AIDI 2005 – Capstone Term II

Title: Energy Consumption Prediction Using Data Mining Models

Submitted By:

- Bavithra Ganesan (100900119)
- Jasmeet Kaur (100881373)
- Pritesh Dalal (100872247)
- Rutvik Shah (100886648)

Table of Content

Energy Consumption Prediction Using Data Mining Models

1. Abstract

This research paper emphasizes the importance of energy as a vital resource for human subsistence, everyday activities, and economic growth. However, its availability is limited, particularly for non-renewable resources such as fossil fuels, which is why energy conservation is critical for long-term growth. The article focuses on the forecast of electrical energy usage using machine learning and deep learning models, which is critical at the national or regional level, and research is required to optimise electrical energy consumption. Industrial sectors, which are important indicators of a country's economic level, consume a lot of energy when compared to other industries. As a result, there is a need to limit electricity use, particularly in the industrial sector. Using DAEWOO steel industry energy consumption in South Korea as an example, the study recommends creating, comparing, and evaluating data mining-based models for forecasting energy consumption in the manufacturing sector. By identifying maintenance difficulties and establishing the optimal power management tactics, the proposed best model can help estimate energy consumption in similar businesses and enhance energy management strategies.

2. Introduction:

Energy is considered one of the most substantial resources because it is vital for human subsistence, driving everyday activities, and pushing economic growth and development. But its accessibility is restricted, specifically of non-renewable resources like fossil fuels. Therefore, the necessity for energy continues to increase as the world population grows. Energy preservation is required not merely for developing a sustainable environment for future productivity but also for use by local consumers and power production corporations. There is therefore a need for electrical power energy use at different industrial and domestic levels. Subsequently, the prediction of electrical energy usage becomes pressing and crucial at a national or regional level, and research to optimize the consumption of electrical energy is required.

Industrial sectors or manufacturing industries are crucial sectors and important indicators of a country's economic level. But they have a high rate of energy consumption compared to other industries and therefore occurs a need to regulate power utilization primarily in the Industrial sector.

Taking into consideration the need for enhancing, optimizing, and handling the energy consumption within the Industrial sector, our research would like to develop and compare data mining-based models for forecasting Industry and manufacturing sector energy consumption by considering the data of DAEWOO steel industry energy consumption of South Korea [1]. The developed model can be used to predict energy consumption by similar industries and hence energy demand of the steel manufacturing sector for South Korea or regions with similar industrial composition, expertise level, power value, financial scale, and public strategy. A precise and consistent energy forecast scheme can help industrial managers identify maintenance challenges and define the best power management strategies.

3. Related Work:

According to Mosav, Amir et al. (2019) [2] the use of composite machine learning models and other models such as Artificial Neural Networks (ANNs), and Decision Trees (DT) have increased in recent years and the accuracy of such models have increased a lot. The work conducted a detailed review of the related literature to pinpoint popular ML techniques and evaluation of their effectiveness in energy consumption prediction. This paper concludes that ML models are remarkably effective in increasing accuracy and performance and shows a magnificent rise in predicting technologies with hybrid and embedded models.

Ahmad, Tanveer, et al. (2022) [3] conducted a study to demonstrate the importance of data-driven probabilistic ML methods and their real-time applicability to intelligent energy networks and systems. The research focused on two principal areas: the application of machine learning in basic energy technologies and ML use cases for electrical utilities. The study stated that the utility industry and power sector could probably save between $250 billion and $800 billion with the use of intelligent machine learning automation algorithms in the energy infrastructure.

Yu, Zhun, et al. (2010) [4] utilized a machine learning technique based on a decision tree to predict the energy demand of the building. According to their research, a Tree-based model known as Classification and Regression Trees (CART) can be used to provide a detailed forecast of developing energy demand with the least error. The technique predicts and classifies categorical variables with a flow diagram-like tree diagram that can be used by customers to quickly acquire helpful data. The study used the technique to predict housing building energy performance indexes. The results of the study establish the use of this method of decision tree to classify and forecast building energy demand levels with 93% accuracy level for training data. The technique, therefore, provided the blend of substantial factors and the threshold estimates that can be used to provide high building energy performance.

To predict energy consumption by a building, Yang, Jin, et al. (2005) [5] used the technique of adaptive Artificial Neural Networks (ANNs) which can adjust themselves to unforeseen configuration modifications in the inbound data and consequently can be utilized for real-time online energy prediction of the infrastructural unit. The paper presented and tested two adaptive ANN models: accumulative training and sliding window training and the computational trials used synthetic data and measured data. The study showed that adaptable Artificial Neural Networks (ANNs) have higher accuracy than static ANNs.

Maha, Alanbar (2020) [6] has employed the deep learning concept of neural networks to forecast the energy consumption of an educational building. To make this prediction, the Long short-term memory (LSTM) method was used for medium-term energy consumption predictions in the building. LSTM is a type of neural network that can use the output from the previous layer as input to the next layer to tackle the vanishing gradient issue, allowing the model to remember it. The study demonstrated the causes for the rise in energy consumption in educational buildings and the model achieved an impressive accuracy of 94.31% when predicting energy consumption with the help of this technique.

