INTERNSHIP - I (ELECTRICITY PRICE PREDICTION)

Exploratory Data Analysis (EDA)

Workflow

1) Importing Necessary Packages 2) Data Exploration 3) Data Cleaning 4) Data Visualization 5) Data Wrangling 6) Extraction of Cleaned Dataset for Model Building

Importing Necessary Packages

In [2]:
```
import numpy as np # linear algebra
import pandas as pd # data processing

# Visualization
import shap
%matplotlib inline
import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
import eli5
from eli5.sklearn import PermutationImportance

# Data Wrangling
pd.set_option('display.max_columns',50)
```

Data Exploration

In [3]:
```
df = pd.read_csv("C:/Users/Arul Selvaraj/Desktop/dval/energy_dataset.csv")
# Preview of data
df.head(5)
```

Out[3]:

	time	generation biomass	generation fossil brown coal/lignite	generation fossil coal-derived gas	generation fossil gas	generation fossil hard coal	generation fossil oil	generation fossil oil shale	generation fossil peat	generation geothermal	generation hydro pumped storage aggregated
0	2015-01-01 00:00:00+01:00	447.0	329.0	0.0	4844.0	4821.0	162.0	0.0	0.0	0.0	NaN
1	2015-01-01 01:00:00+01:00	449.0	328.0	0.0	5196.0	4755.0	158.0	0.0	0.0	0.0	NaN
2	2015-01-01 02:00:00+01:00	448.0	323.0	0.0	4857.0	4581.0	157.0	0.0	0.0	0.0	NaN
3	2015-01-01 03:00:00+01:00	438.0	254.0	0.0	4314.0	4131.0	160.0	0.0	0.0	0.0	NaN
4	2015-01-01 04:00:00+01:00	428.0	187.0	0.0	4130.0	3840.0	156.0	0.0	0.0	0.0	NaN

In [4]:
```
# Describe to show overview of data
df.describe()
```

Out[4]:

	generation biomass	generation fossil brown coal/lignite	generation fossil coal-derived gas	generation fossil gas	generation fossil hard coal	generation fossil oil	generation fossil oil shale	generation fossil peat	generation geothermal	generation hydro pumped storage aggregated	g
count	35045.000000	35046.000000	35046.0	35046.000000	35046.000000	35045.000000	35046.0	35046.0	35046.0	0.0	3
mean	383.513540	448.059208	0.0	5622.737488	4256.065742	298.319789	0.0	0.0	0.0	NaN	
std	85.353943	354.568590	0.0	2201.830478	1961.601013	52.520673	0.0	0.0	0.0	NaN	
min	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0.0	0.0	NaN	
25%	333.000000	0.000000	0.0	4126.000000	2527.000000	263.000000	0.0	0.0	0.0	NaN	
50%	367.000000	509.000000	0.0	4969.000000	4474.000000	300.000000	0.0	0.0	0.0	NaN	
75%	433.000000	757.000000	0.0	6429.000000	5838.750000	330.000000	0.0	0.0	0.0	NaN	
max	592.000000	999.000000	0.0	20034.000000	8359.000000	449.000000	0.0	0.0	0.0	NaN	

In [5]:
```
# Info to show datatype, nulls, and count
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35064 entries, 0 to 35063
Data columns (total 29 columns):
 #   Column                                       Non-Null Count  Dtype  
---  ------                                       --------------  -----  
 0   time                                         35064 non-null  object 
 1   generation biomass                           35045 non-null  float64
 2   generation fossil brown coal/lignite         35046 non-null  float64
 3   generation fossil coal-derived gas           35046 non-null  float64
 4   generation fossil gas                        35046 non-null  float64
 5   generation fossil hard coal                  35046 non-null  float64
 6   generation fossil oil                        35045 non-null  float64
 7   generation fossil oil shale                  35046 non-null  float64
 8   generation fossil peat                       35046 non-null  float64
 9   generation geothermal                        35046 non-null  float64
 10  generation hydro pumped storage aggregated   0 non-null      float64
 11  generation hydro pumped storage consumption  35045 non-null  float64
 12  generation hydro run-of-river and poundage   35045 non-null  float64
 13  generation hydro water reservoir             35046 non-null  float64
 14  generation marine                            35045 non-null  float64
 15  generation nuclear                           35047 non-null  float64
 16  generation other                             35046 non-null  float64
 17  generation other renewable                   35046 non-null  float64
 18  generation solar                             35046 non-null  float64
 19  generation waste                             35045 non-null  float64
 20  generation wind offshore                     35046 non-null  float64
 21  generation wind onshore                      35046 non-null  float64
 22  forecast solar day ahead                     35064 non-null  float64
 23  forecast wind offshore eday ahead            0 non-null      float64
 24  forecast wind onshore day ahead              35064 non-null  float64
 25  total load forecast                          35064 non-null  float64
 26  total load actual                            35028 non-null  float64
 27  price day ahead                              35064 non-null  float64
 28  price actual                                 35064 non-null  float64
dtypes: float64(28), object(1)
memory usage: 7.8+ MB
```

