Optimized Building Energy Consumption Prediction with ML

In [1]:
```python
# Import necessary libraries for data manipulation and visualization.
import pandas as pd  # Pandas for data handling
import numpy as np   # NumPy for numerical operations

import seaborn as sns           # Seaborn for data visualization
import matplotlib.pyplot as plt # Matplotlib for creating plots
%matplotlib inline

# Define a flag to control exporting, presumably for saving plots or data.
FLAG_EXPORT = True

# Define the output path where exported files will be saved.
out_path = 'assets/'
```

In [2]:
```python
# Read a CSV file named "dataset_climatic.csv" into a Pandas DataFrame.
raw1 = pd.read_csv("Data/dataset_climatic.csv", header=0)

# Rename the 'pression' column to 'pressure' for consistency or clarity.
raw1 = raw1.rename(columns={'pression': 'pressure'})

# Display the first few rows of the DataFrame to inspect the data.
raw1.head()
```

Out[2]:

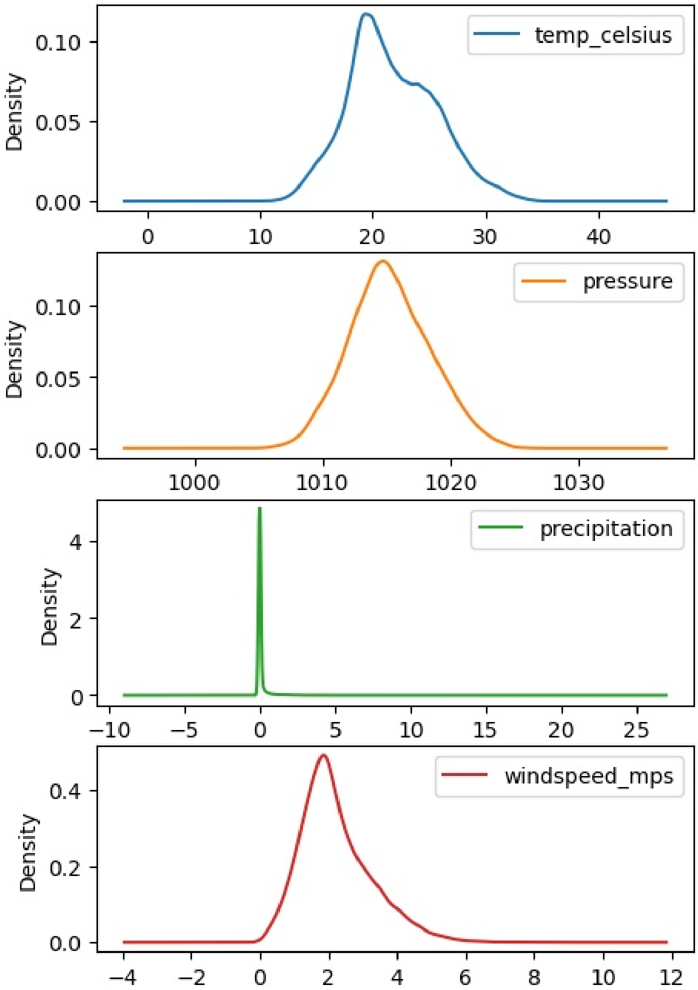
	timestamp	temp_celsius	pressure	precipitation	windspeed_mps
0	2018-01-31 00:00:00-02	21	1011.7	0.0	1.2
1	2018-01-31 01:00:00-02	21	1011.5	0.0	1.5
2	2018-01-31 02:00:00-02	20	1011.0	0.0	1.8
3	2018-01-31 03:00:00-02	20	1010.2	0.0	1.7
4	2018-01-31 04:00:00-02	20	1009.9	0.0	1.8

In [3]:
```python
# Generate summary statistics for the DataFrame 'raw1'.
raw1.describe()
```

Out[3]:

	temp_celsius	pressure	precipitation	windspeed_mps
count	16365.000000	16365.000000	16365.000000	16365.000000
mean	21.709502	1015.286227	0.093981	2.228555
std	3.870892	3.177252	0.504643	1.040920
min	10.000000	1005.100000	0.000000	0.000000
25%	19.000000	1013.100000	0.000000	1.500000
50%	21.000000	1015.100000	0.000000	2.000000
75%	24.000000	1017.400000	0.000000	2.800000
max	34.000000	1026.300000	18.000000	7.900000

In [4]:
```python
# Generate Kernel Density Estimation (KDE) plots for each numerical column in the DataFrame 'raw1'.
raw1.plot.kde(subplots=True, sharex=False, figsize=(5, 8), layout=(4, 1))

# Check if the FLAG_EXPORT is set to True (presumably for exporting plots).
if FLAG_EXPORT:
    # Save the generated KDE plots as a PDF file named "KDE_climaTIC.pdf".
    plt.savefig("KDE_climaTIC.pdf", format='pdf')
```



In [5]:

```
# Convert the 'timestamp' column to a datetime format, assuming the timestamps are in UTC.
raw1['timestamp'] = pd.to_datetime(raw1['timestamp'], utc=True)

# Set the DataFrame index to the 'timestamp' column.
raw1 = raw1.set_index(raw1['timestamp'])

# Drop the 'timestamp' column as it's now the index.
raw1 = raw1.drop('timestamp', axis=1)

# Adjust the index to UTC-2 (2 hours behind UTC) using tz_convert.
raw1 = raw1.set_index(raw1.index.tz_convert(None) + pd.offsets.Hour(-2))

# Print the first few rows of the DataFrame after these transformations.
print(raw1.head())
```

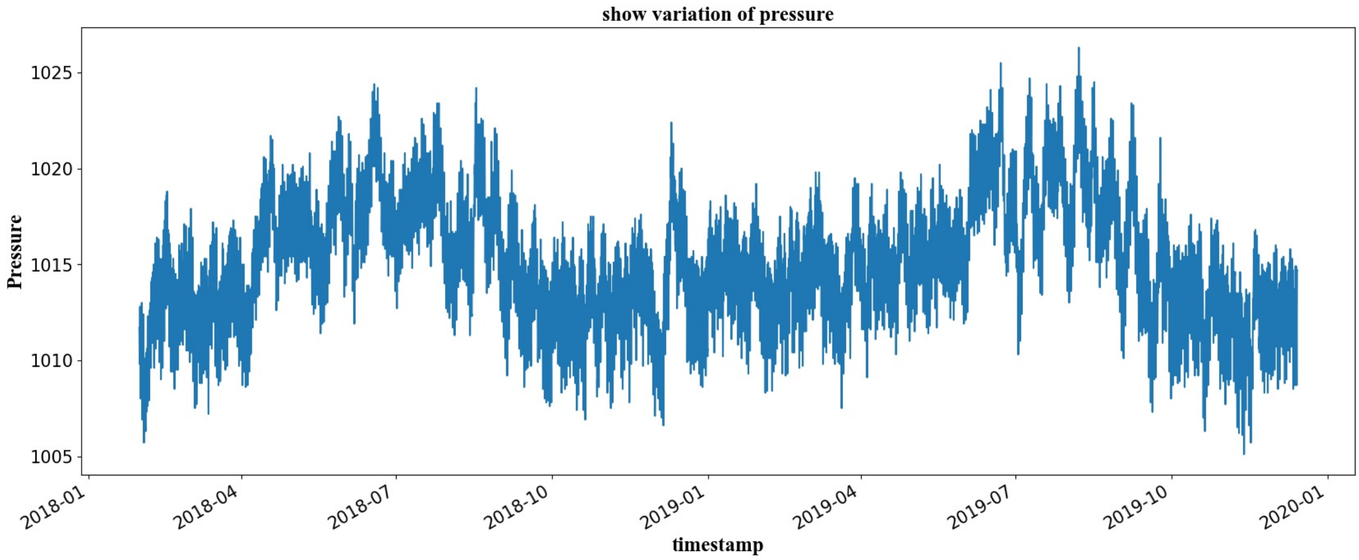
```
                     temp_celsius  pressure  precipitation  windspeed_mps
timestamp
2018-01-31 00:00:00            21    1011.7            0.0            1.2
2018-01-31 01:00:00            21    1011.5            0.0            1.5
2018-01-31 02:00:00            20    1011.0            0.0            1.8
2018-01-31 03:00:00            20    1010.2            0.0            1.7
2018-01-31 04:00:00            20    1009.9            0.0            1.8
```

In [6]:

```
# Create a new figure for the plot with a specific size.
plt.figure(figsize=(20, 8))

# Set the y-axis label with custom font properties.
plt.ylabel("Pressure", fontsize=18, fontname="Times New Roman", fontweight="bold")

# Set the x-axis label with custom font properties.
plt.xlabel("Date", fontsize=18, fontname="Times New Roman", fontweight="bold")

# Set the title of the plot with custom font properties.
plt.title("Variation of Pressure", fontsize=18, fontname="Times New Roman", fontweight="bold")

# Replace any NaN values in the 'pressure' column with 0.
raw1["pressure"] = raw1["pressure"].replace(np.NaN, 0)

# Print the data type of the 'pressure' column.
print(raw1["pressure"].dtype)

# Generate a basic plot of the 'pressure' column.
raw1["pressure"].plot()
```

```
# Customize tick parameters for both axes with a larger font size.
plt.tick_params(axis='both', which='major', labelsize=15)

# Save the plot as a PNG and PDF file in the current directory.
plt.savefig('Electricity_Price.png', format='png')
plt.savefig('Electricity_Price.pdf', format='pdf')
```

```
float64
```



In [7]:
```
# Resample the 'raw1' DataFrame at 10-minute intervals and interpolate missing values using linear interpolatio
# The 'limit_area' parameter specifies that interpolation should occur only inside data boundaries.
filtered1 = raw1.resample('10min').interpolate(method='linear', limit_area='inside')
```

Electrical database exploration

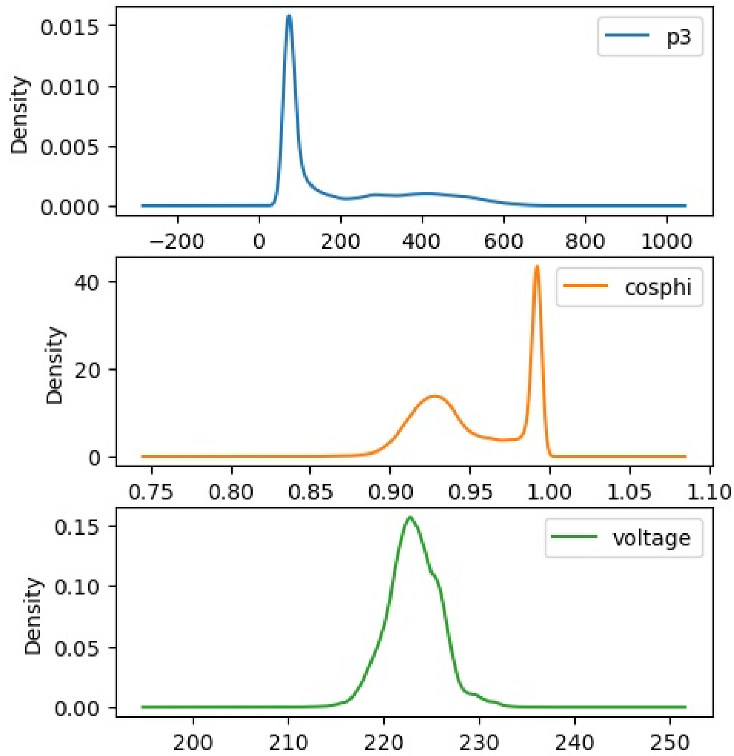
- Convert index to timestamp
- Resampled by hour
- Calculate load factor

In [8]:
```
# Read another CSV file named "dataset_electric.csv" into a new Pandas DataFrame 'raw2'.
raw2 = pd.read_csv("dataset_electric.csv", header=0)

# Drop the 's3' column from the DataFrame 'raw2'.
raw2 = raw2.drop('s3', axis=1)

# Generate summary statistics for the DataFrame 'raw2'.
raw2.describe()
```

Out[8]:

	p3	cosphi	voltage	month	hour	dayofweek
count	857147.000000	857147.000000	857147.000000	987841.000000	987841.000000	987841.000000
mean	183.425452	0.953158	223.184563	6.672017	11.499988	3.000001
std	155.860108	0.032707	2.766074	3.268192	6.922196	2.000000
min	48.737500	0.830174	208.966000	1.000000	0.000000	0.000000
25%	74.080833	0.925084	221.431000	4.000000	5.000000	1.000000
50%	89.919167	0.946635	223.134667	7.000000	11.000000	3.000000
75%	287.483155	0.990457	224.994306	9.000000	17.000000	5.000000
max	713.428333	0.999890	237.393444	12.000000	23.000000	6.000000

In [9]:
```
# Generate Kernel Density Estimation (KDE) plots for specific columns in the 'raw2' DataFrame.
raw2[['p3', 'cosphi', 'voltage']].plot.kde(subplots=True, sharex=False, figsize=(5, 6), layout=(3, 1))

# Check if the FLAG_EXPORT is set to True (presumably for exporting plots).
if FLAG_EXPORT:
    # Save the generated KDE plots as a PDF file named 'KDEelectric.pdf'.
    plt.savefig('KDEelectric.pdf', format='pdf')
```



In [10]:
```
# Convert the 'timestamp' column in 'raw2' to a datetime format.
raw2['timestamp'] = pd.to_datetime(raw2['timestamp'])

# Set the DataFrame index to the 'timestamp' column.
raw2 = raw2.set_index(raw2['timestamp'])

# Drop the 'timestamp' column as it's now the index.
raw2 = raw2.drop('timestamp', axis=1)
```

In [11]:
```
# Resample the 'raw2' DataFrame at 10-minute intervals and aggregate data using various functions.
resampled2 = raw2.resample('10min').agg({
    'voltage': ['mean', 'count'],    # Calculate mean and count for 'voltage'
    'cosphi': ['mean', 'std'],       # Calculate mean and standard deviation for 'cosphi'
    'month': ['mean'],               # Calculate mean for 'month'
    'hour': ['mean'],                # Calculate mean for 'hour'
    'dayofweek': ['mean'],           # Calculate mean for 'dayofweek'
    'p3': ['mean', 'max', 'std']     # Calculate mean, max, and standard deviation for 'p3'
})

# Join multi-level column names into a single level.
resampled2.columns = resampled2.columns.map('_'.join)

# Drop rows with NaN values.
resampled2 = resampled2.dropna()
```

In [12]:
```
# Filter the 'resampled2' DataFrame to select rows where 'voltage_count' is equal to 10.
filtered2 = resampled2[resampled2['voltage_count'] == 10]

# Drop the 'voltage_count' column from the filtered DataFrame.
filtered2 = filtered2.drop('voltage_count', axis=1)
```

In [13]:
```
# Calculate the 'load_factor' by dividing the mean of 'p3_mean' by the max of 'p3_max'.
filtered2['load_factor'] = filtered2['p3_mean'] / filtered2['p3_max']
```

In [14]:
```
# Drop the 'p3_max' column from the 'filtered2' DataFrame.
filtered2 = filtered2.drop('p3_max', axis=1)
```

