Theory 1: Correlation between DA prices and gas peaker cost Analysis

Mathis Laurent

July 2024

1 Introduction

In this paper we will analyse the possible trends between DA prices, WD prices and gas peaker cost. From this we will see if there is a strong enough trend between the cost of those assets, Daily auction prices and we will see if we can predict this trend if the accuracy is high enough. The idea here is to eventually validate a theory (explained later) or if not idealise and understand a repetitive trend (That has been observed by EDF DA traders over the past few month). The idea here is to:

- Analyse the possible theory or trend
- Determine an automated algorithm that follows the fundamentals of the trend and takes into account Dynamic Data Sets
- Verify and test the algorithm on DA trends (using forcasted Gas Peaker cost)
- And finally as a conclusion, we will test the algorithm based on the original (theory) to a end goal to make sustainable profit.

Table of content

1.1 Definition and terms:

- DA prices: "Day Ahead prices" ⇒ DA prices are fixed through an auction process based on supply and demand bids.
- WD prices: Within Day prices ⇒ Within-day prices are a critical component of the energy market, allowing for real-time adjustments to supply and demand. They provide essential flexibility and risk management tools for market participants, ensuring that the energy system remains balanced and stable throughout the day. The volatility and immediate nature of these prices highlight the importance of real-time data and responsive strategies in energy trading.
- Gas Peaker cost: Gas Peaker Cost refers to the total cost incurred by a gas peaking power plant (gas peaker) to generate electricity. These costs include all expenses associated with running the plant, such as fuel costs, operational and maintenance costs, startup costs, and environmental compliance costs.
- EPEX: EPEX SPOT plays a crucial role in the European electricity market by providing transparent and efficient platforms for day-ahead and intra-day trading. Its operations enhance market integration, promote competition, and offer flexibility to market participants, ensuring a reliable and efficient electricity supply across Europe. ⇒ EPEX prices are fixed hourly.
- Nordpool: Nord Pool is a pivotal power market in Europe, offering platforms for day-ahead and intra-day trading. It enhances market integration, promotes competition, and ensures efficient price discovery and flexibility for market participants. Through its operations, Nord Pool supports the reliable and efficient functioning of the electricity markets in the regions it serves. ⇒ NordPool trades half hourly.
- Settlement period: Denoted as SPxx where xx is between 01 and 48 that represents every half hourly time periods a day. This is sufficiently closed to get accurate price volatility and flexibility within the energy market.
- IC: The information coefficient (IC) is a measure used to evaluate the skill an investment analyst or active portfolio manager. An IC of +1.0 indicates a perfect prediction of actual returns, while an IC of 0.0 indicates no linear relationship.
- IMRP: The Intermittent Market Reference Price is calculated using day-ahead data received from EPEX Spot and NordPool. An IMRP is calculated for every hour of the day.
- OTC: Over-the-counter (OTC) securities are securities that are not listed on a major exchange in the United States and are instead traded via a broker-dealer network.
- REMIT: REMIT is a pivotal legislative framework for energy markets within the European Union (EU). Submission awareness must be made when outage occurs.

1.2 Theory introduction and context:

EDF has been operating multiple assets for the last couple of months (roughly since the end of 2023 till now). For around 6 months of trading time, traders have observed interesting trends between DA prices, WD prices, and gas peaker strike prices. This results in the following theory: "Within day prices will trade below the Day Ahead prices (on a gas-adjusted basis) where the Day Ahead auction clears above gas peaker strike prices."
This can be summarised mathematically as follows:

If DA prices - Gas peaker cost > 0,

Then (WD prices - Gas Peaker Cost) < (DA prices - Gas Peaker Cost).

2 Market Analysis (theory validation, Data choices and Test):

2.1 Procedure:

- We started to select and close down a certain time period. Indeed, we took the longest data set (for accuracy) that could compile and had matching data sets; indeed, we used diverse arrays.

⇒ From the 16th of February till the 3rd of July (This is for the analysis; later on, the algorithm will use up-

INPUT Variables:

Time span	16/02/2024 at 00:00 till 16/02/2024 at 23:30 and from 20/02/2024 at 00:00 till 03/07/2024 at 21:30 (the 17th, 18th, and 19th of February are skipped due to no operation)
DA prices	We fixed every price in a half-hourly time zone and averaged out ”Nordpool” and ”APEX” prices for each settlement period a day
WD	We took every WD price at a half-hourly rate for the corresponding time period
Gas Peaker Cost	Selection and weighted average of gas peaker cost per time period

2.2 Asset selection for Gas Peaker cost:

Asset	Asset ID	Reason (if ignored)
BurtonHead	92706e24-d539-4dbd-9dcd-d5da2472e07c	Ignore this one as well because is an outlier and has a weighted average cost higher than the median
Caledon Green	1a9ee21f-dfb9-4e84-8ae4-1751af3c98ce	
Abercorn	1000000-1000-1000-1001-100000000002	
Medway A	1000000-1000-1000-1001-100000000003	Ignore cost of this asset due to the fact that it is very cheap
Medway B	1000000-1000-1000-1001-100000000004	Ignore cost of this asset due to the fact that it is very cheap
Erskine	1000000-1000-1000-1001-100000000007	Ignore this one as well because is an outlier and has a weighted average cost higher than the median
Carrington	1000000-1000-1000-1001-100000000008	

Now, we filtered our assets and we can average out the gas peaker cost of every asset per time period per day.

```
Sub CalculateAverageCostPerSettlementPeriod()
    Dim ws As Worksheet
    Set ws = ThisWorkbook.Sheets("Sheet1") ' Change to your sheet name

    Dim lastRow As Long
    lastRow = ws.Cells(ws.Rows.Count, "A").End(xlUp).Row

    Dim spRange As Range
    Set spRange = ws.Range("C2:C49") ' Change if your range is different

    Dim spCell As Range
    Dim sp As String
    Dim sum As Double
    Dim count As Long
    Dim cell As Range

    For Each spCell In spRange
        sp = spCell.Value
        sum = 0
        count = 0

        For Each cell In ws.Range("A2:A" & lastRow)
            If InStr(cell.Value, sp) > 0 Then
                sum = sum + cell.Offset(0, 1).Value
                count = count + 1
            End If
        Next cell

        If count > 0 Then
```

```
            spCell.Offset(0, 1).Value = sum / count
        Else
            spCell.Offset(0, 1).Value = "No Data"
        End If
    Next spCell
End Sub
```

From there we now have an average Gas Peaker Cost per settlement period per day. So there is one Gas Peaker Cost per DA and WD price.