In [6]:
```
# To check null % amongst columns
round((df.isnull().sum()/len(df)*100),2)
```

Out[6]:
```
time                                          0.00
generation biomass                            0.05
generation fossil brown coal/lignite          0.05
generation fossil coal-derived gas            0.05
generation fossil gas                         0.05
generation fossil hard coal                   0.05
generation fossil oil                         0.05
generation fossil oil shale                   0.05
generation fossil peat                        0.05
generation geothermal                         0.05
generation hydro pumped storage aggregated  100.00
generation hydro pumped storage consumption   0.05
generation hydro run-of-river and poundage    0.05
generation hydro water reservoir              0.05
generation marine                             0.05
generation nuclear                            0.05
generation other                              0.05
generation other renewable                    0.05
generation solar                              0.05
generation waste                              0.05
generation wind offshore                      0.05
generation wind onshore                       0.05
forecast solar day ahead                      0.00
forecast wind offshore eday ahead           100.00
forecast wind onshore day ahead               0.00
total load forecast                           0.00
total load actual                             0.10
price day ahead                               0.00
price actual                                  0.00
dtype: float64
```

In [7]:
```
# To find correlation
df.corr()
```

```
The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Selec
t only valid columns or specify the value of numeric_only to silence this warning.
```

Out[7]:

	generation biomass	generation fossil brown coal/lignite	generation fossil coal-derived gas	generation fossil gas	generation fossil hard coal	generation fossil oil	generation fossil oil shale	generation fossil peat	generation geothermal	generation hydro pumped storage aggregated	g con
generation biomass	1.000000	0.229809	NaN	-0.021660	0.433522	0.459530	NaN	NaN	NaN	NaN	
generation fossil brown coal/lignite	0.229809	1.000000	NaN	0.499808	0.768710	0.314869	NaN	NaN	NaN	NaN	
generation fossil coal-derived gas	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
generation fossil gas	-0.021660	0.499808	NaN	1.000000	0.541635	0.309623	NaN	NaN	NaN	NaN	
generation fossil hard coal	0.433522	0.768710	NaN	0.541635	1.000000	0.440837	NaN	NaN	NaN	NaN	
generation fossil oil	0.459530	0.314869	NaN	0.309623	0.440837	1.000000	NaN	NaN	NaN	NaN	
generation fossil oil shale	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
generation fossil peat	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
generation geothermal	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
generation hydro pumped storage aggregated	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
generation hydro pumped storage consumption	-0.044898	-0.323771	NaN	-0.420646	-0.406116	-0.331011	NaN	NaN	NaN	NaN	
generation hydro run-of-river and poundage	-0.284877	-0.525005	NaN	-0.271527	-0.497940	-0.106753	NaN	NaN	NaN	NaN	
generation hydro water reservoir	-0.033675	-0.229455	NaN	0.060173	-0.157677	0.160465	NaN	NaN	NaN	NaN	
generation marine	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
generation nuclear	-0.021279	-0.008440	NaN	-0.112904	-0.023930	0.015619	NaN	NaN	NaN	NaN	
generation other	0.658488	0.097600	NaN	-0.066279	0.264383	0.375046	NaN	NaN	NaN	NaN	
generation other renewable	-0.560588	0.104552	NaN	0.334880	-0.019426	-0.115087	NaN	NaN	NaN	NaN	
generation solar	-0.004687	0.040447	NaN	0.074716	0.046185	0.100211	NaN	NaN	NaN	NaN	
generation waste	-0.346343	0.282810	NaN	0.275053	0.170235	-0.175741	NaN	NaN	NaN	NaN	
generation wind offshore	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
generation wind onshore	-0.068658	-0.434118	NaN	-0.397298	-0.441853	-0.051787	NaN	NaN	NaN	NaN	
forecast solar day ahead	-0.008713	0.042306	NaN	0.080171	0.047356	0.096435	NaN	NaN	NaN	NaN	
forecast wind offshore eday ahead	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
forecast wind onshore day ahead	-0.072368	-0.436031	NaN	-0.397303	-0.444490	-0.058244	NaN	NaN	NaN	NaN	
total load forecast	0.085216	0.278503	NaN	0.543711	0.394291	0.498637	NaN	NaN	NaN	NaN	