In [15]:
```
filtered2.head()
```

Out[15]:

timestamp	voltage_mean	cosphi_mean	cosphi_std	month_mean	hour_mean	dayofweek_mean	p3_mean	p3_std	load_factor
2018-02-01 00:00:00	222.422553	0.931323	0.003132	2.0	0.0	4.0	76.384250	2.901398	0.936588
2018-02-01 00:10:00	221.822300	0.938584	0.003327	2.0	0.0	4.0	79.409299	4.679058	0.916590
2018-02-01 00:20:00	222.539326	0.933304	0.004393	2.0	0.0	4.0	74.200246	3.366907	0.950006
2018-02-01 00:30:00	222.744070	0.934903	0.005175	2.0	0.0	4.0	76.059915	2.742411	0.949632
2018-02-01 00:40:00	222.832179	0.936927	0.003765	2.0	0.0	4.0	77.216275	2.191767	0.963748

Data integration

In [16]:
```
# Merge the 'filtered1' and 'filtered2' DataFrames using an inner join based on their indices.
# Also, cast the merged DataFrame to a float data type.
merged = pd.merge(filtered1, filtered2, how='inner', left_index=True, right_index=True).astype('float')
```

In [17]:
```
# Set values in the 'precipitation' column to 0 where they are less than 0.
merged.loc[merged['precipitation'] < 0, 'precipitation'] = 0
```

In [18]:
```
# Rename columns in the 'merged' DataFrame to more descriptive names.
merged = merged.rename(columns={
    'temp_celsius': 'temperature',        # Rename 'temp_celsius' to 'temperature'
    'pressure': 'pressure',              # Rename 'pressure' to 'pressure'
    'precipitation': 'precipitation',    # Rename 'precipitation' to 'precipitation'
    'windspeed_mps': 'windspeed',        # Rename 'windspeed_mps' to 'windspeed'
    'voltage_mean': 'voltage',           # Rename 'voltage_mean' to 'voltage'
    'cosphi_mean': 'cos_phi',            # Rename 'cosphi_mean' to 'cos_phi'
    'cosphi_std': 'cos_phi_std',         # Rename 'cosphi_std' to 'cos_phi_std'
    'load_factor': 'load_factor',        # Rename 'load_factor' to 'load_factor'
    'month_mean': 'month',               # Rename 'month_mean' to 'month'
    'dayofweek_mean': 'day_of_week',     # Rename 'dayofweek_mean' to 'day_of_week'
    'hour_mean': 'hour',                # Rename 'hour_mean' to 'hour'
    'p3_std': 'p3_std',                  # Rename 'p3_std' to 'p3_std'
    'p3_mean': 'p3'                     # Rename 'p3_mean' to 'p3'
})
```

In [19]:
```
# Check if the FLAG_EXPORT is set to True (presumably for exporting data statistics).
if FLAG_EXPORT:
    # Generate summary statistics for the 'merged' DataFrame and save them to a LaTeX table file.
    merged.describe().to_latex('table_dataset_stats.tex')
```

```
C:\Users\Guest1\AppData\Local\Temp\ipykernel_11368\1885669551.py:1: FutureWarning: In future versions `DataFrame
.to_latex` is expected to utilise the base implementation of `Styler.to_latex` for formatting and rendering. The
arguments signature may therefore change. It is recommended instead to use `DataFrame.style.to_latex` which also
contains additional functionality.
  if FLAG_EXPORT: merged.describe().to_latex('table_dataset_stats.tex')
```

In [20]:
```
merged.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 84926 entries, 2018-02-01 00:00:00 to 2019-12-13 23:00:00
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   temperature    84926 non-null  float64
 1   pressure       84926 non-null  float64
 2   precipitation  84926 non-null  float64
 3   windspeed      84926 non-null  float64
 4   voltage        84926 non-null  float64
 5   cos_phi        84926 non-null  float64
 6   cos_phi_std    84926 non-null  float64
 7   month          84926 non-null  float64
 8   hour           84926 non-null  float64
 9   day_of_week    84926 non-null  float64
 10  p3             84926 non-null  float64
 11  p3_std         84926 non-null  float64
 12  load_factor    84926 non-null  float64
dtypes: float64(13)
memory usage: 9.1 MB
```

In [21]:

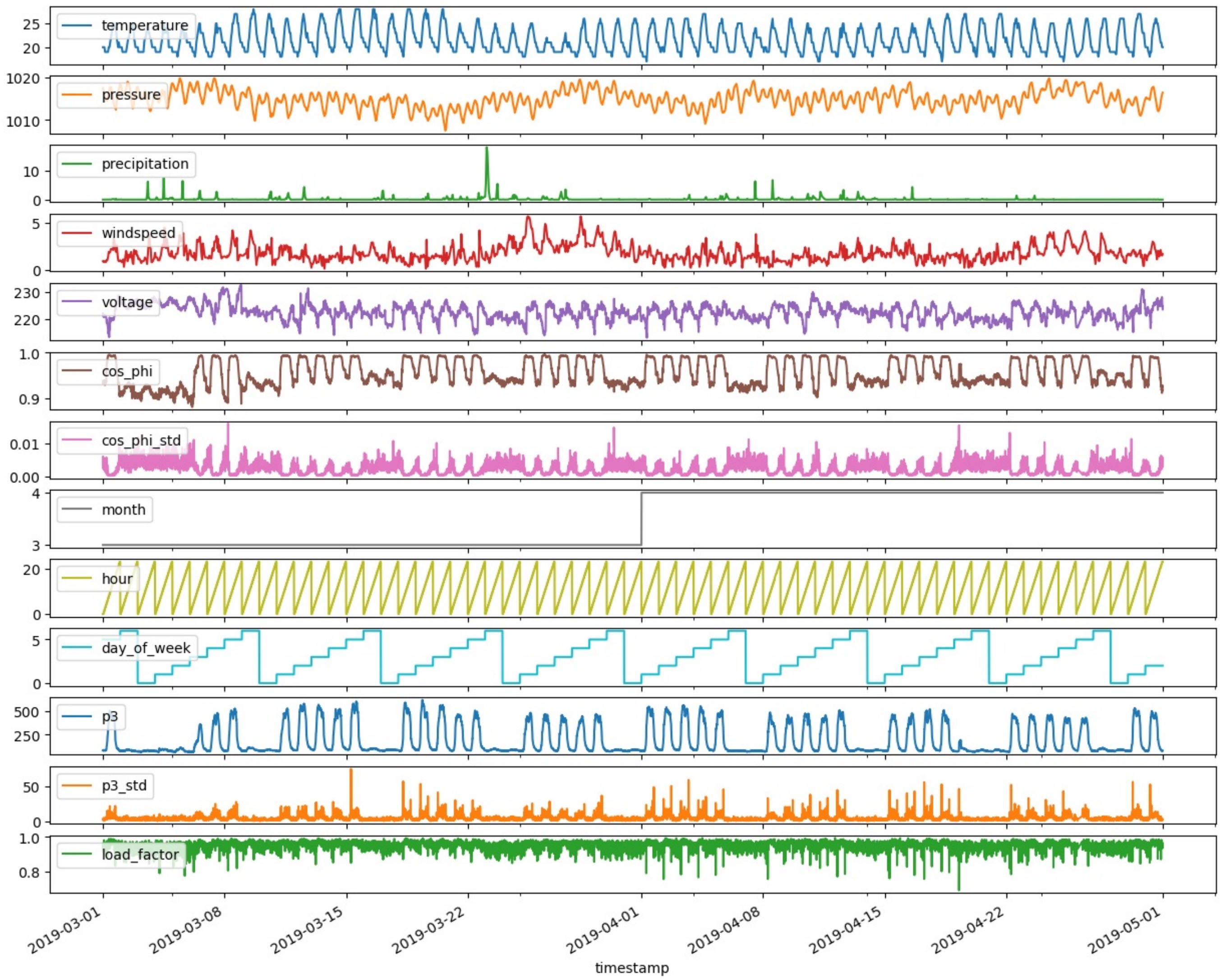
```
# Extract and display the first few rows of data from 'merged' DataFrame
# that fall within the date range from March 1, 2019, at 00:00:00 to May 1, 2019, at 00:00:00.
subset_data = merged['2019-03-01 00:00:00':'2019-05-01 00:00:00'].head()
```

Out[21]:

	temperature	pressure	precipitation	windspeed	voltage	cos_phi	cos_phi_std	month	hour	day_of_week	
timestamp											
2019-03-01 00:00:00	20.0	1017.500000	0.0	0.900000	222.118755	0.933534	0.002797	3.0	0.0	5.0	78.
2019-03-01 00:10:00	20.0	1017.383333	0.0	0.883333	221.155083	0.938093	0.006035	3.0	0.0	5.0	81.
2019-03-01 00:20:00	20.0	1017.266667	0.0	0.866667	221.255644	0.933861	0.002610	3.0	0.0	5.0	78.
2019-03-01 00:30:00	20.0	1017.150000	0.0	0.850000	221.314422	0.937841	0.002337	3.0	0.0	5.0	81.
2019-03-01 00:40:00	20.0	1017.033333	0.0	0.833333	221.310550	0.935942	0.002764	3.0	0.0	5.0	78.

In [22]:

```
# Create subplots for each column within the specified date range and plot them.
Axis = merged['2019-03-01 00:00:00':'2019-04-30 23:59:59'].plot(subplots=True, sharex=True, figsize=(15, 13))

# Add legends to each subplot, positioning them at the upper left corner.
for k in range(0, merged.shape[1], 1):
    Axis[k].legend(loc='upper left')

# Check if the FLAG_EXPORT is set to True (presumably for exporting the plot).
if FLAG_EXPORT:
    # Save the plot as an SVG file named 'graph_temporal.svg'.
    plt.savefig('graph_temporal.svg', format='svg')
```



In [23]:
```
merged.describe()
```

Out[23]:

	temperature	pressure	precipitation	windspeed	voltage	cos_phi	cos_phi_std	month	
count	84926.000000	84926.000000	84926.000000	84926.000000	84926.000000	84926.000000	84926.000000	84926.000000	84926.00
mean	21.606675	1015.557522	0.078656	2.249284	223.184579	0.953134	0.002702	6.204814	11.53
std	3.846077	3.098239	0.426773	1.023453	2.747639	0.032593	0.002173	3.113732	6.92
min	10.000000	1005.700000	0.000000	0.000000	210.030339	0.853330	0.000076	1.000000	0.00
25%	19.000000	1013.433333	0.000000	1.550000	221.439586	0.924797	0.000692	4.000000	6.00
50%	21.000000	1015.416667	0.000000	2.050000	223.130786	0.946207	0.002361	6.000000	12.00
75%	24.333333	1017.633333	0.000000	2.800000	224.990057	0.990512	0.004066	9.000000	18.00
max	34.000000	1026.300000	18.000000	7.900000	236.857867	0.997968	0.026059	12.000000	23.00

In [24]:
```
merged.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 84926 entries, 2018-02-01 00:00:00 to 2019-12-13 23:00:00
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   temperature    84926 non-null  float64
 1   pressure       84926 non-null  float64
 2   precipitation  84926 non-null  float64
 3   windspeed      84926 non-null  float64
 4   voltage        84926 non-null  float64
 5   cos_phi        84926 non-null  float64
 6   cos_phi_std    84926 non-null  float64
 7   month          84926 non-null  float64
 8   hour           84926 non-null  float64
 9   day_of_week    84926 non-null  float64
 10  p3             84926 non-null  float64
 11  p3_std         84926 non-null  float64
 12  load_factor    84926 non-null  float64
dtypes: float64(13)
memory usage: 11.1 MB
```

In [25]:
```python
# Math and linear algebra
import pandas as pd
import math
import numpy as np

# Vizualization
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# Utils
import sklearn.metrics as metrics
from sklearn.model_selection import GridSearchCV
from joblib import dump, load
import gc
```

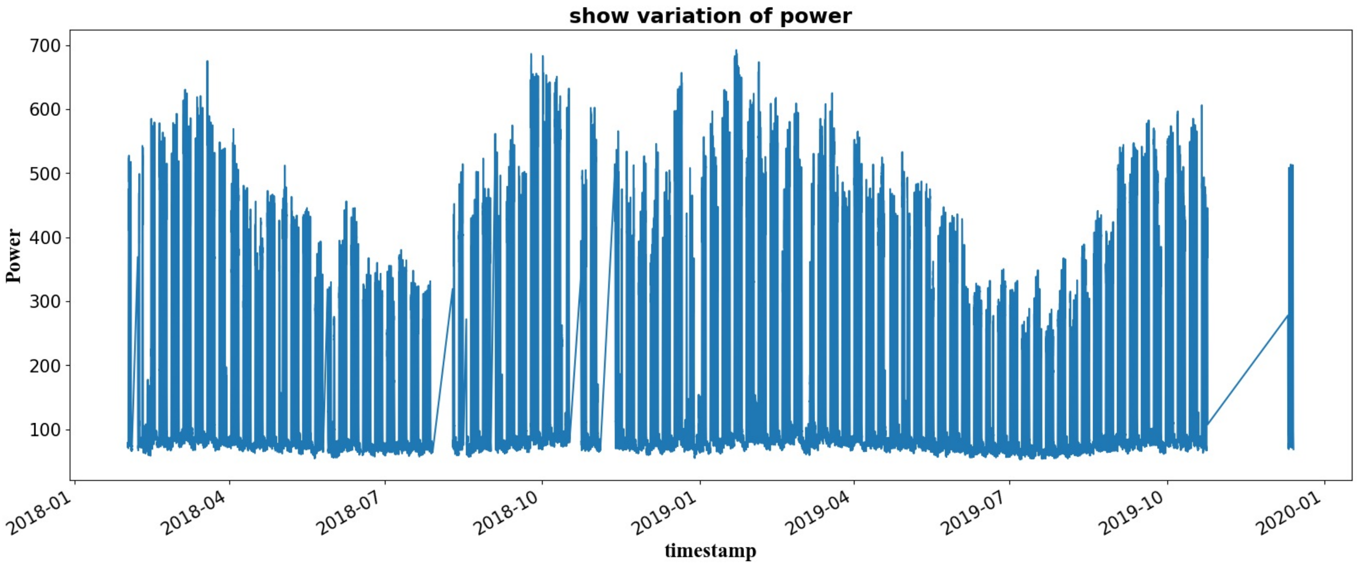
In [26]:
```python
merged.head()
```

Out[26]:

	temperature	pressure	precipitation	windspeed	voltage	cos_phi	cos_phi_std	month	hour	day_of_week	
timestamp											
2018-02-01 00:00:00	21.000000	1012.4	0.200000	2.000000	222.422553	0.931323	0.003132	2.0	0.0	4.0	76.384
2018-02-01 00:10:00	20.833333	1012.3	0.216667	2.016667	221.822300	0.938584	0.003327	2.0	0.0	4.0	79.409
2018-02-01 00:20:00	20.666667	1012.2	0.233333	2.033333	222.539326	0.933304	0.004393	2.0	0.0	4.0	74.200
2018-02-01 00:30:00	20.500000	1012.1	0.250000	2.050000	222.744070	0.934903	0.005175	2.0	0.0	4.0	76.059
2018-02-01 00:40:00	20.333333	1012.0	0.266667	2.066667	222.832179	0.936927	0.003765	2.0	0.0	4.0	77.216

