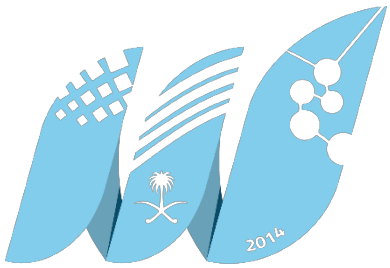


جامعة جدة
University of Jeddah

Houses' Energy Efficiency Prediction

Senior Project / Final Report

By

Amal Abdulmalik Alharthie	1905307
Bedar Omar Bahassan	1913685
Ghaida Mahmoud Allukmani	1911655
Waad Abdullah Aljehani	1913730

Supervised By Dr. Ines Boufateh

Department of Information Systems & Technology
College of Computer sciences & Engineering

University of Jeddah

Jeddah– Saudi Arabia

[Fall/Spring 2023]

Declaration by Author

"We certify that this work has not been accepted in substance for any degree, and is not concurrently being submitted for any degree other than that of BS Data science being studied at university of Jeddah. we also declare that this work is the result of my/our own findings and investigations except where otherwise identified by references and that we have not plagiarized another's work".

Authors' Name:

Bedar Bahassan
Amal Alharthie
Ghaida Allukmani
Waad Aljehani

Declaration by Supervisor

I, the undersigned hereby certify that I have read this project report and finally approve it with recommendation that this report may be submitted by the authors above to the final year project evaluation committee for final evaluation and presentation, in partial fulfillment of the requirements for the degree of BS data science at the Department of Information systems and Technology, Faculty of computer sciences and Engineering, University of Jeddah.

Dr Ines Boufateh

Dedication

All praise is to Allah, praise in abundance, good and blessed. Our completion of this project could not have been accomplished without the support of Allah. Thanks to Allah for giving us knowledge, guidance, and the opportunity to complete this project and reach our objectives. with full gratitude to everyone who supported us, starting with our parents, and friends who believed in our capabilities to finish this semester. We also address a special thanks to Dr. Ines Boufateh for standing with us step by step and finally, we would like to thank ourselves for completing this project

Abstract

In a city with many energy-consuming buildings, estimating and understanding building energy efficiency is critical. A better understanding of building energy efficiency can assist to decrease overall household energy consumption while also providing guidance for future housing improvements.

The purpose of our project is to develop energy prediction models based on the Glasgow 'UK' dataset. Furthermore, the chatbot assists consumers to understand the energy consumption of their dwellings based on property features.

In this report, we outlined the problem we want to solve, the expected solution, and the project's goal. Following that, we discussed the previous research and how it related to the project idea. Rapid Miner and Excel were then used to explore and visualize data.

In addition, identify the features that affect residential energy use and discuss data problems and solutions. Furthermore, preparing and processing data in Python for use in prediction models.

Following that, we established several deep ANN and machine learning models and perform numerous experiments to improve them. We found that SVM had the lowest prediction accuracy 72%. The accuracy of Random Forest was 86%. Deep ANN after improvement has an excellent performance of 87%. Deep ANN is highly recommended when computational time is not a primary concern. The energy efficiency rating band is predicted by XGBoost, and it achieves 89%. Since XGBoost provided the highest accuracy, it was used in the chatbot to achieve our objective.

Table of Content

Table of Contents

List of Figures

List of Tables

Chapter I

1.1 Introduction

According to The Energy Statistics Department in the Kingdom of Saudi Arabia, the residential sector has consumed 47.58% of the total electrical energy in 2020 [1]. As well as Saudi Arabia's energy consumption equals **10,997,456,810,000 BTU (11.00 quadrillion BTU)** of energy, which represents 1.89% of global energy consumption [2].

However, the kingdom of Saudi Arabia has developed programs as a part of Vision 2030 that concentrates on sustainability such as "Saudi Energy Efficiency Program".

In addition, the Saudi Energy Efficiency Center focuses on increasing the energy efficiency in production and consumption to preserve KSA natural resources and enhancing the economic and social welfare of KSA population.

In this study, we target the residential sector and aim to build a chatbot that will help consumers estimate houses' energy efficiency [3].

Due to struggling to find real data in Saudi Arabia that could serve us in this study, we will be using the Scottish Energy Performance Certificate Register Publication of Energy Performance Data for Domestic Buildings Dataset.

The knowledge extracted from this dataset will help us to identify the variables that will contribute the most in predicting houses' energy efficiency and adapt the study to Saudi Arabia context in the future.

1.2 Problem definition

Recently, people's interest in energy saving has increased, and it has become a topic of interest to many customers. Suppliers now offer different solutions to save energy, such as choosing LED lights instead of halogen.

In addition, consumers are interested in reading the energy label on electrical appliances. All these factors have made it important for customers to consider the energy efficiency aspect when buying or renting a house to save the world. Unfortunately, this information is not available and is very complex to be estimated by the consumer. In this project we will help the consumers to evaluate the house from aspect of energy efficiency by applying machine learning model and providing them the information through the use of an easy and friendly- user tool which is a Chatbot.

1.3 The recommended solution

Applying energy efficiency estimation in houses by using machine learning models. We will predict energy efficiency based on some columns that are useful in predicting the energy efficiency of the house by train a part of the data to predict the house energy efficiency and then test the model and try to get the best accuracy. We will build a chatbot that can help a customer to evaluate houses' energy efficiency easily.

1.4 Project Scope

The main objective of our project is to apply energy efficiency estimation in houses by using machine learning models. In this Section, the aim of our project is explained in Section 1.4.1, and the objectives of our project are shown in Section 1.4.2.

1.4.1 Aims

Build a chatbot that can evaluate the energy efficiency rating of the houses based on some variables by using machine learning models to help the consumers evaluate the house from aspect of energy efficiency by calculate some variable of data based on some columns.

1.4.2 Objectives

To reach the aim of the project, several objectives must be achieved, and they are as follow:

1. Discovering the business domain which we have done by reading similar work and distinguishing our work by developing new ideas which use the machine learning to help the consumers to evaluate the house from aspect of energy efficiency.
2. Preparing the data, based on our idea of using some variable to calculate the energy efficiency of the houses based on some variables.
3. In planning model, we will use previous data to establish our model about energy efficiency.
4. Building the model and getting the optimal result.
5. Communicating results and making recommendations for future work or improvements to existing processes.

1.5 Target Users

The stakeholders could be classified into three categories:

- All consumers who want to evaluate the house from aspect of energy efficiency Building construction, electricity, and energy companies that focus on increasing the energy efficiency in production and consumption can identify the potential features to achieve improvements.
- The government can also take advantage of the outcomes of this study in defining energy efficiency labeling policies in the residential building domain.

1.6 Methodology

To achieve our goal of predicting of energy efficiency ratings in buildings, we will use previous data and research to establish our model about energy efficiency. We recommend using supervised learning techniques such as SVM and ANN, as well as the ensemble learning technique used in machine learning to improve overall performance by combining multiple models for example Random Forests [4]. In our case, we intend to estimate the energy efficiency of a building using data collected in the UK [5]. We will employ the data science cycle methodology, which typically involves the following: business understanding, data mining, data cleaning, data exploration, feature engineering, predictive modelling, and data visualization. The project follows a systematic process as follows:

1.6.1 Business Understanding

Identify data's power and how to use it to accomplish the project objectives. Furthermore, the primary goal is to estimate building energy efficiency ratings by using machine learning techniques.

1.6.2 Data collection

The project's initial stage is to collect data from data published Zendo repository which combines research articles, data sets, research, and reports. The data was gathered between 2012 and 2022 in the United Kingdom [6].

1.6.3 Expletory data analysts

The exploratory analysis represents brainstorming for data and the relation between features. First import libraries and comprise studying the data with Pandas and plotting a graph using matplotlib. Moreover, we start to make assumptions about our data and the problem we are attempting to solve.

1.6.4 Data preparation and preprocessing

The most time-consuming stage of all data cleaning and processing involves resolving inconsistencies, handling incorrect and missing values, and dropping outliers and redundant features.

1.7 Project plan

In our project to predict the energy efficiency of houses in the UK, we evaluate the variables affecting energy consumption and develop a prediction model that uses machine learning techniques and classification algorithms across six months that starts in September 2022 and finishes in the second semester of 2023.

Based on the data science lifecycle approach, the project will cover seven main chapters: an introduction to the project idea and its value, a second background and review of literature, the third understanding of data, fourth data preparation and processing, a fifth model building, a sixth result and discussion, and a final is the conclusion and future work.

1.8 The Gant Chart

We used Gantt Chart to illustrate the project time as shown in Figure:

	Task Name	Duration	Start	Finish
4	group formation	1 day?	Thu 01/09/22 8:00 AM	Thu 01/09/22 5:00 PM
5	**PHASE 1**	**9 days?**	**Mon 05/09/22 8:00 A**	**Thu 15/09/22 5:00 PM**
6	problem defntion	1 day?	Mon 05/09/22 8:00 AI	Mon 05/09/22 5:00 PI
7	problem solution	1 day?	Wed 07/09/22 8:00 AI	Wed 07/09/22 5:00 PI
8	project objective	1 day?	Fri 09/09/22 8:00 AM	Fri 09/09/22 5:00 PM
9	identify the data	1 day?	Mon 12/09/22 8:00 AI	Mon 12/09/22 5:00 PI
10	project planning	1 day?	Thu 15/09/22 8:00 AM	Thu 15/09/22 5:00 PM
11	submit chapter 1	1 day?	Thu 15/09/22 8:00 AM	Thu 15/09/22 5:00 PM
12	**PHASE 2 : LITERATURE**	**18 days?**	**Thu 22/09/22 8:00 AI**	**Mon 17/10/22 5:00 P**
13	knowledge	1 day?	Thu 22/09/22 8:00 AM	Thu 22/09/22 5:00 PM
14	collect related work	1 day?	Fri 23/09/22 8:00 AM	Fri 23/09/22 5:00 PM
15	compare the previous researches	8 days?	Thu 06/10/22 8:00 AM	Mon 17/10/22 5:00 PI
16	submit chapter 2	1 day?	Thu 13/10/22 8:00 AM	Thu 13/10/22 5:00 PM
17	**PHASE 3 : DATA COLLECTION**	**8 days?**	**Tue 11/10/22 8:00 AM**	**Thu 20/10/22 5:00 PM**
18	data collection	1 day?	Tue 11/10/22 8:00 AM	Tue 11/10/22 5:00 PM
19	data description	1 day?	Wed 12/10/22 8:00 AI	Wed 12/10/22 5:00 PI
20	data exploration	1 day?	Fri 14/10/22 8:00 AM	Fri 14/10/22 5:00 PM
21	submit chapter 3	1 day?	Thu 20/10/22 8:00 AM	Thu 20/10/22 5:00 PM
22	**PHASE 4 : : Data preparation**	**14 days?**	**Mon 17/10/22 8:00 A**	**Thu 03/11/22 5:00 PM**
23	fix data structure	1 day?	Mon 17/10/22 8:00 AI	Mon 17/10/22 5:00 PI
24	manging outlier	1 day?	Mon 17/10/22 8:00 AI	Mon 17/10/22 5:00 PI
25	handle missing value	1 day?	Thu 20/10/22 8:00 AM	Thu 20/10/22 5:00 PM
26	submit chapter 4	1 day?	Thu 03/11/22 8:00 AM	Thu 03/11/22 5:00 PM
27	submit final report	1 day?	Fri 04/11/22 8:00 AM	Fri 04/11/22 5:00 PM
28	presentation	1 day?	Wed 09/11/22 8:00 AI	Wed 09/11/22 5:00 PI
29	revision	4 days?	Mon 05/12/22 8:00 AI	Thu 08/12/22 5:00 PM
30	**PHASE 5 : MODEL**	**30 days?**	**Mon 12/12/22 8:00 A**	**Fri 20/01/23 5:00 PM**
31	Develop data sets for testing, training	1 day?	Mon 12/12/22 8:00 AI	Mon 12/12/22 5:00 PI
32	Initialize the model	1 day?	Thu 15/12/22 8:00 AM	Thu 15/12/22 5:00 PM
33	apply the selected model to solve	1 day?	Mon 19/12/22 8:00 AI	Mon 19/12/22 5:00 PI
34	Evaluate the model	1 day?	Fri 20/01/23 8:00 AM	Fri 20/01/23 5:00 PM
35	**PHASE 6 : RESULT**	**12 days?**	**Fri 20/01/23 8:00 AM**	**Mon 06/02/23 5:00 P**
36	Present results numerically an/or visually.	12 days?	Fri 20/01/23 8:00 AM	Mon 06/02/23 5:00 PI
37	**PHASE 7 : (Conclusion & Future work)**	**10 days?**	**Mon 13/02/23 8:00 A**	**Fri 24/02/23 5:00 PM**
38	Conclude your work	5 days?	Mon 13/02/23 8:00 AI	Fri 17/02/23 5:00 PM
39	Submit Final Report to Supervisor and	5 days?	Mon 20/02/23 8:00 AI	Fri 24/02/23 5:00 PM
40	Final Report and Presentation	4 days?	Wed 01/03/23 8:00 AI	Mon 06/03/23 5:00

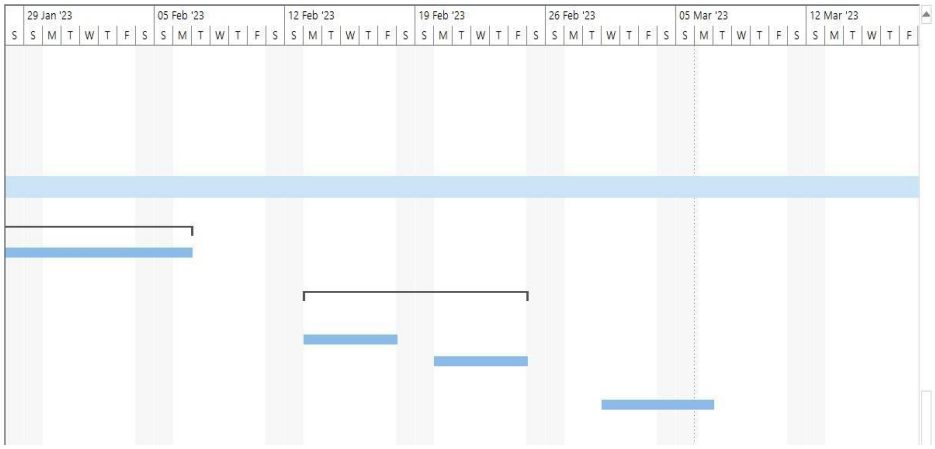


Figure 1 Gantt chart of the project plan

1.9 Tools and requirement

We start by working on Project Plan 365, which is used to create timelines and project plans. Afterward, for analytics and model development, we will utilize R studio and Python to use some crucial libraries that help us in preprocessing and visualize data, and to illustrate the results, we will use Python to use some crucial libraries, Power BI and Rapid Miner. Microsoft Office software will be employed to write the report, work steps, and project presentation.

1.10 Conclusion

This study aims to investigate the energy efficiency in the residential sector by analysing publicly available data of property-level buildings and applying machine learning techniques to achieve a high accuracy prediction of energy efficiency. Secondly, we will create a chatbot that will help the customer get an estimation of their houses' energy efficiency.

Chapter II: Background and Literature Review/ Related Work

2.1 Introduction

In previous research, estimate energy efficiency have been carried out using engineering calculations, simulation-based benchmarking, data-driven statistical models, and artificial intelligence techniques. Engineering methods rely on system complex details, including mathematics and building dynamics, as well as all building components, which are not readily available to the public in a large area; simulation-based benchmarking uses software and computer models that have complex details and can be used for a variety of applications, but it can be very expensive and time-consuming.[1]

Utilizing data-driven statistical models, which are more effective than engineering and simulation-based approaches, is now accessible thanks to the development of computational methods and data. Compared to other statistical models.

This project is a chatbot that will help consumers estimate houses' energy efficiency. Section 2.1 briefly describes the idea and its importance nowadays. Section 2.2 presents the knowledge area and main concepts that must be known to achieve the project's objectives then in Section 2.3, some related work was reviewed. At the end of the chapter, a comparison between the proposed solution and what is in the literature is laid out in Section 2.4. This is followed by a conclusion that is given in Section 2.5.

2.2 Background knowledge review

2.2.1 Energy efficiency for the residential domain.

The term Energy Efficiency basically refers to the use of less energy to provide the same service. It is one of the most cost-effective ways to protect the environment by reducing greenhouse gas (GHG) emissions and other pollutants that are created from energy production and use, reducing the amount of wasted energy, lowering overall electricity demand, and consequently reducing the costs for consumers [2].

In 2021, the residential sector emitted 68.1 MtCO2, accounting for 19.9% of all carbon dioxide emissions in the UK. The increase in carbon dioxide emissions between 2020 and 2021 was affected by the colder weather in 2021, resulting in more energy being used to heat homes.[3] The main energy sources in the residential sector are electricity and fossil fuels (natural gas or oil for heating, cooling, or cooking purposes). In residences, consumers power most appliances and equipment with electricity [4]. The electricity is basically pulled from power plants, which burn fossil fuels to produce energy. A by-product of this process is GHG such as carbon dioxide, methane, and other harmful pollutants.

Energy-efficient homes use less energy to heat, cool, and run appliances and electronics. This can be achieved by replacing older, less efficient appliances with newer, more efficient ones, better air-sealing, and better insulation. Some products, like energy-efficient LED light bulbs, use less energy to produce the same amount of light as incandescent light bulbs by using 75 to 80 percent less electricity. Other products don't use energy directly, but they improve the overall efficiency and comfort of a house or a building (such as thermal insulation or windows).[5]

2.2.2 Machine Learning Models

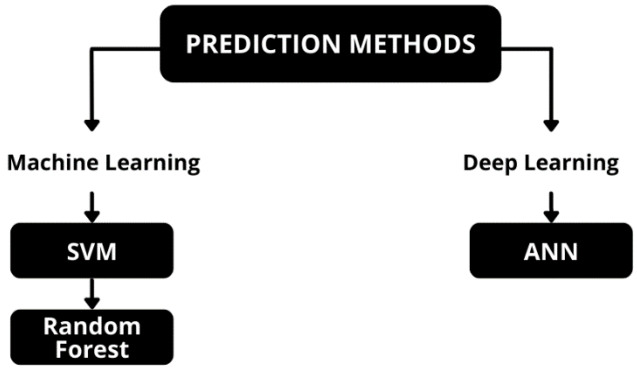


Figure 2 prediction models that will be used

As shown in figure 1, the methods that will be used in energy efficiency prediction are Machine Learning methods such as Support Vector Machine (SVM) and Random Forest (RF). As well as ANN which is a Deep Learning model

- **Random Forest**

RF is an ensemble learning method that is constructed from decision tree algorithm, it trains multiple decision trees using different subset of available features in parallel with bootstrapping followed by aggregation of decisions of individual trees (referred to as "Bagging") used in classification and regression problems.[6]

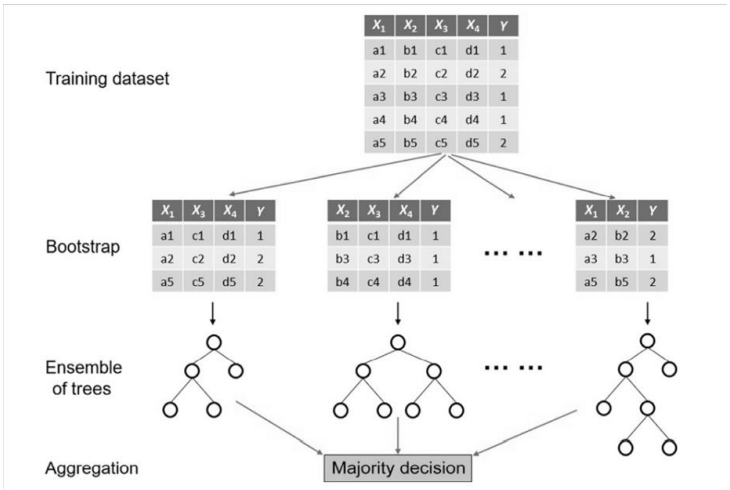


Figure 3 Implementation of RF classifier on a dataset that has four features (X1, X2, X3, and X4)

Random forests have the ability of generalization, which can prevent overfitting. The generalization error can be calculated as follows: $E * = PX,Y\,(M(X,Y) \lhd 0)$ where X and Y indicate the probability P of covering the X and Y spaces, respectively.

- **Support Vector Machine**

SVM is a linear classifier and the most popular Supervised Learning algorithm is used to train the learning and prediction, and the prediction error is analysed. After training the learning samples, the SVM can be used to predict the data, and the optimizer is required to optimize the support vector machine parameters.

- **Artificial Neural Network**

ANN is a computational model consisting of basic processing units, called "artificial neurons," which imitate the structure of biological neurons. Each biological neuron includes three major parts: dendrites, soma, and axon. Correspondingly, each artificial neuron also consists of three major parts: inputs ("dendrites"), transformation function ("soma"), and output ("axon")

The neuron computes the weighted sum of the input signals and compares the result with a threshold value θ. When the output is greater than 0 the neuron becomes activated.

$$X = \sum_{i=1}^{n} x_i w_i \qquad Y = \begin{cases} +1, if\ X \geq \theta \\ -1, if\ X < \theta \end{cases}$$

Equation 1 neuron activation function (Sign Function)

The perceptron proposed by Frank Rosenblatt based on McCulloch and Pitts neuron model is able to classify linearly separable inputs into two classes by dividing the n-dimensional space by a hyperplane into two decision regions using the linearly separable function.

$$X = \sum_{i=1}^{n} x_i w_i - \theta$$

However, it cannot solve linear inseparable problems, therefore it was necessary to construct a multi-layer network to solve problems with higher complexity.

Present-day neural networks analyse the data and use non-linear statistical methods to model complex relationships between inputs and outputs and to recognize patterns.[7]

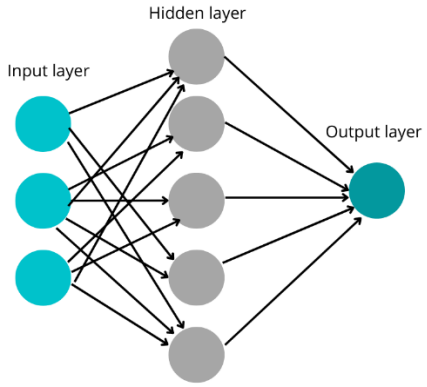


Figure 4 Artificial Neural Network

2.2.3 Chatbot Technology

The rapid growth and improvement of technology convinced many businesses to adopt digital modes of communication rather than traditional ones to engage with consumers. The digital transformation in business is taking place through the implementation of AI technologies such as chatbots.

A chatbot is an automated computer program that simulates and analyses human conversation (spoken or written) to interact with customers, allowing communication between humans and electronic devices.[8]

2.3 Related work

In the following sections, some relevant studies have been briefly summarized. Some of which are related to energy efficiency, some are associated with the chatbot in energy consumption we have choose this papers because the domain of research is like our energy field, and in machine learning domain.

No.	study
1	**Title:** Understanding building energy efficiency with administrative and emerging urban big data by deep learning in Glasgow **Data type:** Image, descriptive variables **Attribute: Not defined** **Model:** Neural network **Accuracy:** Accuracy image 57.2%, 79.5% descriptive features, 86.8% multi-branch **Energy efficiency availability:** Yes

2 **Title:** A Systematic Analysis for Energy Performance Predictions in Residential Buildings Using Ensemble Learning

Data type: Multivariate, Integer ,Real

Attribute: Relative Compactness, Surface Area, Wall Area, Roof Area, Overall Height

,Orientation ,Glazing Area ,Glazing Area Distribution ,Heating Load ,Cooling Load.

Model: MLR, KNN, SVR, RF, GBM, XGBoost

Accuracy: MLR: 88.59 Y1, 85.26 Y2

KNN: 91.91 Y1, 94.47 Y2

SVR: 82.38 Y1, 89.32 Y2

RF: 99.74 Y1, 94.79 Y2

GBM: 99.73 Y1, 96.22 Y2

Boost: 99.75 Y1, 95.94 Y2

Energy efficiency availability: No

3 **Title:** Effective Features to Predict Residential Energy Consumption Using Machine Learning

Data type: Multivariate, Integer ,Real

Attribute: NUMFRIG Number of refrigerators used

SIZRFRI1 Size of most-used refrigerator

ICE Through-the-door ice on most-used refrigerator

SIZFREEZ Size of most-used freezer

OVEN Number of separate ovens

TVCOLOR Number of televisions used

NUMCFAN Number of ceiling fans used

Behaviour MONPOOL Months swimming pool used in the last year

DRYRUSE Frequency clothes dryer used

TVONWE1 Most-used TV usage on weekends

Technology UATYP10 Census 2010 Urban Type

TYPEHUQ Type of housing unit

NCOMBATH Number of full bathrooms

TOTROOMS Total number of rooms in the unit, excluding

bathrooms

UGASHERE Natural gas available in neighborhood

POOL Heated swimming pool

FUELTUB Fuel used for heating hot tub

FUELHEAT Main space heating fuel

AIRCOND Air conditioning equipment used

COOLTYPE Type of air conditioning equipment used

CENACHP Central air conditioner is a heat pump

FUELH2O Fuel used by main water heater

FUELH2O2 Fuel used by secondary water heater

ELWARM Electricity used for space heating

ELWATER Electricity used for water heating

ELFOOD Electricity used for cooking

FOWATER Fuel oil used for water heating

CLIMATE_REGION Building America Climate Zone

Demographic NHSLDMEM Number of household members

NUMADULT Number of household members age 18 or older

NOACBROKE Unable to use cooling equipment in the last year

because equipment was broken/ could not afford

repair or replacement

PERIODNG Number of days covered by Energy Supplier Survey

gas billing data/used to calculate annual consumption

and expenditures

Model: SVM, RF

Accuracy: the selected 32 features reach the 78% of accuracy when predicting home energy consumption and remain 98% of prediction power of the whole 271 features.

Energy efficiency availability: Yes

4 **Title:** ANN-based energy consumption prediction model up to 2050 for a residential building: Towards sustainable decision making

Data type: Real, string

Attribute: atetime_utc _conds _dewptm _fog _hail _heatindexm _hum _precipm _pressurem _rain _snow _tempm _thunder _tornado _vism _wdird _wdire _wgustm _windchillm _wspdm

Model: ANN

Accuracy: ANN-based models have demonstrated greater accuracy in identifying dynamic nonlinear dynamics compared to conventional methods

Energy efficiency availability: No

5 **Title:** Small-Scale Solar Photovoltaic Power Prediction for Residential Load in Saudi Arabia Using Machine Learning

Data type: different data types such as integer, float, and object

Attribute: Rainfall, Rain rate, Solar radiation, Temperature, Ultra violet (UV) radiation dose, Humidity, Dew point

Model: SVM, XGBoost, LASSO, RF, LR, PR

Accuracy: PR was the best after selecting 10 features, SVM was steady after selecting 13 features, and RF the worst in terms with a single feature

Energy efficiency availability: Yes

6 **Title:** Machine learning for estimation of building energy consumption and performance

Data type: Not defined

Attribute: Date, Outdoor temperature, Bedroom, Living temperature, Living over one year, humidity, Bedroom humidity, Outdoor humidity, Water temperature.

Model: ANN, SVM, Gaussian-based regressions and clustering

Accuracy: SVM surpasses ANN in load forecasting, and GB used for only model training with uncertainty assessment.

Energy efficiency availability: No

7 **Title:** Application of neural networks for evaluating energy performance certificates of residential buildings

Data type: Not defined

Attribute: degree days, net volume, net floor area, dispersant surface, opaque to glazed ratio, year of construction, thermal conductivity (walls, windows, roof, basement), average floor height, opaque surface area, glazed surface area, construction period, non-linear features

Model: ANN

	Accuracy: 95% **Energy efficiency availability:** Yes
8	**Title:** Enhancing building energy efficiency using a random forest model :A hybrid prediction approach **Data type:** **Attribute:** A1 Comprehensive heat transfer coefficient of exterior walls A2 Solar radiation absorption coefficient of exterior walls A3 Comprehensive heat transfer coefficient of the roof A4 Solar radiation absorption coefficient of the roof A5 Comprehensive heat transfer coefficient of outer windows A6 Window-wall ratio **Model:** RF, SVM, Bp-ANN **Accuracy:** prediction results with high accuracy and reliability can be obtained by predicting building energy consumption based on the RF prediction model. **Energy efficiency availability:** Yes
9	**Title:** Application of Deep Learning Model in Building Energy Consumption Prediction **Data type:** Not defined **Attribute:** Not defined **Model:** ANN **Accuracy:** Not defined **Energy efficiency availability:** Yes
10	**Title:** A Chatbot Solution for Self-Reading Energy Consumption via Chatting Applications **Data type:** Numerical **Attribute:** **Model:** CNN **Accuracy:** 97% **Energy efficiency availability:** Not defined

Table 1 Related work

In the next section, comparisons are illustrated between related work that is similar to what this project attends to apply and our project.

2.4 Comparison between the proposed system and literature

In this project, we intend to use machine learning frameworks such as SVM, Random Forest, and ANN, which are generally used by researchers due to their reliable estimates and the benefits of overcoming nonlinearity between the input and output of data related to energy[9]. Furthermore, developing a model by using attributes such as total floor area and structural properties will improve the prediction of energy efficiency.

In the following sections, tables compare some published work related to what this project aims to apply.

2.4.1 predict energy efficiency using machine learning.

Using the energy efficiency band attribute tends to result in a more reliable estimation of building energy efficiency. Moreover, understand the relationships between building features and building energy efficiency. A comparison between related published work is given in Table 2 [10] [11] [12] [13] [14] [15] [16] [17] [18].