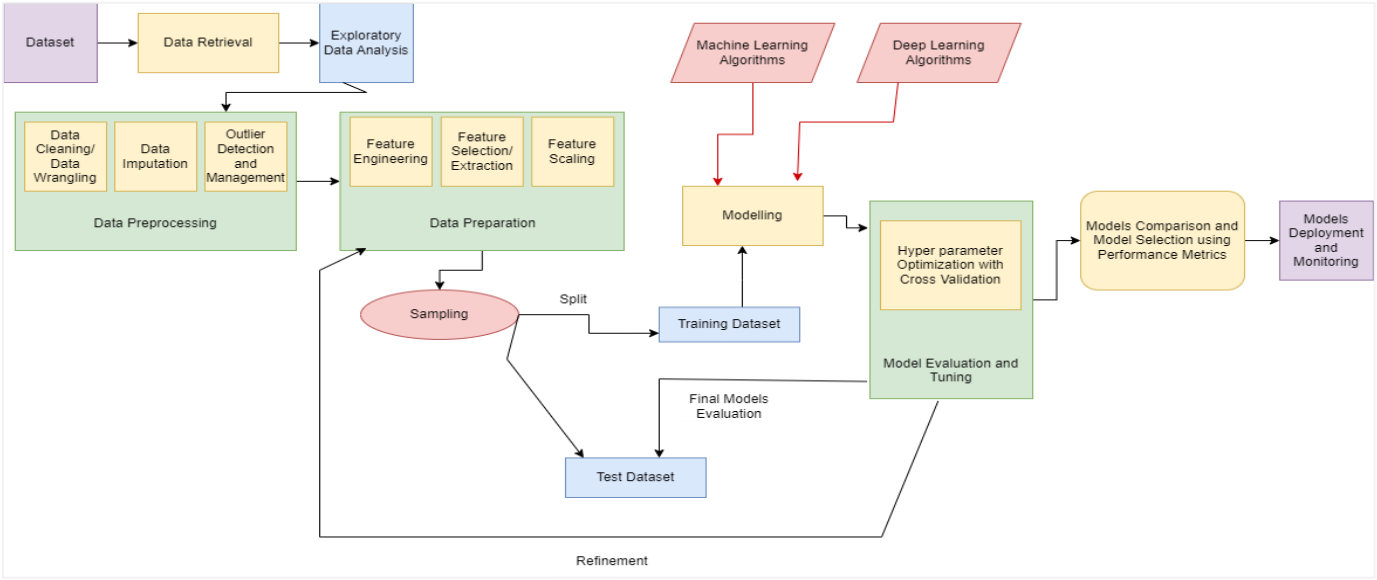
Our work is different from these works because most of the research considered the energy consumption prediction for a singular domestic unit of building, whereas our focus is the steel industry and hence manufacturing unit which has more energy requirements than a building. Also, the above works considered only a single data mining model in their respective research papers and no comparison of different data mining models is made to find the best model, but we would compare different models in our research to find the best model.

4. Experimental Design

The **main purpose** of the research is to look at the possibility of using data mining methods to forecast energy use in industrial settings. The study is focused on predicting daily energy consumption patterns in a South Korean steel manufacturing factory utilising data mining techniques like machine learning algorithms and artificial neural networks. The goal of the study is to assess how well these models forecast energy consumption and to find areas where energy can be saved, efficiency can be increased and to offer information about how they might be used in real-world.

3.2 Proposed Idea/ Flowchart

The following flowchart shows the proposed idea of the research:



Data analysis often consists of multiple stages, beginning with data retrieval from a dataset and progressing to exploratory data analysis (EDA) to obtain insights and uncover trends. Following that is data pre-processing, which involves cleaning, wrangling, imputing missing values, and managing outliers. Following feature engineering, scaling, and selection/extraction, the dataset is sampled into training and test sets. Using hyper-parameter optimisation and cross-validation procedures, the models are then applied, tested, and refined. Finally, the best performing model is chosen for deployment and its performance is monitored. The overarching goal is to get accurate predictions and insights from data that may be used to inform decision-making in a range of fields.

3.3 Dataset:

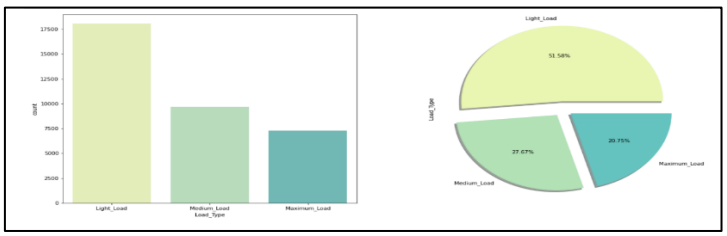
The data was obtained from the South Korean company DAEWOO steel.co. Ltd in Gwanyang. Coils of various sorts, steel plates, and iron plates are produced by it. On the Korea Electric Power Company website (pccs.kepco.go.kr), the data on electricity use is saved in a cloud-based system, and extensive insights on daily, monthly, and yearly energy consumption patterns are computed and shown.

It consists of six continuous attributes like Date, Industry energy consumption, lagging current power factor, Leading current power factor, Number of seconds from midnight, CO2 and three categorical attributes like Week status, Day of Week, and Load Type. Also, it consists of Lagging current reactive factor and Leading current reactive factor which are of float values.

Link to dataset: [UCI Machine Learning Repository: Steel Industry Energy Consumption Dataset Data Set.]

Exploratory Data Analysis

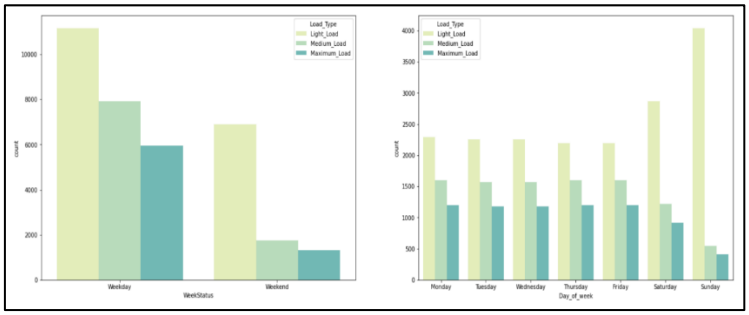
EDA Chart 1: Visualization of frequency of usage of different load types



The visualization shows that 'Light_Load' has highest frequency (51.58%) among different types of loads, followed by 'Medium_Load' (27.67%) and 'Maximum_Load' (20.75%) has lowest frequency.