2.3 Applying to the assimilated data the theory calculation:

So as the hypothesis implies: "If DA prices - Gas Peaker Cost ¿ 0 " this implies a form of profit. Using basic excel formula we subtract the whole Average DA price column to the gas peaker cost column. Using a VBA code, we would select and highlight the positive values and check the corresponding row for each one of them:

Here is the VBA code for highlighting positive values:

```
Sub HighlightPositiveValues()
    Dim ws As Worksheet
    Dim rng As Range
    Dim cell As Range

    ' Set the worksheet to the active sheet
    Set ws = ThisWorkbook.ActiveSheet

    ' Define the range (change "B" to the desired column and adjust the last
        row as necessary)
    Set rng = ws.Range("B2:B" & ws.Cells(ws.Rows.Count, "B").End(xlUp).Row)

    ' Loop through each cell in the range
    For Each cell In rng
        ' Check if the cell value is positive
        If IsNumeric(cell.Value) And cell.Value > 0 Then
            ' Highlight the cell with a color (e.g., yellow)
            cell.Interior.Color = RGB(255, 255, 0)
        End If
    Next cell
End Sub
```

2.4 Comparing

In the next part we are just going to compute the difference between the WD price and the gas peaker cost. Now we should get two column, the "DA spread" and the "WD spread". Now we would just right one VBA code that would just compare for each time the DA spread is positive if the the DA spread is as well greater than the WD spread. The best way to check the accuracy of this is to just compute the number of time we are "in the money" so when the value of the spread is positive. And then, compare to the number of times the DA spread is greater than the WD spread in that case. We can use the following VBA code:

Computing the number of time a number is positive in the DA spread:

Here is the VBA code for counting positive values:

```
Sub CountPositiveValues()
    Dim ws As Worksheet
    Dim rng As Range
    Dim cell As Range
    Dim positiveCount As Long

    ' Set the worksheet to the active sheet
    Set ws = ThisWorkbook.ActiveSheet

    ' Initialize the count of positive numbers
    positiveCount = 0

    ' Define the range (change "B" to the desired column and adjust the last
       row as necessary)
    Set rng = ws.Range("B2:B" & ws.Cells(ws.Rows.Count, "B").End(xlUp).Row)

    ' Loop through each cell in the range
    For Each cell In rng
        ' Check if the cell value is positive
        If IsNumeric(cell.Value) And cell.Value > 0 Then
            ' Increment the count of positive numbers
            positiveCount = positiveCount + 1
        End If
    Next cell

    ' Output the count of positive numbers
    MsgBox "The number of positive values in the column is: " & positiveCount
End Sub
```

For the given data set: we get 200 times a positive DA spread.

Now we are going to compute the number of times the DA spread is greater than the WD spread when the DA spread is greater than 0.

Here is the VBA code for counting positive values and checking if they are greater than corresponding values in another column:

```vba
Sub CountPositiveValuesGreaterThanCorresponding()
    Dim ws As Worksheet
    Dim rngA As Range
    Dim rngB As Range
    Dim cellA As Range
    Dim positiveCount As Long
    Dim greaterCount As Long

    ' Set the worksheet to the active sheet
    Set ws = ThisWorkbook.ActiveSheet

    ' Initialize the counts
    positiveCount = 0
    greaterCount = 0

    ' Define the ranges (change "A" and "B" to the desired columns)
    Set rngA = ws.Range("A2:A" & ws.Cells(ws.Rows.Count, "A").End(xlUp).Row)
    Set rngB = ws.Range("B2:B" & ws.Cells(ws.Rows.Count, "B").End(xlUp).Row)

    ' Ensure rngA and rngB have the same number of rows
    If rngA.Rows.Count <> rngB.Rows.Count Then
        MsgBox "The number of rows in Column A and Column B do not match."
        Exit Sub
    End If

    ' Loop through each cell in the range
    For Each cellA In rngA
        ' Check if the cell value in Column A is positive
        If IsNumeric(cellA.Value) And cellA.Value > 0 Then
            ' Increment the count of positive numbers
            positiveCount = positiveCount + 1

            ' Check if the corresponding value in Column B is less than the value in Column A
            If cellA.Value > cellA.Offset(0, 1).Value Then
                ' Increment the count of greater values
                greaterCount = greaterCount + 1
            End If
        End If
    Next cellA

    ' Output the counts
    MsgBox "The number of positive values in Column A is: " & positiveCount & vbCrLf & _
        "The number of times these positive values in Column A are greater than the corresponding values in Column B is: " & greaterCount
End Sub
```

For the given data set: we get around 125 times the DA spread is greater than the WD spread.

Conclusion: We can conclude that we get approximately 62.5% accuracy on this hypothesis. So we conclude that this isn't always the case and therefore despite the sources of errors, this cannot be a theory applicable in every case. Therefore, this type of accuracy is sufficient to say that there is a marked trend in the market.

2.5 Data representation and visualisation:

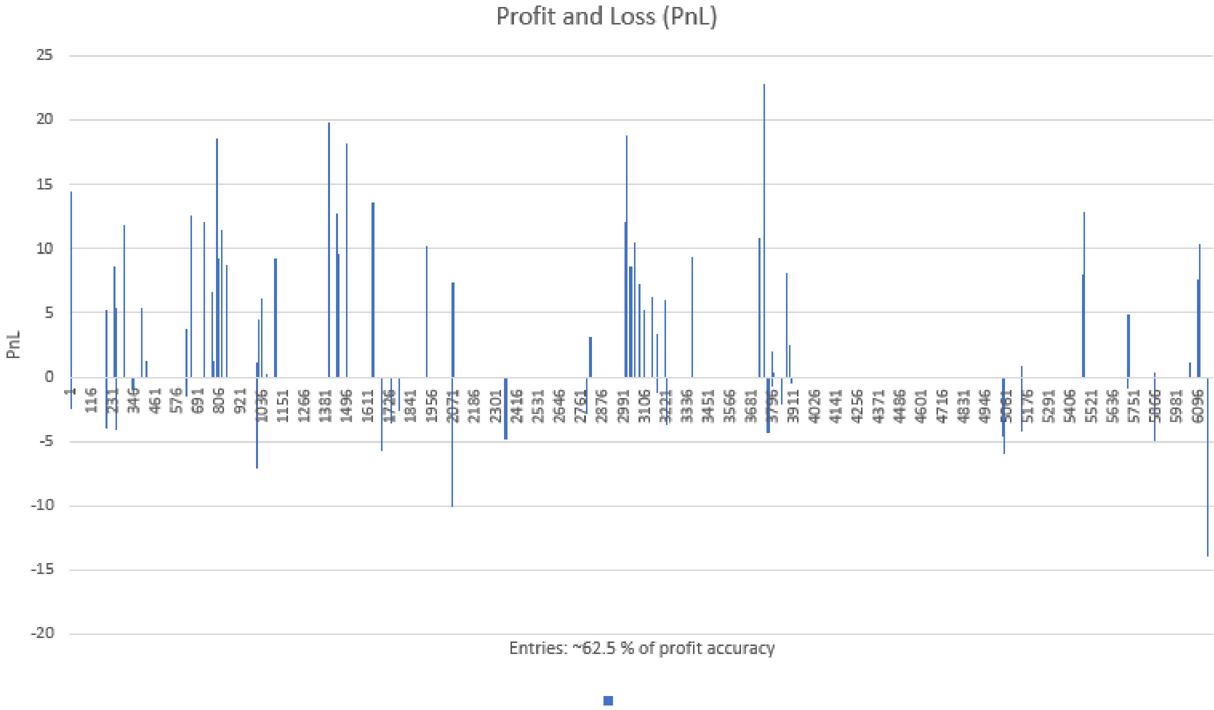

Figure 1: Overview of the amplitude of the profit and loss from the 16 of February till the 3rd of July. We can see here that if we compound an average across the year, there is quite a positive trend in profit.

