Search Medium Write

Energy Supply and Demand Optimisation: Mathematical Modelling Using Gurobi Python

Efficient energy allocation through mathematical optimisation

Kong You Liow · Follow
Published in Towards Data Science · 11 min read · 5 days ago

135 1

Photo of power pylons at sunset. Source: image by Matthew Henry from Unsplash.

We are currently in an era where the equilibrium between energy supply and demand has become particularly important. Traditional energy sources like oil and gas are facing difficult challenges like geopolitical factors such as the Russian-Ukrainian conflict, which has disrupted natural gas supplies, and the overall depletion of natural reserves.

On the other hand, there is also a growing demand for energy, particularly as more countries transition from developing to developed status. This economic growth signifies an increased need for energy to fuel production and industrial processes. The urgency of addressing climate change further emphasises the need for renewable energy sources. Combining all factors, this means that achieving a well-balanced energy supply and demand has only become increasingly crucial.

Mathematical optimisation is the study of selecting criteria to minimise or maximise a specific objective within predefined constraints, all managed through mathematical models. It is useful in solving a variety of real-world problems, each with its own unique complexities.

For instance, in logistics and transportation, mathematical optimisation can be used to solve challenges like the vehicle routing problem, which optimises routes for fleets, therfore reducing costs and environmental impact by using efficient routes. In computational biology, this approach is instrumental in understanding complex biological processes, improving drug discovery, genomics, and the study of disease pathways by developing precise models. Additionally, mathematical optimisation plays a key role in supply chain, a crucial element in modern commerce and industry. It optimises resource allocation, ensuring efficient production and distribution

systems that minimize waste and costs while enhancing customer satisfaction. In summary, mathematical optimisation empowers decision makers in various fields to navigate complex problems to achieve optimal solutions for their problems.

In this article, we show how to use mathematical optimisation to solve a basic energy supply and demand problem. We will start by explaining the problem statement and outlining the mathematical formulation. Then, we will take a practical example and use the Gurobi solver in Python to run the model. Finally, we will analyse the results obtained from this simulation.

1. Problem statement

Let us envision a hypothetical situation where we are responsible for formulating the yearly budget for energy production and storage in a fictitious country. In this country, there are only four energy sources, namely solar, wind, natural gas, and nuclear.

Artist illustrations of the different energy sources in our problem: solar, wind, natural gas, and nuclear. Sources: icons by small.smiles, wanicon, Freepik from Flaticon.

We have access to data detailing their monthly production and storage capacities, and also insights into the associated production and storage costs. Additionally, we assume to have the understanding of the energy demand throughout the year. The key question is: how can we optimise the monthly production and storage quantities for each energy source to minimise the total annual cost?

2. Mathematical formulation

Now, let us delve into the mathematical formulation of our model. Feel free to skip this section if you are not a fan of equations. However, for those who find mathematics intriguing, the insights gained will be well worth the journey. We will outline the parameters, objective functions, and constraints in precise mathematical terms.

2.1. Parameters

Let us define the mathematical notations used in our model. We denote $e \in E$ as the type of energy source in the set of all energy sources E (solar, wind, natural gas, and nuclear), and $t \in T$ as the month in the set of all the months T (January, February, March, and so on).

Now, we introduce the following parameters given for our problem:

- D_t is the total projected energy demand for month t.
- P_{te} is the production upper limit of energy source e in month t.
- S_{te} *is* the storage upper limit of energy source e in month t.
- A_e is the production cost of energy source e .
- B_e is the storage cost of energy source e.

The variables of our model are defined as follow:

- x_{te} is the amount of energy source e produced in month t.
- y_{te} is the amount of energy source e remained in month t, which is also the amount that will be stored during that month.
- z_{te} is the amount of energy source e used in month t to satisfy the energy demand for that month.

