**IBM AI : MEASURE ENERGY CONSUMPTION**

**Mohan Krishna.N (**[**mohankrish3027@gmail.com**](mailto:mohankrish3027@gmail.com)**)**

**Prakash.A (**[**prakashmokesh2004@gmail.com**](mailto:prakashmokesh2004@gmail.com)**)**

**Raguram.R (**[**raguramrramanan@gmail.com**](mailto:raguramrramanan@gmail.com)**)**

**Reegan Anto.J (**[**reegananto21@gmail.com**](mailto:reegananto21@gmail.com)**)**

**Problem Definition:**

**The constant improvement and industrialization causes the rise in energy consumption day by day and in order to keep the economy, environment and resources under check it is not only sufficient to evaluate current energy expenditure but also to predict the future energy consumptions.**

**Being able to calculate future energy expenses will be helpful for small and large companies alike, and is also essential to estimate the growth of our nation.**

**Our plan is to predict the future energy consumption based on past data using one of many machine learning algorithms like ARIMA, SARIMA, LSTM, XGboost, etc. Applying whatever provides the best accuracy for the given dataset and to automate the project such that it updates itself in order to make itself more relevant even in the future.**

**Design Thinking:**

1. **Data Source: Our primary dataset is taken from** [**https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption**](https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption) **, we are planning to further append data updates to our dataset.**
2. **Data Preprocessing: We can use techniques such as EDA, Bayesian method, classical data analysis or any other modern data analysis technique in order to clean, filter, transform and prepare our dataset.**
3. **Feature Extraction: We can extract the various trends and patterns which are formed in our dataset and extract the relevant metrics in order to improve the efficiency of our model.**
4. **Model Development:We will analyse the data statistically and using z-score and IQR analysis we can remove the outliers found in the dataset.**
5. **Visualization: Using various visualization tools such as matplotlib, power Bi, etc we could visualize the data in order to find furthermore trends and patterns making it easier to verify, modify or present the obtained model.**
6. **Automation: Using techniques like data warehousing and web scraping we could continue adding dataset to our model in order to improve its accuracy further automatically.**

**Project Goal:**

**The goal of our project is to predict the future energy consumption based on the hourly energy consumptions from 2004 to 2018.**

**Approach:**

**First we need to investigate the given dataset; the dataset contains what appears to be various cities with the amount of electrical energy consumed alongside it.**

**Our initial plan was to build a linear regression model around the dataset, but after visualization we concluded that linear regression wouldn’t suit the given dataset.**

**Our second approach was to find any form of seasonal growth or trends within the dataset, and we seem to find some seasonal pattern within the dataset.**

**We concluded that univariate time series analysis would be the best method to solve the problem.**

**Analysis:**

**We then started to analyze the best algorithm within the time series analysis which would suit the given dataset perfectly.**

**The algorithms we compared are listed below:**

* **Seasonal Auto regressive Integrated Moving Average (SARIMA)**
* **Single Exponential Smoothing (SES)**
* **Prophet**
* **XGBoost**
* **LSTM**

**Our analysis on each machine (and deep) learning algorithms are given below:**

**Seasonal Auto Regressive Integrated Moving Average:**

**SARIMA is an improved version of ARIMA which adds seasonal moving average to the previous autoregressive model which is vital for our given time series dataset. The additional hyperparameters would aid us in predicting the measure of electrical energy.**

**Single Exponential Smoothing:**

**This technique uses the weighted sum of all values obtained below to predict the approximate future value. This technique also applies exponentially decreasing weights to prevent exploding gradient problem. The ability to control the alpha value of the problem would be useful to tune the model based on what timeline the required model should focus on thus tuning its time-accuracy combination.**

**Prophet:**

**This is an open-source library developed by meta for machine learning on the python programming language. The automated forecasting system can reduce the need for human interference but would make it hard to debug.**

**LSTM:**

**Long Short Term Memory deep learning algorithm has been rejected as we don’t have as much knowledge on memory bounded algorithms compared to other machine learning algorithms.**

**XGBoost:**

**This algorithm contains gradient boosted trees in order to quicky and efficiently calculate tabular data. The high popularity within data science contests, speed and the amount of online sources to learn the working of this algorithm has made us choose this algorithm over all others mentioned before.**

**Plan:**

**The algorithm we are going to be using here is Extreme Gradient Boosting (XGBoost).Our plan is to split 80% of the first half of the dataset as training dataset and the remainder 20% of the dataset as testing dataset in order to train and verify the model.Currently the result and accuracy of the model can’t be estimated but we aim to obtain at least 70% accuracy.**

**Data Visualization:**

**Data visualization is an integral part of machine learning, and it is also necessary to guide us to come up with various interpretations of the obtained data.**

**The visualized data is split into 80% training data(blue) and 20% testing data(orange).**

**The charts given below describe my partitions.**

**(Note: The name of the city is represented in the legend of the charts)**

**Insights Obtained from Data Visualization:**

**From our visualization process we noticed that the charts don’t seem to show any gradual increase even when the number of electrical appliances has increased per home.**

