
# Application of Supervised and Unsupervised Learning for Data Prediction and Data Clustering

Internship Final Project Report

Submitted as part of the requirements for the Generative AI and Data Science Internship at FlexiSaf Limited

Prepared By:

Buhari Bashir Adamu

201605005

Department of Computer Science, Nile University of Nigeria

Supervisors:

Internship Coordinator: Mariam Ali

Program Support Officer: Michael Okoronu

Date of Submission: 28/09/2025

TABLE OF CONTENTS

# CHAPTER ONE:

# INTRODUCTION

## 1.1 INTRODUCTION

This paper documents the development process of two machine learning models: a predictive model for forecasting electricity demand using supervised learning and a clustering model for grouping patient data using unsupervised learning.

## 1.2 PROJECT OBJECTIVES

The project objectives of the Supervised learning portion is as follows:

•Train a predictive model to make forecasts using labelled data

•Make a forecast using the model

The project objectives of the Supervised learning portion is as follows:

•Train a clustering model to find patterns in unlabelled data

•Visualize the identified patterns

## 1.3 PAIN POINTS

Pain Points for Demand Predictive Model:

•Price Optimization

Concern:

Electricity Demand Forecasting models addresses several potential pain points experienced in the power industry. One concern among companies operating in the electricity industry in pricing. It is a major concern in countries with established electricity markets in place where producers, middle men, and utility companies sell or purchase contracts dictating electricity delivery.

Solution:

Electricity demand forecasting allows each participant to set the buy or sale price of electricity contracts according to forecasts in order to buy as cheaply as possible or sell at as high a price as possible. For example, in order for producers to maximize profit, complex forecasting models are utilized, allowing them to set prices based on expected changes in demand allowing them to capture as much profit as possible with every sale. While, in order for utility firms to reduce costs, they utilize models to set prices as low as possible for every sale.

Pain points for patient data clustering model:

•Misdiagnoses

Concern:

In the medical field, misdiagnoses are an infrequent but very serious concern. Giving a patient the incorrect diagnosis can have cascading health effects. In the event of a misdiagnosis the patient's condition is not being addressed and has the potential to worsen as a result. Additionally, any treatment given may cause the patient to present side-effects which add to their discomfort with no benefit to them. In the case of medicine with particularly serious side-effects being prescribed, there is the potential for the patient to require further treatment as a result of the side-effects.

Solution:

The use of clustering models on patient data can provided health practitioners with new insights into complex associations between certain age, geographic, race, and gender groups with certain conditions. Leveraging new insights gained for diagnostic purposes could make all the difference in reaching the proper diagnosis and providing needed care.

# CHAPTER TWO:

# DEMAND PREDICTIVE MODEL

## 2.1 DATASET

The dataset used to develop the demand predictive model was sourced from Kaggle.com. It contains data on electricity demand, spanning a period of five years, from 2020-2025. It contains the following columns: 'Timestamp', 'hour', 'dayofweek', 'month', 'year', 'dayofyear', 'Temperature', 'Humidity', 'Demand'. It holds a total of 43,848 rows, with each row representing an hourly reading.

## 2.2 DATA CLEANING & PREPROCESSING

Given that the dataset contains time-series data, the 'Timestamp' column was converted into a datetime object and set as the index.

Rows containing missing values were identified and isolated. Rows containing only null values were removed from the dataset. The progression of the data for each column was studied in order to select the most suitable technique to address missing values. Linear interpolation was employed for the 'hour' column as a linear relationship was identified between the entries. A combination of forward fill and backward fill were used to address null values in the 'dayofweek', 'month', 'year', and 'dayofyear' columns as the entries were noted to be frequently repeated due to the hourly change across entries. Time interpolation was applied to the 'Temperature', 'Humidity', and 'Demand' columns in order to address missing values as they hold time-series entries.