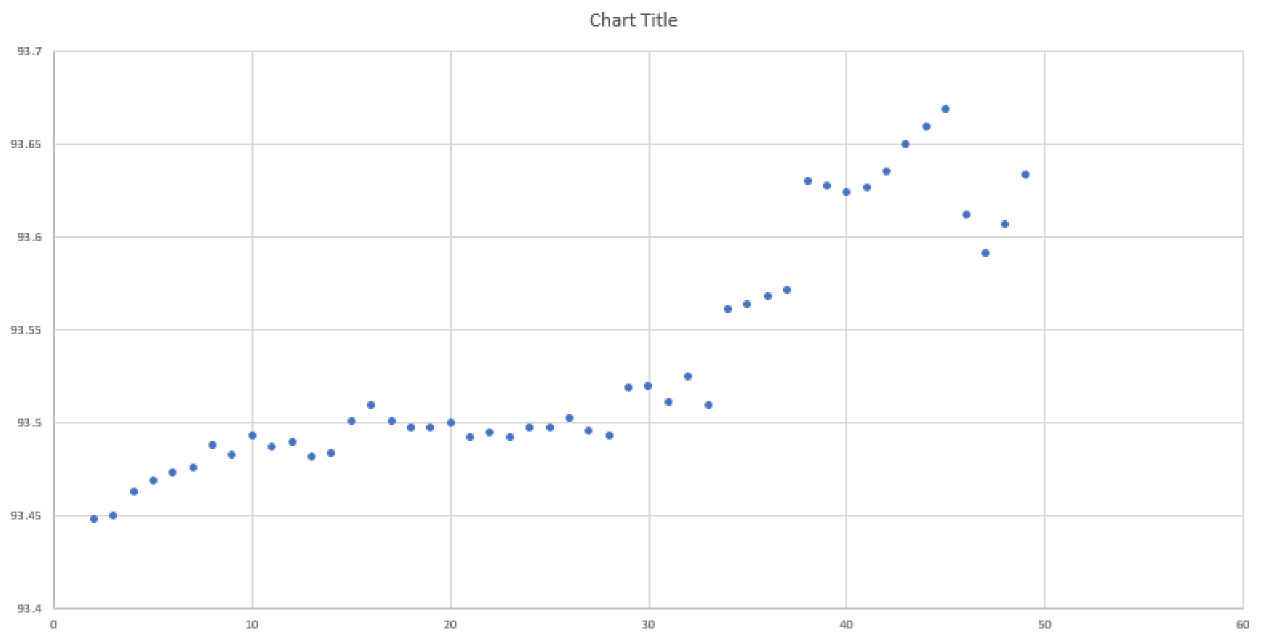

Figure 2: Average Gas Peaker Cost for each time period over the span of a day. We can see that the average Gas Peaker Cost across the day is increasing with a sudden jump around SP35 - SP36.

⇒ Now we are going to try and model a PnL calculator across a day depending on each time period.
We model an accumulated sum for each individual time period and an automated calculator for any size purchase.

<u>Excel model:</u>

SP	Profit	Hypothetical Po		MW	100
SP01	£0.00	£0.00			
SP02	£0.00	£0.00			
SP03	£0.00	£0.00			
SP04	£0.00	£0.00			
SP05	£0.00	£0.00			
SP06	£0.00	£0.00			
SP07	£0.00	£0.00			
SP08	£0.00	£0.00			
SP09	£0.00	£0.00			
SP10	£0.00	£0.00			
SP11	£0.00	£0.00			
SP12	£0.00	£0.00			
SP13	£27.72	£2,771.75			
SP14	£7.84	£784.25			
SP15	£48.53	£4,853.25			
SP16	£3.16	£316.25			
SP17	£0.53	£52.75			
SP18	£5.98	£597.75			
SP19	£0.00	£0.00			
SP20	£0.00	£0.00			
SP21	£0.00	£0.00			
SP22	£0.00	£0.00			
SP23	£0.00	£0.00			
SP24	£0.00	£0.00			
SP25	£0.00	£0.00			
SP26	£0.00	£0.00			
SP27	£0.00	£0.00			
SP28	£0.00	£0.00			
SP29	£0.00	£0.00			
SP30	£0.00	£0.00			
SP31	£0.00	£0.00			
SP32	£0.00	£0.00			
SP33	£0.00	£0.00			
SP34	£0.00	£0.00			
SP35	-£7.83	-£782.50			
SP36	-£26.93	-£2,692.50			
SP37	£54.95	£5,494.50			
SP38	£36.23	£3,623.00			
SP39	£47.21	£4,721.00			
SP40	£58.61	£5,861.00			
SP41	£25.18	£2,518.25			
SP42	£33.26	£3,326.25			
SP43	£7.82	£781.75			
SP44	£6.92	£691.75			
SP45	£0.00	£0.00			
SP46	£0.00	£0.00			
SP47	£0.00	£0.00			
SP48	£0.00	£0.00			
		£32,918.50			

Figure 3: Here is a brief overlook of the model for this specific data base. We can see that in majority, the data base acts in a parallel trend to the hypothesis. For a weighted average computation we can see that the profit act as 63 percent similarity. We can observe a sudden drop around SP36 which correlates to a sudden jump in figure 2. This examples shows that for a 100 MGWh purchase over the time periods, the profit would come at an average of 32.9 k £.

