paraguay-energy-consump

October 7, 2024

1 Task 1: Data Exploration

1.0.1 First rows and useful statistics

Displaying the first rows of the dataset.

```
              datetime substation feeder  consumption
0  2017-01-01T00:00:00          A     A1    64.671363
1  2017-01-01T01:00:00          A     A1    58.000000
2  2017-01-01T02:00:00          A     A1    58.000000
3  2017-01-01T03:00:00          A     A1    58.000000
4  2017-01-01T04:00:00          A     A1    52.000000
5  2017-01-01T05:00:00          A     A1    45.000000
6  2017-01-01T06:00:00          A     A1    42.000000
7  2017-01-01T07:00:00          A     A1    43.000000
8  2017-01-01T08:00:00          A     A1    45.000000
9  2017-01-01T09:00:00          A     A1    44.000000
```

Displaying the shape of the dataset on form (rows, column)

```
(1928520, 4)
```

From this we can see that they have split the time-series in intervals of 1 hour, and a total of 24 points of data per day. Furthermore, the dataset also points to a substation. All the data from substation A is the following:

```
(70128, 4)
```

With 70 128 rows of data for one subregion we have the following dates, weeks and years spanning in the dataset:

```
Days in dataset: 2922.0
Weeks in dataset: 417.42857142857144
Years in dataset: 8.027472527472527
```

Creating statistics of the dataset

```
        consumption
count  1.884960e+06
mean   1.018534e+02
std    6.220330e+01
```

```
 min    1.264627e+00
 25%    5.300000e+01
 50%    8.825000e+01
 75%    1.410000e+02
 max    4.120000e+02
```

Identifyting missing values

```
datetime           0
substation         0
feeder             0
consumption    43560
dtype: int64
```

Identifying unique values

```
 113867
```

There are 113 864 unique values for the consumption columm.

Outliers

Have chosen to use Interquartile Range (IQR) as this is more suitable to handle skewed data. This is certainly the case here, as electricity consumption varies from day to day, as well as the season and weather conditions. If the data had been normally ditributed, calculating ouliers using z-score would be more useful.

```
                    datetime substation feeder  consumption
71104    2017-02-10T16:00:00          B     B1        285.0
71392    2017-02-22T16:00:00          B     B1        288.0
71600    2017-03-03T08:00:00          B     B1        288.0
71601    2017-03-03T09:00:00          B     B1        281.0
71603    2017-03-03T11:00:00          B     B1        307.0
…                        …          …    …            …
1892579  2020-11-25T11:00:00          N     N3        276.0
1892582  2020-11-25T14:00:00          N     N3        306.0
1892583  2020-11-25T15:00:00          N     N3        288.0
1892966  2020-12-11T14:00:00          N     N3        277.0
1892967  2020-12-11T15:00:00          N     N3        277.0

[17444 rows x 4 columns]
```

Of the total of 1 928 520 total entries, there are 17 444 outliers.