Additional time features were created to allow for the data to be represented across other timeframes. A 'weekofyear', 'weekend', and 'quarter' feature were added. Several lagging features were created to allow the time-series dataset to be used with memory-less models. The following lagging features were created: 'Demand_Lag1H', 'Demand_Lag24H', 'Demand_Lag_1W', 'Demand_Lag_1M'.

To indicate the overall trend of the demand as well as its volatiliy, the following features were created: 'Rolling_Mean_24H_(Demand)', 'Rolling_Std_24H_(Demand)'.

The creation of lagging features resulted in the creation of new null fields. These were addressed by removing any row containing a null field from the dataset.

## 2.3 MACHINE LEARNING ALGORITHM

The machine learning algorithm selected to develop the predictive model was XGBoost. XGBoost is a memory-less machine learning algorithm designed to applied in regression and classification problems. It creates many decision trees, which are machine learning algorithms and combines the results in order to produce a model. This strategy results in a more accurate model than if fewer decision trees were used. Additionally, XGBoost employs sequential Error correction, where each new tree accounts for the error of the previous tree. This results in even higher accuracy compared to traditional decision trees.

## 2.4 MODEL TRAINING PROCESS

To train the model The XGBRegressor module was used. The parameters were set as the following:

n_estimators=1000,

early_stopping_rounds=50,

learning_rate=0.01,

random_state=50,

objective=”reg: squared error”

The number of training cycles were set to 1000, in order to prevent overfitting the model to the data. The termination condition for the training process was set to end training if there was no improvement after 50 rounds, in order to reduce computing costs. The learning rate, or the weight of the effect of the result of each round on the final model was set as 0.01 to prevent overfitting. The random seed variable was set to allow for repeatable results. The loss function selected for minimization was the mean squared error.

## 2.5 EVALUATION METRICS

Over the course of training, the loss function gradually reduced with each round, starting at a

approximate value of 1403.08 and ending with a value of approximately 170.588. This indicates that the model continuously improved over the training cycles.

To evaluate the performance of the model, the following metrics were used: mean squared error and mean absolute error.

Mean Squared Error

The difference between the actual values and the predicted values is determined. The differences are squared and averaged in order to produce the mean squared error.

$$\mathbf{MSE} = \frac{1}{n}\sum(Y_i - \hat{Y}_i)^2$$

n – Number of Trials

$Y_i$ – Actual Value

$\hat{Y}_i$ – Predicted Value

Mean Absolute Error

The difference between the actual values and the predicted values is determined. The absolute values of the differences are calculated and averaged to produce the mean squared error.

$$\mathbf{MAE} = \frac{1}{n}\sum_{i=1}^{n}|x_i - x|$$

n – Number of Trials

$x_i$ – Actual Value

x – Predicted Value

## 2.6 MODEL FORECAST

To make a forecast using the model, I passed a Dataframe created from a dictionary containing the following values to the model:

“hour”: [0.0], “dayofweek”:[2], “month”:[1], “year”:[2025], “dayofyear”:[1], “weekofyear”:[1], “weekend”:[0], “quarter”:[1], “Temperature”:[5.0427876], “Humidity”:[60.46985], “Demand_Lag_24H”:[2722.480185], “Demand_Lag_1W”:[2756.898645], “Demand_Lag_1M”:[4556.916590], “Rolling_Mean_24H(Demand)”[4065.985340]:, “Rolling_Std_24H(Demand)”:[18.215325]

The resulting prediction was 2806.2727.

# CHAPTER THREE:

# PATIENT DATA CLUSTERING MODEL

## 3.1 DATASET

The dataset used in the development of the clustering model was obtained from Kaggle.com. It contains data on hospital patients collected from various hospitals. It contains 10,234 rows, each representing an individual patient. The dataset gives a general overview of the condition of the patients, with features indicating medical test results and medical conditions.