Picture Credits: author

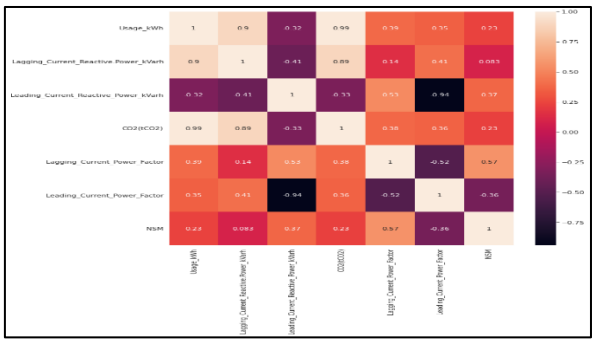
EDA Chart 2: Visualization of frequency of usage of different load types on different days of week



The first plot shows that weekdays have higher frequency than weekends. The second plot shows that: 'Light_Load' frequency is high on weekends and constant for weekdays and 'Medium_Load' frequency and 'Maximum_Load' frequency are low on weekends (lowest on Sundays) and constant for weekdays.

Picture Credits: author

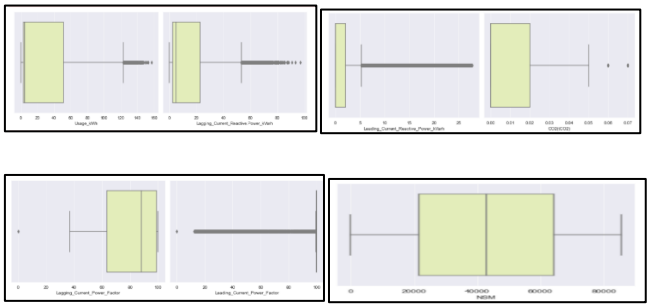
EDA Chart 3: A map of correlations to analyse the relationship between attributes.



From the correlation graph, it can be observed that lagging current reactive power is directly proportional to usage, leading to greater energy consumption. This usage is also highly correlated with tCO2, resulting in increased emission of CO2 that is detrimental to the environment. Receiving more power from the source, lagging reactive power has a high correlation with tCO2, thus causing a rise in CO2 emission.

Picture Credits: author

EDA Chart 4: Box plots for outlier detection



Picture Credits: author

In the above box plots, it can see that there are outliers in all features except NSM.

Google Collab link: https://colab.research.google.com/drive/1l9pknfU92CU80wVgdgANyXD61IaVYsiF?usp=sharing

3.4 Data Pre-processing and Preparation:

3.4.1 Data Pre-processing

Data Cleaning /Data Wrangling: Data cleaning, also known as data wrangling, is the process of identifying and correcting errors, inconsistencies, and inaccuracies in a dataset prior to analysing. On an abstract level, our data does not contain any null values.

Data Imputation: The data doesn't contain any null or missing values; we are performing data imputation once we are treating the outliers in our dataset.

Outlier Detection and Management

Outlier detection is a machine learning technique that identifies and handles unusual or observations in a dataset. We have used Interquartile Range (IQR) method for identifying outliers by comparing the upper and lower quartiles of a dataset. Data points that fall outside of a certain range are considered outliers.

MICE imputation is an effective method for dealing with missing data, including outliers. In each iteration, the imputer generates new imputed values based on the previously imputed values, and the process continues until convergence or until a maximum number of iterations is reached. If the imputer is run multiple times on the same dataset, it may result in overfitting, which occurs when the imputer fits the imputed values too closely to the observed values, causing the imputed values to lose generalizability. To avoid overfitting and data loss, we performed the imputer iteration only once.

3.4.2 Data Preparation

Feature Engineering: The process of creating new features or modifying existing ones from raw data to improve the performance of machine learning models is referred to as feature engineering. We made use of the fit transform () method of the Label Encoder class can be used to fit and transform the categorical variable into numerical values.

Feature Scaling: Feature scaling is a technique which is utilized to standardize a dataset's range of independent variables or features. StandardScaler of python ensures that the new data is on the same scale as the training data and that accurate predictions are produced.

Feature Selection/Extraction: These are the techniques for reducing the number of input features in a regression model are known as feature selection techniques. 'SelectKBest' is a feature selection technique for selecting the top K features based on univariate statistical tests.

3.5 Model Training and Evaluation:

Machine Learning model – Linear Regression

Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables. The process of hyperparameter optimization in linear regression involves trying out different combinations of alpha and regularization types.

For both Linear Regression model – one before and one after hyper parameter optimization – the accuracy measured by R2 score is magnificently similar for training and testing datasets. This means that neither model overfits the data. R2 score greater than 0.98 for both model's training and testing dataset indicates strong relationship between the independent and dependent variables in the model, showing that the model is perfectly fit for the data.

Machine Learning model – Random Forest Regressor (RFR)

Random Forest Regressor is a well-known machine learning algorithm that belongs to the ensemble learning method family. It combines several decision trees to form a powerful regression model capable of making accurate predictions on a wide range of data. For this research we are using two RFR models - a simple model with predefined methods and the other with hyper parameter optimization.

For both RFR models - one before and one after hyper parameter optimization - the accuracy measured by R2 score is significantly similar for the training and testing datasets. This means that neither model overfits the data. R2 scores greater than 0.99 for both models' training and test datasets indicate a strong relationship between the independent and dependent variables in the model, indicating that the model is a good fit for the data.

Deep Learning model - Artificial Neural Network using Regression:

A form of machine learning algorithm known as an artificial neural network (ANN) is based on the biological neural networks seen in the human brain. It is made up of a network of connected nodes, commonly referred to as artificial neurons, that collaborate to process and analyse large amounts of data. The input layer, hidden layer, and output layer are the three different kinds of layers that make up an ANN.

