_revenue_per_tariff,revenue_per_tariff],axis = 1)
table

# Number of Customer per Tariff
plt.bar(table.index,customer_counts_per_tariff['ACCOUNT']) #bar graph
plt.xticks(rotation=70)#xticks
plt.xlabel('Customer Tariff groups') #x-axis labels
plt.ylabel('Count per Tariff') #y-axis labels
plt.title('Number of Customer per Tariff') #plot title
plt.show();#display
```

```
# Consumption of Customer per Tariff
plt.bar(table.index,consumption_per_tariff['_TOTAL_CONSUMPTION']) #bar graph
plt.xticks(rotation=70)#xticks
plt.xlabel('Customer Tariff groups') #x-axis labels
plt.ylabel('Consumption per Tariff') #y-axis labels
plt.title('Consumption of Customer per Tariff') #plot title
plt.show();#display

# Customer Revenue per Tariff
plt.bar(table.index,revenue_per_tariff['_TOTAL_AMOUNT']) #bar graph
plt.xticks(rotation=70)#xticks
plt.xlabel('Customer Tariff groups') #x-axis labels
plt.ylabel('Revenue per Tariff') #y-axis labels
plt.title('Customer Revenue per Tariff') #plot title
plt.show();#display

# ### 7. Select the features

# ----- RAW_DATA ------------
#x1= raw_data.iloc[:,0].fillna(0)
#x2= raw_data.iloc[:,18:96].fillna(0)
#x3 = pd.concat([x1,x2],axis = 1)

# ---WORK_DATA-------------
#x1= raw_data.iloc[:,0].fillna(0)
#x2= work_data.iloc[:,18:27].fillna(0)
x1= work_data.iloc[:,0]
x2= work_data.iloc[:,18:24]
x3 = pd.concat([x1,x2],axis = 1)
x3.head()

# Our selected features
features_to_explore =
['AVG_KVA_UNITS','AVG_ACTIVE_UNITS','AVG_CONSUMPTION','TOTAL_KVA_UNITS','TOT
AL_ACTIVE_UNITS','_TOTAL_CONSUMPTION']

# Let's create two new dataframes with our new data called customer_data and another where we
do a groupby on customer_data (ACCOUNT)
customer_data = x3.dropna(axis=0)[features_to_explore + ['ACCOUNT']]
customer_groups = x3.groupby('ACCOUNT').mean().reset_index().dropna(axis=0)

customer_data.head()

customer_data.shape

customer_groups.head()

customer_groups.shape
customer_groups.describe()

# ### Standardizing our data

# Standardizing data, so that all features have equal weight. This is important for modeling.
# Otherwise, in our case, _TOTAL_CONSUMPTION would be considered much more important
than the rest of the features, for Instance.
# We do not know if this is the case, so we would not like to introduce it to our model.
# This is what is also referred to as bias.
customer_groups_std = customer_groups.copy()
cols =
['AVG_KVA_UNITS','AVG_ACTIVE_UNITS','AVG_CONSUMPTION','TOTAL_KVA_UNITS','TOT
AL_ACTIVE_UNITS','_TOTAL_CONSUMPTION']

customer_groups_std[cols] = preprocessing.scale(customer_groups_std[cols])

customer_groups_std.head()

customer_groups_std.describe()

# ### 8. Clustering

# #### K-Means Clustering

# #### 1. Elbow Method

# Perform K-means clustering. We consider 1 to 15 clusters, so our for loop runs 14 iterations.
# In addition we run the algortihm at many different starting points - k means++.
# And we set a random state for reproducibility.
wcss = [] # Initialize the wcss variable by making it an empty list
for i in range(1,15): # for loop for trying out several clustering solutions
kmeans = KMeans(n_clusters = i, init = 'k-means++',max_iter=300,n_init=10, random_state = 50)
kmeans.fit(customer_groups_std)
wcss.append(kmeans.inertia_)

# Plot the Within Cluster Sum of Squares for different number of clusters.
# From this plot we choose the number of clusters.
# We look for a kink in the graph, after which the descent of wcss isn't as pronounced.
plt.figure(figsize = (14,8))
plt.plot(range(1, 15), wcss, marker = 'o', linestyle = '--')
plt.title('The Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()

