PHASE3: development phase part 1

INTRODUCTION TO LOADING AND PREPROCESSING THE DATA

Loading and preprocessing electricity price prediction datasets involves preparing the data for analysis or machine learning models. In Python, this is commonly done using libraries such as `pandas` for data manipulation and `scikit-learn` for preprocessing. Here's a concise introduction:

**1.loading dataset**

**code:**

**import matplotlib as plt**

**import numpy as np**

**import pandas as pd**

**import seaborn as sns**

**from xgboost import XGBRegressor**

**from sklearn.metrics import mean\_absolute\_error**

**from sklearn.ensemble import RandomForestRegressor**

**from sklearn.model\_selection import train\_test\_split**

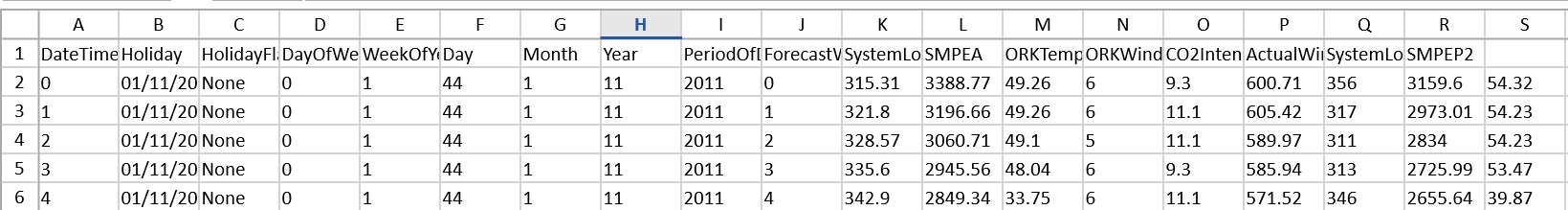
**from sklearn.model\_selection import cross\_val\_score**

