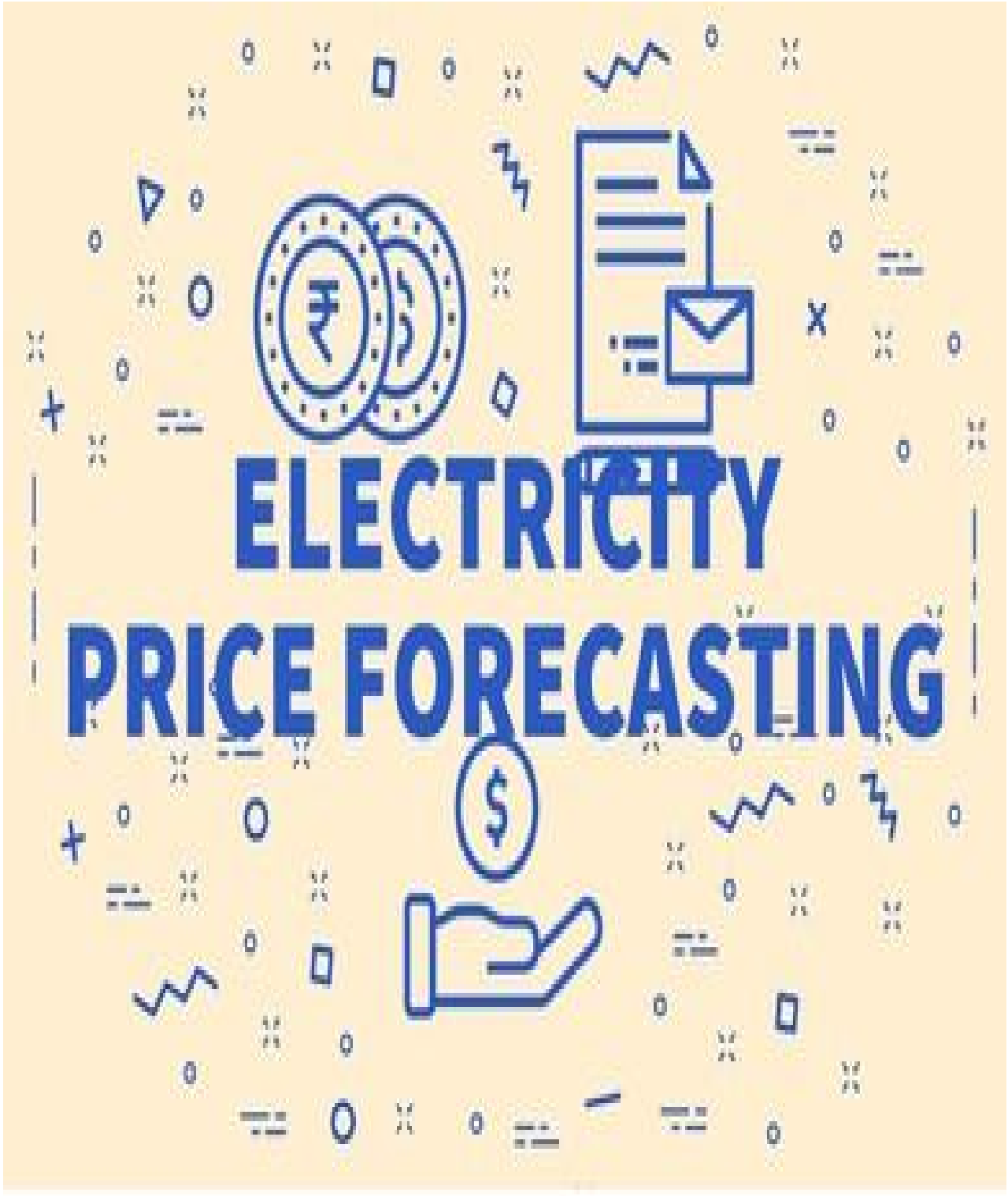


PHASE:3 PROJECT

ELECTRICITY PRICE PREDICTION

PROJECT SUBMISSION ON:01/11/23

SUMMITED BY,

1. **R.PRIYASHAKHI**
2. **S.SUBALAKSHMI**
3. **M.MAHALAKSHMI**
4. **A.KASTHOORI**
5. **G.KRISHNAVENI**

COLLEGE NAME:SACS MAVMM ENGINEERING COLLEGE

COLLEGE CODE:9123

ELECTRICITY PRICE PREDICTION

INTRODUCTION:

The electric daily peak load is the maximum of the electricity power demand curve over one day. Having an accurate forecast of the daily peak enables independent system operators (ISOs) and energy providers to better deliver electricity and optimise power plant schedules. The importance of such a forecast is increasing as the integration of intermittent renewable production sources progresses. In particular, renewable energy sources are at the bottom of the merit order curve, which makes them (currently) the most economical source of energy used to serve the market. However, they are intermittent and provide time-varying levels of power generation that are only partially under human control. If electricity demand is high and renewables cannot provide for it alone, ISOs must deliver electricity from sources with higher marginal costs (e.g. gas-fired plants) for stakeholders as well as for the environment in terms of carbon dioxide emissions. In such a context, accurately forecasting the peak demand magnitude and timing is essential for determining the generation capacity that must be held in reserve.

Electrical equipment is tailored to support a specific peak load. If the demand comes close to or exceeds the network capacity, it can lead to distribution inefficiencies and ultimately power system failures, such as blackouts. With the

increasing number of electric vehicles (EVs) in circulation, a further source of stress is added to the electricity system. For instance, 46% of vehicles sold in Norway in 2019 were EVs (International Energy Agency, 2019). The challenge posed by the additional EV demand must be met by more tailored management systems and policies, if expensive infrastructural works are to be avoided. Dynamic electricity pricing schemes, for example, the Triads in the UK or the Global Adjustment in Ontario, Canada, have been developed to reduce the system peak load. Consumers who can correctly estimate and cut their use during peak events can unlock great savings. Peak demand forecasts will thus be key for the development of such policies.

To account for the increasing demand for electricity and to prevent system failures, smart grid technologies and policies are being implemented to foster communication between the various stakeholders in the electricity supply chain to achieve a more efficient use of energy. One major objective is to maximise the load factor. The load factor is the average load over a specific time period divided by the peak load over the same period. Maximising it leads to a more even use of energy over time, thus preventing system failures and surges in electricity prices. One of the most common ways to achieve load factor maximisation is peak shaving, which refers to the flattening of electrical load peaks. Three major strategies have been proposed for peak shaving, namely integration of the energy storage system (ESS), vehicle-to-grid (V2G) integration, and demand-side management (DSM) (Uddin, Romlie, Abdullah, Abd Halim,

Abu Bakar, & Chia Kwang, 2018). ESS and V2G integration provide ancillary sources to balance the grid through batteries while DSM shifts consumer demand to flatten the peak. To be activated adequately, all these strategies require accurate forecasts of the demand peak magnitude (DP) and of the instant at which it occurs (IP).

This article proposes novel methods to forecast the DP and the IP by leveraging information at different time resolutions. In particular, the multi-resolution approach proposed here is illustrated in the context of two model classes: generalised additive models (GAMs) and neural networks (NNs). Both are state-of-the-art predictive models, widely used to forecast electrical load in industry and academia. The performance of the multi-resolution framework under both model classes is assessed using aggregate UK electricity demand data from the National Grid (National Grid, 2021).

The rest of the paper is structured as follows: Section 2 presents a literature review of daily peak forecasting methodologies. Section 3 introduces multi-resolution modelling using GAMs and NNs. Section 4 explains how the different models were setup in the high-resolution, low-resolution, and multi-resolution settings. Section 5 analyses the results of the models described in Section 4, using UK demand data.