	generation biomass	generation fossil brown coal/lignite	generation fossil coal-derived gas	generation fossil gas	generation fossil hard coal	generation fossil oil	generation fossil oil shale	generation fossil peat	generation geothermal	generation hydro pumped storage aggregated	g con
total load actual	0.083288	0.280461	NaN	0.548913	0.396564	0.497089	NaN	NaN	NaN	NaN	
price day ahead	0.108945	0.567905	NaN	0.640895	0.671596	0.292793	NaN	NaN	NaN	NaN	
price actual	0.142369	0.364088	NaN	0.461706	0.465641	0.284679	NaN	NaN	NaN	NaN	

In [8]:
```
# Correlation with respect to target variable
correlations = df.corr(method='pearson')
print(correlations['price actual'].sort_values(ascending=False).to_string())
```

```
price actual                                  1.000000
price day ahead                               0.732155
generation fossil hard coal                   0.465641
generation fossil gas                         0.461706
total load actual                             0.436127
total load forecast                           0.435864
generation fossil brown coal/lignite          0.364088
generation fossil oil                         0.284679
generation other renewable                    0.256181
generation waste                              0.169605
generation biomass                            0.142369
forecast solar day ahead                      0.101402
generation other                              0.100048
generation solar                              0.098488
generation hydro water reservoir              0.071549
generation nuclear                           -0.052596
generation hydro run-of-river and poundage   -0.137106
generation wind onshore                      -0.220830
forecast wind onshore day ahead              -0.221706
generation hydro pumped storage consumption  -0.426417
generation fossil coal-derived gas                 NaN
generation fossil oil shale                        NaN
generation fossil peat                             NaN
generation geothermal                              NaN
generation hydro pumped storage aggregated         NaN
generation marine                                  NaN
generation wind offshore                           NaN
forecast wind offshore eday ahead                  NaN
```

```
The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Selec
t only valid columns or specify the value of numeric_only to silence this warning.
```