**X = pd.read\_csv('/content/electricity price 2 - Sheet1.csv')**

**X\_full = X.copy()**

**X.head()**

**Output:**

****

**2.INFORMATION ABOUT THE DATASET:**

**CODE: X.info()**

**OUTPUT:**

**<class 'pandas.core.frame.DataFrame'>**

**RangeIndex: 5 entries, 0 to 4**

**Data columns (total 19 columns):**

**# Column Non-Null Count Dtype**

**--- ------ -------------- -----**

**0 DateTime 5 non-null int64**

**1 Holiday 5 non-null object**

**2 HolidayFlag 5 non-null object**

**3 DayOfWeek 5 non-null int64**

**4 WeekOfYear 5 non-null int64**

**5 Day 5 non-null int64**

**6 Month 5 non-null int64**

**7 Year 5 non-null int64**

**8 PeriodOfDay 5 non-null int64**

**9 ForecastWindProduction 5 non-null int64**

**10 SystemLoadEA 5 non-null float64**

**11 SMPEA 5 non-null float64**

**12 ORKTemperature 5 non-null float64**

**13 ORKWindspeed 5 non-null int64**

**14 CO2Intensity 5 non-null float64**

**15 ActualWindProduction 5 non-null float64**

**16 SystemLoadEP2 5 non-null int64**

**17 SMPEP2 5 non-null float64**

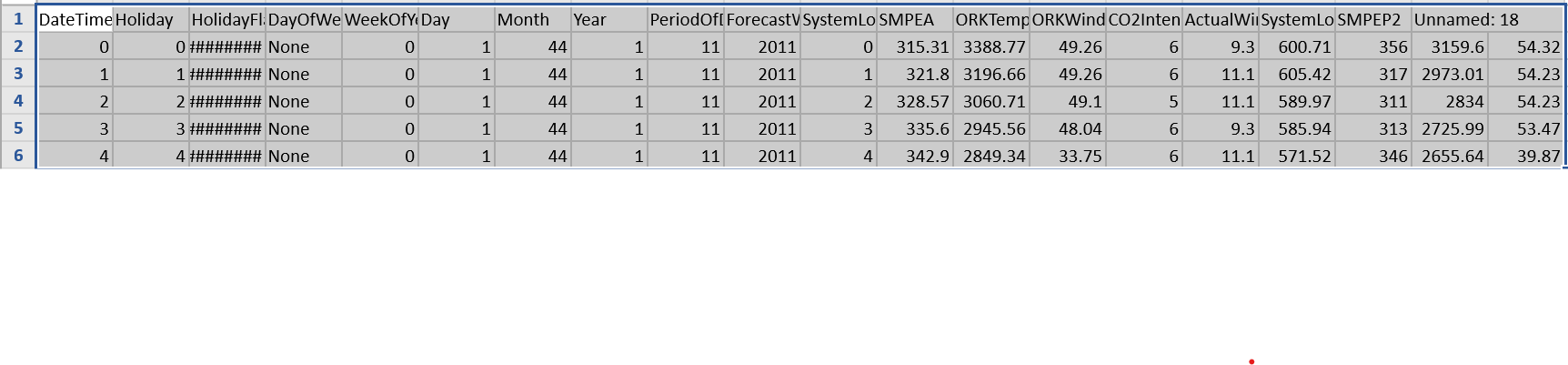
**18 Unnamed: 18 5 non-null float64**

**dtypes: float64(7), int64(10), object(2)**

**memory usage: 888.0+ bytes**

**3.code: X.tail()**

**Output:**

****

**4.replacing the null values**

**Code:**

**X=X.replace('?', np.NaN)**

**X.isnull().sum()**

**Output:**

**DateTime 0**

**Holiday 0**

**HolidayFlag 0**

**DayOfWeek 0**

**WeekOfYear 0**

**Day 0**

**Month 0**

**Year 0**

**PeriodOfDay 0**

**ForecastWindProduction 0**

**SystemLoadEA 0**

**SMPEA 0**

**ORKTemperature 0**

**ORKWindspeed 0**

**CO2Intensity 0**

**ActualWindProduction 0**

**SystemLoadEP2 0**

**SMPEP2 0**

**Unnamed: 18 0**

**dtype: int64**

**5.CHECKING THE DATE AND TIME IN AN INDEX:**

**CODE:**

**X['DateTime']=pd.to\_datetime(X['DateTime'],dayfirst=True)**

**X=X.dropna()**

**OUTPUT:**

**TRUE**

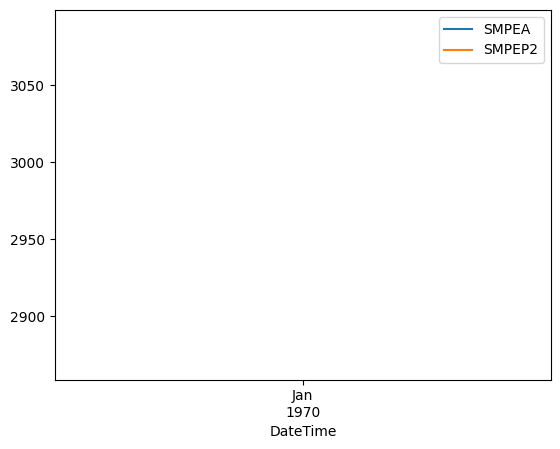
**6.MEAN OF SMPEA AND SMPEA2:**

**CODE:**

**X\_eda=X.set\_index('DateTime')**

**X\_eda[['SMPEA','SMPEP2']].resample('M').mean().plot()**

**OUTPUT:**

****

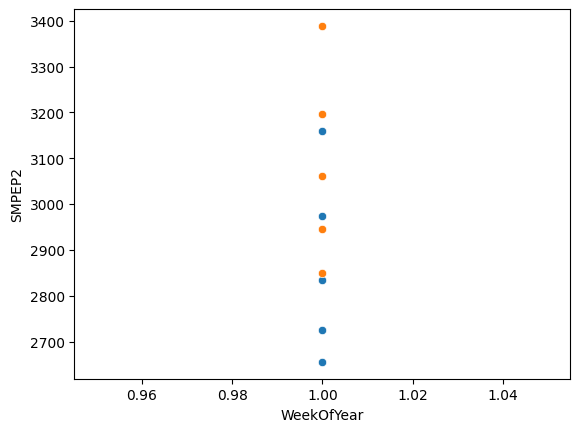
**7.SCATTER PLOTTING THE SMPEA AND SMPEA2:**