In [27]:
```python
# Create a new figure for the plot with a specific size.
plt.figure(figsize=(20, 8))

# Set the y-axis label with custom font properties.
plt.ylabel("Power", fontsize=18, fontname="Times New Roman", fontweight="bold")

# Set the x-axis label with custom font properties.
plt.xlabel("Date", fontsize=18, fontname="Times New Roman", fontweight="bold")

# Set the title of the plot with custom font properties.
plt.title("Variation of Power", fontsize=18, fontweight="bold")

# Replace any NaN values in the 'p3' column with 0.
merged["p3"] = merged["p3"].replace(np.NaN, 0)

# Print the data type of the 'pressure' column.
print(merged["pressure"].dtype)

# Generate a basic plot of the 'p3' column.
merged["p3"].plot()

# Customize tick parameters for both axes with a larger font size.
plt.tick_params(axis='both', which='major', labelsize=15)

# Save the plot as both a PNG and PDF file in the current directory.
plt.savefig('Electricity_Price.png', format='png')
plt.savefig('Electricity_Price.pdf', format='pdf')
```

```
float64
```



In [28]:
```
merged.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 84926 entries, 2018-02-01 00:00:00 to 2019-12-13 23:00:00
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   temperature    84926 non-null  float64
 1   pressure       84926 non-null  float64
 2   precipitation  84926 non-null  float64
 3   windspeed      84926 non-null  float64
 4   voltage        84926 non-null  float64
 5   cos_phi        84926 non-null  float64
 6   cos_phi_std    84926 non-null  float64
 7   month          84926 non-null  float64
 8   hour           84926 non-null  float64
 9   day_of_week    84926 non-null  float64
 10  p3             84926 non-null  float64
 11  p3_std         84926 non-null  float64
 12  load_factor    84926 non-null  float64
dtypes: float64(13)
memory usage: 11.1 MB
```

In [29]:
```
merged.columns
```

Out[29]:
```
Index(['temperature', 'pressure', 'precipitation', 'windspeed', 'voltage',
       'cos_phi', 'cos_phi_std', 'month', 'hour', 'day_of_week', 'p3',
       'p3_std', 'load_factor'],
      dtype='object')
```

In [30]:
```
merged.describe()
```

Out[30]:

	temperature	pressure	precipitation	windspeed	voltage	cos_phi	cos_phi_std	month	
count	84926.000000	84926.000000	84926.000000	84926.000000	84926.000000	84926.000000	84926.000000	84926.000000	84926.00
mean	21.606675	1015.557522	0.078656	2.249284	223.184579	0.953134	0.002702	6.204814	11.53
std	3.846077	3.098239	0.426773	1.023453	2.747639	0.032593	0.002173	3.113732	6.92
min	10.000000	1005.700000	0.000000	0.000000	210.030339	0.853330	0.000076	1.000000	0.00
25%	19.000000	1013.433333	0.000000	1.550000	221.439586	0.924797	0.000692	4.000000	6.00
50%	21.000000	1015.416667	0.000000	2.050000	223.130786	0.946207	0.002361	6.000000	12.00
75%	24.333333	1017.633333	0.000000	2.800000	224.990057	0.990512	0.004066	9.000000	18.00
max	34.000000	1026.300000	18.000000	7.900000	236.857867	0.997968	0.026059	12.000000	23.00

In [31]:

```
from sklearn.preprocessing import LabelBinarizer
from sklearn.feature_selection import SelectKBest, chi2



# Separate the features and target variable
X = merged.drop(columns=["p3"])
y = merged["p3"]

# Reset the index of the y variable
# y = y.reset_index(drop=True)

# Convert the target variable to binary labels
y_binary = np.where(y >= y.mean(), 1, 0)

lb = LabelBinarizer()
y = lb.fit_transform(y_binary)

# Select the K best features using chi-squared score
selector = SelectKBest(score_func=chi2, k=10)
X_new = selector.fit_transform(X, y)

# Get the selected features
selected_features = X.columns[selector.get_support(indices=True)]
selected_features
```

Out[31]:

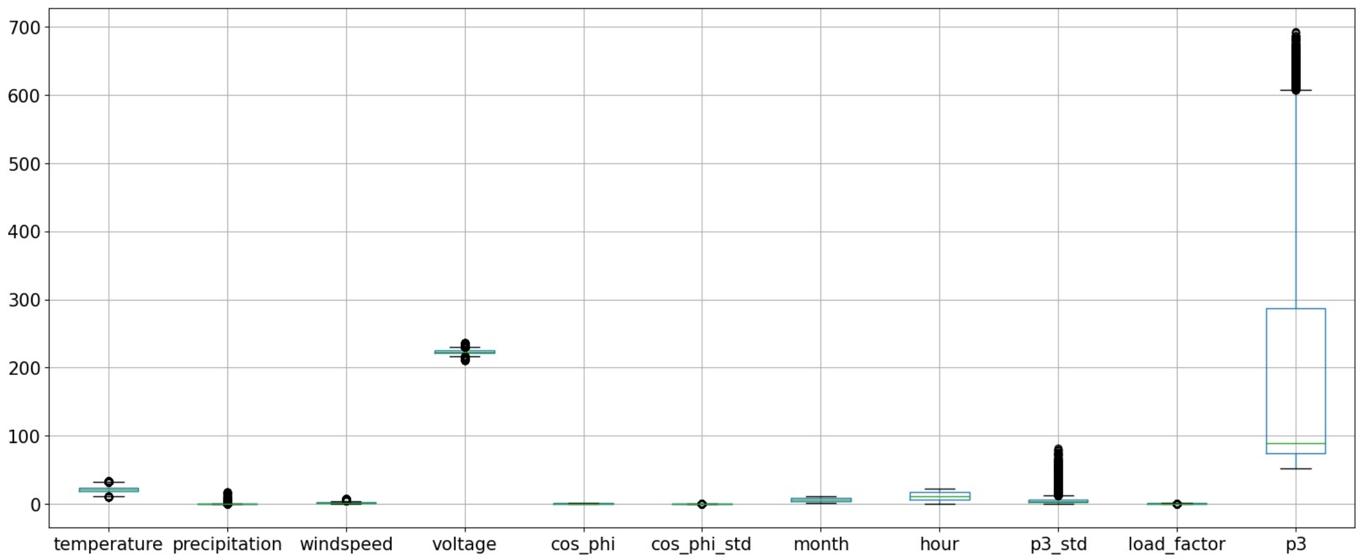
```
Index(['temperature', 'precipitation', 'windspeed', 'voltage', 'cos_phi',
       'cos_phi_std', 'month', 'hour', 'p3_std', 'load_factor'],
      dtype='object')
```

In [32]:

```
merged = merged[['temperature', 'precipitation', 'windspeed', 'voltage', 'cos_phi',
       'cos_phi_std', 'month', 'hour', 'p3_std', 'load_factor','p3']]
```

In [33]:

```
plt.figure(figsize=(20,8))
# Box Plot
import seaborn as sns
boxplot = merged.boxplot(column=['temperature', 'precipitation', 'windspeed', 'voltage', 'cos_phi',
       'cos_phi_std', 'month', 'hour', 'p3_std', 'load_factor','p3'])
plt.tick_params(axis='both', which='major', labelsize=15)
plt.savefig('Outliers_data.png', format='png')
plt.savefig('Outliers_data.pdf', format='pdf')
```



In [34]:

```
for col in merged.columns:
    if col != 'p3':
```

```python
        median = merged[col].median()
        std = merged[col].std()
        lower_bound = median - 3*std
        upper_bound = median + 3*std
        merged.loc[(merged[col] < lower_bound) | (merged[col] > upper_bound), col] = median

# show the new data
print(merged)
```

```
                     temperature  precipitation  windspeed     voltage  \
timestamp
2018-02-01 00:00:00    21.000000       0.200000   2.000000  222.422553
2018-02-01 00:10:00    20.833333       0.216667   2.016667  221.822300
2018-02-01 00:20:00    20.666667       0.233333   2.033333  222.539326
2018-02-01 00:30:00    20.500000       0.250000   2.050000  222.744070
2018-02-01 00:40:00    20.333333       0.266667   2.066667  222.832179
...                          ...            ...        ...         ...
2019-12-13 22:20:00    21.666667       0.000000   0.366667  225.512978
2019-12-13 22:30:00    21.500000       0.000000   0.450000  226.156717
2019-12-13 22:40:00    21.333333       0.000000   0.533333  226.981687
2019-12-13 22:50:00    21.166667       0.000000   0.616667  226.475511
2019-12-13 23:00:00    21.000000       0.000000   0.700000  226.586756

                      cos_phi  cos_phi_std  month  hour    p3_std  \
timestamp
2018-02-01 00:00:00  0.931323     0.003132    2.0   0.0  2.901398
2018-02-01 00:10:00  0.938584     0.003327    2.0   0.0  4.679058
2018-02-01 00:20:00  0.933304     0.004393    2.0   0.0  3.366907
2018-02-01 00:30:00  0.934903     0.005175    2.0   0.0  2.742411
2018-02-01 00:40:00  0.936927     0.003765    2.0   0.0  2.191767
...                       ...          ...    ...   ...       ...
2019-12-13 22:20:00  0.930168     0.002083   12.0  22.0  2.059276
2019-12-13 22:30:00  0.928147     0.005057   12.0  22.0  6.957287
2019-12-13 22:40:00  0.924425     0.003074   12.0  22.0  1.536393
2019-12-13 22:50:00  0.921028     0.002361   12.0  22.0  6.269609
2019-12-13 23:00:00  0.920921     0.004348   12.0  23.0  2.256251

                     load_factor         p3
timestamp
2018-02-01 00:00:00     0.936588  76.384250
2018-02-01 00:10:00     0.916590  79.409299
2018-02-01 00:20:00     0.950006  74.200246
2018-02-01 00:30:00     0.949632  76.059915
2018-02-01 00:40:00     0.963748  77.216275
...                          ...        ...
2019-12-13 22:20:00     0.940818  70.696167
2019-12-13 22:30:00     0.953135  74.598750
2019-12-13 22:40:00     0.970661  70.590538
2019-12-13 22:50:00     0.916546  70.484667
2019-12-13 23:00:00     0.961968  68.601167

[84926 rows x 11 columns]
```

In [35]:
```python
merged.shape
```

Out[35]:
```
(84926, 11)
```

In [36]:
```python
merged = merged[merged['p3'] <= 105]
```

In [37]:
```python
merged = merged[merged['p3_std'] <= 10]
```

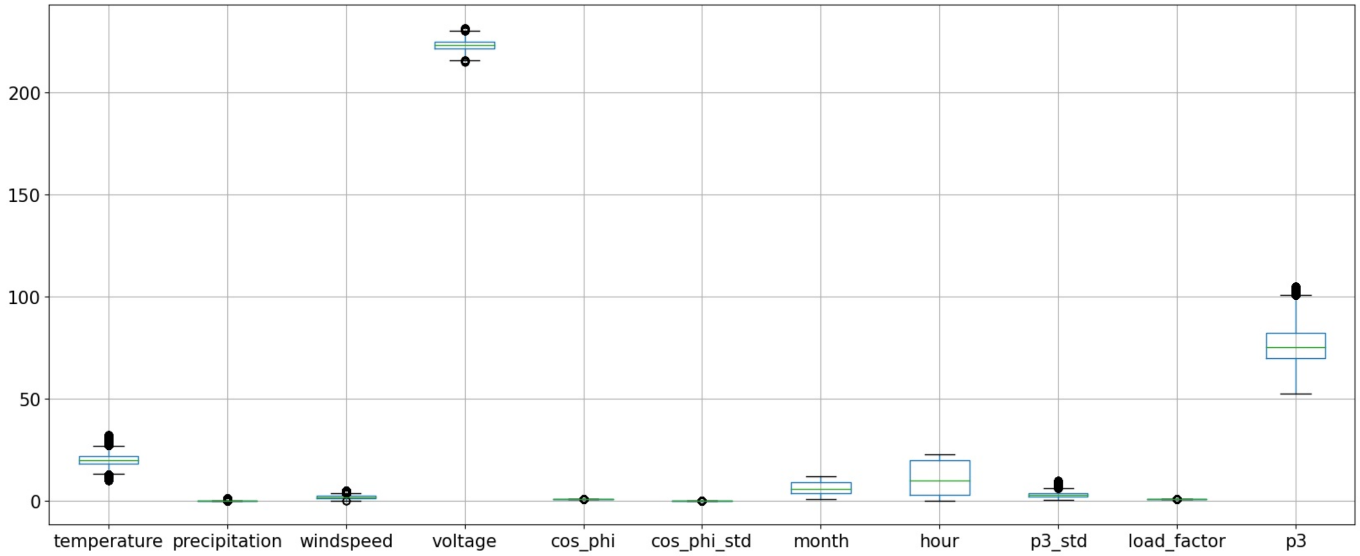
In [38]:
```python
merged.describe()
```

Out[38]:

	temperature	precipitation	windspeed	voltage	cos_phi	cos_phi_std	month	hour	p3
count	47932.000000	47932.000000	47932.000000	47932.000000	47932.000000	47932.000000	47932.000000	47932.000000	47932.00
mean	20.437345	0.035512	2.042403	223.216483	0.927248	0.003949	6.254465	10.943921	3.18
std	3.312441	0.141484	0.821039	2.669233	0.016025	0.001641	3.077895	8.368517	1.29
min	10.000000	0.000000	0.000000	214.937500	0.853330	0.000389	1.000000	0.000000	0.42
25%	18.458333	0.000000	1.500000	221.371107	0.916615	0.002678	4.000000	3.000000	2.23
50%	20.000000	0.000000	1.933333	223.156061	0.927100	0.003708	6.000000	10.000000	2.98
75%	22.000000	0.000000	2.433333	224.984613	0.937587	0.004980	9.000000	20.000000	3.92
max	32.500000	1.266667	5.116667	231.370206	0.985139	0.008877	12.000000	23.000000	9.94