2.2. Objective function

Now, let us define our objective function. Our aim is to minimise the total cost for the entire year. This can be expressed in mathematical terms as follows:

$$\min \sum_{e \in E} \sum_{t \in T} \left(A_e x_{te} + B_e y_{te} \right)$$

Here, $A_e x_{te}$ represents the production cost of energy e in month t, while $B_e y_{te}$ denotes the storage cost of energy e in month t. The two summation signs Σ signify the summation of these production and storage costs over all energy sources throughout the year. The goal is to find the values for the variables x_{te} and y_{te} that minimise the total cost.

2.3. Constraints

Constraints are a vital part of our problem-solving approach. They ensure the solutions we find are both realistic and feasible, given our parameters.

2.3.1. Mass balance constraint

The first constraint, also known as the mass balance constraint, ensures that for each energy source in a given month, the sum of the amount produced and the amount carried over from the previous month equals the sum of the amount used and the amount carried over to the next month. We assume that in the first month t_0 (January), there is no energy in storage yet. Mathematically, this constraint can be expressed as:

$$x_{te} + y_{t-1,e} = z_{te} + y_{te} \quad \forall t \in T/t_0, \forall e \in E$$

$$x_{t_0 e} = z_{t_0 e} + y_{t_0 e} \quad \forall e \in E$$

2.3.2. Demand constraint

Next, we have the demand constraint, which is a fundamental aspect of our problem. It ensures that in any given month, the combined energy produced and carried forward from the previous month must be sufficient to meet the energy demand for that month. Mathematically, this constraint can be expressed as:

$$\sum_{e \in E} \left(x_{te} + y_{t-1,e} \right) \geq D_t \quad \forall t \in T/t_0$$

$$\sum_{e \in E} x_{t_0 e} \geq D_{t_0}$$

Here, Σ_e represents the sum over all energy source e. The constraint ensures that the total energy available (the sum of production and storage across all energy sources) in a given month t is greater than or equal to the energy demand D_t for that month. Given that there is no energy storage in the first month t_0, we simply ensure that the total energy produced in the first month is sufficient to meet the demand for that month.

2.3.3. Variable domains constraints

The final set of constraints involves defining the domains of our variables, which are crucial for ensuring that the values of these variables remain within meaningful and feasible bounds. In our case, we have production limits and storage limits that restrict the values of x_{te} and y_{te}, respectively. Additionally, given that these variables should not take negative values, we need to set lower limits on their domains. Mathematically, the domain constraints can be expressed as:

$$0 \leq x_{te} \leq P_{te} \quad \forall t \in T, \forall e \in E$$

$$0 \leq y_{te} \leq S_{te} \quad \forall t \in T, \forall e \in E$$

$$z_{te} \geq 0 \quad \forall t \in T, \forall e \in E$$

There is no need to explicitly define the upper limit for z_{te} because it is inherently determined by the mass balance constraint.

3. Code

Here, we will provide a brief description of the code implementation. We start by importing the Gurobi library and a custom `Data` class responsible for loading data. The code can be found on GitHub here: energy-optimisation.

The first step is to define the model and the necessary variables.

```
import gurobipy
from gurobipy import GRB

from data_extraction import Data

# Model
model = gurobipy.Model("Energy optimisation")

# Variables
X = model.addVars(
    Data.months,
    Data.energies,
    name="Amount supplied",
    ub=Data.production_limit,
)
```

```
Y = model.addVars(Data.months, Data.energies, name="Amount remained")
Z = model.addVars(Data.months, Data.energies, name="Amount used")
```

We have the option to set the upper limit of a variable using either the `ub` argument in the `model.addVars()` method, or by specifying constraints using the `model.addConstrs()` method. Here, we have illustrate how to set the upper limit for energy production `X` within the `model.addVars()` method. Later on, we will demonstrate how to establish the upper limit for energy storage `Y` using the `model.addConstrs()` method.

