Development Part-2

Electricity Price Forecasting and Resource Management in Cloud Based Industrial IoT Systems: ABSTRACT:

Cloud computing is gaining popularity as a storage platform, allowing organizations to reduce hardware and procurement expenses. The exponential growth in data consumption necessitates more data center requirements, which consume significant electricity. Data centers are responsible for 2% of the total energy consumption worldwide. Furthermore, estimates indicate that this percentage is expected to grow by 12% annually. Cooling accounts for 39% of total electricity use, operating IT infrastructure accounts for 45%, and lighting.

*I* NTRODUCTION:

Cloud computing is gaining popularity as a storage platform, allowing organizations to reduce hardware and procurement expenses. The exponential growth in data consumption necessitates more data center requirements, which consume significant electricity. Data centers are responsible for 2% of the total energy consumption worldwide. Furthermore, estimates indicate that this percentage is expected to grow by 12% annually. Cooling accounts for 39% of total electricity use, operating IT infrastructure accounts for 45%, and lighting accounts for 13%. In 2008, this level of consumption resulted in a 30 billion dollar loss to the business community [1]. In the logistics context, the adoption of distributed computing with virtualization has the potential to significantly enhance productivity, although its usage is still limited. In the realm of server utilization, Ericsson’s insightful research reveals that non-virtualized servers often operate at a mere fraction of their potential, harnessing only 5-14% of their maximum capacity. In stark contrast, virtualized servers shine by unlocking their potential and reaching impressive utilization rates of up to 29% [2]. To fortify reliability, data center operators strategically disperse their facilities across diverse locations, embracing replication techniques as an assurance of seamless operations. While this approach meets latency. requirements, it can lead to unforeseen expenses due to fluctuating power costs in different regions. Energy markets exhibit high volatility, with prices surging by a factor of 10 within a mere 60 minutes. Therefore, conducting research on leveraging the volatility of deregulated energy markets becomes crucial in order to predict value spikes and optimize power usage, thereby minimizing energy expenditures in the logistics industry [3]. Businesses like Netflix rely on Content Delivery Networks (CDNs) to provide their content, and locating data centers closer to clients can improve service quality and reduce energy costs. This method involves moving capacity from centrally controlled data centers to hubs on the system’s outskirts [4]. In recent decades, there has been a growing urgency to prioritize sustainable practices and adopt energy-efficient measures to protect the environment. As a result, researchers have employed a range of traditional and innovative techniques to tackle these issues. For instance, [5] has demonstrated that power costs can be reduced in various locations. This feature significantly enhances the practicality and usefulness of our approach, making it a viable alternative for optimizing energy consumption in data storage facilities. In the logistics landscape, the growing demands of big data and cloud computing call for the establishment of expansive cloud data centres. Yet, the colossal energy consumption of these facilities presents a pressing challenge to their sustainability and efficiency. In response, researchers are fervently investigating diverse methodologies to curtail energy usage in cloud data centres, all while upholding optimal performance and reliability. In the ensuing section, we delve into pioneering tactics and approaches that spearhead this field, exploring noteworthy research on the reduction of energy consumption in cloud data centres Researchers have turned to virtual machine consolidation in their pursuit for energy-efficient data centres. This technology tries to save energy use by combining underutilised virtual machines onto fewer servers. This approach has been a subject of intense scrutiny in recent times, with numerous algorithms proposed to achieve optimal VM consolidation. However, the effectiveness of this approach hinges on the workload’s inherent characteristics, and it may falter when workloads are highly erratic and unpredictable, thereby limiting its potential benefits. In the realm of data centres, Dynamic Voltage and Frequency Scaling (DVFS) acts like an intuitive personal assistant, adapting to your work style and conserving energy. This advanced technology efficiently adjusts the frequency and voltage of processors in real-time according to workload demands. By preventing over-exertion, DVFS serves as a valuable energy-saving tool for data centres. This innovative method, meanwhile, also has certain difficulties that need to be resolved. To ensure maximum efficiency, DVFS must accurately interpret each individual workload, which can be hindered by the intricate, nonlinear relationships between frequency, voltage, and workload characteristics. If not applied properly, DVFS may lead to performance degradation or instability, much like an ill-designed work schedule may cause exhaustion or injury. The pursuit of energy-efficient task scheduling requires a delicate balance between minimizing energy consumption and meeting the resource demands of high-intensity applications. Energy-aware task scheduling achieves this balance by intelligently scheduling tasks to optimize resource utilization, while ensuring application-level constraints are satisfied. However, the effectiveness of this technique depends on the specific workload characteristics and optimization objectives at hand. Numerous algorithms, such as genetic algorithms, ant colony optimization, and particle swarm optimization, have been proposed to achieve energy-aware task scheduling. Nevertheless, these algorithms may introduce significant overhead or result in suboptimal solutions. Energy effectiveness and performance must be perfectly balanced, which calls for meticulous planning and close attention to detail. When executed correctly, energy-aware task scheduling can lead to substantial energy savings and enhanced system performance. In order to increase energy efficiency in the constantly changing environment of cloud data centres, researchers have resorted to machine learning-based solutions. The crux of these techniques lies in the use of machine learning models to predict workload demand and resource usage patterns, and subsequently, make dynamic resource allocation decisions that minimize energy consumption while maintaining performance and reliability. One such technique is multi-task learning, which is proving to be an increasingly powerful tool in this space. By leveraging the interdependencies between electricity price forecasting and resource management tasks, multi-task learning is helping to achieve improved accuracy and efficiency in both domains, driving energy savings and better system performance. Basically, researchers have investigated various techniques to reduce energy consumption in cloud data centres, such as VM consolidation, DVFS, energy-aware task scheduling, and machine learning-based techniques. Multi-task learning is a promising approach that can lead to better results than single-task learning or other approaches by exploiting the interdependencies between electricity price forecasting and resource management tasks. The ever-increasing demand for cloud computing services to manage and process large volumes of data has compelled cloud providers to constantly seek innovative techniques to reduce the energy consumption required to store this data [14]. Additionally, cloud providers face the challenge of maintaining government expectations and earning profits through Service Level Agreements (SLAs) while ensuring energy effectiveness [15], [16], [17]. The fluctuating nature of deregulated energy prices has created a strong incentive to explore whether these variations can be leveraged to minimize energy costs while preserving optimal performance . This study investigates whether machine learning techniques can effectively capitalize on significant energy price spikes and reduce operational expenses associated with data centres. The above-mentioned is addressed in this article with the following contributions:

• An optimization method DMOA has been used to significantly reduce energy consumption in data centres.

• A new model called Alex Net-DMOA has been proposed to optimize storage location and predict power prices more accurately.

• The model has been trained with 75% of available data to ensure high precision, while the remaining 25% has been used for testing purposes.

• The Alex Net-DMOA model forecasts power prices with an MAE of 2.22% and an MSE of 6.33%, resulting in an average reduction of 22.21% in electricity expenses.

• The proposed algorithm outperforms 11 benchmark algorithms applied in the latest literature in terms of performance metrics, accuracy, time complexity, data processing, and model overfitting issues.