Paper name	*Algorithm used*	*Similarity*	*Difference*
1. *Understanding building energy efficiency with administrative and emerging urban big data by deep learning in Glasgow.*	ANN	• Both are concerned with energy efficiency. • Use ANN. • Use the same data source.	This paper uses multi-source data and uses images in building energy efficiency prediction with an image model, Descriptive feature model, and multi-branch model. However, we will establish a descriptive feature model.
2. *A Systematic Analysis for Energy Performance Predictions in Residential Buildings Using Ensemble Learning.*	MLR KNN SVM RF GBM XGBoost	Predict the energy consumption of residential buildings. Employ one of the ensemble learning: Random Forest Models, because it produces the optimal predictive model [11]	The paper's models predict heating and cooling loads. in contrast, our model focus on the energy efficiency band.
3. *Effective Features to Predict Residential Energy Consumption Using Machine Learning.*	SVM RF	• Concerned with energy efficiency. • Using SVM and random forest. • using a dataset with a diverse set of features.	This study predicts occupant behavior from energy consumption. In our case, we will concentrate on consumption based on the building's features and area.
4. *ANN-based energy consumption prediction model up to 2050 for a residential building: Towards sustainable decision making.*	ANN & TRNSYS software	• Concerned with energy efficiency. • Using ANN.	The dataset we will be using focuses on the properties of the house rather than external factors like temperature and humidity.
5. *Small-Scale Solar Photovoltaic Power Prediction for Residential Load in Saudi Arabia Using Machine Learning*	SVM XGBoost LASSO RF LR PR	• Concerned with energy. • Using SVM. • Feature elimination to find the most relevant set of features.	The paper predicts the generated power of a photovoltaic system for residential buildings. Therefore, we compute the overall energy consumption of the building and predict its efficiency.
6. *Machine learning for estimation of building energy consumption and performance: a review*	ANN SVM GPR clustering	• Using SVM and ANN. • Building energy efficiency	• The lighting and HVAC systems are used to make predictions in the paper. However, in our case, we predict based on the construction fabric and the area.

			• Use unsupervised learning
7. *Application of neural networks for evaluating energy performance certificates of residential buildings*	ANN	• Energy efficiency of Residential buildings. • Using ANN.	Use the artificial neural network as an efficient and reliable alternative for forecasting heat demand indicators
8. *Enhancing building energy efficiency using a random forest model :A hybrid prediction approach*	RF SVM Bp-ANN	• Building energy efficiency. • Using the same ML technique.	This paper forecasts building energy consumption based on the design of the building covering.
9. *Application of Deep Learning Model in Building Energy Consumption Prediction*	ANN	• concerned with energy efficiency. • Understand relationships between building energy consumption and related variables.	• Use MAPE and RMSE. • uses the prediction time and input parameter dimensions to evaluate the time cost of the prediction model.

Table 2 Comparison between published work using machine learning

2.4.2 Chatbots for energy efficiency.

Based on the energy prediction models that we will develop. The chatbot assists users to understand energy efficiency and its class. Accordingly, we will compare research on chatbots in the energy domain and discuss the similarities and differences and the techniques used. A comparison between related published work is given in Table 3. [19]

Paper name	*Algorithm used*	*Similarity*	*Difference*
10. *A Chatbot Solution for Self-Reading Energy Consumption via Chatting Applications*	image processing	• Energy domain • Chatbot service	The main difference is using an image database for reading and recognition of an energy digital meter.

Table 3 Comparison between published work of chatbots

2.5 Conclusion

In this chapter, we discussed energy efficiency through machine learning and the chatbot that will assist consumers in estimating the energy efficiency of their homes. As an introduction, we discussed previously estimated energy efficiency calculations and background. Then, a brief overview of machine learning techniques and models is provided, followed by a comparison of our idea to related works.

Chapter III: Data Understanding

3.1 Introduction

The exploratory analysis represents brainstorming for data and the relation between features. In our project we analyzed the data using visual techniques and used to discover different data information with the help of statistical summary and graphical representations.

The main goal of our project is predicting of energy efficiency ratings in buildings. We aim to build a chatbot that can evaluate the energy efficiency rating of the houses based on some variables by using machine learning models to help the consumers evaluate the house from aspect of energy efficiency by calculate some variable of data based on some columns. To achieve this goal, we used some techniques to collect datasets laid out in Section 3.2, more details on each dataset that will be used in Section 3.3. Exploratory data analytics techniques that will be employed to give general information about datasets in use are presented in Section 3.4.

3.2 Data collection

Data scientists use data that has been collected using a variety of data collecting techniques when doing data analysis tasks. The most popular strategies are primary data collection which is original data that has been collected for a specific research objective and Secondary data collection which refers to the data that was originally collected for a different purpose and can be reused for another research question [1]. In our case, we chose a secondary data collection strategy for several reasons, including the fact that secondary data is easier to obtain than primary data because we do not have to collect the data by ourselves, and secondary data often have been prepped and validated statistically to be used immediately.

The source of the dataset is The Scottish Energy Performance Certificate Register SEPCR. This data is lodged to the SEPCR by Energy Performance Certificate (EPC) Assessors registered with one of the Approved Organizations appointed by Scottish Ministers. It presents data from every valid Domestic EPC assessment held by the Scottish EPC Register (SEPCR) from commencement of central lodgment of the current EPC format to the SEPCR in October 2012 to the end of the last full lodgment year. The data was extracted from the register on 15 June 2021.

We choose this dataset because it has many features that will help us in prediction to get the best accuracy.

3.3 Data description

We utilized a secondary data collection technique to find related data to match the project's purposes. Furthermore, the EPC dataset is split into four files, each containing 49999 records from October 2012 to the present. It is critical to understand the features and their relations.

The section that follows summarizes the data in a meaningful way that allows us to generate insights from it. Thereafter then determines the dataset's quality.

3.3.1 Data Dictionary

Table 5 presents the metadata for 103 columns of EPC data submitted between October 2012 and the present based on energy performance data for local buildings published via the Scottish Energy Performance Certificate Register [2]. Additionally, it covers features, descriptions, data types, and statistics.

No.	*features*	*types*	*descriptions*	*statistics*
1	Property UPRN	discrete	The Unique Property Reference Number is a 10-digit identifier for the property located in the UK.	Average:1053731830 Min:1000001636 Max:1235961347
2	OSG_UPRN	discrete	The Unique digit property reference number is assigned to a building by a local authority.	Average:137276434353 Min:35000065 Max:906701000000
3	ADDRESS1	nominal	building address, and property type.	Most: Flat 1/1 Least: Zamalek Range: 43061 Value
4	ADDRESS2	nominal	building address, the street name.	Most: East Kilbride Least:gedintailor Range:13253 value
5	POST_TOWN	nominal	building address, the town.	Most:GLASGOW Least: by Cumnock Range:682 value
6	POSTCODE	nominal	The postcode of the property.	Most: EH3 8FU Last: ZE3 9JS Range:32275 value
7	Date of Assessment	continues	Date on which assessment of the energy performance of the building was undertaken	Start from 05/06/2015 To 31/03/2022
8	Type of Assessment	nominal	Tool used (SAP, RdSAP) and whether assessment is for new or existing dwelling.	Most: RdSAP, existing dwelling Last: SAP , existing dwelling
9	Date of Certificate	continues	Date on which the data from the energy performance assessment was lodged to the Scottish EPC Register	Start from 31/ 01/01/2022 To 31/03/2022
10	Total floor area	continues	The total floor area of the buildings m².	Average:88 Min: 15 Max:1310
11	Improvements	nominal	A list of the improvement measures identified as relevant to the building at the date of assessment.	Most: Solar water heating; Indicative Cost: Â£4,000 - Â£6,000 Last: Indicative Cost: Â£4,000 - Â£6,000 Range:43858 value
12	Primary Energy Indicator (kWh/m2/year)	continues	The quantity of energy needed at the source, before conversion, to determine the home's energy needs	Average:249 Min: -255 Max:5324
13	Current energy efficiency rating	discrete	The energy cost of the building is determined by considering both the building's energy efficiency and the price of fuels used	Average:69 Min: 1 Max:158
14	Current energy efficiency rating Band	ordinal	Rating of current energy on a Scale of A to G, with an A (best) rating band.	Most: C Least: G Range: 7 value
15	Potential energy Efficiency Rating	discrete	The energy cost of the building after applying the improvement measures.	Average:82 Min: 14 Max:160
16	Potential Energy Efficiency Rating Band	ordinal	Rating of potential energy on a Scale of A to G, with an A (best) rating band.	Most: B Least: G Range: 7 value
17	Current Environmental Impact Rating	discrete	The current environmental impact of the building is based on emissions.	Average:67 Min: 1 Max:153
18	Current Environmental Impact Rating band	ordinal	On a scale of A to G, the current environmental impact rating is expressed.	Most: C Least: G Range: 7 value
19	Potential Environmental Impact Rating	discrete	The potential environmental impact of a listed building, as calculated using emissions associated with the building's energy use.	Average:80 Min: 19 Max:154

20	Potential Environmental Impact Rating Band	ordinal	On a scale of A to G, the current environmental impact rating is expressed.	Most: B Least: G Range: 7 value
21	WALL_DESCRIPTION	nominal	Building wall component.	Most: Cavity wall, filled cavity Least: Timber frame Range: 617 value
22	WALL_ENERGY_EFF	ordinal	Energy Efficiency Rating scale of 1 to 5 (very poor, poor, average, good, very good).	Most: Good Least: Very poor| Poor Range: 5 value
23	WALL_ENV_EFF	ordinal	Environmental Efficiency Rating scale of 1 to 5 (very poor, poor, average, good, very good).	Most: Good Least: Very poor| Poor Range: 5 value
24	ROOF_DESCRIPTION	nominal	Describes the roof of the dwelling.	Most:another dwelling above Least: Thatched Range: 845 value
25	ROOF_ENERGY_EFF	ordinal	Energy Efficiency Rating scale of 1 to 5 (very poor, poor, average, good, very good).	Most: Good Least: Very poor| Good Range: 5 value
26	ROOF_ENV_EFF	ordinal	Environmental Efficiency Rating scale of 1 to 5 (very poor, poor, average, good, very good).	Most: Good Least: Very poor| Good Range: 5 value
27	FLOOR_DESCRIPTION	nominal	Describes the floor of the dwelling.	Most: Suspended, no insulation Least: To external air Range: 205 value
28	FLOOR_ENERGY_EFF	ordinal	Energy Efficiency Rating scale of 1 to 5 (very poor, poor, average, good, very good).	Most: null Least: Very poor Range: 5 value
29	FLOOR_ENV_EFF	ordinal	Environmental Efficiency Rating scale of 1 to 5 (very poor, poor, average, good, very good).	Most: null Least: Very poor Range: 5 value
30	WINDOWS_DESCRIPTION	nominal	Describes windows of the dwelling.	Most: Fully double glazed Least: Some multiple glazing Range: 17 value
31	WINDOWS _ENERGY_EFF	ordinal	Energy Efficiency Rating scale of 1 to 5 (very poor, poor, average, good, very good).	Most: Average Least: Poor Range: 5 value
32	WINDOWS _ENV_EFF	ordinal	Environmental Efficiency Rating scale of 1 to 5 (very poor, poor, average, good, very good).	Most: Average Least: Poor Range: 5 value
33	MAINHEAT_DESCRIPTION	nominal	Describes the central heating system of the dwelling.	Most: Boiler and radiators Least: Water source heat pump Range: 122 value
34	MAINHEAT_ENERGY_EFF	ordinal	Energy Efficiency Rating scale of 1 to 5 (very poor, poor, average, good, very good).	Most: Good Least: Very poor |Average Range: 5 value
35	MAINHEAT_ENV_EFF	ordinal	Environmental Efficiency Rating scale of 1 to 5 (very poor, poor, average, good, very good).	Most: Good Least: Very poor |Good Range: 5 value
36	MAINHEATCONT_DESCRIPTION	nominal	Describes main heating system controls of dwelling.	Most: Programmer, room thermostat and TRVs Least: Time and temperature zone control | TRVs and bypass Range: 92 value
37	MAINHEATC_ENERGY_EFF	ordinal	Energy Efficiency Rating scale of 1 to 5 (very poor, poor, average, good, very good).	Most: Good Least: Very poor Range: 5 value
38	MAINHEATC_ENV_EFF	ordinal	Environmental Efficiency Rating scale of 1 to 5 (very poor, poor, average, good, very good).	Most: Good Least: Very poor Range: 5 value
39	SECONDHEAT_DESCRIPTION	nominal	Describes secondary heating of the dwelling.	Most: none Least: Room heaters Range: 16 value
40	SHEATING_ENERGY_EFF	ordinal	Energy Efficiency Rating scale of 1 to 5 (very poor, poor, average, good, very good).	Most: null Least: null Range: null
41	SHEATING_ENV_EFF	ordinal	Environmental Efficiency Rating scale of 1 to 5 (very poor, poor, average, good, very good).	Most: null Least: null Range: null
42	HOTWATER_DESCRIPTION	nominal	Describes the hot water system of the dwelling.	Most: Main system Least: Solid fuel Range: 39 value

43	HOT_WATER_ENERGY_EFF	ordinal	Energy Efficiency Rating scale of 1 to 5 (very poor, poor, average, good, very good).	Most: Good Least: Very good Range: 5 value
44	HOT_WATER_ENV_EFF	ordinal	Environmental Efficiency Rating scale of 1 to 5 (very poor, poor, average, good, very good).	Most: Good Least: Very Poor Range: 5 value
45	LIGHTING_DESCRIPTION	nominal	Describes the lighting in a property	Most: low energy lighting in all fixed energy Least: low energy lighting in 34% of fixed energy Range: 99 value
46	LIGHTING_ENERGY_EFF	ordinal	Energy Efficiency Rating scale of 1 to 5 (very poor, poor, average, good, very good).	Most: Very good Least: Poor Range: 5 value
47	LIGHTING_ENV_EFF	ordinal	Environmental Efficiency Rating scale of 1 to 5 (very poor, poor, average, good, very good).	Most: Very good Least: Poor Range: 5 value
48	AIR_TIGHTNESS_DESCRIPTION	nominal	Describes the building's airtightness.	Most:5.0 assumed Least: 9,4 tested Range: 157 value
49	AIR_TIGHTNESS_ENERGY_EFF	ordinal	Energy Efficiency Rating scale of 1 to 5 (very poor, poor, average, good, very good).	Most: Good Least: Poor Range: 5 value
50	AIR_TIGHTNESS_ENV_EFF	ordinal	Environmental Efficiency Rating scale of 1 to 5 (very poor, poor, average, good, very good).	Most: Good Least: Poor Range: 5 value
51	CO2 Emissions Current Per Floor Area (kg.CO2/m2/yr)	continues	The building's total annual emissions reduction, heating, cooling, lighting, and ventilation.	Average:44.30 Min: -43 Max:816
52	Current Emissions (T.CO2/yr)	continues	Total yearly emissions for the building of the identified improvement measures are reduced.	Average:9826 Min: -11.200 Max:134
53	Potential Reduction in Emissions (T.CO2/yr)	continues	the Total expected reduction in CO2 emissions.	Average:1.550 Min: -1 Max:102
54	Total current energy costs over 3 years	continues	The total energy cost in general over the stated period.	Average:2686 Min: 552 Max:158220
55	Potential future savings over 3 years	continues	The total future saving over the stated period.	Average:696 Min: 95016 Max:-3
56	Current heating costs over 3 years	continues	The total energy expense for heating during the stated time.	Average:2026 Min: 186 Max:156276
57	Potential heating costs over 3 years	continues	The overall cost of energy of heating during the specified time with improvement measures.	Average:1488 Min: 123 Max:61485
58	Current hot water costs over 3 years	continues	The total energy expense for heating water during the specified time.	Average:407 Min: 0 Max:3939
59	Potential hot water costs over 3 years	continues	The overall energy expense for heating water during the specified time with improvement measures.	Average:282 Min: 0 Max:3521
60	Current lighting costs over 3 years	continues	The overall energy expense for lighting during the specified time.	Average:252 Min: 60 Max: 1848
61	Potential lighting costs over 3 years	continues	The overall energy expense for lighting during the specified time improvement measures.	Average:218 Min: 60 Max:1266
62	Alternative Measures	nominal	List of potential improvements to the building that can be considered.	Most: External insulation Least: Micro CHP Range: 39 value
63	LZC Energy Source	nominal	Low or zero-carbon technology is present in the dwelling. Solar thermal and photovoltaic panels are two examples.	Most: solar photovoltaics Least: solar water heating Range:73 value
64	Space Heating	continues	Calculated heat demand for the current home's space heating.	Average:9826 Min: 0 Max:352110
65	Water Heating	continues	Calculated heat demand for the current dwelling supply of hot water according to the water heating system.	Average:2164 Min:0 Max:9965

66	Impact Of Loft Insulation	continues	If the dwelling loft is insulated, the calculated heat demand for space heating may be reduced.	Average: -394 Min: -68272 Max:0
67	Impact Of Cavity Wall Insulation	continues	If the dwelling cavity walls are insulated, the calculated heat demand for space heating may be reduced.	Average: -228 Min: -11550 Max:0
68	Impact Of Solid Wall Insulation	continues	If external or internal insulation is applied to the dwelling walls, the calculated heat demand for space heating may be reduced.	Average : -747 Min: -156787 Max:0
69	Addendum Text	-	Used to record a range of specific observations about the dwelling construction that are relevant to the assessment. Current valid options are listed in Section S15 in Appendix S of the current SAP 2012 specification.	-
70	Part 1 Construction Age Band	nominal	Age band when building part constructed. For Scotland.	Most: before1919 Least: 1919-1929 Range: 11 value
71	Part 1 Floor 0 Room Height	Continues	Average height of the lowest storey of the dwelling (Units: m)	Average: 2.50 Min: 0 Max: 8.26
72	Data Zone	nominal	Data Zone code and name - Data zones are the key geography for the dissemination of small area statistics in Scotland.	Most: S01004730 (Kirkshaws) Least: S01006501 (Linlithgow North) Range: 6411 value
73	Energy Consumption Potential	discrete	Reports Primary Energy - the amount of energy required at source, before conversion and transmission, to meet the calculated energy demand of the dwelling, if recommended improvement measures are carried out (Units: kWh/m2/year).	Average: 150 Min: -396 Max: 1125
74	Extensions Count	Discrete	The number of extensions added to the property. Between 0 and 4 can be reported.	Average: 0.24 Min: 0 Max: 4
75	Fixed Lighting Outlets Count	discrete	The number of fixed lighting outlets in the dwelling.	Average:10.8 Min: 1 Max: 322
76	Low Energy Lighting Outlets Count	discrete	The number of low-energy fixed lighting outlets.	Average: 8.90 Min: 0 Max: 322
77	Low Energy Lighting %	discrete	The percentage of low energy lighting present in the property as a percentage of the total fixed lights in the property. 0% indicates that no low-energy lighting is present.	Average: 81.07 Min: 0 Max: 100
78	Flat Level	nominal	Floor level relative to the lowest level of the building.	Most: top floor Last: basement Range: 4 value
79	Flat Location	discrete	The number of storeys in the flatted block.	Min: -1 Max: 18
80	Glazed Area	discrete	Ranged estimate of the total glazed area of the Habitable Area.	Average: 0.9 Min: 0 Max: 5
81	Habitable Room Count	discrete	Habitable rooms include any living room, sitting room, dining room, bedroom, study and similar. Excludes any room used solely as a kitchen, utility room, sanitary accommodation and any hallway, stairs or landing; and also any conservatory or room without a window.	Average: 3.44 Min: 0 Max: 73
82	Heat Loss Corridor	nominal	Flats and maisonettes only. Indicates that the flat contains a corridor through which heat is lost. Heat loss corridor, one of: no corridor; heated corridor; unheated corridor.	Most: unheated corridor Least: heated corridor Range:3 value
83	Heated Room Count	discrete	The number of heated habitable rooms in the property.	Average: 3.39 Min: 0 Max: 31
84	Local Authority	nominal	The name of the local authority area in which the building is located.	Most: Glasgow City Least: Eilean Siar Range:32 value

85	Main Gas	-	Whether mains gas is available. Yes means that there is a gas meter or a gas-burning appliance in the dwelling	-
86	Main Heating 1 Category	nominal	Reports the type of main heating system present in the dwelling. Lists both heat source and heat emitters (e.g. gas boiler and radiators).	Most: boiler with radiators or underfloor heating Least: other systems Range:10 value
87	Main Heating 1 Fuel Type	nominal	Reports the fuel type used by the main heating system present in the dwelling.	Most: mains gas Least: coal Range:31 value
88	Main Heating 1 Control	discrete	Type of main heating controls. Includes both main heating systems if there are two.	Average: 2163 Min: 2101 Max: 2706
89	Mechanical Ventilation	nominal	Identifies the type of mechanical ventilation the property has.	Most: natural Least: mechanical, extract only Range: 3 value
90	Meter Type	nominal	Type of electricity tariff for the property (by meter type), e.g. single.	Most: single Least:off-peak 18 hour Range: 5 value
91	Multiple Glazed Proportion	discrete	The estimated banded range (e.g. 0% - 10%) of the total glazed area of the Property that is multiple glazed.	Average: 81.7 Min: 0 Max: 100
92	Multiple Glazing Type	nominal	The type of glazing. From British Fenestration Rating Council or manufacturer declaration, one of; single; double; triple.	Most: double glazing installed during or after 2002 Least: single glazing Range: 9 value
93	Open Fireplaces Count	discrete	The number of Open Fireplaces in the Property. An Open Fireplace is a fireplace that still allows air to pass between the inside of the Property and the outside.	Average: 0.07 Min: 0 Max: 14
94	Photovoltaic Supply	nominal	Photovoltaic area as a percentage of total roof area. 0% indicates that a Photovoltaic Supply is not present in the property. Note: multiple arrays can be reported. Each array reported is separated by a vertical line ["|" - ASCII (decimal) 124]	Most:Roof Area: 0%; Connection Least:Roof Area: 95%; Connection Range:1184 value
95	Solar Water Heating	nominal 'binary'	Indicates whether the water heating in the property is served by solar thermal panels.	Most: n Least:y Range: 2 value
96	Tenure	nominal	Describes the tenure type of the property. One of: Owner- occupied; Rented (social); Rented (private); Unknown	Most: owner-occupied Least:unknown Range:4 value
97	Transaction Type	nominal	Type of transaction or purpose that triggered EPC assessment. One of: - new dwelling; marketed sale; non-marketed sale; rental; not sale or rental; assessment for Green Deal; following Green Deal; FIT application; RHI application; ECO assessment; none of the above. Where the reason for the assessment is unknown by the energy assessor the transaction type will be recorded as 'none of the above'.	Most: marketed sale Least: assessment for green deal Range: 9 value
98	Unheated Corridor Length	Continues	The total length of unheated corridor in the flat. Only populated if flat or maisonette contains unheated corridor. If unheated corridor, length of sheltered wall (m2).	Average: 6.74 Min: 0 Max: 35.5
99	Ward Code	nominal	Code for Parliamentary constituency in which the building is located. Scottish area defined as 'Ward', not 'constituency'	Most: 00QSMH Least: 00RJME Range: 353 value
100	Ward Name	nominal	Name of Parliamentary constituency in which the building is located. Scottish area defined as 'Ward', not 'constituency'	Most: Southside Central Least:Sgire an Rubha Range: 351 value
101	Wind Turbines Count	discrete	Number of wind turbines; 0 if none.	Average: 0.0008 Min: 0 Max: 2
102	Built Form	nominal	The building type of the Property e.g. Detached, Semi-Detached, Terrace etc. Together with the Property Type, the Build Form produces a structured description of the property	Most: Semi-Detached Least:Enclosed End-Terrace Range:6 value
103	Property Type	nominal	Describes the type of property such as House, Flat, Maisonette etc. This is the type differentiator for dwellings.	Most: house Least:park home Range:5 value
104	Data Zone 2011	nominal	Data Zone code and name - Data zones are the key geography for the dissemination of small area statistics in Scotland.	Most: S01006586 (City Centre West - 05) Least: S01013476 (Broxburn South - 05) Range: 6821value

Table 4 EPC metadata

3.3.2 Data Quality

The assessment of how well-suited, reliable, and valid an EPC dataset is to the project's requirement. measure data quality based on accuracy, completeness, consistency, and timeliness, relevancy.

	applicable	inapplicable	Explanation
Accuracy	✓		The dataset was obtained from a reliable website.
Relevancy	✓		The dataset provides numerous relevant features to satisfy the need, including Total floor area, Flat Location, and Current energy efficiency rating.
Completeness	✓		The missing value in the columns of datasets relating to the project topic is less than 20%.
Timeliness	✓		According to statistics.gov.scot [2], the dataset was last modified on August 8, 2022.
Consistency	✓		Apply a certain format to several columns, such as Total floor area (m2), and ensure it is in the range value 15; where the value >1500, To eliminate errors in records.

Table 5 Assessment EPC data table

3.4 Exploratory data analytics

Exploratory Data Analysis is an important step in our Data Science project. We will be investigating the dataset to discover correlation, and anomalies (outliers) based on our understanding of the dataset. EDA involves generating summary statistics for numerical data in the dataset and creating various graphical representations to understand the data better.

Shape of the dataset: (54495, 104)

Numerical features statistics

1		count	std	25%	50%	75%
2	Property_UPRN	5.449500e+04	9.766713e+07	1.000768e+09	1.001577e+09	1.002503e+09
3	OSG_UPRN	5.029100e+04	3.196190e+11	1.260232e+08	1.510111e+08	9.051024e+09
4	Primary Energy Indicator (kWh/m²/year)	54495.00000	148.58805	160.00000	224.00000	302.00000
5	Total floor area (m²)	54495.000000	50.870847	62.000000	77.000000	98.000000
6	Total current energy costs over 3 years (£)	54495.000000	2195.474796	1527.000000	2115.000000	3114.000000
7	Potential future savings over 3 years (£)	54495.000000	1245.611202	114.000000	333.000000	777.000000
8	Current energy efficiency rating	54495.000000	13.585607	64.000000	71.000000	78.000000
9	Potential Energy Efficiency Rating	54495.000000	8.111226	78.000000	82.000000	87.000000
10	Current Environmental Impact Rating	54495.000000	16.167127	58.000000	70.000000	79.000000
11	Potential Environmental Impact Rating	54495.000000	10.503293	76.000000	82.000000	87.000000
12	CO2 Emissions Current Per Floor Area (kg.CO2/m²/yr)	54495.0000	26.621238	28.000000	40.000000	54.000000
13	Current Emissions (T.CO2/yr)	54495.000000	3.182152	2.000000	3.000000	4.500000
14	Potential Reduction in Emissions (T.CO2/yr)	54495.000000	2.026069	0.300000	1.100000	1.900000
15	Current heating costs over 3 years (£)	54495.000000	1998.885559	1035.000000	1524.000000	2337.000000
16	Potential heating costs over 3 years (£)	54495.000000	1157.635278	864.000000	1218.000000	1731.000000
17	Current hot water costs over 3 years (£)	54495.000000	312.168098	234.000000	279.000000	450.000000
18	Potential hot water costs over 3 years (£)	54495.000000	165.480073	192.000000	231.000000	294.000000
19	Current lighting costs over 3 years (£)	54495.000000	99.439179	186.000000	234.000000	300.000000
20	Potential lighting costs over 3 years (£)	54495.000000	73.717889	174.000000	210.000000	252.000000
21	Space Heating	54495.000000	8664.261726	4637.000000	7795.000000	12153.000000
22	Water Heating	54495.000000	629.558074	1800.000000	2028.000000	2293.000000
23	Impact Of Loft Insulation	54495.000000	1336.151595	0.000000	0.000000	0.000000
24	Impact Of Cavity Wall Insulation	54495.000000	692.398820	0.000000	0.000000	0.000000
25	Impact Of Solid Wall Insulation	54495.000000	2043.334188	0.000000	0.000000	0.000000
26	Addendum Text	0.0	NaN	NaN	NaN	NaN
27	Part 1 Floor 0 Room Height	53205.000000	0.266045	2.360000	2.400000	2.503714
28	Energy Consumption Potential	54495.000000	89.762715	99.000000	135.000000	179.000000
29	Extensions Count	54495.000000	0.579185	0.000000	0.000000	0.000000
30	Fixed Lighting Outlets Count	54495.000000	7.680630	7.000000	9.000000	12.000000
31	Low Energy Lighting Outlets Count	54495.000000	7.701098	5.000000	8.000000	10.000000
32	Low Energy Lighting %	54495.000000	28.158240	67.000000	100.000000	100.000000
33	Glazed Area	54495.000000	0.654936	1.000000	1.000000	1.000000
34	Habitable Room Count	54495.000000	1.988568	2.000000	3.000000	4.000000
35	Heated Room Count	54495.000000	1.955287	2.000000	3.000000	4.000000
36	Main Gas	0.0	NaN	NaN	NaN	NaN
37	Main Heating 1 Control	54495.000000	132.188172	2106.000000	2106.000000	2110.000000
38	Multiple Glazed Proportion	54495.000000	37.967002	100.000000	100.000000	100.000000
39	Open Fireplaces Count	54495.000000	0.357247	0.000000	0.000000	0.000000
40	Unheated Corridor Length	15346.000000	2.969044	5.000000	6.500000	8.220000
41	Wind Turbines Count	54495.000000	0.030280	0.000000	0.000000	0.000000

Table 6 Numerical features statistics Table Numerical features statistics Table

- There are no duplicated rows

```
1  epc_all.duplicated().sum()

   0
```

Figure 5 Number of Duplicated row

- Percentage of Null in each column

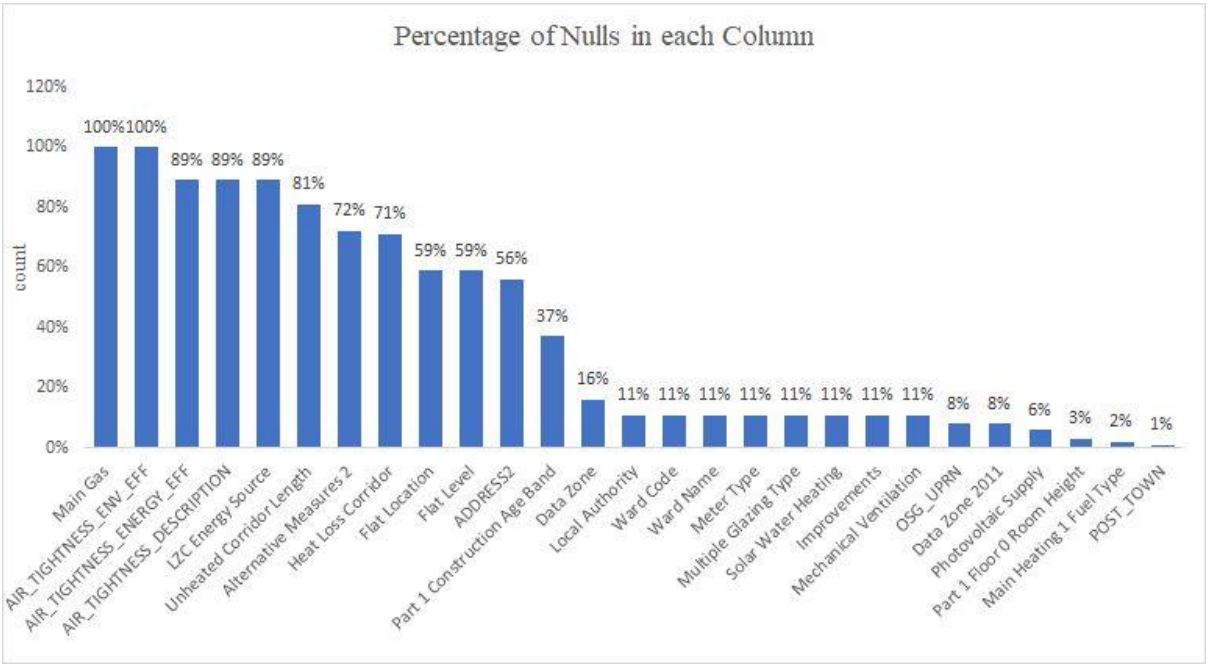


Figure 6 percentage of Null in the columns

- Unique values in categorical columns

Post_town

```
epc_all['POST_TOWN'].unique()
```

	0
0	FALKIRK
1	DUNFERMLINE
2	PERTH
3	GLASGOW
4	TURRIFF
5	ABERDEEN
6	LARKHALL
7	ANNAN

683 rows × 1 columns Open in new tab

Figure 7 Unique value in post_town column

Type of assessment

```
epc_all['Type of Assessment'].unique()
```

	0
0	RdSAP, existing dwelling
1	SAP, new dwelling
2	SAP, existing dwelling

3 rows × 1 columns Open in new tab

Figure 8 Unique value in Type of assessment column

Enery Efficiency rating band

```
epc_all['Current energy efficiency rating band'].unique()
```

	0
0	E
1	C
2	D
3	B
4	F
5	G
6	A

7 rows × 1 columns Open in new tab

Figure 9 Unique value in Current energy efficiency rating band column

Wall description

```
epc_all['WALL_DESCRIPTION'].unique()
```

	0
0	Sandstone or limestone, as built, no insulation (assumed) | Solid brick, as bui...
1	Cavity wall, filled cavity
2	Sandstone or limestone, as built, no insulation (assumed)
3	Cavity wall, as built, insulated (assumed)
4	Granite or whinstone, with internal insulation
5	Granite or whinstone, as built, no insulation (assumed)
6	Sandstone or limestone, as built, no insulation (assumed) | Timber frame, as bu...
7	Cavity wall, filled cavity | Sandstone or limestone, as built, no insulation (a...