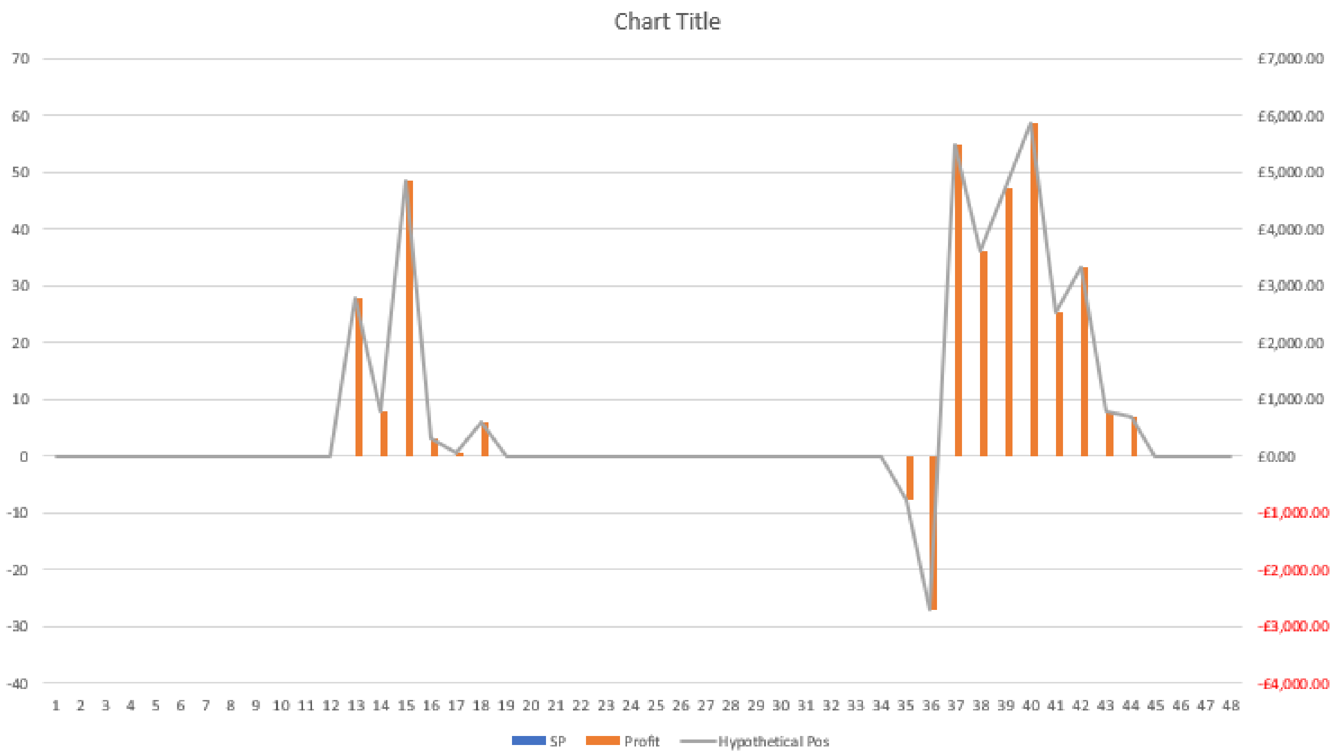

Figure 4: And here is a overlook of the PnL over the 48 periods. This graph could be quite useful to select the interval of trading hours, to predict Sell and/or Buy orders.

⇒ The periods of interest here are going to be the profit zones, which are: [SP12; SP19] and [SP37; SP45]

2.6 Optimizing Analysis: Individual analysis (EPEX and Nordpool)

In order to refine the analysis and eventually identify sources of errors we are going to individually perform the analysis in order to get a more consistant outcome.

2.6.1 Data Alignment:

- Align the hourly EPEX prices with the 30-minute Nord Pool prices.
- Ensure you have predictions for gas peaker costs to account for cost adjustments.

2.6.2 Identify Positive Trading Intervals:

Using similar VBA code we highlight and count the number of time positive values occurs in each one of the two column
⇒ For the EPEX values theres approximatly 182 values that are positive for that given data set. So we are in the money *182. For corresponding EPEX value, the theory validates 45 times. So 24.7 % accuracy which is quite low.
⇒ For the Nordpool prices, the difference is positive around 239 times. So we are in the money *239. For corresponding Nordpool value, the theory validates 68 times. So 28.7 % accuracy which is quite low as well.
Now, in the following individual analysis, after computing the following column, we developped

a VBA code that identifies thye following relation:

$$\text{IF } G > 0,\ H > 0 \text{ AND } G > F,\ H > F \Rightarrow PROFIT$$

Here is the VBA code for marking profit:

```
Sub MarkProfit()
    Dim ws As Worksheet
    Dim lastRow As Long
    Dim i As Long

    ' Set the worksheet to the active sheet
    Set ws = ThisWorkbook.ActiveSheet

    ' Find the last used row in Column G
    lastRow = ws.Cells(ws.Rows.Count, "G").End(xlUp).Row

    ' Loop through each cell in Column G from row 2 to the last used row
    For i = 2 To lastRow
        On Error Resume Next
        ' Check if the value in Column G is numeric and greater than 0
        If IsNumeric(ws.Cells(i, "G").Value) And CDbl(ws.Cells(i, "G").Value) > 0 Then
            ' Check if the value in Column G is greater than the value in Column F
            ' and if the value in Column H is greater than the value in Column F
            If CDbl(ws.Cells(i, "G").Value) > CDbl(ws.Cells(i, "F").Value) And CDbl(ws.Cells(i, "H").Value) > CDbl(ws.Cells(i, "F").Value) Then
                ' Write "PROFIT" in Column I
                ws.Cells(i, "I").Value = "PROFIT"
            Else
                ' Write nothing in Column I
                ws.Cells(i, "I").Value = ""
            End If
        Else
            ' Write nothing in Column I
            ws.Cells(i, "I").Value = ""
        End If
        On Error GoTo 0
    Next i

    ' Notify the user that the process is complete
    MsgBox "Profit marking complete.", vbInformation
End Sub
```

Now we want to extract the settlement period in column A correspond9ing to a profit. The idea here is to average and compound out where does there profits appears over the course of the day.

Here is the VBA code for extracting settlement periods:

```
Sub ExtractSettlementPeriod()
    Dim ws As Worksheet
    Dim lastRow As Long
    Dim i As Long
    Dim cellValue As String
    Dim settlementPeriod As String

    ' Set the worksheet to the active sheet
    Set ws = ThisWorkbook.ActiveSheet

    ' Find the last used row in Column I
    lastRow = ws.Cells(ws.Rows.Count, "I").End(xlUp).Row

    ' Loop through each cell in Column I from row 2 to the last used row
    For i = 2 To lastRow
        ' Check if the cell value in Column I is "PROFIT"
        If ws.Cells(i, "I").Value = "PROFIT" Then
            ' Extract the settlement period from the corresponding cell in
              Column A
            cellValue = ws.Cells(i, "A").Value
            settlementPeriod = ExtractSP(cellValue)

            ' Write the settlement period in Column J
            ws.Cells(i , "J").Value = settlementPeriod
        End If
    Next i

    ' Notify the user that the process is complete
    MsgBox "Settlement periods extraction complete.", vbInformation
End Sub

Function ExtractSP(cellValue As String) As String
    Dim parts() As String
    parts = Split(cellValue, "-")
    ExtractSP = Trim(parts(UBound(parts)))
End Function
```