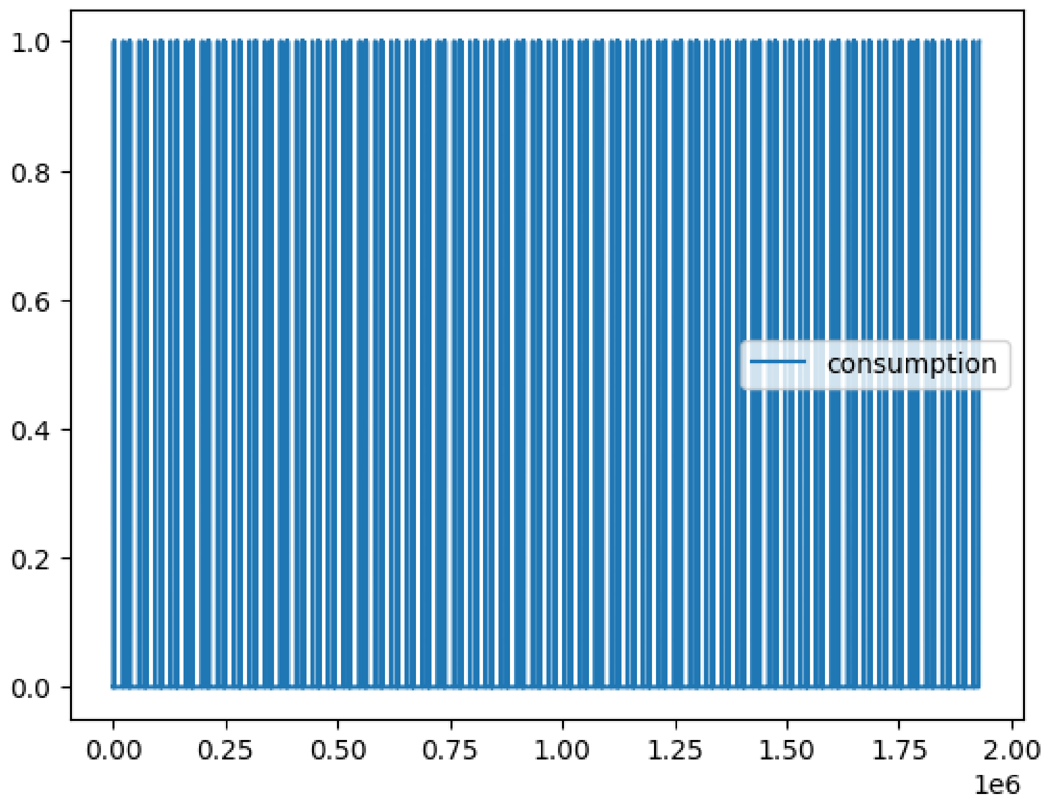
```
 0.9045
```

This accounts for approximately 0.9045 %

2 Task 2: Data Cleaning

Displaying where the missing values are located in the dataset.

```
<matplotlib.legend.Legend at 0x2d06817b020>
```

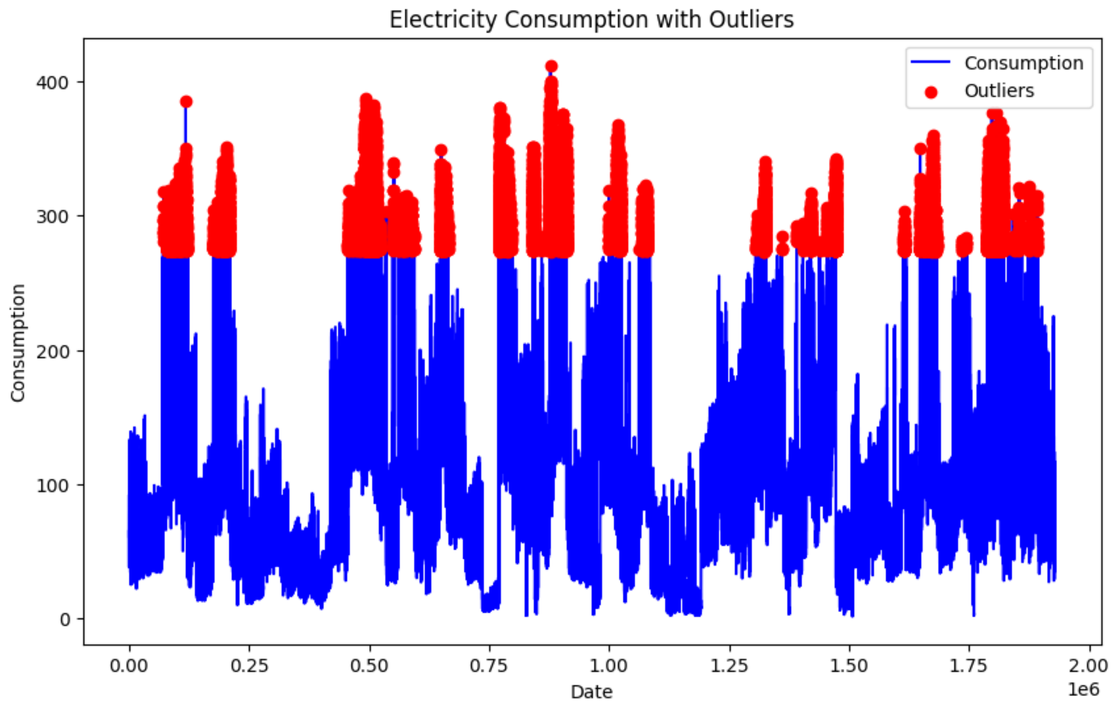


We chose to linear interpolate between the last known value and the next known. This is mainly because missing datapoints seems to be random. It seems logical to interpolate the data as it maps back to the real world, where seasonal changes vary, but dates within a short timeperiod should corralate more or less.