617 rows × 1 columns Open in new tab

Figure 10 Unique value in wall_description column

```
epc_all['WALL_ENERGY_EFF'].unique()
```

	0
0	Poor | Average
1	Good
2	Poor
3	Poor | Good
4	Average | Poor
5	Poor | Poor
6	Average
7	Average | Very Good

130 rows × 1 columns Open in new tab

Figure 11 Unique value in wall_energy_eff

LZC Energy Source

```
epc_all['LZC Energy Source'].unique()
```

	0
0	NaN
1	Description: Air source heat pump |
2	Description: Solar photovoltaics |
3	escription: Biomass secondary heating | Description: Air source heat pump | Description...
4	Description: Biomass secondary heating |
5	escription: Biomass secondary heating | Description: Solar water heating | Description:...
6	escription: Biomass main heating | Description: Biomass secondary heating | Description...
7	Description: Air source heat pump | Description: Solar photovoltaics |

74 rows × 1 columns Open in new tab

Figure 12 Unique value in LZC Energy Source columns

```
epc_all['Part 1 Construction Age Band'].unique()
```

	0
0	before 1919
1	2003-2007
2	1999-2002
3	1950-1964
4	1965-1975
5	1976-1983
6	1930-1949
7	1984-1991

12 rows × 1 columns Open in new tab

Figure 13 Unique value in part 1 construction Age Band columns

Flat level

```
epc_all['Flat Level'].unique()
```

	0
0	top floor
1	NaN
2	ground floor
3	mid floor
4	basement

5 rows × 1 columns Open in new tab

Figure 14 Unique value in flat Level column

Property type

```
epc_all['Property Type'].unique()
```

	0
0	Flat
1	House
2	Bungalow
3	Maisonette
4	Park home

5 rows × 1 columns Open in new tab

Figure 15 Unique value in Property Type

```
epc_all['Property Type'].value_counts()
```

	Property Type
House	24223
Flat	23002
Bungalow	6228
Maisonette	1006
Park home	36

Length: 5, dtype: int64 Open in new tab

Figure 16 Mode in Property Type column

Most properties are houses

Multiple glazing type

```
epc_all['Multiple Glazing Type'].value_counts()
```

	Multiple Glazing Type
double glazing installed during or after 2002	18393
double glazing installed before 2002	13988
double glazing, unknown install date	11955
not defined	3077
triple glazing	545
secondary glazing	221
double, known data	77
triple, known data	48

Length: 9, dtype: int64 Open in new tab

Figure 17 Mode in Multiple Glazing Type column

Most have double glazing installed during or after 2002

```
epc_all['Flat Level'].value_counts()
```

	Flat Level
top floor	8630
mid floor	7730
ground floor	7486
basement	162

Length: 4, dtype: int64 Open in new tab

Figure 18 Mode in Flat Level column

Most are in top floor

```
epc_all['Type of Assessment'].value_counts()
```

	Type of Assessment
RdSAP, existing dwelling	48306
SAP, new dwelling	6181
SAP, existing dwelling	8

Length: 3, dtype: int64 Open in new tab

Figure 19 Mode in Type of Assessment column

RdSAP, existing dwelling, is used the most.

Take a look on properties with best energy efficiency rating and worst energy efficiency rating

```
epc_all[epc_all['Current energy efficiency rating band']=='A'].head()
```

	Current energy efficiency rating	Current energy efficiency rating band	Potential Energy Efficiency Rating	Po
)	99	A	100	
)	100	A	104	
)	95	A	99	
)	95	A	98	
)	94	A	98	

5 rows × 104 columns Open in new tab

```
epc_all[epc_all['Current energy efficiency rating band']=='G'].head()
```

	Current energy efficiency rating	Current energy efficiency rating band	Potential Energy Efficiency Rating	F
.0	18	G	75	
.0	18	G	49	
.0	17	G	50	
.0	5	G	64	
.0	15	G	79	

5 rows × 104 columns Open in new tab

```
epc_all[epc_all['Current energy efficiency rating band']=='A'].head()
```

	Primary Energy Indicator (kWh/m²/year)	Total floor area (m²)	Total current energy costs over 3 years (£)
1	38.0	176	2337.0
1	29.0	202	5136.0
1	33.0	165	4326.0
2	392.0	50	2820.0
2	446.0	49	3033.0

5 rows × 104 columns Open in new tab

```
epc_all[epc_all['Current energy efficiency rating band']=='G'].head()
```

	Primary Energy Indicator (kWh/m²/year)	Total floor area (m²)	Total current energy costs over 3 years (£)	F
)	955.0	102	11463.0	
)	893.0	33	5454.0	
)	765.0	1001	89268.0	
)	593.0	172	19326.0	
)	640.0	68	8313.0	

5 rows × 104 columns Open in new tab

Figure 20 Relation between the total floor area and primary energy indicator

Carbon dioxide emission

```
ciency rating band']=='A'].head()
```

CO2 Emissions Current Per Floor Area (kg.CO2/m²/yr)	
7.0	)
5.0	)
6.0	)
66.0	)
75.0	)

```
ciency rating band']=='G'].head()
```

CO2 Emissions Current Per Floor Area (kg.CO2/m²/yr)
161.0
151.0
133.0
100.0
108.0

Figure 21 Carbon dioxide emission inverse relationship with current energy efficiency rating band

```
epc_all[epc_all['Current energy efficiency rating band']=='A'].head()
```

WALL_ENERGY_EFF	WALL_ENV_EFF	ROOF_DESCRIPTION	ROOF_ENERGY_EFF	ROOF_ENV_EFF	FLOOR_DESCRIPTION
Very Good	Very Good	Pitched, 250 mm loft insulation	Good	Good	Suspended, insulated (assumed)
Good | Very Good	Good | Very Good	Pitched, 300 mm loft insulation	Very Good	Very Good	Suspended, limited insulation (assumed)
Good	Good	Roof room(s), insulated (assumed)	Good	Good	Solid, insulated (assumed)
Good	Good	Pitched, 300 mm loft insulation	Very Good	Very Good	Suspended, insulated
Good	Good	Pitched, 300 mm loft insulation	Very Good	Very Good	Suspended, insulated

5 rows × 104 columns Open in new tab

```
epc_all[epc_all['Current energy efficiency rating band']=='G'].head()
```

WALL_ENERGY_EFF	WALL_ENV_EFF	ROOF_DESCRIPTION	ROOF_ENERGY_EFF	ROOF_ENV_EFF	FLOOR_DESCRIPTION
Poor | Poor	Poor | Poor	Roof room(s), ceiling insulated	Poor	Poor	Suspended, no insu
Very Poor	Very Poor	Pitched, no insulation (assumed)	Very Poor	Very Poor	(another
Poor	Poor	Pitched, no insulation (assumed) | Roof room(...	Very Poor | Very Poor	Very Poor | Very Poor	Solid, no insu
Poor	Poor	Pitched, no insulation (assumed)	Very Poor	Very Poor	Suspended, no insu
Poor | Average	Poor | Average	Pitched, no insulation	Very Poor	Very Poor	(another

Figure 22 current energy efficiency rating band positive relationships

- Outliers

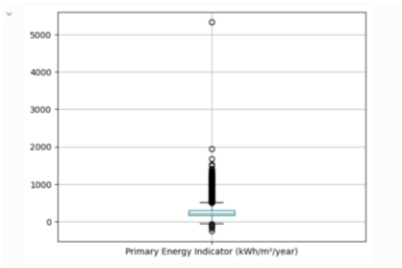


Figure 23 Box plot for outlier detection

```
epc_all[epc_all['Primary Energy Indicator (kWh/m²/year)']>5000].head()
```

ertificate	Primary Energy Indicator (kWh/m²/year)	Total floor area (m²)	Total current energy costs over 3 years (£)	Potential future savings
2022-02-10	5324.0	119	61743.0	

Figure 24

There is one row where 'Primary Energy Indicator' is > 5000

There are 1869 rows where 'Total floor area' is > 200

```
len(epc_all[epc_all['Total floor area (m²)']>200])
```

```
1869
```

Figure 25

And the rest 52592 has 'Total floor area' less than 200

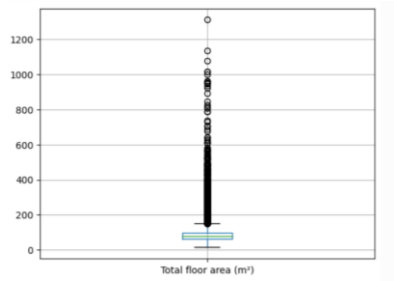


Figure 28 Box plot for outlier detection of Total floor area(m2)

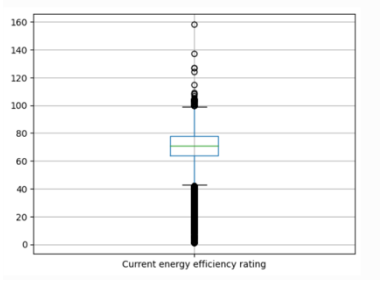


Figure 27 Box plot for outlier detection of current energy efficiency rating

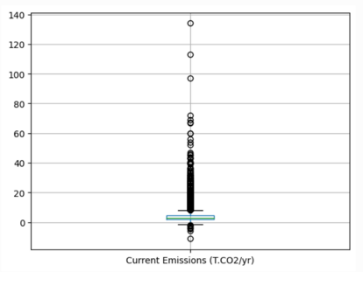


Figure 26 Box plot for outlier detection of current Emissions(T.CO2/yr)

- Negative correlation

Current Environmental Impact Rating	CO2 Emissions Current Per Floor Area (kg.CO2/mÂ²/yr)	-0.924
Primary Energy Indicator (kWh/mÂ²/year)	Current Environmental Impact Rating	-0.883
Current energy efficiency rating	CO2 Emissions Current Per Floor Area (kg.CO2/mÂ²/yr)	-0.863
Primary Energy Indicator (kWh/mÂ²/year)	Current energy efficiency rating	-0.841
Current Environmental Impact Rating	Current Emissions (T.CO2/yr)	-0.743
Current energy efficiency rating	Current Emissions (T.CO2/yr)	-0.689
Current energy efficiency rating	Current heating costs over 3 years (Â£)	-0.641
Current Environmental Impact Rating	Current heating costs over 3 years (Â£)	-0.588
Current Environmental Impact Rating	Energy Consumption Potential	-0.583
Current lighting costs over 3 years (Â£)	Low Energy Lighting %	-0.550
Current energy efficiency rating	Current hot water costs over 3 years (Â£)	-0.528
Current energy efficiency rating	Energy Consumption Potential	-0.519
Current Emissions (T.CO2/yr)	Impact Of Solid Wall Insulation	-0.509
Current Environmental Impact Rating	Current hot water costs over 3 years (Â£)	-0.489
Current heating costs over 3 years (Â£)	Impact Of Solid Wall Insulation	-0.480
Current heating costs over 3 years (Â£)	Impact Of Loft Insulation	-0.371
Current Emissions (T.CO2/yr)	Impact Of Loft Insulation	-0.358
Current energy efficiency rating	Main Heating 1 Control	-0.334

Figure 29 Column that are negatively regulated

Current Environmental Impact Rating	CO2 Emissions Current Per Floor Area (kg.CO2/mÂ²/yr)	-0.924

Figure 30 correlation between environmental impact rating and CO2 Emission

Negative correlation between environmental impact rating and CO2 Emission

As CO2 emission increase, the environmental impact rating decrease. which means it has more impact on the environment.

The environmental impact rating is **a measure of a home's impact on the environment in terms of carbon dioxide (CO2) emissions**. The higher the rating the less impact it has on the environment.

Primary Energy Indicator (kWh/mÂ²/year)	Current Environmental Impact Rating	-0.883

Figure 31 correlation between environmental impact rating and CO2 Emission

‘Primary energy indicator’ has a negative correlation with ‘current environmental impact rating’
The higher the primary energy indicator, the more impact it has on the environment

Current energy efficiency rating	CO2 Emissions Current Per Floor Area (kg.CO2/mÂ²/yr)	-0.863

Figure 32 correlation between environmental impact rating and CO2 Emission

The higher energy efficiency rating is, the lower CO2 Emissions

Current energy efficiency rating	Current heating costs over 3 years (Â£)	-0.641
Current Environmental Impact Rating	Current heating costs over 3 years (Â£)	-0.588

Figure 33 correlation between environmental impact rating and CO2 Emission

Higher energy efficiency rating, lower heating costs
Higher environmental impact rating (less impact on environment), lower heating costs

- Positive correlation

Primary Energy Indicator (kWh/mÂ²/year)	CO2 Emissions Current Per Floor Area (kg.CO2/mÂ²/yr)	0.966
Current energy efficiency rating	Current Environmental Impact Rating	0.918
Fixed Lighting Outlets Count	Low Energy Lighting Outlets Count	0.901
Current Emissions (T.CO2/yr)	Current heating costs over 3 years (Â£)	0.859
Total floor area (mÂ²)	Current lighting costs over 3 years (Â£)	0.744
Primary Energy Indicator (kWh/mÂ²/year)	Energy Consumption Potential	0.727
CO2 Emissions Current Per Floor Area (kg.CO2...	Energy Consumption Potential	0.663
Total floor area (mÂ²)	Fixed Lighting Outlets Count	0.654
Current hot water costs over 3 years (Â£)	Main Heating 1 Control	0.647
CO2 Emissions Current Per Floor Area (kg.CO2...	Current Emissions (T.CO2/yr)	0.631
Total floor area (mÂ²)	Low Energy Lighting Outlets Count	0.572
Primary Energy Indicator (kWh/mÂ²/year)	Current Emissions (T.CO2/yr)	0.567
Total floor area (mÂ²)	Current heating costs over 3 years (Â£)	0.565
Current Emissions (T.CO2/yr)	Current lighting costs over 3 years (Â£)	0.556
Total floor area (mÂ²)	Current Emissions (T.CO2/yr)	0.554
Primary Energy Indicator (kWh/mÂ²/year)	Current hot water costs over 3 years (Â£)	0.532
Energy Consumption Potential	Main Heating 1 Control	0.528
Primary Energy Indicator (kWh/mÂ²/year)	Current heating costs over 3 years (Â£)	0.516

Figure 34 Column that are positive regulated

Primary Energy Indicator (kWh/mÂ²/year)	CO2 Emissions Current Per Floor Area (kg.CO2/mÂ²/yr)	0.966

Figure 35 correlation between environmental impact rating and CO2 Emission

The higher Primary energy indicator is , the higher CO2 emissions

Total floor area (mÂ²)	Current lighting costs over 3 years (Â£)	0.744

Figure 36 correlation between environmental impact rating and CO2 Emission

The greater total floor area, the higher lighting costs

Visualizations

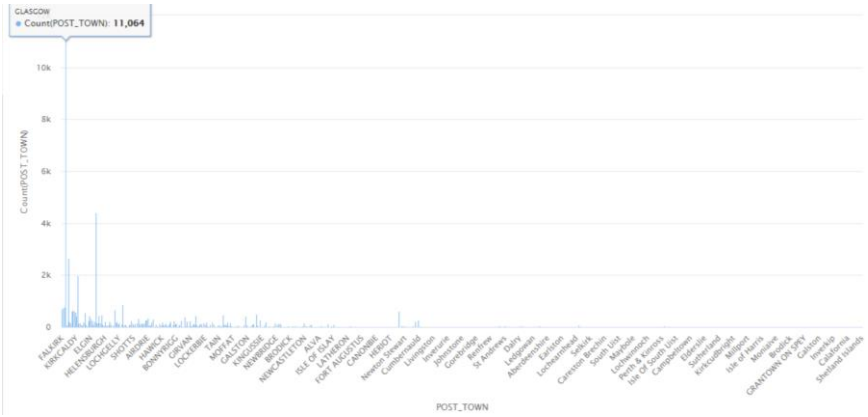


Figure 37 Mode in post town column

Most is Glasgow

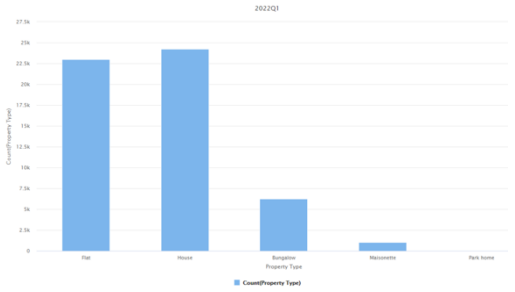


Figure 39

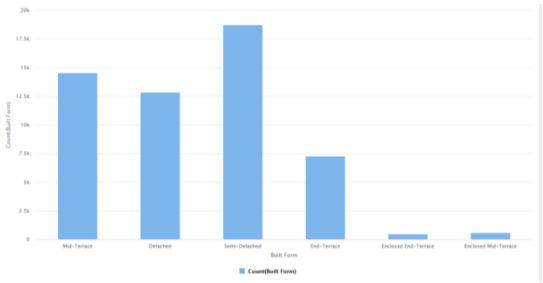


Figure 38

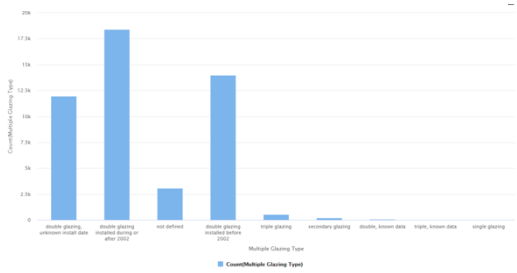


Figure 40

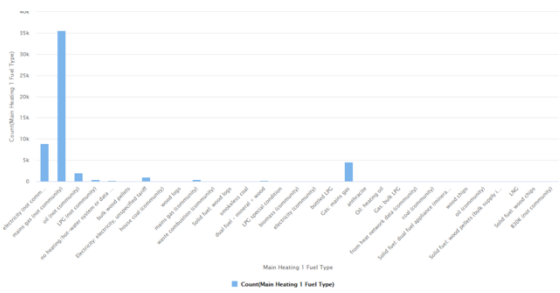


Figure 41

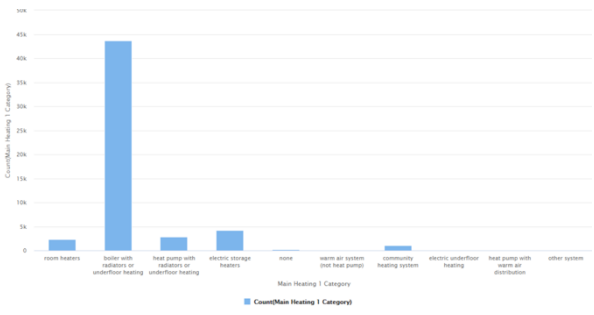


Figure 42

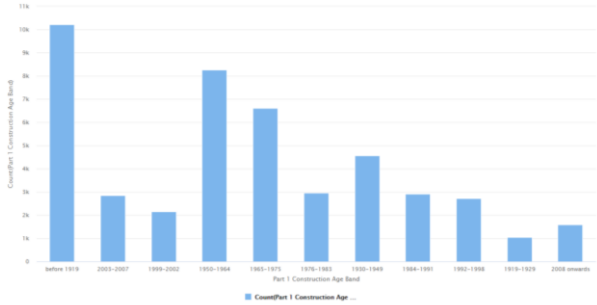


Figure 43

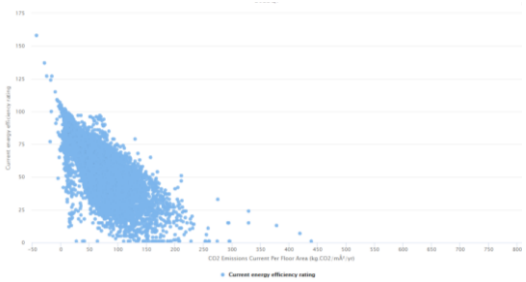


Figure 45

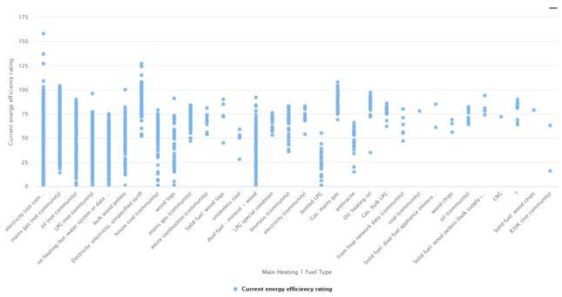


Figure 44

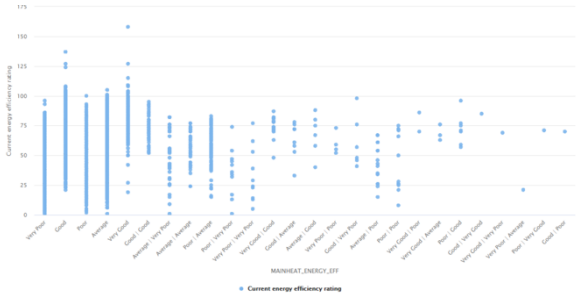


Figure 47

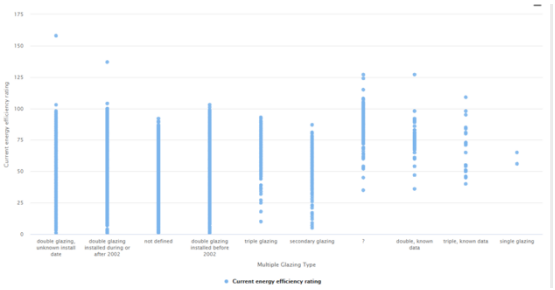


Figure 46

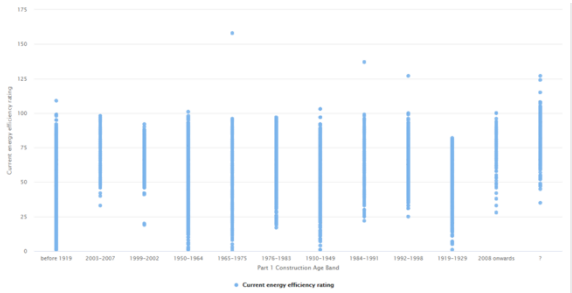


Figure 49

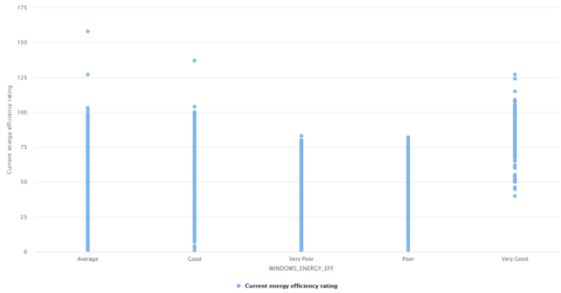


Figure 48

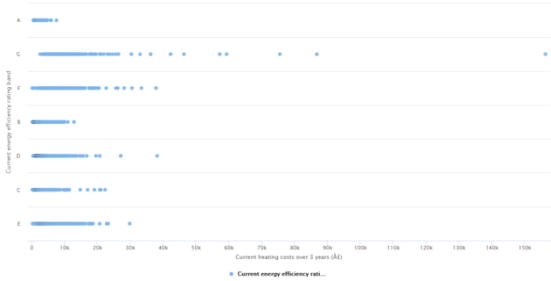


Figure 50

3.5 Conclusion

As an introduction to this chapter, we discussed the exploratory analysis and provided a brief review of our project. Then we discussed the approaches used to obtain data as well as a description of each dataset utilized in the project. Finally, exploratory data analytics techniques that gave a general information about datasets in use are presented in Section.

Chapter IV: Data Preparation & Preprocessing

4.1 Introduction

Real-world data typically contains noise, and missing values, and possibly in an unusable format that prevents it directly used in machine learning models [1]. Consequently, Data preprocessing is an essential stage in machine learning, since the quality of data and the useful information acquired from it immediately affect the learning capability of the model and enhance the accuracy and efficiency of the machine learning model. Section 4.2 of this chapter discusses the problems we found in the dataset. After that, in Section 4.3, we described how we prepared the data to use in training the models. We then conclude the chapter with Section 4.4.

4.2 Problems with the dataset

Since a huge amount of data is gathered simultaneously, the data usually involves noise, incompleteness, missing values, input issues, format incompatibilities, difficulties handling mixed data types, etc. In this section, we will discover the data problems that we faced and how we will fix them to move to the stage of data processing.

4.2.1 Missing Value

The dataset often has a lot of missing values. The cause of the presence of missing values in the dataset can be loss of information, disagreement in uploading the data, and many more. Missing values need to be imputed to proceed to the next step of the model development. Before imputing the missing values, it's important to understand the type of missing value present in the dataset.

	Percentage of Nulls in Dataset				
1	Main Gas	100%	15	Ward Code	11%
2	AIR_TIGHTNESS_ENV_EFF	100%	16	Ward Name	11%
3	AIR_TIGHTNESS_ENERGY_EFF	89%	17	Meter Type	11%
4	AIR_TIGHTNESS_DESCRIPTION	89%	18	Multiple Glazing Type	11%
5	LZC Energy Source	89%	19	Solar Water Heating	11%
6	Unheated Corridor Length	81%	20	Improvements	11%
7	Alternative Measures 2	72%	21	Mechanical Ventilation	11%
8	Heat Loss Corridor	71%	22	OSG_UPRN	8%
9	Flat Location	59%	23	Data Zone 2011	8%
10	Flat Level	59%	24	Photovoltaic Supply	6%
11	ADDRESS2	56%	25	Part 1 Floor 0 Room Height	3%
12	Part 1 Construction Age Band	37%	26	Main Heating 1 Fuel Type	2%
13	Data Zone	16%	27	POST_TOWN	1%
14	Local Authority	11%			

Table 7 Null values in the dataset.

4.2.2 Outliers

In the previous chapter, we performed an EDA on the dataset and identified outliers, which are data points that are fundamentally different from the rest of the data. In other words, abnormal distance from the rest of the data [1]. For example, a negative value in the column “Primary Energy Indicator (kWh/m2/year)” and “Potential future savings over 3 years (£)” and “CO2 Emissions Current Per Floor Area (kg.CO2/mÂ²/yr)” and “Current Emissions (T.CO2/yr)” that effects on the mean and standard deviation of the column values.

4.2.3 Inconsistency

Inconsistencies in naming conventions, and format. Occur in column "ADDRESS1" the error is present when collecting the type of property, it is confused between "Flat 1/Jan" and the correct value "Flat 1/1 "because the value format is like a date format.

Index	Nominal value	Absolute count	Fraction
24	FLAT A	91	0.002
25	01-Jan	89	0.002
26	FLAT B	80	0.001
27	02-Jan	79	0.001
28	FLAT 3/3	78	0.001
29	01-Feb	76	0.001
30	FLAT D	75	0.001
31	02-Feb	72	0.001

Figure 51

4.2.4 High Dimensionality

The dataset contains 104 features. however, some of them have neither correlation nor impact on energy efficiency. In 4.3.2 section, we will start the process of extracting features that will lead to dimensionality reduction.

