1. Introduction

In the modern world, energy management is a critical issue for both households and energy providers. Predicting energy consumption accurately enables better planning, cost reduction, and optimization of resources. The goal of this project is to develop a machine learning model that can predict household energy consumption based on historical data. Using this model, consumers can gain insights into their usage patterns, while energy providers can forecast demand more efficiently

By the end of this project, learners should provide actionable insights into energy usage trends and deliver a predictive model that can help optimize energy consumption for households or serve as a baseline for futher research into energy management systems.

1. Dataset Overview

* Link download: <https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption>

A screenshot of a computer program

AI-generated content may be incorrect.

* In the original dataset, there are 9 columns, which are Date, Time, Global\_active\_power, Global\_reactive\_power, Voltage, Global\_intensity, Sub\_metering\_1, Sub\_metering\_2, Sub\_metering\_3. Besides, the above dataset has 2049280 rows
* Attribute information
  + Date: Date in format dd/mm/yyyy
  + Time: time in format hh:mm:ss
  + Global\_active\_power: household global minute-averaged active power (in kilowatt)
  + Global\_reactive\_power: household global minute-averaged reactive power (in kilowatt)
  + Voltage: minute-averaged voltage (in volt)
  + Global\_intensity: household global minute-averaged current intensity (in ampare)
  + Sub\_metering\_1: energy sub-metering No.1 (in watt-hour of active energy).
  + Sub\_metering\_2: energy sub-metering No.2 (in watt-hour of active energy)
  + sub\_metering\_3: energy sub-metering No.3 (in watt-hour of active energy)

A black rectangular object with white lines

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

- The image above shows the number of null values present in the dataset.

A screenshot of a computer

AI-generated content may be incorrect.  
  
- The image above shows the first 5 values present in the original dataset

1. Feature Engineering

A screen shot of a computer program

AI-generated content may be incorrect.

* First, convert the data type of the colmns (except Date and Time columns) to float64 to handle invalid values (converted to NaN). This is a data cleaning step that ensures the numeric columns are in the right format for use in machine learning models.

A screenshot of a computer program

AI-generated content may be incorrect.

* The two lines in the image above are used to check and remove null values. This is part of the data cleaning process, because missing values can affect the performance of the model. Therefore, removing null rows ensures that the dataset no longer contains missing values, making the data ready fo further analysis or modeling steps

A screenshot of a computer

AI-generated content may be incorrect.

* Even though there are no more null values, the code still uses the ffill() (forward fill) and bfill() (backward fill) methods to fill in missing values (if any). This is a technique for replacing missing values, which is also part of the data cleaning process

A screen shot of a computer

AI-generated content may be incorrect.

* The above code uses StandardScaler to standardize the numeric columns in the DataFrame df, then updates df\_scaled and prints the first 5 rows. The final result is to display 5 rows of df\_scaled, with the numeric columns standardized

A screen shot of a computer

AI-generated content may be incorrect.

* The above code creates lag features for the numeric columns in df\_scaled with lag 1 to 3, deletes rows containing NaN, and prints the first 5 rows. Lag features is an important technique in problems related to time series. In time series data, the current value often depends on past values. For example, the electricity consumption at time t may be affected by the consumption at t -1, t-2….

1. Exploratory Data Analysis (EDA)

* Definition of EDA:
  + Exploratory Data Analysis (EDA) is the initial data analysis step to understand the structure, characteristics, and patterns in the data
  + The main goal of EDA is exploring data through descriptive statistics and visualization.
* Descriptive statistics
  + A screenshot of a computer