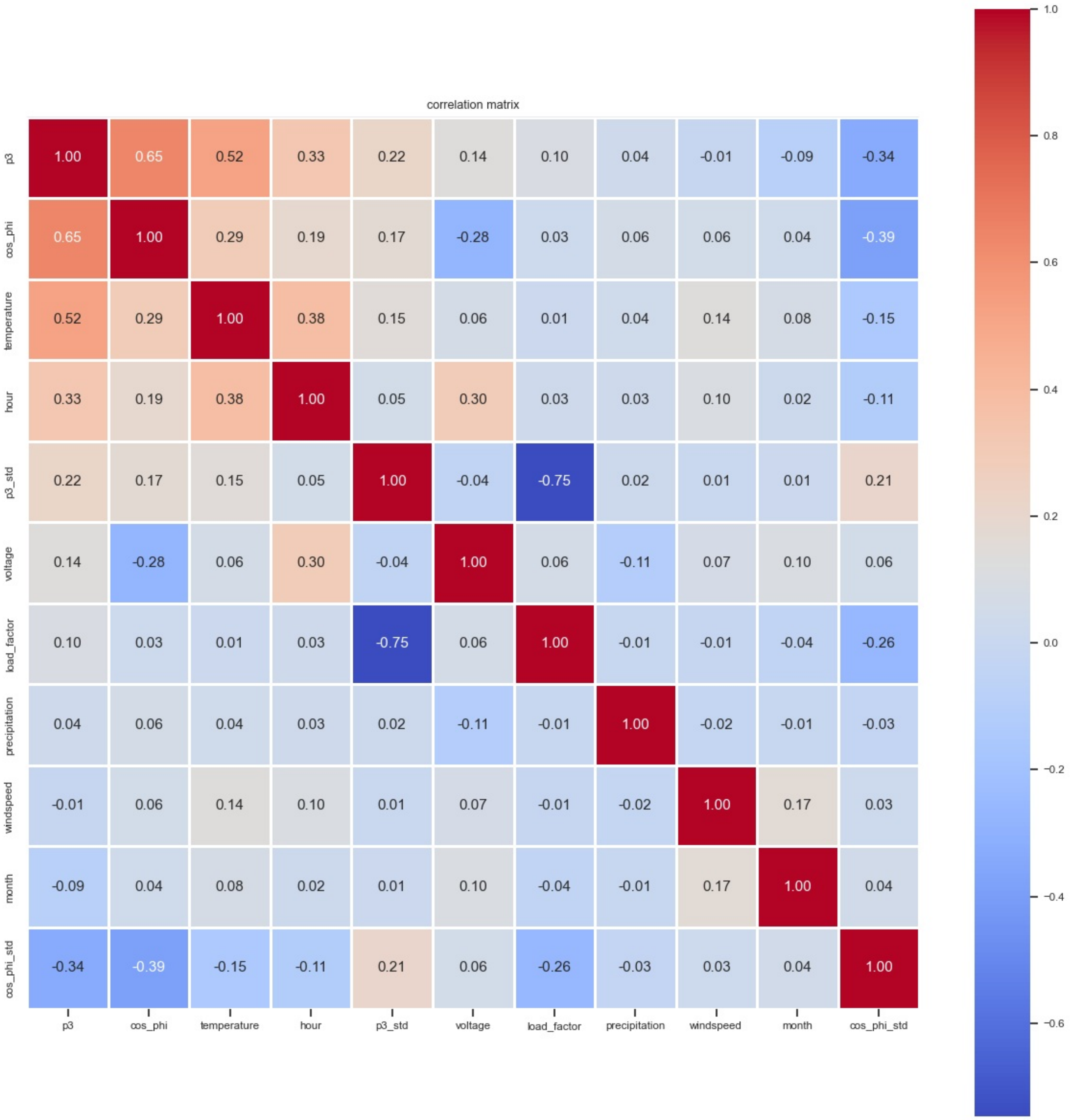
In [39]:
```python
plt.figure(figsize=(20,8))
# Box Plot
import seaborn as sns
```

```
boxplot = merged.boxplot(column=['temperature', 'precipitation', 'windspeed', 'voltage', 'cos_phi',
       'cos_phi_std', 'month', 'hour', 'p3_std', 'load_factor','p3'])
plt.tick_params(axis='both', which='major', labelsize=15)
plt.savefig('Outliers_data.png', format='png')
plt.savefig('Outliers_data.pdf', format='pdf')
```



In [40]:
```
# check correlations of features with price
df_corr = merged.corr(method="pearson")
print(df_corr.shape)
print("correlation with p3:")
df_corrP = pd.DataFrame(df_corr["p3"].sort_values(ascending=False))
print(df_corrP)

# correlation matrix, limited to highly correlated features
df3 = merged[df_corrP.index]

idx = df3.corr().sort_values("p3", ascending=False).index
df3_sorted = df3.loc[:, idx]  # sort dataframe columns by their correlation with Appliances

import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(15, 15))
sns.set(font_scale=0.75)
ax = sns.heatmap(df3_sorted.corr().round(3),
                 annot=True,
                 square=True,
                 linewidths=.75,
                 cmap="coolwarm",
                 fmt=".2f",
                 annot_kws={"size": 11})
ax.xaxis.tick_bottom()

plt.title("correlation matrix")
plt.savefig('Correlation.png', format='png')
plt.savefig('Correlation.pdf', format='pdf')
plt.show()
```

```
(11, 11)
correlation with p3:
                     p3
p3             1.000000
cos_phi        0.644984
temperature    0.520951
hour           0.330611
p3_std         0.217955
voltage        0.137490
load_factor    0.099053
precipitation  0.035236
windspeed     -0.014505
month         -0.089774
cos_phi_std   -0.336197
```




```
In [41]: X = df3.drop('p3', axis=1).values
         y = df3['p3'].values
         print(X.shape)
         print(y.shape)
         y1 = y.reshape(-1, 1)
```

```
(47932, 10)
(47932,)
```

```
In [42]: from sklearn.preprocessing import StandardScaler
         s = StandardScaler()
         s1 = StandardScaler()

         # standardization
         X1 = s.fit_transform(X)
         y_2d = y.reshape(-1, 1) # Reshape y to a 2D array with a single column
         y1 = s1.fit_transform(y_2d)
         print(X1.shape)
```

```
print(y1.shape)
```

```
(47932, 10)
(47932, 1)
```

In [43]:
```
# X = merged.drop('p3', axis=1).values
# y = merged ['p3'].values.reshape(-1, 1)

# """X_mean = np.mean(X, axis=0)
# X_std = np.std(X, axis=0)
# X = (X - X_mean) / X_std

# merged = pd.merge (X, y, how='inner', left_index=True, right_index=True)"""
# from sklearn.preprocessing import StandardScaler
# s= StandardScaler()
# s1= StandardScaler()
# # standardization
# X1 = s.fit_transform(X)
# y1 = s1.fit_transform(y)
```

In [44]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size=0.3, random_state=200)
```

In [45]:
```
y1.shape
```

Out[45]:
```
(47932, 1)
```

In [46]:
```
y_train.shape
```

Out[46]:
```
(33552, 1)
```

In [47]:
```
y_test.shape
```

Out[47]:
```
(14380, 1)
```

In [48]:
```
from sklearn import metrics
import scipy as sp
import numpy as np
import math
from sklearn import metrics

def perturbation_rank(model, x, y, names, regression):
    errors = []

    for i in range(x.shape[1]):
        hold = np.array(x[:, i])
        np.random.shuffle(x[:, i])

        if regression:
            pred = model.predict(x)
            error = metrics.mean_absolute_error(y, pred)
        else:
            pred = model.predict_proba(x)
            error = metrics.log_loss(y, pred)

        errors.append(error)
        x[:, i] = hold

    max_error = np.max(errors)
    importance = [e/max_error for e in errors]

    data = {'name':names,'error':errors,'importance':importance}
    result = pd.DataFrame(data, columns = ['name','error','importance'])
    result.sort_values(by=['importance'], ascending=[0], inplace=True)
    result.reset_index(inplace=True, drop=True)
    return result
```

RIDGE REGRESSION

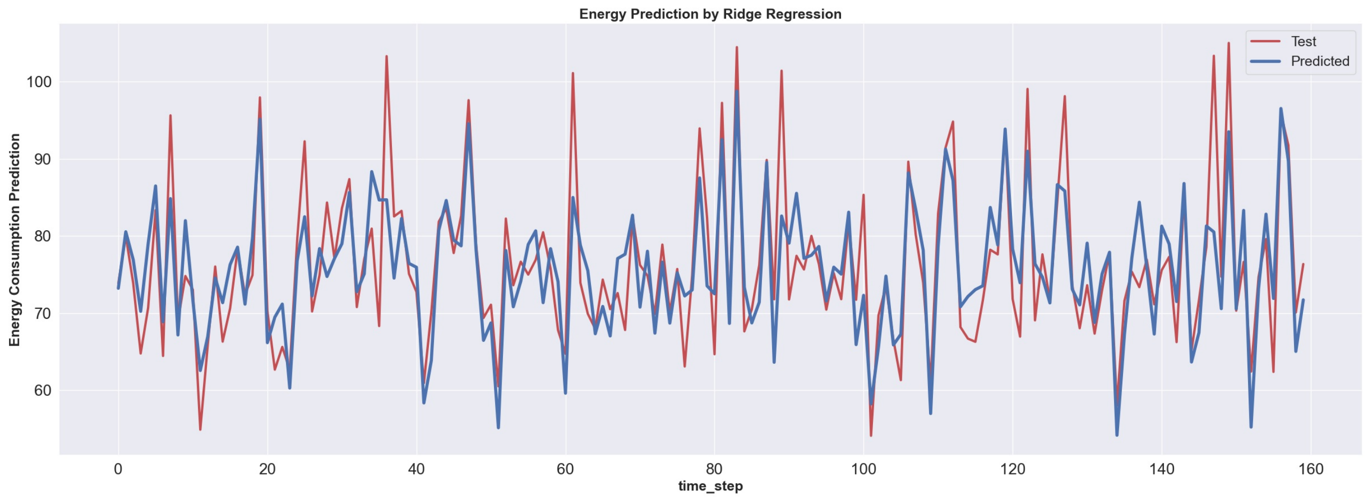
In [49]:
```
#ridge regg.
import time
from sklearn.linear_model import Ridge
start_time = time.time()

model1 = Ridge()
model1.fit (X_train, y_train)

end_time = time.time()
training_time = end_time - start_time
print("Training time:", training_time, "seconds")
```

```
Training time: 0.011968135833740234 seconds
```

In [50]:
```python
X_test1 = s.fit_transform(X_test)

y_hat1 = model1.predict(X_test1)
y_hat1n = s1.inverse_transform(y_hat1.reshape(-1, 1))
y_test1 = s1.inverse_transform(y_test)
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_squared_error,mean_absolute_percentage_error
import sklearn.metrics as metrics
from math import sqrt
RMSE = sqrt(mean_squared_error(y_test1, y_hat1n))
print("RMSE:",RMSE)
#mse

#mape
mape = mean_absolute_percentage_error(y_test1, y_hat1n)
print("MAPE:",mape)
print ("Percentual:", metrics.mean_absolute_error(y_test1,y_hat1n)/y_test1.mean()*100, "%")
```

```
RMSE: 5.588523937262939
MAPE: 0.05741663812768239
Percentual: 5.696290696921264 %
```

In [51]:
```python
import numpy as np
import matplotlib.pyplot as plt

# assuming y_test and y_hat1 are already defined

# downsample data to reduce number of data points
downsample_factor = 90
y_test_downsampled = y_test1[::downsample_factor]
y_hat1_downsampled = y_hat1n[::downsample_factor]

# create line plot
plt.figure(figsize=(30,10))
plt.plot(y_test_downsampled , 'r-', linewidth=3)
plt.plot(y_hat1_downsampled,'b-' , linewidth=4)
plt.xlabel('time_step', fontsize = 18, fontweight="bold")
plt.ylabel('Energy Consumption Prediction', fontsize = 18, fontweight="bold")
plt.legend (('Test','Predicted'), fontsize = 18)
plt.title("Energy Prediction by Ridge Regression", fontsize = 18, fontweight="bold")
plt.tick_params(axis='both', which='major', labelsize=20)
plt.savefig('PREDICTION by Ridge Regression Line Plot.png', format='png')
plt.savefig('PREDICTION by Ridge Regression Line Plot.pdf', format='pdf')
plt.show()
```



EXREMELY RANDOMIZED TREE

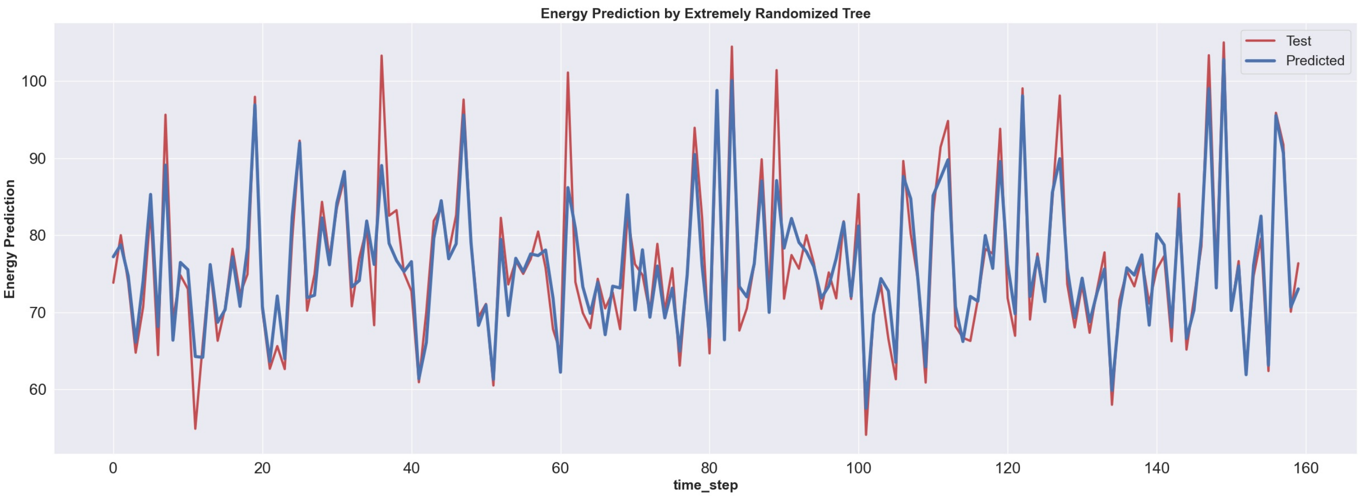
In [52]:
```python
#xrf
import time
from sklearn.ensemble import ExtraTreesRegressor
start_time = time.time()
model2 = ExtraTreesRegressor(max_depth=25,
                             n_estimators=400,
                             bootstrap=True,
                             max_samples=0.7)
model2.fit(X_train,y_train)
end_time = time.time()
training_time = end_time - start_time
print("Training time:", training_time, "seconds")
```

```
C:\Users\Guest1\AppData\Local\Temp\ipykernel_11368\1924099623.py:9: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  model2.fit(X_train,y_train)
Training time: 19.98242688179016 seconds
```

In [53]:
```python
X_test1 = s.fit_transform(X_test)

y_hat2 = model2.predict(X_test1)
y_hat2n = s1.inverse_transform(y_hat2.reshape(-1, 1))
y_test1 = s1.inverse_transform(y_test)
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_squared_error,mean_absolute_percentage_error
import sklearn.metrics as metrics
from math import sqrt
RMSE = sqrt(mean_squared_error(y_test1, y_hat2n))
print("RMSE:",RMSE)
#mse

#mape
mape = mean_absolute_percentage_error(y_test1, y_hat2n)
print("MAPE:",mape)
print ("Percentual:", metrics.mean_absolute_error(y_test1,y_hat2n)/y_test1.mean()*100, "%")
```

```
RMSE: 3.3635529627653025
MAPE: 0.032868682003098726
Percentual: 3.286631263016239 %
```

In [54]:
```python
import numpy as np
import matplotlib.pyplot as plt

# assuming y_test and y_hat1 are already defined

# downsample data to reduce number of data points
downsample_factor = 90
y_test_downsampled = y_test1[::downsample_factor]
y_hat2_downsampled = y_hat2n[::downsample_factor]