In this paper we build two ANN models – the simple one with predefined parameters and other by using the hyper parameters optimization technique. On assessing the 'Simple ANN model', it is discovered that the accuracy measured through R2 score is similar for the training and testing dataset. Also, as R2 score value is high for both training and test dataset, it can be concluded that 'Simple ANN Model' precisely and accurately predicts the energy consumption prediction. On evaluating the 'ANN MODEL after HYPER PARAMETER OPTIMIZATION, it is discovered that the accuracy measured through R2 score is significantly similar for the training and testing. Also, as R2 score value is high for both training and test dataset, it can be concluded that this model accurately predicts the energy consumption prediction.

3.6 Baseline:

Through this research, our purpose is to develop data mining techniques that improves the prediction of energy consumption compared to baseline methods currently available. To forecast industrial energy consumption, our study suggests using data mining techniques like linear regression, random forest, and artificial neural networks. We can contrast our work with the approaches used in our research by considering the following to confirm that our proposed work advances upon what is currently available.

- **Data pre-processing techniques**: We can consider the effect of various data pre-processing techniques done by previous models on the precision of the predictions for buildings/ industries energy consumption to compare the performance of our suggested model.

- **Model architecture:** By analysing the effect of various model architectures on the precision of the predictions, we can compare how well our suggested model performs. For instance, we could assess how well the models used in the paper compared to those mentioned in the literature review section.

- **Feature selection:** All the features that are available are used for prediction in previous research paper for energy consumption prediction. By considering how feature selection methods affect the precision of the predictions, our research can compare the performance of our suggested model to those currently available.

We can show how your proposed work improves on what is currently available in the field of predicting industrial energy consumption by contrasting it with the methods mentioned above.

3.7 Performance Metrics:

Performance metrics are used to quantify the performance of a model, algorithm, or system. Performance metrics are frequently used in machine learning and data analysis to assess the accuracy and effectiveness of predictive models. Here are some common performance metrics used in machine learning:

- **Mean Squared Error (MSE):** In regression problems, this metric is used to calculate the average squared difference between predicted and actual values.
- **R-squared:** In regression problems, this metric is used to calculate the proportion of the variance in the dependent variable that is explained by the independent variables.
- **Mean Absolute Error (MAE):** This is a common evaluation metric for machine learning models that calculates the average absolute difference between predicted and actual values.

5. Experimental Results:

The following is the summary of the main findings of Machine learning – Linear using Regression:

	MODEL TYPE	DATASET TYPE	R-2 SCORE	MEAN SQUARED ERROR	MEAN ABSOLUTE ERROR
0	SIMPLE Linear Regression MODEL	Training Dataset	0.981203	19.261986	2.481332
1	SIMPLE Linear Regression MODEL	Test Dataset	0.984596	16.070619	2.438943
2	Linear Regression MODEL after HYPER PARAMETER ...	Training Dataset	0.981203	19.261986	2.481332
3	Linear Regression MODEL after HYPER PARAMETER ...	Test Dataset	0.984596	16.070619	2.438943

Picture Credits: author

For both Simple Linear Regression Model and Linear Regression Model after Hyperparameter Tuning, the R-2 Score on the test dataset is higher than the training dataset, which indicates that the model generalizes well on unseen data. The Mean Squared Error (MSE) is lower for the test dataset than the training dataset, which indicates that the model performs better.

The following is the summary of the main findings of Machine learning – Random Forest using Regression:

	MODEL TYPE	DATASET TYPE	R-2 SCORE	MEAN SQUARED ERROR	MEAN ABSOLUTE ERROR
0	SIMPLE RFR MODEL	Training Dataset	0.999176	0.844026	0.309130
1	SIMPLE RFR MODEL	Test Dataset	0.997045	3.082759	0.474755
2	RFR MODEL after HYPER PARAMETER OPTIMIZATION	Training Dataset	0.992581	7.602452	1.596773
3	RFR MODEL after HYPER PARAMETER OPTIMIZATION	Test Dataset	0.990845	9.550956	1.656992

Picture Credits: author

For both RFR models - one before and one after hyper parameter optimization - the accuracy measured by R2 score is significantly similar for the training and testing datasets. This means that neither model overfits the data. R2 scores greater than 0.99 for both models' training and test datasets indicate a very strong relationship, indicating that the model is a good fit for the data.

The following is the summary of the main findings of Artificial Neural Network using Regression:

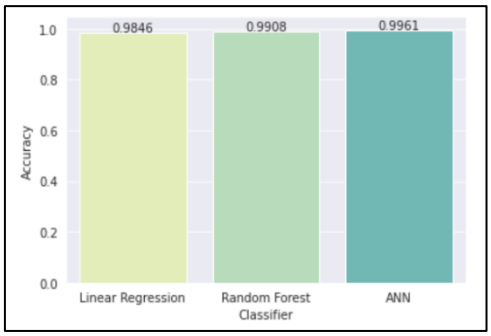
	MODEL TYPE	DATASET TYPE	R-2 SCORE	MEAN SQUARED ERROR	MEAN ABSOLUTE ERROR
0	SIMPLE ANN MODEL	Training Dataset	0.993807	6.346304	1.367634
1	SIMPLE ANN MODEL	Test Dataset	0.991999	8.347358	1.425600
2	ANN MODEL after HYPER PARAMETER OPTIMIZATION	Training Dataset	0.996994	3.080203	0.854162
3	ANN MODEL after HYPER PARAMETER OPTIMIZATION	Test Dataset	0.995571	4.620149	0.878981

Picture Credits: author

The accuracy measured through R2 score is significantly similar for the training and testing dataset for both ANN models – one before and other after hyper parameter optimization. This signifies that both models do not overfit the data. The R2 score of greater than 0.99 for training and test datasets for both models are an indication of an extraordinarily strong relationship indicating that the model is a great fit for the data.

To make comparison of the performance of models, let's compare the R2 Score and MSE of models after hyper parameter optimization by plotting the bar graphs:

Picture Credits: author



From the graphs, ANN has highest R2 score and lowest MSE among all models. The high R2 score implies that these models predict the energy consumption precisely and accurately. Also, there is no overfitting of data as explained above in modelling section. It can imply that right set of features are selected in feature extraction and steps of data preparation and data pre-processing have prepared the data well for modelling. It can be concluded that ANN performs better than other models and predict the energy consumption accurately.