4.2.5 Other problems

Some columns like "Improvements" and "WINDOWS_DESCRIPTION" have an unnecessary keyword at the beginning of the sentence and make the values lengthy when represented graphically so, we will remove them.

Index	Nominal value	Absolute co...	Fraction
1	Description: Solar water heating; Indicative ...	121	0.002
2	Description: Solar water heating; Indicative ...	91	0.002
3	Description: Solar water heating; Indicative ...	90	0.002
4	Description: Solar water heating; Indicative ...	66	0.001
5	Description: Solar water heating; Indicative ...	62	0.001
6	Description: Solar water heating; Indicative ...	57	0.001
7	Description: Solar water heating; Indicative ...	57	0.001
8	Description: Solar water heating; Indicative ...	54	0.001

Figure 52 Unwanted string in the dataset.

4.3 Data preparation and pre-processing

Data pre-processing includes data preparation, compounded by reduction, cleaning, and transformation of data; and data reduction tasks, such as feature extraction. The result expected after reliable chaining of data pre-processing tasks is a final dataset, which can be considered correct and useful for further data mining algorithms. And we will use Rapid Miner and Python to handle the issues. the following three sections are divided into data cleaning Section 4.3.1, data reduction Section 4.3.2, and data transformation Section 4.3.3, all these sections show how we pre-process and prepare our datasets to get them ready to build the models. And finally, we will present the code in the appendix.

4.3.1 Data Cleaning

Having clean data will ultimately increase overall productivity and allow for the highest quality information in your decision-making. Data cleaning include:

4.3.1.1 Handling missing value

The dataset often has a lot of missing values. The cause of the presence of missing values in the dataset can be loss of information, disagreement in uploading the data, and many more. Missing values need to be imputed to proceed to the next step of the model development. Before imputing the missing values, it's important to understand the type of missing value present in the dataset.

In our project, we dropped the columns that have above than 20% of missing values because if we fill it that mean values will be not real, and the columns that have less than 20% of missing value we impute the null with the most frequent values because the number of nominal attributes that we have in the dataset is greater than the numerical ones, so we decided to impute the missing values with the most frequent values in each attribute at once.

4.3.1.2 Handling noise data

Outliers are data points that are far from other data points. In other words, they are unusual values in a dataset. Outliers can distort statistical analysis and violate the assumptions.

Instead of dropping the row that has an outlier, we replaced it with a Null value, that way, we can impute it without losing data.

4.3.1.3 Handling inconsistency

We use “replace” function to change inconsistency value by the correct value.

4.3.2 Data Transformation

There are so many pre-processing steps before the modelling. And that further adds the confusion around the effect and use of transformation and scaling.

- **Scaling**

Scaling and Transformation are important steps of numeric feature, and they are being used to treat some features and rescale them for modelling.

4.3.3 Data Reduction

Data reduction achieves a reduction in volume, making it easy to represent and run data through advanced analytical algorithms. In the first select the columns which is include house feature and the number is 49 and column energy efficiency rating band which is a target.

- **Correlation Analysis**

We dropped one of each pair of correlated columns that had a correlation greater than 0.60, because if we did not remove one of the correlated variables it will affect the performance of the model correlation coefficients whose magnitude are between 0.7 and 0.9 indicate variables which can be considered highly correlated.

- **Feature selection**

At first we did label encoding for object to numeric conversion, then we utilized mutual_info_classif from sklearn.feature_selection that used to select best feature when dataset contain categorical feature more than numeric .In this phase, we select 20 best features according to the mutual information and , the next we handle the text in description column and repeated the mutual info to select best feature.

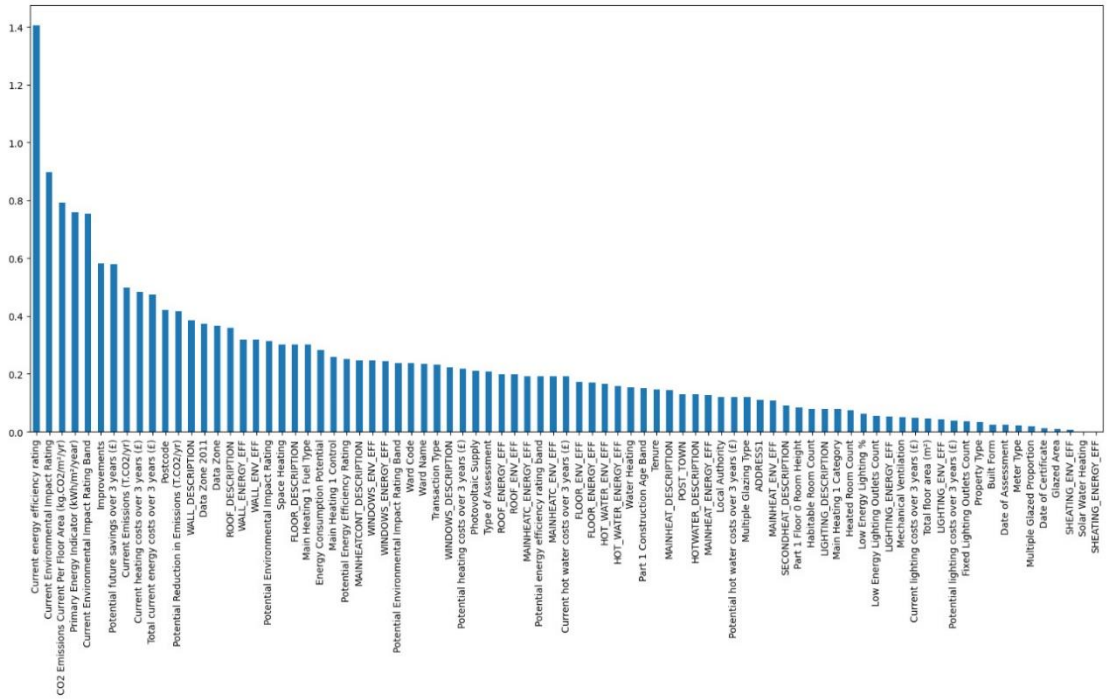


Figure 53 Mutual info.

4.4 Conclusion

In conclusion, this chapter contains a series of operations on the dataset, the most essential is exploring data issues and identifying solutions to them. To use the data in models in the following chapter, we use data pre-processing, which includes data cleaning, data reduction, etc.

Chapter V: Model Building

5.1 Introduction

Improving the energy efficiency of houses is a critical global challenge for reducing carbon dioxide emissions. Accurate forecasting and comprehension of energy efficiency are beneficial for maximizing energy usage and savings in the building industry. consequently, in this chapter, we utilize machine learning models to optimize the building industry on the data processed in the previous chapter. In this chapter, Section 5.2 describes the tools and libraries used to initialize machine learning models. Section 5.3 discusses Initial parameters and selection criteria and machine learning models, while Section 5.4 provides the conclusion of the chapter.

5.2 Experiments setup and tools

For the experimental setup of our models, we have prepared the data and selected the columns we need. We also set up all the necessary libraries and extensions for our pre-process and our prediction model to be set. With regards to the tools, we utilized Python since it has many capabilities and rich libraries that make it one of the top programming languages for machine learning. Furthermore, Python is free and open-source software. It consists of exhaustive libraries, and it has smooth implementation and integration. In Section 5.2.1 the common libraries are described, and in section 5.2.2 the tools are explained. As well as 5.2.3 describes how to prepare text data for description columns by splitting it up into several columns which are more simply encoded and manipulated by models and chatbot.

5.2.1 libraries

	Library	Extensions		Library	Extensions
1	pandas	None	12	keras.layers	Dense Dropout
2	Numpy	None	13	keras.optimizers	Adam SGD Adagrad
3	sklearn.feature_selection	mutual_info_classif SelectKBest	14	sklearn.metrics	accuracy_score precision_score recall_score f1_score confusion_matrix classification_report
4	sklearn.model_selection	train_test_split, RandomizedSearchCV GriedSearchCV	15	keras.utils	to_categorical
5	sklearn.preprocessing	StandardScaler LabelEncoder OneHotEncoder	16	sklearn.compose	ColumnTransformer
6	scipy	stats	17	keras.models	Sequential
7	Seaborn	None	18	tensorflow.keras.layers	Dense
8	matplotlib.pyplot	None	19	xgboost	XGBClassifier
9	regressormetricgraphplot	*	20	scikeras.wrappers	KerasClassifier
10	sklearn.svm	svc	21	sklearn.ensemble	RandomForestClassifier
11	keras.preprocessing.text	Tokenizer			

Table 8:Libraries and extensions.

Pandas:

Pandas is a Python data analysis library that is mostly used for data processing and analysis. It is important before the training dataset is available since it facilitates the merging and combining of datasets Reshaping and pivoting datasets. Also, Data alignment and handling missing data. Furthermore, there are several indexing options, such as fancy indexing and hierarchical axis indexing.

NumPy:

NumPy is a Python library that focuses on managing huge, multi-dimensional datasets. The complicated mathematical operations applied to the dataset NumPy allows for rapid computation and the execution of complex functions utilizing arrays. such as Support for logical and mathematical processes. Shape alteration. Data selection and sorting capabilities, and a variety of Fourier transformations.

Sickit-Learn or Sklearn:

The best Python machine-learning library is undoubtedly scikit-learn. Classification, regression, clustering, and dimensionality reduction are just a few useful statistical modelling techniques and machine learning techniques available in the sklearn library. In our model, we need to use three extensions from Sklearn.

sklearn.model_selection:

Is for comparing, validating, and choosing parameters and models.

sklearn.model_selection import RandomizedSearchCV:

Helps us combine an estimator with a random preamble to tune hyper-parameters.

sklearn.model_selection import GriedSearchCV:

Helps us combine an estimator with a grid preamble to tune hyper-parameters.

sklearn.preprocessing:

Provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators.

sklearn.compose:

Allows different columns or column subsets of the input to be transformed separately and the features generated by each transformer will be concatenated to form a single feature space. This is useful for heterogeneous or columnar data, to combine several feature extraction mechanisms or transformations into a single transformer [1].

sklearn.metrics:

To implements a number of loss, score, and utility methods to gauge classification performance.

scikeras.wrappers:

Extend with functionality specific to classifiers and regressors respectively.

scipy:

Is a scientific computation library that uses NumPy underneath. SciPy stands for Scientific Python. It provides more utility functions for optimization, stats, and signal processing.

Seaborn:

Is a library that uses Matplotlib underneath to plot graphs. It will be used to visualize random distributions.

matplotlib.pyplot:

Collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

sklearn.svm import SVC:

It's a supervised machine learning algorithm that's frequently applied to classification tasks. SVC separates the data into groups by mapping the data points to a high-dimensional space and then selecting the ideal hyperplane.

Keras:

Keras extends the usability of TensorFlow with these additional features for ML and DL programming. With a helpful community and a dedicated Slack channel, getting support is easy. Support for the convolutional and recurrent neural network also exists along with standard neural networks.

Keras.models:

Keras models are unique neural network-oriented models that arrange several layers and weed out crucial information. The Keras model comes in two different iterations: the Keras Sequential Model and the Keras Functional API, both of which may be customized and changed to fit different scenarios.

Keras.layers:

Keras layers serve as the fundamental building component upon which Keras models are built. For Keras models, they serve as the fundamental building blocks. Each layer in the Keras models is in charge of taking certain input values, executing some calculations and manipulations, and finally producing the appropriate output containing the necessary changed data. Additionally, it is a repeated structure where the computed data that is the output of a specific layer serves as the input values for subsequent models.

keras.optimizers:

Keras optimizer helps us achieve the ideal weights and get a loss function that is completely optimized. One of the most popular of all optimizers is gradient descent [2].

keras.utils:

Provides the numpy utility library which provides functions for performing actions onto the arrays of numpy. By using the method of to_categorical() vector numpy array with an integer that represents the different categories was converted into a numpy array which binary contains the matrix values for the number of data categories [3].

Tensorflow:

When we start to split our dataset after we done the filtering we choose to use Scikit-learn. When we start to build our model, we choose to use Tensorflow and keras that is part of Tensorflow.

5.2.2 Tools

We used Jupyter notebook because it has an interactive way of documenting and executing live code from a web-based interface, easy to use and easy to share it with team members.

5.2.3 Text Data Preparation

A common operation to handle text features is to split the text and extract meaningful information that is not explicit in the data. In this section of the report, we clarify the work of splitting columns with text and numerous values into separate columns and transforming the descriptive columns into several categorical, ordinal, and binary columns.

The objective of splitting these columns is to turn text into features, to make them more understandable, and to prepare the data sufficiently to permit categorical data encoding so that the model may use all the features obtained using mutual information in the previous chapter. Furthermore, we prepare them to adapt to the user interface in the chatbot. We created multiple functions, one for basic text cleaning like removing stop words and punctuations and unnecessary words like "assumed" and "as built" since they do not add any value to the information we are trying to extract, and other functions for each of the 9 descriptive columns to split the text into categories. Since the descriptive columns in the dataset had multiple values separated by a comma in each row, we were able to split each part into new columns (see table 9) from each part, and finally, created new columns from each.

	Original Feature	Derived Feature		Original Feature	Derived Feature
1	FLOOR_ DESCRIPTION	floor_insulation insulated_floor floor_type	6	ROOF DESCRIPTION	roof_insulation insulated_roof roof_type
2	LIGHTING_ DESCRIPTION	low_lighting	7	WINDOWS_ DESCRIPTION	windows_glazing
3	WALL_ DESCRIPTION	Wall_insulation insulated_wall wall_type	8	SECONDHEAT_ DESCRIPTION	SH_oil, SH_bioethanol SH_anthracite …etc
4	HOTWATER_ DESCRIPTION	hws_from_main system hws_from_secondary syste m hws_community scheme,.. etc	9	MAINHEAT_ DESCRIPTION	MHS_boiler, MHS_radiators, MHS_electric heaters …etc
5	MAINHEATCONT_ DESCRIPTION	MHCS_programmer MHCS_room thermostat(s) MHCS_none, …etc			

Table 9 : Original Descriptive Features and Derived Features

After splitting and extracting are done, we reduced the number of categories derived from some of the original features like WINDOWS_DESCRIPTION (figure 54) by grouping the similar into the same category (see figure 55).

```
description: fully double glazed            44008
description: high performance glazing        6287
description: single glazed                   1679
description: partial double glazing           723
description: mostly double glazing            656
description: fully triple glazed              560
description: some double glazing              316
description: full secondary glazing           112
description: partial secondary glazing         59
description: some secondary glazing            26
description: mostly secondary glazing          24
description: multiple glazing throughout       17
description: mostly multiple glazing           10
description: mostly triple glazing              7
description: partial triple glazing             5
description: partial multiple glazing           5
description: some multiple glazing              1
Name: WINDOWS_DESCRIPTION, dtype: int64
```

Figure 55 WINDOWS_DESCRIPTION and count each value

```
double glazed               45703
high performance glazing     6287
single glazed                1679
triple glazed                 572
secondary glazing             221
multiple glazing               33
Name: windows_glazing, dtype: int64
```

Figure 54 after pre-process

```
In [8]: epc_all['MAINHEAT_DESCRIPTION'].value_counts()

Out[8]: Boiler and radiators, mains gas                                                                39829
        Electric storage heaters                                                                        4013
        Room heaters, electric                                                                          2179
        Boiler and radiators, oil                                                                       1976
        Air source heat pump, radiators, electric                                                       1633
        Community scheme                                                                                1092
        Boiler and radiators, electric                                                                   653
        Air source heat pump, underfloor, electric                                                       568
        Boiler and radiators, LPG                                                                        496
        No system present: electric heaters assumed                                                      222
        Air source heat pump, Systems with radiators, electric                                           209
        Ground source heat pump, underfloor, electric                                                    152
        Electric storage heaters | Electric storage heaters                                              138
        Boiler and underfloor heating, mains gas                                                         131
        Boiler and radiators, dual fuel (mineral and wood)                                               114
        Boiler and underfloor heating, oil                                                                99
        Air source heat pump, warm air, electric                                                          88
        Ground source heat pump, radiators, electric                                                      85
        Boiler and radiators, wood pellets                                                                85
        Air source heat pump, radiators, electric | Boiler and radiators, mains gas                       64
        Boiler and radiators, coal                                                                        53
        Electric underfloor heating                                                                       47
        Warm air, mains gas                                                                               44
        Room heaters, dual fuel (mineral and wood)                                                        40
        Electric storage heaters | Room heaters, electric                                                 39
        Boiler and radiators, wood logs                                                                   33
        Room heaters, mains gas                                                                           30
        Warm air, Electricaire                                                                            22
        Air source heat pump , electric                                                                   20
        Air source heat pump, Underfloor heating and radiators, pipes in screed above insulation, electric  19
        Boiler and radiators, mains gas | Boiler and underfloor heating, mains gas                        19
        Room heaters, wood logs                                                                           17
        Air source heat pump, Underfloor heating, pipes in screed above insulation, electric              16
        Boiler and underfloor heating, LPG                                                                15
        Boiler and radiators, dual fuel (mineral and wood) | Boiler and radiators, oil                    14
        Portable electric heaters assumed for most rooms                                                  14
        Room heaters, coal                                                                                13
        Room heaters, anthracite                                                                          11
        Boiler and radiators, anthracite                                                                  10
        Boiler and underfloor heating, electric                                                            9
        Air source heat pump, radiators, electric | Air source heat pump, underfloor, electric            9
        Air source heat pump, Underfloor heating, pipes in concrete slab, electric                         8
        Air source heat pump, radiators, electric | Boiler and radiators, oil                             8
        Water source heat pump, radiators, electric                                                        7
        Ground source heat pump, Underfloor heating, pipes in screed above insulation, electric            7
        Boiler, mains gas                                                                                  7
```

Figure 56 MAINHEAT_DESCRIPTION and count each value

Furthermore, the above figure 56 show that there is a huge number of values in the feature MAINHEAT_DESCRIPTION, to handle this, we created two temporary columns **'main_heat_system'** and **'main_heat_system_2',** each holding a part of the text of the original feature, using str.extract() with some Regex. Then, we extracted MHS_boiler, MHS_radiators, MHS_heat pump, MHS_room heaters, MHS_community scheme, MHS_underfloor heating, MHS_warm air, and MHS_electric heaters from the first column. As well as MHS_mains gas, MHS_None, MHS_electric, MHS_oil, MHS_radiators, MHS_underfloor heating, MHS_lpg, MHS_dual fuel mineral and wood, MHS_wood pellets_logs, MHS_coal from the second one, and grouped the less frequent ones into the category 'MHS_other', so we end up with fewer but more important features.

```
boiler and radiators                          43349
electric storage heaters                       4192
air source heat pump                           2651
room heaters                                   2305
community scheme                               1092
boiler and underfloor heating                   259
ground source heat pump                         253
no system present: electric heaters             222
warm air                                         67
electric underfloor heating                      47
air source heat pump                             20
portable electric heaters  for most rooms        14
water source heat pump                            8
boiler                                            7
boiler &amp;amp; underfloor                       6
electric ceiling heating                          2
boiler &amp;amp; fan coil units                   1
Name: main_heat_system, dtype: int64
```

Figure 58 main_heat_system and count each value

```
mains gas                          45646
electric                            2869
oil                                 2105
radiators                           1809
underfloor                           729
lpg                                  518
systems with radiators               209
dual fuel mineral and wood           180
warm air                             102
wood pellets                          87
coal                                  68
wood logs                             53
underfloor heating                    33
underfloor heating and radiators      26
electricaire                          22
anthracite                            21
smokeless fuel                         6
wood chips                             4
fan coil units                         4
b30k                                   2
bottled lpg                            1
lng                                    1
```

Figure 57 main_heat_system_2 and count each value

	Derived Feature	Description
1	floor_insulation	Indicating the type of wall insulation (no insulation, insulated, limited insulation,..etc)
2	insulated_floor	Whether or not the floor is insulated
3	floor_type	Indicating the floor construction/ material
4	low_lighting	Percentage of low energy lighting in fixed outlets
5	Wall_insulation	Indicating the type of wall insulation (no insulation, insulated, external insulation, internal insulation, ..etc)
6	insulated_wall	Whether or not the wall is insulated
7	wall_type	Indicating the wall construction/ material
8	roof_insulation	Indicating the type of wall insulation (no insulation, insulated, limited insulation,... etc)
9	insulated_roof	Whether or not the roof is insulated
10	roof_type	Indicating the roof construction/ material
11	windows_glazing	Indicating the type of windows glazing (high performance, single, double, tripe,…etc)

Table 10 description of derived features

```
def insulation_extract(df):
    '''takes the second part after ',' that indicates insulation in description columns '''

    df['wall_insulation'] = df['WALL_DESCRIPTION'].str.extract('(?<=,\s)([^,]+)')
    df['roof_insulation'] = df['ROOF_DESCRIPTION'].str.extract('(?<=,\s)([^,]+)')
    df['floor_insulation'] = df['FLOOR_DESCRIPTION'].str.extract('(?<=,\s)([^,]+)')
    df[['wall_insulation','roof_insulation','floor_insulation']] = df[['wall_insulation','roof_insulation','floor_insulation']].a

    return df
epc_49 = insulation_extract(epc_49)
```

```
def wall_insulation(df):
    '''returns df with a new column 'insulated_wall' classifying wall insulation 1 if insulated 0 if not'''
     ##=============insulated=============##
    df.loc[df['wall_insulation'].str.contains('with external insulation') , 'insulated_wall'] = 1
    df.loc[df['wall_insulation'].str.contains('with internal insulation') , 'insulated_wall'] = 1
    df.loc[df['wall_insulation'].str.contains('with additional insulation') , 'insulated_wall'] = 1
    df.loc[df['wall_insulation'].str.contains('partial insulation') , 'insulated_wall'] = 1
    df.loc[df['wall_insulation'].str.contains('with insulation') , 'insulated_wall'] = 1
    df.loc[df['wall_insulation'].str.contains('filled cavity') , 'insulated_wall'] = 1
    df.loc[df['wall_insulation'].str.contains('insulated') , 'insulated_wall'] = 1
    df.loc[df['wall_insulation'].str.contains('average thermal transmittance') , 'insulated_wall'] = 1

     ##=============no insulation=============##
    df.loc[df['wall_insulation'].str.contains('no insulation') , 'insulated_wall'] = 0

    return df
epc_49 = wall_insulation(epc_49)
```

Figure 59 function to pre-process the descriptive features

Insulation_extract() is a function to extract the part that indicates the insulation in each of WALL_DESCRIPTION, ROOF_DESCRIPTION and FLOOR_DESCRIPTION into new columns.

```
def windows_glazing(df):
    '''returns a df with new column 'windows_glazing' indicating the type of windows glazing '''
    df['windows_glazing'] = df['WINDOWS_DESCRIPTION'].str.replace("description:","")
    df['windows_glazing'] = df['windows_glazing'].str.replace("fully","")
    df['windows_glazing'] = df['windows_glazing'].str.replace("partial","")
    df['windows_glazing'] = df['windows_glazing'].str.replace("mostly","")
    df['windows_glazing'] = df['windows_glazing'].str.replace("full","")
    df['windows_glazing'] = df['windows_glazing'].str.replace("some","")
    df['windows_glazing'] = df['windows_glazing'].str.replace("^\s+","")
    df['windows_glazing'] = df['windows_glazing'].str.replace("triple glazing","triple glazed")
    df['windows_glazing'] = df['windows_glazing'].str.replace("multiple glazing throughout","multiple glazing")
    df['windows_glazing'] = df['windows_glazing'].str.replace("double glazing","double glazed")

    return df
epc_49 = windows_glazing(epc_49)
```

Figure 60 function to pre-process the descriptive features

Reducing the number of categories in windows_glazing column by removing unnecessary words at the beginning.

```
def secondheat(df):
    df['second_heat'] = df['SECONDHEAT_DESCRIPTION']
    df['second_heat'] = df['SECONDHEAT_DESCRIPTION'].fillna(df['second_heat'].value_counts().index[0])

    df['SH_room heaters'] = df['second_heat'].str.contains('room heaters')
    df['SH_none'] = df['second_heat'].str.contains('none')
    df['SH_electric'] = df['second_heat'].str.contains('electric')
    df['SH_mains gas'] = df['second_heat'].str.contains('mains gas')
    df['SH_portable electric heaters'] = df['second_heat'].str.contains('portable electric heaters')
    df['SH_dual fuel mineral and wood'] = df['second_heat'].str.contains('dual fuel mineral and wood')
    df['SH_wood logs'] = df['second_heat'].str.contains('wood logs')
    df['SH_coal'] = df['second_heat'].str.contains('coal')
    df['SH_lpg'] = df['second_heat'].str.contains('lpg')
    df['SH_wood pellets'] = df['second_heat'].str.contains('wood pellets')
    df['SH_smokeless fuel'] = df['second_heat'].str.contains('smokeless fuel')
    df['SH_anthracite'] = df['second_heat'].str.contains('anthracite')
    df['SH_bioethanol'] = df['second_heat'].str.contains('bioethanol')
    df['SH_oil'] = df['second_heat'].str.contains('oil')

    return df
epc_49 = secondheat(epc_49)
```

Figure 61 function to pre-process the descriptive features

secondheat() function to create new columns of the unique categories exists in SECONDHEAT_DESCRIPTION.

```
#dropping description attributes
epc_49 = epc_49.drop(['WALL_DESCRIPTION', 'ROOF_DESCRIPTION', 'FLOOR_DESCRIPTION',
                      'WINDOWS_DESCRIPTION', 'MAINHEAT_DESCRIPTION','MAINHEATCONT_DESCRIPTION',
                      'SECONDHEAT_DESCRIPTION', 'HOTWATER_DESCRIPTION', 'LIGHTING_DESCRIPTION'] , axis=1)
```

Figure 62 dropping the descriptive features

finally, description columns are dropped.

5.2.4 Chatbot setup

To set up the chatbot, at the beginning, the best 20 features were chosen, and we will choose the best accuracy of Models. We chose the Python language to make the chatbot because of its ease and the availability of supporting resources.

5.3 Initial parameters and selection criteria

In the following sections we will identify the parameters for each model.

5.3.1 Machine Learning Models

Machine learning is the process of predicting and classifying data using various machine learning models based on the dataset. These models can contain numerous parameters and identify the optimum combination of parameters. Classifiers used to predict the "Energy Efficiency Rating Band" are highlighted in the following sections, along with information on how to hyperparameter adjust the model to enhance performance.

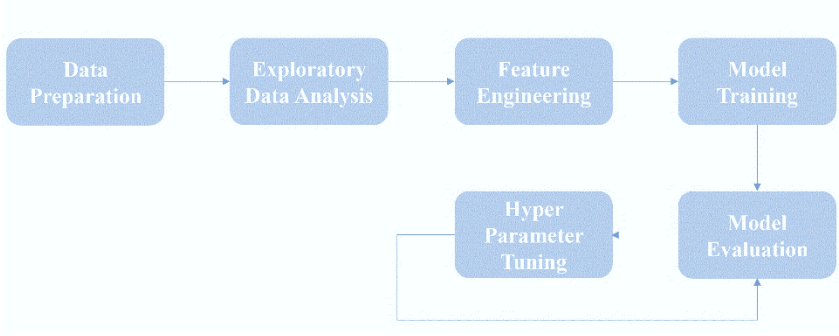


Figure 63:hyperparameter machine learning model

5.3.1.1 Support Vector Machine

In training models, we start with an SVM classifier to predict the 'Energy Efficacy Rating Band', fits the dataset in the model then train it to learn then optimize this model via Random Grid Search to choose values randomly rather than using a predefined set of values like the grid search method. As SVM is suitable for binary classification, the actual multi-class classification is broken down to multiple binary classification.

To improve the model accuracy, there are several parameters need to be tuned via RandomizedCVsearch. Three key factors are as follow [4]:

- Kernels: The kernel's main role is to take a low-dimensional input space and transform it into a higher-dimensional space.
- C (Regularisation): C is the penalty parameter and controlling error. When C is high, all data points are accurately classified.
- Gamma: This parameter specifies how much the computation of a reasonable line of separation is influenced.

We construct a dictionary in the model named param grid, populate it with C and gamma parameters, and keep the kernel to the default value “RBF". Fit a RandomizedCVsearch object to the training data after creating it. finally found the best estimator using random search take this model to create some predictions using the test set.

```
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf']}
```

Figure 64: Dictionary of SVM parameters

5.3.1.2 Tree-based Models.

Tree-based algorithms are considered one of best supervised machine learning methods and they are commonly used to deal with both classification and regression problems, one of the reasons tree-based models are one of the most used algorithms is that they are adaptable at solving any problem, using any kind of data whether it is categorical or continuous.

Tree-based models, basically, use a decision tree to split the input variables repeatedly into subsets based on the most significant splitter which will be the root node of the tree, and each branch is tested for prediction accuracy and evaluated for efficiency, Thus, they can empower predictive models with high accuracy while still being easy to interpret.

5.3.1.2.1 Random Forest Classifier

Random forest is made up of a number of an uncorrelated forest of decision trees, it generates a random subset of features using the bagging method (feature randomness), which leads to having a more diverse collection of sub-samples of the dataset and less correlation among trees. It also uses averaging to enhance the accuracy of prediction while reducing the risk of over-fitting.

In this part, we will discuss the implementation of this model to predict our target variable "Energy Efficiency Rating Band", In order to implement the random forest classifier, we first importRandomForestClassifier from sklearn.ensemble, we create an instance of the model forest= RandomForestClassifier(), then we can fit the model to our data with its default parameters model.fit(X_train, y_train) and make predictions on the test set y_pred= model.predict(X_test).