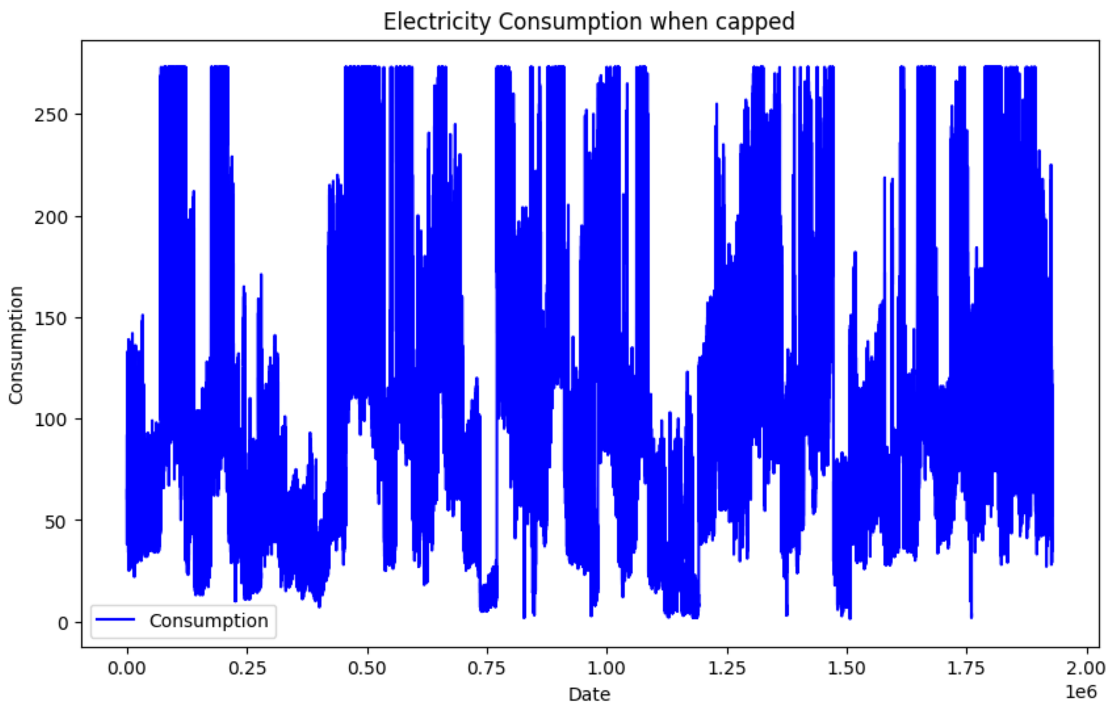
```
0          64.671363
1          58.000000
2          58.000000
3          58.000000
4          52.000000
             …
1928515    35.000000
1928516    35.000000
1928517    35.000000
1928518    35.000000
1928519    35.000000
Name: consumption, Length: 1928520, dtype: float64
```

3 Task 3: Handling outliers

Display IQR (Interquartile range), outliers shown by the red circles.



We chose to cap the data, due to the fact that the amount i rather large (a bit less than 70 000) and skewed. This will help reduce the impact of extreme values, but not remove them completely. Furthermore, our interest in this dataset is to make predictions and spot general trends. In this case outliers are not perticularly interesting, and we therefore find it natural to minimise their effect without completely removing them. To cap the dataset rather than transforming it also makes it easier to explain the data to a potential client.