Although our research offers important insights into the potential of data mining techniques for energy consumption prediction in industry, **there are a few questions that are not completely answered by the experimental design.** The first question is what causes the patterns of energy usage seen in the study? The study does not give a thorough analysis of the underlying factors that influence energy use, such as equipment efficiency, production schedules, or weather conditions, even though the data mining models can reliably estimate energy use. The other issue not answered by experiment is how effective are the data mining models over longer time horizons? The study's main goal is to forecast daily energy use. Yet, patterns of energy usage can change across longer time, like monthly or yearly cycles. A deeper understanding of the data mining models' applicability in actual energy management situations may be gained by analysing their performance over this extended time.

There could be **next steps for future direction of research in this direction**. The first is expanding the analysis to more locations and business sectors to evaluate how well the data mining models can be applied broadly. This can entail gathering information from many sources and assessing the models' performance in various scenarios. The second step could be analysing the underlying causes of energy use more thoroughly to find areas where energy can be saved, and efficiency can be increased. This can entail combining more data sources or running extra statistical analyses on the data. The last step could be analysing the effectiveness of the data mining models across extended time spans to determine their applicability in actual energy management scenarios. This could entail evaluating the viability of applying the models in practise and comparing the performance of the models to other prediction approaches or industry standards.

6. Conclusion:

In conclusion, the steel industry is one of the largest energy consumers in the world. To reduce its energy consumption and carbon emissions, the industry can benefit from the use of artificial intelligence (AI) algorithms to predict energy consumption and optimize processes. Various AI algorithms, such as Linear regression, Random Forest Classifier, and Artificial Neural Network (ANN), have been used in the paper to predict energy consumption in the steel industry. Our research indicated that ANN performs better than other models and predict the energy consumption accurately. The use of ANN model to predict energy consumption can be extended to any other industry and can help reduce energy waste, optimize production, and minimize costs. ANN model can identify patterns and anomalies in energy consumption data and alert operators to take corrective action. This can help reduce energy consumption, improve energy efficiency, and reduce costs. To summarize, the use of AI model can enable real-time monitoring of energy consumption and performance, allowing operators to make informed decisions and take corrective action promptly. This can help prevent energy waste and optimize production, leading to significant cost savings and reduced carbon emissions.

7. Bibliography:

[1] "UCI Machine Learning Repository: Steel Industry Energy Consumption Dataset Data Set." *Archive.ics.uci.edu*, https://archive.ics.uci.edu/ml/datasets/Steel+Industry+Energy+Consumption+Dataset

[2] Mosavi, Amir, and Abdullah Bahmani. "Energy Consumption Prediction Using Machine Learning; a Review." 2019, https://doi.org/10.20944/preprints201903.0131.v1.

[3] Ahmad, Tanveer, et al. "Data-Driven Probabilistic Machine Learning in Sustainable Smart Energy/Smart Energy Systems: Key Developments, Challenges, and Future Research Opportunities in the Context of Smart Grid Paradigm." *Renewable and Sustainable Energy Reviews*, vol. 160, May 2022, p. 112128, 10.1016/j.rser.2022.112128. Accessed 8 Mar. 2022.

[4] Yang, Jin, et al. "On-Line Building Energy Prediction Using Adaptive Artificial Neural Networks." *Energy and Buildings*, vol. 37, no. 12, Dec. 2005, pp. 1250–1259, 10.1016/j.enbuild.2005.02.005. Accessed 5 May 2020.

[5] Yu, Zhun, et al. "A Decision Tree Method for Building Energy Demand Modeling." Energy and Buildings, vol. 42, no. 10, Oct. 2010, pp. 1637–1646, 10.1016/j.enbuild.2010.04.006

[6] Alanbar, Maha, et al. "Energy Consumption Prediction Using Deep Learning Technique." International Journal of Interactive Mobile Technologies (IJIM), vol. 14, no. 10, 30 June 2020, p. 166, 10.3991/ijim.v14i10.14383. Accessed 13 July 2020.

8. Google Colab Link:

https://colab.research.google.com/drive/1l9pknfU92CU80wVgdgANyXD61IaVYsiF?usp=sharing