After the RandomForestClassifier model fits the data and finishes training using its default parameters, we used the RandomizedSearchCV to perform hyperparameter tuning with 3-fold cross-validation, so we get the best parameters that give the highest accuracy. First, we define the set of parameters to be tuned and their list of values, then we create random_grid python dictionary holding each parameter and setting it to the list of values we defined previously, lastly, we use RandomizedSearchCV() and set its param_distirbutions to random_grid.

```
n_estimators = [600, 800, 1000, 1200, 1400, 1600, 1800, 2000]
max_features = ['auto', 'sqrt']
max_depth = [int(x) for x in np.linspace(10, 120, num = 12)]
min_samples_split = [2, 6, 10]
min_samples_leaf = [1, 3, 4]
bootstrap = [True, False]
```

Figure 65: Dictionary of RF parameters

```
random_grid = {'n_estimators': n_estimators,
'max_features': max_features,
'max_depth': max_depth,
'min_samples_split': min_samples_split,
'min_samples_leaf': min_samples_leaf,
'bootstrap': bootstrap}

rf = RandomForestClassifier()
from sklearn.model_selection import RandomizedSearchCV
rf_random = RandomizedSearchCV(estimator = rf,
                               param_distributions = random_grid,
                               n_iter = 30, cv = 3, verbose=2,
                               random_state=35, n_jobs = -1)
```

Figure 66: RF parameter tuning

```
# print the best parameters
print ('Best Parameters: ', rf_random.best_params_, ' \n')
```

```
Best Parameters:  {'n_estimators': 1600, 'min_samples_split': 6, 'min_samples_leaf': 1, 'max_features': 'auto', 'max_depth': 8
0, 'bootstrap': False}
```

Figure 67: RF model best parameters

- n_estimators: number of trees in the random forest (default=100)
- max_features: number of features in consideration at every split (default=”sqrt”)
- max_depth: maximum number of levels allowed in each decision tree (default=None, meaning nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples)
- min_samples_split: minimum sample number to split a node (default=2)
- min_samples_leaf: minimum sample number that can be stored in a leaf node
- bootstrap: Whether samples are used when building trees. (default = True, If False, the whole dataset is used to build each tree).

5.3.1.2.2 XGB Classifier

XGBoost (Extreme Gradient Boosting) is an ensemble machine learning algorithm that uses gradient boosting framework which is based on decision tree, it is considered one of the most powerful algorithms due to its ability to handle large datasets with its built-in support for parallel processing, to train models on large datasets in a reasonable amount of time, and its ability to perform very well in both classification and regression problems. It uses lasso and ridge to penalize the high complexity of the model, it also utilizes the Depth First approach to improve computational performance significantly by optimizing the max_depth parameter it can be specified as the stopping criteria for splitting the branch to start pruning the trees backward. XGB prevents overfitting by using its built-in cross-validation method.

The implementation of the XGBoost model starts with importing the XGBClassifier from the xgboost library, then we create the model instance xgboost = XGBClassifier(), after that, we fit the model with its default parameters to our data model.fit(X_train, y_train), and finally make predictions on the test set y_pred = model.predict(X_test).

We also create another optimized model by using RandomizedSearchCV(), in order to do that, we first define the search space by creating a python dictionary of all the features to be optimized and their values (figure 68), then we use RandomizedSearchCV() and set its param_distributions=param_grid1.

```
"learning_rate": [.2,.3, 0.01, 0.1, 1] ,
"max_depth": [int(x) for x in np.linspace(10, 120, num = 12)],
"gamma": [0.5, 1, 1.5, 2, 5],
"colsample_bytree": [0.6, 0.8, 1.0],
"n_estimators": [600, 800, 1000, 1200, 1400, 1600, 1800, 2000]
```

Figure 68: Dictionary of XGB parameters

```
# Random search of parameters, using 3 fold cross validation,
# search across 30 different combinations, and use all available cores
random_search = RandomizedSearchCV(estimator=xgboost, param_distributions=param_grid1,
                                   n_iter = 30, cv = 3, verbose=2, random_state=42, n_jobs = -1)
random_search.fit(X_train, y_train)
```

Figure 69: XGB parameter tuning

```
print(random_search.best_params_)
```

```
{'n_estimators': 1400, 'max_depth': 30, 'learning_rate': 0.01, 'gamma': 1, 'colsample_bytree': 0.8}
```

Figure 70: XGB model best parameters

- Learning rate: (default=0.3) shrinks the feature weights to make the boosting process more conservative and prevent overfitting.
- max_depth: (default=6) maximum depth of the tree, increasing it increases the model complexity.
- gamma: (default=0) specifies the minimum loss reduction required to make a split.
- colsample_bytree: Percentage of columns to be randomly sampled for each tree.
- n_estimators: (default=100) Number of trees.

5.3.2 Deep Learning Neural Network

Deep learning is a class of machine learning that is basically a three- or more-layered neural network. These neural networks seek to simulate the behavior of the human brain, although with limited success, allowing it to "learn" from enormous volumes of data. Whereas a neural network with a single layer may still produce approximate predictions, more hidden layers can assist to optimize and tune for accuracy [5].

The conventional neural network has been chosen for this project because it is the best for predicting the target value. It is a mathematical construct with four components: Convolution layer, Pooling layer, Fully Connected layer, and Activation function.

In this part, we discuss the deep learning model used to predict and how to tune the parameters. So, we start explaining the structure of the neural network. Typically, three-layer types are specified:

- The input layer is where you will send the features of your dataset. In this layer, no calculation takes place. Its purpose is to transport features to the hidden layers.
- The hidden layers are typically the layers between the input and output layers—and there may be more than one. These layers carry out calculations and send the results to the output layer.
- The output layer of the neural network will provide the results after training the model. It is responsible for producing the output variables.

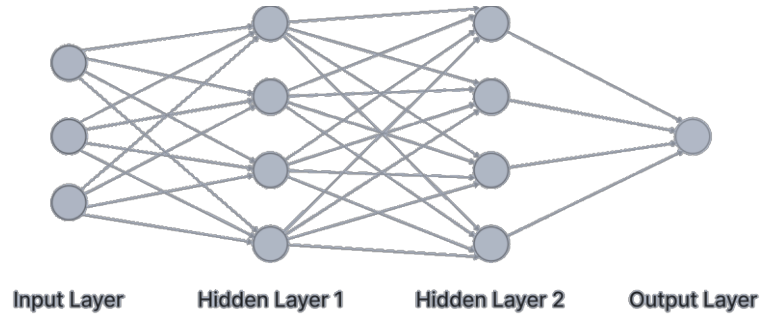


Figure 71: Deep learning

We shall discuss the model when we understand the structure of the neural network. After encoding the categorical data by using One Hot Encoder. then use the Keras library to construct NN and the sequential model to generate a linear array of layers. Since it is a prediction problem, a classified variable will be generated.

```
model = Sequential()
model.add(Dense(64, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dropout(0.5))  # Adding dropout to prevent overfitting
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))  # Adding dropout to prevent overfitting
model.add(Dense(7, activation='sigmoid'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Figure 72 neural network model

After creating a sequential of NN use add function in the input layer to specify some parameters:

- The first parameter is the number of nodes '64' .and the connection between different nodes is what forms the neural network.
- The second parameter is the activation function. it will learn the model, Use the 'Relu' activation function because it generalizes well on our data.
- The third parameter is the input shape, reshape the input in the training and test sets to have a fixed size. This is necessary because the model we are using expects its input data to have a fixed shape.

Between the layers add Dropout layer '0.5' is a mask that eliminates some neurons' contributions to the following layer while leaving all others unchanged, also preventing overfitting. After that, the hidden layer has 64 neurons and uses the 'Relu' activation function. The last layer is the output layer takes the following parameters:

- The number of output nodes. 7 refer to the number of classes that will be predicted as a result of this layer.
- As starting model uses the sigmoid activation function so that you can get the predicted classes.

Gradient descent is implemented using the compile function, which accepts the following parameters:

- use the categorical_crossentropy loss function, since this is a categorical prediction issue.
- The last parameter is the metric used to assess the model. In this case, evaluate it based on its accuracy when making predictions.

After creating the first model, we will adjust the Hyperparameters to improve performance. We made two experiments to improve the model, in the first attempt we set the parameters manually and in the second attempt we used KerasClassifier and grid search to optimize it. In both attempts, we tried to improve the model based on Hyperparameters related to Network structure, especially the number of neurons and activation function, and Hyperparameters related to the Training Algorithm selecting the number of epochs and Batch size and optimizer.

Model	No. of neurons	Activation Output	optimizer	epochs	batch_size
Model A	128	Sofmax	Adam	100	32
Model B	64	sigmoid	Adagrad	100	80
Model A uses manual optimization. Model B uses KerasClassifier and grid search to optimize it.					

Table 11 Optimization NN

Model A increase the number of neurons in the input and hidden layer to 128hich asset the NN to learn all possible situation and increases the recognition rate. Also, 'Softmax' which is used as the activation function in the output layer of neural network models that predict a multinomial probability. Moreover, when increasing the number of epochs determines how frequently the network's weights will be changed. As the number of epochs grows, so do the weights in the neural network, and the boundary shifts from underfitting to the optimum to overfitting. The optimizer kept the 'Adam' because is the most used and the results of the Adam optimizer are generally better than every other optimization algorithm we trained, for batch size remains the default.

Model B uses Keras models that can be used in scikit-learn by wrapping. To use these wrappers, define a function that creates and returns Keras sequential model, then passes this function to the model argument when constructing the KerasClassifier class. In this model tune three parameters optimizer, epoch, and batch size. We construct a list for epoch and batch size, and optimizer to use it in the param

grid. Fit a GridSearchCV object to the training data after creating it. finally found the best estimator using grid search Take this model to create some predictions using the test set.

5.3.3 Chatbot

As mentioned earlier, we use machine learning models to improve them using the data processed in the previous chapter, then we compare the results and choose the model that produces the best results to serve as the basis for a chatbot that will help the user assess the energy efficiency of his house. Then, to establish a chatbot first evaluate the selection feature in the dataset where the user knows this information, and whether can provide it in the chatbot or not. The table below shows this evaluation.

	Feature	**Type**	**Description**	**Can provide**
1	Total floor area (m²)	numeric	The total floor area of the buildings m2.	✓
2	Total current energy costs over 3 years (£)	numeric	The total energy cost in general over the stated period.	✓
3	Current hot water costs over 3 years (£)	numeric	The total energy expense for heating water during the specified time.	✓
4	Part 1 Construction Age Band	numeric	Age band when building part constructed. For Scotland.	✓
5	Part 1 Floor 0 Room Height	numeric	Average height of the lowest storey of the dwelling (Units: m)	✓
6	Low Energy Lighting %'	numeric	The percentage of low energy lighting present in the property as a percentage of the total fixed lights in the property.	✓
7	Mechanical Ventilation	nominal	Identifies the type of mechanical ventilation the property	✓
8	Tenure	nominal	Describes the tenure type of the property.	✓
9	Transaction Type	nominal	Type of transaction or purpose that triggered EPC assessment	✓
10	Property type	nominal	Describes the type of property.	✓
11	insulated_wall	binary	If the wall is insulated 1, otherwise 0.	✓
12	wall_type	nominal	The type of material of wall.	✓
13	roof_insulation	nominal	The type of roof insulation.	✓

14	roof_type	nominal	the type of heat loss the roof of dwelling has.	✓
15	floor_type	nominal	The type of floor construction.	✓
16	windows_glazing	nominal	the glazing type of window.	✓
17	MMH_mains gas	binary	if the main heat system uses mains gas.	✓
18	MHCS_programmer	binary	if the main heat control system includes a programmer.	✓
19	low_lighting	nominal	percentage of low lighting.	✓
20	SH_room heaters	binary	if the second heat system is room heaters.	✓

Table 12: chatbot features evaluation

5.4 Conclusion

To wrap up this chapter, we started by talking about the required setups environment for the experiments and the necessary libraries, tools, and configurations. Finally, a detailed explanation of how the model was implemented and the selection criteria for the model parameters.

Chapter VI: Results and Discussions

6.1 Introduction

Our idea's major objective is to evaluate the energy efficiency of the houses by using machine learning models and to provide to the user a simple tool consisting in a chatbot that provides the prediction of the house energy profile. For this, we used some of machine learning models and CNN with various architectures.

Recent research has shown the ability of convolutional neural networks (CNN) to deal with complex machine learning problems. Unprecedented results were achieved in tasks such as classification and segmentation, often outperforming human accuracy. Different layers of the network are capable of different levels of abstraction and capture the different amounts of structure from the patterns present in the data. Consequently, a significant performance increase can be achieved as soon as faster hardware and more training data become available [1].

In this chapter, Section 6.2 illustrates the performance evaluation metrics. Section 6.3 display the comparison of the experimental results of our previous versions with our final versions of the models. Then, interpreting the results, discussing whether the objectives were achieved, and describing the limitations are placed in Section 6.4. At the end of the chapter, the conclusion is given in Section 6.5.

6.2 Performance evaluation metrics

We can assess the effectiveness of deep learning and machine learning models for classification using a variety of performance evaluation metrics. Here are a few of the most crucial performance evaluation metrics you should use to assess the effectiveness of a machine learning and deep learning model.

6.2.1 Performance evaluation metrics

Classification models have discrete outputs, so we need a metric that compares discrete classes in some form. Classification Metrics evaluate a model's performance and tell you how good or bad the classification is, but each of them evaluates it in a different way.

6.2.1.1 Confusion matrix

In the field of machine learning and specifically the problem of statistical classification, a confusion matrix also known as an error matrix is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix) [2]. confusion matrix 2*2 classes can be visualized as below:

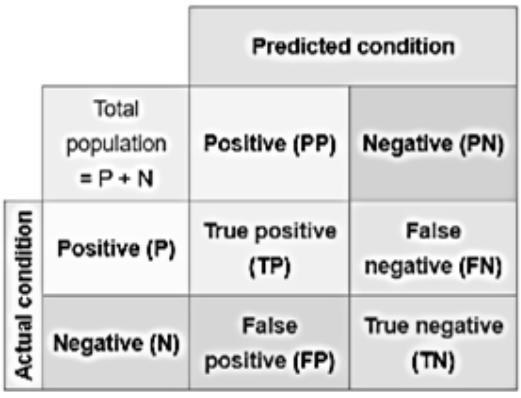


Figure 73: confusion matrix

- **condition positive (P)**

the number of real positive cases in the data

- **condition negative (N)**

the number of real negative cases in the data

true positive (TP)

A test result that correctly indicates the presence of a condition or characteristic.

true negative (TN)

A test result that correctly indicates the absence of a condition or characteristic.

false positive (FP)

A test result that wrongly indicates that a particular condition or attribute is present.

false negative (FN)

A test result that wrongly indicates that a particular condition or attribute is absent.

6.2.1.2 Accuracy

The most straightforward way to measure a classifier's performance is using the Accuracy metric. We compare the actual and predicted class of each data point, and each match counts for one correct prediction. Accuracy is often used as the measure of classification performance because it is simple to compute and easy to interpret. However, it can turn out to be misleading in some cases [3].

$$\text{accuracy} = \frac{\text{\# of correct predictions}}{\text{\# of total predictions}}$$

Figure 74:Accuracy formula

6.2.1.3 Precision

Precision is the ability of a classifier not to label an instance positive that is negative. For each class, it is defined as the ratio of true positives to the sum of true and false positives [4].

$$Precision = \frac{truepositives}{truepositives + falsepositives}$$

Figure 75:Precision formula

6.2.1.4 Recall

The recall is the ability of a classifier to find all positive instances. For each class, it is defined as the ratio of true positives to the sum of true positives and false negatives [4].

$$Recall = \frac{truepositives}{truepositives + falsenegatives}$$

Figure 76:Recall the formula.

6.2.1.5 F1-score

The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy [4].

$$F1 = 2 \times \frac{precision \times recall}{precision + recall}$$

Figure 77:F1-score formula

6.2.1.6 ROC Curve

A ROC curve is a plot of the true positive rate (Sensitivity) in the function of the false positive rate (100-Specificity) for different cut-off points of a parameter. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold. The Area Under the ROC curve (AUC) is a measure of how well a parameter can distinguish between two diagnostic groups (diseased/normal) [5].

6.2.1.7 Model Loss

One of the most often used graphs for debugging neural networks is the loss curve during training [8].

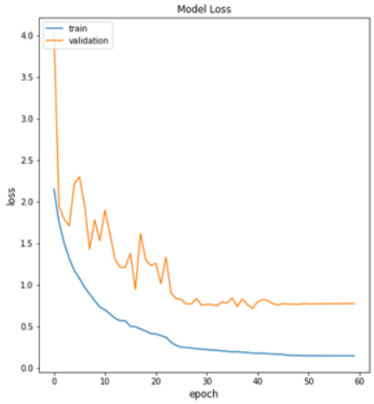


Figure 78:Model loss Example

6.2.1.8 Model Accuracy

Accuracy is often graphed and monitored during the training phase though the value is often associated with the overall or final model accuracy. Accuracy is easier to interpret than loss [8].

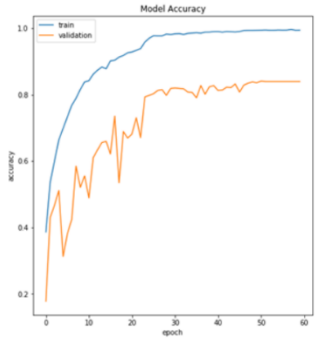


Figure 79:Model Accuracy Example

6.3 Experiments results

In this section, we examine the findings of model experiments and modified models to choose the most appropriate model for prediction. The models we tried are described in Section 6.3.1 for the machine learning model and Section 6.3.2 for the deep learning model.

6.3.1 Machine learning

6.3.1.1 Support Vector Machine

- **Default:**

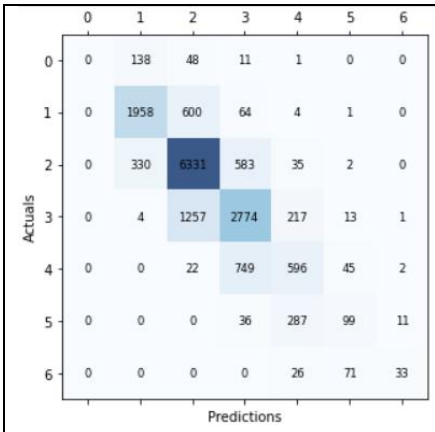


Figure 80:Confusion matrix of SVM

	precision	recall	f1-score	support
198	0.00	0.00	0.00	0
2627	0.77	0.75	0.81	1
7281	0.81	0.87	0.77	2
4266	0.65	0.65	0.66	3
1414	0.46	0.42	0.51	4
433	0.30	0.23	0.43	5
130	0.37	0.25	0.70	6
accuracy			0.72	16349
macro avg	0.55	0.45	0.48	16349
weighted avg	0.70	0.72	0.71	16349

r2: 0.5790604702424811

RMSE: 0.5790604702424811

Figure 81:SVM performance

Figure 80 above is the confusion matrix of the SVM classifier which visualizes the performance of the algorithm and helps to compare the actual vs predicted values. The matrix uses to estimate the 'Energy Efficiency Rating Band,' which ranges from A to G. As well, can measure Accuracy, Precision, Recall, and F1 Score. The overall TP of the outcome where the model correctly predicts positive class is 11791 from 16345. Figure 81 represents the metric to evaluate the performance. The overall accuracy is calculated by summing the number of correctly classified values and dividing by the total number of values is 72 .14%, the recall is 55%, the Precision is 45%, F1 Score is 48%.

- **After optimization**

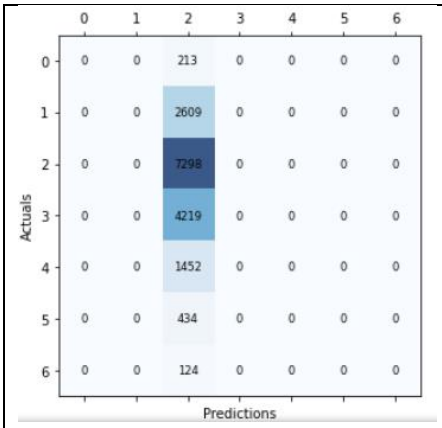


Figure 82:Confusion matrix of SVM

	precision	recall	f1-score	support
213	0.00	0.00	0.00	0
2609	0.00	0.00	0.00	1
7298	0.62	1.00	0.45	2
4219	0.00	0.00	0.00	3
1452	0.00	0.00	0.00	4
434	0.00	0.00	0.00	5
124	0.00	0.00	0.00	6
accuracy			0.45	16349
macro avg	0.06	0.14	0.09	16349
weighted avg	0.20	0.45	0.28	16349

r2: 1.0887016443516988

RMSE: 1.0887016443516988

Figure 83:SVM performance

Figure 82 above is the confusion matrix of the SVM classifier which visualizes the performance of the algorithm. The overall TP of the outcome where the model correctly predicts positive class is 7298 from 16349. Figure 83 represents the metric to evaluate the performance. The overall accuracy is 45%, the recall is 14%, the Precision is 6%, F1 Score is 9%. The performance is very low because of the failure in transformation from multi-class to a binary class classification problem.

6.3.1.2 Random Forest

- **Default:**

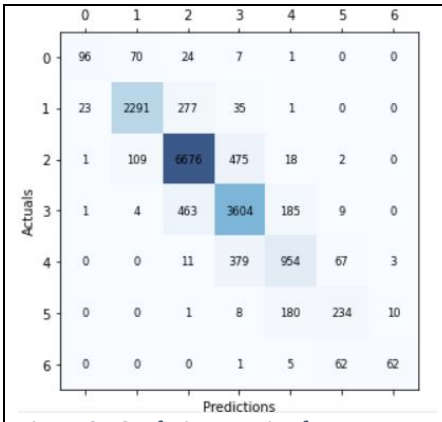


Figure 84:Confusion matrix of RF

	precision	recall	f1-score	support
198	0.60	0.48	0.79	0
2627	0.90	0.87	0.93	1
7281	0.91	0.92	0.90	2
4266	0.82	0.84	0.80	3
1414	0.69	0.67	0.71	4
433	0.58	0.54	0.63	5
130	0.60	0.48	0.83	6
accuracy			0.85	16349
macro avg	0.80	0.69	0.73	16349
weighted avg	0.85	0.85	0.85	16349

r2: 0.42152901284157984

RMSE: 0.42152901284157984

Figure 85:RF performance

The confusion matrix of the Random Forest classifier shown in Figure 84 visualizes the algorithm's performance and assists in comparing actual and predicted values. Furthermore, the four variables that can be calculated are TP, TN, FP, and FN. For example, the total diagonal representing the TP is 13917 from 16345. We can measure the performance using these components, as shown in Figure 85. The overall accuracy is 85.12%, the recall is 80%, the precision is 69%, and the F1 Score is 73%.

- **After optimization:**

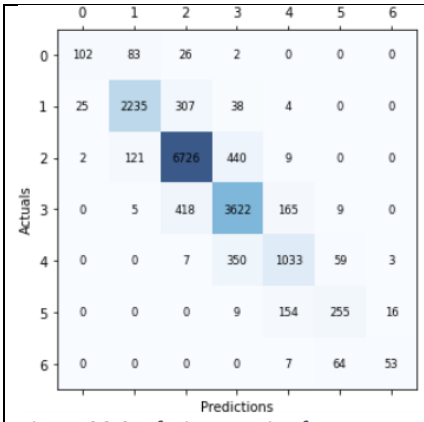


Figure 86:Confusion matrix of RF

	precision	recall	f1-score	support
213	0.60	0.48	0.79	0
2609	0.88	0.86	0.91	1
7298	0.91	0.92	0.90	2
4219	0.83	0.86	0.81	3
1452	0.73	0.71	0.75	4
434	0.62	0.59	0.66	5
124	0.54	0.43	0.74	6
accuracy			0.86	16349
macro avg	0.79	0.69	0.73	16349
weighted avg	0.86	0.86	0.86	16349

r2: 0.40758602599357424

RMSE: 0.40758602599357424

Figure 87:RF performance

Figure 86 above is the confusion matrix of the Random Forest classifier which visualizes the performance of the algorithm and assists in comparing actual and predicted values. Furthermore, the four variables that can be calculated are TP, TN, FP, and FN. For example, the total of diagonal representing the TP is 14026 from 16349. We can measure the performance using these components, as shown in Figure 87. The overall accuracy is 86%, the recall is 69%, the precision is 79%, and the F1 Score is 73%.

6.3.1.3 XGBoost.

- **Default:**

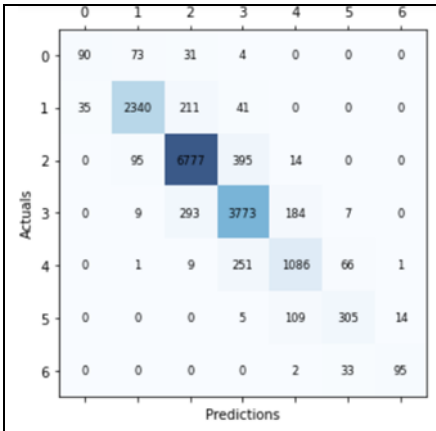


Figure 88:Confusion matrix of XGBoost

	precision	recall	f1-score	support
198	0.56	0.45	0.72	0
2627	0.91	0.89	0.93	1
7281	0.93	0.93	0.93	2
4266	0.86	0.88	0.84	3
1414	0.77	0.77	0.78	4
433	0.72	0.70	0.74	5
130	0.79	0.73	0.86	6
accuracy			0.88	16349
macro avg	0.83	0.77	0.79	16349
weighted avg	0.88	0.88	0.88	16349

r2: 0.37344085318831977

RMSE: 0.37344085318831977

Figure 89:XGBoost performance

The confusion matrix of the XGBoost classifier, shown in Figure 88, visualizes the algorithm's performance and assists in comparing actual and predicted values. It is also used to calculate the 'Energy Efficiency Rating Band,' which ranges from A to G. (0 to 6). Moreover, four variables can be calculated: TP, TN, FP, and FN. For example, the TP is 14466 from 16345 which represents the diagonal total. We can assess performance using these components; as shown in Figure 89, the overall accuracy is 88.48%, the recall is 83%, the precision is 77%, and the F1 Score is 79%.

- **After optimization:**

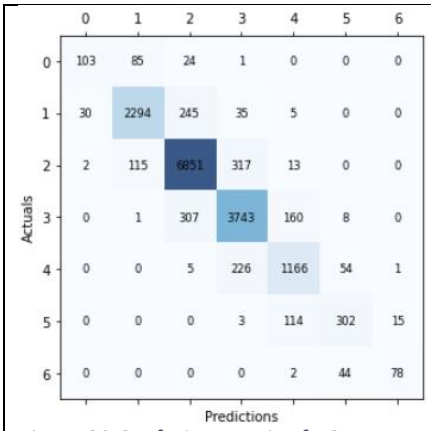


Figure 90:Confusion matrix of XGBoost

	precision	recall	f1-score	support
213	0.59	0.48	0.76	0
2609	0.90	0.88	0.92	1
7298	0.93	0.94	0.92	2
4219	0.88	0.89	0.87	3
1452	0.80	0.80	0.80	4
434	0.72	0.70	0.74	5
124	0.72	0.63	0.83	6
accuracy			0.89	16349
macro avg	0.83	0.76	0.79	16349
weighted avg	0.89	0.89	0.89	16349

r2: 0.3619629643977013

RMSE: 0.3619629643977013

Figure 91:XGBoost performance

The confusion matrix of the XGBoost classifier, shown in Figure 90, visualizes the algorithm's performance. Moreover, four variables can be calculated: TP, TN, FP, and FN. For example, the TP is 14537 from 16349 which represents the diagonal's total. We can assess performance using these components; as shown in Figure 91, the overall accuracy is 89%, the recall is 76%, the precision is 83%, and the F1 Score is 79%.

6.3.2 Deep learning

To evaluate the model in this section, use A learning curve and confusion matrix, Precision, Recall, F1 Score, R^2, RMSE .and the ROC curve created for the model found in the appendix.

6.3.2.1 Default Model

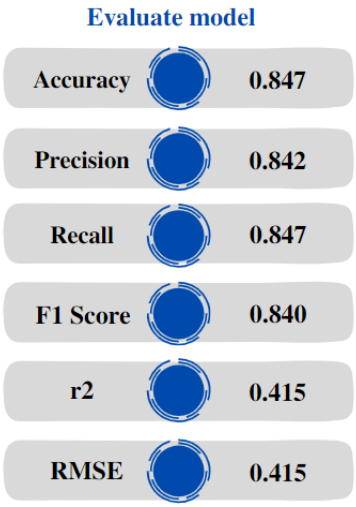


Figure 92:Evaluate default model

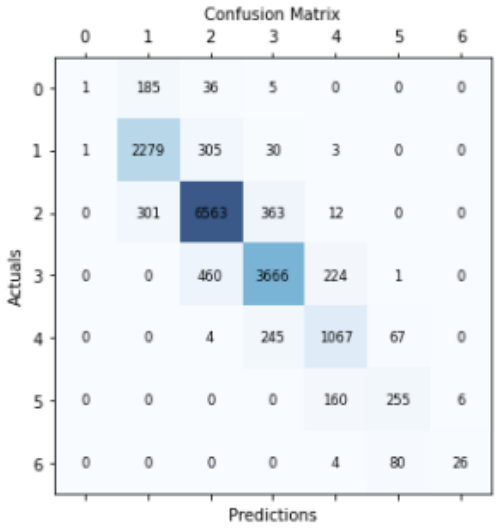


Figure 93:Confusion matrix of default

In models, we had to assign a number for epochs, we chose 20, figure 93 is a Multiclass Confusion Matrix. The 7x7 confusion matrix enables the evaluation of the NN model's performance, where N = 7 is the number of target classes. Additionally, this confusion matrix provides a variety of information regarding the model's performance can show in the figure 92.

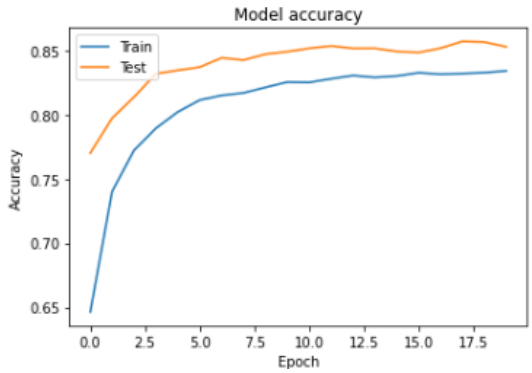


Figure 94:Model accuracy for default model

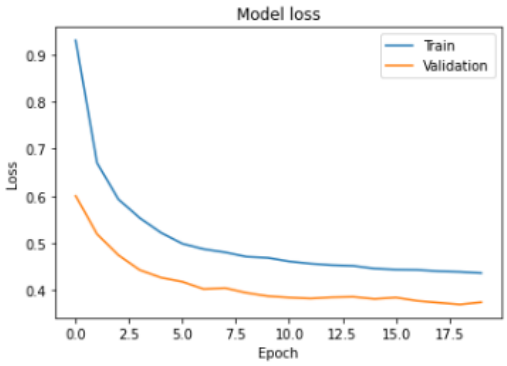


Figure 95:Model loss of default model

The loss curve shows the validation loss is significantly lower than the training loss, indicating that the validation dataset is easier to predict than the training dataset. An accuracy curve is a graph that shows the relationship between the number of training examples used to train a model and its accuracy on unseen data. Typically, the accuracy starts low and increases as the model trains, reaching a

maximum accuracy value. because the accuracy starts from lower and learning good and picking its maximum on validation and test data both.

6.3.2.2 Model A "Manual optimization"

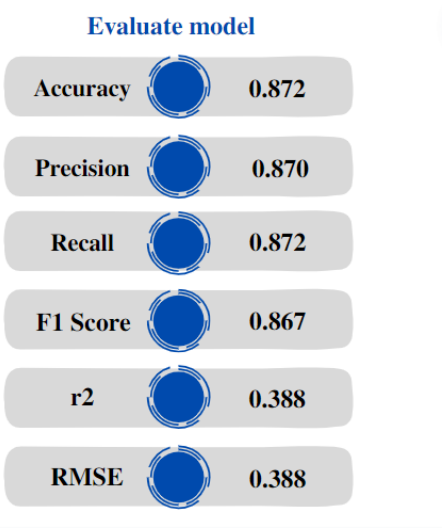


Figure 96:Evaluate model A

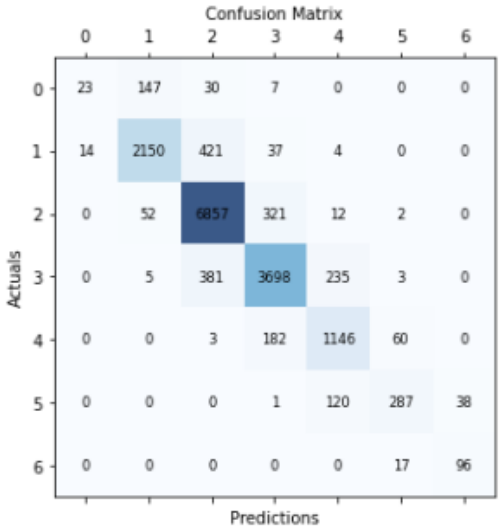


Figure 97:Confusion matrix of model A

Figure 97 displays a confusion matrix that may use to visualize the performance of a prediction model. Each item in the confusion matrix represents the number of predictions produced by the model as its prediction the categories correctly or incorrectly. We have more than one class in this matrix to estimate the 'Energy Efficiency Rating Band' which ranges from A to G. Furthermore, figure 96 displays some of the most popular performance metrics.

For example, the test set contains 7244 values from class C, and the model predicts 6857, indicating a prediction precision of 94.658% for this class. As a sequence, Class D with a rate of 85.562%.

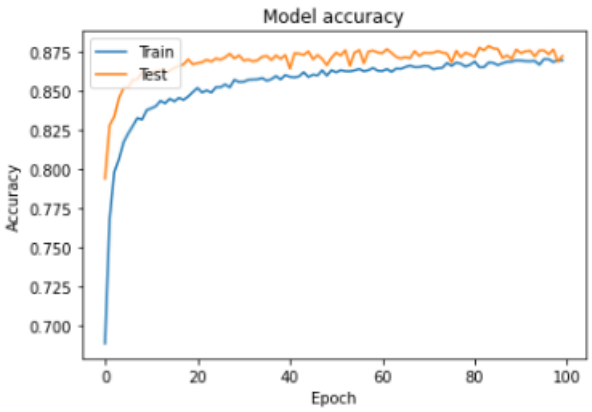


Figure 98:Model accuracy for model A

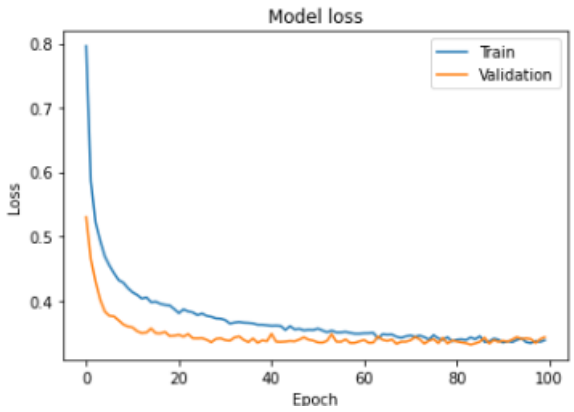


Figure 99:Model loss for model A

The loss curve vibrates a little, indicating that our model is overfitted. Furthermore, training loss decreases with experience, while validation loss decreases to a point before increasing significantly. The point at which training may be discontinued as an experiment may be the turning point in the loss of validation. Model accuracy shows a kind of low overfitting and provide a chance to optimize it.

6.3.2.3 Model B "KerasClassifier and grid search to optimize."

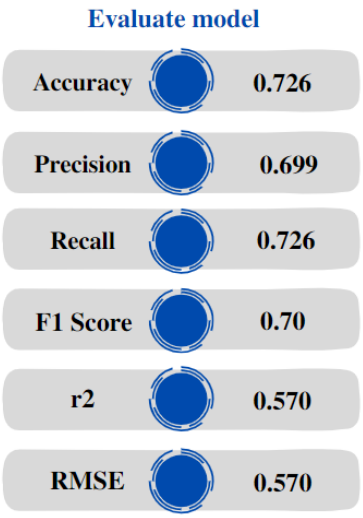


Figure 100:Evaluate model B

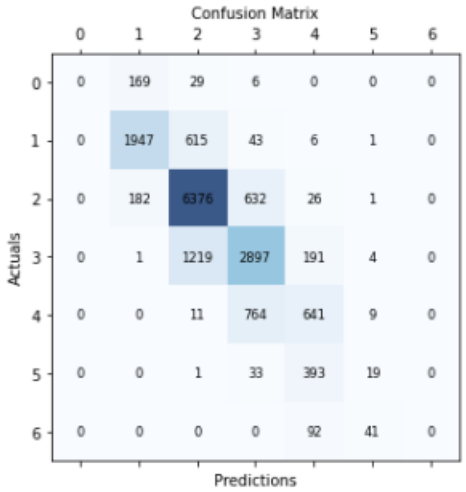


Figure 101:Confusion matrix of model B

Figure 101 demonstrates how the Confusion Matrix for Multiclass Prediction is used to evaluate a model's performance. It is a comparison of actual and expected values. The metrics derived from the confusion matrix are shown in the other figure 100.

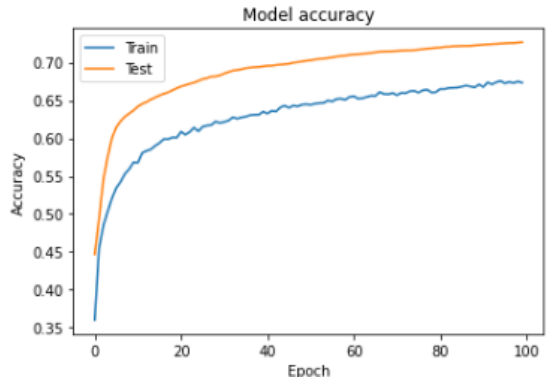


Figure 102 Model accuracy for model B

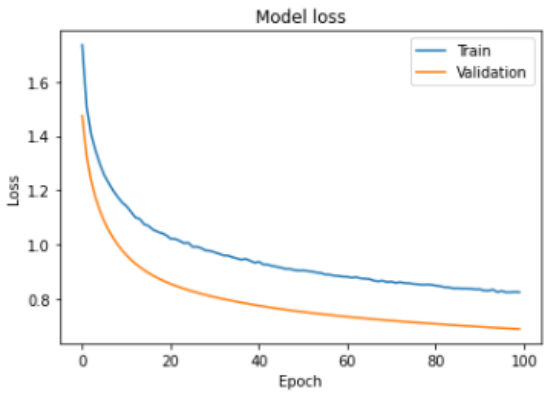


Figure 103 Model loss for model B

A plot of learning curves shows training loss is constant regardless of training. The training loss is constant regardless of training. Training loss continues to decrease until the completion of training. This shows that the model is capable of future learning and possible improvement. The model accuracy shows the accuracy begins low and gradually increases as the model trains, achieving a high accuracy value. It's performing very well.

6.3.2.4 chatbot result

After evaluating each model, we choose XGBoost model to be the core of the chatbot because it has the highest performance and accuracy. Figure 104 show the first message that shows up to the user, figure 105 indicate the prediction after the user enters the features of his house.