**After analyzing various generations of electrical appliances and their usage we noticed that as time passes even though the number of electrical appliances increases, and so does its efficiency and due to which the graph shows a steady pace instead of gradually rising.**

**But this is not enough to draw any solid conclusions, hence we must opt for different forms of analysis to obtain some valuable insights.**

**Feature Extraction:**

**The little information we gained from visualization is not sufficient to draw many conclusions, so we opt to seaborn library to clearly understand more trends beyond the seasonal ones, the results from feature extraction are shared below.**

**Insights Obtained from Feature Extraction:**

**From the feature extraction process of the given dataset, we have noticed that the amount of energy used by each city follows a seasonal pattern. The electricity usage is extremely high around the summer, which must be due to the extensive usage of air conditioners.**

**From that we can conclude that the data follows a seasonal pattern.**

**DEVELOPMENT:**

**Import pandas as pd**

**Import numpy as np**

**Import matplotlib.pyplot as plt**

**Import seaborn as sns**

**Import xgboost as xgb**

**From sklearn.metrics import mean\_squared\_error**

**Def create\_features(df):**

**Df = df.copy()**

**Df[‘hour’] = df.index.hour**

**Df[‘dayofweek’] = df.index.dayofweek**

**Df[‘quarter’] = df.index.quarter**

**Df[‘month’] = df.index.month**

**Df[‘year’] = df.index.year**

**Df[‘dayofyear’] = df.index.dayofyear**

**Df[‘dayofmonth’] = df.index.day**

**Df[‘weekofyear’] = df.index.isocalendar().week**

**Return df**

**Def process(fileHead, fileName, date):**

**Df = pd.read\_csv(fileName)**

**Df = df.set\_index(‘Datetime’)**

**Df.index = pd.to\_datetime(df.index)**

**Train = df.loc[df.index < date]**

**Test = df.loc[df.index >= date]**

**Df = create\_features(df)**

**Train = create\_features(train)**

**Test = create\_features(test)**

**Features = [‘dayofyear’, ‘hour’, ‘dayofweek’, ‘quarter’, ‘month’, ‘year’]**

**Target = fileHead**

**X\_train = train[features]**

**Y\_train = train[target]**

**X\_test = test[features]**

**Y\_test = test[target]**

**Reg = xgb.XGBRegressor(base\_score=0.5, booster=’gbtree’, n\_estimators=1000, early\_stopping\_rounds=50, objective=’reg:linear’, max\_depth=3, learning\_rate=0.01)**

**Reg.fit(x\_train, y\_train, eval\_set=[(x\_train, y\_train), (x\_test, y\_test)], verbose=100)**

**Test[‘prediction’] = reg.predict(x\_test)**

**Score = np.sqrt(mean\_squared\_error(test[fileHead], test[‘prediction’]))**

**Print(f’RMSE Score on Test set: {score:0.2f}’)**

**Test[‘error’] = np.abs(test[target] – test[‘prediction’])**

**Test[‘date’] = test.index.date**

**Test.groupby([‘date’])[‘error’].mean().sort\_values(ascending=False).head(10)**

**If \_\_name\_\_ == ‘\_\_main\_\_’:**

**File\_header = [‘AEP\_MW’,’COMED\_MW’,’DAYTON\_MW’,’DEOK\_MW’,’DOM\_MW’,’DUQ\_MW’,’EKPC\_MW’,’FE\_MW’,’NI\_MW’]**

**Data = [[‘../IBM Project/Dataset/AEP\_hourly.csv’,’02-01-2017’],**

**[‘../IBM Project/Dataset/COMED\_hourly.csv’,’08-01-2017’],**

**[‘../IBM Project/Dataset/DAYTON\_hourly.csv’,’10-01-16’],**

**[‘../IBM Project/Dataset/DEOK\_hourly.csv’,’06-01-17’],**

**[‘../IBM Project/Dataset/DOM\_hourly.csv’,’01-01-17’],**

**[‘../IBM Project/Dataset/DUQ\_hourly.csv’,’06-01-16’],**

**[‘../IBM Project/Dataset/EKPC\_hourly.csv’,’08-01-17’],**

**[‘../IBM Project/Dataset/FE\_hourly.csv’,’02-01-17’],**

**[‘../IBM Project/Dataset/NI\_hourly.csv’,’07-01-09’]]**

**Print(“1.AEP\n2.COMED\n3.DAYTON\n4.DEOK\n5.DOM\n6.DUQ\n7.EKPC\n8.FE\n9.NI\n”)**

**N = int(input())**

**If(n in range(1,10)):**

**Process(file\_header[n-1],data[n-1][0],data[n-1][1])**

**Else:**

**Print(“Invalid Input!”)**

**Conclusion:**

**The algorithm we are going to be using is decided to be XGBoost, the dataset will be split as per the industry standard 80% training and 20% testing and we hope to obtain an estimated average accuracy rating of at least 20% when our project has been completed.**