9. Appendix - Code:

```
#Importing dependencies for data processing
import pandas as pd
import numpy as np
from google.colab import files
uploaded = files.upload()
# Loading dataset
data = pd.read_csv('Steel_industry_data.csv')
#EDA
#Analysing for no. of values
data.shape
data.head()
#Displaying the number of null or missing values in Dataframe
data.isnull().sum()
data.nunique()
#Calculating the correlation coefficients between all pairs of numeric columns in a DataFrame for identifying patterns and
 relationships in the data
data.corr()
#Importing dependancies for data visualization
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go

# EDA Chart 1
# Using seaborn library to assign 'Yellow-Green-Blue' palette to different load types
sns.set_palette('YlGnBu')
# Defining figure size using Matplotlib
plt.figure(figsize=(20,8))
# Subplot 1: frequency distribution of Load type in form of bar plot
axs_bar = plt.subplot(1,2,1)
axs_bar = sns.countplot(x = 'Load_Type', data = data)
# Subplot 2: frequency distribution of Load type in form of pie chart
axs_pie =plt.subplot(1,2,2)
axs_pie=data['Load_Type'].value_counts().plot.pie(explode=[0.1, 0.1,0.1],autopct='%1.2f%%',shadow=True);
# EDA Chart 2
#We will continue using same palette of 'YlGnBu' for different load types to maintain symmetry
# Defining figure size using Matplotlib
plt.figure(figsize=(24,8))
# Subplot 1: frequency distribution of variation in load usage on weekdays and weekends using bar plot
plt.subplot(1,2,1)
sns.countplot(data=data, x="WeekStatus", hue="Load_Type")
# Subplot 2: frequency distribution of variation in load usage on different days using bar plot
plt.subplot(1,2,2)
sns.countplot(data=data, x="Day_of_week", hue="Load_Type")
plt.show()
#EDA Chart 3
#Visualizing corrrealtion
plt.figure(figsize=(10,10))
sns.heatmap(data.corr(),annot=True)
# EDA Chart 4
#Creating box plots to identify skewness in data and indentify outliers
def plot_box_plots(dataframe):
    numeric_columns = dataframe.select_dtypes(include=['number']).columns.tolist()
    dataframe = dataframe[numeric_columns]
    for i in range(0,len(numeric_columns),2):
        if len(numeric_columns) > i+1:
            plt.figure(figsize=(10,3))
            plt.subplot(121)
            sns.boxplot(dataframe[numeric_columns[i]])
            plt.subplot(122)
            sns.boxplot(dataframe[numeric_columns[i+1]])
            plt.tight_layout()
            plt.show()
        else:
            sns.boxplot(dataframe[numeric_columns[i]])
print('\nBox Plots\n')
plot_box_plots(data)


# Data Pre processing
print(data.isnull().sum())
print(data.describe())
# Outlier Management
new_data = data.copy()
new_data.shape
#Selecting columns which is of numeric datatype for identifying outliers in the dataset
column = ['Usage_kWh', 'Lagging_Current_Reactive.Power_kVarh',
       'Leading_Current_Reactive_Power_kVarh', 'CO2(tCO2)',
       'Lagging_Current_Power_Factor', 'Leading_Current_Power_Factor']
#Using IQR function for handling numeric data in dataframe in order to identify and remove outliers from dataset
def out_null(column):
        IQR=column.quantile(0.75)-column.quantile(0.25)
        UL=column.quantile(0.75)+1.5*IQR
        LL=column.quantile(0.25)-1.5*IQR
        column.where(column.between(LL,UL),np.nan,inplace=True)
 for i in column:
    out_null(new_data[i])

#Checking for any missing or NaN values in the new DataFrame for cleaning and preparing the data, as it removes any extrem
e values and identifies any missing data that might need to be imputed
new_data.isnull().sum()
```

```
#Using Multiple imputations by chained equations (MICE) imputation on a pandas DataFrame for handling missing data
!pip install impyute
from impyute.imputation.cs import mice
#Importing dependencies for performing imputation on missing data in a dataset
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
mice_imputer = IterativeImputer(max_iter=50, random_state=42, initial_strategy='median', imputation_order='random', skip_c
omplete=True)
mice_imputer.fit(new_data[column])
new_data[column] = mice_imputer.transform(new_data[column])
# Displaying column names of copy of dataframe
new_data.columns
# Saving changes in original database
data = new_data
# Checking null values to see whether original data is altered
data.isnull().sum()
# Creating a new dataframe so that original dataframe is not altered
new_data_check = data.copy()

for i in column:
    out_null(new_data_check[i])
new_data_check.isnull().sum()
data.isnull().sum()
# Data Preparation
y = data['Usage_kWh']
X = data.drop(['Usage_kWh','date'], axis = 1)
#Feature Engineering
#Importing Labelencoder for interpreting categorical variables
from sklearn.preprocessing import LabelEncoder
# Creating a new DataFrame that contains only the categorical variables
cat_variables = X[['WeekStatus','Day_of_week','Load_Type']]
# Initializing label encoder
le = LabelEncoder()
for col in cat_variables:
    X[col] = le.fit_transform(X[col])
#Feature Extraction/ Selection
#Importing module
from sklearn.feature_selection import SelectKBest, f_regression
#Selecting the top 5 features that have the highest correlation with the target variable
k = 5
selector = SelectKBest(score_func=f_regression, k=k)
X_new = selector.fit_transform(X, y)
selected_features = X.columns[selector.get_support()]
print(selected_features)
# Updating X
X = X[['Lagging_Current_Reactive.Power_kVarh', 'CO2(tCO2)', 'Lagging_Current_Power_Factor', 'WeekStatus', 'Load_Type']]