PHASES OF DEVELOPMENT:

1. **General Framework for the Development of Price Forecasting Method Forecasting of electricity prices is generally divided into short, medium and long term [13] categories. Nonetheless, there is no particular boundary line in the literature to distinguish them. Generally,**
    - **Short-term: This significant subcategory is the most relevant for daily market operations, and the forecast time varies from a few or several minutes to several days upwards.**
    - **Medium-term: The establishment of balancing sheet estimation includes medium electricity price forecasting, such as the derivatives pricing (change structure), and strategies of**

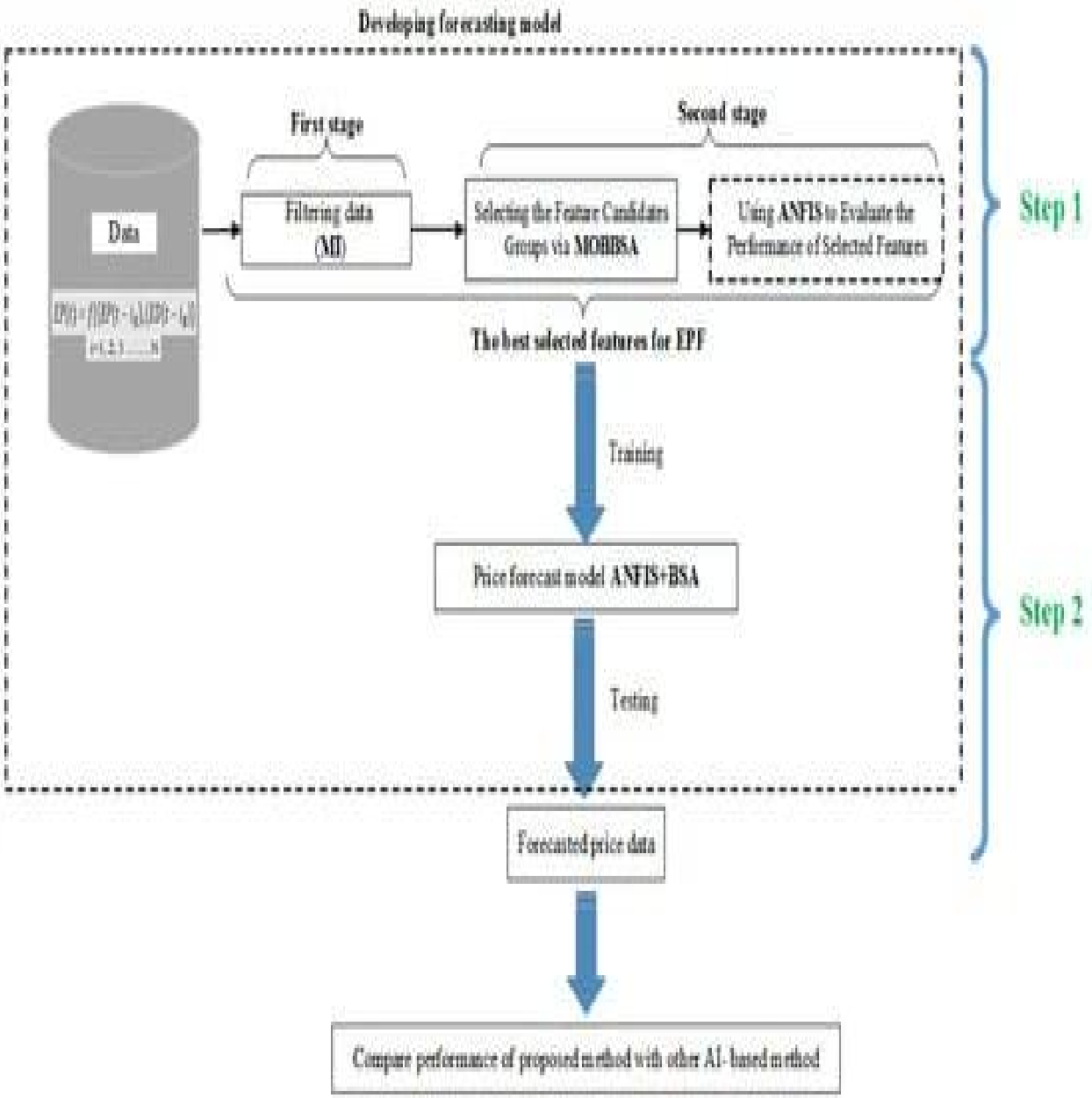
the risk management process, and the forecast duration starts from a few days to a few months after. The development in the forecast of electricity prices is generally based on the factor of price distribution on the future horizon rather than on the actual point predictions.

- **Long-term: The forecasting implementation of this scenario concentrates mostly on the preparation of profitability investment analysis and planning; the prediction duration is carried out for a month, quartile or even years in the future. This type of forecasting has generated useful information, which is appropriate to evaluate the potential site or generating facility-based fuel sources.**

Precise forecasting is a prerequisite for key players and decision-makers in the electricity market to develop an optimal strategy that includes the improvement of socio-economic benefits and risk reduction. Short-term forecasts attract substantial attention and are extensively utilized for economic dispatch and power system control in electricity markets. Therefore, removing impediments in the

short-term forecasting of electricity prices will play an instrumental role in managing power systems to meet the growing demand, keeping in line with economic growth that is imperative for sustainable development of different competitive electricity markets.

The proposed electricity price forecasting strategy is presented in Figure



1. After collecting data on historical prices and demand, it is required to prepare constrained data through significant feature selection. Therefore, in the first step, an enhanced feature selection is utilized via hybrid filtering and embedded techniques to assess the quality of features for the forecasting process. In the first stage, MI is applied to reduce the time of training due to the availability of data with a high dimension. In the next stage, MOBBSA is applied to select the features that represent the most important information of the original set.

In step 2, the robust forecasting technique, based on combined ANFIS-BSA, is designed for day-ahead price forecasting in the highly volatile Queensland market. In this study, two types of ANFIS are developed. During the feature selection, in order to evaluate the selected input variable for forecasting purposes, as well as during forecasting, the electricity price is used to improve the forecasting accuracy. In the second step, BSA and other well-known optimization techniques are utilized to tune the membership function parameters to improve the price forecasting accuracy

PREPROCESSING DATA:

This step is an important step in data mining process. Because it improves the quality of the experimental raw data.

- **Removal of Null values:**

  In this step, the null values in the fields Product Category2 and Product Category3 are filled with the mean value of the feature.

- **Converting Categorical values into numerical:**

  Machine learning deal with numerical values easily because of the machine readable form. Therefore, the categorical values like Product ID, Gender, Age and City Category are converted to numerical values.

  - **Step1: Based on its datatype, we have selected the categorical values.**
  - **Step2: By using python, we have converting the categorical values into numerical values.**

- **Separate the target variable:**

  Here, we have to separate the target feature in which we are going to predict.

  In this case, purchase is the target variable.

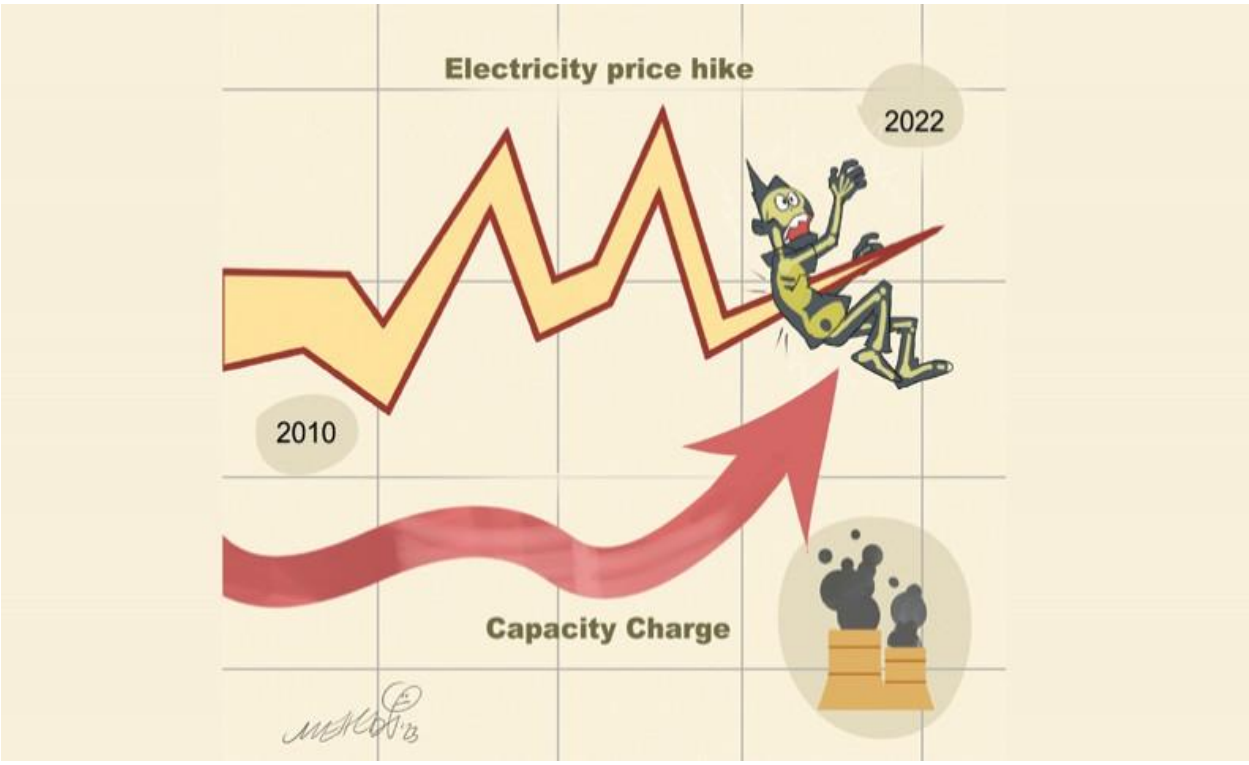
  - **Step1: The target lable purchase is assigned to the variable 'y'.**
  - **Step2: The preprocessed data except the target lable purchase is assigned to the variable 'X'.**

- **Standardize the features:**

  Here, we have to standardize the features because it arranges the data in a standard normal

distribution. The standardization of the data is made only for training data most of the time because any kind of transformation of the features only be fitted on the training data.