# create line plot
plt.figure(figsize=(30,10))
plt.plot(y_test_downsampled , 'r-', linewidth=3)
plt.plot(y_hat2_downsampled,'b-' , linewidth=4)
plt.xlabel('time_step', fontsize = 18, fontweight="bold")
plt.ylabel('Energy Prediction', fontsize = 18, fontweight="bold")
plt.legend (('Test','Predicted'), fontsize = 18)
plt.title("Energy Prediction by Extremely Randomized Tree", fontsize = 18, fontweight="bold")
plt.tick_params(axis='both', which='major', labelsize=20)
plt.savefig('PREDICTION by EXREMELY RANDOMIZED TREE Line Plot.png', format='png')
plt.savefig('PREDICTION by EXREMELY RANDOMIZED TREE Line Plot.pdf', format='pdf')
plt.show()
```



RANDOM FOREST REGRESSOR

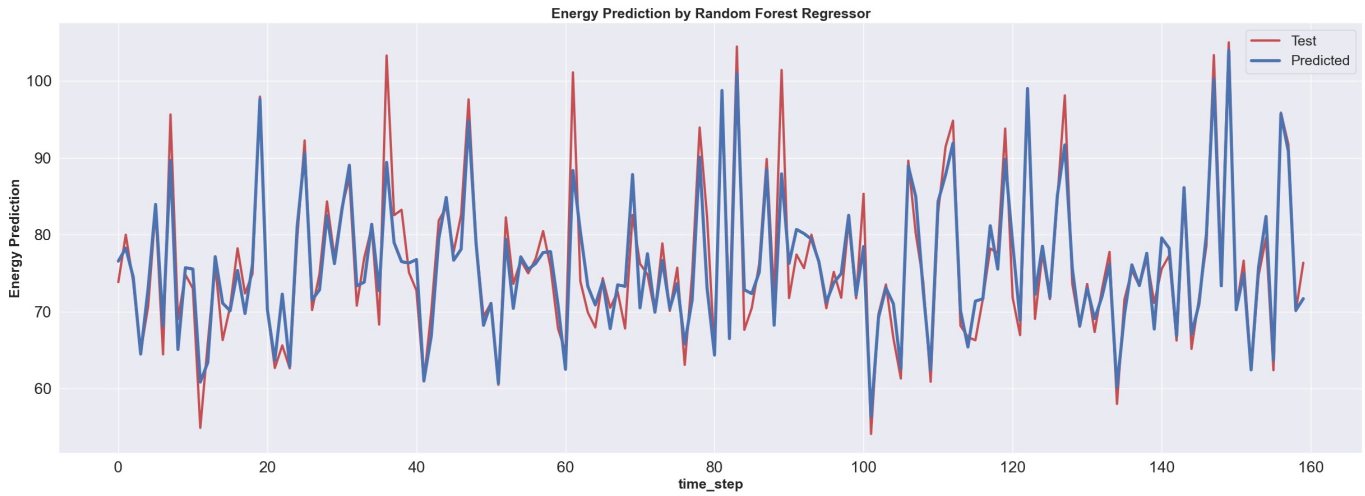
In [55]:
```python
start_time = time.time()
from sklearn.ensemble import RandomForestRegressor
model3 = RandomForestRegressor(max_depth=20, n_estimators=400, max_features=0.9)
model3.fit(X_train,y_train)
end_time = time.time()
training_time = end_time - start_time
print("Training time:", training_time, "seconds")
```

```
C:\Users\Guest1\AppData\Local\Temp\ipykernel_11368\1501663348.py:4: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  model3.fit(X_train,y_train)
Training time: 99.90072631835938 seconds
```

In [56]:
```
X_test1 = s.fit_transform(X_test)

y_hat3 = model3.predict(X_test1)
y_hat3n = s1.inverse_transform(y_hat3.reshape(-1, 1))
y_test3 = s1.inverse_transform(y_test)
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_squared_error,mean_absolute_percentage_error
import sklearn.metrics as metrics
from math import sqrt
RMSE = sqrt(mean_squared_error(y_test3, y_hat3n))
print("RMSE:",RMSE)
#mse

#mape
mape = mean_absolute_percentage_error(y_test3, y_hat3n)
print("MAPE:",mape)
print ("Percentual:", metrics.mean_absolute_error(y_test3,y_hat3n)/y_test3.mean()*100, "%")
```

```
RMSE: 3.362781547168306
MAPE: 0.03244923466575623
Percentual: 3.2503359231103266 %
```

In [57]:
```
import numpy as np
import matplotlib.pyplot as plt

# assuming y_test and y_hat1 are already defined

# downsample data to reduce number of data points
downsample_factor = 90
y_test_downsampled = y_test1[::downsample_factor]
y_hat3_downsampled = y_hat3n[::downsample_factor]

# create line plot
plt.figure(figsize=(30,10))
plt.plot(y_test_downsampled , 'r-', linewidth=3)
plt.plot(y_hat3_downsampled,'b-' , linewidth=4)
plt.xlabel('time_step', fontsize = 18, fontweight="bold")
plt.ylabel('Energy Prediction', fontsize = 18, fontweight="bold")
plt.legend (('Test','Predicted'), fontsize = 18)
plt.title("Energy Prediction by Random Forest Regressor", fontsize = 18, fontweight="bold")
plt.tick_params(axis='both', which='major', labelsize=20)
plt.savefig('PREDICTION by RANDOM FOREST REGRESSOR Line Plot.png', format='png')
plt.savefig('PREDICTION by RANDOM FOREST REGRESSOR Line Plot.pdf', format='pdf')
plt.show()
```



Gradient boosting machines

In [58]:
```
start_time = time.time()
from sklearn.ensemble import GradientBoostingRegressor
model4 = GradientBoostingRegressor(max_depth=8,
                                   loss='squared_error',
                                  n_estimators=400)
model4.fit(X_train,y_train)
end_time = time.time()
training_time = end_time - start_time
print("Training time:", training_time, "seconds")
```

```
C:\Users\Guest1\anaconda3\lib\site-packages\sklearn\ensemble\_gb.py:437: DataConversionWarning: A column-vector
y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using rave
l().
  y = column_or_1d(y, warn=True)
Training time: 86.12333059310913 seconds
```

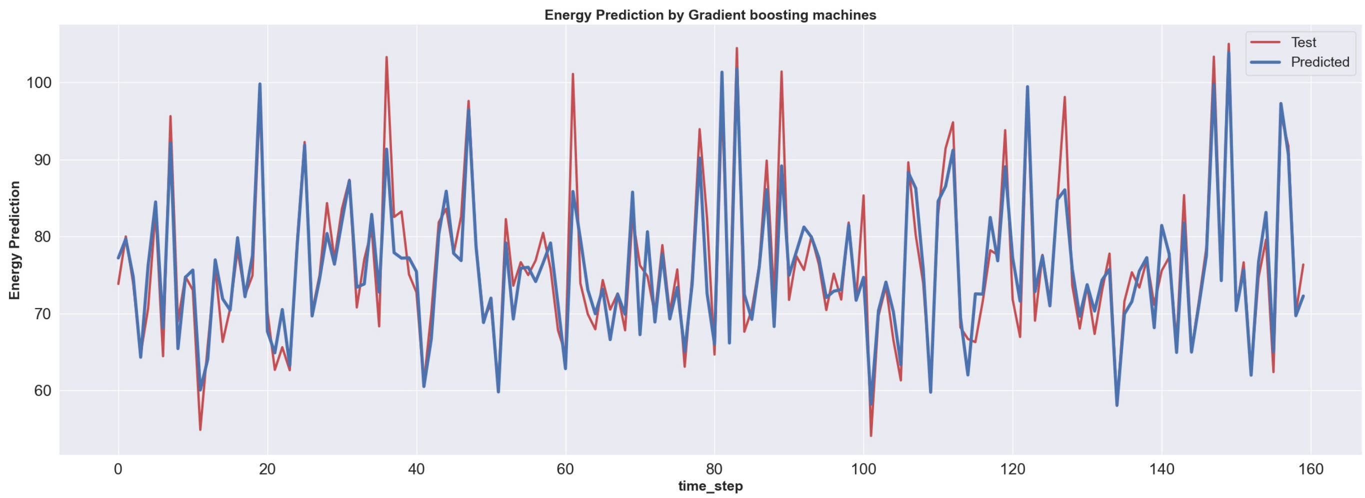
In [59]:

```python
X_test1 = s.fit_transform(X_test)

y_hat4 = model4.predict(X_test1)
y_hat4n = s1.inverse_transform(y_hat4.reshape(-1, 1))
y_test4 = s1.inverse_transform(y_test)
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_squared_error,mean_absolute_percentage_error
import sklearn.metrics as metrics
from math import sqrt
RMSE = sqrt(mean_squared_error(y_test4, y_hat4n))
print("RMSE:",RMSE)
#mse

#mape
mape = mean_absolute_percentage_error(y_test4, y_hat4n)
print("MAPE:",mape)
print ("Percentual:", metrics.mean_absolute_error(y_test4,y_hat4n)/y_test4.mean()*100, "%")
```

```
RMSE: 3.4736337849790413
MAPE: 0.03406318630550938
Percentual: 3.405146278126864 %
```

In [60]:

```python
import numpy as np
import matplotlib.pyplot as plt

# assuming y_test and y_hat1 are already defined

# downsample data to reduce number of data points
downsample_factor = 90
y_test_downsampled = y_test1[::downsample_factor]
y_hat4_downsampled = y_hat4n[::downsample_factor]

# create line plot
plt.figure(figsize=(30,10))
plt.plot(y_test_downsampled , 'r-', linewidth=3)
plt.plot(y_hat4_downsampled,'b-' , linewidth=4)
plt.xlabel('time_step', fontsize = 18, fontweight="bold")
plt.ylabel('Energy Prediction', fontsize = 18, fontweight="bold")
plt.legend (('Test','Predicted'), fontsize = 18)
plt.title("Energy Prediction by Gradient boosting machines", fontsize = 18, fontweight="bold")
plt.tick_params(axis='both', which='major', labelsize=20)
plt.savefig('PREDICTION by Gradient boosting machines Line Plot.png', format='png')
plt.savefig('PREDICTION by Gradient boosting machines Line Plot.pdf', format='pdf')
plt.show()
```



SVM

- To 85k points, it runs in 2hs 26min
- C defaut:1. The strength of the regularization is inversely proportional to C
- gamma default = scale = 1 / (n_features * X.var())

In [61]:

```python
from sklearn.svm import SVR
start_time = time.time()
model5 = SVR(kernel='rbf',
            C=900,
            epsilon=1,
```

```
              gamma='scale',
              cache_size=1000)
model5.fit(X_train, y_train)
end_time = time.time()
training_time = end_time - start_time
print("Training time:", training_time, "seconds")
```

```
C:\Users\Guest1\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWarning: A column-ve
ctor y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
Training time: 23.442914485931396 seconds
```

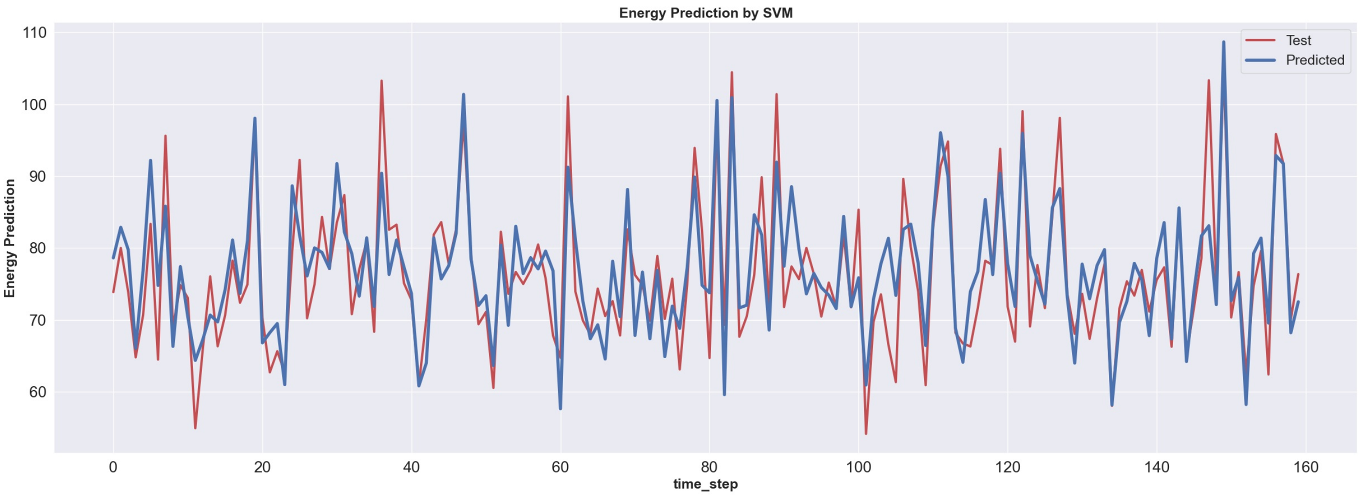
In [62]:

```
X_test1 = s.fit_transform(X_test)

y_hat5 = model5.predict(X_test1)
y_hat5n = s1.inverse_transform(y_hat5.reshape(-1, 1))
y_test1 = s1.inverse_transform(y_test)
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_squared_error,mean_absolute_percentage_error
import sklearn.metrics as metrics
from math import sqrt
RMSE = sqrt(mean_squared_error(y_test1, y_hat5n))
print("RMSE:",RMSE)
#mse

#mape
mape = mean_absolute_percentage_error(y_test1, y_hat5n)
print("MAPE:",mape)
print ("Percentual:", metrics.mean_absolute_error(y_test1,y_hat5n)/y_test1.mean()*100, "%")
```

```
RMSE: 5.109171758509057
MAPE: 0.0536080181419316
Percentual: 5.31065600581782 %
```

In [63]:

```
import numpy as np
import matplotlib.pyplot as plt

# assuming y_test and y_hat1 are already defined

# downsample data to reduce number of data points
downsample_factor = 90
y_test_downsampled = y_test1[::downsample_factor]
y_hat5_downsampled = y_hat5n[::downsample_factor]

# create line plot
plt.figure(figsize=(30,10))
plt.plot(y_test_downsampled , 'r-', linewidth=3)
plt.plot(y_hat5_downsampled,'b-' , linewidth=4)
plt.xlabel('time_step', fontsize = 18, fontweight="bold")
plt.ylabel('Energy Prediction', fontsize = 18, fontweight="bold")
plt.legend (('Test','Predicted'), fontsize = 18)
plt.title("Energy Prediction by SVM", fontsize = 18, fontweight="bold")
plt.tick_params(axis='both', which='major', labelsize=20)
plt.savefig('PREDICTION by SVM Line Plot.png', format='png')
plt.savefig('PREDICTION by SVM Line Plot.pdf', format='pdf')
plt.show()
```



ANN

- To 85k points, it runs in 7 min
- We wrap the model to allow compatibility with Scikitlearn

In [67]:
```
import keras
from keras import Sequential
from keras.layers import Dropout, Dense
from keras.wrappers.scikit_learn import KerasRegressor
from keras.callbacks import EarlyStopping
from keras.callbacks import ModelCheckpoint