It contains the following columns: 'encounter_id', 'hospital_id', 'age', 'bmi', 'elective_surgery', 'ethnicity', 'gender', 'height', 'hospital_admit_source', 'icu_admit_source', 'icu_id', 'icu_stay_type', 'icu_type', 'pre_icu_los_days', 'readmission_status', 'weight', 'albumin_apache', 'apache_2_diagnosis', 'apache_3j_diagnosis', 'apache_post_operative', 'arf_apache', 'bilirubin_apache', 'bun_apache', 'creatinine_apache', 'fio2_apache', 'gcs_eyes_apache', 'gcs_motor_apache', 'gcs_unable_apache', 'gcs_verbal_apache', 'glucose_apache', 'heart_rate_apache', 'hematocrit_apache', 'intubated_apache', 'map_apache', 'paco2_apache', 'paco2_for_ph_apache', 'pao2_apache', 'ph_apache', 'resprate_apache', 'sodium_apache', 'temp_apache', 'urineoutput_apache', 'ventilated_apache', 'wbc_apache', 'd1_diasbp_invasive_max', 'd1_diasbp_invasive_min', 'd1_diasbp_max', 'd1_diasbp_min', 'd1_diasbp_noninvasive_max', 'd1_diasbp_noninvasive_min', 'd1_heartrate_max', 'd1_heartrate_min', 'd1_mbp_invasive_max', 'd1_mbp_invasive_min', 'd1_mbp_max', 'd1_mbp_min', 'd1_mbp_noninvasive_max', 'd1_mbp_noninvasive_min', 'd1_resprate_max', 'd1_resprate_min', 'd1_spo2_max', 'd1_spo2_min', 'd1_sysbp_invasive_max', 'd1_sysbp_invasive_min', 'd1_sysbp_max', 'd1_sysbp_min', 'd1_sysbp_noninvasive_max', 'd1_sysbp_noninvasive_min', 'd1_temp_max', 'd1_temp_min', 'h1_diasbp_invasive_max', 'h1_diasbp_invasive_min', 'h1_diasbp_max', 'h1_diasbp_min', 'h1_diasbp_noninvasive_max', 'h1_diasbp_noninvasive_min', 'h1_heartrate_max', 'h1_heartrate_min', 'h1_mbp_invasive_max', 'h1_mbp_invasive_min', 'h1_mbp_max', 'h1_mbp_min', 'h1_mbp_noninvasive_max', 'h1_mbp_noninvasive_min', 'h1_resprate_max', 'h1_resprate_min', 'h1_spo2_max', 'h1_spo2_min', 'h1_sysbp_invasive_max', 'h1_sysbp_invasive_min', 'h1_sysbp_max', 'h1_sysbp_min', 'h1_sysbp_noninvasive_max', 'h1_sysbp_noninvasive_min', 'h1_temp_max', 'h1_temp_min', 'd1_albumin_max', 'd1_albumin_min', 'd1_bilirubin_max', 'd1_bilirubin_min', 'd1_bun_max', 'd1_bun_min', 'd1_calcium_max', 'd1_calcium_min', 'd1_creatinine_max', 'd1_creatinine_min', 'd1_glucose_max', 'd1_glucose_min', 'd1_hco3_max', 'd1_hco3_min', 'd1_hemaglobin_max', 'd1_hemaglobin_min', 'd1_hematocrit_max', 'd1_hematocrit_min',