- **<u>Step1</u>: Only trained data was taken.**
- **<u>Step2</u>: By using the Standard Scaler API, we have standardize the features.**



<u>PREPROCESSING DATASET FOR ELECTRICITY PRICE PREDICTION:</u>

- **<u>Step1:</u> <u>Getting the Dataset</u>**

  Data set for Predicting Electricity Price

https://www.kaggle.com/datasets/chakradharmattapalli/electricity-price-prediction

- **<u>Step2:</u> <u>Importing Libraries</u>**

Import pandas

Import datetime

- **Step3:Importing Dataset of Electricity Priice Prediction** https://www.kaggle.com/datasets/chakradharmattapalli/electricity-price-prediction
- **Step4:Finding Missing Data**

Find missing Data in the Dataset of https://www.kaggle.com/datasets/chakradharmattapalli/electricity-price-prediction

- **Step5:Encoding Categorical Data**

Encoded Categorical Data in the Dataset of https://www.kaggle.com/datasets/chakradharmattapalli/electricity-price-prediction

- **Step6:Splitting Dataset into training set and test set**

The data set undergoes training and test set https://www.kaggle.com/datasets/chakradharmattapalli/electricity-price-prediction

ALGORITHM FOR PREDICTING ELECTRICITY PRICE:

- **Proposed Trust algorithm — general**

The Trust algorithm converts the feature importance details provided by local (SHAP) and global explanation (PFI and LR) methods into a simple trust score. We will first look at prediction error to understand how the explanation scores and their correlations can act as pointers to forecast quality. Errors in model prediction are attributed to two factors:
(i) input features that are highly dissimilar to those seen by the model during training (density principle); (ii) Failure of the model to learn some aspect of the stochastic process from the given training dataset (local fit principle). It is known that prediction error is given by
(15)
ε
From Eq. (15), it is seen that error vector
ε
is a function of the actual electricity price , input features , and the trained model . Electricity prices to be predicted are unknown quantities. However, the other two quantities are known. SHAP scores and their correlation with PFI and LR explanations are affected by and . This is explained in Sections

- **First stage of the proposed algorithm,**

Second stage of the proposed algorithm. Hence, these quantities have the underlying ability to estimate the quality of the prediction, i.e., the presence/absence of error in the said prediction. This is the basis of the proposed Trust algorithm.

- **The proposed algorithm is executed in two stages. It implements the local fit principle in the first stage and the density principle in both — the first and second stages. Section 3.2 explains the first stage of the proposed work. It also discusses the reasoning behind the use of correlations in global and local explanation techniques. Section 3.3 explains the second stage of the algorithm. It also elaborates on the rationale behind using SHAP scores to evaluate forecast quality.**

- **First stage of the proposed algorithm**

The first stage uses similarity instead of distance to estimate density. Based on correlation statistics, the algorithm finds the similarity of the given input with the training set. Since all the input features are individual time series with definitive patterns of seasonality, the correlation between the input and the training data from the same period is evaluated. These correlation tests would produce two statistics: correlation coefficient and -value. The correlation coefficient is statistically significant only if the -value is below 0.08, see [30]. The average of the obtained correlation coefficients quantifies density in terms of similarity. Appendix B explains the process of averaging correlation coefficients and cites the research related to it. A high and statistically significant correlation indicates that the input has data points similar to those in the training set.

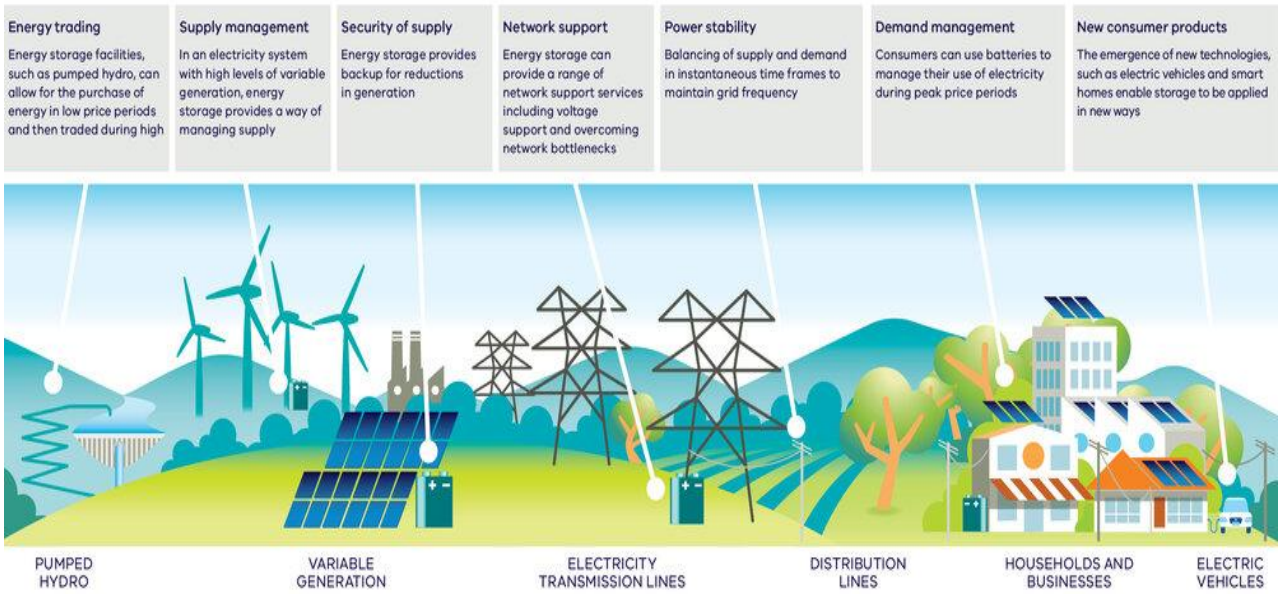
After this, the local fit principle is implemented by evaluating the correlation of SHAP scores for the predicted period with the following quantities:

(i)SHAP scores for the same period from the previous year;

(ii)PFI scores;

(iii)LR scores.

- **Rationale behind using correlations: Correlations can help identify whether or not the model predictions align with the process traits observed in the training data.**
- **To understand this, let us consider two scenarios. In the first scenario, assume that a trained model, , has learned all that it should from the training dataset of the stochastic process.**
- **For the second scenario, consider another model trained on the same data, , that has failed to learn a specific process trait.**



- **Now consider the PFI and SHAP scores for a moment. It is deduced from (4), (A.1), that the PFI scores are a function of the training set and the trained model . It is mathematically represented as(16)**
- **It shows that PFI scores learn about the stochastic process from the training dataset, which consists of both the actual output of the process and the response of the trained model. On the other hand, from (5), (A.2), and (A.3), it is seen that SHAP scores are a function of the input vector and the trained model , and mathematically can be given by(17)**

So, it can be said in general that the explanation function of SHAP has not learned anything additional about the stochastic process that the PFI has not already captured.