RELATED WORK

The increasing emphasis on sustainability within the logistics industry has raised concerns regarding energy consumption and its environmental impact. This paper provides a concise overview of previous methodologies employed to forecast power usage in logistics operations. It also highlights the limitations of existing research, prompting the exploration of more robust and effective approaches. To address these challenges, researchers have utilized a Multi-Layer Neural Network (MLNN) model, as demonstrated in [21], [22], and [23], to estimate power load and overall electricity consumption in logistics operations. By leveraging the Ensemble technique, which combines multiple machine learning models to reduce errors and eliminate noise, significant improvements have been achieved in the accuracy of energy consumption predictions. These advancements hold promising implications for optimizing energy usage and sustainability in the logistics sector. This combination of techniques allows for a more precise estimation of energy usage. It will enable cloud providers to make better-informed decisions about power usage and resource allocation. Ultimately, this leads to improved energy efficiency and cost savings for cloud data centres. While their approach showed competitive accuracy, it lacked resilience due to longer processing times and high loss rates during live testing Similarly, in [24], the author proposed a hybrid method called EPNet for energy price prediction, using LSTM and CNN models that produced MSE and MAE of 7.74 and 16.8, respectively. Despite the favourable results, these models had high error rates and required significant computational power for real-time predictions. Moreover, the model’s performance was impacted by the heavy normalization of the dataset, and it failed to reproduce the same results when applied to real-time data. In [25], the author proposed a model similar to those above, combining support vector regression with other optimization methods. The model yielded a 6.82 MAE, but only for one-day-ahead forecasts, rendering the results unreliable. Moreover, the model’s results are inconsistent and subject to change, making it unsuitable for real-time application. Additionally, the models incur high computational costs. In [26] and [27], researchers conducted a comparative study of DL-based methods for predicting electricity consumption and green energy. They evaluated the performance of 23 benchmark methods, including CNN, GRU-DNN, and LSTM-DNN. They proposed a DL-based algorithm for power price prediction, demonstrating results comparable to prior studies. Nonetheless, the proposed model incurs high computational costs and generates inaccurate predictions when used in real-time applications, resulting in a significant testing loss. The comparison was based on a single, thoroughly normalized dataset. In [28], the author proposed a hybrid approach for power price prediction that integrated both SVM and Kernel Principal Component Analysis (KPCA). The proposed technique delivered promising results, with a low error rate of 5.7 percent for one threshold value and a higher but still reasonable error rate of 47.9 percent for another. This hybrid method presented in [29] can reduce energy consumption and costs in data centres by allowing for a more accurate and efficient In [28], the author proposed a hybrid approach for power price prediction that integrated both SVM and Kernel Principal Component Analysis (KPCA). The proposed technique delivered promising results, with a low error rate of 5.7 percent for one threshold value and a higher but still reasonable error rate of 47.9 percent for another. This hybrid method presented in [29] can reduce energy consumption and costs in data centres by allowing for a more accurate and efficient. In the logistics domain, the proposed method differs from other models as it takes into account the static nature and regional dependencies of energy prices, considering variations across seasons and locations. Researchers in previous studies, as seen in references [30] and [31], demonstrated successful outcomes by adopting location-specific data collection and a combination of Autoencoder models and NN-based models. Additionally, advanced deep learning techniques, highlighted in [32], were employed to enhance the accuracy of energy cost forecasts in the European market. These researchers utilized sophisticated feature selection methods, resulting in promising results using a simplified model. Nevertheless, the MAE and MSE values were relatively high, and the methodology did not tackle the problem comprehensively. The model presented in [33] employed multivariate techniques to estimate energy costs hourly and used dimension reduction to address over-fitting concerns. The author of [34] introduced a DNN-based model that combined LSTM and LSTM-based models to predict power prices and load, but the outcomes were inadequate in predicting power prices. Most of the current research has been centerd around applying established deep-learning techniques. Nevertheless, these methods can be computationally demanding and may yield unforeseeable results, mainly when dealing with large-scale datasets [35]. Alternatively, [36] took a different approach by emphasizing feature selection, which led to an MAE of 3.18. However, using a sizable dataset, their model was only appropriate for offline prediction. The article [37] delved into the combination of power cost estimation and energy demand prediction, utilizing the Artificial Bee Colony and SVM algorithms with Least Square. On the other hand, [38] proposed an ANN-based approach, and [39] put forward a hybrid methodology employing a model based on a biweight kernel with dynamical system reconstruction to forecast electricity prices using datasets from the ISO of New York, the US, and the South Wales markets. However, these models are computationally expensive, generate inaccurate predictions resulting in significant losses, and are inefficient for real-time use. The article [37] delved into the combination of power cost estimation and energy demand prediction, utilizing the Artificial Bee Colony and SVM algorithms with Least Square. On the other hand, [38] proposed an ANN-based approach, and [39] put forward a hybrid methodology employing a model based on a bi weight kernel with dynamical system reconstruction to forecast electricity prices using datasets from the ISO of New York, the US, and the South Wales markets. However, these models are computationally expensive, generate inaccurate predictions resulting in significant losses, and are inefficient for real-time use. Energy price prediction has been an essential topic of discussion for many years, with a wealth of literature available to estimate power consumption in DCs and reduce it. However, existing techniques have limitations in providing efficient results for the global market with low MSE and MAE. Most of them are computationally expensive and unsuitable for real-time usage. Countless studies have delved into the search for more energy-efficient cloud data centres. They have explored several strategies to lessen energy consumption while maintaining system performance and reliability. A strategy that aims to minimize energy consumption by matching the workload demand with processor performance is dynamic voltage and frequency scaling (DVFS). However, researchers have noted the challenges posed by the non-linear relationship between frequency, voltage, and workload characteristics []. Similarly, the consolidation of underutilized virtual machines (VMs) onto fewer servers has been investigated as a method for diminishing energy consumption. However, its effectiveness is limited when workloads are highly dynamic and unpredictable [39]. The attention of researchers has been drawn towards the potential of machine learning-based methods for reducing energy consumption in cloud data centres. For instance, multi-task learning has been investigated as a powerful machine learning technique for both electricity price forecasting and resource management tasks in cloud-based Industrial IoT systems, leading to improved accuracy and efficiency. In order to enhance feature representations and increase performance, the research suggests a semisupervised feature analysis method for multi-task learning. The suggested issue of projecting power prices and resource management can be enhanced by taking use of the relationship between the two jobs [42]. Other researchers have proposed using machine learning models to predict workload demand and resource usage patterns, which are then used to dynamically adjust resource allocation to minimize energy consumption while maintaining performance and reliability. In order to translate across languages, this paper suggests an unsupervised multi-modal machine translation strategy that pivots on movies. The proposed approach’s multi-task learning component views the tasks of resource management and energy price forecasting as separate languages that require translation. Videos can serve as a springboard for innovative ideas on how to best take advantage of the relationship between the two jobs and produce superior outcomes.

**About Dataset**

The price of electricity depends on many factors. Predicting the price of electricity helps many businesses understand how much electricity they have to pay each year. The Electricity Price Prediction task is based on a case study where you need to predict the daily price of electricity based on the daily consumption of heavy machinery used by businesses.

You do not know the actual cost of the electricity consumed by the machines throughout the day, but the organization has provided you with historical data of the price of the electricity consumed by the machines. Below is the information of the data we have for the task of forecasting electricity prices:

* DateTime: Date and time of the record
* Holiday: contains the name of the holiday if the day is a national holiday
* HolidayFlag: contains 1 if it’s a bank holiday otherwise 0
* DayOfWeek: contains values between 0-6 where 0 is Monday
* WeekOfYear: week of the year
* Day: Day of the date
* Month: Month of the date
* Year: Year of the date
* PeriodOfDay: half-hour period of the day
* ForcastWindProduction: forecasted wind production
* SystemLoadEA :forecasted national load
* SMPEA: forecasted price
* ORKTemperature: actual temperature measured
* ORKWindspeed: actual windspeed measured
* CO2Intensity: actual C02 intensity for the electricity produced
* ActualWindProduction: actual wind energy production
* SystemLoadEP2: actual national system load
* SMPEP2: the actual price of the electricity consumed (labels or values to be predicted)

So your task here is to use this data to train a machine learning model to predict the price of electricity consumed by the machines.

add Codeadd Markdown

## Veriseti Hakkında

Elektriğin fiyatı birçok faktöre bağlıdır. Elektriğin fiyatını tahmin etmek, birçok işletmenin her yıl ne kadar elektrik ödemek zorunda olduklarını anlamalarına yardımcı olur. Elektrik Fiyat Tahmini görevi, işletmeler tarafından kullanılan ağır makinelerin günlük tüketimine dayalı olarak günlük elektrik fiyatını tahmin etmeniz gereken bir vaka çalışmasına dayanmaktadır.

Gün boyunca makinelerin tükettiği elektriğin gerçek maliyetini bilmiyorsunuz, ancak kuruluş size makinelerin tükettiği elektriğin fiyatının tarihsel verilerini sağladı. Elektrik fiyatlarını tahmin etme görevi için sahip olduğumuz verilerin bilgileri aşağıdadır:

* DateTime
* Holiday
* HolidayFlag
* DayOfWeek
* WeekOfYear
* Day
* Month
* Year
* PeriodOfDay
* ForcastWindProduction
* SystemLoadEA
* SMPEA
* ORKTemperature
* ORKWindspeed
* CO2Intensity
* ActualWindProduction
* SystemLoadEP2
* SMPEP2

**Analysis Content**

* 1.[Python Libraries](https://kkb-production.jupyter-proxy.kaggle.net/static/dist/jupyterlab/jupyterlab-index-20a71085d35681864457.html?session=eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.#1)
* 2.[data loading](https://kkb-production.jupyter-proxy.kaggle.net/static/dist/jupyterlab/jupyterlab-index-20a71085d35681864457.html?session=eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.#2)
* 3.[EDA](https://kkb-production.jupyter-proxy.kaggle.net/static/dist/jupyterlab/jupyterlab-index-20a71085d35681864457.html?session=eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.#3)
* 4.[data Preprocessing](https://kkb-production.jupyter-proxy.kaggle.net/static/dist/jupyterlab/jupyterlab-index-20a71085d35681864457.html?session=eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.#4)
* 5.[Modelling](https://kkb-production.jupyter-proxy.kaggle.net/static/dist/jupyterlab/jupyterlab-index-20a71085d35681864457.html?session=eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.#5)
* 6.[RandomForest](https://kkb-production.jupyter-proxy.kaggle.net/static/dist/jupyterlab/jupyterlab-index-20a71085d35681864457.html?session=eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.#6)
* 7.[Conclusion](https://kkb-production.jupyter-proxy.kaggle.net/static/dist/jupyterlab/jupyterlab-index-20a71085d35681864457.html?session=eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.#7)

**1.Python Libraries**

*#Let's load the relevant libraries (İlgili kütüphaneleri yükleyelim);*

​

**import** numpy **as** np

**import** pandas **as** pd

**from** sklearn.model\_selection **import** train\_test\_split,GridSearchCV,RandomizedSearchCV

**from** sklearn.metrics **import** mean\_squared\_error,r2\_score

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**import** plotly.express **as** px

**from** sklearn.preprocessing **import** scale

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn **import** model\_selection