'd1_inr_max', 'd1_inr_min', 'd1_lactate_max', 'd1_lactate_min', 'd1_platelets_max', 'd1_platelets_min', 'd1_potassium_max', 'd1_potassium_min', 'd1_sodium_max', 'd1_sodium_min', 'd1_wbc_max', 'd1_wbc_min', 'h1_albumin_max', 'h1_albumin_min', 'h1_bilirubin_max', 'h1_bilirubin_min', 'h1_bun_max', 'h1_bun_min', 'h1_calcium_max', 'h1_calcium_min', 'h1_creatinine_max', 'h1_creatinine_min', 'h1_glucose_max', 'h1_glucose_min', 'h1_hco3_max', 'h1_hco3_min', 'h1_hemaglobin_max', 'h1_hemaglobin_min', 'h1_hematocrit_max', 'h1_hematocrit_min', 'h1_inr_max', 'h1_inr_min', 'h1_lactate_max', 'h1_lactate_min', 'h1_platelets_max', 'h1_platelets_min', 'h1_potassium_max', 'h1_potassium_min', 'h1_sodium_max', 'h1_sodium_min', 'h1_wbc_max', 'h1_wbc_min', 'd1_arterial_pco2_max', 'd1_arterial_pco2_min', 'd1_arterial_ph_max', 'd1_arterial_ph_min', 'd1_arterial_po2_max', 'd1_arterial_po2_min', 'd1_pao2fio2ratio_max', 'd1_pao2fio2ratio_min', 'h1_arterial_pco2_max', 'h1_arterial_pco2_min', 'h1_arterial_ph_max', 'h1_arterial_ph_min', 'h1_arterial_po2_max', 'h1_arterial_po2_min', 'h1_pao2fio2ratio_max', 'h1_pao2fio2ratio_min', 'aids', 'cirrhosis', 'hepatic_failure', 'immunosuppression', 'leukemia', 'lymphoma', 'solid_tumor_with_metastasis'.

## 3.2 DATA CLEANING & PREPROCESSING

First, I addressed null values. I dropped columns from the dataset where the columns were more than 50% null, using a python program that compares the number of null values in a column to the half the length of the index of the dataframe.

Due to the presence of so many null values, rather than relying on traditional imputation techniques, I used the machine-learning based imputation method, IterativeImputer. IterativeImputer fills in missing fields with data gotten from rows determined to be closest to the row valuewise. It relies on the assumption that rows found to have similar values for certain features, will have similar values for other features. I applied IterativeImputer across the entire dataset, using it to address all missing values.

I created a new dataset, only including the features based on their perceived revelance. The resulting dataframe contained the following features: 'age', 'bmi', 'height', 'weight', 'gender_F', 'gender_M', 'ethnicity_African American', 'ethnicity_Asian', 'ethnicity_Caucasian', 'ethnicity_Hispanic', 'ethnicity_Native American', 'ethnicity_Other/Unknown', 'aids', 'cirrhosis', 'hepatic_failure', 'immunosuppression', 'leukemia', 'lymphoma', 'solid_tumor_with_metastasis', 'elective_surgery', 'd1_heartrate_max', 'd1_heartrate_min', 'heart_rate_apache', 'intubated_apache', 'icu_type_CCU-CTICU', 'icu_type_CSICU', 'icu_type_CTICU', 'icu_type_Cardiac ICU', 'icu_type_MICU', 'icu_type_Med-Surg ICU', 'icu_type_Neuro ICU', and 'icu_type_SICU'.

## 3.3 MODEL TRAINING PROCESS

To train the model the parameters of KMeans were set as the following:

n_clusters=5

random_state=21,

n_init="auto"

The number of clusters the data should be sorted into was specified as 5. After experimenting with the number of clusters, 5 produced the most favorable silhouette score. The random seed variable was set to 21, to allow for repeatable results. The number of different centroid values that will be tested was set to 10 by setting the "n_init" parameter as 10, allowing for the best value to be determined over multiple runs.

## 3.4 MACHINE LEARNING ALGORITHM

The machine learning algorithm used to train the clustering model was the KMeans algorithm. The KMeans algorithm is a machine learning algorithm that groups unlabelled data in a specified number of a groups, referred to as clusters. Each data point is assigned to a cluster based on its similarities to the members of the cluster. The resulting groups can help to indicate hidden patterns in the data. After the number of clusters has been indicated, centroids, which are the center of the clusters are randomly chosen from the data. The data points are assigned to the nearest centroid. This results in clusters. New centroid values are calculated by finding the mean of the points. These steps are repeated until the centroids no become stable.