- **In the first scenario, we assumed that the trained model has learned the process traits. So, the explanations of PFI and SHAP will have a similar ordering of the feature ranks and inter-distance score values. As discussed in [31], this would result in a strong correlation between PFI and SHAP scores, i.e., the Pearson correlation coefficient, will be closer to 1. Moreover, its -value will be less than the minimum threshold value, making it statistically significant. The correlation coefficient is given by(18)**

- **On the other hand, in the second scenario, we assumed that the trained model has failed to learn a specific process trait. In this case, the PFI explanation may reflect additional process traits because of training/test output values considered while evaluating PFI scores. However, the SHAP explanation function will only learn those process traits that**

are reflected from the input and the trained model. The actual stochastic output includes the impact of the missing attribute, which is not known from the SHAP explanation. So the correlation between PFI and SHAP scores, represented by (18), will be weak. In other words, either the Pearson correlation coefficient will be closer to 0, or its -value will be higher than the minimum threshold value, or both. Thus, we can conclude from the two scenarios that the correlation coefficients can capture whether the trained model is accurate for a given input or not. Here, the underlying assumption for both scenarios is that the input vector is within the training data space, i.e., IID in nature.

- **Implementation steps of the first stage: The first stage compares the local SHAP scores of the ML prediction with the PFI, LR, and SHAP scores of last year through correlations. While PFI and LR scores look at the model globally, SHAP scores from the previous year look at the scores locally. Obtained correlations would throw light on whether or not the trained model is close to the trained model's accuracy for IID data for the given input at that instance. The mathematical expression for the correlation between SHAP scores of the predicted region and those of the same period last year are given by(19) where represents the SHAP scores for the previous year for the same period as the predicted period. Similarly, the expression for the correlation between SHAP and LR scores is given by(20)**

- **Expression for the correlation between input vector and the data from training set for the same period is given by(21)**
- **Since these correlations were evaluated for a similar period last year, there will be a vector of correlations. As discussed earlier, these correlations need to be averaged, as discussed in Appendix B. The -values obtained during correlation statistics between the two SHAP scores, SHAP & PFI scores, SHAP & LR scores, and the input of predicted region and training set of same period are represented by , , , and , respectively.**

- **Coarse form of Trust Score: The first stage produces the coarse value of the trust score . If then(22)**
- **If both and then takes the value of whichever is greater between and . If both correlations and are statistically insignificant and ,(23)**
- **On the other hand, if or the correlation is statistically insignificant then all the above conditions of hold true except for . This is so because there is no/less correlation between input and data from training set in the same period and hence correlations among local SHAP explanations are not applicable. It is to be noted that the algorithm obtains all correlation coefficients and -values using Pearson correlation. The choice of Pearson correlation is based on the model performance experience with the previous year's data behavior. For different data, the nature of the correlation could change. Accordingly, one may have to resort to Spearman or Kendall correlation.**

- **Second stage of the proposed algorithm**

The second stage of the proposed algorithm fine-tunes , to form the final trust score . It implements the density principle through SHAP scores. SHAP scores of the predicted quantity help with fine-tuning using two characteristics of SHAP values: (i) the maximum SHAP value among all the features , and (ii) the sum of all SHAP values . is modified based on the comparison of and with . is the threshold limit for SHAP values for all instances, in general.

- **Rationale behind using SHAP values: The question is how the SHAP scores can act as pointers to forecast quality. To understand this, we consider two scenarios. The first is**

wherein the input features , are similar to those used for training the model. The other is when input features are highly dissimilar. This similarity/dissimilarity could be either in the magnitude of individual features, the inter-relationship among features, or both.

- **Generally, the difference between the terms and , seen in (5), will be lower in the first scenario (similar input features) as compared to the second (dissimilar input features). To understand this statement, we use the example taken up in (6), (7) wherein the marginal contribution of feature was to be estimated. Consider that the randomly sampled background values of feature , are . Assume that in the first scenario, the value of . It is in the range of the background samples taken from the training set for the feature . Similarly, assume that in the second scenario, . Here, is a very high magnitude positive or negative value, which makes a value far from the range of background samples considered for .**

- **Accordingly, in the first scenario, and are as shown in (24), (25), respectively:(24)(25)**
- **Here, could be either any one of the values from or , the mean of all values, as shown by(26)**
- **For the first scenario, when the input vector is similar to the data from training set, the feature value is close to . Hence, from (24), (25), and (26), it is seen that and will be similar. So, the difference between the terms and is low. Therefore, the magnitude of the respective SHAP score, given by(27)will be low.**

- **Conversely in the second scenario, is given by(28) while is given by(29)**
- **Here, could be either any one of the values from , or the mean of all values, , as shown by(30)**
- **From Eqs. (28), (29), and (30), it can be seen that the vectors and are very dissimilar due to their respective feature**

values and . The only exception to this is when itself. Hence, the difference between the terms and will be high. Therefore, generally, the SHAP score for such dissimilar input features, given by(31)will be high.From the description of Eqs. (27), (31), it is deduced that(32)

- **Based on this finding, the second stage of the proposed algorithm uses SHAP scores to distinguish between inputs that are (dis)similar to the training set. This, in turn, points to whether the predictions can be trusted or not. It is to be noted that the difference in absolute values of SHAP scores is substantial for the two scenarios only if does not take up the value . It is one of the reasons why the proposed Trust score may fail in some instances.**

1. **Trust Score: Algorithm 1 shows the pseudo-code for the flow of the proposed Trust Algorithm. Considering the forecast horizon as , the forecast step begins at one and is incremented in unitary steps until the algorithm reaches , with each forecast step acting as an iteration. In each iteration, the algorithm compares the trust score with the Trust threshold . The prediction is good if the trust score is equal to or greater than the Trust threshold. Conversely, suppose the trust score is less than the Trust threshold. In that case, the forecast will likely have a larger error than MAE/MAPE (depending on the comparison considered for decision-making), meaning it is a dire prediction. The limiting values and are decided using experience with the prediction error and its respective SHAP scores from the training/testing set.**

2. Overall flow of the proposed Trust algorithm

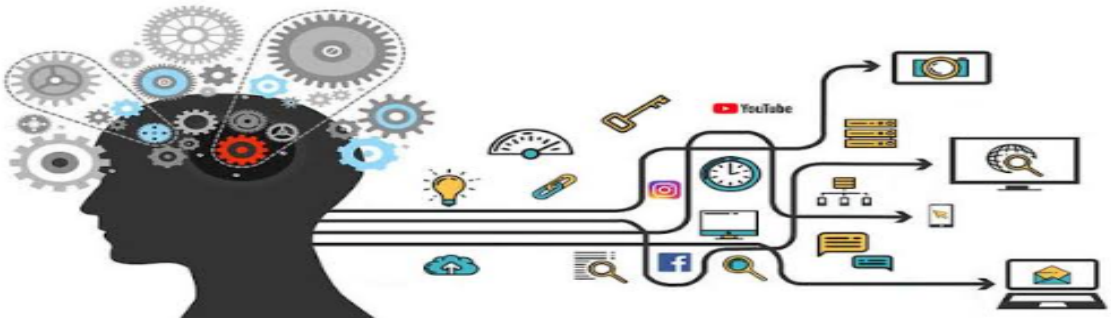
Algorithm 2 shows the pseudo-code for the overall algorithm of the proposed approach. Considering the forecast horizon as , the forecast step begins at one and is incremented in unitary steps until the algorithm reaches with each forecast step acting as an iteration. In each iteration, the algorithm compares the trust score, , of each output element, , with the Trust threshold . The prediction is good if the trust score is equal to or greater than the Trust threshold. Conversely, suppose the trust score is less than the Trust threshold. In that case, the forecast will more likely have an error greater than MAE/MAPE (depending on the comparison considered for decision-making), meaning it is a dire prediction. It is to be noted that the algorithm is sensitive.

ENHANCED MODEL PERFORMANCE:

Well-engineered features can capture underlying patterns and relationships in the data, leading to more accurate predictions. By selecting the right features and transforming them appropriately, you can extract valuable information that might be hidden in the raw data.

IMPROVED INTERPRETABILITY:

Feature engineering can make your models more interpretable. By creating meaningful features, you can gain insights into which factors are driving demand and how they impact your predictions. This knowledge is invaluable for making informed business decisions.



HANDLING NON-LINEARITY:

Real-world demand data is often non-linear, and feature engineering allows you to transform variables to better fit the assumptions of your chosen machine learning algorithm. This can result in better model performance.

IMPROVED INTERPRETABILITY:

Feature engineering can make your models more interpretable. By creating meaningful features, you can gain insights into which factors are driving demand and how they impact your predictions. This knowledge is invaluable for making informed business decisions.

HANDLING NON-LINEARITY:

Real-world demand data is often non-linear, and feature engineering allows you to transform variables to better fit the assumptions of your chosen machine learning algorithm. This can result in better model performance.