Next, we define the objective function.

```
# Objective function
obj = gurobipy.quicksum(
    (Data.production_cost[energy] * X[month, energy])
    + (Data.storage_cost[energy] * Y[month, energy])
    for energy in Data.energies
    for month in Data.months
)
model.setObjective(obj, GRB.MINIMIZE)
```

Lastly, we define the constraints, with the first one is the mass balance constraint.

```
# Mass balance constraints
model.addConstrs(
    (
        Y[Data.months[m_ind - 1], energy] + X[month, energy]
        == Z[month, energy] + Y[month, energy]
        for energy in Data.energies
        for m_ind, month in enumerate(Data.months)
        if month != Data.months[0]
```

```
        ),
        name="Energy balance",
    )
    model.addConstrs(
        (
            X[Data.months[0], energy]
            == Z[Data.months[0], energy] + Y[Data.months[0], energy]
            for energy in Data.energies
        ),
        name="Energy balance",
    )
```

After defining the mass balance constraint, we proceed to establish the demand constraint.

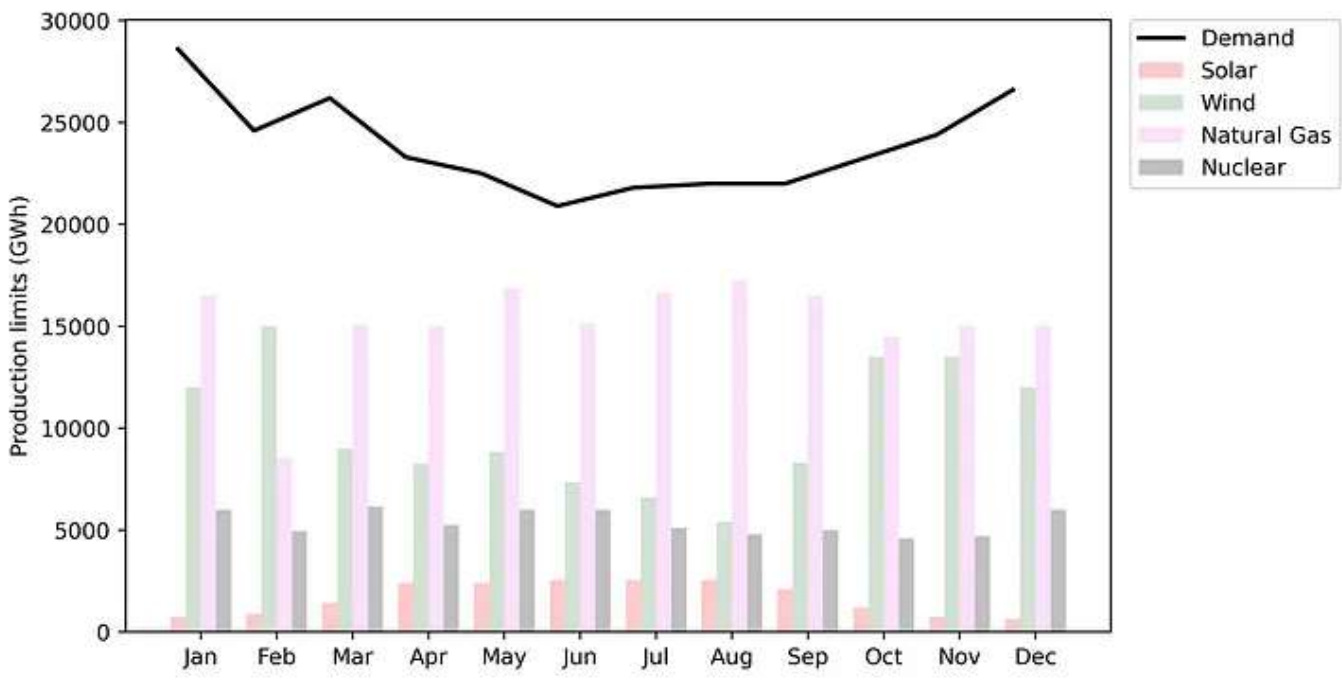
```
# Demand constraint
model.addConstrs(
    (
        gurobipy.quicksum(
            Y[Data.months[m_ind - 1], energy] + X[month, energy]
            for energy in Data.energies
        )
        >= Data.demand[month]
        for m_ind, month in enumerate(Data.months)
        if month != Data.months[0]
    ),
    name="Demand",
)
model.addConstr(
    (
        gurobipy.quicksum(
          X[Data.months[0], energy] for energy in Data.energies
        ) >= Data.demand[Data.months[0]]
    ),
    name="Demand",
)
```