To asses the eventual paterns there are in profit per settlement period we create a Graph taking the settlement periods in the x axis and the amplitude of number of profit per setlement periods in the y axis. We get:

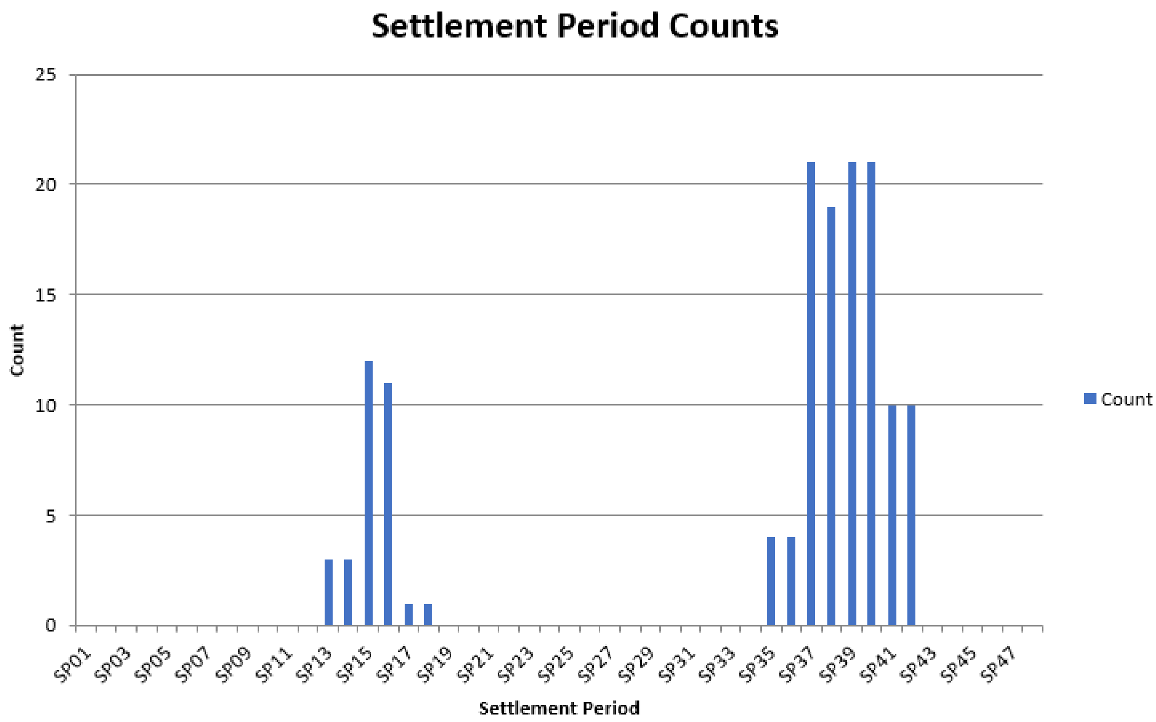

Figure 5: We can see from this graph for individual combined profit gains for both two Auctions, we get a similar pattern to the previous graph we obtained using the DA average of both trading auctions. We conclude that the hottest periods of trades for maximum profit are settled on the intervalls: [SP15;SP17] and [SP37;SP41]. From this data we can average out the settlement periods where selling or buying is the most optimal.

3 Applying the precedent analysis to an automated strategy:

Now that the primary analysis is done and we have demonstrated a strong enough correlation between the forecatsed DA prices and Gas Peaker cost, we will now using a dynamic Data Set create a form of algorithm that will procceed by performing the preceeding computations based on up to date Data.

Inputs	Outputs	Specifications
Forecasted DA Prices and Gas Peaker Cost, the previously tested data	A superposition of the profit graph relative to the settlement periods. We will in the first place get a graphical representation to visualize if the trends are corresponding.	The forecasted data will come as a dynamic data set (CSV) and the previously tested data will come as the previous raw data
	Time periods of Sell orders within the profit intervals (of the previous analysis)	This second output will just be another form of execution, input data and code would not change.

In this section we will try to establish the best strategy to evaluate real time updated data and the best trading periods. W4e will optimise our algorithm using the previously used hirstorical data, automated data sets as well as numerous Python function. In the following order:

- We will start by importing the needed libraries
- Then we will connect the automated SQL data sets to python on a CSV file.
- We will also import the data from the previous analysis in order to do complete a comparason.
- Then we will merge the files together to get the price and cost values pefectly correlated to the settlement periods.
- Then we will apply this data to the hypothesis principle.
- As a first output we will try to plot the coverage of live prices so that we get an overview of the price variation and make a direct comparison to the previous data analysis. After plotting this we will see if there is a continuous correlation.
- If there is a strong enough correlation, we will analyse the profit intervals (positive area under the curve) and will execute sell orders if condition is met.
- In addition to the live plot coverage we will get as a final output the settlement period where sell orders should be executed.

In this section we are going to break down a python algorithm that will automate short orders when the condition is satisfied. The ideal situation would be that the next day, the short order is buyed back in the goal to make profit.

Steps:

1 Uploading Data Sets to Cloud/Desktop ⇒ List of Data sets: "Colab-GPC", "Colab-PPF", "Colab-DA-WD"

Inputs	Specifications and details
Colab-PPF	Forecasted Data set since 20/02/2024 at 01:00 of Power Prices
Colab-GPC	Forecasted Data set since 20/02/2024 at 01:00 of the gas peaker cost per asset. The following algorithm will average out each gas peaker cost of each asset per settlement periods
Colab-DA-WD	Fixed Day-Ahead and Within-Day prices per settlement period. The within day is given by the SWAP prices per settlement period and the DA (EPEX auction price) is given hourly and must be converted later into the settlement period basis.

2 Translating in a half-hourly basis: In Excel, we will be using VBA code to convert the EPEX auction prices half-hourly into settlement periods.

Here is the VBA code for converting hourly values to half-hourly values:

```
Sub ConvertHourlyToHalfHourly()
    Dim ws As Worksheet
    Dim lastRow As Long
    Dim i As Long
    Dim currentValue As Variant
    Dim outputRow As Long

    ' Set the worksheet to the active sheet
    Set ws = ThisWorkbook.ActiveSheet

    ' Find the last used row in Column C
    lastRow = ws.Cells(ws.Rows.Count, "C").End(xlUp).Row

    ' Start output in Column D
    outputRow = 2

    ' Loop through each cell in Column C from row 2 to the last used row
    For i = 2 To lastRow
        currentValue = ws.Cells(i, "C").Value

        ' Write the current value twice in Column D
        ws.Cells(outputRow, "D").Value = currentValue
        ws.Cells(outputRow + 1, "D").Value = currentValue

        ' Move to the next pair of rows in Column D
        outputRow = outputRow + 2
    Next i

    ' Notify the user that the process is complete
    MsgBox "Conversion to half-hourly values complete.", vbInformation
End Sub
```

(3) Running the algorithm: The data sets that we know have uploaded and cleaned will know be processed by the developed python algorithm. This will allow us to process dynamical and large data sets quicker with more precised outputs. Additionally, we will continue developing the code for it to simulate and forecast virtual trades. From this if the active PnL is good enough we will implement such a strategy and could be even implemented into algorithmic trading later on.