#utils
from keras_tqdm import TQDMNotebookCallback
from keras.models import save_model, load_model
from keras.utils.vis_utils import plot_model
```

In [68]:
```
from keras import backend as K

def val_mean_absolute_error(y_true, y_pred):
    return K.mean(K.abs(y_pred - y_true), axis=-1)

# Register custom metric
from keras.utils import get_custom_objects
get_custom_objects().update({'val_mean_absolute_error': val_mean_absolute_error})
```

In [69]:
```
# Define the ANN architecture
start_time = time.time()
def create_model():
    model6 = Sequential()
    model6.add(Dense(256, activation='relu', input_dim=X_train.shape[1]))
    model6.add(Dropout(0.3))
    model6.add(Dense(128, activation='relu'))
    model6.add(Dropout(0.2))
    model6.add(Dense(64, activation='relu'))
    model6.add(Dropout(0.2))
    model6.add(Dense(32, activation='relu'))
    model6.add(Dropout(0.1))
    model6.add(Dense(1, activation='linear'))

    # Compile the model
    model6.compile(loss='mae', optimizer='adam', metrics=['mae', 'mse', 'mape'])

    return model6

# Initialize the ANN model, callbacks, and training parameters
model6 = create_model()
model_save_path = out_path + 'model6.h5'
callback_cp = ModelCheckpoint(model_save_path, monitor='val_loss', mode='min', verbose=1, save_best_only=True)
callback_es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=20)

# Train the ANN model
history = model6.fit(X_train, y_train, epochs=200, verbose=1, batch_size=32, validation_split=0.1, callbacks=[ca

# Evaluate the ANN model
train_loss, train_mae, train_mse, train_mape = model6.evaluate(X_train, y_train, verbose=0)
test_loss, test_mae, test_mse, test_mape = model6.evaluate(X_test, y_test, verbose=0)

# Print the results
print("Training Loss: {:.4f}, Training MAE: {:.4f}, Training MSE: {:.4f}, Training MAPE: {:.4f}".format(train_lo
print("Testing Loss: {:.4f}, Testing MAE: {:.4f}, Testing MSE: {:.4f}, Testing MAPE: {:.4f}".format(test_loss, t

# Extract loss values for each epoch from history
train_loss = history.history['loss']
val_loss = history.history['val_loss']
train_mae = history.history['mae']
val_mae = history.history['val_mae']
train_mse = history.history['mse']
val_mse = history.history['val_mse']
train_mape = history.history['mape']
val_mape = history.history['val_mape']

# Create a list of loss values with their corresponding labels
loss_data = [train_loss, val_loss, train_mae, val_mae, train_mse, val_mse, train_mape, val_mape]
labels = ['Training Loss', 'Validation Loss', 'Training MAE', 'Validation MAE', 'Training MSE', 'Validation MSE

end_time = time.time()
training_time = end_time - start_time
print("Training time:", training_time, "seconds")
```

```
Epoch 1/200
923/944 [============================>.] - ETA: 0s - loss: 0.4355 - mae: 0.4355 - mse: 0.3281 - mape: 271.6059
Epoch 1: val_loss improved from inf to 0.37219, saving model to assets\model6.h5
944/944 [==============================] - 3s 2ms/step - loss: 0.4352 - mae: 0.4352 - mse: 0.3275 - mape: 268.19
94 - val_loss: 0.3722 - val_mae: 0.3722 - val_mse: 0.2422 - val_mape: 160.8932
Epoch 2/200
932/944 [============================>.] - ETA: 0s - loss: 0.3925 - mae: 0.3925 - mse: 0.2636 - mape: 289.3296
Epoch 2: val_loss improved from 0.37219 to 0.37182, saving model to assets\model6.h5
```

```
944/944 [==============================] - 2s 2ms/step - loss: 0.3920 - mae: 0.3920 - mse: 0.2630 - mape: 297.89
64 - val_loss: 0.3718 - val_mae: 0.3718 - val_mse: 0.2467 - val_mape: 149.4633
Epoch 3/200
942/944 [============================>.] - ETA: 0s - loss: 0.3793 - mae: 0.3793 - mse: 0.2492 - mape: 290.1564
Epoch 3: val_loss improved from 0.37182 to 0.35672, saving model to assets\model6.h5
944/944 [==============================] - 2s 2ms/step - loss: 0.3794 - mae: 0.3794 - mse: 0.2493 - mape: 289.83
02 - val_loss: 0.3567 - val_mae: 0.3567 - val_mse: 0.2220 - val_mape: 144.7511
Epoch 4/200
929/944 [============================>.] - ETA: 0s - loss: 0.3679 - mae: 0.3679 - mse: 0.2335 - mape: 237.2220
Epoch 4: val_loss improved from 0.35672 to 0.34272, saving model to assets\model6.h5
944/944 [==============================] - 2s 2ms/step - loss: 0.3681 - mae: 0.3681 - mse: 0.2341 - mape: 237.64
36 - val_loss: 0.3427 - val_mae: 0.3427 - val_mse: 0.2083 - val_mape: 140.3230
Epoch 5/200
925/944 [============================>.] - ETA: 0s - loss: 0.3606 - mae: 0.3606 - mse: 0.2260 - mape: 287.1761
Epoch 5: val_loss did not improve from 0.34272
944/944 [==============================] - 2s 2ms/step - loss: 0.3607 - mae: 0.3607 - mse: 0.2262 - mape: 284.94
67 - val_loss: 0.3448 - val_mae: 0.3448 - val_mse: 0.2103 - val_mape: 148.4540
Epoch 6/200
933/944 [============================>.] - ETA: 0s - loss: 0.3553 - mae: 0.3553 - mse: 0.2212 - mape: 242.5967
Epoch 6: val_loss did not improve from 0.34272
944/944 [==============================] - 2s 2ms/step - loss: 0.3555 - mae: 0.3555 - mse: 0.2215 - mape: 241.88
18 - val_loss: 0.3465 - val_mae: 0.3465 - val_mse: 0.2151 - val_mape: 142.1277
Epoch 7/200
937/944 [============================>.] - ETA: 0s - loss: 0.3515 - mae: 0.3515 - mse: 0.2170 - mape: 246.0629
Epoch 7: val_loss did not improve from 0.34272
944/944 [==============================] - 2s 2ms/step - loss: 0.3512 - mae: 0.3512 - mse: 0.2166 - mape: 245.99
05 - val_loss: 0.3436 - val_mae: 0.3436 - val_mse: 0.2146 - val_mape: 156.1059
Epoch 8/200
944/944 [==============================] - ETA: 0s - loss: 0.3467 - mae: 0.3467 - mse: 0.2124 - mape: 259.8314
Epoch 8: val_loss improved from 0.34272 to 0.31795, saving model to assets\model6.h5
944/944 [==============================] - 2s 2ms/step - loss: 0.3467 - mae: 0.3467 - mse: 0.2124 - mape: 259.83
14 - val_loss: 0.3179 - val_mae: 0.3179 - val_mse: 0.1811 - val_mape: 145.5335
Epoch 9/200
942/944 [============================>.] - ETA: 0s - loss: 0.3435 - mae: 0.3435 - mse: 0.2085 - mape: 210.0636
Epoch 9: val_loss did not improve from 0.31795
944/944 [==============================] - 2s 2ms/step - loss: 0.3435 - mae: 0.3435 - mse: 0.2084 - mape: 209.85
56 - val_loss: 0.3380 - val_mae: 0.3380 - val_mse: 0.2096 - val_mape: 141.6499
Epoch 10/200
937/944 [============================>.] - ETA: 0s - loss: 0.3396 - mae: 0.3396 - mse: 0.2040 - mape: 239.1338
Epoch 10: val_loss did not improve from 0.31795
944/944 [==============================] - 2s 2ms/step - loss: 0.3395 - mae: 0.3395 - mse: 0.2040 - mape: 238.17
90 - val_loss: 0.3229 - val_mae: 0.3229 - val_mse: 0.1918 - val_mape: 151.4029
Epoch 11/200
925/944 [============================>.] - ETA: 0s - loss: 0.3374 - mae: 0.3374 - mse: 0.2014 - mape: 279.3955
Epoch 11: val_loss did not improve from 0.31795
944/944 [==============================] - 2s 2ms/step - loss: 0.3376 - mae: 0.3376 - mse: 0.2016 - mape: 282.88
71 - val_loss: 0.3328 - val_mae: 0.3328 - val_mse: 0.1976 - val_mape: 135.9854
Epoch 12/200
934/944 [============================>.] - ETA: 0s - loss: 0.3356 - mae: 0.3356 - mse: 0.2008 - mape: 260.4789
Epoch 12: val_loss did not improve from 0.31795
944/944 [==============================] - 2s 2ms/step - loss: 0.3355 - mae: 0.3355 - mse: 0.2008 - mape: 258.81
59 - val_loss: 0.3362 - val_mae: 0.3362 - val_mse: 0.2045 - val_mape: 133.7824
Epoch 13/200
941/944 [============================>.] - ETA: 0s - loss: 0.3331 - mae: 0.3331 - mse: 0.1970 - mape: 239.5349
Epoch 13: val_loss did not improve from 0.31795
944/944 [==============================] - 2s 2ms/step - loss: 0.3331 - mae: 0.3331 - mse: 0.1972 - mape: 239.25
04 - val_loss: 0.3246 - val_mae: 0.3246 - val_mse: 0.1895 - val_mape: 144.9116
Epoch 14/200
933/944 [============================>.] - ETA: 0s - loss: 0.3329 - mae: 0.3329 - mse: 0.1967 - mape: 224.7742
Epoch 14: val_loss improved from 0.31795 to 0.30748, saving model to assets\model6.h5
944/944 [==============================] - 2s 2ms/step - loss: 0.3331 - mae: 0.3331 - mse: 0.1971 - mape: 225.76
97 - val_loss: 0.3075 - val_mae: 0.3075 - val_mse: 0.1711 - val_mape: 151.8966
Epoch 15/200
923/944 [============================>.] - ETA: 0s - loss: 0.3294 - mae: 0.3294 - mse: 0.1937 - mape: 212.6762
Epoch 15: val_loss did not improve from 0.30748
944/944 [==============================] - 2s 2ms/step - loss: 0.3294 - mae: 0.3294 - mse: 0.1937 - mape: 210.15
39 - val_loss: 0.3234 - val_mae: 0.3234 - val_mse: 0.1913 - val_mape: 134.9145
Epoch 16/200
921/944 [============================>.] - ETA: 0s - loss: 0.3270 - mae: 0.3270 - mse: 0.1919 - mape: 218.7128
Epoch 16: val_loss did not improve from 0.30748
944/944 [==============================] - 2s 2ms/step - loss: 0.3274 - mae: 0.3274 - mse: 0.1922 - mape: 216.94
53 - val_loss: 0.3463 - val_mae: 0.3463 - val_mse: 0.2119 - val_mape: 125.3739
Epoch 17/200
922/944 [============================>.] - ETA: 0s - loss: 0.3291 - mae: 0.3291 - mse: 0.1935 - mape: 219.3088
Epoch 17: val_loss did not improve from 0.30748
944/944 [==============================] - 2s 2ms/step - loss: 0.3289 - mae: 0.3289 - mse: 0.1932 - mape: 216.72
63 - val_loss: 0.3438 - val_mae: 0.3438 - val_mse: 0.2115 - val_mape: 129.5732
Epoch 18/200
944/944 [==============================] - ETA: 0s - loss: 0.3258 - mae: 0.3258 - mse: 0.1904 - mape: 199.9736
Epoch 18: val_loss did not improve from 0.30748
944/944 [==============================] - 2s 2ms/step - loss: 0.3258 - mae: 0.3258 - mse: 0.1904 - mape: 199.97
36 - val_loss: 0.3169 - val_mae: 0.3169 - val_mse: 0.1850 - val_mape: 134.7139
Epoch 19/200
```

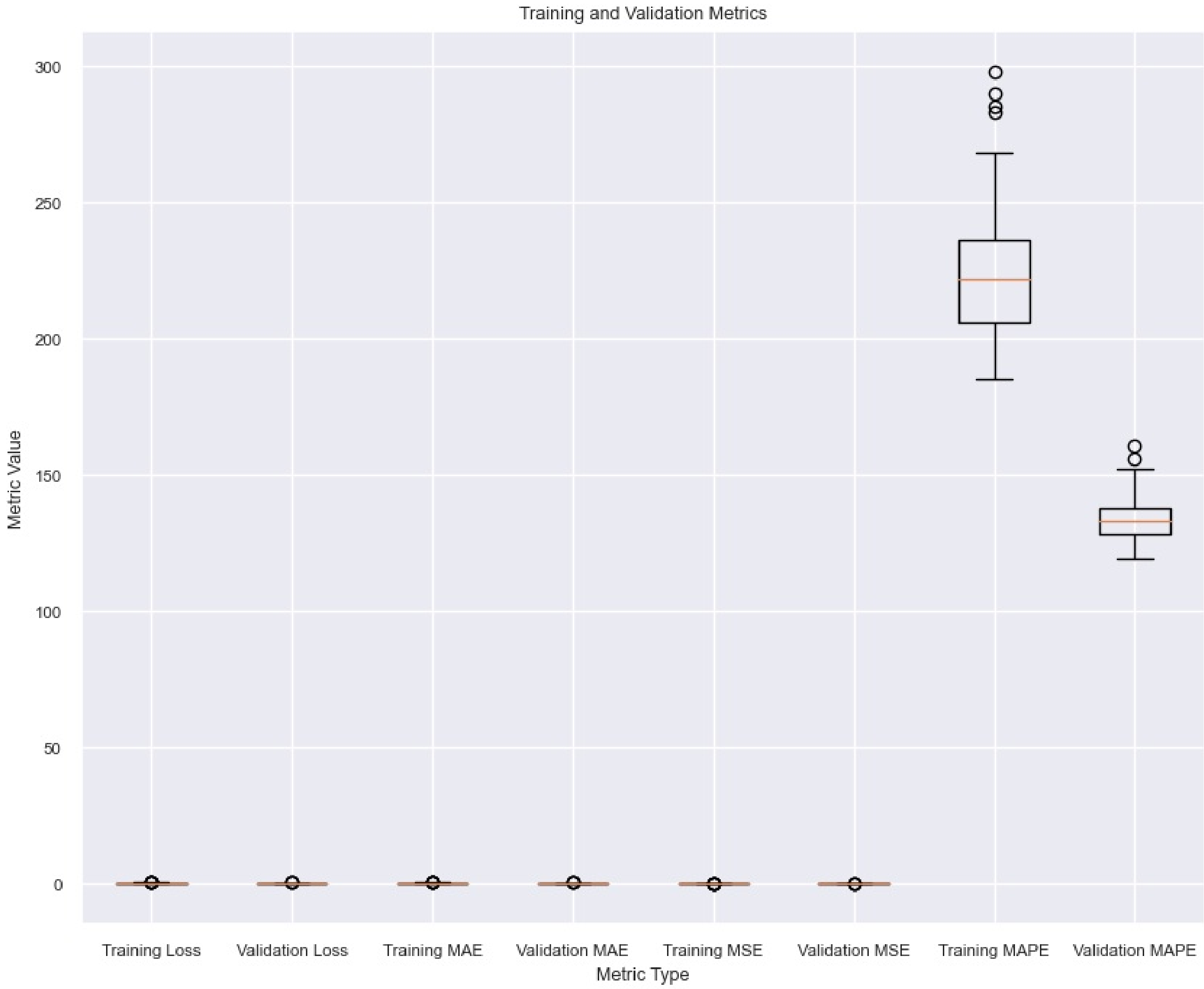
```
937/944 [============================>.] - ETA: 0s - loss: 0.3247 - mae: 0.3247 - mse: 0.1892 - mape: 208.4115
Epoch 19: val_loss did not improve from 0.30748
944/944 [==============================] - 2s 2ms/step - loss: 0.3249 - mae: 0.3249 - mse: 0.1898 - mape: 208.38
61 - val_loss: 0.3212 - val_mae: 0.3212 - val_mse: 0.1840 - val_mape: 132.5751
Epoch 20/200
928/944 [============================>.] - ETA: 0s - loss: 0.3242 - mae: 0.3242 - mse: 0.1888 - mape: 236.0405
Epoch 20: val_loss did not improve from 0.30748
944/944 [==============================] - 2s 2ms/step - loss: 0.3244 - mae: 0.3244 - mse: 0.1891 - mape: 234.87
30 - val_loss: 0.3076 - val_mae: 0.3076 - val_mse: 0.1727 - val_mape: 138.0580
Epoch 21/200
921/944 [============================>.] - ETA: 0s - loss: 0.3228 - mae: 0.3228 - mse: 0.1872 - mape: 228.5316
Epoch 21: val_loss did not improve from 0.30748
944/944 [==============================] - 2s 2ms/step - loss: 0.3231 - mae: 0.3231 - mse: 0.1874 - mape: 229.28
17 - val_loss: 0.3261 - val_mae: 0.3261 - val_mse: 0.1956 - val_mape: 136.9400
Epoch 22/200
942/944 [============================>.] - ETA: 0s - loss: 0.3209 - mae: 0.3209 - mse: 0.1849 - mape: 227.6378
Epoch 22: val_loss did not improve from 0.30748
944/944 [==============================] - 2s 2ms/step - loss: 0.3209 - mae: 0.3209 - mse: 0.1851 - mape: 227.62
48 - val_loss: 0.3237 - val_mae: 0.3237 - val_mse: 0.1885 - val_mape: 129.4207
Epoch 23/200
927/944 [============================>.] - ETA: 0s - loss: 0.3212 - mae: 0.3212 - mse: 0.1854 - mape: 205.7398
Epoch 23: val_loss did not improve from 0.30748
944/944 [==============================] - 2s 2ms/step - loss: 0.3210 - mae: 0.3210 - mse: 0.1851 - mape: 204.30
56 - val_loss: 0.3356 - val_mae: 0.3356 - val_mse: 0.2077 - val_mape: 127.5855
Epoch 24/200
933/944 [============================>.] - ETA: 0s - loss: 0.3198 - mae: 0.3198 - mse: 0.1850 - mape: 221.4676
Epoch 24: val_loss did not improve from 0.30748
944/944 [==============================] - 2s 2ms/step - loss: 0.3198 - mae: 0.3198 - mse: 0.1848 - mape: 220.15
28 - val_loss: 0.3113 - val_mae: 0.3113 - val_mse: 0.1772 - val_mape: 134.3022
Epoch 25/200
923/944 [============================>.] - ETA: 0s - loss: 0.3201 - mae: 0.3201 - mse: 0.1850 - mape: 235.0983
Epoch 25: val_loss did not improve from 0.30748
944/944 [==============================] - 2s 2ms/step - loss: 0.3203 - mae: 0.3203 - mse: 0.1853 - mape: 234.03
90 - val_loss: 0.3142 - val_mae: 0.3142 - val_mse: 0.1829 - val_mape: 127.0350
Epoch 26/200
940/944 [============================>.] - ETA: 0s - loss: 0.3187 - mae: 0.3187 - mse: 0.1834 - mape: 188.7922
Epoch 26: val_loss improved from 0.30748 to 0.29936, saving model to assets\model6.h5
944/944 [==============================] - 2s 2ms/step - loss: 0.3186 - mae: 0.3186 - mse: 0.1832 - mape: 188.43
13 - val_loss: 0.2994 - val_mae: 0.2994 - val_mse: 0.1602 - val_mape: 140.1492
Epoch 27/200
933/944 [============================>.] - ETA: 0s - loss: 0.3177 - mae: 0.3177 - mse: 0.1811 - mape: 206.9467
Epoch 27: val_loss did not improve from 0.29936
944/944 [==============================] - 2s 2ms/step - loss: 0.3179 - mae: 0.3179 - mse: 0.1813 - mape: 206.38
07 - val_loss: 0.3174 - val_mae: 0.3174 - val_mse: 0.1850 - val_mape: 130.3228
Epoch 28/200
930/944 [============================>.] - ETA: 0s - loss: 0.3166 - mae: 0.3166 - mse: 0.1811 - mape: 219.3417
Epoch 28: val_loss did not improve from 0.29936
944/944 [==============================] - 2s 2ms/step - loss: 0.3167 - mae: 0.3167 - mse: 0.1814 - mape: 228.59
96 - val_loss: 0.3056 - val_mae: 0.3056 - val_mse: 0.1724 - val_mape: 136.7348
Epoch 29/200
925/944 [============================>.] - ETA: 0s - loss: 0.3156 - mae: 0.3156 - mse: 0.1792 - mape: 235.9478
Epoch 29: val_loss did not improve from 0.29936
944/944 [==============================] - 2s 2ms/step - loss: 0.3154 - mae: 0.3154 - mse: 0.1791 - mape: 233.56
88 - val_loss: 0.3221 - val_mae: 0.3221 - val_mse: 0.1881 - val_mape: 129.1976
Epoch 30/200
932/944 [============================>.] - ETA: 0s - loss: 0.3157 - mae: 0.3157 - mse: 0.1796 - mape: 244.4145
Epoch 30: val_loss did not improve from 0.29936
944/944 [==============================] - 2s 2ms/step - loss: 0.3157 - mae: 0.3157 - mse: 0.1796 - mape: 244.14
06 - val_loss: 0.3141 - val_mae: 0.3141 - val_mse: 0.1814 - val_mape: 130.2898
Epoch 31/200
936/944 [============================>.] - ETA: 0s - loss: 0.3155 - mae: 0.3155 - mse: 0.1799 - mape: 206.9315
Epoch 31: val_loss did not improve from 0.29936
944/944 [==============================] - 2s 2ms/step - loss: 0.3157 - mae: 0.3157 - mse: 0.1800 - mape: 206.51
60 - val_loss: 0.3027 - val_mae: 0.3027 - val_mse: 0.1718 - val_mape: 137.5539
Epoch 32/200
924/944 [============================>.] - ETA: 0s - loss: 0.3161 - mae: 0.3161 - mse: 0.1799 - mape: 236.3562
Epoch 32: val_loss did not improve from 0.29936
944/944 [==============================] - 2s 2ms/step - loss: 0.3160 - mae: 0.3160 - mse: 0.1795 - mape: 234.89
82 - val_loss: 0.3035 - val_mae: 0.3035 - val_mse: 0.1697 - val_mape: 139.9302
Epoch 33/200
936/944 [============================>.] - ETA: 0s - loss: 0.3132 - mae: 0.3132 - mse: 0.1777 - mape: 193.8730
Epoch 33: val_loss improved from 0.29936 to 0.29909, saving model to assets\model6.h5
944/944 [==============================] - 2s 2ms/step - loss: 0.3134 - mae: 0.3134 - mse: 0.1780 - mape: 193.12
60 - val_loss: 0.2991 - val_mae: 0.2991 - val_mse: 0.1669 - val_mape: 132.7009
Epoch 34/200
924/944 [============================>.] - ETA: 0s - loss: 0.3116 - mae: 0.3116 - mse: 0.1757 - mape: 211.5742
Epoch 34: val_loss improved from 0.29909 to 0.29802, saving model to assets\model6.h5
944/944 [==============================] - 2s 2ms/step - loss: 0.3116 - mae: 0.3116 - mse: 0.1759 - mape: 212.62
75 - val_loss: 0.2980 - val_mae: 0.2980 - val_mse: 0.1611 - val_mape: 135.3036
Epoch 35/200
943/944 [============================>.] - ETA: 0s - loss: 0.3115 - mae: 0.3115 - mse: 0.1767 - mape: 202.9347
Epoch 35: val_loss did not improve from 0.29802
944/944 [==============================] - 2s 2ms/step - loss: 0.3114 - mae: 0.3114 - mse: 0.1766 - mape: 202.95
```