```
Welcome to our Energy Efficiency Prediction chatbot
==================================================
Please choose one of the choices:
1- Calculate energy efficiency rating band
2- exit from chat
```

Figure 104 beginning of chatbot

```
The prediction of Energy Efficiency Rating Band is:  D
```

Figure 105 prediction result

6.4 Discussion

In this section we will discuss the objectives, results of this project, and limitations. Our objective is to utilize the data of residential sector and machine learning techniques to assess evaluating energy efficiency in the building industry with high reliability, and to see the potential of machine learning in this field.

We aimed to classify the energy efficiency of houses into seven classes ranging from A to G based on a set of features indicating details about the houses chosen by using the mutual information method after the data has been cleaned and pre-processed. It has been shown that by using the correct and limited set of features the model can be carried out with a good accuracy while still being simple enough to be used as the chatbot's core we aimed to make to complement our project by assisting the users in evaluating their houses' energy efficiency. We experimented different machine learning models such as RF and XGBoost, and deep learning techniques to choose the most suited one.

	Accuracy	Precision	Recall	F1-score
Support Vector Machine	72%	0.45	0.55	0.48
Random Forest	86%	0.69	0.79	0.73
XGboost	89%	0.76	0.83	0.79
Deep Learning - Model A	87%	0.87	0.87	0.87

Figure 106 performance of each model

These results indicates that the XGboost is the most accurate model to be used in our project, since it gives the highest accuracy. However, as we can see in the Figure 104, the other models have quite similar accuracies. therefore, we would focus on f-score (harmonic mean of precision and recall) as our main evaluation metric for a number of reasons, accuracy is the percentage of all correctly classified observations both positive and negative, while f1-score is calculated from precision (correct positive predictions relative to total positive predictions) and recall (Correct positive predictions relative to total actual positives) which are calculated on the positive class, in our problem we care more about true positive than true negative. Another reason is that accuracy does not consider the distribution of the data, so it is best used when the data is even balanced, unlike f1-score.

So, as shown in the Figure 104, XGboost has the highest accuracy then model A of deep learning comes after, but if we look at the f1-score, we can clearly see that model A of deep learning has the highest score.

These results encourage us to continue in this line of research in the immediate future, while previous research has shown the accuracy of the model that was developed using the same dataset that we used is not quite accurate compared to our model since we were focusing on how to make the prediction as reliable as we can, while still being able to implement through the chatbot to be user-friendly [9].

The most features of the dataset are categorical which is often imbalanced, meaning that one class may be significantly more represented than other. This can lead to models that are biased towards the more frequent class and not accurately represent the minority class. So, to solve this problem split the columns to multiple. Second, the tune parameter process by using grid search and random search take long time and consume the power of device. So, to handle it split this job in our device and do it in parallel.

6.5 Conclusion

We examined several categorical prediction models, including RF, XGBoost. We compare the results of each model with the improved models based on numerous metrics such as accuracy, recall, f1-score, confusion matrix, and so on. We concluded that the three best models achieving the expected outcomes were RF, XGBoost, and Model A from deep learning models. Consequently, can be substituted the machine learning model in the chatbot using one of these models to ensure high performance in chatbot prediction.

Chapter VII: Conclusion and future work

7.1 Introduction

Improving building energy efficiency is key to the global carbon emission reduction task. Accurate prediction and understanding of building energy efficiency is beneficial for better utilizing and saving energy in the building sector by using machine learning models and CNN with various architectures. In this project, based on the processed data, many models were experienced with the dataset to optimize the building industry, while experiencing different models, many obstacles were encountered.

In this chapter, Section 7.2 defines the final conclusion that we reach in our project. Section 7.3 illustrates the limitations and difficulties we have faced during the models' buildings and executions. At the end of the chapter, our aspirations for this project and improvements that will increase the effectiveness of the models are given as future work in Section 7.4.

7.2 Conclusion

In this project, a prediction algorithm for a residential building is developed based on its specifications and the type of material used. The dataset used in the project was provided by gov.scot, and it was processed and cleaned before being used to train machine learning models and deep ANN. Therefore, the accuracy of various machine learning models was then compared, and the result indicated

that the XGBoost model is best appropriate for predicting the energy efficiency rating band. Other report findings include:

- SVM had the lowest prediction accuracy (72%) but was the fastest in calculations.
- RF is a popular machine learning technique due to the minimal necessary data pre-processing. however, Random forests require less encoding than other models, so running them takes less time. Additionally, can work with numerical and categorical data .so it contributed to raising the accuracy rate and which was 86%.
- XGBoost predicts the energy efficiency rating band and achieves 89%, It has improved the result of SVM and RF, but it takes longer to execute and optimize.
- Deep ANN after improvement performs excellently at 87%. Deep ANN is highly recommended in cases where computational time is not a main consideration.
- We created a prototype of a chatbot that helps predict the energy efficiency rating band for the home after training the models to select the best model with high accuracy.

7.3 Difficulties & Limitations

It's challenging to find data on which you can implement your idea, which is why we fall into some difficulties when searching for an efficient data for us. One of the limitations in our dataset as general, that it is contain categorical features more than other type of data, which mean we need to preprocess it carefully to build a perfect model. The data contains some features that has textual content and multiple value in the same column that need to be separated into a more understandable feature to use it on chatbot later. Other limitation is that we found some difficulties in running some codes because the size of the dataset is large and difficult for our devices to process it. Moreover, we tried to search for a chatbot builder based on machine learning and can't find one that serves our idea.

7.4 Future work

A person requires energy because he uses it continuously in all aspects of his life. As a result, always will be a work to maintain it and ensure its sustainability. Our initial role is to contribute to the improvement of models for predicting energy use in houses. As well as to the improvement of the features selection process. Furthermore, additional research can be conducted to improve the ANN model's hyperparameters, such as learning rate, momentum, epochs, frequencies, and batch size.

We will attempt to enhance the house's features to improve the predictability of secondary energy sources such as solar energy. Concerning the chatbot, we will work on developing a graphical interface that is simple to use and reduces the response time.

References:

Chapter 1

[1] "Saudi Arabia Energy .]متصل["،Available:
https://www.worldometers.info/energy/saudi-arabia-

energy/#:~:text=Energy%20Consumption%20in%20Saudi%20Arabia,its%20annual
%20energy%20consumption%20needs.

[2] "Energy Consumption in Saudi Arabia .]متصل["،Available:
https://www.worldometers.info/energy/saudi-arabia-

energy/#:~:text=Energy%20Consumption%20in%20Saudi%20Arabia,its%20annual
%20energy%20consumption%20needs..

[3] "Energy & Sustainability .]متصل["،Available:
https://www.vision2030.gov.sa/thekingdom/explore/energy./

[4] JEAN-CHRISTOPHE-CHOUINARD ،"Learn Ensemble Learning Algorithms (Machine Learning)،"5May2022.]متصل[.Available:

https://www.jcchouinard.com/ensemble-learning./

[5] " [متصل]. 2022 Domestic Energy Performance Certificates - Dataset to Q1" Available:https://statistics.gov.scot/data/domestic-energy-performance-

certificates.

[6] "MaoranSun/building energy efficiency: Building Energy Efficiency Glasgow،" 27] July 2022 .]متصل[.Available: https://zenodo.org/record/6913572#.YyxtsKRBy3A.

Chapter 2

[١] R. A. J. S. Rajesh Kumar ،"Energy analysis of a building using artificial neural network: A review "،*https://doi.org/10.1016/j.enbuild.2013.06.007* ،/٠١/١٠/٢٠١٣ رقم ،pp. 352-358 .٢٠١٣ ،

[٢] C. H. Q. N. J. X. F. Z. Q. Z. Maoran Sun ،"Understanding building energy efficiency with administrative and emerging urban big data by deep learning in Glasgow "،*sciencedirect* المجلد ، https://doi.org/10.1016/j.enbuild.2022.112331 .٢٠٢٢ ،١٥/١٠/٢٠٢٢ رقم ،.

[٣] E. Lutins ،"Ensemble Methods in Machine Learning: What are They and Why Use Them?،" 02 08 2017 .]متصل[.Available: https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f.

[٤] M. P. Monika Goyal ،"A Systematic Analysis for Energy Performance Predictions in Residential Buildings Using Ensemble Learning "،*Arabian Journal for Science and*

*Engineering*٢% من ١% المجلدات ،https://doi.org/10.1007/s13369-020-05069-2 ،٠١/٠٤/٢٠٢١ ، رقم ، ٢٠٢١. pp. 3155-3168

[٥] D. Z. ,. M. G. M. S. Yunjeong Mo ،"Effective Features to Predict Residential Energy Consumption Using Machine Learning المجلد "، https://www.researchgate.net/publication/333758716 ،١٣/٠٦/٢٠١٩ ،pp. 284-291 ،٢٠١٩.

[٦] S. P. A. K. Anurag Verma ،"ANN-based energy consumption prediction model up to 2050 "، *Environmental Progress & Sustainable Energy*/١٠,١٠٠٢ المجلد ،ep.13544 ،٣٠/١٠/٢٠٢٠ ، رقم ، ٢٠٢٠.

[٧] A. S. S. ,. S. A. ,. M. J. A. Mohamed Mohana ،"Small-Scale Solar Photovoltaic Power Prediction for Residential Load in Saudi Arabia Using Machine Learning المجلد "، https://doi.org/10.3390/en14206759 ،٢٠٢١.

[٨] F. P. R. I. G. a. M. R. Saleh Seyedzadeh1 ،"Machine learning for estimation of building energy consumption and performance: a review "،*Visualization in Engineering* من ١% المجلدات ، ٢%https://doi.org/10.1186/s40327-018-0064-7 .٢٠١٨ ،٠٢/١٠/٢٠١٨ ، رقم

[٩] L. S. G. D. Fazel Khayatian ،"Application of neural networks for evaluating energy performance certificates of residential buildings "،*Energy and Buildings* المجلد ، https://www.sciencedirect.com/science/article/pii/S0378778816303322 ،٠١/٠٨/٢٠١٦ ،pp. رقم ، ٢٠١٦. 45-54

[١٠] H. C. L. Z. Z. F. Yang Liu ،"Enhancing building energy efficiency using a random forest model: A hybrid prediction approach "،*Energy Reports* المجلد ، https://www.sciencedirect.com/science/article/pii/S2352484721006016 ،٠١/١١/٢٠٢١ ،pp. رقم ، ٢٠٢١. 5003-5012

[١١] C. V. M. R. ·. A. A. L. ·. P. H. C. V. · ،"A Chatbot Solution for Self-Reading Energy Consumption via Chatting "،*Journal of Control, Automation and Electrical Systems* المجلد ، ،٠١/٠٢/٢٠٢٢https://doi.org/10.1007/s40313-021-00818-6 ،pp. 229-240 ،٢٠٢٢ رقم

[١٢] U. S. E. P. Agency ،"https://www.epa.gov/statelocalenergy/local-energy-efficiency-benefits-and-opportunities .[متصل] "،

[١٣] https://www.gov.uk/government/collections/uk-territorial-greenhouse-gas-emissions-national-statistics ،"2021 UK greenhouse gas emissions, provisional figures،" 2021.

[١٤] R. f. t. Future ،"https://www.rff.org/publications/explainers/energy-efficiency-101 .[متصل] "،/ Available: https://www.rff.org/publications/explainers/energy-efficiency-101./

[١٥] energystar.gov ،"https://www.energystar.gov/about/about_energy_efficiency .[متصل] "، Available: https://www.energystar.gov/about/about_energy_efficiency.

[١٦] H. L. Siddharth Misra ،Chapter 9 - Noninvasive fracture characterization based on the classification of sonic wave travel times, Machine Learning for Subsurface Characterization ، Gulf Professional Publishing ،٢٠٢٠.

[١٧] V. E. S. P. Bondarenko ،"Artificial neural networks "،Bondarenko, Vladimir E., Salem Press Encyclopedia of Science.٢٠٢٢ ،

[١٨] spiceworks ،"https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-chatbot .[متصل] "،/Available: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-chatbot./

Chapter 3

[1] Hox, J. J., & Boeije, H. R. (2005). “Data collection, primary vs. Secondary". *Encyclopedia of social measurement*, *1*(1), 593-599. Available: http://joophox.net/publist/ESM_DCOL05.pdf

[2] Scottish Government ،"Domestic Energy Performance Certificates - Dataset to Q1 2022،" 08/08/2022. [متصل]. Available: https://statistics.gov.scot/data/domestic-energy-performance-certificates.

Chapter 4

[1] javatpoint ،"Data Preprocessing in Machine learning .[متصل] "،Available: https://www.javatpoint.com/data-preprocessing-machine-learning.

[٢] M. Bolle ،"5 Easy Ways to Detect Outliers in Python،" 01 04 2022 .[متصل] .Available: https://datasciencesphere.com/analytics/5-easy-ways-to-detect-outliers-in-python./

Chapter 5

[1] Available: https://scikit-[متصل]" ،"scikit-learn.org learn.org/stable/modules/generated/sklearn.compose.ColumnTransformer.html

[2] educba ،"keras-optimizers،" [متصل]. Available: https://www.educba.com/keras-optimizers/.

[3] “Python Keras | keras.utils.to_categorical()،" 10 Jan 2023. Available: https://www.geeksforgeeks.org/python-keras-keras-utils-to_categorical/.

[4] C. Liu ،"SVM Hyperparameter Tuning using GridSearchCV،" 10th March 2020 . [متصل]. Available: https://www.vebuso.com/2020/03/svm-hyperparameter-tuning-using-gridsearchcv/.

[5] "What is deep learning?،" [متصل]. Available: https://www.ibm.com/topics/deep-learning.

Chapter 6

[1] F. Milletari, S.-A. Ahmadi, C. Kroll, A. Plate, V. Rozanski, J. Maiostre, J. Levin, O. Dietrich, B. Ertl-Wagner, K. Bötzel, and N. Navab, “Hough-CNN: Deep Learning for segmentation of deep brain regions in MRI and ultrasound,” arXiv.org, 31-Jan-2016. [Online]. Available: https://arxiv.org/abs/1601.07014.

[2] wikipedia ،"Confusion matrix،" [متصل]. Available: https://en.wikipedia.org/wiki/Confusion_matrix.

[3] M. Amer ،"Classification Evaluation Metrics: Accuracy, Precision, Recall, and F1 Visually Explained،" 7 jun 2022. Available: https://txt.cohere.ai/classification-eval-metrics/.

[4] "Understanding the Classification report through sklearn،" 7 july 2018. Available: https://muthu.co/understanding-the-classification-report-in-sklearn/.

[5] medcalc ،"ROC curve analysis،" Available: https://www.medcalc.org/manual/roc-curves.php.

[6] A. Bajaj ،"Performance Metrics in Machine Learning [Complete Guide]،" 26 jan 2023. Available: https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide.

[7] THE INVESTOPEDIA TEAM ،"R-Squared vs. Adjusted R-Squared: What's the Difference?،" 11 feb 2022. Available: https://www.investopedia.com/ask/answers/012615/whats-difference-between-rsquared-and-adjusted-rsquared.asp#:~:text=What%20Is%20the%20Difference%20Between,and%20R%2Dsquared%20does%20not.

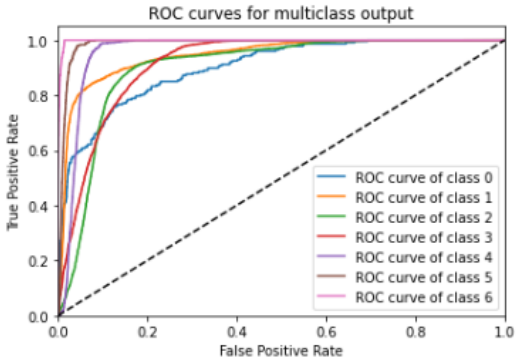
[8] "Accuracy and Loss،" Available: https://machine-learning.paperspace.com/wiki/accuracy-and-loss.

[9] C. H. Q. N. J. X. F. Z. Q. Z. Maoran Sun ،"Understanding building energy efficiency with administrative and emerging urban big data by deep learning in Glasgow "،*sciencedirect* المجلد ، ،. رقم ٢٠٢٢/١٠/١٥، ٢٠٢٢. https://doi.org/10.1016/j.enbuild.2022.112331

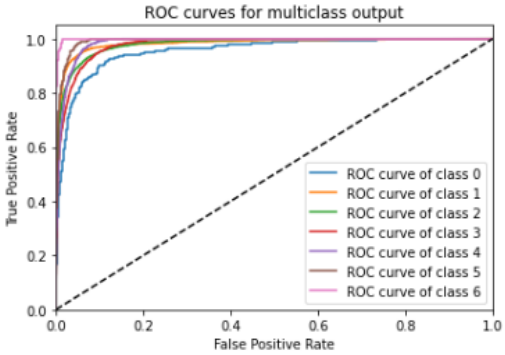
Appendix

Deep learning

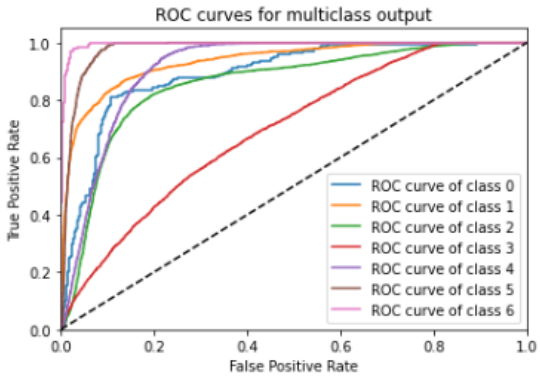
- Default Model.



- Model A



- Model B



Code of pre-processing:

```
# In[2]:

import numpy as np

import pandas as pd

from sklearn.preprocessing import StandardScaler

from scipy import stats

from sklearn.model_selection import train_test_split

import warnings

import matplotlib.pyplot as plt

import datetime


# In[3]:

pd.set_option('display.max_rows', 99999)

pd.set_option('display.max_columns', 500)

warnings.filterwarnings("ignore")


# In[4]:

epc_49 = pd.read_csv('2022_49.csv')

##2022_49.csv contains only chosen features(house characteristics) from the original dataset

epc_49.shape


# In[5]:

epc_49.head(1)



# In[6]:

##drop Photovoltaic Supply and improvements ( because they are correlated(reduce) with energy consumption)

epc_49 = epc_49.drop(['Photovoltaic Supply', 'Improvements'], axis=1)


# In[7]:

#drop columns having more than 20% null
```

```
epc_49 = epc_49.replace([0,' ','NULL'],np.nan)
null_percentage = epc_49.isnull().sum()/epc_49.shape[0]*100
col_to_drop = null_percentage[null_percentage>20].keys()
epc_49 = epc_49.drop(col_to_drop, axis=1)



# In[8]:
#remove outliers using stats
epc_num = epc_49.select_dtypes(include= ['number'])
min = 0
maxPercentile = 0.9999
for _ in epc_num:
    epc_49[] =  epc_49[].clip(0,(epc_49[_].quantile(maxPercentile)))



# In[9]:
#impute nulls with most frequent values
epc_49 = epc_49.apply(lambda x: x.fillna(x.value_counts().index[0]))



# In[10]:
epc_49 =epc_49.drop(['Current Environmental Impact Rating','Current Environmental Impact Rating
Band',
                'Space Heating','Water Heating', 'Main Heating 1 Category', 'Main Heating 1 Fuel Type',
                 'Main Heating 1 Control','Multiple Glazing Type', 'Multiple Glazed Proportion',
                'Current energy efficiency rating'
                ], axis=1)


# Descriptive features handling
# In[11]:
descriptive_columns =
['WALL_DESCRIPTION','ROOF_DESCRIPTION','FLOOR_DESCRIPTION','WINDOWS_DESCR
IPTION',
```

```
'MAINHEAT_DESCRIPTION','MAINHEATCONT_DESCRIPTION','SECONDHEAT_DESCRIPTION','HOTWATER_DESCRIPTION',
                    'LIGHTING_DESCRIPTION',
                    #'AIR_TIGHTNESS_DESCRIPTION'
                   ]


for col in descriptive_columns:
    epc_49[col] = epc_49[col].str.replace(r"\|(.*)","")
    epc_49[col] = epc_49[col].str.replace(r"\|(.*)","")
    epc_49[col] = epc_49[col].str.replace('  ',' ')
    epc_49[col] = epc_49[col].str.lower()
    epc_49[col] = epc_49[col].str.replace(", as built","")
    epc_49[col] = epc_49[col].str.replace("assumed","")
    epc_49[col] = epc_49[col].str.replace('(',")
    epc_49[col] = epc_49[col].str.replace(')',")
    epc_49[col] = epc_49[col].str.replace("[ \t]+$","")
    epc_49[col] = epc_49[col].str.replace("w/m?k","w/m²k")
    epc_49[col] = epc_49[col].str.replace('Average thermal transmittance 1 ','Average thermal transmittance 1.00 ')
    epc_49[col] = epc_49[col].str.replace('Average thermal transmittance =','Average thermal transmittance')    # standardising the unit used
    epc_49[col] = epc_49[col].str.replace(r"w(.*?)k",'w/m²k')