# Feature Scaling
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Data splitting
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test= train_test_split(X_scaled,y,test_size=0.2,random_state=42)

# Modelling
# Linear Regression
from sklearn import linear_model
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
# Building and Training the Linear Regression Model
lnr = linear_model.LinearRegression()
lnr.fit(x_train, y_train)
y_pred_train = lnr.predict(x_train)
mse_train = mean_squared_error(y_train, y_pred_train)
mae_train = mean_absolute_error(y_train, y_pred_train)
r2score_train = r2_score(y_train, y_pred_train)
print(f"R2 score: {r2score_train:.2f}")
print(f"MAE: {mae_train:.2f}")
print(f"Mean squared error: {mse_train:.2f}")
# Model Evaluation
y_pred = lnr.predict(x_test)
mse_test = mean_squared_error(y_test, y_pred)
mae_test = mean_absolute_error(y_test, y_pred)
r2score_test = r2_score(y_test, y_pred)
print(f"R2 score: {r2score_test:.2f}")
print(f"MAE: {mae_test:.2f}")
print(f"Mean squared error: {mse_test:.2f}")
# Importing modules for Hyper Parameter Optimization using Grid SearchCV
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import r2_score
# Building and Training the Model
hyperparameters = {
    'fit_intercept': [True, False],
    'copy_X': [True, False],
}
gridsearch = GridSearchCV(lnr, hyperparameters, cv=5)
gridsearch.fit(x_train, y_train)
msehyptrain = mean_squared_error(y_train, y_pred_train)
maehyptrain = mean_absolute_error(y_train, y_pred_train)
r2scorehyptrain = r2_score(y_train, y_pred_train)
print(f"R2 score: {r2scorehyptrain:.2f}")
print(f"MAE: {maehyptrain:.2f}")
print(f"Mean squared error: {msehyptrain:.2f}")
```

```
# best hyperparameters
print("Best hyperparameters: ", gridsearch.best_params_)
msehyptrain = mean_squared_error(y_train, y_pred_train)
maehyptrain = mean_absolute_error(y_train, y_pred_train)
r2scorehyptrain = r2_score(y_train, y_pred_train)
print(f"R2 score: {r2scorehyptrain:.2f}")
print(f"MAE: {maehyptrain:.2f}")
print(f"Mean squared error: {msehyptrain:.2f}")
# best hyperparameters
print("Best hyperparameters: ", gridsearch.best_params_)
# Summary of Linear Regression Model in form of table
prediction_columns = ["MODEL TYPE", "DATASET TYPE", "R-2 SCORE", "MEAN SQUARED ERROR", "MEAN ABSOLUTE ERROR"]
df_pred = {"MODEL TYPE" : ["SIMPLE Linear Regression MODEL", "SIMPLE Linear Regression MODEL", "Linear Regression MODEL af
ter HYPER PARAMETER OPTIMIZATION", "Linear Regression MODEL after HYPER PARAMETER OPTIMIZATION"],
           "DATASET TYPE" : ["Training Dataset", "Test Dataset", "Training Dataset", "Test Dataset"],
           "R-2 SCORE" : [r2score_train, r2score_test, r2scorehyptrain, r2scorehyptest],
           "MEAN SQUARED ERROR" : [mse_train, mse_test, msehyptrain, msehyptest],
           "MEAN ABSOLUTE ERROR" :[mae_train, mae_test, maehyptrain, maehyptest]
          }
df_predictions = pd.DataFrame (df_pred)
df_predictions

# Random Forest Model
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor(n_estimators=100, random_state=42)
rfr.fit(x_train, y_train)
y_pred_train = rfr.predict(x_train)
mse_train = mean_squared_error(y_train, y_pred_train)
mae_train = mean_absolute_error(y_train, y_pred_train)
r2score_train = r2_score(y_train, y_pred_train)
print(f"R2 score: {r2score_train:.2f}")
print(f"MAE: {mae_train:.2f}")
print(f"Mean squared error: {mse_train:.2f}")
# Model Evaluation
y_pred = rfr.predict(x_test)
mse_test = mean_squared_error(y_test, y_pred)
mae_test = mean_absolute_error(y_test, y_pred)
r2score_test = r2_score(y_test, y_pred)
print(f"R2 score: {r2score_test:.2f}")
print(f"MAE: {mae_test:.2f}")
print(f"Mean squared error: {mse_test:.2f}")
#Hyper parameter Optimization
Citation: [12]Sociopath. "Random Forest Using GridSearchCV." Kaggle, 16 Oct. 2018, www.kaggle.com/code/sociopath00/random-
forest-using-gridsearchcv
parameters = {
    'n_estimators':[50, 100, 200],
    'max_depth':[3, 5, 7],
    'max_features':['sqrt', 'log2', None]
    }
grid_search = GridSearchCV(rfr, parameters, cv=5)
grid_search.fit(x_train, y_train)
y_pred_train = grid_search.predict(x_train)
mse_hyp_train = mean_squared_error(y_train, y_pred_train)
mae_hyp_train = mean_absolute_error(y_train, y_pred_train)
r2score_hyp_train = r2_score(y_train, y_pred_train)
print(f"R2 score: {r2score_hyp_train:.2f}")
print(f"MAE: {mae_hyp_train:.2f}")
print(f"Mean squared error: {mse_hyp_train:.2f}")
# best hyperparameters
print("Best hyperparameters: ", grid_search.best_params_)
mse_hyp_test = mean_squared_error(y_test, y_pred)
mae_hyp_test = mean_absolute_error(y_test, y_pred)
r2score_hyp_test = r2_score(y_test, y_pred)
print(f"R2 score: {r2score_hyp_test:.2f}")
print(f"MAE: {mae_hyp_test:.2f}")
print(f"Mean squared error: {mse_hyp_test:.2f}")
print("Best hyperparameters: ", grid_search.best_params_)
# Summary of Random Forest Model in form of table
prediction_columns = ["MODEL TYPE", "DATASET TYPE", "R-2 SCORE", "MEAN SQUARED ERROR", "MEAN ABSOLUTE ERROR"]
df_pred = {"MODEL TYPE" : ["SIMPLE RFR MODEL", "SIMPLE RFR MODEL", "RFR MODEL after HYPER PARAMETER OPTIMIZATION", "RFR MO
DEL after HYPER PARAMETER OPTIMIZATION"],
           "DATASET TYPE" : ["Training Dataset", "Test Dataset", "Training Dataset", "Test Dataset"],
           "R-2 SCORE" : [r2score_train, r2score_test, r2score_hyp_train, r2score_hyp_test],
           "MEAN SQUARED ERROR" : [mse_train, mse_test, mse_hyp_train, mse_hyp_test],
           "MEAN ABSOLUTE ERROR" :[mae_train, mae_test, mae_hyp_train, mae_hyp_test]