## 3.5 EVALUATION METRICS

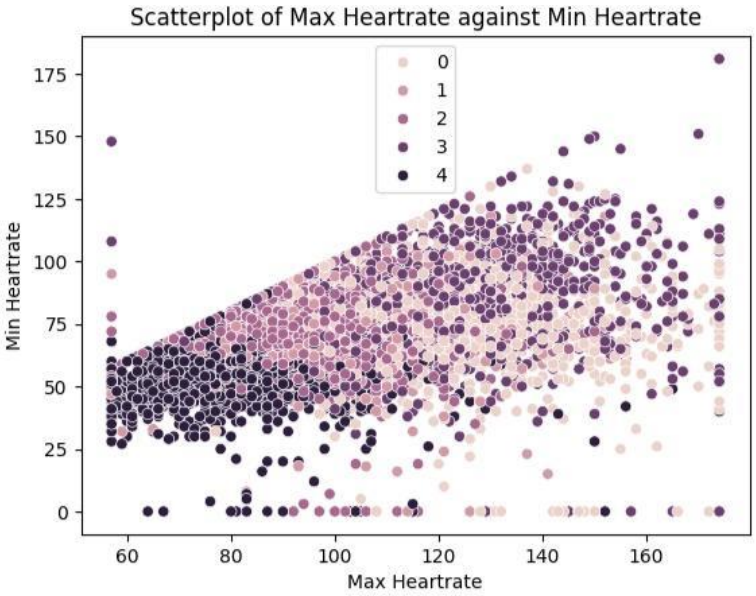
The metric used to evaluate the clustering model is silhouette scoring. This involves calculating the silhouette coefficient for each data point and averaging the results. The silhouette score indicates that the resulting clusters are well-defined.

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

a(i) – the average distance between the ith point and the other points in the same cluster

b(i) – the minimum average distance between the ith point and and points in other clusters.

## 3.6 DATA VISUALIZATION




The above graph showcases the relationship between the minimum and maximum heartrates of the patients. The legend of the graph indicates the grouping of data after clustering. Five clusters were created: Cluster 0, Cluster 1, Cluster 2, Cluster 3, and Cluster 4. There is the presence of overlapping in the data which indicates the model faced difficulty grouping the data. This likely occurred as a result of the high dimensionality of the dataset, which introduced noise.

REFERENCES

•DataCamp. K-Means Clustering in Python: How to Build Your First Model. Retrieved from https://www.datacamp.com/tutorial/k-means-clustering-python

•GeeksforGeeks. Machine Learning | Unsupervised Learning. Retrieved from https://www.geeksforgeeks.org/machine-learning/unsupervised-learning/

•GeeksforGeeks. Python | How to Calculate Mean Absolute Error (MAE) in Python. Retrieved from https://www.geeksforgeeks.org/python/how-to-calculate-mean-absolute-error-in-python/

•GeeksforGeeks. Python | Python Mean Squared Error. Retrieved from https://www.geeksforgeeks.org/python/python-mean-squared-error/

•GeeksforGeeks. XGBoost Parameters Explained. Retrieved from https://www.geeksforgeeks.org/machine-learning/xgboost-parameters/

•Help Desk Geek. How to Calculate Mean Squared Error (MSE) in Microsoft Excel. Retrieved from https://helpdeskgeek.com/how-to-calculate-mean-squared-error-mse-in-microsoft-excel/

•IBM. K-means clustering. Retrieved from https://www.ibm.com/think/topics/k-means-clustering

•scikit-learn. sklearn.cluster.KMeans. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

•Statistics How To. Absolute Error. Retrieved from https://www.statisticshowto.com/absolute-error/

•Tahera, Firdose. Understanding the Silhouette Score. Medium. Retrieved from https://tahera-firdose.medium.com/understanding-the-silhouette-score-fb5109ea28dc

•XGBoosting.com. Configure XGBoost random_state parameter. Retrieved from https://xgboosting.com/configure-xgboost-random_state-parameter/

•XGBoosting.com. What is the XGBoost Algorithm? Retrieved from https://xgboosting.com/what-is-the-xgboost-algorithm/

One app for all your Word, Excel, PowerPoint and PDF needs. Get the Microsoft 365 app: https://aka.ms/GetM365