Data Cleaning

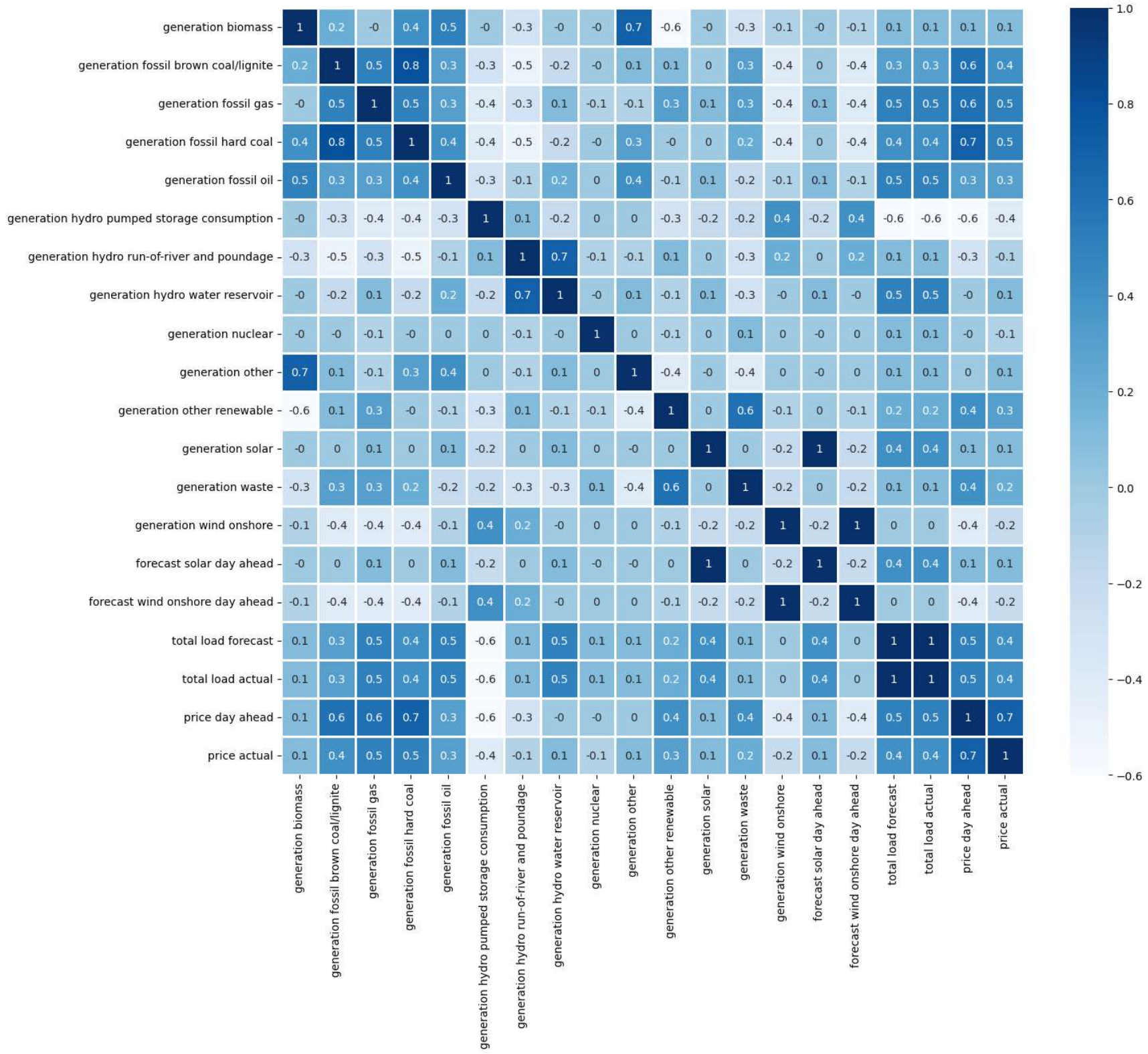
In [9]:
```
# To drop columns with NaN
zero_val_cols = ['generation marine',
                 'generation geothermal',
                 'generation fossil peat',
                 'generation wind offshore',
                 'generation fossil oil shale',
                 'forecast wind offshore eday ahead',
                 'generation fossil coal-derived gas',
                 'generation hydro pumped storage aggregated']
```

In [10]:
```
# To drop columns with zero
heat_map_features = df.drop(columns=zero_val_cols,axis=1)
```