**from** sklearn.linear\_model **import** Ridge,Lasso,RidgeCV,LassoCV,ElasticNet,ElasticNetCV,LinearRegression

**from** sklearn.tree **import** DecisionTreeRegressor

**from** sklearn.neighbors **import** KNeighborsRegressor

**from** sklearn.neural\_network **import** MLPRegressor

**from** sklearn.ensemble **import** RandomForestRegressor

**from** sklearn.ensemble **import** GradientBoostingRegressor

**from** sklearn.ensemble **import** AdaBoostRegressor

**from** sklearn **import** neighbors

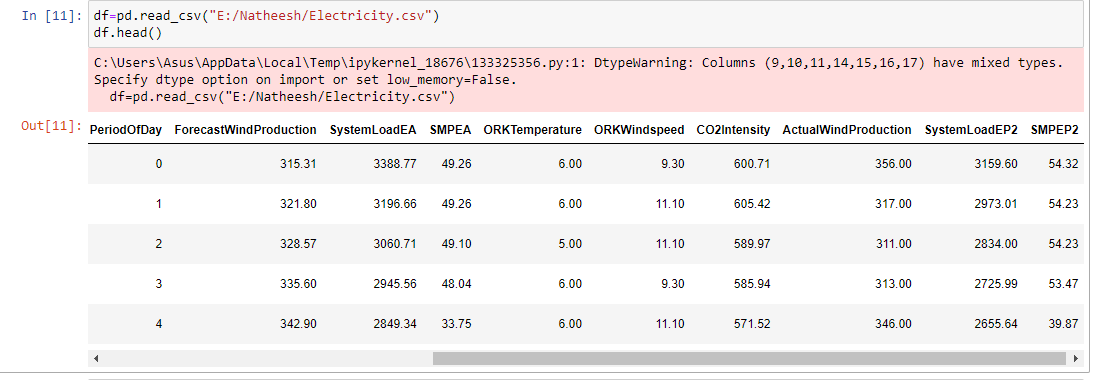
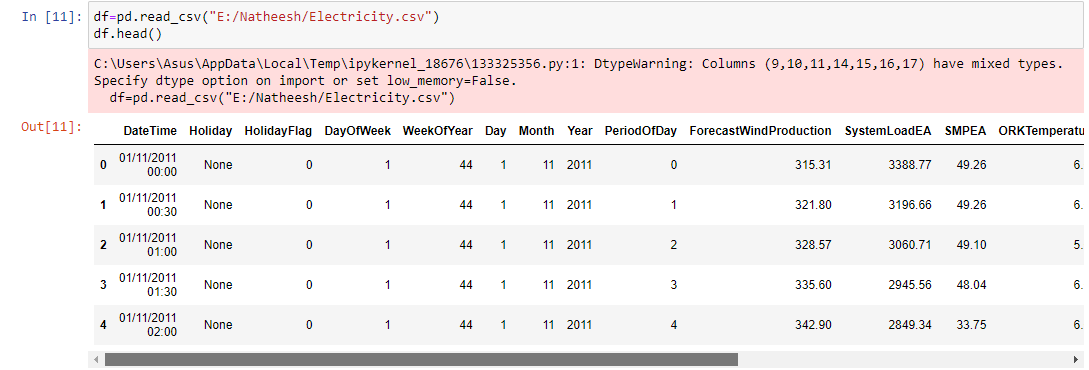
**from** sklearn.svm **import** SVR

## 2.Data Loading

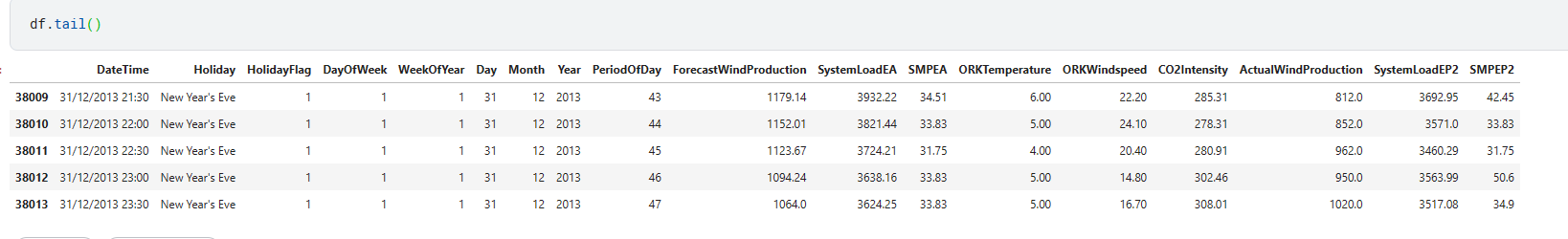
df=pd.read\_csv("/kaggle/input/electric/electricity.gui")

df.head()

OUTPUT:



df.tail()

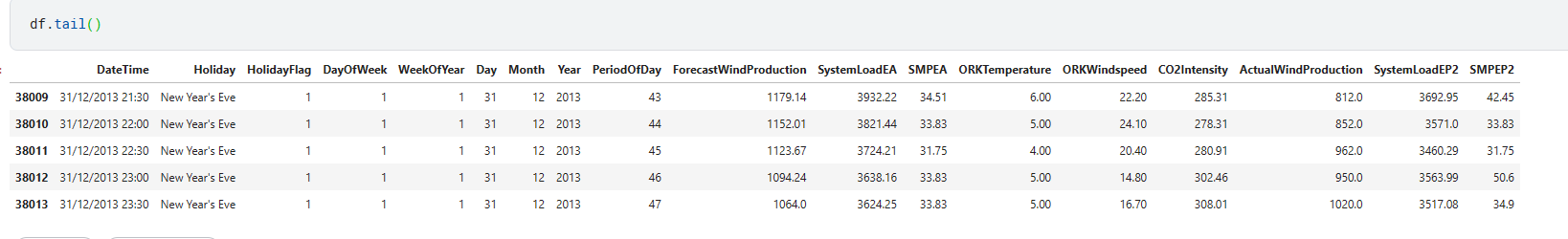


## 3.EDA

df.shape

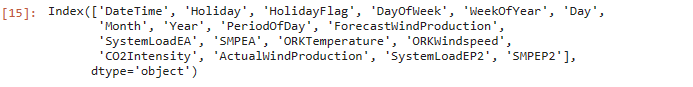
**(38014, 18)**

# our dataset consists of 38014 observations and 18 attributes



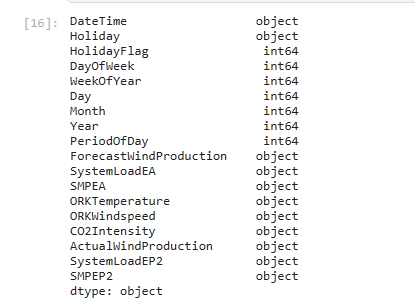
#columns

df.columns



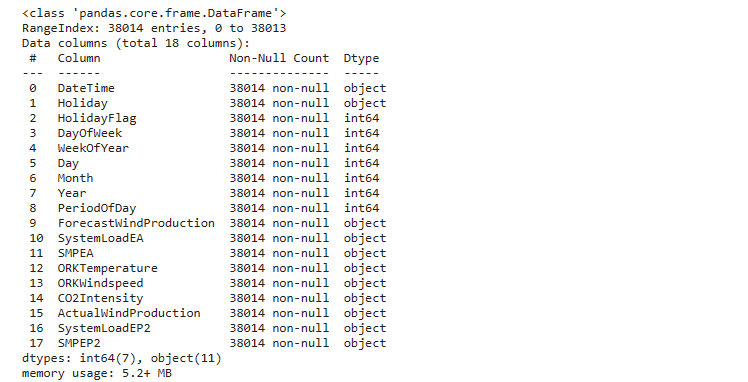
# datatypes

df.dtypes



#structural information

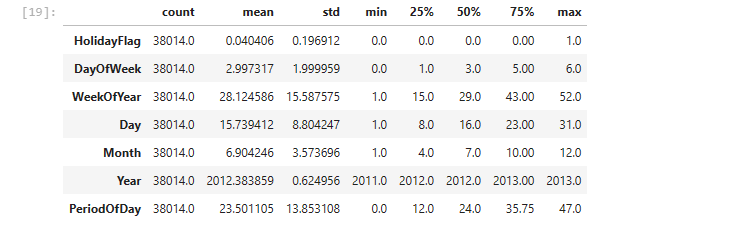
df.info()



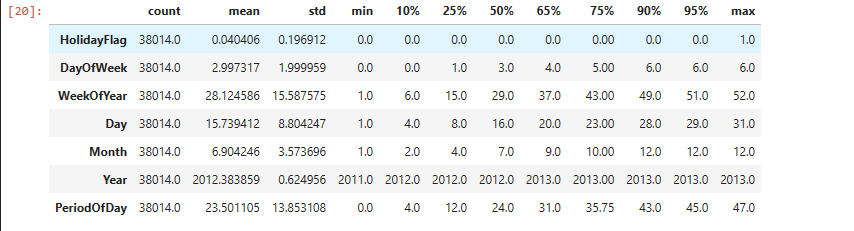
# we got information about the structural features of our dataset

# dataset summary

df.describe().T

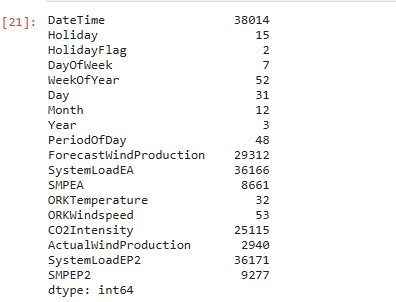


df.describe([0.1,0.25,0.5,0.65,0.75,0.9,0.95]).Tv



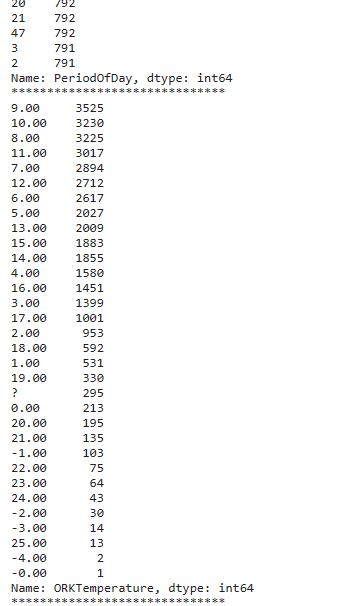
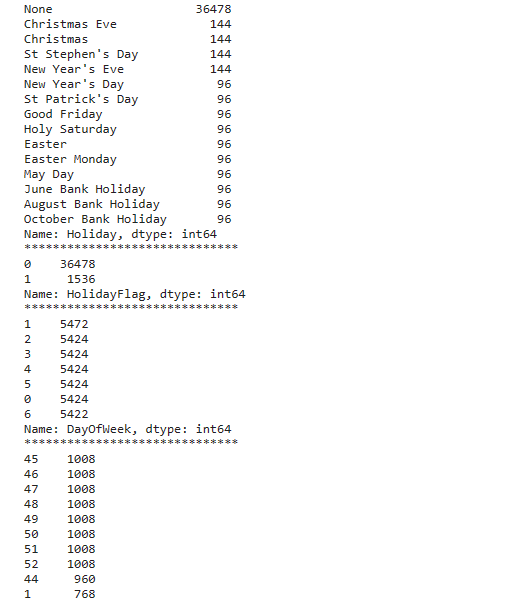
# unique value counts

df.nunique()



col=["Holiday","HolidayFlag","DayOfWeek","WeekOfYear","Day","Month",

"Year","PeriodOfDay","ORKTemperature"]



for i in col:

print(df[i].value\_counts())

print("\*"\*30)

# we have accessed the class counts for each category

#Let's convert string values to floats;

#pd.to\_numeric?