# In[12]:

def roof_floor_cleanup(df):
    '''cleaning ROOF_DESCRIPTION and FLOOR_DESCRIPTION'''
   # standardising descriptions
    df['ROOF_DESCRIPTION'] = df['ROOF_DESCRIPTION'].str.replace("another dwelling above","other premises above")
    df['ROOF_DESCRIPTION'] = df['ROOF_DESCRIPTION'].str.replace("+","")
    df['FLOOR_DESCRIPTION'] = df['FLOOR_DESCRIPTION'].str.replace("another dwelling above","other premises above")
    df['FLOOR_DESCRIPTION'] = df['FLOOR_DESCRIPTION'].str.replace("another dwelling below","other premises below")
```

```
    df['FLOOR_DESCRIPTION'] = df['FLOOR_DESCRIPTION'].str.replace("  \+"," +")
    df['FLOOR_DESCRIPTION'] = df['FLOOR_DESCRIPTION'].str.replace("[ \t]+$","")
    return df
epc_49 = roof_floor_cleanup(epc_49)


# In[13]:
def insulation_extract(df):
    '''takes the second part after ',' that indicates insulation in description columns '''
    df['wall_insulation'] = df['WALL_DESCRIPTION'].str.extract('(?<=,\s)([^,]+)')
    df['roof_insulation'] = df['ROOF_DESCRIPTION'].str.extract('(?<=,\s)([^,]+)')
    df['floor_insulation'] = df['FLOOR_DESCRIPTION'].str.extract('(?<=,\s)([^,]+)')
    df[['wall_insulation','roof_insulation','floor_insulation']] =
df[['wall_insulation','roof_insulation','floor_insulation']].apply(lambda x:
x.fillna(x.value_counts().index[0]))
    return df
epc_49 = insulation_extract(epc_49)


# In[14]:
def wall_insulation(df):
    '''returns df with a new column 'insulated_wall' classifying wall insulation 1 if insulated 0 if not'''
     ##=============insulated=============##
    df.loc[df['wall_insulation'].str.contains('with external insulation') , 'insulated_wall'] = 1
    df.loc[df['wall_insulation'].str.contains('with internal insulation') , 'insulated_wall'] = 1
    df.loc[df['wall_insulation'].str.contains('with additional insulation') , 'insulated_wall'] = 1
    df.loc[df['wall_insulation'].str.contains('partial insulation') , 'insulated_wall'] = 1
    df.loc[df['wall_insulation'].str.contains('with insulation') , 'insulated_wall'] = 1
    df.loc[df['wall_insulation'].str.contains('filled cavity') , 'insulated_wall'] = 1
    df.loc[df['wall_insulation'].str.contains('insulated') , 'insulated_wall'] = 1
    df.loc[df['wall_insulation'].str.contains('average thermal transmittance') , 'insulated_wall'] = 1
     ##=============no insulation=============##
    df.loc[df['wall_insulation'].str.contains('no insulation') , 'insulated_wall'] = 0
    return df
epc_49 = wall_insulation(epc_49)
```

```
# In[15]:
epc_49['wall_insulation'].value_counts()

# In[16]:
def roof_insulation(df):
    '''returns df with a new column 'insulated_roof' classifying wall insulation 1 if insulated 0 if not'''
    ##=============insulated=============##
    df.loc[df['roof_insulation'].str.contains('loft insulation') , 'insulated_roof'] = 1
    df.loc[df['roof_insulation'].str.contains('insulated') , 'insulated_roof'] = 1
    df.loc[df['roof_insulation'].str.contains('limited insulation') , 'insulated_roof'] = 1
    df.loc[df['roof_insulation'].str.contains('with additional insulation') , 'insulated_roof'] = 1
    df.loc[df['roof_insulation'].str.contains('average thermal') , 'insulated_roof'] = 1
    ##=============no insulation=============##
    df.loc[df['roof_insulation'].str.contains('no insulation') , 'insulated_roof'] = 0
    return df
epc_49 = roof_insulation(epc_49)

# In[17]:
epc_49['roof_insulation'].value_counts()

# In[18]:
def floor_insulation(df):
    '''returns df with a new column 'insulated_floor' classifying wall insulation 1 if insulated 0 if not'''
    ##=============insulated=============##
    df.loc[df['floor_insulation'].str.contains('insulated') , 'insulated_floor'] = 1
    df.loc[df['floor_insulation'].str.contains('limited insulation') , 'insulated_floor'] = 1
    df.loc[df['floor_insulation'].str.contains('average thermal') , 'insulated_floor'] = 1
    ##=============no insulation=============##
    df.loc[df['floor_insulation'].str.contains('no insulation') , 'insulated_floor'] = 0
    return df
epc_49 = floor_insulation(epc_49)
```

```
# In[20]:
epc_49['floor_insulation'].value_counts()



# In[66]:
def wall_type(df):
    '''returns df with a new column 'wall_type' indicating type/ material of the walls'''
    df['wall_type'] = df['WALL_DESCRIPTION'].str.extract('^([^,]+)')
    df.loc[df['wall_type'].str.contains('average ') , 'wall_type'] = 'Unknown'
    df.loc[df['wall_type'].str.contains('park home wall') , 'wall_type'] = 'park home wall'
    return df
epc_49 = wall_type(epc_49)

# In[67]:
def roof_type(df):
    '''returns df with a new column 'roof_type' indicating type/ material of the roofs'''
    df.loc[df['ROOF_DESCRIPTION'].str.contains('pitched') , 'roof_type'] = 'Pitched'
    df.loc[df['ROOF_DESCRIPTION'].str.contains('flat') , 'roof_type'] = 'Flat'
    df.loc[df['ROOF_DESCRIPTION'].str.contains('roof room') , 'roof_type'] = 'Roof rooms'
    df.loc[df['ROOF_DESCRIPTION'].str.contains('thatched') , 'roof_type'] = 'Thatched'
    df.loc[df['ROOF_DESCRIPTION'].str.contains('average thermal transmittance') , 'roof_type'] =
'Unknown'
    df.loc[df['ROOF_DESCRIPTION'].str.contains('other premises above') , 'roof_type'] = 'Unknown'
    return df
epc_49 = roof_type(epc_49)

# In[68]:
def floor_type(df):
    '''returns df with a new column 'floor_type' indicating type/ material of the floors'''
    df.loc[df['FLOOR_DESCRIPTION'].str.contains('suspended') , 'floor_type'] = 'Suspended'
    df.loc[df['FLOOR_DESCRIPTION'].str.contains('solid') , 'floor_type'] = 'Solid'
```

```
    df.loc[df['FLOOR_DESCRIPTION'].str.contains('to unheated space') , 'floor_type'] = 'To unheated space'

    df.loc[df['FLOOR_DESCRIPTION'].str.contains('to external air') , 'floor_type'] = 'To external air'

    df.loc[df['FLOOR_DESCRIPTION'].str.contains('average thermal transmittance') , 'floor_type'] = 'Unknown'

    df.loc[df['FLOOR_DESCRIPTION'].str.contains('other premises below') , 'floor_type'] = 'Unknown'

    df.loc[df['FLOOR_DESCRIPTION'].str.contains('other premises above') , 'floor_type'] = 'Unknown'

    return df

epc_49 = floor_type(epc_49)

# In[21]:

def windows_glazing(df):

    '''returns a df with new column 'windows_glazing' indicating the type of windows glazing '''

    df['windows_glazing'] = df['WINDOWS_DESCRIPTION'].str.replace("description:","")

    df['windows_glazing'] = df['windows_glazing'].str.replace("fully","")

    df['windows_glazing'] = df['windows_glazing'].str.replace("partial","")

    df['windows_glazing'] = df['windows_glazing'].str.replace("mostly","")

    df['windows_glazing'] = df['windows_glazing'].str.replace("full","")

    df['windows_glazing'] = df['windows_glazing'].str.replace("some","")

    df['windows_glazing'] = df['windows_glazing'].str.replace("^\s+","")

    df['windows_glazing'] = df['windows_glazing'].str.replace("triple glazing","triple glazed")

    df['windows_glazing'] = df['windows_glazing'].str.replace("multiple glazing throughout","multiple glazing")

    df['windows_glazing'] = df['windows_glazing'].str.replace("double glazing","double glazed")

    return df

epc_49 = windows_glazing(epc_49)

# In[23]:

epc_49['windows_glazing'].value_counts()


# In[24]:

def mainheat_sys(df):

    df['main_heat_system'] = df['MAINHEAT_DESCRIPTION'].str.extract('^([^,]+)')

#     df['main_heat_system'] = df['main_heat_system'].str.replace("no system present: ","")
```

```
    df['main_heat_system'] = df['main_heat_system'].str.replace("^[ \t]+","")
    df['main_heat_system'] =
df['main_heat_system'].fillna(df['main_heat_system'].value_counts().index[0])
    df['MHS_boiler'] = df['main_heat_system'].str.contains('boiler')
    df['MHS_radiators'] = df['main_heat_system'].str.contains('radiators')
    df['MHS_electric heaters'] = df['main_heat_system'].str.contains('electric heaters')
    df['MHS_electric heaters'] = df['main_heat_system'].str.contains('electric underfloor heating')
    df['MHS_electric heaters'] = df['main_heat_system'].str.contains('electric ceiling heating')
    df['MHS_electric heaters'] = df['main_heat_system'].str.contains('electric storage heaters')
    df['MHS_electric heaters'] = df['main_heat_system'].str.contains('portable electric heaters  for most
rooms')


    df['MHS_heat pump'] = df['main_heat_system'].str.contains('ground source heat pump')
    df['MHS_heat pump'] = df['main_heat_system'].str.contains('air source heat pump')
    df['MHS_heat pump'] = df['main_heat_system'].str.contains('water source heat pump')
    df['MHS_room heaters'] = df['main_heat_system'].str.contains('room heaters')
    df['MHS_community scheme'] = df['main_heat_system'].str.contains('community scheme')
    df['MHS_underfloor heating'] = df['main_heat_system'].str.contains('underfloor heating')
    df['MHS_warm air'] = df['main_heat_system'].str.contains('warm air')
    #====================#
    df['main_heat_system_2'] = df['MAINHEAT_DESCRIPTION'].str.extract('(?<=,\s)([^,]+)')
    df['main_heat_system_2'] =
df['main_heat_system_2'].fillna(df['main_heat_system_2'].value_counts().index[0])
    df['MHS_mains gas'] = df['main_heat_system_2'].str.contains('mains gas')
    df['MHS_None'] = df['main_heat_system_2'].str.contains('None')
    df['MHS_electric'] = df['main_heat_system_2'].str.contains('electric')
    df['MHS_oil'] = df['main_heat_system_2'].str.contains('oil ')
    df['MHS_radiators'] = df['main_heat_system_2'].str.contains('radiators')
    df['MHS_underfloor heating '] = df['main_heat_system_2'].str.contains('underfloor')
    df['MHS_underfloor heating '] = df['main_heat_system_2'].str.contains('underfloor heating')
    df['MHS_lpg'] = df['main_heat_system_2'].str.contains('lpg')
    df['MHS_lpg'] = df['main_heat_system_2'].str.contains('bottled lpg')
    df['MHS_dual fuel mineral and wood '] = df['main_heat_system_2'].str.contains('dual fuel mineral
and wood ')
```

```
    df['MHS_warm air'] = df['main_heat_system_2'].str.contains('warm air')
    df['MHS_wood pellets_logs'] = df['main_heat_system_2'].str.contains('wood pellets')
    df['MHS_wood pellets_logs'] = df['main_heat_system_2'].str.contains('wood logs')
    df['other'] = df['main_heat_system_2'].str.contains('wood chips')
    df['MHS_coal'] = df['main_heat_system_2'].str.contains('coal')
    df['MHS_other'] = df['main_heat_system_2'].str.contains('electricaire')
    df['MHS_other'] = df['main_heat_system_2'].str.contains('anthracite')
    df['MHS_other'] = df['main_heat_system_2'].str.contains('smokeless fuel')
    df['MHS_other'] = df['main_heat_system_2'].str.contains('fan coil units')
    df['MHS_other'] = df['main_heat_system_2'].str.contains('b30k')
    df['MHS_other'] = df['main_heat_system_2'].str.contains('lng')
        return df
epc_49 = mainheat_sys(epc_49)


# In[38]:

epc_49['main_heat_system_2'].value_counts()


# In[71]:

epc_49= epc_49.drop(['main_heat_system', 'main_heat_system_2',
'wall_insulation','roof_insulation','floor_insulation'],axis=1 )


# In[72]:

def main_heat_control(df):
    df['MAINHEATCONT_DESCRIPTION'] =
df['MAINHEATCONT_DESCRIPTION'].str.replace("2207","")
    df['MAINHEATCONT_DESCRIPTION'] =
df['MAINHEATCONT_DESCRIPTION'].str.replace("room thermostats","room thermostat")
    df['MAINHEATCONT_DESCRIPTION'] =
df['MAINHEATCONT_DESCRIPTION'].str.replace("programmer and at least two room
stats","room thermostat")
    df['MAINHEATCONT_DESCRIPTION'] =
df['MAINHEATCONT_DESCRIPTION'].str.replace("room thermostat","room thermostat(s)")
    df['mainheat_cont_system'] = df['MAINHEATCONT_DESCRIPTION'].str.replace("^\s+","")
```

```
#     df['mainheat_cont_system'] = df['MAINHEATCONT_DESCRIPTION'].str.extract('^([^,]+)')

#     df['mainheat_cont_system'] = df['mainheat_cont_system'].str.replace("^[ \t]+","")

    df['mainheat_cont_system'] =
df['MAINHEATCONT_DESCRIPTION'].fillna(df['mainheat_cont_system'].value_counts().index[0])

    df['MHCS_programmer'] = df['mainheat_cont_system'].str.contains('programmer')

    df['MHCS_time and temperature zone control'] = df['mainheat_cont_system'].str.contains('time and
temperature zone control')

    df['MHCS_manual charge control'] = df['mainheat_cont_system'].str.contains('manual charge
control')

    df['MHCS_room thermostat(s)'] = df['mainheat_cont_system'].str.contains('room thermostat(s)')

    df['MHCS_room thermostat(s)'] = df['mainheat_cont_system'].str.contains('at least two room
thermostat(s)')

    df['MHCS_appliance thermostats'] = df['mainheat_cont_system'].str.contains('appliance
thermostats')

    df['MHCS_controls for high heat retention storage heaters'] =
df['mainheat_cont_system'].str.contains('controls for high heat retention storage heaters')

    df['MHCS_automatic charge control'] = df['mainheat_cont_system'].str.contains('automatic charge
control')

    df['MHCS_charging system linked to use of community heating'] =
df['mainheat_cont_system'].str.contains('charging system linked to use of community heating')

    df['MHCS_flat rate charging'] = df['mainheat_cont_system'].str.contains('flat rate charging')

    df['MHCS_trvs'] = df['mainheat_cont_system'].str.contains('trvs')

    df['MHCS_bypass'] = df['mainheat_cont_system'].str.contains('bypass')

    df['MHCS_none'] = df['mainheat_cont_system'].str.contains('none')

    df['MHCS_no time or thermostatic control of room temperature'] =
df['mainheat_cont_system'].str.contains('no time or thermostatic control of room temperature')

    df['MHCS_other'] = df['mainheat_cont_system'].str.contains('temperature zone control')

    df['MHCS_other'] = df['mainheat_cont_system'].str.contains('celect control')

    df['MHCS_boiler energy manager'] = df['mainheat_cont_system'].str.contains('boiler energy
manager')

    df['MHCS_no time or thermostatic control of room temperature'] =
df['mainheat_cont_system'].str.contains('no thermostatic control of room temperature')

    return df

epc_49 = main_heat_control(epc_49)


# In[73]:
```

```
epc_49= epc_49.drop(['mainheat_cont_system'],axis=1 )


# In[74]:

def hotwater_sys(df):
    df['hotwater_system'] = df['HOTWATER_DESCRIPTION']
    df['hotwater_system'] = df['hotwater_system'].fillna(df['hotwater_system'].value_counts().index[0])
    df['hws_from main system'] = df['hotwater_system'].str.contains('from main system')
    df['hws_from secondary system'] = df['hotwater_system'].str.contains('from secondary system')
    df['hws_community scheme'] = df['hotwater_system'].str.contains('community scheme')
    df['hws_electric immersion'] = df['hotwater_system'].str.contains('electric immersion')
    df['hws_electric instantaneous at point of use'] = df['hotwater_system'].str.contains('electric
instantaneous at point of use')
    df['hws_heat pump'] = df['hotwater_system'].str.contains('electric heat pump for water heating
only')
    df['hws_heat pump'] = df['hotwater_system'].str.contains('electric heat pump')
    df['hws_gas multipoint'] = df['hotwater_system'].str.contains('gas multipoint')
    df['hws_gas instantaneous at point of use '] = df['hotwater_system'].str.contains('gas instantaneous
at point of use')
    df['hws_other'] = df['hotwater_system'].str.contains('solid fuel boiler/circulator')
    df['hws_other'] = df['hotwater_system'].str.contains('oil boiler/circulator')
    df['hws_other'] = df['hotwater_system'].str.contains('gas boiler/circulator')
    df['hws_other'] = df['hotwater_system'].str.contains('oil range cooker')
    df['hws_other'] = df['hotwater_system'].str.contains('solid fuel range cooker')
    df['hws_other'] = df['hotwater_system'].str.contains('gas range cooker')
    #2nd part
    df['hws_None'] = df['hotwater_system'].str.contains('None')
    df['hws_off-peak'] = df['hotwater_system'].str.contains('off-peak')
    df['hws_no cylinder thermostat'] = df['hotwater_system'].str.contains('no cylinder thermostat')
    df['hws_standard tariff'] = df['hotwater_system'].str.contains('standard tariff')
    df['hws_flue gas heat recovery'] = df['hotwater_system'].str.contains('flue gas heat recovery')
    df['hws_plus solar'] = df['hotwater_system'].str.contains('plus solar')
    df['hws_waste water heat recovery'] = df['hotwater_system'].str.contains('waste water heat
recovery')
    return df
```

```
epc_49 = hotwater_sys(epc_49)


# In[75]:

epc_49= epc_49.drop(['hotwater_system'],axis=1 )


# In[76]:

def lighting_perc(df):
    df['low_energy_lighting_perc'] = df['LIGHTING_DESCRIPTION'].str.findall(r'(\d*)\%')
    df['low_energy_lighting_perc'] = round(df['low_energy_lighting_perc'].str[0].astype(float),-1)
    df['low_energy_lighting_perc'] =
df['low_energy_lighting_perc'].fillna(df['low_energy_lighting_perc'].value_counts().index[0])
    df['low_lighting'] = df.apply(lambda row: 'low energy lighting %d%% of fixed outlets' %
(int(row['low_energy_lighting_perc'])) if '%' in str(row['LIGHTING_DESCRIPTION']) else
row['LIGHTING_DESCRIPTION'],axis=1)
    df['low_lighting'] = df['low_lighting'].str.replace("low energy lighting in all fixed outlets","low
energy lighting 100% of fixed outlets")
    df['low_lighting'] = df['low_lighting'].str.replace("no low energy lighting","low energy lighting 0%
of fixed outlets")
    return df
epc_49 = lighting_perc(epc_49)


# In[77]:

epc_49 = epc_49.drop(['low_energy_lighting_perc'], axis=1)


# In[78]:

def secondheat(df):
    df['second_heat'] = df['SECONDHEAT_DESCRIPTION']
    df['second_heat'] =
df['SECONDHEAT_DESCRIPTION'].fillna(df['second_heat'].value_counts().index[0])
    df['SH_room heaters'] = df['second_heat'].str.contains('room heaters')
    df['SH_none'] = df['second_heat'].str.contains('none')
    df['SH_electric'] = df['second_heat'].str.contains('electric')
    df['SH_mains gas'] = df['second_heat'].str.contains('mains gas')
    df['SH_portable electric heaters'] = df['second_heat'].str.contains('portable electric heaters')
```

```
    df['SH_dual fuel mineral and wood'] = df['second_heat'].str.contains('dual fuel mineral and wood')
    df['SH_wood logs'] = df['second_heat'].str.contains('wood logs')
    df['SH_coal'] = df['second_heat'].str.contains('coal')
    df['SH_lpg'] = df['second_heat'].str.contains('lpg')
    df['SH_wood pellets'] = df['second_heat'].str.contains('wood pellets')
    df['SH_smokeless fuel'] = df['second_heat'].str.contains('smokeless fuel')
    df['SH_anthracite'] = df['second_heat'].str.contains('anthracite')
    df['SH_bioethanol'] = df['second_heat'].str.contains('bioethanol')
    df['SH_oil'] = df['second_heat'].str.contains('oil')
    return df
epc_49 = secondheat(epc_49)


# In[79]:
epc_49= epc_49.drop(['second_heat'],axis=1 )


# In[80]:
#combine cate variables
built_dict = dict.fromkeys(['Mid-Terrace','End-Terrace','Enclosed Mid-Terrace','Enclosed End-
Terrace'],'terraced')
built_dict1 = dict.fromkeys(['Semi-Detached','Detached'],'detached')
built_dict.update(built_dict1)


# In[81]:
epc_49['built_form'] =epc_49['Built Form'].replace(built_dict)



# In[82]:
epc_49= epc_49.drop(['Built Form'], axis=1)


# In[83]:
#dropping description attributes
epc_49 = epc_49.drop(['WALL_DESCRIPTION', 'ROOF_DESCRIPTION',
'FLOOR_DESCRIPTION',
```

```
                'WINDOWS_DESCRIPTION',
'MAINHEAT_DESCRIPTION','MAINHEATCONT_DESCRIPTION',
                'SECONDHEAT_DESCRIPTION', 'HOTWATER_DESCRIPTION',
'LIGHTING_DESCRIPTION'] , axis=1)


# In[84]:

epc_49.shape


# In[85]:

epc_49.dtypes


# In[86]:

epc_49['insulated_wall'] = epc_49['insulated_wall'].astype(int)

epc_49['insulated_floor'] = epc_49['insulated_floor'].astype(int)

epc_49['insulated_roof'] = epc_49['insulated_roof'].astype(int)


# In[87]:

epc_49.head()


# In[88]:

objList = epc_49.select_dtypes(include = "object").columns

# Label Encoding for object to numeric conversion

epc_encoded = epc_49.copy()

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

for feat in objList:

    epc_encoded[feat] = le.fit_transform(epc_49[feat].astype(str))

# epc_encoded.info()



# In[89]:

epc_49.to_csv('epc_encoded.csv')
```

Code of feature selection:

```
# In[2]:
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from scipy import stats
from sklearn.model_selection import train_test_split
import warnings
import matplotlib.pyplot as plt
import datetime

# In[3]:
pd.set_option('display.max_rows', 99999)
pd.set_option('display.max_columns', 500)
warnings.filterwarnings("ignore")

# In[4]:
epc_49 = pd.read_csv('epc_encoded.csv')
##epc_encoded from previous step(preprocessing)
epc_49.shape

# In[5]:
epc_49.head(1)

# In[6]:
df_x = epc_49.drop(['Current energy efficiency rating band'], axis = 1)
def drop_correlated_features(dataframe, threshold):
    '''remove correlated features'''
    corr_matrix = dataframe.corr().abs()
    upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))
    to_drop = [column for column in upper.columns if any(upper[column] > threshold)]
    dataframe = dataframe.drop(to_drop, axis=1)
```

```
    print('Removed Columns {}'.format(to_drop))
    return dataframe

epc = drop_correlated_features(df_x,0.6)
epc['Current energy efficiency rating band']= epc_49['Current energy efficiency rating band']

# In[7]:
epc.head(2)

# In[8]:
epc['Current energy efficiency rating band'].value_counts()

# In[9]:
epc.shape

# In[10]:
# # Dataset sampling for faster computation
# epc = epc.groupby('Current energy efficiency rating band', group_keys=False).apply(lambda x:
x.sample(frac=0.6))

# In[11]:
# epc['Current energy efficiency rating band'].value_counts()

# In[12]:
epc = epc.drop('Unnamed: 0', axis=1)

# In[13]:
y = epc['Current energy efficiency rating band']
X = epc.drop(['Current energy efficiency rating band'], axis = 1)
X_train, X_test, y_train, y_test = train_test_split(X, y,random_state=40,  test_size=0.3)

# In[14]:
# #==========Mutual info=======#
```

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import mutual_info_classif
#No we Will select the top 20 important features
sel_cols = SelectKBest(mutual_info_classif, k=20)
sel_cols.fit(X, y)
X.columns[sel_cols.get_support()]


# In[ ]:
epc_df =epc[['Total floor area (m²)', 'Total current energy costs over 3 years (£)',
       'Current hot water costs over 3 years (£)',
       'Part 1 Construction Age Band', 'Part 1 Floor 0 Room Height',
       'Low Energy Lighting %', 'Mechanical Ventilation', 'Tenure',
       'Transaction Type', 'Property Type', 'insulated_wall', 'insulated_roof',
       'wall_type', 'roof_type', 'floor_type', 'windows_glazing',
       'MMH_mains gas', 'MHCS_programmer', 'low_lighting', 'SH_room heaters', 'Current energy efficiency rating band']]


# In[ ]:
def make_mi_scores(X, y):
    mi_scores = mutual_info_classif(X, y)
    mi_scores = pd.Series(mi_scores, name="MI Scores", index=X.columns)
    mi_scores = mi_scores.sort_values(ascending=False)
    return mi_scores
mi_scores = make_mi_scores(X, y)
mi_scores[::3]  # show a few features with their MI scores


# In[102]:
def plot_mi_scores(scores):
    scores = scores.sort_values(ascending=True)
    width = np.arange(len(scores))
    ticks = list(scores.index)
    plt.barh(width, scores)
    plt.yticks(width, ticks)
```

```
    plt.title("Mutual Information Scores")
plt.figure(dpi=100, figsize=(8, 14))
plot_mi_scores(mi_scores)

# In[106]:
epc_df.to_csv('epc_best20.csv')

# In[104]:
epc_best20.head(10)

# In[105]:
epc_best20.shape
```

Code of Machine Learning Models:

```
# In[1]:
import numpy as np
import pandas as pd
import warnings
import matplotlib.pyplot as plt
import datetime

# In[2]:
pd.set_option('display.max_rows', 99999)
pd.set_option('display.max_columns', 500)
warnings.filterwarnings("ignore")

# In[3]:
df = pd.read_csv('epc_best20.csv')
df.shape

# In[4]:
df.head()
```

```
# In[5]:
objList = df.select_dtypes(include = "object").columns
# Label Encoding for object to numeric conversion
epc_encoded = df.copy()
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for feat in objList:
    epc_encoded[feat] = le.fit_transform(df[feat].astype(str))
epc_encoded.info()

# In[6]:
epc_encoded.head()

# In[7]:
from sklearn.preprocessing import LabelEncoder
# Split data into features and labels
X = epc_encoded.drop("Current energy efficiency rating band", axis=1)
y = epc_encoded["Current energy efficiency rating band"]

# In[8]:
from sklearn.model_selection import train_test_split
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# In[9]:
def plot_conf_mat(y_test, y_pred):
    '''prints the confusion matrix'''
    from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test, y_pred)
    fig, ax = plt.subplots(figsize=(5,5))
    ax.matshow(cm, cmap=plt.cm.Blues, alpha=0.8)
```

```
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            ax.text(x=j, y=i,s=cm[i, j], va='center', ha='center', size='small')
    plt.xlabel('Predictions', fontsize=10)
    plt.ylabel('Actuals', fontsize=10)
    plt.title('Confusion Matrix', fontsize=10)
    plt.show()
    return

# In[10]:
#libraries we need
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import AdaBoostClassifier, ExtraTreesClassifier
from sklearn.metrics import mean_squared_error
import time
from sklearn.metrics import classification_report
from sklearn.ensemble import GradientBoostingClassifier
import sklearn.metrics as metrics
# In[11]:
tree= DecisionTreeClassifier()
forest= RandomForestClassifier()
extra_tree= ExtraTreesClassifier()
knn= KNeighborsClassifier()
xboost= XGBClassifier()
abr = AdaBoostClassifier()
svc=SVC(C=100.0)
gbc = GradientBoostingClassifier()
```

```
lg= LogisticRegression()
models= [
#     extra_tree,
        tree, forest,xboost, svc ]
for model in models:
    start = time.time() #start time
    model.fit(X_train, y_train) # fit the model
    y_pred= model.predict(X_test) #  predict on the test set
    end = time.time() # end time
    #========metrics======#
    print(model)
    clf_report= classification_report(y_test, y_pred)  #with precision, recall and f1-score for each class
    print(clf_report)
    rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
    r2 = metrics.r2_score(y_test, y_pred)
    print(f"\nr2: {rmse}") ##print mean square error and r2
    print(f"\nRMSE: {rmse}")

    print(f"\nRuntime of the model is {end - start}") ##run time
    plot_conf_mat(y_test, y_pred)
    print("\n")
```