# #### 2. Silhouette Coefficient
for n_cluster in range(2, 11):
    kmeans = KMeans(n_clusters=n_cluster).fit(customer_groups_std[features_to_explore])
    label = kmeans.labels_
    sil_coeff = silhouette_score(customer_groups_std[features_to_explore], label,
metric='euclidean')
    print("For n_clusters={}, The Silhouette Coefficient is {}".format(n_cluster, sil_coeff))

# Let's take k to be 3 clusters
num_k = 3
kmeans = cluster.KMeans(n_clusters=num_k,init='k-means++', max_iter=300,n_init=10,
random_state=42)
identified_clusters = kmeans.fit_predict(customer_groups_std)

#identified_clusters = list(set(identified_clusters))
#identified_clusters

data_with_clusters = customer_groups_std.copy()
data_with_clusters['Clusters'] = identified_clusters
data_with_clusters.head()

#identified_clusters = list(set(identified_clusters))
#identified_clusters
data_with_clusters['Clusters'].unique()

# ### 9. Cluster Analysis

# We create a new data frame with the original features and add a new column with the assigned
clusters for each point.
df_clust_kmeans = customer_groups.copy()
df_clust_kmeans['Cluster K-means'] = kmeans.labels_

# Calculate mean values for the clusters
df_clust_analysis = df_clust_kmeans.groupby(['Cluster K-means']).mean()
df_clust_analysis

# let’s see the number of customers in each of the clusters and their proportions:
df_clust_analysis['Number of Customers'] = df_clust_kmeans[['Cluster K-
means','ACCOUNT']].groupby(['Cluster K-means']).count()
df_clust_analysis['Proportion of Customers'] = df_clust_analysis['Number of Customers'] /
df_clust_analysis['Number of Customers'].sum()
df_clust_analysis

# Rename the clusters accordingly from the analysis results
df_clust_analysis.rename({0:'Commercial Customers',
                          1:'Industrial(Large Power) Customers',
                          2:'Domestic(Standard Power) Customers'})

# Plot our Cluster Counts
data_with_clusters.groupby('Clusters')['ACCOUNT'].agg('count').plot(kind='bar')
```

```
# Add the segment labels to our table
df_clust_kmeans['Labels'] = df_clust_kmeans['Cluster K-means'].map({0:'Commercial Customers',
                        1:'Industrial(Large Power) Customers',
                        2:'Domestic(Standard Power) Customers'})

# We plot the results from the K-means algorithm.
# Each point in our data set is plotted with the color of the clusters it has been assigned to.
x_axis = df_clust_kmeans['ACCOUNT']
y_axis = df_clust_kmeans['_TOTAL_CONSUMPTION']
plt.figure(figsize = (10, 8))
#sns.scatterplot(x_axis, y_axis, hue = df_clust_kmeans['Labels'], palette = ['g', 'r', 'c', 'm','b'])
sns.scatterplot(x_axis, y_axis, hue = df_clust_kmeans['Labels'], palette = ['g', 'r', 'c'])
plt.title('Cluster K-means')
plt.show()

# ### 10. K-Means Clustering with PCA

# let’s see the number of customers in each of the clusters:
clustered_customers = pd.DataFrame(customer_groups_std)
clustered_customers['cluster'] = identified_clusters
clustered_customers['cluster'].value_counts(ascending =True)

# Principal Component Analysis for Visualization
pca = decomposition.PCA(n_components=2, whiten=True)
data_with_clusters['x'] = pca.fit_transform(data_with_clusters)[:, 0]
data_with_clusters['y'] = pca.fit_transform(data_with_clusters)[:, 1]
plt.figure(figsize=(10,6))
scatter = plt.scatter(data_with_clusters['x'],
data_with_clusters['y'],c=data_with_clusters['Clusters'],cmap='rainbow')

plt.title('Customer Clusters', weight='bold').set_fontsize('14')
plt.xlabel('Dimension 1', weight='bold').set_fontsize('10')
plt.ylabel('Dimension 2', weight='bold').set_fontsize('10')
L = plt.legend(*scatter.legend_elements(), loc="upper right", title="Clusters")
L.get_texts()[0].set_text('Commercial Customers')
L.get_texts()[1].set_text('Industrial(Large Power) Customers')
L.get_texts()[2].set_text('Domestic(Standard Power) Customers')
plt.show()