    AI-generated content may be incorrect.
  + The image contains a Python code snippet and its output. The code defines a function detailed\_summary(df) that calculates statistical summaries for a DataFrame (df) containing numeric data. The function includes:
    - A summary of DataFrame using df.describe() with the option include=’number’
    - Selection of only numeric columns using df.select\_dtypes(include=[‘number’])
    - Calculation of skewness (skew) and kurtosis (kurtosis) for the numeric data
    - The function returns the enhanced summary
  + The output shows the detailed statistical summary for a DataFrame with the following columns:
    - Global\_active\_power
    - Global\_reactive\_power
    - Voltage
    - Global\_intensity
    - Sub\_metering\_1
    - Sub\_metering\_2
    - Sub\_metering\_3
  + The summary includes:
    - Count: number of non-null entries
    - Mean: average value
    - Std: standard deviation
    - Min: minimum value
    - 25%: 25th percentile (first quartile)
    - 50%: 50th percentile (median)
    - 75%: 75th percentile (third quartile)
    - Max: maximum value
    - Skew: skewness (measure of asymmetry)
    - Kurtosis: kutosis (measure of peakedness or tailedness)
  + Code breakdown
    - Summary = df.describe().T: transposes the basic statistical summary (count, mean, std, min, 25%, 50%, 75%, max) of the DataFrame
    - Numeric\_df = df.select\_dtypes(include=[‘number’]): filters the DataFrame to include only numeric columns
    - Summary[‘skew’] = numeric\_df.skew(): adds a new row for skewness, which indicates the distribution’s asymmetry
    - Summary[‘kurtosis’] = numeric \_df.kurtosis(): adds a new row forn kurtosis, which measures the “tailedness” of the distribution
    - Return summary: returns the augmented summary
  + Output Analysis:
    - Global\_active\_power: mean ~ 1.09, skew ~ 1.79, kurtosis ~ 4.22, therefore, we can have a right-skewed distribution with heavy tails
    - Global\_reactive\_power: mean ~ 0.12, skew ~ 1.26, kurtosis ~ 2.61, indicating some asymmetry and heavier tails
    - Voltage: mean ~ 240.84, skew ~ -0.33, kurtosis ~ 0.72, therefore, we can have a slightly left-skewed distribution with light tails
    - Global\_intensity: mean ~ 4.63, skew ~ 1.85, kurtosis ~ 4.60, indicating a right-skewed distribution with heavy tails
    - Sub\_metering\_1: mean ~ 1.12, skew ~ 5.94, kurtosis ~ 35.64, so we should extreme right skewness and heavy tails, possibly due to many zero or low values
    - Sub\_metering\_2: mean ~ 1.30, skew ~ 7.09, kurtosis ~ 57.90, indicating extreme right skewness and very heavy tails
    - Sub\_metering\_3: mean ~ 6.46, skew ~ 0.72, kurtosis ~ -1,28, so we should have a left-skewed distribution with light tails
  + Summary
    - Columns like Sub\_metering\_1 and Sub\_metering\_2 show extreme skewness and kurtosis, which might indicate sparse usage or many zero values, typical in sub-metering data where certain appliances are not always active
    - Voltage has a relatively symmetric distribution with a mean close to 240V, which is typical for household voltage
* Outlier Detection
  + A screenshot of a computer program

    AI-generated content may be incorrect.
  + The code in the image above shows counting outliers using IQR. IQR stands for Interquartile Range, a statistical measure used to describe the dispersion of data and is often used to detect outliers in a dataset
  + IQR is the distance between the third quartile (Q3) and the first quartile (Q1) in the dataset
    - Q1: The value below which 25% of dataset lies (first quartile)
    - Q3: The value below which 75% of dataset lies (third quartile)
    - IQR = Q3 – Q1: represents the range of the middle 50% of the data, helping to eliminate the influence of other extremes and extremes
  + The code uses the IQR method to identify and count outliers in numeric columns. Outlier detection is part of Feature Engineering, as outliers can skew a machine learning model if not handled (removed, replaced, or transformed)

1. Methodology

* Data Loading
  + Reading the file household\_power\_consumption.txt
  + Using pandas to read file with ‘;’ seperator and handle large data (low\_memory=False)
* Data preprocessing
  + Data type conversion
    - Converting numeric columns (except Date and Time) to float64 type using pd.to\_numeric, handling invalid values using errors=’coerce’
  + Handling missing values
    - Checking for missing values using df.isnull().sum()
    - Removing rows containing missing values using df.dropna(inplace=True)
    - Appling Forward Fill (ffill) and Backward Fill (bfill) to ensure there are no missing values
  + Checking data
    - Looking at the data types (df.dtypes), data size (df.shape), and the first 5 rows (df.head()) to confirm data has been cleaned
* Exploratory Data Analysis (EDA)
  + Descriptive statistics
    - Caculating statistical indices (count, mean, std, min, 25%, 50%, 75%, max) for numeric columns
    - Adding skew and kurtosis metrics to evaluate data distribution
  + Outliers Detection
    - Using the Interquartile Range (IQR) method
* Visualizing the results
  + Drawing a scatter plot using matplotlib to compare actual energy consumption (y\_test) and predict energy consumption (y\_pred)

1. Evaluation & Results

* In the above problem, we use 2 machine learning models, Linear Regression and Decision Tree, for training and the results obtained will be RMSE and MAE
  1. Split Test and Train dataset

A screenshot of a computer program

AI-generated content may be incorrect.

* Before applying Machine learning models for calculation, we proceed to split the dataset into 2 parts: Train and Test.
* In the first code snippet in the figure, target is defined as ‘Global\_active\_power’, which is the column that the model will predict and features is the list of columns used as input for the model, including ‘Global\_reactive\_power’, ‘Voltage’, ‘Global\_intensity’, ‘Sub\_metering\_1’, ‘Sub\_metering\_2’, ‘Sub\_metering\_3’. Next is creating the input and output datasets.
  + X = df[features]: Get the feature columns from DataFrame (df) and assign them to X
  + Y = df[target]: Get target columns from DataFrame (df) and assign them to y
* In the second code snippet, using train\_test\_split from sklearn.model\_selection to split the data into 2 training sets (X\_train, y\_train) and test sets (X\_test, y\_test)
* In the third code snippet, using StandardScaler from sklearn.preprocessing to standardize the data
  + X\_train\_scaled = scaler.fit\_transform(X\_train): using fit\_transform to calculate the mean and standard deviation of X\_train, then normalize the training data
  + X\_test\_scaled = scaler.transform(X\_test): applying the same parameters (mean and standard deviation) from X\_train to normalize X\_test, ensuring consistency without “leaking” information from the test set

5.2 Linear Regression

A computer screen shot of a program

AI-generated content may be incorrect.