**CODE:**

**sns.scatterplot(data=X\_eda, x='WeekOfYear', y='SMPEA2')**

**sns.scatterplot(data=X\_eda, x='WeekOfYear', y='SMPEA')**

**OUTPUT:**

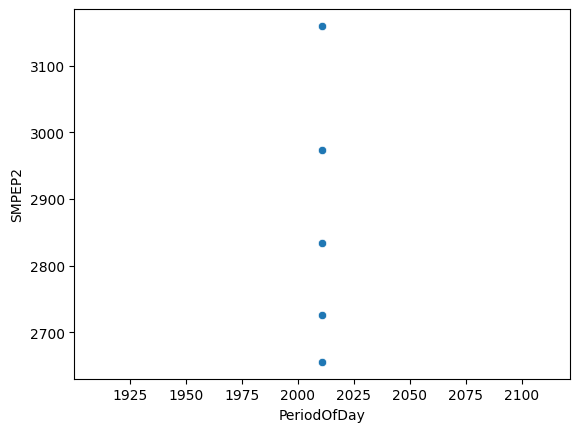
****

**8.SCATTER PLOTTING PERIOD OF DAY AND SMPEA2:**

**CODE:**

**sns.scatterplot(data=X\_eda, x='PeriodOfDay', y='SMPEP2')**

**OUTPUT:**

****

**9.FINDING DUPLICATES:**

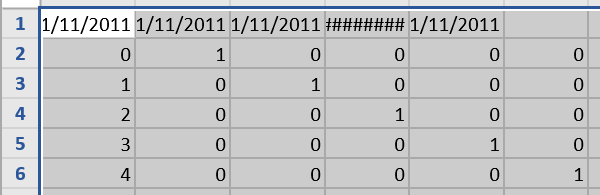
**CODE:**

**X.Holiday.nunique()**

**one\_hot = pd.get\_dummies(X['Holiday'])**

**X=X.drop('Holiday', axis=1)**

**OUTPUT:**

****

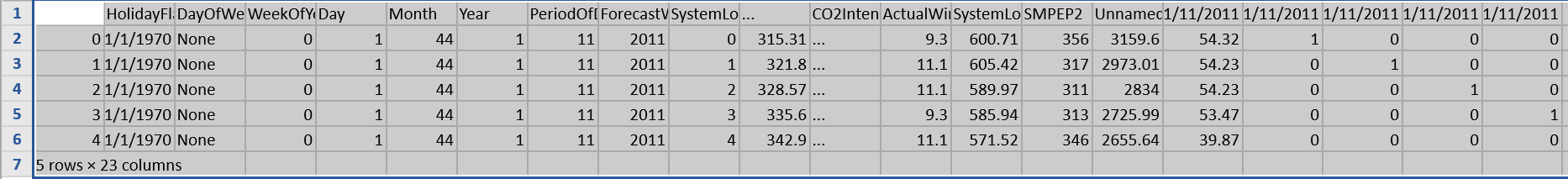
**10.MERGING LEFT INDEX AND RIGHT INDEX:**