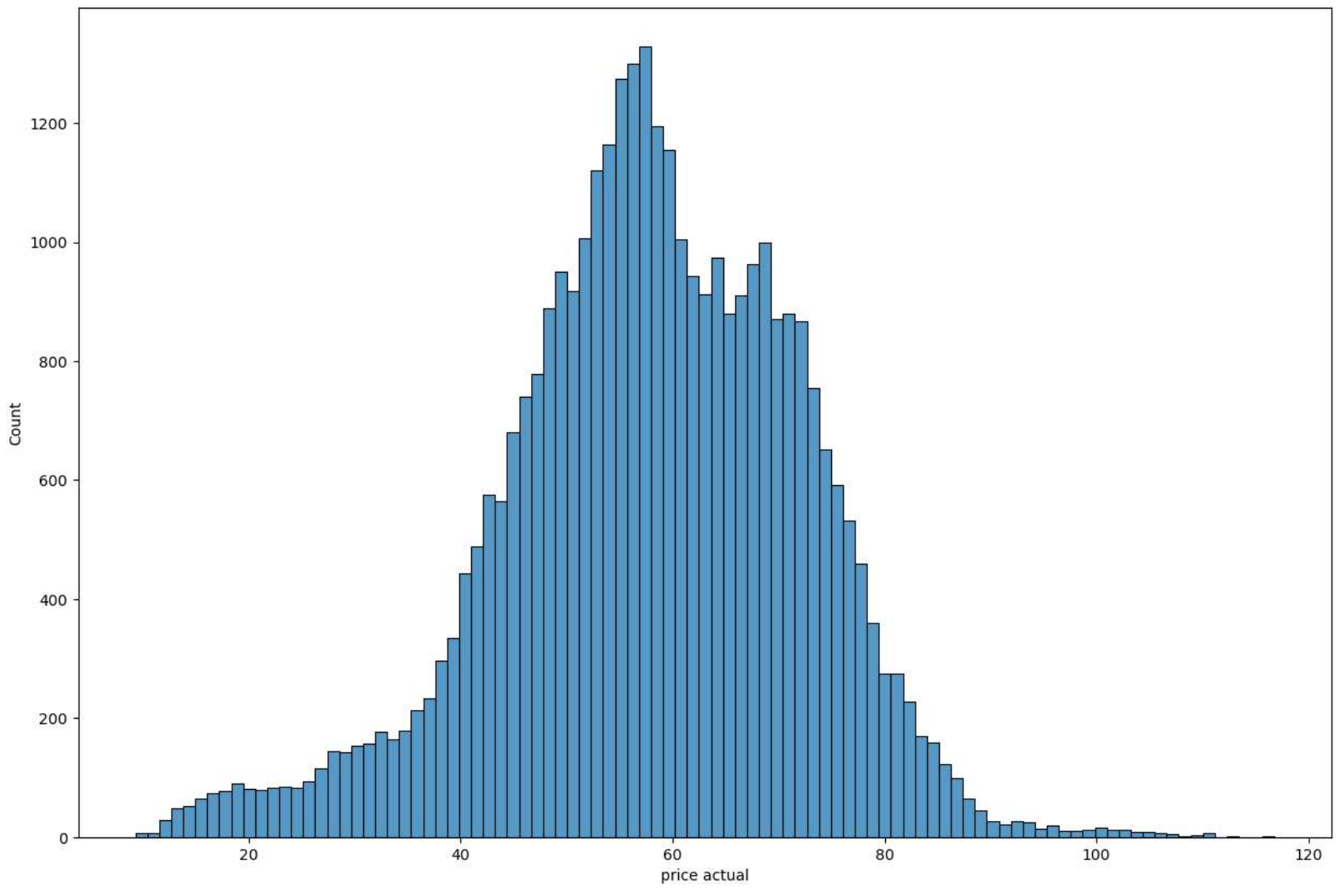
Data Visualization

In [13]:
```
# Heatmap
plt.figure(figsize=(15,12.5))
sns.heatmap(round(heat_map_features.corr(),1),annot=True,cmap='Blues',linewidth=0.9)
plt.show()
```

```
The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Selec
t only valid columns or specify the value of numeric_only to silence this warning.
```



In [14]:
```
# Histogram
plt.figure(figsize=(15,10))
sns.histplot(df,x='price actual')
plt.show()
```



Data Wrangling

In [15]:
```
# Creating a user-defined function to perform data wrangling
def wrangle(filepath):

    # Read in the data, parse dates and set the index
    df = pd.read_csv(filepath, parse_dates=['time'], index_col='time')

    # Rename columns by replacing all - or blank space with _
    df.columns = df.columns.str.replace(' ','_').str.replace('-','_')

    # Make the index DT
    df.index = pd.to_datetime(df.index, utc=True)

    # Drop all columns with data leakage, or 90% + null
    df.drop(columns=['price_day_ahead',
                     'generation_marine',
                     'total_load_forecast',
                     'generation_geothermal',
                     'generation_fossil_peat',
                     'generation_wind_offshore',
                     'forecast_solar_day_ahead',
                     'generation_fossil_oil_shale',
                     'forecast_wind_onshore_day_ahead',
                     'forecast_wind_offshore_eday_ahead',
                     'generation_fossil_coal_derived_gas',
                     'generation_hydro_pumped_storage_aggregated'],inplace=True)

    # Drop Outlier row 2014 for plotting
    df = df.drop(pd.Timestamp('2014-12-31 23:00:00+00:00'))

    # Sort index
    df = df.sort_index()

    # Set conditional satements for filtering times of month to season value
    condition_winter = (df.index.month>=1)&(df.index.month<=3)
    condtion_spring = (df.index.month>=4)&(df.index.month<=6)
    condition_summer = (df.index.month>=7)&(df.index.month<=9)
    condition_automn = (df.index.month>=10)@(df.index.month<=12)

    # Create column in dataframe that inputs the season based on the conditions created above
    df['season'] = np.where(condition_winter,'winter',
                            np.where(condtion_spring,'spring',
                                     np.where(condition_summer,'summer',
                                              np.where(condition_automn,'automn',np.nan))))

    return df

# Applying the wrangle function to the dataset
df=wrangle('C:/Users/Arul Selvaraj/Desktop/dval/energy_dataset.csv')
```

Extracting Cleaned Dataset for Model Building

In [16]:
```
# To get a cleaned dataset
df.to_csv("energy_cleaned_dataset.csv")
```