FEATURE ENGINEERING:

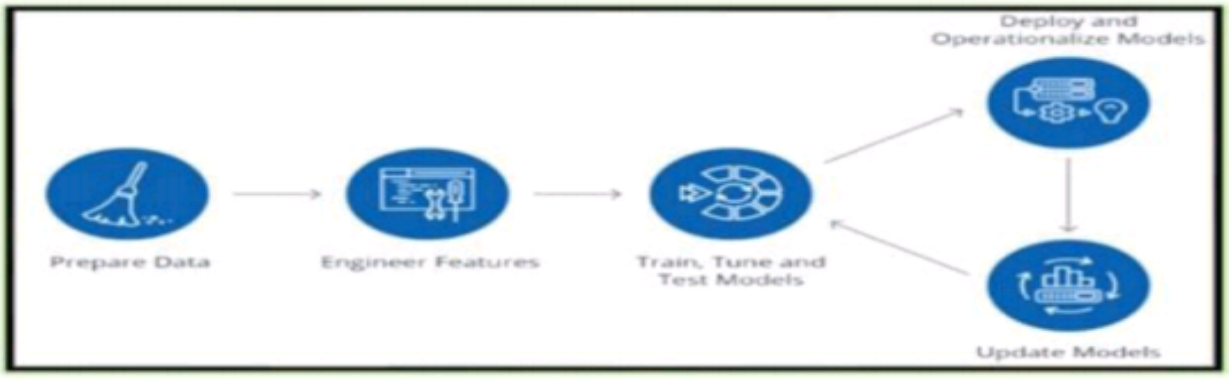
Feature engineering is the process of selecting and transforming relevant features from the raw data to improve the performance of ML models. In demand prediction for drugs on pharmacies, some of the most important features are:

TIME-BASED FEATURES:

These features capture trends and patterns over time. Examples include day of the week, month, year, and holidays.

STORE-BASED FEATURES:

These features capture pharmacy-specific characteristics. Examples include the location of the pharmacy, the size of the pharmacy, and the customer demographics.
In addition to these features, external factors such as economic conditions, weather conditions, and cultural event can also be taken into account.



A typical approach to feature engineering in future sales prediction forecasting involves the following types of features:

LAGGED VARIABLES:

By incorporating previous time series values as features, patterns such as seasonality and trends can be captured. For example, if we want to predict today's sales, using lagged variables like yesterday's sales can provide valuable information about the ongoing trend.

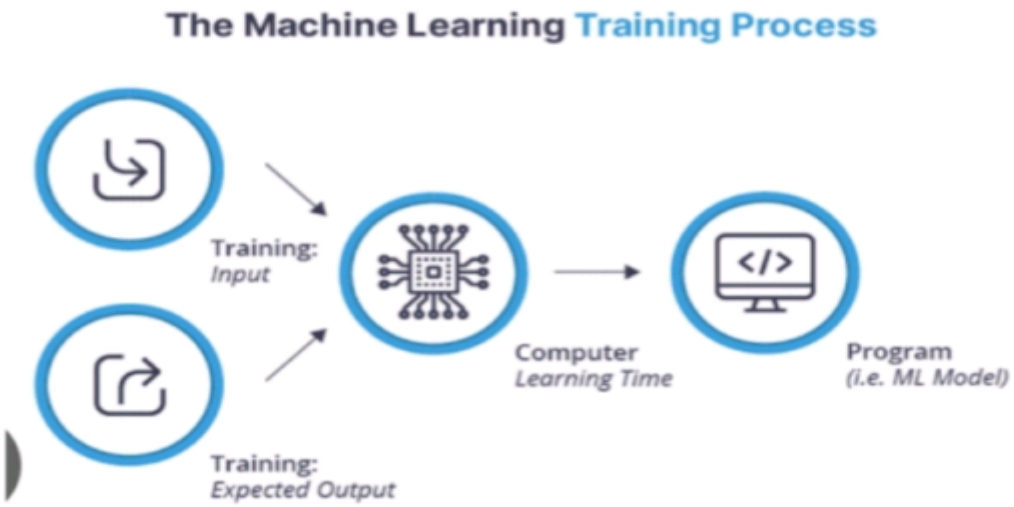
MOVING WINDOW STATISTICS:

This involves aggregating the time series values over a rolling window. By doing so, noise is smoothed out, shifting the focus to the underlying trends. Moving windows can help identify patterns that may not be immediately apparent in the raw data.

TIME-BASED FEATURES:

Such as the day of the week, the month of the year, holiday indicators, seasonality, and other time. related patterns can be valuable for predictions. For instance, if certain products tend to have higher.

To train a model using the given dataset, you can follow these general steps:



1.DATA EXPLORATION:

Start by exploring the dataset to understand its structure, features, and target variable. This will help you gain insights into the data and make informed decisions during the modeling process.

2.DATA PREPROCESSING:

Clean and preprocess the data to handle missing values, outliers, and categorical variables. This may involve techniques such as imputation, scaling, encoding, etc.

3. FEATURE ENGINEERING:

Create new features or transform existing ones to improve the predictive power of your model. This can include techniques like one-hot encoding, feature scaling, dimensionality reduction, etc.

4. MODEL SELECTION:

Choose an appropriate machine learning algorithm based on your problem type (classification or regression) and requirements. Some popular algorithms for regression tasks include linear regression, decision trees, random forests, gradient boosting methods (e.g. XGBoost), etc.

5.MODEL TRAINING:

Split your dataset into training and validation sets. Use the training set to train your chosen model using appropriate hyperparameters. Evaluate the model's performance on the validation set using suitable evaluation metrics (eg, mean squared error for regression).

6. MODEL EVALUATION:

Assess how well your trained model performs on unseen data by evaluating it on a separate test set or using cross-validation techniques. Compare different models if necessary and select the best-performing one.

7.HYPERPARAMETER TUNING:

Fine-tune your chosen model by adjusting its hyperparameters to optimize its performance further. This can be done using techniques like grid search or random search.

CODING:

```
# Aggregating in 1H intervals
#
============================================================
============================
# The Date column is eliminated so that it does not generate
an error when aggregating.
# The Holiday column does not generate an error since it is
Boolean and is treated as 0-1.
data = data.drop(columns='Date')
```

```
data = data.resample(rule='H', closed='left', label
='right').mean()
data
```

OUTPUT:

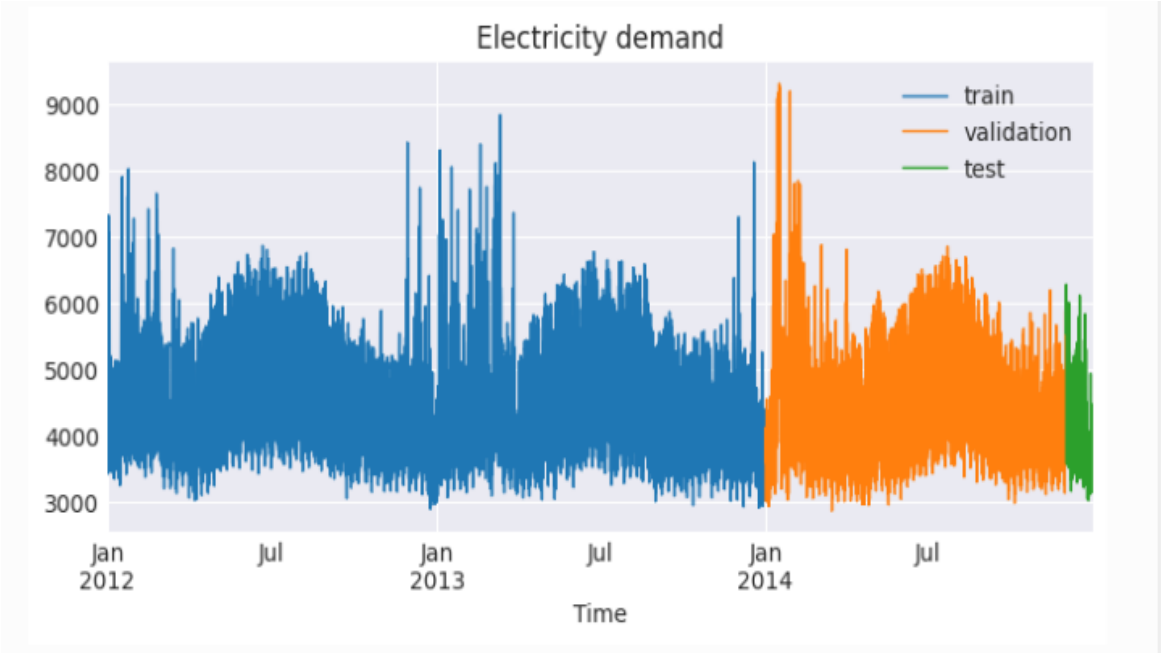
Time	Demand	Temperature	Holiday
2011-12-31 14:00:00	4323.095350	21.225	1.0
2011-12-31 15:00:00	3963.264688	20.625	1.0
2011-12-31 16:00:00	3950.913495	20.325	1.0
2011-12-31 17:00:00	3627.860675	19.850	1.0
2011-12-31 18:00:00	3396.251676	19.025	1.0
...	...	...	...
2014-12-31 09:00:00	4069.625550	21.600	0.0
2014-12-31 10:00:00	3909.230704	20.300	0.0
2014-12-31 11:00:00	3900.600901	19.650	0.0
2014-12-31 12:00:00	3758.236494	18.100	0.0
2014-12-31 13:00:00	3785.650720	17.200	0.0