          }
df_predictions = pd.DataFrame (df_pred)

# ANN model
#Importing Required Libraries for ANN
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.wrappers.scikit_learn import KerasRegressor
ann_model1 = Sequential()
ann_model1.add(Dense(16, input_dim=X.shape[1], activation='relu'))
ann_model1.add(Dense(8, activation='relu'))
ann_model1.add(Dense(1))
ann_model1.compile(loss='mean_squared_error', optimizer='adam')
ann_model1.fit(x_train, y_train, epochs=25, batch_size=5)
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
mse_neural_train_ann1 = ann_model1.evaluate(x_train, y_train)
```

```
y_pred_train_ann1 = ann_model1.predict(x_train)
r2_train_ann1 = r2_score(y_train, y_pred_train_ann1)
mae_neural_train_ann1 = mean_absolute_error(y_train, y_pred_train_ann1)
print('Mean squared error from neural net: ', mse_neural_train_ann1)
print('Mean absolute error from neural net: ', mae_neural_train_ann1)
print('Accuracy score for neural net: ', r2_train_ann1)
# Model Evaluation
mse_neural_test_ann1 = ann_model1.evaluate(x_test, y_test)
y_pred_test_ann1 = ann_model1.predict(x_test)
r2_test_ann1 = r2_score(y_test, y_pred_test_ann1)
mae_neural_test_ann1 = mean_absolute_error(y_test, y_pred_test_ann1)
print('Mean squared error from neural net: ', mse_neural_test_ann1)
print('Mean absolute error from neural net: ', mae_neural_test_ann1)
print('Accuracy score for neural net: ', r2_test_ann1)

Citation: [14]Aaryandhore. "Neural Network + GridSearchCV Explanations." Kaggle, 13 Apr. 2020, www.kaggle.com/code/aaryandhore/neural-network-gridsearchcv-explanations
# Hyper Parameter Optimization using Grid SearchCV
def create_model(hidden_layers=1, neurons=32, dropout_rate=0.0, learning_rate=0.001, l2=0.0):
    model_ann2 = Sequential()
    model_ann2.add(Dense(neurons, input_dim=x_train.shape[1], activation='relu', kernel_regularizer=keras.regularizers.l2(
l2)))
    for i in range(hidden_layers-1):
        model_ann2.add(Dense(neurons, activation='relu', kernel_regularizer=keras.regularizers.l2(l2)))
        if dropout_rate > 0:
            model.add(Dropout(dropout_rate))
    model_ann2.add(Dense(1))
    optimizer = keras.optimizers.Adam(lr=learning_rate)
    model_ann2.compile(loss='mean_squared_error', optimizer=optimizer)
    return model_ann2
model_ann2 = KerasRegressor(build_fn=create_model, verbose=0)
param_grid = {
    'hidden_layers': [1, 2, 3],
    'neurons': [16, 32, 64],
    'dropout_rate': [0.0, 0.2, 0.4],
    'learning_rate': [0.001, 0.01, 0.1],
    'l2': [0.0, 0.001, 0.01]
}
grid = GridSearchCV(estimator=model_ann2, param_grid=param_grid, cv=KFold(n_splits=5), scoring='neg_mean_squared_error')
grid_result = grid.fit(x_train, y_train)
# Printing the best hyperparameters and performance
print("Best model parameters: ", grid_result.best_params_)
model_ann2 = create_model(**grid_result.best_params_)
model_ann2.fit(x_train, y_train, epochs=100, batch_size=30, verbose=0)
mse_neural_train_ann2 = model_ann2.evaluate(x_train, y_train)
y_pred_train_ann2 = model_ann2.predict(x_train)
r2_train_ann2 = r2_score(y_train, y_pred_train_ann2)
mae_neural_train_ann2 = mean_absolute_error(y_train, y_pred_train_ann2)
print('Mean squared error from neural net: ', mse_neural_train_ann2)
print('Mean absolute error from neural net: ', mae_neural_train_ann2)
print('Accuracy score for neural net: ', r2_train_ann2)
# Model Evaluation
mse_neural_test_ann2 = model_ann2.evaluate(x_test, y_test)
y_pred_test_ann2 = model_ann2.predict(x_test)
r2_test_ann2 = r2_score(y_test, y_pred_test_ann2)
mae_neural_test_ann2 = mean_absolute_error(y_test, y_pred_test_ann2)
print('Mean squared error from neural net: ', mse_neural_test_ann2)
print('Mean absolute error from neural net: ', mae_neural_test_ann2)
print('Accuracy score for neural net: ', r2_test_ann2)
# Summary of ANN in form of table
prediction_columns = ["MODEL TYPE", "DATASET TYPE", "R-2 SCORE", "MEAN SQUARED ERROR", "MEAN ABSOLUTE ERROR"]
df_pred = {"MODEL TYPE" : ["SIMPLE ANN MODEL", "SIMPLE ANN MODEL", "ANN MODEL after HYPER PARAMETER OPTIMIZATION", "ANN MO
DEL after HYPER PARAMETER OPTIMIZATION"],
           "DATASET TYPE" : ["Training Dataset", "Test Dataset", "Training Dataset", "Test Dataset"],
           "R-2 SCORE" : [r2_train_ann1, r2_test_ann1, r2_train_ann2, r2_test_ann2],
           "MEAN SQUARED ERROR" : [mse_neural_train_ann1, mse_neural_test_ann1, mse_neural_train_ann2, mse_neural_test_ann
2],
           "MEAN ABSOLUTE ERROR" :[mae_neural_train_ann1, mae_neural_test_ann1, mae_neural_train_ann2, mae_neural_test_ann
2]
         }
df_predictions = pd.DataFrame (df_pred)
df_predictions
# Comparision of the performance of models based on R2 Score
Name = ['Linear Regression', 'Random Forest', 'ANN']
Score = [r2scorehyptest, r2score_hyp_test, r2_test_ann2]
# Plotting bar graphs for comparision
axis = sns.barplot(x = Name , y = Score, data = data)
axis.set(xlabel='Classifier', ylabel='Accuracy')
for p in axis.patches:
    height = p.get_height()
    axis.text(p.get_x() + p.get_width()/2, height + 0.005, '{:1.4f}'.format(height), ha="center")
plt.show()
# Comparision of the performance of models based on MSE
MSE = [msehyptest, mse_hyp_test, mse_neural_test_ann2]
# Plotting bar graphs for comparision
axis = sns.barplot(x = Name , y = MSE, data = data)
axis.set(xlabel='Classifier', ylabel='MSE')

for p in axis.patches:
    height = p.get_height()
    axis.text(p.get_x() + p.get_width()/2, height + 0.005, '{:1.4f}'.format(height), ha="center")
plt.show()
```