Code of Optimizing Machine Learning Models:

```
# In[1]:
import numpy as np
import pandas as pd
import warnings
import matplotlib.pyplot as plt
import datetime

# In[2]:
pd.set_option('display.max_rows', 99999)
```

```
pd.set_option('display.max_columns', 500)
warnings.filterwarnings("ignore")

# In[3]:
df = pd.read_csv('epc_best20.csv')
df.shape

# In[5]:
df.head()

# In[6]:
objList = df.select_dtypes(include = "object").columns
# Label Encoding for object to numeric conversion
epc_encoded = df.copy()
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for feat in objList:
    epc_encoded[feat] = le.fit_transform(df[feat].astype(str))
epc_encoded.info()

# In[7]:
epc_encoded.head()

# In[8]:
from sklearn.preprocessing import LabelEncoder
# Split data into features and labels
X = epc_encoded.drop("Current energy efficiency rating band", axis=1)
y = epc_encoded["Current energy efficiency rating band"]

# In[9]:
from sklearn.model_selection import train_test_split
# Split data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)


# In[10]:
def plot_conf_mat(y_test, y_pred):
    '''prints the confusion matrix'''
    from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test, y_pred)
    fig, ax = plt.subplots(figsize=(5,5))
    ax.matshow(cm, cmap=plt.cm.Blues, alpha=0.8)
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            ax.text(x=j, y=i,s=cm[i, j], va='center', ha='center', size='small')
    plt.xlabel('Predictions', fontsize=10)
    plt.ylabel('Actuals', fontsize=10)
    plt.title('Confusion Matrix', fontsize=10)
    plt.show()
    return


# # Hyperparameter using random search
# # Hyperparameter svm
# In[10]:
from sklearn.svm import SVC


# In[11]:
param_grid = {'C': [0.1, 1, 10, 100],
              'gamma': [1, 0.1, 0.01, 0.001],
              'kernel': ['rbf']}
SVM_Model = SVC(gamma='auto')
from sklearn.model_selection import RandomizedSearchCV
random_Grid = RandomizedSearchCV (estimator = SVM_Model, param_distributions = param_grid,
cv = 3, verbose=2, n_jobs = -1)


# In[12]:
```

```
random_Grid.fit(X,y)


# In[13]:
random_Grid.best_params_


# # Hyperparameter XGB
# In[12]:
from xgboost import XGBClassifier
from sklearn.model_selection import RandomizedSearchCV


# In[13]:
# Define the search space
param_grid1 = {
    # Learning rate shrinks the weights to make the boosting process more conservative
    "learning_rate": [.2,.3, 0.01, 0.1, 1] ,
    # Maximum depth of the tree, increasing it increases the model complexity.
    "max_depth": [int(x) for x in np.linspace(10, 120, num = 12)],
    # Gamma specifies the minimum loss reduction required to make a split.
    "gamma": [0.5, 1, 1.5, 2, 5],
    # Percentage of columns to be randomly samples for each tree.
    "colsample_bytree": [0.6, 0.8, 1.0],
    # Number of trees
    "n_estimators": [600, 800, 1000, 1200, 1400, 1600, 1800, 2000]
}
xgboost = XGBClassifier()
# Random search of parameters, using 3 fold cross validation,
# search across 30 different combinations, and use all available cores
random_search = RandomizedSearchCV(estimator=xgboost, param_distributions=param_grid1,
n_iter = 30, cv = 3, verbose=2, random_state=42, n_jobs = -1)


# In[14]:
random_search.fit(X_train, y_train)
```

```
# In[15]:

print(random_search.best_params_)


# In[ ]:

n_estimators=1400,max_depth=30,learning_rate=0.01,gamma= 1, colsample_bytree=0.8


# # Hyperparameter RF


# In[24]:

n_estimators = [600, 800, 1000, 1200, 1400, 1600, 1800, 2000] # number of trees in the random forest

max_features = ['auto', 'sqrt'] # number of features in consideration at every split

max_depth = [int(x) for x in np.linspace(10, 120, num = 12)] # maximum number of levels allowed in each decision tree

min_samples_split = [2, 6, 10] # minimum sample number to split a node

min_samples_leaf = [1, 3, 4] # minimum sample number that can be stored in a leaf node

bootstrap = [True, False] # method used to sample data points

random_grid = {'n_estimators': n_estimators,

'max_features': max_features,

'max_depth': max_depth,

'min_samples_split': min_samples_split,

'min_samples_leaf': min_samples_leaf,

'bootstrap': bootstrap}

rf = RandomForestClassifier()

from sklearn.model_selection import RandomizedSearchCV

rf_random = RandomizedSearchCV(estimator = rf,param_distributions = random_grid,n_iter = 30, cv = 3, verbose=2, random_state=35, n_jobs = -1)


# In[25]:

rf_random.fit(X_train, y_train)


# In[27]:

# print the best parameters

print ('Best Parameters: ', rf_random.best_params_, ' \n')
```

```
# In[ ]:

n_estimators=1600,min_samples_split=6,min_samples_leaf=
1,max_features='auto',max_depth=80,bootstrap=False


# # model after optimize

# In[16]:

#libraries we need

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from xgboost import XGBClassifier

from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import AdaBoostClassifier, ExtraTreesClassifier

from sklearn.metrics import mean_squared_error

import time

from sklearn.metrics import classification_report

from sklearn.ensemble import GradientBoostingClassifier

import sklearn.metrics as metrics


# In[18]:

tree= DecisionTreeClassifier()

forest= RandomForestClassifier(n_estimators=1600,min_samples_split=6,min_samples_leaf=
1,max_features='auto',max_depth=80,bootstrap=False)

xboost= XGBClassifier(n_estimators=1400,max_depth=30,learning_rate=0.01,gamma= 1,
colsample_bytree=0.8)

svc=SVC(C=0.1,gamma=0.01,kernel='rbf',)

models= [

#     extra_tree,

        forest, tree,svc,xboost ]

for model in models:

   start = time.time() #start time
```

```
    model.fit(X_train, y_train) # fit the model
    y_pred= model.predict(X_test) #  predict on the test set
    end = time.time() # end time
    #========metrics======#
    print(model)
    clf_report= classification_report(y_test, y_pred)  #with precision, recall and f1-score for each class
    print(clf_report)
    rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
    r2 = metrics.r2_score(y_test, y_pred)
    print(f"\nr2: {rmse}") ##print mean square error and r2
    print(f"\nRMSE: {rmse}")
    print(f"\nRuntime of the model is {end - start}") ##run time
    plot_conf_mat(y_test, y_pred)
    print("\n")
```

Code of Deep Learning Default Model:

```
# In[1]:
import sklearn.metrics as metrics
import pandas as pd
from sklearn.model_selection import train_test_split
from keras.preprocessing.text import Tokenizer
from sklearn.preprocessing import LabelEncoder
from keras.models import Sequential
from keras.layers import Dense,Dropout
from keras.optimizers import Adam
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from keras.utils import to_categorical
from sklearn.compose import ColumnTransformer
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
# In[2]:
df = pd.read_csv("epc_best20.csv")
# In[3]:
```

```
# Split data into features and labels
X = df.drop("Current energy efficiency rating band", axis=1)
y = df["Current energy efficiency rating band"]
encoder = LabelEncoder()
y= encoder.fit_transform(y)
# One-hot encode labels
y = to_categorical(y, num_classes=7)


# In[4]:
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)


# In[5]:
# Define which columns are categorical and which are numerical
categorical_cols = ["Part 1 Construction Age Band", "Mechanical Ventilation", "Tenure",
"Transaction Type",
                    "wall_insulation", "roof_insulation", "wall_type", "roof_type", "floor_type",
                    "windows_glazing", "low_lighting"]
numerical_cols = ["Total floor area (m²)", "Total current energy costs over 3 years (£)",
                  "Current hot water costs over 3 years (£)", "Part 1 Floor 0 Room Height",
                  "Low Energy Lighting %", "insulated_wall", "MMH_mains gas", "MHCS_programmer",
                  "SH_room heaters"]


# In[6]:
# Create ColumnTransformer to handle different data types
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_cols),
        ('cat', OneHotEncoder(), categorical_cols)])



# In[7]:
# Fit and transform the data using the ColumnTransformer
```

```
X_train = preprocessor.fit_transform(X_train)
X_test = preprocessor.transform(X_test)

# In[8]:
model = Sequential()
model.add(Dense(64, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dropout(0.5))  # Adding dropout to prevent overfitting
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))  # Adding dropout to prevent overfitting
model.add(Dense(7, activation='sigmoid'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# In[9]:
history = model.fit(X_train, y_train,validation_data = (X_test,y_test), epochs=20, batch_size=32)

# In[10]:
# Evaluate model
score = model.evaluate(X_test, y_test)
print("Test loss: ", score[0])
print("Test accuracy: ", score[1])

# In[11]:
import matplotlib.pyplot as plt
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper right')
plt.show()
```

```
# In[12]:
# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

# In[13]:
from sklearn.metrics import roc_auc_score, roc_curve
import matplotlib.pyplot as plt
# Compute the predicted probabilities for each class
y_pred_proba = model.predict(X_test)
# Compute the average AUC score across all classes
auc = roc_auc_score(y_test, y_pred_proba, multi_class='ovr')
print('Average AUC:', auc)
# Compute the FPR and TPR for each class
fpr = dict()
tpr = dict()
for i in range(7):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_pred_proba[:, i])
# Plot the ROC curve for each class
plt.figure()
for i in range(7):
    plt.plot(fpr[i], tpr[i], label='ROC curve of class {}'.format(i))
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

```
plt.title('ROC curves for multiclass output')
plt.legend(loc="lower right")
plt.show()


# In[14]:
# Make predictions on the test set
y_pred_proba = model.predict(X_test)
y_pred = y_pred_proba.argmax(axis=1)
#convert one hot encoded y_test to numerical format
y_test = y_test.argmax(axis=1)
# Evaluate model
print("Accuracy: ", accuracy_score(y_test, y_pred))
print("Precision: ", precision_score(y_test, y_pred, average='weighted'))
print("Recall: ", recall_score(y_test, y_pred, average='weighted'))
print("F1 Score: ", f1_score(y_test, y_pred, average='weighted'))

# In[15]:
import numpy as np
rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
r2 = metrics.r2_score(y_test, y_pred)
print(f"\nr2: {rmse}") ##print mean square error and r2
print(f"\nRMSE: {rmse}")

# In[16]:
def plot_conf_mat(y_test, y_pred):
    '''prints the confusion matrix'''
    from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test, y_pred)
    fig, ax = plt.subplots(figsize=(5,5))
    ax.matshow(cm, cmap=plt.cm.Blues, alpha=0.8)
    for i in range(cm.shape[0]):
```

```
        for j in range(cm.shape[1]):
            ax.text(x=j, y=i,s=cm[i, j], va='center', ha='center', size='small')
    plt.xlabel('Predictions', fontsize=10)
    plt.ylabel('Actuals', fontsize=10)
    plt.title('Confusion Matrix', fontsize=10)
    plt.show()
    return
# In[17]:
plot_conf_mat(y_test, y_pred)
```

Code of Deep Learning Model A:

```
# In[1]:
import sklearn.metrics as metrics
import pandas as pd
from sklearn.model_selection import train_test_split
from keras.preprocessing.text import Tokenizer
from sklearn.preprocessing import LabelEncoder
from keras.models import Sequential
from keras.layers import Dense,Dropout
from keras.optimizers import Adam
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from keras.utils import to_categorical
from sklearn.compose import ColumnTransformer
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# In[2]:
df = pd.read_csv("epc_best20.csv")

# In[3]:
# Split data into features and labels
X = df.drop("Current energy efficiency rating band", axis=1)
y = df["Current energy efficiency rating band"]
```

```
encoder = LabelEncoder()
y= encoder.fit_transform(y)
# One-hot encode labels
y = to_categorical(y, num_classes=7)


# In[4]:
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)


# In[5]:
# Define which columns are categorical and which are numerical
categorical_cols = ["Part 1 Construction Age Band", "Mechanical Ventilation", "Tenure",
"Transaction Type",
                    "wall_insulation", "roof_insulation", "wall_type", "roof_type", "floor_type",
                    "windows_glazing", "low_lighting"]
numerical_cols = ["Total floor area (m²)", "Total current energy costs over 3 years (£)",
                  "Current hot water costs over 3 years (£)", "Part 1 Floor 0 Room Height",
                  "Low Energy Lighting %", "insulated_wall", "MMH_mains gas", "MHCS_programmer",
                  "SH_room heaters"]


# In[6]:
# Create ColumnTransformer to handle different data types
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_cols),
        ('cat', OneHotEncoder(), categorical_cols)])


# In[7]:
# Fit and transform the data using the ColumnTransformer
X_train = preprocessor.fit_transform(X_train)
X_test = preprocessor.transform(X_test)


# In[9]:
```

```
model = Sequential()
model.add(Dense(128, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dropout(0.5))  # Adding dropout to prevent overfitting
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))  # Adding dropout to prevent overfitting
model.add(Dense(7, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# In[10]:
history = model.fit(X_train, y_train,validation_data = (X_test,y_test), epochs=100, batch_size=32)



# In[11]:
# Evaluate model
score = model.evaluate(X_test, y_test)
print("Test loss: ", score[0])
print("Test accuracy: ", score[1])



# In[12]:
import matplotlib.pyplot as plt
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper right')
plt.show()



# In[13]:
# Plot training & validation accuracy values
```

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()



# In[14]:
from sklearn.metrics import roc_auc_score, roc_curve
import matplotlib.pyplot as plt

# Compute the predicted probabilities for each class
y_pred_proba = model.predict(X_test)

# Compute the average AUC score across all classes
auc = roc_auc_score(y_test, y_pred_proba, multi_class='ovr')
print('Average AUC:', auc)

# Compute the FPR and TPR for each class
fpr = dict()
tpr = dict()
for i in range(7):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_pred_proba[:, i])

# Plot the ROC curve for each class
plt.figure()
for i in range(7):
    plt.plot(fpr[i], tpr[i], label='ROC curve of class {}'.format(i))
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
```

```
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC curves for multiclass output')
plt.legend(loc="lower right")
plt.show()

# In[15]:
# Make predictions on the test set
y_pred_proba = model.predict(X_test)
y_pred = y_pred_proba.argmax(axis=1)

#convert one hot encoded y_test to numerical format
y_test = y_test.argmax(axis=1)

# Evaluate model
print("Accuracy: ", accuracy_score(y_test, y_pred))
print("Precision: ", precision_score(y_test, y_pred, average='weighted'))
print("Recall: ", recall_score(y_test, y_pred, average='weighted'))
print("F1 Score: ", f1_score(y_test, y_pred, average='weighted'))

# In[19]:
import numpy as np
rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
r2 = metrics.r2_score(y_test, y_pred)
print(f"\nr2: {rmse}") ##print mean square error and r2
print(f"\nRMSE: {rmse}")

# In[20]:
def plot_conf_mat(y_test, y_pred):
    '''prints the confusion matrix'''
    from sklearn.metrics import confusion_matrix
```

```
    cm = confusion_matrix(y_test, y_pred)

    fig, ax = plt.subplots(figsize=(5,5))
    ax.matshow(cm, cmap=plt.cm.Blues, alpha=0.8)
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            ax.text(x=j, y=i,s=cm[i, j], va='center', ha='center', size='small')
    plt.xlabel('Predictions', fontsize=10)
    plt.ylabel('Actuals', fontsize=10)
    plt.title('Confusion Matrix', fontsize=10)
    plt.show()
    return

# In[21]:
plot_conf_mat(y_test, y_pred)
```

Code of Optimizing Model B:

```
# In[17]:
import pandas as pd
from sklearn.model_selection import train_test_split
from keras.preprocessing.text import Tokenizer
from sklearn.preprocessing import LabelEncoder
from keras.models import Sequential
from keras.layers import Dense,Dropout
from keras.optimizers import Adam
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from keras.utils import to_categorical
from sklearn.compose import ColumnTransformer
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import tensorflow as tf
from sklearn.model_selection import GridSearchCV
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Dense
from scikeras.wrappers import KerasClassifier

# In[18]:
df = pd.read_csv("epc_best20.csv")

# In[19]:
df.head()
# In[20]:
df.tail()

# In[21]:
# Define which columns are categorical and which are numerical
categorical_cols = ["Part 1 Construction Age Band", "Mechanical Ventilation", "Tenure",
"Transaction Type",
                    "wall_insulation", "roof_insulation", "wall_type", "roof_type", "floor_type",
                    "windows_glazing", "low_lighting"]
numerical_cols = ["Total floor area (m²)", "Total current energy costs over 3 years (£)",
                  "Current hot water costs over 3 years (£)", "Part 1 Floor 0 Room Height",
                  "Low Energy Lighting %", "insulated_wall", "MMH_mains gas", "MHCS_programmer",
                  "SH_room heaters"]

# In[22]:
# Create ColumnTransformer to handle different data types
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_cols),
        ('cat', OneHotEncoder(), categorical_cols)])

# In[23]:
# Split data into features and labels
X = df.drop("Current energy efficiency rating band", axis=1)
y = df["Current energy efficiency rating band"]
```

```
encoder = LabelEncoder()
y= encoder.fit_transform(y)

# One-hot encode labels
y = to_categorical(y, num_classes=7)

# In[24]:
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# In[25]:
# Fit and transform the data using the ColumnTransformer
X_train = preprocessor.fit_transform(X_train)
X_test = preprocessor.transform(X_test)

# In[26]:
# fix random seed for reproducibility
seed = 7
tf.random.set_seed(seed)

# In[27]:
# Function to create model, required for KerasClassifier
def create_model():
    model = Sequential()
    model.add(Dense(64, activation='relu', input_shape=(X_train.shape[1],)))
    model.add(Dropout(0.5))  # Adding dropout to prevent overfitting
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.5))  # Adding dropout to prevent overfitting
    model.add(Dense(7, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
```

```
# In[28]:
model = KerasClassifier(model=create_model, verbose=0)
# # Tune Batch Size and Number of Epochs

# In[13]:
# define the grid search parameters
batch_size = [10, 20, 40, 60, 80, 100]
epochs = [20, 50, 100]
param_grid = dict(batch_size=batch_size, epochs=epochs)
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=3)
grid_result = grid.fit(X_train, y_train)

# In[14]:
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))

# In[29]:
# define the grid search parameters
optimizer = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam']
param_grid = dict(optimizer=optimizer)
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=3)
grid_result = grid.fit(X_train, y_train)

# In[30]:
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
```

```
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

Code of deep learning Model B:

```
# In[1]:
import sklearn.metrics as metrics
import pandas as pd
from sklearn.model_selection import train_test_split
from keras.preprocessing.text import Tokenizer
from sklearn.preprocessing import LabelEncoder
from keras.models import Sequential
from keras.layers import Dense,Dropout
from keras.optimizers import Adam ,SGD ,Adagrad
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from keras.utils import to_categorical
from sklearn.compose import ColumnTransformer
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# In[2]:
df = pd.read_csv("epc_best20.csv")

# In[17]:
df.shape

# In[9]:



# Split data into features and labels
X = df.drop("Current energy efficiency rating band", axis=1)
```

```
y = df["Current energy efficiency rating band"]
encoder = LabelEncoder()
y= encoder.fit_transform(y)
# One-hot encode labels
y = to_categorical(y, num_classes=7)



# In[10]:
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)



# In[14]:
df["Current energy efficiency rating band"].value_counts()



# In[17]:
# Define which columns are categorical and which are numerical
categorical_cols = ["Part 1 Construction Age Band", "Mechanical Ventilation", "Tenure",
"Transaction Type",
                    "wall_insulation", "roof_insulation", "wall_type", "roof_type", "floor_type",
                    "windows_glazing", "low_lighting"]
numerical_cols = ["Total floor area (m²)", "Total current energy costs over 3 years (£)",
                  "Current hot water costs over 3 years (£)", "Part 1 Floor 0 Room Height",
                  "Low Energy Lighting %", "insulated_wall", "MMH_mains gas", "MHCS_programmer",
                  "SH_room heaters"]

# In[18]:
# Create ColumnTransformer to handle different data types
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_cols),
        ('cat', OneHotEncoder(), categorical_cols)])
```

```
# In[19]:
# Fit and transform the data using the ColumnTransformer
X_train = preprocessor.fit_transform(X_train)
X_test = preprocessor.transform(X_test)

# In[20]:
model = Sequential()
model.add(Dense(64, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dropout(0.5))  # Adding dropout to prevent overfitting
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))  # Adding dropout to prevent overfitting
model.add(Dense(7, activation='sigmoid'))

model.compile(loss='categorical_crossentropy', optimizer='Adagrad', metrics=['accuracy'])

# In[21]:
history = model.fit(X_train, y_train,validation_data = (X_test,y_test), epochs=100, batch_size=80)

# In[22]:
# Evaluate model
score = model.evaluate(X_test, y_test)
print("Test loss: ", score[0])
print("Test accuracy: ", score[1])

# In[23]:
import matplotlib.pyplot as plt
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
```

```
plt.legend(['Train', 'Validation'], loc='upper right')
plt.show()


# In[24]:
# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

# In[25]:
from sklearn.metrics import roc_auc_score, roc_curve
import matplotlib.pyplot as plt

# Compute the predicted probabilities for each class
y_pred_proba = model.predict(X_test)

# Compute the average AUC score across all classes
auc = roc_auc_score(y_test, y_pred_proba, multi_class='ovr')
print('Average AUC:', auc)

# Compute the FPR and TPR for each class
fpr = dict()
tpr = dict()
for i in range(7):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_pred_proba[:, i])

# Plot the ROC curve for each class
```

```
plt.figure()
for i in range(7):
    plt.plot(fpr[i], tpr[i], label='ROC curve of class {}'.format(i))
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC curves for multiclass output')
plt.legend(loc="lower right")
plt.show()

# In[26]:
# Make predictions on the test set
y_pred_proba = model.predict(X_test)
y_pred = y_pred_proba.argmax(axis=1)

#convert one hot encoded y_test to numerical format
y_test = y_test.argmax(axis=1)

# Evaluate model
print("Accuracy: ", accuracy_score(y_test, y_pred))
print("Precision: ", precision_score(y_test, y_pred, average='weighted'))
print("Recall: ", recall_score(y_test, y_pred, average='weighted'))
print("F1 Score: ", f1_score(y_test, y_pred, average='weighted'))



# In[31]:
import numpy as np
rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
r2 = metrics.r2_score(y_test, y_pred)
print(f"\nr2: {rmse}") ##print mean square error and r2
```

```
print(f"\nRMSE: {rmse}")



# In[38]:
def plot_conf_mat(y_test, y_pred):
    '''prints the confusion matrix'''
    from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test, y_pred)

    fig, ax = plt.subplots(figsize=(5,5))
    ax.matshow(cm, cmap=plt.cm.Blues, alpha=0.8)
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            ax.text(x=j, y=i,s=cm[i, j], va='center', ha='center', size='small')

    plt.xlabel('Predictions', fontsize=10)
    plt.ylabel('Actuals', fontsize=10)
    plt.title('Confusion Matrix', fontsize=10)
    plt.show()
    return



# In[39]:
plot_conf_mat(y_test, y_pred)
```

Code of chatbot:

```
# In[1]:
import numpy as np
import pandas as pd
import warnings
import matplotlib.pyplot as plt
import datetime
```

```
# In[2]:
pd.set_option('display.max_rows', 99999)
pd.set_option('display.max_columns', 500)
warnings.filterwarnings("ignore")


# In[3]:
df = pd.read_csv('epc_best20.csv')
df.shape


# In[4]:
objList = df.select_dtypes(include = "object").columns
# Label Encoding for object to numeric conversion
epc_encoded = df.copy()
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for feat in objList:
    epc_encoded[feat] = le.fit_transform(df[feat].astype(str))

epc_encoded.info()


# In[5]:


from sklearn.preprocessing import LabelEncoder
# Split data into features and labels
X = epc_encoded.drop(["Current energy efficiency rating band", "Unnamed: 0"], axis=1)
y = epc_encoded["Current energy efficiency rating band"]
```

In[6]:

```
from sklearn.model_selection import train_test_split
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In[7]:

```
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score
# fit model no training data
model = XGBClassifier()
model.fit(X_train, y_train)
# make predictions for test data
y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]
# evaluate predictions
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

In[8]:

```
dic20 ={2 : 'C',
        3 : 'D',
        1 : 'B',
```

```
        4 : 'E',
        5 : 'F',
        0 : 'A',
        6 : 'G' }


# In[9]:


dic19 = {1:1 ,
         2:0 }


# In[10]:


dic18 = {1:2 ,
         2:9 ,
         3:10 ,
         4:7 ,
         5:8 ,
         6:6 ,
         7:0 ,
         8:5 ,
         9:4 ,
         10:3 ,
         11:1 }


# In[11]:
```

```
dic17 = {1:1 ,
        2:0 }


# In[12]:


dic16 = {1:1 ,
        2:0 }


# In[13]:


dic15 = {1:0 ,
        2:1 ,
        3:4 ,
        4:5 ,
        5:3 ,
        6:2 }


# In[14]:


dic14 = {1:1 ,
        2:4 ,
        3:0 ,
        4:3 ,
        5:2 }
```

In[15]:

```
dic13 = {1:1 ,
        2:4 ,
        3:0 ,
        4:2 ,
        5:3 }
```

In[16]:

```
dic12 = {1:1 ,
        2:8 ,
        3:5 ,
        4:0 ,
        5:7 ,
        6:6 ,
        7:3 ,
        8:4 ,
        9:2 }
```

In[17]:

```
dic9 = {1:2 ,
        2:1 ,
        3:0 ,
        4:3 ,
        5:4 }
```

In[18]:

```
dic8 = {1:4 ,
        2:8 ,
        3:7 ,
        4:5 ,
        5:0 ,
        6:2 ,
        7:6 ,
        8:1 ,
        9:3 }
```

In[19]:

```
dic7 = {1:0 ,
        2:2 ,
        3:1 ,
        4:3 }
```

In[20]:

```
dic6 = {1:2 ,
        2:1 ,
        3:0 }
```

```
# In[21]:


dic3 = {1:10 ,
        2:0 ,
        3:1 ,
        4:2 ,
        5:3 ,
        6:4 ,
        7:5 ,
        8:6 ,
        9:7 ,
        10:8 ,
        11:9 }


# In[24]:


print(" Welcome to our Energy Efficiency Prediction chatbot\n",
      "=======================================================\n",
      "Please choose one of the choices:\n",
        "1- Calculate energy efficiency rating band\n",
        "2- exit from chat")

c = int(input())

number_list = ['The total floor area of the buildings m2',
               'The total current energy costs over 3 years (SR)',
               'The current hot water costs over 3 years (SR)',
               'The age band when building part constructed\n 1- before 1919\n 2- 1919-1929\n 3- 1930-
1949\n 4- 1950-1964\n '
```

```
'5- 1965-1975\n 6- 1976-1983\n 7- 1984-1991\n 8- 1992-1998\n 9- 1999-2002\n 10- 2003-2007\n 11- 2008 onwards\n',

'The average height of the lowest storey of the dwelling (Units: m)',

'The percentage of low energy lighting ',

'The type of mechanical ventilation\n 1- natural\n 2- mechanical, supply and extract\n 3- mechanical, extract only',

'The tenure type of the property\n 1- owner-occupied\n 2- rented (social)\n 3- rented (private)\n 4- unknown',

'The type of transaction or purpose that triggered EPC assessment\n 1- marketed sale\n 2- rental\n '

'3- none of the above\n 4- new dwelling\n 5- ECO assessment\n 6- RHI application\n 7- non marketed sale\n '

'8- FiT application\n 9- assessment for green deal',

'The type of property:\n 1- House\n 2- Flat\n 3- Bungalow\n 4- Maisonette\n 5- Park home',

'If the wall is insulated 1, otherwise 0',

'If the roof is insulated 1, otherwise 0',

'The type of material of wall:\n 1- cavity wall\n 2- timber frame\n 3- sandstone or limestone\n 4- Unknown\n 5- system built\n 6- solid brick\n 7- granite or whinstone\n 8- park home wall\n 9- cob',

'The type of material of roof:\n 1- Pitched\n 2- Unknown\n 3- Flat\n 4- Roof rooms\n 5- Thatched',

'The type of floor construction:\n 1- Suspended\n 2- Unknown\n 3- Solid\n 4- To unheated space\n 5- To external air ',

'The glazing type of window:\n 1- double glazed\n 2- high performance glazing\n 3- single glazed\n 4- triple glazed\n 5- secondary glazing\n 6- multiple glazing',

'Does the main heat system uses mains gas?\n 1- yes\n 2- no',

'Does the main heat control system includes a programmer?\n 1- yes\n 2- no  ',

'The percentage of low lighting\n 1- 0%\n 2- 10%\n 3- 20%\n 4- 30%\n 5- 40%\n 6- 50%\n 7- 60%\n 8- 70%\n 9- 80%\n 10- 90% \n 11- 100%',

'Does the second heat system is room heaters? \n 1- yes\n 2- no']


if c == 1:

   print()

   print("Before you answer the question please check the url documentation below to help you answer the questions")
```

```
print("https://drive.google.com/file/d/1wKlgjZtFxceuaHkzFsmxWZU41WQXXjmk/view?usp=sharing")
    print()
    features = []
    for i in range(20):
        print("Enter your answer:\n", number_list[i], )
        if i == 0:
            value = float(input())
            features.append(value)
            print()
            continue
        elif i == 1:
            value = float(input())
            value = value * 0.22
            features.append(value)
            print()
            continue
        elif i == 2:
            value = float(input())
            value = value * 0.22
            features.append(value)
            print()
            continue
        elif i == 3:
            value = int(input())
            features.append(dic3[value])
            print()
            continue
        elif i == 4:
            value = float(input())
            features.append(value)
            print()
```

```
        continue
    elif i == 5:
        value = int(input())
        features.append(value)
        print()
        continue
    elif i == 6:
        value = int(input())
        features.append(dic6[value])
        print()
        continue
    elif i == 7:
        value = int(input())
        features.append(dic7[value])
        print()
        continue
    elif i == 8:
        value = int(input())
        features.append(dic8[value])
        print()
        continue
    elif i == 9:
        value = float(input())
        features.append(dic9[value])
        print()
        continue
    elif i == 10:
        value = float(input())
        features.append(value)
        print()
        continue
    elif i == 11:
```

```
        value = float(input())
        features.append(value)
        print()
        continue
    elif i == 12:
        value = int(input())
        features.append(dic12[value])
        print()
        continue
    elif i == 13:
        value = int(input())
        features.append(dic13[value])
        print()
        continue
    elif i == 14:
        value = int(input())
        features.append(dic14[value])
        print()
        continue
    elif i == 15:
        value = int(input())
        features.append(dic15[value])
        print()
        continue
    elif i == 16:
        value = int(input())
        features.append(bool(dic16[value]))
        print()
        continue
    elif i == 17:
        value = int(input())
        features.append(bool(dic17[value]))
```

```
            print()
            continue
        elif i == 18:
            value = int(input())
            features.append(dic18[value])
            print()
            continue
        elif i == 19:
            value = int(input())
            features.append(bool(dic19[value]))
            print()
            continue
    f = np.array([features])
    # using inputs to predict the output
    prediction = model.predict(f)
    print("\nThe prediction of Energy Efficiency Rating Band is: ",dic20[prediction[0]])
else:
    print("\nThank you and see you soon...")
```