# convert

df["ForecastWindProduction"]=pd.to\_numeric(df["ForecastWindProduction"], errors= 'coerce')

df["SystemLoadEA"] = pd.to\_numeric(df["SystemLoadEA"], errors= 'coerce')

df["SMPEA"] = pd.to\_numeric(df["SMPEA"], errors= 'coerce')

df["ORKTemperature"] = pd.to\_numeric(df["ORKTemperature"], errors= 'coerce')

df["ORKWindspeed"] = pd.to\_numeric(df["ORKWindspeed"], errors= 'coerce')

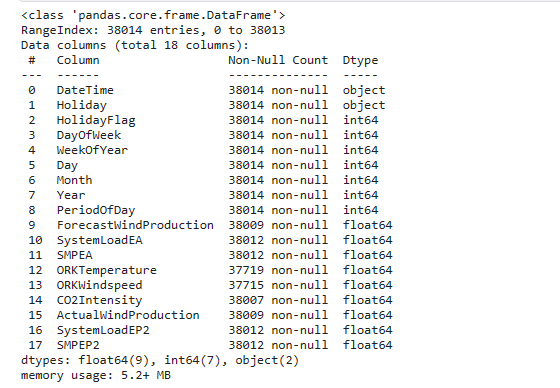
df["CO2Intensity"] = pd.to\_numeric(df["CO2Intensity"], errors= 'coerce')

df["ActualWindProduction"] = pd.to\_numeric(df["ActualWindProduction"], errors= 'coerce')

df["SystemLoadEP2"] = pd.to\_numeric(df["SystemLoadEP2"], errors= 'coerce')

df["SMPEP2"] = pd.to\_numeric(df["SMPEP2"], errors= 'coerce')

df.info()



df.describe([0.05,0.1,0.25,0.35,0.5,0.65,0.75,0.9,0.95,0.98]).T

# estimated wind speed (highest-lowest 10)

df.sort\_values("ForecastWindProduction",ascending=False).head(10)

df.sort\_values("ForecastWindProduction").head(10)

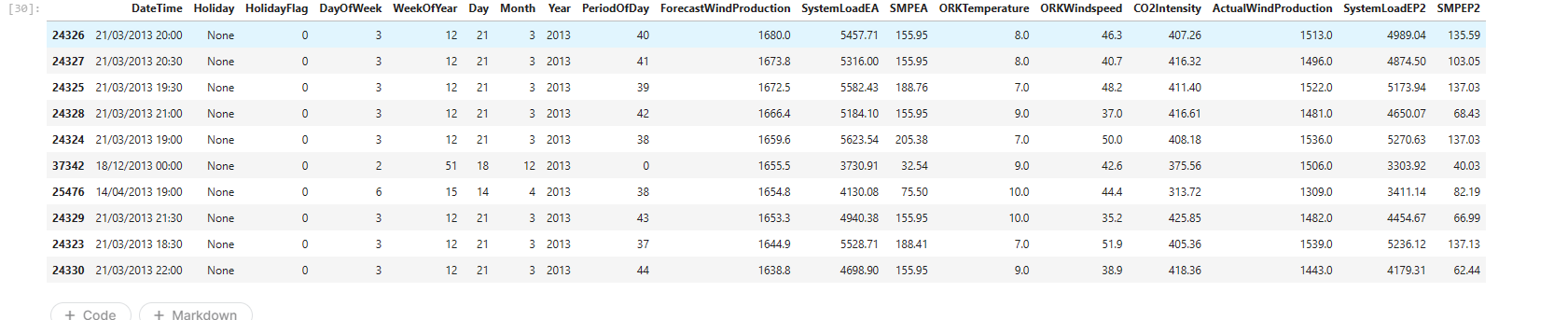
d actual wind speed (highest-lowest)

df.sort\_values("ORKWindspeed",ascending=False).head(10) # highest

df.sort\_values("ORKWindspeed").head(10) # lowest

# estimated price highest-lowest

df.sort\_values("SMPEA",ascending=False).head(10) # highest



df.sort\_values("SMPEA").head(10) # lowest

# statistics of holiday status by months and years

df.groupby("Holiday")[["Month","Year"]].describe().T

#The lowest and highest value for the real price of electricity consumed

df[df.SMPEP2==-47.74]

df[df.SMPEP2<0]

# for actual measured temperature

df[df.ORKTemperature<0]

## 4.Data Preprocessing

# missing value query

df.isna().sum()

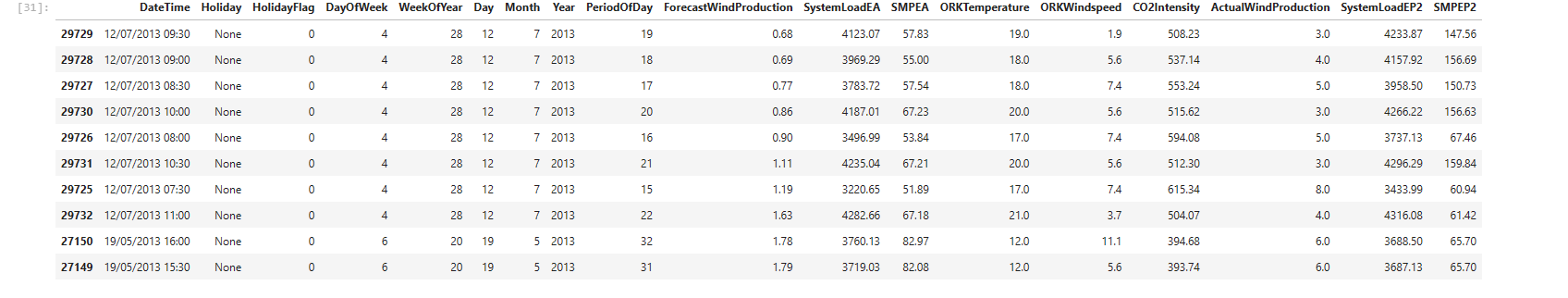
* we can see that there are missing values in our dataset, for this, let's look at the distribution situations before removing the missing values, then we can develop a strategy for missing values

# create a list for numeric and categorical values

cat\_list=[]

num\_list=[]

for i in df.columns:



unique\_val=len(df[i].unique())

if unique\_val<40:

cat\_list.append(i)

else:

num\_list.append(i)

cat\_list.append("WeekOfYear")

cat\_list

num\_list

# distributions of numeric attributes

num\_list.remove("DateTime")

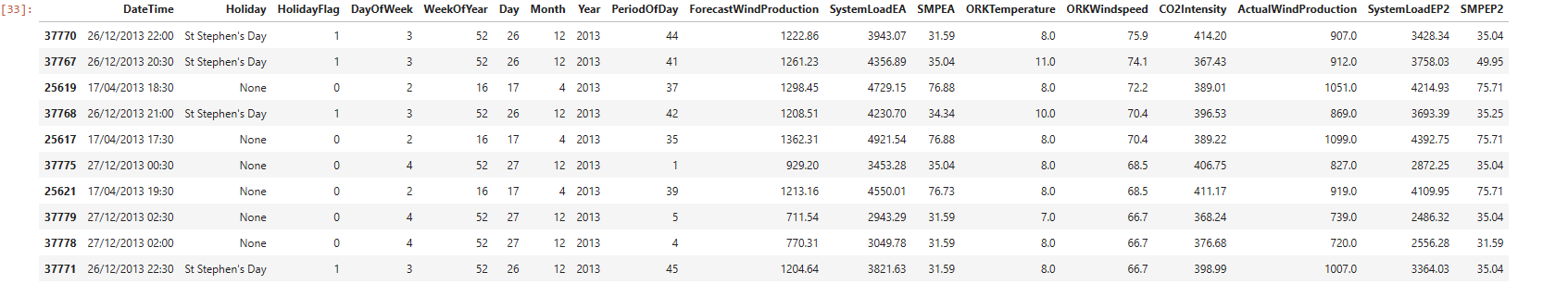
num\_list

num\_list.append("ORKTemperature")

k=1

plt.figure(figsize=(12,12))

plt.suptitle("distribution of numerical values")



for i in df.loc[:,num\_list]:

plt.subplot(6,2,k)

sns.distplot(df[i])

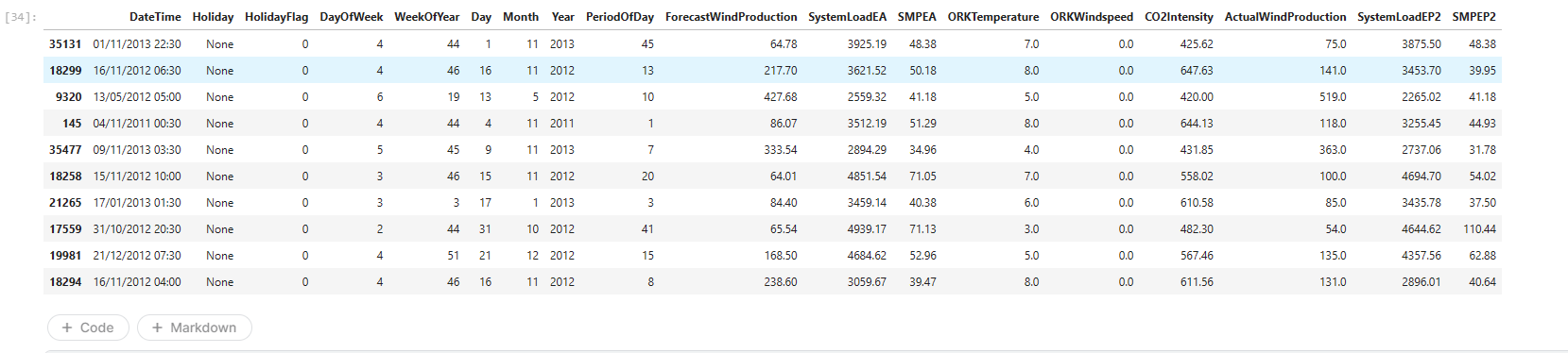
plt.title(i)

k+=1

plt.tight\_layout()