```
06 - val_loss: 0.3191 - val_mae: 0.3191 - val_mse: 0.1866 - val_mape: 134.1890
Epoch 36/200
920/944 [============================>.] - ETA: 0s - loss: 0.3133 - mae: 0.3133 - mse: 0.1770 - mape: 250.3448
Epoch 36: val_loss did not improve from 0.29802
944/944 [==============================] - 2s 2ms/step - loss: 0.3128 - mae: 0.3128 - mse: 0.1765 - mape: 247.37
57 - val_loss: 0.3163 - val_mae: 0.3163 - val_mse: 0.1847 - val_mape: 132.0036
Epoch 37/200
931/944 [============================>.] - ETA: 0s - loss: 0.3105 - mae: 0.3105 - mse: 0.1744 - mape: 236.2184
Epoch 37: val_loss improved from 0.29802 to 0.29666, saving model to assets\model6.h5
944/944 [==============================] - 2s 2ms/step - loss: 0.3104 - mae: 0.3104 - mse: 0.1743 - mape: 235.67
02 - val_loss: 0.2967 - val_mae: 0.2967 - val_mse: 0.1579 - val_mape: 137.8961
Epoch 38/200
938/944 [============================>.] - ETA: 0s - loss: 0.3093 - mae: 0.3093 - mse: 0.1734 - mape: 259.3853
Epoch 38: val_loss did not improve from 0.29666
944/944 [==============================] - 2s 2ms/step - loss: 0.3093 - mae: 0.3093 - mse: 0.1733 - mape: 258.55
51 - val_loss: 0.3057 - val_mae: 0.3057 - val_mse: 0.1725 - val_mape: 133.6963
Epoch 39/200
936/944 [============================>.] - ETA: 0s - loss: 0.3104 - mae: 0.3104 - mse: 0.1759 - mape: 224.0755
Epoch 39: val_loss improved from 0.29666 to 0.29516, saving model to assets\model6.h5
944/944 [==============================] - 2s 2ms/step - loss: 0.3103 - mae: 0.3103 - mse: 0.1758 - mape: 223.72
90 - val_loss: 0.2952 - val_mae: 0.2952 - val_mse: 0.1581 - val_mape: 128.5017
Epoch 40/200
920/944 [============================>.] - ETA: 0s - loss: 0.3096 - mae: 0.3096 - mse: 0.1745 - mape: 229.8047
Epoch 40: val_loss did not improve from 0.29516
944/944 [==============================] - 2s 2ms/step - loss: 0.3097 - mae: 0.3097 - mse: 0.1744 - mape: 226.51
88 - val_loss: 0.3172 - val_mae: 0.3172 - val_mse: 0.1753 - val_mape: 125.6230
Epoch 41/200
933/944 [============================>.] - ETA: 0s - loss: 0.3086 - mae: 0.3086 - mse: 0.1730 - mape: 210.2147
Epoch 41: val_loss did not improve from 0.29516
944/944 [==============================] - 2s 2ms/step - loss: 0.3088 - mae: 0.3088 - mse: 0.1732 - mape: 209.40
47 - val_loss: 0.3108 - val_mae: 0.3108 - val_mse: 0.1770 - val_mape: 133.3281
Epoch 42/200
924/944 [============================>.] - ETA: 0s - loss: 0.3089 - mae: 0.3089 - mse: 0.1733 - mape: 201.6021
Epoch 42: val_loss did not improve from 0.29516
944/944 [==============================] - 2s 2ms/step - loss: 0.3087 - mae: 0.3087 - mse: 0.1729 - mape: 201.07
08 - val_loss: 0.3193 - val_mae: 0.3193 - val_mse: 0.1834 - val_mape: 127.5603
Epoch 43/200
943/944 [============================>.] - ETA: 0s - loss: 0.3070 - mae: 0.3070 - mse: 0.1714 - mape: 220.8095
Epoch 43: val_loss did not improve from 0.29516
944/944 [==============================] - 2s 2ms/step - loss: 0.3070 - mae: 0.3070 - mse: 0.1714 - mape: 220.72
29 - val_loss: 0.3003 - val_mae: 0.3003 - val_mse: 0.1630 - val_mape: 136.2322
Epoch 44/200
931/944 [============================>.] - ETA: 0s - loss: 0.3076 - mae: 0.3076 - mse: 0.1716 - mape: 209.1599
Epoch 44: val_loss did not improve from 0.29516
944/944 [==============================] - 2s 2ms/step - loss: 0.3074 - mae: 0.3074 - mse: 0.1716 - mape: 208.54
54 - val_loss: 0.2972 - val_mae: 0.2972 - val_mse: 0.1561 - val_mape: 141.3833
Epoch 45/200
937/944 [============================>.] - ETA: 0s - loss: 0.3076 - mae: 0.3076 - mse: 0.1708 - mape: 185.9859
Epoch 45: val_loss did not improve from 0.29516
944/944 [==============================] - 2s 2ms/step - loss: 0.3080 - mae: 0.3080 - mse: 0.1714 - mape: 185.26
86 - val_loss: 0.3197 - val_mae: 0.3197 - val_mse: 0.1869 - val_mape: 128.0391
Epoch 46/200
935/944 [============================>.] - ETA: 0s - loss: 0.3079 - mae: 0.3079 - mse: 0.1723 - mape: 234.1709
Epoch 46: val_loss did not improve from 0.29516
944/944 [==============================] - 2s 2ms/step - loss: 0.3078 - mae: 0.3078 - mse: 0.1722 - mape: 233.08
22 - val_loss: 0.3049 - val_mae: 0.3049 - val_mse: 0.1694 - val_mape: 133.4604
Epoch 47/200
937/944 [============================>.] - ETA: 0s - loss: 0.3061 - mae: 0.3061 - mse: 0.1704 - mape: 219.7524
Epoch 47: val_loss did not improve from 0.29516
944/944 [==============================] - 2s 2ms/step - loss: 0.3059 - mae: 0.3059 - mse: 0.1704 - mape: 218.86
12 - val_loss: 0.3148 - val_mae: 0.3148 - val_mse: 0.1804 - val_mape: 125.0306
Epoch 48/200
940/944 [============================>.] - ETA: 0s - loss: 0.3050 - mae: 0.3050 - mse: 0.1700 - mape: 190.8168
Epoch 48: val_loss improved from 0.29516 to 0.29333, saving model to assets\model6.h5
944/944 [==============================] - 2s 2ms/step - loss: 0.3050 - mae: 0.3050 - mse: 0.1702 - mape: 190.36
59 - val_loss: 0.2933 - val_mae: 0.2933 - val_mse: 0.1585 - val_mape: 137.1470
Epoch 49/200
927/944 [============================>.] - ETA: 0s - loss: 0.3053 - mae: 0.3053 - mse: 0.1695 - mape: 191.7774
Epoch 49: val_loss did not improve from 0.29333
944/944 [==============================] - 2s 2ms/step - loss: 0.3055 - mae: 0.3055 - mse: 0.1696 - mape: 191.78
37 - val_loss: 0.3049 - val_mae: 0.3049 - val_mse: 0.1729 - val_mape: 132.4962
Epoch 50/200
939/944 [============================>.] - ETA: 0s - loss: 0.3038 - mae: 0.3038 - mse: 0.1682 - mape: 202.9842
Epoch 50: val_loss did not improve from 0.29333
944/944 [==============================] - 2s 2ms/step - loss: 0.3037 - mae: 0.3037 - mse: 0.1681 - mape: 202.41
56 - val_loss: 0.3025 - val_mae: 0.3025 - val_mse: 0.1727 - val_mape: 126.1917
Epoch 51/200
940/944 [============================>.] - ETA: 0s - loss: 0.3046 - mae: 0.3046 - mse: 0.1697 - mape: 251.0234
Epoch 51: val_loss did not improve from 0.29333
944/944 [==============================] - 2s 2ms/step - loss: 0.3046 - mae: 0.3046 - mse: 0.1697 - mape: 250.58
00 - val_loss: 0.2961 - val_mae: 0.2961 - val_mse: 0.1659 - val_mape: 133.5566
Epoch 52/200
929/944 [============================>.] - ETA: 0s - loss: 0.3032 - mae: 0.3032 - mse: 0.1680 - mape: 203.3856
```