Full time series

```
# Time series plot
#
==========================================================
============================
fig, ax = plt.subplots(figsize=(8, 3.5))
data_train.Demand.plot(ax=ax, label='train', linewidth=1)
data_val.Demand.plot(ax=ax, label='validation', linewidth=1)
data_test.Demand.plot(ax=ax, label='test', linewidth=1)
ax.set_title('Electricity demand')
```

ax.legend();

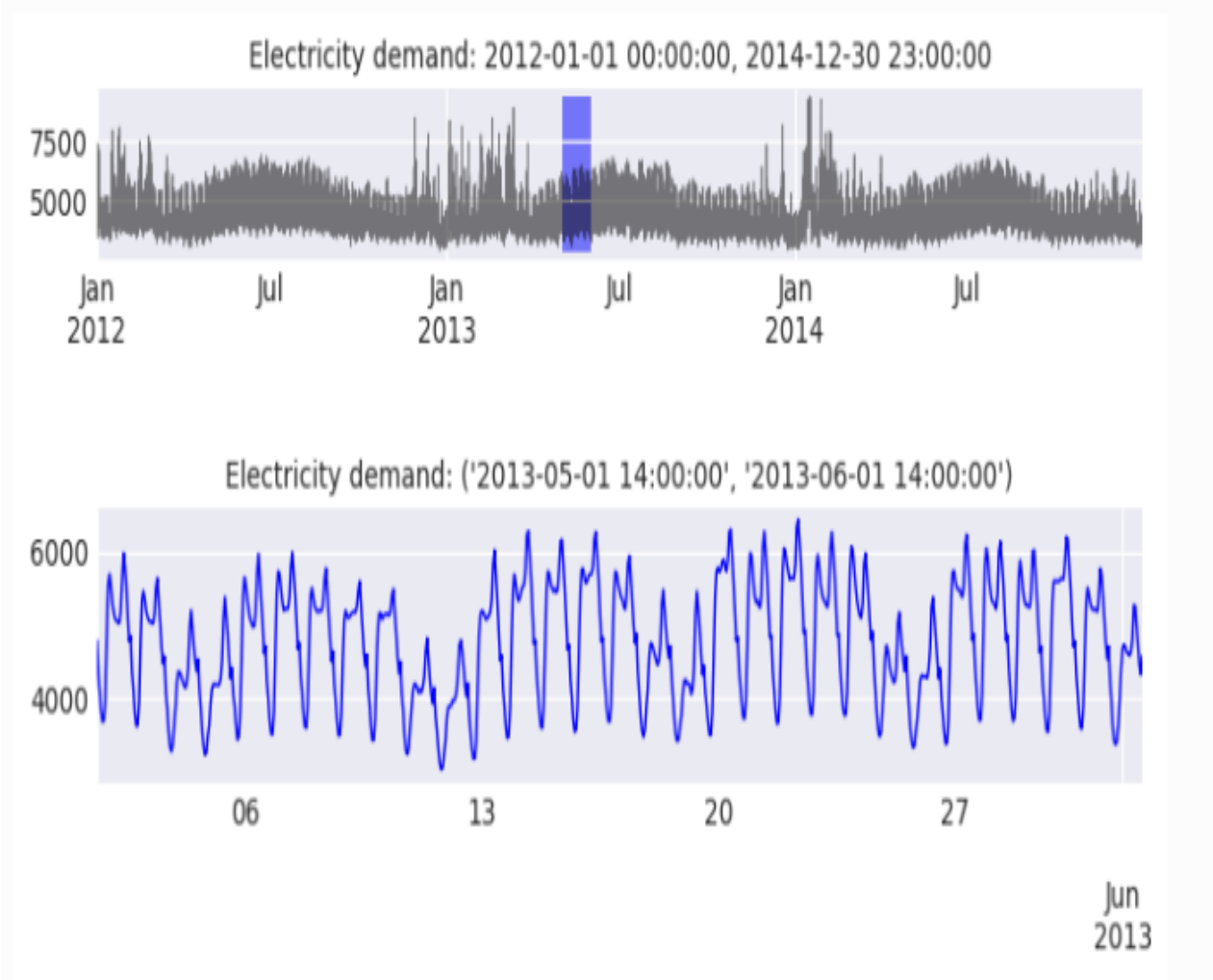
OUTPUT:



 # Zooming time series chart
#
==========
==========
zoom = ('2013-05-01 14:00:00','2013-06-01 14:00:00')
fig = plt.figure(figsize=(8, 4))
grid = plt.GridSpec(nrows=8, ncols=1, hspace=0.6, wspace=0)
main_ax = fig.add_subplot(grid[1:3, :])
zoom_ax = fig.add_subplot(grid[5:, :])
data.Demand.plot(ax=main_ax, c='black', alpha=0.5, linewidth=0.5)
min_y = min(data.Demand)
max_y = max(data.Demand)

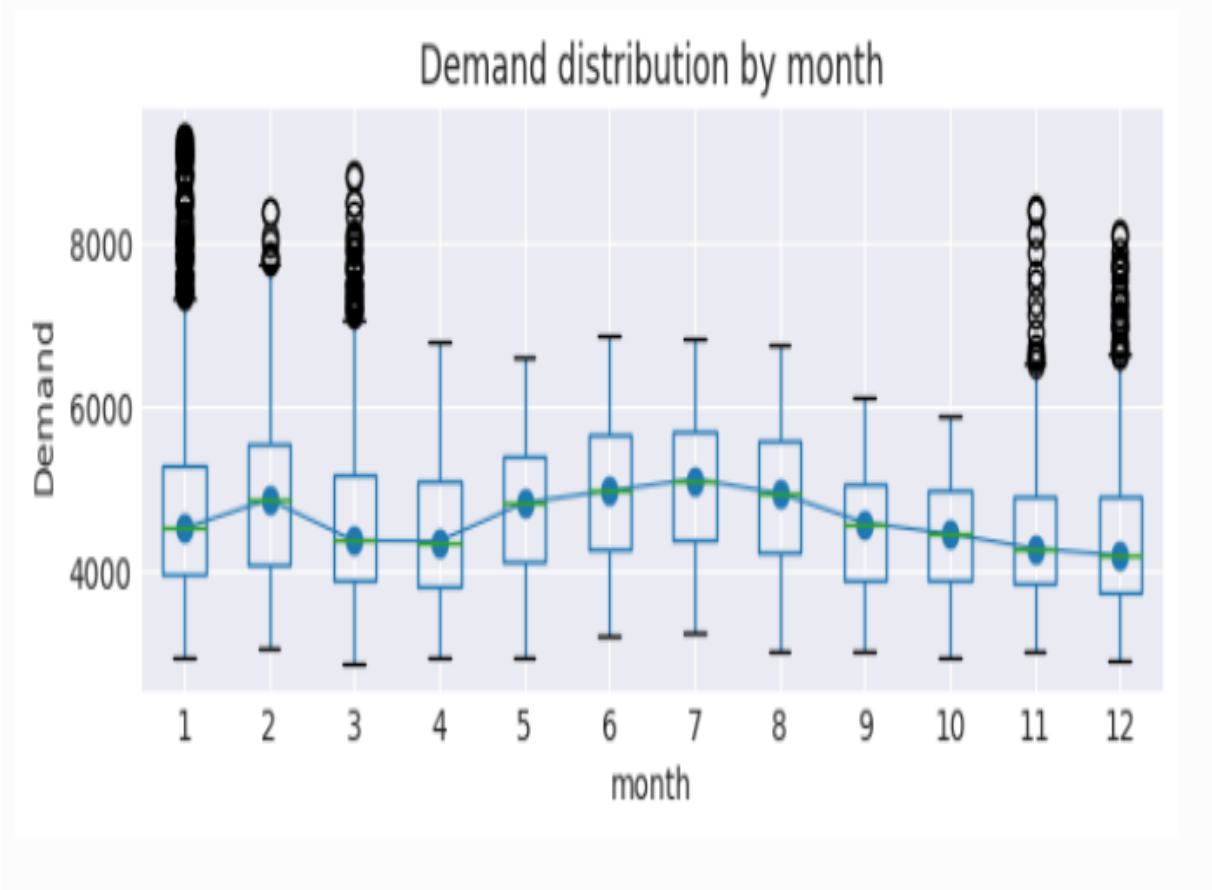
```
main_ax.fill_between(zoom, min_y, max_y, facecolor='blue',
alpha=0.5, zorder=0)
main_ax.set_xlabel('')
data.loc[zoom[0]: zoom[1]].Demand.plot(ax=zoom_ax,
color='blue', linewidth=1)
main_ax.set_title(f'Electricity demand: {data.index.min()},
{data.index.max()}', fontsize=10)
zoom_ax.set_title(f'Electricity demand: {zoom}', fontsize=10)
zoom_ax.set_xlabel('')
plt.subplots_adjust(hspace=1)
```