### Visualization of missing values

import missingno as msno

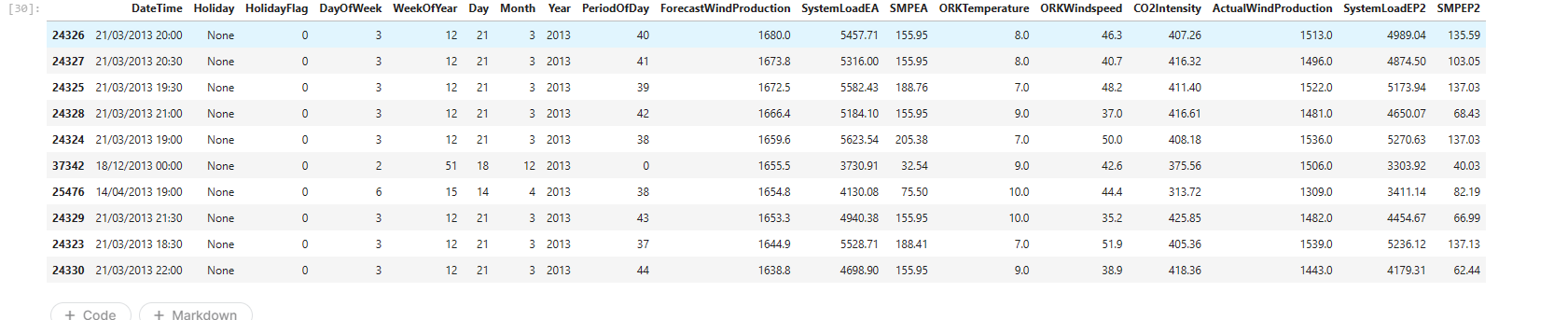
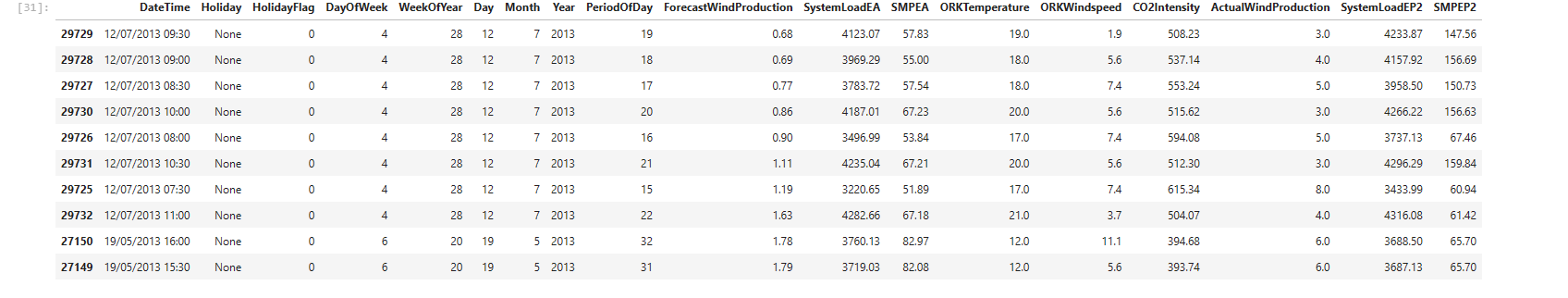


msno.matrix(df);

# let's visualize whether there is a relationship between the missing values

msno.heatmap(df);

\* there is a high correlation between missing values, the number of missing values is high Let's focus on our ORKTemperature and ORKWindspeed variants



# missing values based on distribution states# eksik değer giderme

df["ForecastWindProduction"].fillna(df.ForecastWindProduction.mean(),inplace**=True**)

df["SystemLoadEA"].fillna(df.SystemLoadEA.mean(),inplace**=True**)

df["SMPEA"].fillna(df.SMPEA.mean(),inplace**=True**)

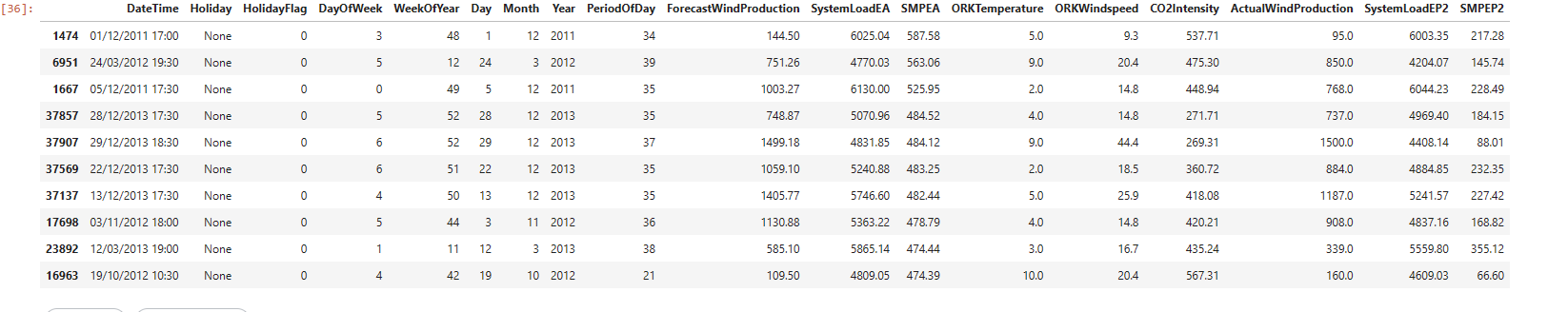
df["CO2Intensity"].fillna(df.CO2Intensity.median(),inplace**=True**)

df["ActualWindProduction"].fillna(value**=**250,inplace**=True**)

df["SystemLoadEP2"].fillna(df.SystemLoadEP2.median(),inplace**=True**)

df["SMPEP2"].fillna(df.SMPEP2.median(),inplace**=True**)

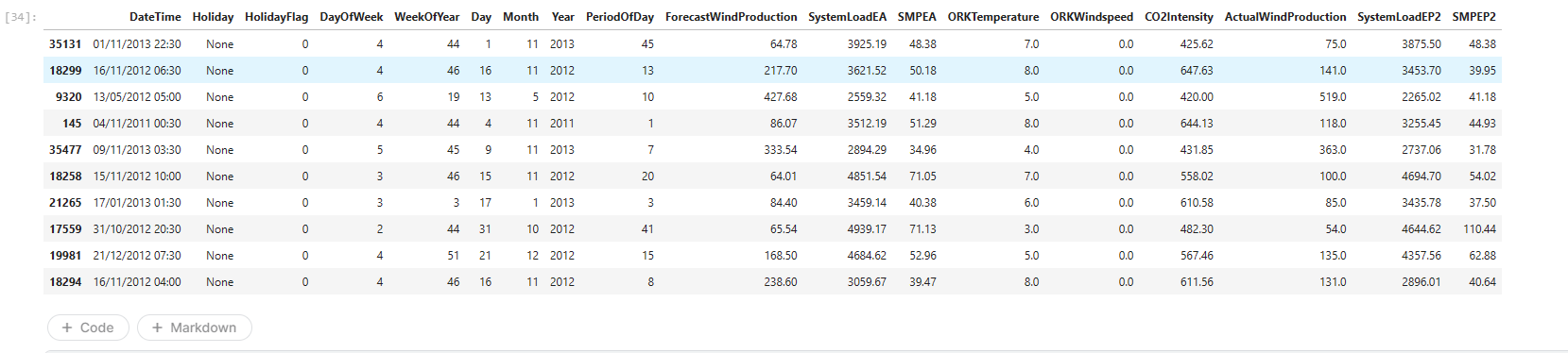
​



df["ORKTemperature"].fillna(value=10,inplace=True)

df["ORKWindspeed"].fillna(value=20,inplace=True)

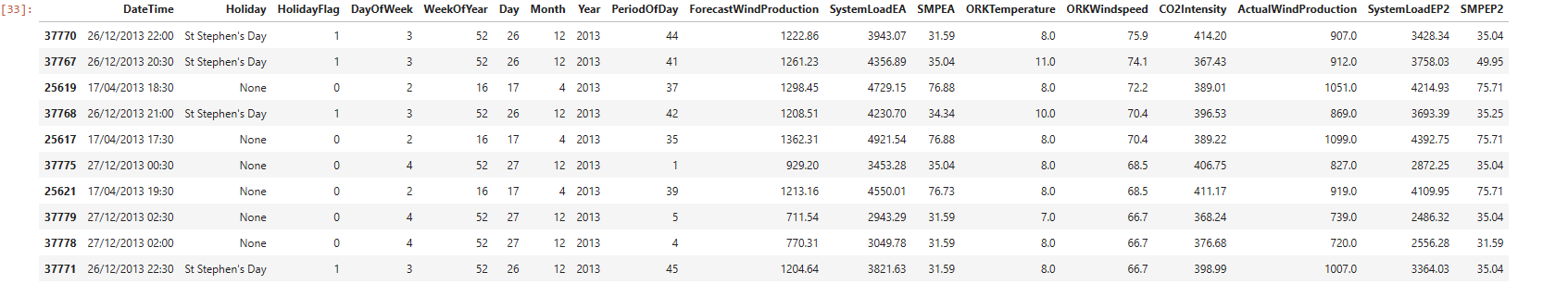
df.isna().sum()



# we have removed the missing values

## Outlier Problem

df.describe([0.05,0.1,0.25,0.35,0.5,0.65,0.75,0.9,0.95,0.98]).T



num\_list

out\_list=["ForecastWindProduction","SystemLoadEA","SMPEA" "ORKWindspeed","SMPEP2"]

for i in df.loc[:,out\_list]:

Q1 = df[i].quantile(0.02)

Q3 = df[i].quantile(0.98)