Code and results:

(1) **Libraries:**

```
import pandas as pd
import matplotlib.pyplot as plt
```

(2) **Function to Convert Excel Serial Dates to Datetime:**

```
def excel_serial_to_datetime(serial):
    # Convert Excel serial date to datetime
    return pd.Timedelta(days=serial) + pd.Timestamp('1899-12-30')
```

(3) **Data Processing and Digestion:**

```python
def read_and_process_data(ppf_file_path, gpc_file_path):
    start_date = pd.Timestamp('2024-02-20 01:00:00')

    # Read PPF forecasted prices from Colab-PPF.xlsx
    ppf_prices = pd.read_excel(ppf_file_path, sheet_name='Sheet1', usecols=['
        Settlement_DateTime', 'Price'], parse_dates=['Settlement_DateTime'])
    ppf_prices.columns = ['Settlement Period', 'PPF_Price']

    # Filter PPF prices to start from the specific start date
    ppf_prices = ppf_prices[ppf_prices['Settlement Period'] >= start_date]

    # Read forecasted gas peaker costs from Colab-GPC.xlsx
    gpc_costs = pd.read_excel(gpc_file_path, sheet_name='Sheet1', usecols=['
        CostDate', 'Cost'], parse_dates=['CostDate'])
    gpc_costs.columns = ['Settlement Period', 'Forecasted_Cost']

    # Convert Excel serial date to datetime if needed
    if gpc_costs['Settlement Period'].dtype == 'float64':
        gpc_costs['Settlement Period'] = gpc_costs['Settlement Period'].apply(
            excel_serial_to_datetime)

    # Filter gas peaker costs to start from the specific start date
    gpc_costs = gpc_costs[gpc_costs['Settlement Period'] >= start_date]

    # Compute the average gas peaker cost for each settlement period
    avg_gpc_costs = gpc_costs.groupby('Settlement Period')['Forecasted_Cost'].
        mean().reset_index()

    # Ensure all Settlement Period columns are datetime
    ppf_prices['Settlement Period'] = pd.to_datetime(ppf_prices['Settlement
        Period'])
    avg_gpc_costs['Settlement Period'] = pd.to_datetime(avg_gpc_costs['
        Settlement Period'])

    # Debugging: Print head of all dataframes before merging
    print("Filtered PPF Prices:")
    print(ppf_prices.head(10))
    print("\nAverage Gas Peaker Costs:")
    print(avg_gpc_costs.head(10))

    # Ensure data alignment by matching on the settlement period
    data = pd.merge(avg_gpc_costs, ppf_prices, on='Settlement Period', how='
        inner')

    # Debugging: Print head of merged dataframe
    print("\nMerged Data:")
    print(data.head(10))

    # Calculate the condition for being in the money
    data['PPF_minus_GPC'] = data['PPF_Price'] - data['Forecasted_Cost']
    data['Is_In_The_Money'] = data['PPF_minus_GPC'] > 0

    # Calculate cumulative in-the-money periods for each day
    data['Date'] = data['Settlement Period'].dt.date
    daily_in_the_money = data.groupby('Date')['Is_In_The_Money'].sum().
        reset_index()

    # Debugging: Print daily in-the-money periods
    print("\nDaily In-The-Money Periods:")
    print(daily_in_the_money.head(10))

    return daily_in_the_money
```

Here is the Python function for reading and processing DA-WD data:

```python
def read_and_process_da_wd_data(da_wd_file_path):
    # Read DA-WD data from Colab-DA-WD.xlsx
    da_wd_data = pd.read_excel(da_wd_file_path, sheet_name='Sheet1', usecols=['DATETIME_UTC', 'WD Price', 'DA Price'], parse_dates=['DATETIME_UTC'])
    da_wd_data.columns = ['Date', 'WD_Price', 'DA_Price']

    # Calculate the PnL
    da_wd_data['PnL'] = (da_wd_data['DA_Price'] - da_wd_data['WD_Price']) * 100

    # Accumulate daily PnL
    da_wd_data['Date'] = da_wd_data['Date'].dt.date
    daily_pnl = da_wd_data.groupby('Date')['PnL'].sum().reset_index()

    return daily_pnl
```

④ Plotting Function for Daily In-The-Money Periods:

```python
def plot_daily_in_the_money(daily_in_the_money):
    plt.figure(figsize=(10, 6))
    plt.plot(daily_in_the_money['Date'], daily_in_the_money['Is_In_The_Money'], marker='o')
    plt.title('Daily In-The-Money Periods')
    plt.xlabel('Date')
    plt.ylabel('Number of In-The-Money Periods')
    plt.grid(True)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

⑤ Plotting DA - WD:

```python
def plot_daily_pnl(daily_pnl):
    plt.figure(figsize=(10, 6))
    plt.plot(daily_pnl['Date'], daily_pnl['PnL'], marker='o', linestyle='-', color='blue', label='PnL')
    plt.title('Daily PnL from DA-WD Prices')
    plt.xlabel('Date')
    plt.ylabel('PnL (  )')
    plt.grid(True)
    plt.legend()
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

5 Statistics:

```python
def compute_statistics(daily_pnl):
    highest_profit = daily_pnl['PnL'].max()
    lowest_profit = daily_pnl[daily_pnl['PnL'] > 0]['PnL'].min()
    highest_loss = daily_pnl['PnL'].min()
    lowest_loss = daily_pnl[daily_pnl['PnL'] < 0]['PnL'].max()
    average_profit = daily_pnl[daily_pnl['PnL'] > 0]['PnL'].mean()
    average_loss = daily_pnl[daily_pnl['PnL'] < 0]['PnL'].mean()
    cumulative_profit_loss = daily_pnl['PnL'].sum()

    stats = {
        "Highest Profit": highest_profit,
        "Lowest Profit": lowest_profit,
        "Highest Loss": highest_loss,
        "Lowest Loss": lowest_loss,
        "Average Profit": average_profit,
        "Average Loss": average_loss,
        "Cumulative Profit + Loss": cumulative_profit_loss
    }

    return stats
```