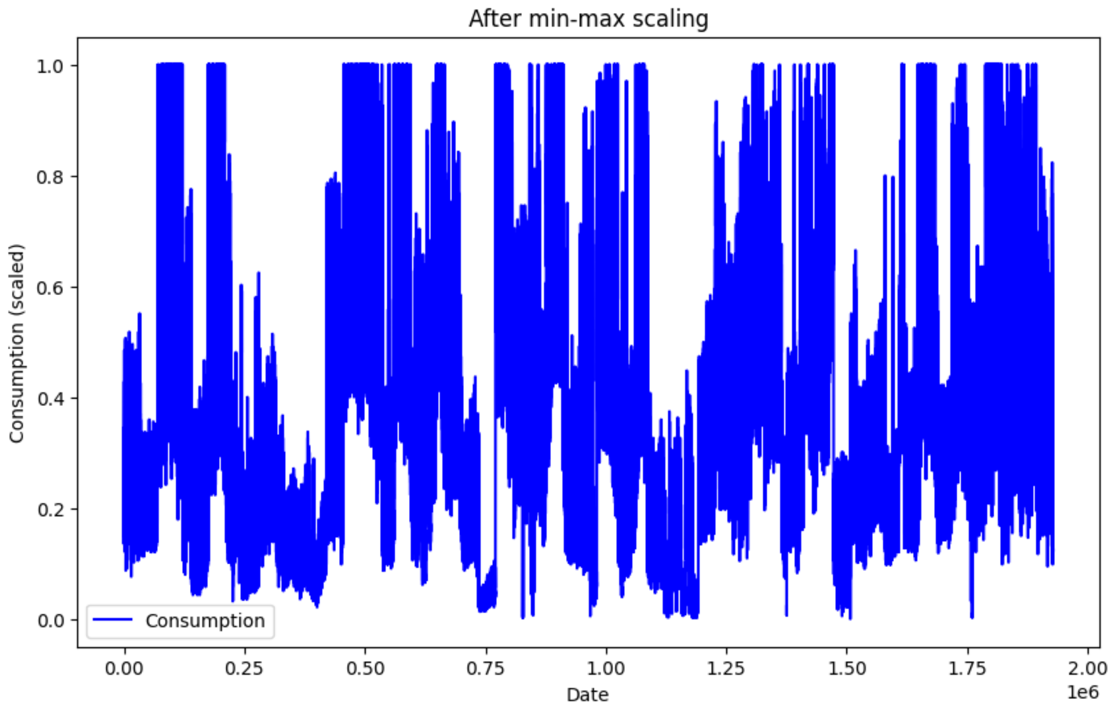
4 Task 4: Data Transformation

From the information we have available about this dataset, the substations are not ordinal and only unique identifiers for the specific substations. We have therefore chosen to apply one-shot encoding to them. Additionally, since the feeders seem like identifiers from information available to us, we've chosen to also apply one-hot encoding to them. This ensures theres no risk of the model assuming false relationships that might have accured if we used label encoding.

```
              datetime  consumption  substation_A  substation_B  substation_C  \
0  2017-01-01T00:00:00    64.671363             1             0             0
1  2017-01-01T01:00:00    58.000000             1             0             0
2  2017-01-01T02:00:00    58.000000             1             0             0
3  2017-01-01T03:00:00    58.000000             1             0             0
4  2017-01-01T04:00:00    52.000000             1             0             0

   substation_D  substation_E  substation_F  substation_G  substation_H  ...  \
0             0             0             0             0             0  ...
1             0             0             0             0             0  ...
2             0             0             0             0             0  ...
3             0             0             0             0             0  ...
4             0             0             0             0             0  ...

   feeder_M2  feeder_M3  feeder_M4  feeder_M5  feeder_M6  feeder_M7  \
0          0          0          0          0          0          0
```

```
1            0            0            0            0            0            0
2            0            0            0            0            0            0
3            0            0            0            0            0            0
4            0            0            0            0            0            0

   feeder_N1  feeder_N2  feeder_N3  feeder_N4
0          0          0          0          0
1          0          0          0          0
2          0          0          0          0
3          0          0          0          0
4          0          0          0          0

[5 rows x 71 columns]
```

Because the data is skewed, large, and the outliers are already getting capped, we've opted to use min-max feature scaling since its outlier sensitivity wasnt a concern anymore. Additionally we wanted to preserve the relationships between the different datapoints that min-max scaling allows us to do.



Feature scaling is necessary to use because it ensures faster convergence and prevents feature dominance. With scaled features, algorithms also perform better and lead to more accurate models.

5 Task 5: Data splitting

```
Training set size: (1542816, 3)
Testing set size: (385704, 3)
```

Splitting data into training and testing sets is important in training machine learning models. When we train a model, we want it to learn underlying patterns from the data that can be applied to new data, not just memorize the training data. This is where the concept of overfitting comes into play.

Overfitting occurs when a model becomes too complex, capturing not only the true patterns in the data but also noise or random fluctuations. This means the model performs very well on the training data but poorly on new data. Essentially, it memorizes the training data instead of generalizing well to other datasets. Overfitting leads to poor performance in real-world scenarios where the model is applied to new data.

By splitting the dataset into a training set and a testing set, we can reduce the risk of overfitting. The training set is used to train the model, learning from the data. The testing set, which the model has never seen before, acts as a test for real-world data. After training, the model is evaluated on the testing set, giving an estimate of how well it is likely to perform on new data.

This split makes us able to evaluate the model's generalization ability. If a model performs well on both the training and testing sets, it's likely capturing the true underlying patterns. If it performs well only on the training set but poorly on the testing set, overfitting is likely, and we may need to adjust the model by reducing its complexity, using regularization techniques, or collecting more data