```
Epoch 52: val_loss did not improve from 0.29333
944/944 [==============================] - 2s 2ms/step - loss: 0.3030 - mae: 0.3030 - mse: 0.1679 - mape: 202.43
91 - val_loss: 0.3251 - val_mae: 0.3251 - val_mse: 0.1876 - val_mape: 126.1670
Epoch 53/200
936/944 [============================>.] - ETA: 0s - loss: 0.3044 - mae: 0.3044 - mse: 0.1692 - mape: 223.7153
Epoch 53: val_loss did not improve from 0.29333
944/944 [==============================] - 2s 2ms/step - loss: 0.3045 - mae: 0.3045 - mse: 0.1693 - mape: 223.19
54 - val_loss: 0.3020 - val_mae: 0.3020 - val_mse: 0.1642 - val_mape: 131.1067
Epoch 54/200
924/944 [============================>.] - ETA: 0s - loss: 0.3032 - mae: 0.3032 - mse: 0.1669 - mape: 218.1068
Epoch 54: val_loss did not improve from 0.29333
944/944 [==============================] - 2s 2ms/step - loss: 0.3031 - mae: 0.3031 - mse: 0.1668 - mape: 216.35
68 - val_loss: 0.2963 - val_mae: 0.2963 - val_mse: 0.1606 - val_mape: 132.9750
Epoch 55/200
921/944 [============================>.] - ETA: 0s - loss: 0.3032 - mae: 0.3032 - mse: 0.1678 - mape: 187.7034
Epoch 55: val_loss did not improve from 0.29333
944/944 [==============================] - 2s 2ms/step - loss: 0.3034 - mae: 0.3034 - mse: 0.1679 - mape: 186.44
76 - val_loss: 0.3073 - val_mae: 0.3073 - val_mse: 0.1758 - val_mape: 127.9851
Epoch 56/200
930/944 [============================>.] - ETA: 0s - loss: 0.3025 - mae: 0.3025 - mse: 0.1660 - mape: 201.1883
Epoch 56: val_loss did not improve from 0.29333
944/944 [==============================] - 2s 2ms/step - loss: 0.3026 - mae: 0.3026 - mse: 0.1660 - mape: 202.99
69 - val_loss: 0.3020 - val_mae: 0.3020 - val_mse: 0.1672 - val_mape: 123.2599
Epoch 57/200
930/944 [============================>.] - ETA: 0s - loss: 0.3025 - mae: 0.3025 - mse: 0.1668 - mape: 201.4669
Epoch 57: val_loss did not improve from 0.29333
944/944 [==============================] - 2s 2ms/step - loss: 0.3024 - mae: 0.3024 - mse: 0.1669 - mape: 200.16
55 - val_loss: 0.3053 - val_mae: 0.3053 - val_mse: 0.1706 - val_mape: 133.7043
Epoch 58/200
941/944 [============================>.] - ETA: 0s - loss: 0.3032 - mae: 0.3032 - mse: 0.1667 - mape: 214.3297
Epoch 58: val_loss did not improve from 0.29333
944/944 [==============================] - 2s 2ms/step - loss: 0.3031 - mae: 0.3031 - mse: 0.1666 - mape: 214.06
20 - val_loss: 0.3011 - val_mae: 0.3011 - val_mse: 0.1658 - val_mape: 125.4962
Epoch 59/200
933/944 [============================>.] - ETA: 0s - loss: 0.3029 - mae: 0.3029 - mse: 0.1656 - mape: 185.9113
Epoch 59: val_loss did not improve from 0.29333
944/944 [==============================] - 2s 2ms/step - loss: 0.3030 - mae: 0.3030 - mse: 0.1657 - mape: 185.77
05 - val_loss: 0.2960 - val_mae: 0.2960 - val_mse: 0.1581 - val_mape: 128.0708
Epoch 60/200
922/944 [============================>.] - ETA: 0s - loss: 0.3018 - mae: 0.3018 - mse: 0.1653 - mape: 227.1424
Epoch 60: val_loss did not improve from 0.29333
944/944 [==============================] - 2s 2ms/step - loss: 0.3017 - mae: 0.3017 - mse: 0.1651 - mape: 231.41
49 - val_loss: 0.3032 - val_mae: 0.3032 - val_mse: 0.1756 - val_mape: 128.2545
Epoch 61/200
933/944 [============================>.] - ETA: 0s - loss: 0.3010 - mae: 0.3010 - mse: 0.1653 - mape: 199.5078
Epoch 61: val_loss did not improve from 0.29333
944/944 [==============================] - 2s 2ms/step - loss: 0.3010 - mae: 0.3010 - mse: 0.1653 - mape: 199.27
55 - val_loss: 0.3070 - val_mae: 0.3070 - val_mse: 0.1735 - val_mape: 121.7589
Epoch 62/200
924/944 [============================>.] - ETA: 0s - loss: 0.2999 - mae: 0.2999 - mse: 0.1645 - mape: 172.9171
Epoch 62: val_loss did not improve from 0.29333
944/944 [==============================] - 2s 2ms/step - loss: 0.3000 - mae: 0.3000 - mse: 0.1645 - mape: 230.48
70 - val_loss: 0.3058 - val_mae: 0.3058 - val_mse: 0.1731 - val_mape: 127.9454
Epoch 63/200
940/944 [============================>.] - ETA: 0s - loss: 0.2997 - mae: 0.2997 - mse: 0.1635 - mape: 193.0153
Epoch 63: val_loss did not improve from 0.29333
944/944 [==============================] - 2s 2ms/step - loss: 0.2997 - mae: 0.2997 - mse: 0.1636 - mape: 192.63
08 - val_loss: 0.3081 - val_mae: 0.3081 - val_mse: 0.1784 - val_mape: 127.9688
Epoch 64/200
933/944 [============================>.] - ETA: 0s - loss: 0.3003 - mae: 0.3003 - mse: 0.1648 - mape: 212.5576
Epoch 64: val_loss did not improve from 0.29333
944/944 [==============================] - 2s 2ms/step - loss: 0.3000 - mae: 0.3000 - mse: 0.1645 - mape: 214.75
43 - val_loss: 0.3016 - val_mae: 0.3016 - val_mse: 0.1663 - val_mape: 122.0062
Epoch 65/200
927/944 [============================>.] - ETA: 0s - loss: 0.2996 - mae: 0.2996 - mse: 0.1636 - mape: 183.6494
Epoch 65: val_loss did not improve from 0.29333
944/944 [==============================] - 2s 2ms/step - loss: 0.2998 - mae: 0.2998 - mse: 0.1639 - mape: 242.46
40 - val_loss: 0.3002 - val_mae: 0.3002 - val_mse: 0.1663 - val_mape: 131.0544
Epoch 66/200
941/944 [============================>.] - ETA: 0s - loss: 0.2995 - mae: 0.2995 - mse: 0.1642 - mape: 228.7600
Epoch 66: val_loss did not improve from 0.29333
944/944 [==============================] - 2s 2ms/step - loss: 0.2996 - mae: 0.2996 - mse: 0.1643 - mape: 228.37
60 - val_loss: 0.3062 - val_mae: 0.3062 - val_mse: 0.1717 - val_mape: 124.5154
Epoch 67/200
933/944 [============================>.] - ETA: 0s - loss: 0.2981 - mae: 0.2981 - mse: 0.1618 - mape: 225.8136
Epoch 67: val_loss did not improve from 0.29333
944/944 [==============================] - 2s 2ms/step - loss: 0.2982 - mae: 0.2982 - mse: 0.1618 - mape: 225.56
93 - val_loss: 0.3013 - val_mae: 0.3013 - val_mse: 0.1672 - val_mape: 119.0794
```

```
Epoch 68/200
938/944 [============================>.] - ETA: 0s - loss: 0.3001 - mae: 0.3001 - mse: 0.1642 - mape: 217.6231
Epoch 68: val_loss did not improve from 0.29333
944/944 [==============================] - 2s 2ms/step - loss: 0.3000 - mae: 0.3000 - mse: 0.1641 - mape: 217.23
98 - val_loss: 0.2972 - val_mae: 0.2972 - val_mse: 0.1627 - val_mape: 128.0505

Epoch 68: early stopping
Training Loss: 0.2790, Training MAE: 0.2790, Training MSE: 0.1457, Training MAPE: 149.1391
Testing Loss: 0.2943, Testing MAE: 0.2943, Testing MSE: 0.1597, Testing MAPE: 324.2614
Training time: 148.8283133506775 seconds
```

In [70]:
```python
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(10, 8))

ax.boxplot(loss_data, labels=labels)

# Set plot title and axis labels
ax.set_title('Training and Validation Metrics')
ax.set_xlabel('Metric Type')
ax.set_ylabel('Metric Value')

# Save the figure as a PDF
fig.savefig('training_validation_metrics.pdf')

plt.show()
```



In [71]:
```python
# Extract loss values for each epoch from history
train_loss = history.history['loss']
val_loss = history.history['val_loss']


# Create a list of loss values with their corresponding labels
loss_data = [train_loss, val_loss]
labels = ['Training Loss', 'Validation Loss']

# Create a box plot
plt.boxplot(loss_data, labels=labels)

# Set plot title and axis labels
plt.title('Training and Validation Loss')
plt.xlabel('Loss Type')
```

```
plt.ylabel('MAE')
fig.savefig('Training and Validation Loss.pdf')
plt.show()
```

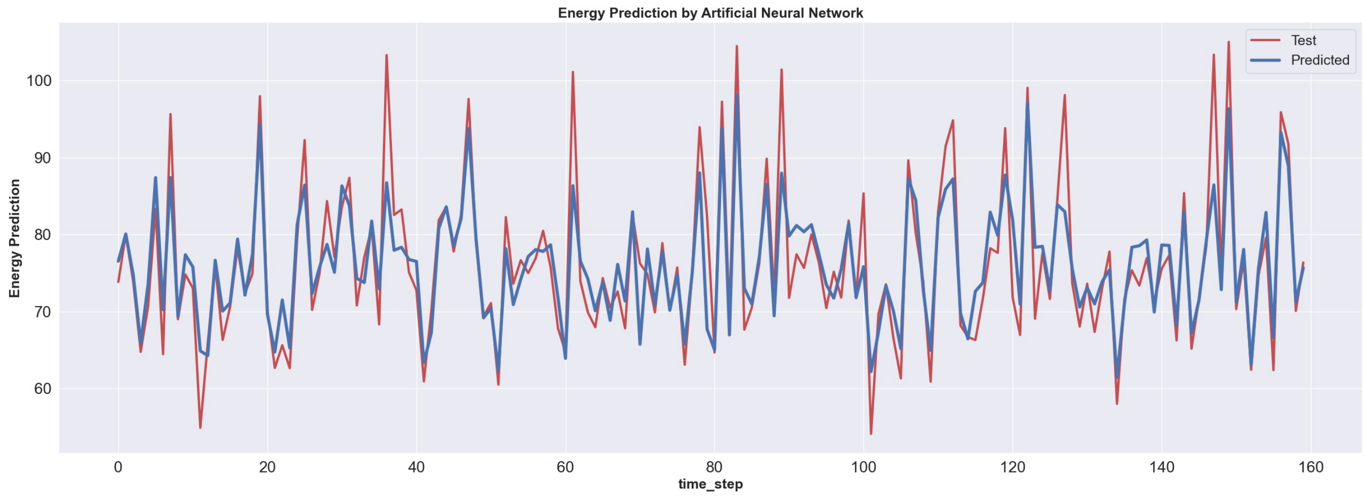
In [72]:

```
X_test1 = s.fit_transform(X_test)

y_hat6 = model6.predict(X_test1)
y_hat6n = s1.inverse_transform(y_hat6.reshape(-1, 1))
y_test1 = s1.inverse_transform(y_test)
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_squared_error,mean_absolute_percentage_error
import sklearn.metrics as metrics
from math import sqrt
RMSE = sqrt(mean_squared_error(y_test1, y_hat6n))
print("RMSE:",RMSE)
#mse

#mape
mape = mean_absolute_percentage_error(y_test1, y_hat6n)
print("MAPE:",mape)
print ("Percentual:", metrics.mean_absolute_error(y_test1,y_hat6n)/y_test1.mean()*100, "%")
```

```
450/450 [==============================] - 1s 969us/step
RMSE: 3.9718766650915436
MAPE: 0.037870907971220995
Percentual: 3.817981865660743 %
```

In [73]:

```
import numpy as np
import matplotlib.pyplot as plt

# assuming y_test and y_hat1 are already defined

# downsample data to reduce number of data points
downsample_factor = 90
y_test_downsampled = y_test1[::downsample_factor]
y_hat6_downsampled = y_hat6n[::downsample_factor]

# create line plot
plt.figure(figsize=(30,10))
plt.plot(y_test_downsampled , 'r-', linewidth=3)
plt.plot(y_hat6_downsampled,'b-' , linewidth=4)
plt.xlabel('time_step', fontsize = 18, fontweight="bold")
plt.ylabel('Energy Prediction', fontsize = 18, fontweight="bold")
plt.legend (('Test','Predicted'), fontsize = 18)
plt.title("Energy Prediction by Artificial Neural Network", fontsize = 18, fontweight="bold")
plt.tick_params(axis='both', which='major', labelsize=20)
plt.show()
plt.savefig('PREDICTION by Artificial Neural Network Line Plot.png', format='png')
plt.savefig('PREDICTION by Artificial Neural Network Line Plot.pdf', format='pdf')
```



<Figure size 640x480 with 0 Axes>

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js