6 Main function:

```python
# File paths to your Excel files in the current working directory
ppf_file_path = 'Colab-PPF.xlsx'
gpc_file_path = 'Colab-GPC2.xlsx'

# Read initial data
daily_in_the_money = read_and_process_data(ppf_file_path, gpc_file_path)

# Print data for verification
print(daily_in_the_money.head())

# Plot the results
plot_daily_in_the_money(daily_in_the_money)
```

```
                      :

0  2024-02-20 01:00:00        50.108
1  2024-02-20 01:30:00        49.476
2  2024-02-20 02:00:00        49.116
3  2024-02-20 02:30:00        48.564
4  2024-02-20 03:00:00        48.372
5  2024-02-20 03:30:00        48.212
6  2024-02-20 04:00:00        48.244
7  2024-02-20 04:30:00        49.300
8  2024-02-20 05:00:00        50.324
9  2024-02-20 05:30:00        52.988

                           :

0  2024-02-20 02:00:00        70.780
1  2024-02-20 02:30:00        70.780
2  2024-02-20 03:00:00        70.780
3  2024-02-20 03:30:00        71.625
4  2024-02-20 04:00:00        71.625
5  2024-02-20 04:30:00        71.625
6  2024-02-20 05:00:00        72.040
7  2024-02-20 05:30:00        72.180
8  2024-02-20 06:00:00        72.250
9  2024-02-20 06:30:00        72.292

             :

0  2024-02-20 02:00:00        70.780            49.116000
```

```
1   2024-02-20 02:30:00         70.780            48.564000
2   2024-02-20 03:00:00         70.780            48.372000
3   2024-02-20 03:30:00         71.625            48.212000
4   2024-02-20 04:00:00         71.625            48.244000
5   2024-02-20 04:30:00         71.625            49.300000
6   2024-02-20 05:00:00         72.040            50.324000
7   2024-02-20 05:30:00         72.180            52.988000
8   2024-02-20 06:00:00         72.250            55.513167
9   2024-02-20 06:30:00         72.292            58.576000

        -   -               :

0   2024-02-20                 0
1   2024-02-21                 0
2   2024-02-22                 0
3   2024-02-23                 5
4   2024-02-24                 4
5   2024-02-25                 0
6   2024-02-26                 0
7   2024-02-27                 4
8   2024-02-28                 2
9   2024-02-29                 4


0   2024-02-20                 0
1   2024-02-21                 0
2   2024-02-22                 0
3   2024-02-23                 5
4   2024-02-24                 4
```

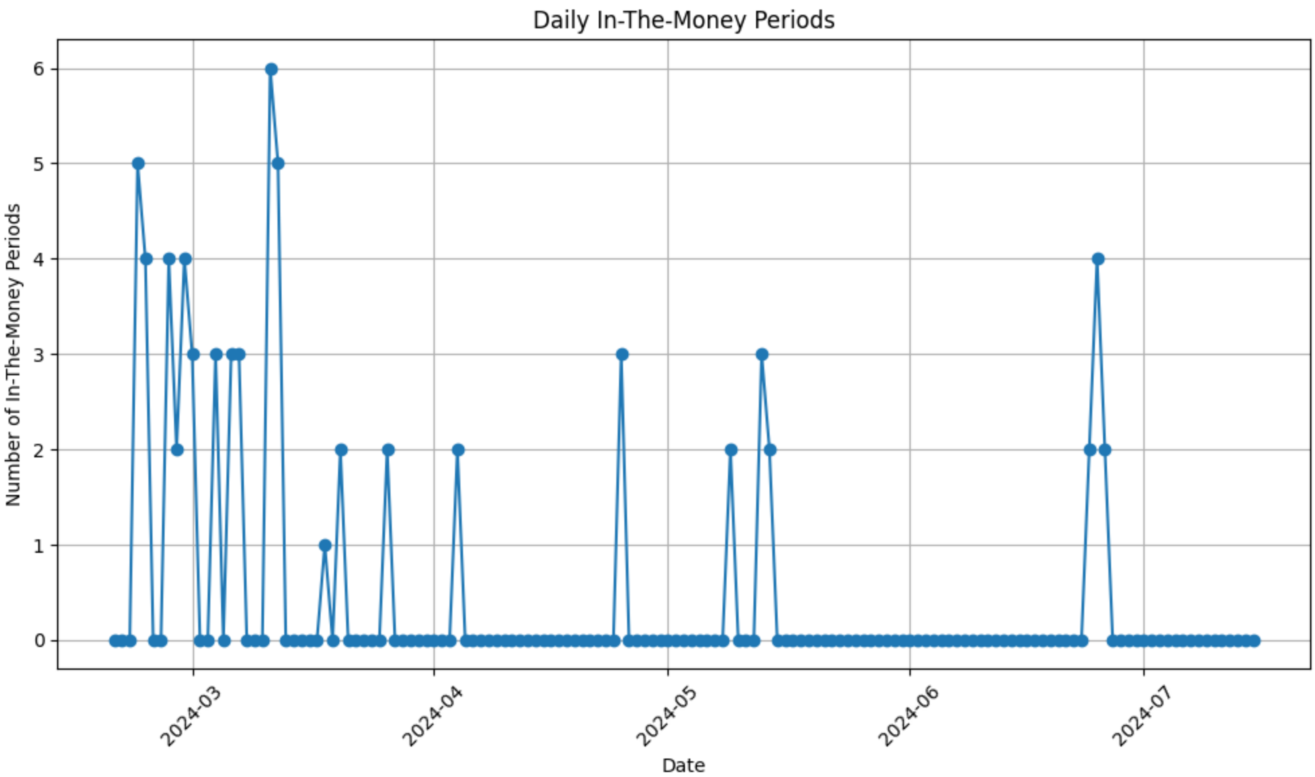

Figure 6:

⑦ PnL Code and Output:

```
# File paths to your Excel files in the current working directory
ppf_file_path = 'Colab-PPF.xlsx'
gpc_file_path = 'Colab-GPC2.xlsx'
da_wd_file_path = 'Colab-DA-WD.xlsx'

# Read and process DA-WD data
daily_pnl = read_and_process_da_wd_data(da_wd_file_path)

# Read and process PPF and GPC data if needed
# daily_in_the_money = read_and_process_data(ppf_file_path, gpc_file_path)

# Print data for verification
print(daily_pnl.head())

# Plot the results
plot_daily_pnl(daily_pnl)

# Compute statistics
stats = compute_statistics(daily_pnl)
print("\nStatistics:")
for key, value in stats.items():
    print(f"{key}: {value}")
```