IQR = Q3-Q1

up = Q3 + 1.5\*IQR

low = Q1 - 1.5\*IQR

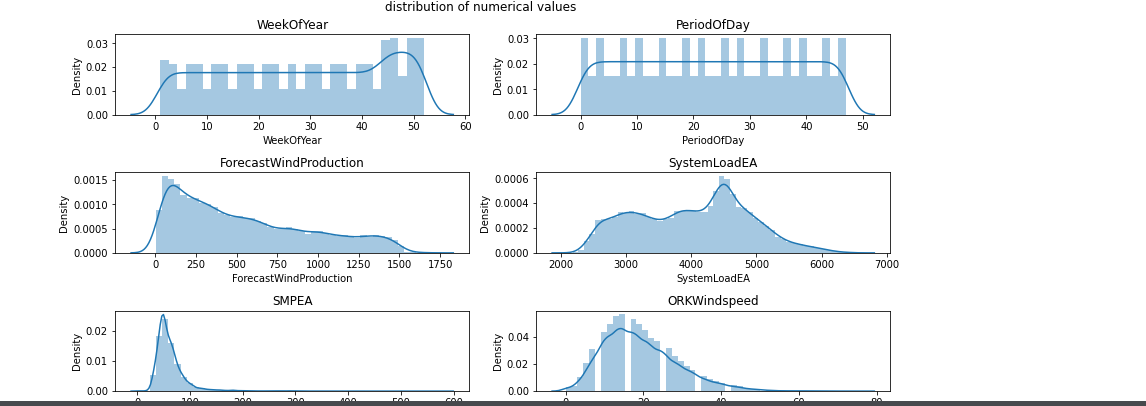
if df[(df[i] > up) | (df[i] < low)].any(axis=None):

print(i,"yes")

else:

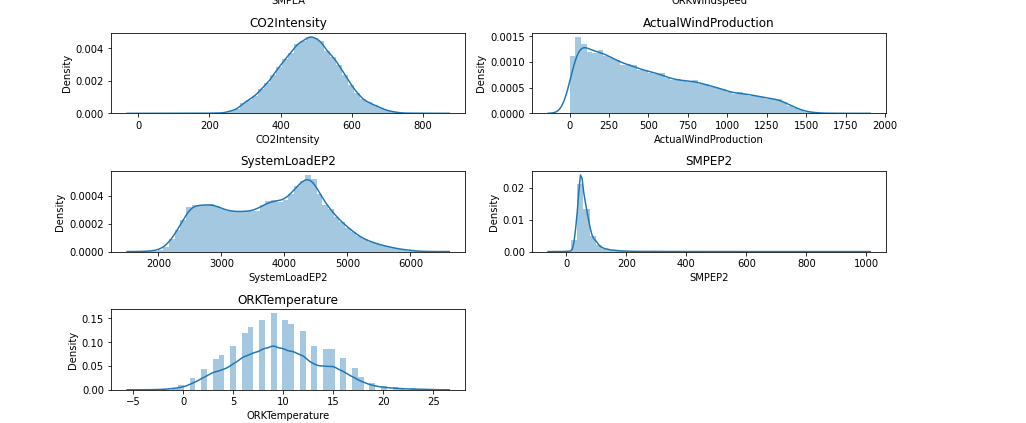
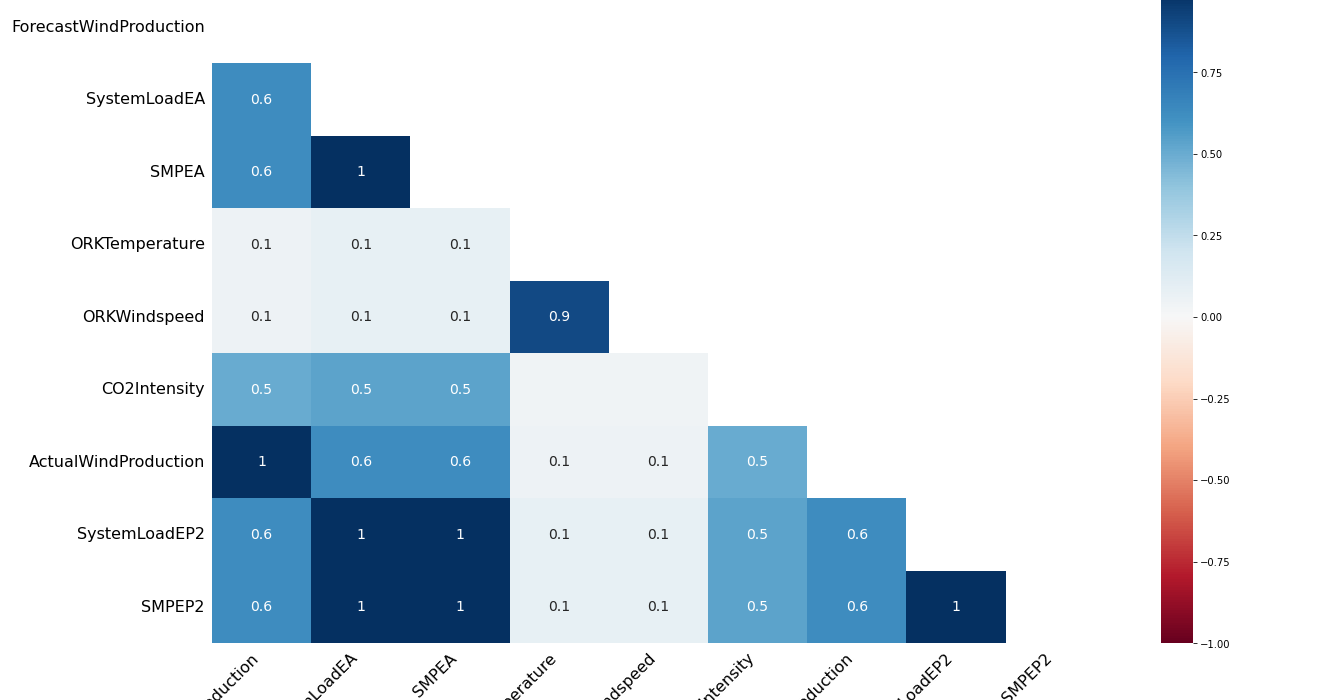
print(i, "no")

# we observed outliers



#accessing outliers

def outliers\_df(df):



q1,q3=np.percentile(df,[0.02,0.98])

ıqr=q3-q1

low,high=q1-1.5\*(ıqr),q3+1.5\*(ıqr)

outliers\_train=[i for i in df if i<low or i>high]

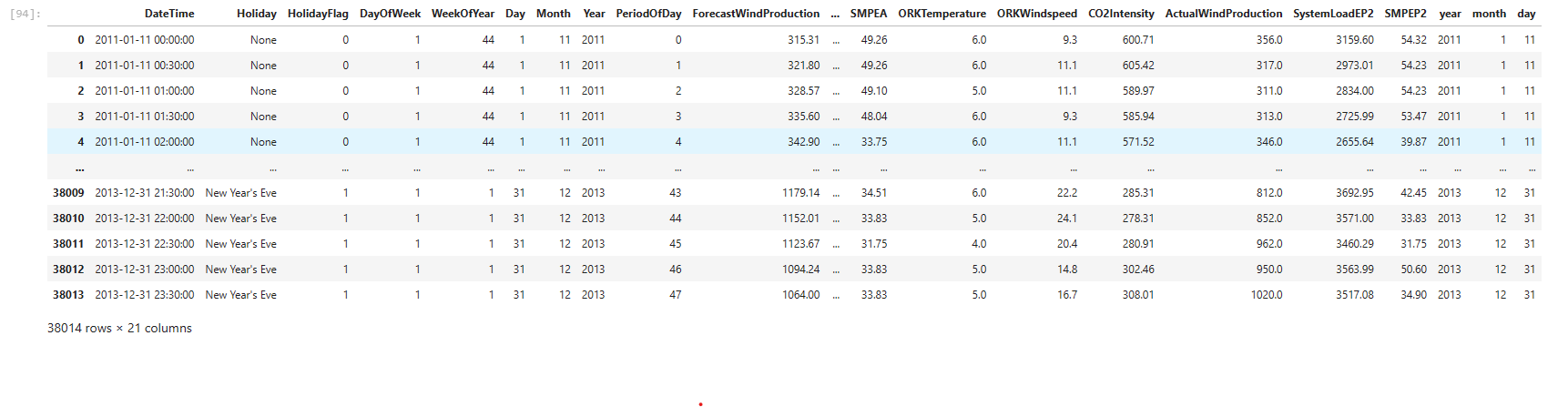
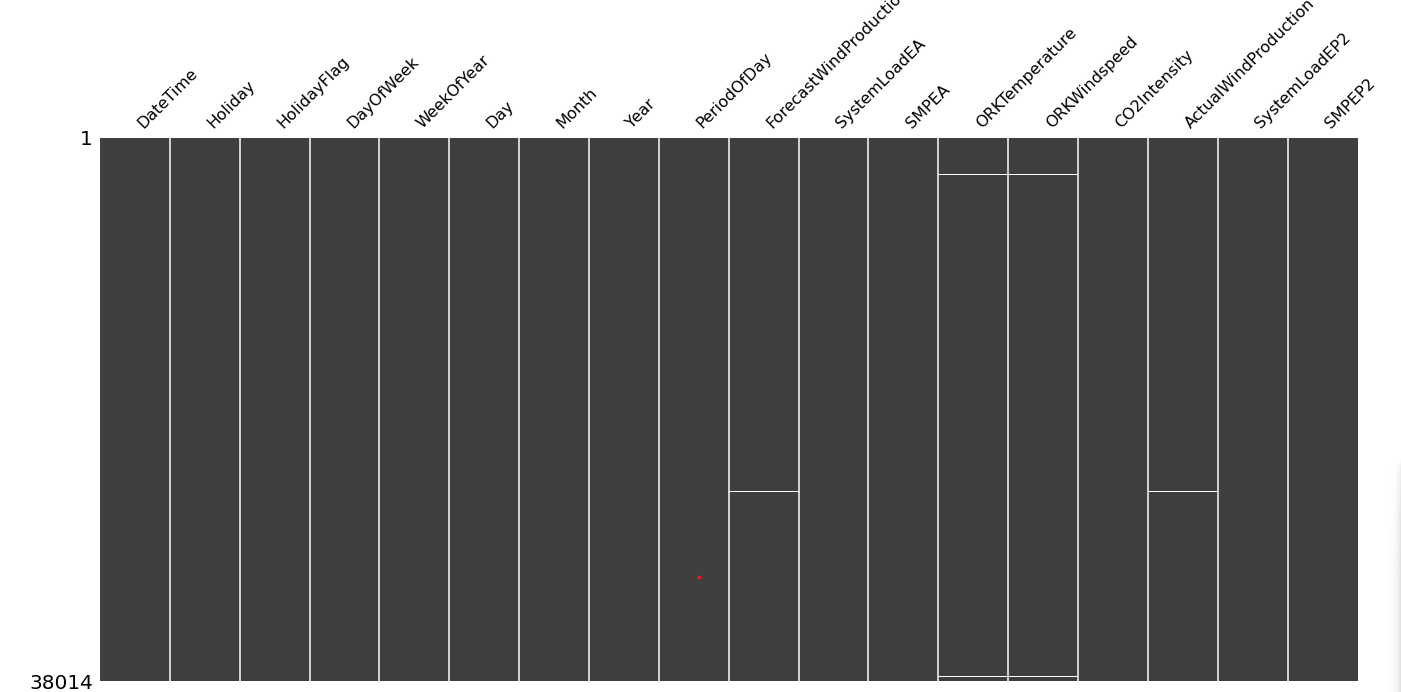
return outliers\_train