# #### 10.1 Testing the model with sample data

sample_test_1 = [(20803318,-0.018618,-0.105839,-0.179932,-0.018618,-0.105839,-0.179932)]
sample_test_2 = [(91870452,-0.018616,-0.092330,-0.157524,-0.018618,-0.092500,-0.157524)]
sample_test_3 = [(10891454,-0.018618, 0.049440, 0.371934,-0.018618, 0.049440, 0.371934)]

sample_test = pd.read_csv('Test_Sample.csv')
```

```
customer_prediction = kmeans.predict(sample_test_1)

customer_prediction

# ### 11. Clusters Mapping with the Actual Customer Tariff Classification

# Map the Customer tariff Categories with the Identified Clusters
cluster_map = customer_groups.copy()
cluster_map['Tariff_Category'] = work_data.iloc[:,4]
cluster_map['cluster'] = kmeans.labels_

cluster_1 = cluster_map[cluster_map.cluster == 0]
cluster_2 = cluster_map[cluster_map.cluster == 1]
cluster_3 = cluster_map[cluster_map.cluster == 2]

# Customer count per Tariff per cluster
Cluster1_customer_counts_per_tariff =
cluster_1.groupby('Tariff_Category').count()[['ACCOUNT']]
Cluster2_customer_counts_per_tariff =
cluster_2.groupby('Tariff_Category').count()[['ACCOUNT']]
Cluster3_customer_counts_per_tariff =
cluster_3.groupby('Tariff_Category').count()[['ACCOUNT']]

# Rename the columns
Cluster1_customer_counts_per_tariff.columns = ['Commercial Customers']
Cluster2_customer_counts_per_tariff.columns = ['Industrial(Large Power) Customers']
Cluster3_customer_counts_per_tariff.columns = ['Domestic(Standard Power) Customers']

# Merge the columns into one table
customer_counts_per_tariff_per_cluster =
pd.concat([Cluster1_customer_counts_per_tariff,Cluster2_customer_counts_per_tariff,Cluster3_cu
stomer_counts_per_tariff],axis = 1)

customer_counts_per_tariff_per_cluster

customer_counts_per_tariff_per_cluster["Number of Customers"] =
customer_counts_per_tariff_per_cluster.sum(axis=1)

customer_counts_per_tariff_per_cluster.fillna(0)

# ### 12. Clustering Performance Evaluation

# #### 12.1 Silhouette Coefficient

# creating instance of labelencoder
labelencoder = LabelEncoder()
# Assigning numerical values and storing in another column
tariff_label_data = raw_data.copy()
```

```
#tariff_label_data['tariff_label'] = labelencoder.fit_transform(tariff_label_data['TARIFF_CODE'])
labels_true = labelencoder.fit_transform(tariff_label_data['TARIFF_CODE'])

kmeans_model = KMeans(n_clusters = 3, random_state = 1).fit(customer_groups_std)
labels_pred = kmeans_model.labels_

#labels_pred = labels_true[:]

silhouette_score(customer_groups_std, labels_pred, metric = 'euclidean')

metrics.adjusted_rand_score(labels_true, labels_pred)

metrics.adjusted_mutual_info_score(labels_true, labels_pred)

metrics.homogeneity_score(labels_true, labels_pred)

metrics.completeness_score(labels_true, labels_pred)

metrics.v_measure_score(labels_true, labels_pred)

metrics.homogeneity_completeness_v_measure(labels_true, labels_pred)

metrics.fowlkes_mallows_score(labels_true, labels_pred)

print('1. adjusted_rand_score: %f'%metrics.adjusted_rand_score(labels_true, labels_pred))
print('2. silhouette_score: %f'%metrics.silhouette_score(customer_groups_std, labels_pred,
metric='euclidean', sample_size=100000))
print('3. homogeneity_score: %f'%metrics.homogeneity_score(labels_true, labels_pred))
print('4. completeness_score: %f'%metrics.completeness_score(labels_true, labels_pred))
print('5. v_measure_score: %f'%metrics.v_measure_score(labels_true, labels_pred))
# print('6.
homogeneity_completeness_v_measure:'%metrics.homogeneity_completeness_v_measure(labels_tr
ue, labels_pred))
print('6. fowlkes_mallows_score: %f'%metrics.fowlkes_mallows_score(labels_true, labels_pred))
print('7. adjusted_mutual_info_score: %f'%metrics.adjusted_mutual_info_score(labels_true,
labels_pred))
```