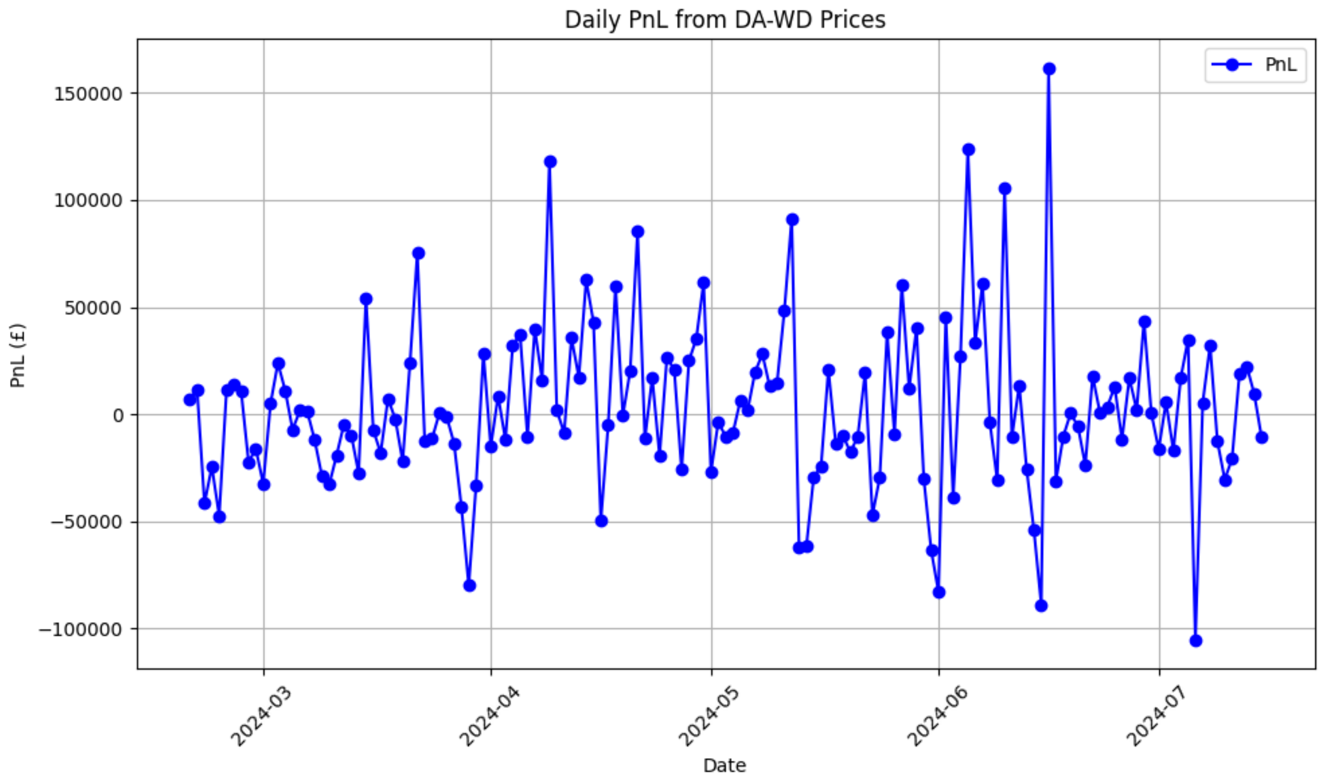

Figure 7:

⑧ **Statistics:**

```
Highest Profit: 161802.60556637225
Lowest Profit: 388.8011309983767
Highest Loss: -105425.57674334443
Lowest Loss: -665.480412686896
Average Profit: 30380.101399969495
Average Loss: -25293.683678045665
Cumulative Profit + Loss: 457362.38017842465
```

4 Executing a Model

In this part, after analysing the data and building historical diagrams to follow the possible hypothesis trend. We conclude to a strong enough correlation between GPC and DA prices. This correlation will be tested on the trading desk following a built model taking into account a automated data set that updates half hourly. We gain forcasted data up to 24 hours in advance which will give us an indication to trade or not the current market price a day before. In the positive outcome and of course if the market permits it, we will follow indication of the created model and perform a trade. We will deduce from this a PnL. Of course we will have to perform this type of trade for an extended period of time to implement it within the DA traders on a daily basis.

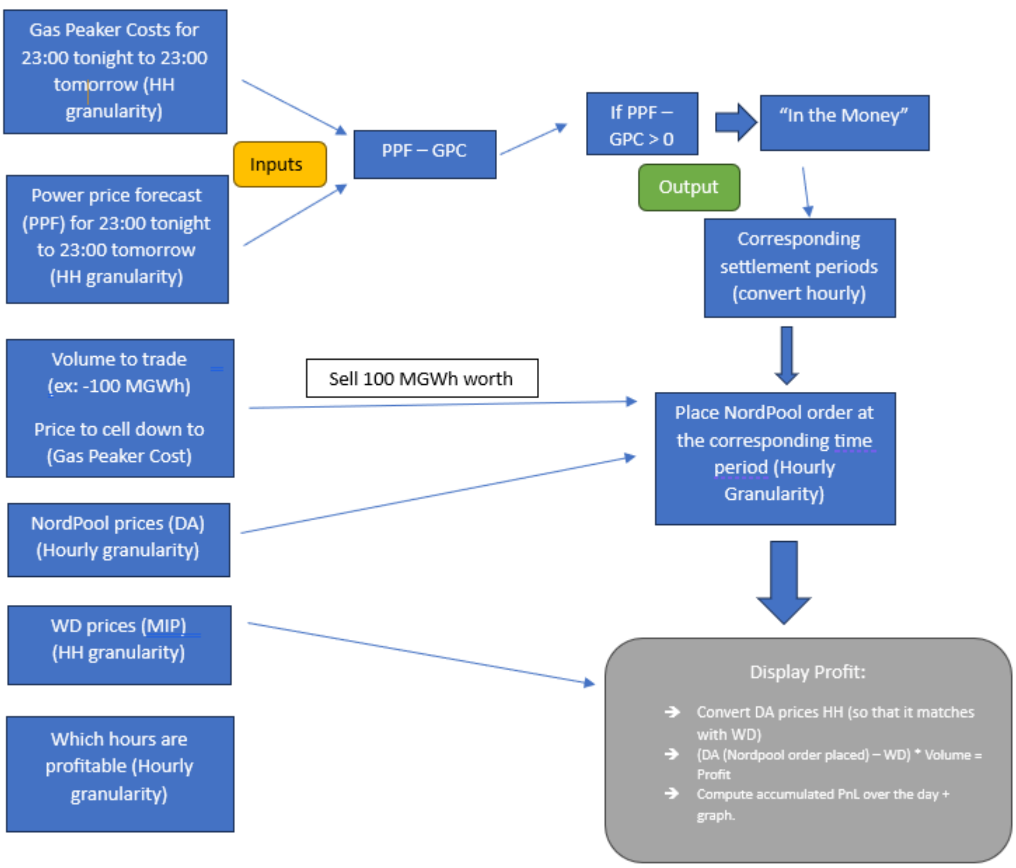

Figure 8: Overview of model