OUTPUT:



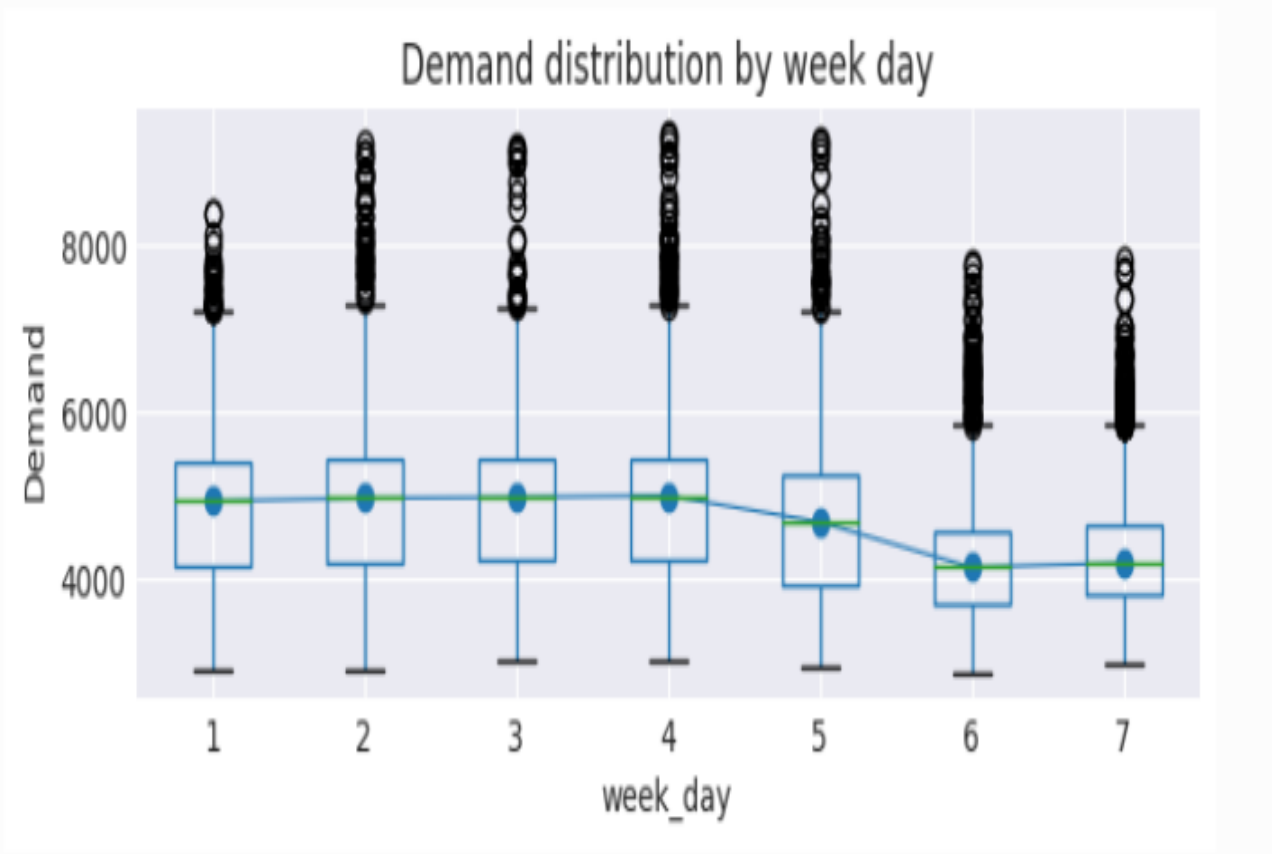
```
# Boxplot for annual seasonality
#
================================================================
============================
fig, ax = plt.subplots(figsize=(7, 2.5))
data['month'] = data.index.month
data.boxplot(column='Demand', by='month', ax=ax,)
data.groupby('month')['Demand'].median().plot(style='o-',
linewidth=0.8, ax=ax)
ax.set_ylabel('Demand')
ax.set_title('Demand distribution by month')
fig.suptitle('');
```

OUTPUT:



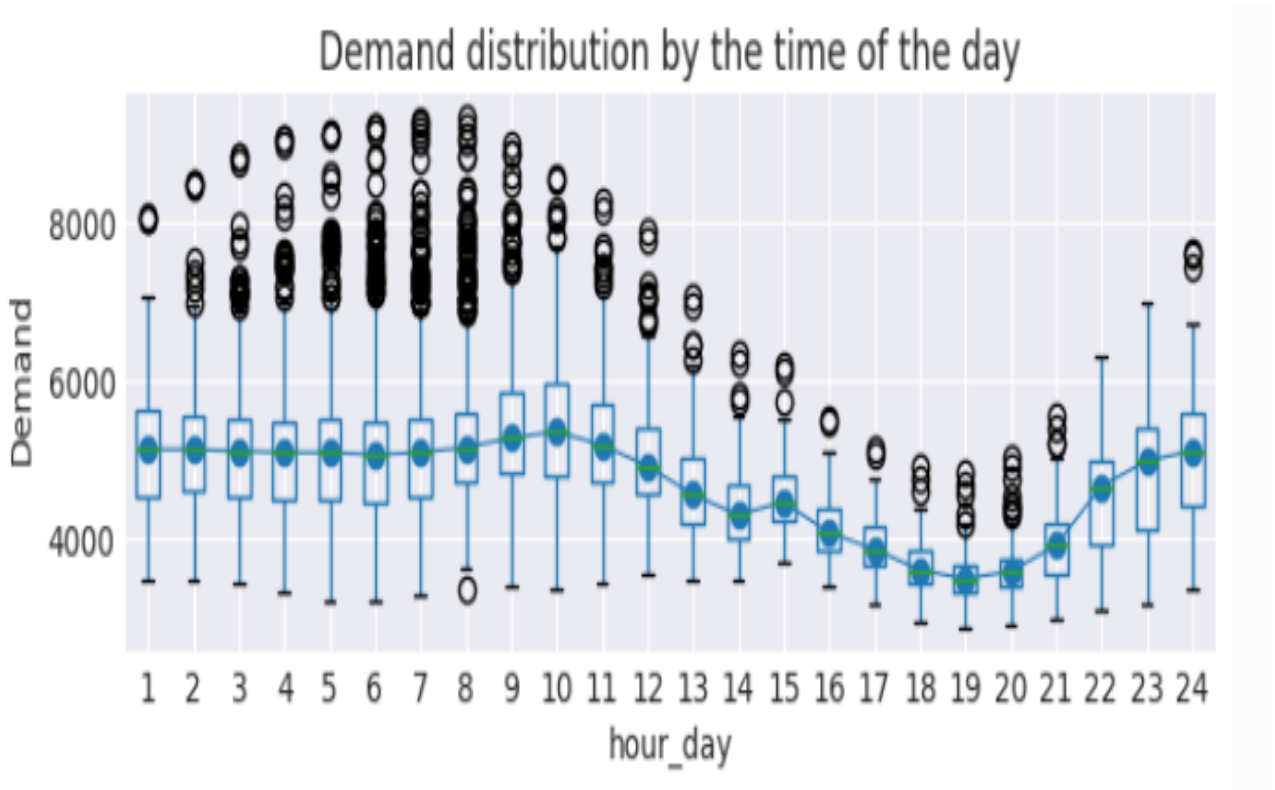
```
# Boxplot for weekly seasonality
#
================================================================
============================
fig, ax = plt.subplots(figsize=(7, 2.5))
data['week_day'] = data.index.day_of_week + 1
data.boxplot(column='Demand', by='week_day', ax=ax)
data.groupby('week_day')['Demand'].median().plot(style='o-',
linewidth=0.8, ax=ax)
ax.set_ylabel('Demand')
ax.set_title('Demand distribution by week day')
fig.suptitle('');
```