len(outliers\_df(df.SMPEA))

len(outliers\_df(df.SMPEP2))

# let's make a copy of the dataset and remove the outliers

df\_remove\_out=df.copy()



# remove outliers;

for i in df\_remove\_out.loc[:,out\_list]:

Q1 = df\_remove\_out[i].quantile(0.02)

Q3 = df\_remove\_out[i].quantile(0.98)

IQR = Q3 - Q1

up\_lim=Q3+1.5 \*IQR

low\_lim=Q1-1.5 \*IQR

df\_remove\_out.loc[df\_remove\_out[i]>up\_lim,i]=up\_lim

df\_remove\_out.loc[df\_remove\_out[i]<low\_lim,i]=low\_lim

for i in df\_remove\_out.loc[:,out\_list]:

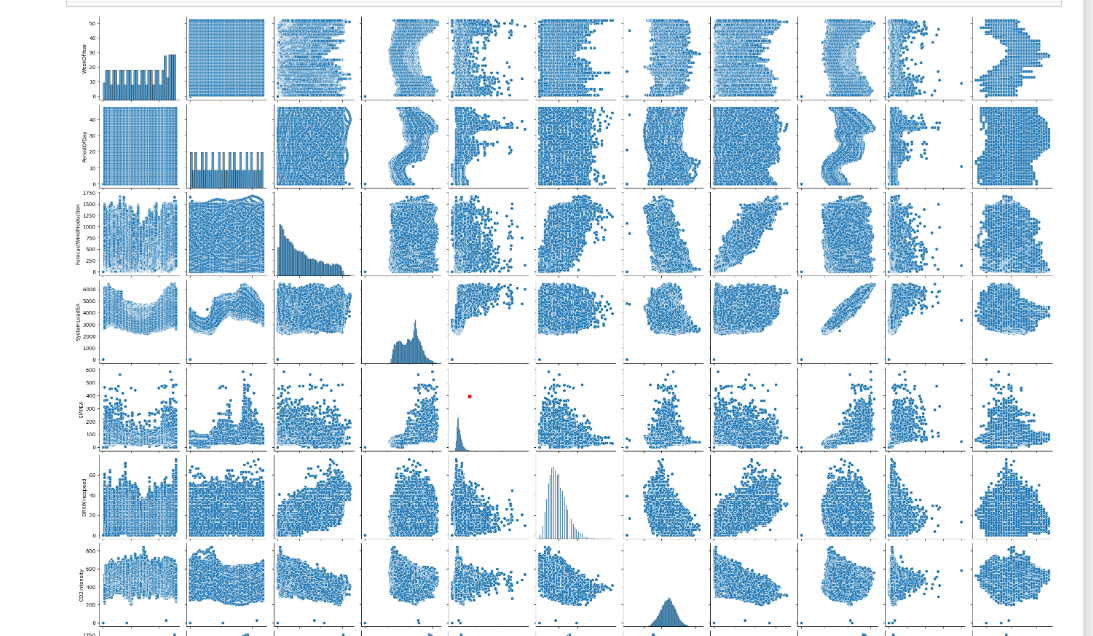
Q1 = df\_remove\_out[i].quantile(0.02)

Q3 = df\_remove\_out[i].quantile(0.98)

IQR = Q3-Q1

up = Q3 + 1.5\*IQR

low = Q1 - 1.5\*IQR



if df[(df\_remove\_out[i] > up) | (df\_remove\_out[i] < low)].any(axis=None):

print(i,"yes")

else:

print(i, "no")

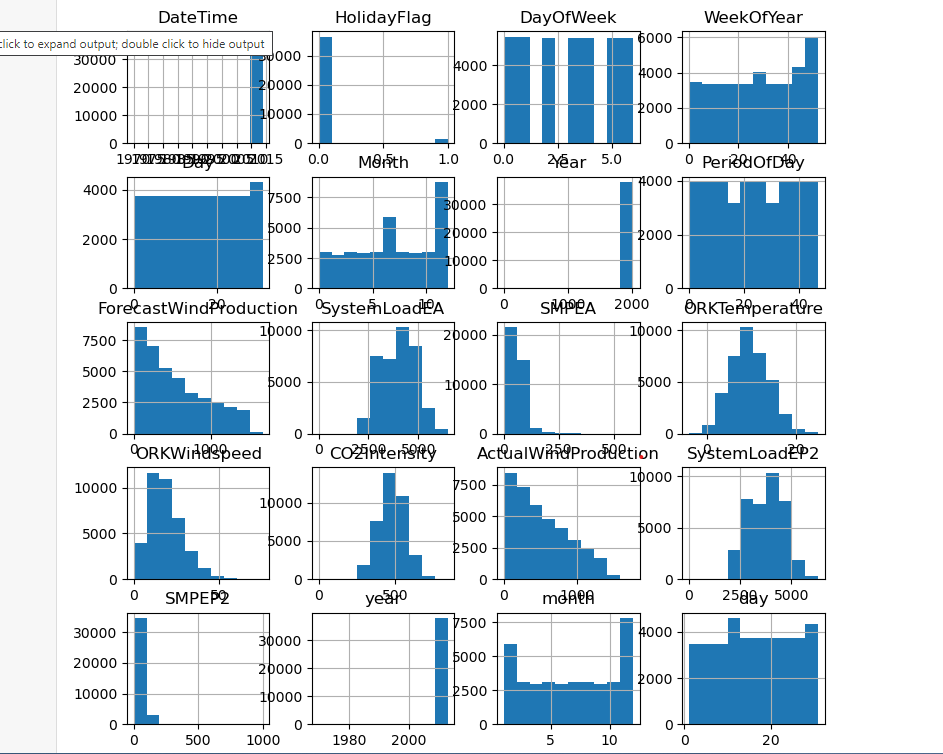
* we removed outliers for df\_remove\_out, now let's look at statistics

df.describe([0.05,0.1,0.25,0.35,0.5,0.65,0.75,0.9,0.95,0.98]).T

df\_remove\_out.describe([0.05,0.1,0.25,0.35,0.5,0.65,0.75,0.9,0.95,0.98]).T

* real price of consumed electricity cannot be below 0, let's fix it in both datasets

df[df.SMPEP2<0]=0



df\_remove\_out[df\_remove\_out.SMPEP2<0]=0

### Time Series Analysis

from datetime import datetime

df["DateTime"] = pd.to\_datetime(df.DateTime)

df['year'] = df['DateTime'].dt.year

df['month'] = df['DateTime'].dt.month

df["day"]=df["DateTime"].dt.day

# We have created 3 new columns

# we can start our time series analysis

# change of real price of consumed electricity with time

custgroup=df.groupby('DateTime').mean()

plt.figure(figsize=(12,5))

custgroup['SMPEP2'].plot(x=df.DateTime)

plt.title("SMPEP2 status")

plt.show()

# There is an increase between 2010 and 2015

custgroup=df.groupby('month').mean()

fig,ax=plt.subplots(figsize=(12,5))

ax.xaxis.set(ticks=range(0,13))

custgroup['SMPEP2'].plot(x=df.DateTime)

plt.title("SMPEP2 status by month")

plt.show()

# change in electricity price by months, there is an increase in the 3rd month at most

custgroup=df.groupby('day').mean()

plt.figure(figsize=(12,5))

custgroup['SMPEP2'].plot(x=df.DateTime)

plt.title("change in SMPEP2 by days")

plt.show()

*# change of electricity prices according to days, at most the first 5 days*

# let's make the necessary updates in df\_remove\_out;

df\_remove\_out["DateTime"] = pd.to\_datetime(df\_remove\_out.DateTime)

df\_remove\_out['year'] = df\_remove\_out['DateTime'].dt.year

df\_remove\_out['month'] = df\_remove\_out['DateTime'].dt.month

df\_remove\_out["day"]=df\_remove\_out["DateTime"].dt.day

df\_remove\_out

# Categoricakl Analysis

for i in cat\_list:

plt.figure(figsize=(13,13))

sns.countplot(x=i,data=df.loc[:,cat\_list])

plt.title(i)

# numerical analysis

sns.pairplot(df.loc[:,num\_list]);