* First, import the necessary libraries
  + Sklearn.linear\_model.LinearRegression: library for building linear regression models.
  + Numpy: library for handling arithmetic operations
  + Sklearn.metrics.mean\_squared\_error: calculating mean square error (RMSE)
  + Sklearn.metrics.mean\_absolute\_error: calculating the mean absolute error (MAE)
* Second, training linear regression model
  + Lr\_model = LinearRegression(): Creating a linear regression model object
  + Lr\_model.fit(X\_train\_scaled, y\_train): Using the normalized training data (X\_train\_scaled) and labels (y\_train) to train the model
  + Y\_pred\_lr = lr\_model.predict(X\_test\_scaled): Using the trained model to predict the energy consumption value (Global\_active\_power) on the test set (X\_test\_scaled). The predicted results are saved to y\_pred\_lr
* Third, model performance evaluation
  + With RMSE (Root Mean Squared Error):
    - mean\_squared\_error(y\_test, y\_pred\_lr): we calculate the MSE between the actual value (y\_test) and the predicted value (y\_pred\_lr)
    - np.sqrt: take the square root of the MSE to calculate the RMSE
    - The result when calculating RMSE using Linear Regression model is 0.0393. The above result shows that the model has low error
  + With MAE (Mean Absolute Error):
    - Mae\_lr = mean\_absolute\_error(y\_test, y\_pred\_lr): MAE is the average of the absolute values of the errors
    - The result when calculating MAE using Linear Regression model is 0.0246. The above result shows that the model has low average error
  1. .Decision Tree

A computer screen shot of text

AI-generated content may be incorrect.

* First, import the necessary libraries
  + Sklearn.tree.DecisionTreeRegressor: library for building decision tree regression models
* Second, training decision tree model
  + Dt\_model = DecisionTreeRegressor(random\_state=42): creating a decision tree
  + Dt\_model.fit(X\_train\_scaled, y\_train): Using the normalized training data (X\_train\_scaled) and labels (y\_train) to train the model
  + Y\_pred\_dt = dt\_model.predict(X\_test\_scaled): Using the trained model to predict the energy consumption value (Global\_active\_power) on the test set (X\_test\_scaled). The prediction result is saved in y\_pred\_dt
* Third, model performance evaluation
  + With RMSE:
    - Mean\_squared\_error(y\_test, y\_pred\_dt): we calculate the MSE between the actual value (y\_test) and the predicted value (y\_pred\_lr)
    - np.sqrt: take the square root of the MSE to calculate the RMSE
    - The result when calculating RMSE using Decision Tree model is 0.0460. The above result shows that the model with acceptable error.
  + With MAE:
    - Mae\_dt = mean\_absoluted\_error(y\_test, y\_pred\_dt): MAE is the average of the absolute values of the errors
    - The result when calculating MAE using Decision Tree model is 0.0229. The above result shows that the model has low average error

1. Discussion

A black background with white text

AI-generated content may be incorrect.A black background with white text

AI-generated content may be incorrect.

* From the results of calculating RMSE and MAE of the two models Linear Regression and Decision Tree, we can see some points to note
  + Performance comparision between Linear Regression and Decision Tree
    - The value of RMSE in the Linear Regression is lower than that in the Decision Tree, which implies that RMSE is sensitive to large errors, which suggests that linear regression is better able to control predictions that deviate from the actual value. Decision Tree may produce some larger errors, resulting in higher RMSE
    - The value of MAE in the Linear Regression is higher than that in the Decision Tree, which implies that MAE measures the absolute average deviation, not magnifying the large error like RMSE. This shows that the Decision Tree can predict more accurately in most cases (with smaller error), but there may be some cases where large error increases RMSE
    - From the above two opinions, we can see this difference may be due to the nature of two models
      * Linear Regression assumes a linear relationship and can be less sensitive to outliers
      * Decision Tree can be affected by outliers and be at risk of overfitting if not fine tuned
  + In summary, if you want to minimize large errors, RMSE is more important, and Linear Regression is the better choice, else if you want to predict accurately in the majority of cases, MAE is more important, and Decision Tree is the better choice

1. Conclusion and Future Work
   1. Conclusion
   * Linear Regression has lower RMSE, suitable if reducing large errors is a priority
   * Decision Tree has lower MAE, which is suitable if the priority is to predict correctly in the majority of cases
   1. Future Work

Considering to apply other models to calculate such as Random Forest, kNN,….

Tweak hyperparameters for Decision Tree to reduce overfitting

1. References

* Linear Regression.pdf – provided by Dr. Nguyen Hong Son
* Feature Engineering.pdf - provided by Dr. Nguyen Hong Son
* Tree – based method.pdf - provided by Dr. Nguyen Hong Son
* <https://scikit-learn.org/stable/modules/model_evaluation.html#mean-squared-error>
* https://www.youtube.com/watch?v=xi0vhXFPegw