**CODE:**

**X\_merged=X.merge(one\_hot, left\_index=True, right\_index=True)**

**X\_merged.head()**

**OUTPUT:**

****

**11.CORRELATING THE DATA:**

**CODE:**

**corr\_data.head(10)**

**OUTPUT:**

**SMPEP2**

**SMPEP2 1.000000**

**SMPEA 0.999291**

**ActualWindProduction 0.862996**

**01/11/2011 0:00 0.809083**

**ORKTemperature 0.692516**

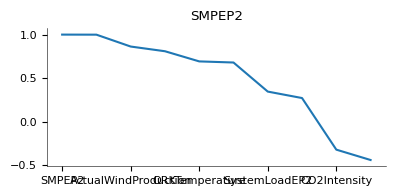
**Unnamed: 18 0.679677**

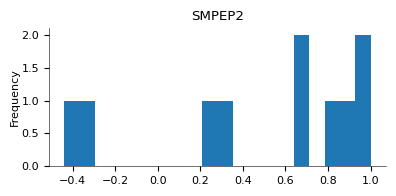
**SystemLoadEP2 0.345972**

**01/11/2011 0:30 0.271911**

**CO2Intensity -0.320300**

**GRAPHICAL REPRESENTATION:**

****

****

**12.CLEANING AND TESTING THE DATA:**

**CODE:**

**X\_clean=X\_merged[['SMPEA','SystemLoadEP2','SystemLoadEA', 'PeriodOfDay']]**

**X\_train, X\_valid, y\_train, y\_valid = train\_test\_split(X\_clean, y)**

**my\_model = XGBRegressor(random\_state=63)**

**my\_model.fit(X\_train, y\_train)**

**OUTPUT:**

**XGBRegressor**

**XGBRegressor(base\_score=None, booster=None, callbacks=None,**

**colsample\_bylevel=None, colsample\_bynode=None,**

**colsample\_bytree=None, device=None, early\_stopping\_rounds=None,**

**enable\_categorical=False, eval\_metric=None, feature\_types=None,**

**gamma=None, grow\_policy=None, importance\_type=None,**

**interaction\_constraints=None, learning\_rate=None, max\_bin=None,**

**max\_cat\_threshold=None, max\_cat\_to\_onehot=None,**

**max\_delta\_step=None, max\_depth=None, max\_leaves=None,**

**min\_child\_weight=None, missing=nan, monotone\_constraints=None,**

**multi\_strategy=None, n\_estimators=None, n\_jobs=None,**

**num\_parallel\_tree=None, random\_state=63, ...)**

**13.PREACTING ACCURACY AND MEAN:**

**CODE:**

**predictions = my\_model.predict(X\_valid)**

**mae\_XBG=mean\_absolute\_error(predictions,y\_valid)**

**mean\_y=X\_merged.SMPEP2.mean()**

**print("Mean Absolute Error: " + str(mae\_XBG))**

**print('prediction accuracy: ' +str(1-mae\_XBG/mean\_y))**

**OUTPUT:**

**Mean Absolute Error: 186.5909667968749**

**prediction accuracy: 0.9351790593918922**

**14.ANALYSING WITH RANDOM FOREST:**

**CODE:**

**forest\_model = RandomForestRegressor(random\_state=63)**

**forest\_model.fit(X\_train, y\_train)**

**fores\_preds = forest\_model.predict(X\_valid)**

**forest\_mae=mean\_absolute\_error(y\_valid, fores\_preds)**

**print("Mean Absolute Error: " + str(forest\_mae))**

**print('prediction accuracy: ' +str(1-forest\_mae/mean\_y))**

**OUTPUT:**

**Mean Absolute Error: 276.9082999999973**

**prediction accuracy: 0.9038031863153808**

**DATASETLINK:"C:\Users\rocky\Downloads\electricity price 2 - Sheet1.csv"**

**SUMMARY:**

**Loading and preprocessing data sets is a crucial step in machine learning. Here's a theoretical summary of the process:**

**1. \*Loading Data:\***

**- \*Data Source:\* Identify the source of your data, which could be CSV files, databases, APIs, or other formats.**

**- \*Data Format:\* Understand the structure of your data, including features and labels.**

**2. \*Import Libraries:\***

**- \*Python Libraries:\* Utilize libraries like Pandas for data manipulation and NumPy for numerical operations.**

**3. \*Loading into Memory:\***

**- \*Read Data:\* Use appropriate functions (e.g., `pd.read\_csv()` in Pandas) to load the data into memory.**

**- \*Data Exploration:\* Perform basic exploratory data analysis to understand the characteristics of the data.**

**4. \*Data Preprocessing:\***

**- \*Handling Missing Values:\* Decide on strategies for dealing with missing data (e.g., imputation or removal).**

**- \*Data Cleaning:\* Address any inconsistencies or errors in the data.**

**- \*Encoding Categorical Variables:\* Convert categorical variables into numerical format, often through one-hot encoding.**

**- \*Feature Scaling:\* Normalize or standardize numerical features to bring them to a similar scale.**

**5. \*Splitting Data:\***

**- \*Training and Testing Sets:\* Split the data into training and testing sets to assess model performance.**

**- \*Cross-Validation (Optional):\* Consider cross-validation techniques for more robust model evaluation.**

**6. \*Feature Engineering (Optional):\***

**- \*Create New Features:\* Derive additional features that may enhance the model's performance.**

**7. \*Data Transformation:\***

**- \*Reshaping Data:\* Depending on the model requirements, reshape the data (e.g., for convolutional neural networks in image data).**

**8. \*Normalization (Optional):\***

**- \*Data Transformation:\* Apply normalization techniques if needed, such as scaling pixel values for images.**

**9. \*Data Augmentation (Optional):\***

**- \*Image Data Augmentation:\* For image datasets, consider techniques like rotation, flipping, or zooming to increase training data diversity.**

**10. \*Finalization:\***

**- \*Save Processed Data:\* Save the preprocessed data to ensure reproducibility.**

**- \*Documentation:\* Document the preprocessing steps and transformations applied.**

**By following these steps, you ensure that your data is ready for training machine learning models, helping to improve model accuracy and generalization.**