OUTPUT:



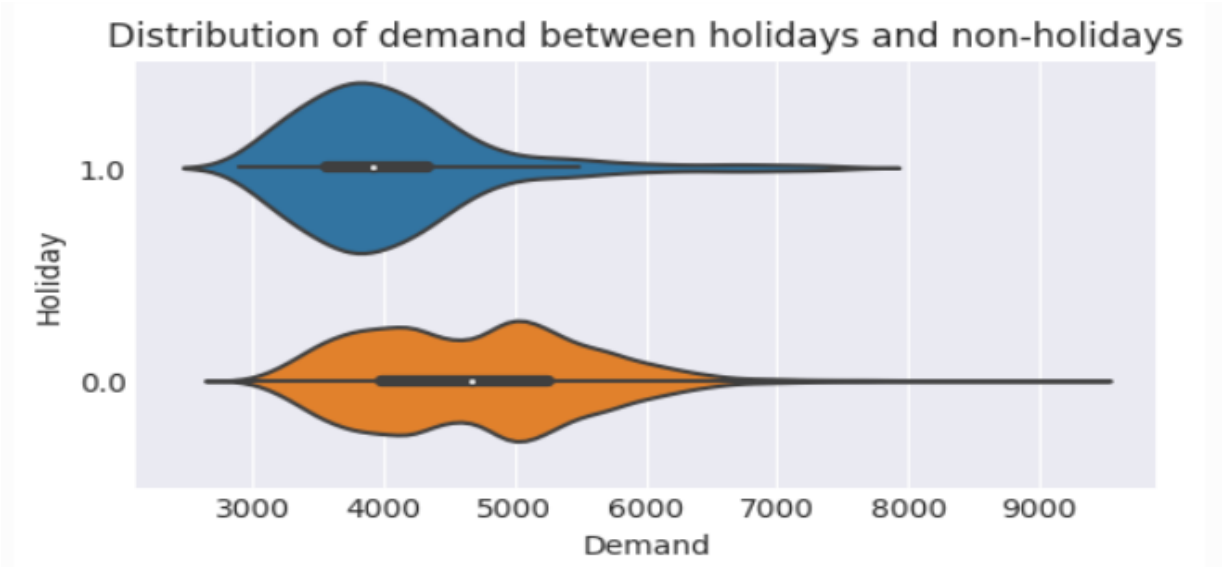
```
# Boxplot for daily seasonality
#
================================================================
============================
fig, ax = plt.subplots(figsize=(7, 2.5))
data['hour_day'] = data.index.hour + 1
data.boxplot(column='Demand', by='hour_day', ax=ax)
data.groupby('hour_day')['Demand'].median().plot(style='o-',
linewidth=0.8, ax=ax)
ax.set_ylabel('Demand')
ax.set_title('Demand distribution by the time of the day')
fig.suptitle('');
```

OUTPUT:



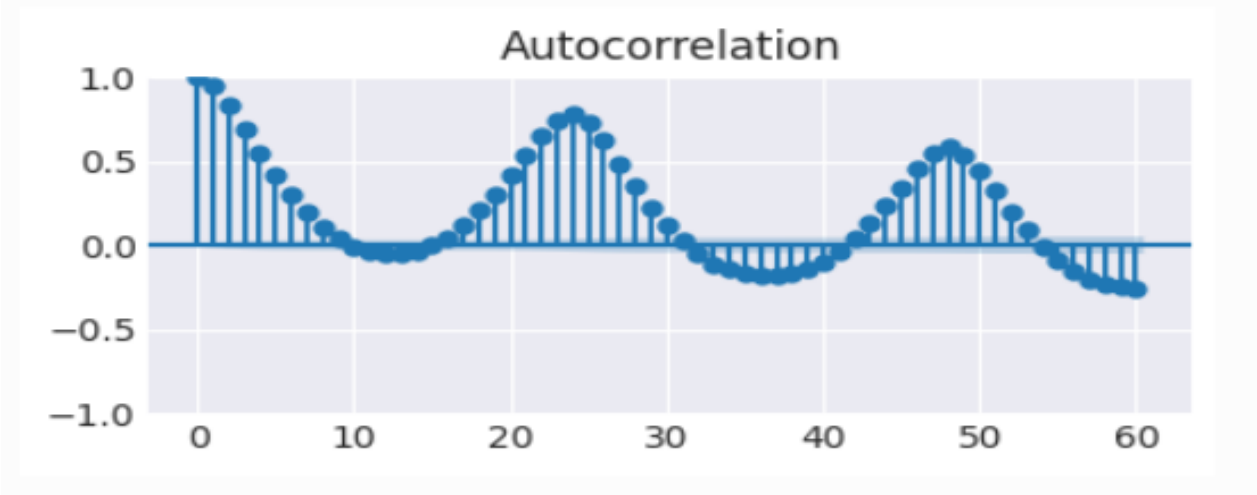
```
# Violinplot
#
================================================================
============================
fig, ax = plt.subplots(figsize=(6, 3))
sns.violinplot(
    x       = 'Demand',
    y       = 'Holiday',
    data    = data.assign(Holiday = data.Holiday.astype(str)),
    palette = 'tab10',
    ax      = ax
)
ax.set_title('Distribution of demand between holidays and
non-holidays')
ax.set_xlabel('Demand')
ax.set_ylabel('Holiday');
```

OUTPUT:



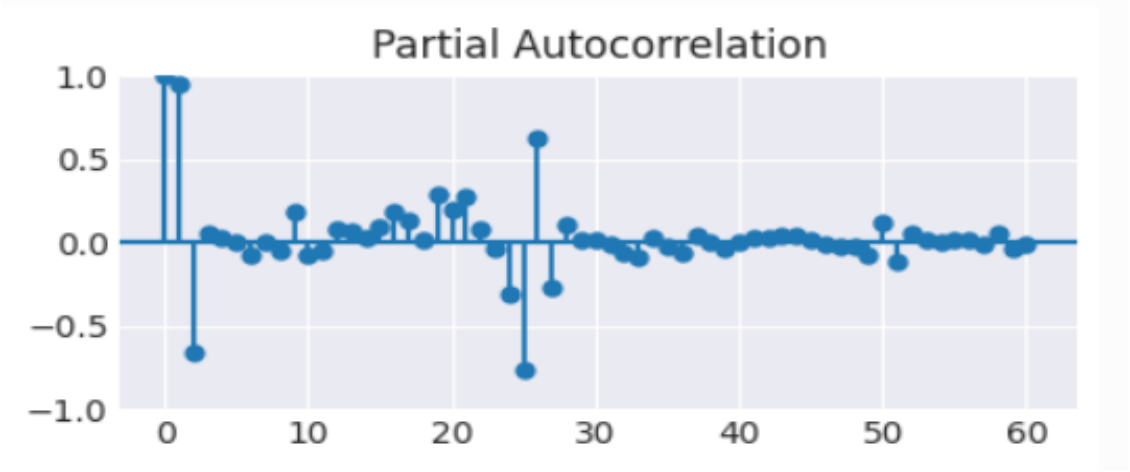
```
# Autocorrelation plot
#
================================================================
============================
fig, ax = plt.subplots(figsize=(5, 2))
plot_acf(data.Demand, ax=ax, lags=60)
plt.show()
```

OUTPUT:




```
# Partial autocorrelation plot
#
================================================================
============================
fig, ax = plt.subplots(figsize=(5, 2))
plot_pacf(data.Demand, ax=ax, lags=60)
plt.show()
```

OUTPUT:



CONCLUSION:

India stays in the place of 5 for Electricity price on Domestic side with the price of ₹6 per. By all this efforts on Electricity Price Prediction we can decrease the price and can detect the future price of the Electricity.