# histogram

df.hist(figsize=(9,9));

cat\_list

num\_list

plt.figure(figsize=(15,15))

plt.subplot(3,2,1)

sns.barplot(x ='Year',y ='SystemLoadEA',data = df)

plt.subplot(3,2,2)

sns.barplot(x="DayOfWeek",y="SMPEP2",data=df)

plt.subplot(3,2,3)

sns.boxplot(x="Month",y="SMPEP2",data=df)

plt.subplot(3,2,4)

sns.boxplot(x="Day",y="ORKWindspeed",data=df)

plt.subplot(3,2,5)

sns.violinplot(x="Holiday",y="SystemLoadEA",data=df)

plt.subplot(3,2,6)

sns.barplot(x="Holiday",y="ORKWindspeed",data=df)

plt.show()

df.drop("DateTime",axis=1,inplace=True)

df.head(2)

dms=pd.get\_dummies(df["Holiday"])

dms

df.drop("Holiday",axis=1,inplace=True)

df=pd.concat([df,dms],axis=1)

df.head()

dms2=pd.get\_dummies(df\_remove\_out["Holiday"])

df\_remove\_out.drop("Holiday",axis=1,inplace=True)

df\_remove\_out=pd.concat([df\_remove\_out,dms2],axis=1)

df\_remove\_out.head()

df\_remove\_out.drop("DateTime",axis=1,inplace=True)

plt.figure(figsize=(12,12))

sns.heatmap(df.corr(),annot=True,linewidths=0.7,fmt=".2f",cmap="coolwarm")

plt.show()

cor=df.corr()["SMPEP2"].sort\_values(ascending=False)

pd.DataFrame({"column":cor.index,"Correlation with a":cor.values})

## 5.Modelling

X=df.drop("SMPEP2",axis=1)

y=df["SMPEP2"]

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.3,random\_state=0)

from xgboost import XGBRegressor

from catboost import CatBoostRegressor

from lightgbm import LGBMRegressor

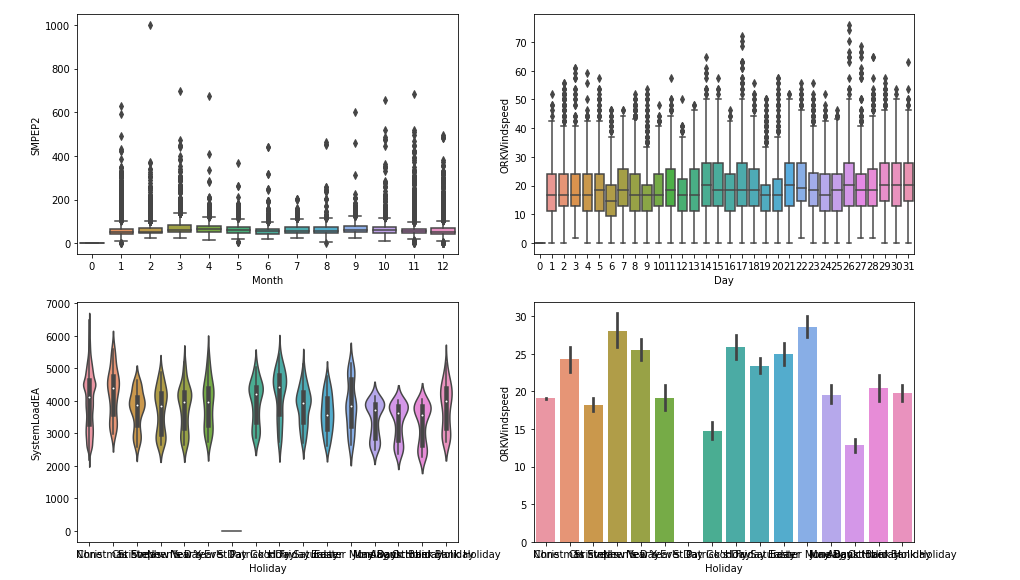
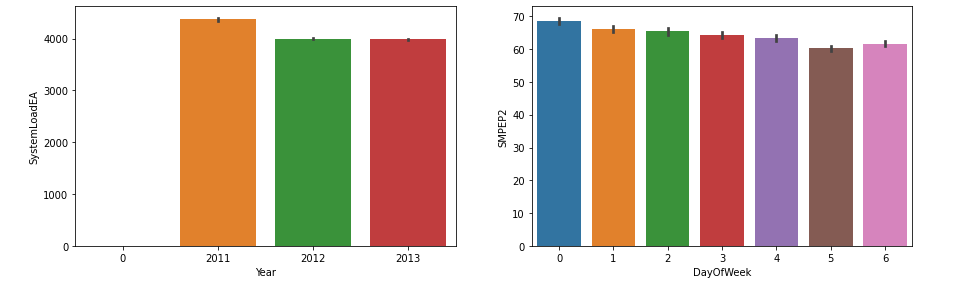
ridge=Ridge().fit(X\_train,y\_train)

lasso=Lasso().fit(X\_train,y\_train)

enet=ElasticNet().fit(X\_train,y\_train)

knn=KNeighborsRegressor().fit(X\_train,y\_train)

ada=AdaBoostRegressor().fit(X\_train,y\_train)



svm=SVR().fit(X\_train,y\_train)

mlpc=MLPRegressor().fit(X\_train,y\_train)

dtc=DecisionTreeRegressor().fit(X\_train,y\_train)

rf=RandomForestRegressor().fit(X\_train,y\_train)

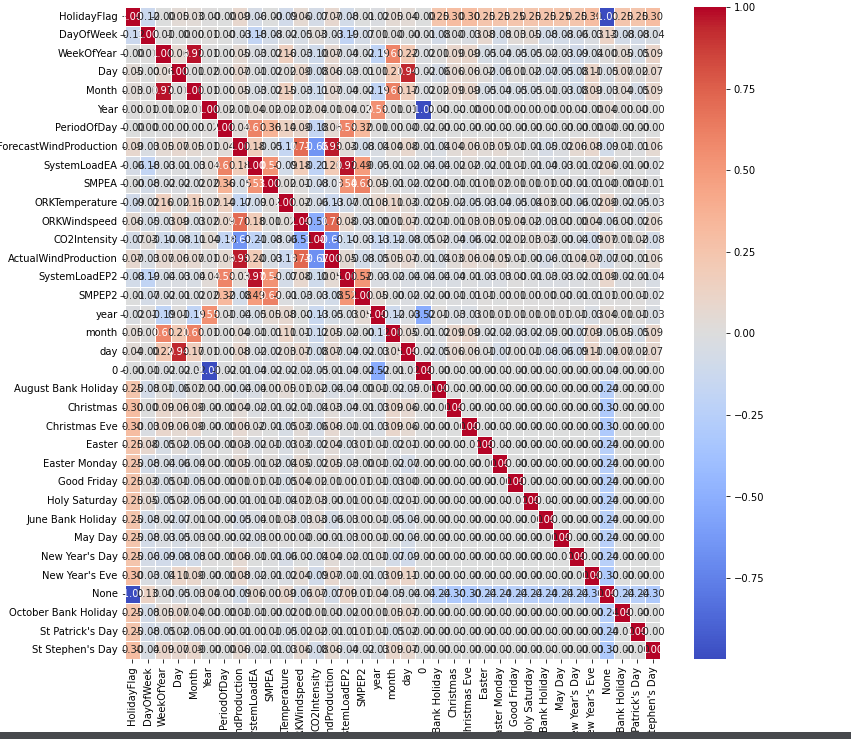
xgb=XGBRegressor().fit(X\_train,y\_train)

gbm=GradientBoostingRegressor().fit(X\_train,y\_train)

lgb=LGBMRegressor().fit(X\_train,y\_train)

catbost=CatBoostRegressor().fit(X\_train,y\_train)

models=[ridge,lasso,dtc,rf,xgb,gbm,lgb,catbost,enet,knn,ada,mlpc,svm]



def ML(y,models):

accuary=models.score(X\_train,y\_train)

return accuary

for i in models:

print(i,"Algorithm succed rate :",ML("SMPEP2",i))

cor=df.corr()["SMPEP2"].sort\_values(ascending=False)

pd.DataFrame({"column":cor.index,"Correlation with a":cor.values})

# let's reduce the number of variables and observe

X2=df[["SMPEA","SystemLoadEP2","SystemLoadEA","PeriodOfDay", "year","ActualWindProduction"]]

y2=df["SMPEP2"]

X\_train2,X\_test2,y\_train2,y\_test2=train\_test\_split(X2,y2,test\_size=0.3,random\_state=0)

rf2=RandomForestRegressor().fit(X\_train2,y\_train2)

rf2.score(X\_train2,y\_train2)

X3=df\_remove\_out.drop("SMPEP2",axis=1)

y3=df\_remove\_out["SMPEP2"]

X\_train3,X\_test3,y\_train3,y\_test3=train\_test\_split(X3,y3,test\_size=0.3,random\_state=0)

rf3=RandomForestRegressor().fit(X\_train3,y\_train3)

OUTPUT:

0.8976476391116986

dtc3=DecisionTreeRegressor().fit(X\_train3,y\_train3)

rf3.score(X\_train3,y\_train3)

OUTPUT:

0.9526290723016684

# Finally, let's perform optimization for rf (df in the initial state)

## 6.RandomForest

#hyperopth;

!pip install hyperopt

from hyperopt import tpe,STATUS\_OK,Trials,fmin,hp

from hyperopt.pyll.base import scope

space={

"max\_depth":hp.randint("max\_depth",2,15),

"min\_samples\_split":hp.randint("min\_samples\_split",2,20),

"min\_samples\_leaf":hp.randint("min\_samples\_leaf",1,20),

"n\_estimators":hp.randint("n\_estimators",50,1000)

}

def hyperparameter\_tuning(params):

clf=RandomForestRegressor(\*\*params).fit(X\_train,y\_train)

acc=rf.score(X\_train,y\_train)

return acc

trials=Trials()

best=fmin(fn=hyperparameter\_tuning,

space=space,

algo=tpe.suggest,max\_evals=100,trials=trials

)

print("best:{}".format(best))

# success rate 94.2%

OUTPUT:

20%|██ | 20/100 [19:27<57:20, 43.01s/trial, best loss: 0.9419941662794095]

## 7.Conclusion

* In this study, we tested various models for electricity price estimation and achieved high success rates...