In the final step, we illustrate how to set the upper limit for storage `Y` using the `model.addConstrs()` method.

```
# Upper limit for Y
model.addConstrs(
    (
        Y[month, energy] <= Data.storage_limit[energy]
        for month in Data.months
        for energy in Data.energies
    ),
    name="Storage",
)
```

4. Mock analysis

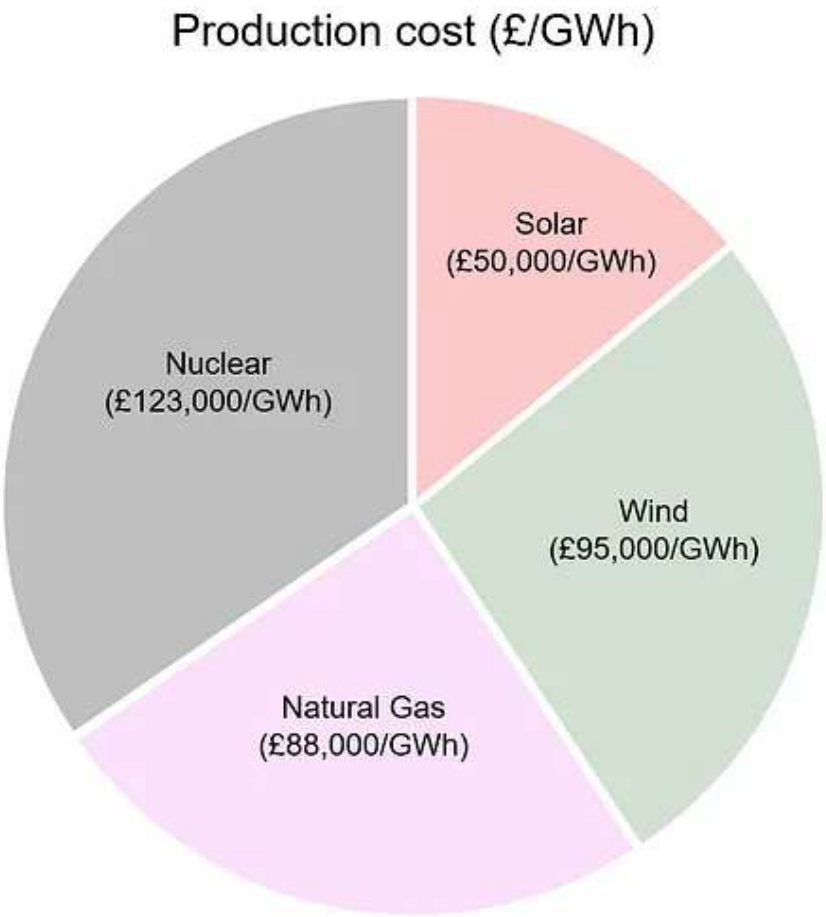
To evaluate our model, we require some mock data. Here, we use the monthly energy demand figures for the UK in the year 2022 as our mock energy demand. We also use the energy production figures for these energy sources in the UK during the year 2022, multiplied by a factor of 1.5, as an approximation for the monthly production capacity. The energy demand and production capacities are shown in figure below.



A bar chart of production limit of each energy source per month and the energy demand of the month. Source: image by the author.

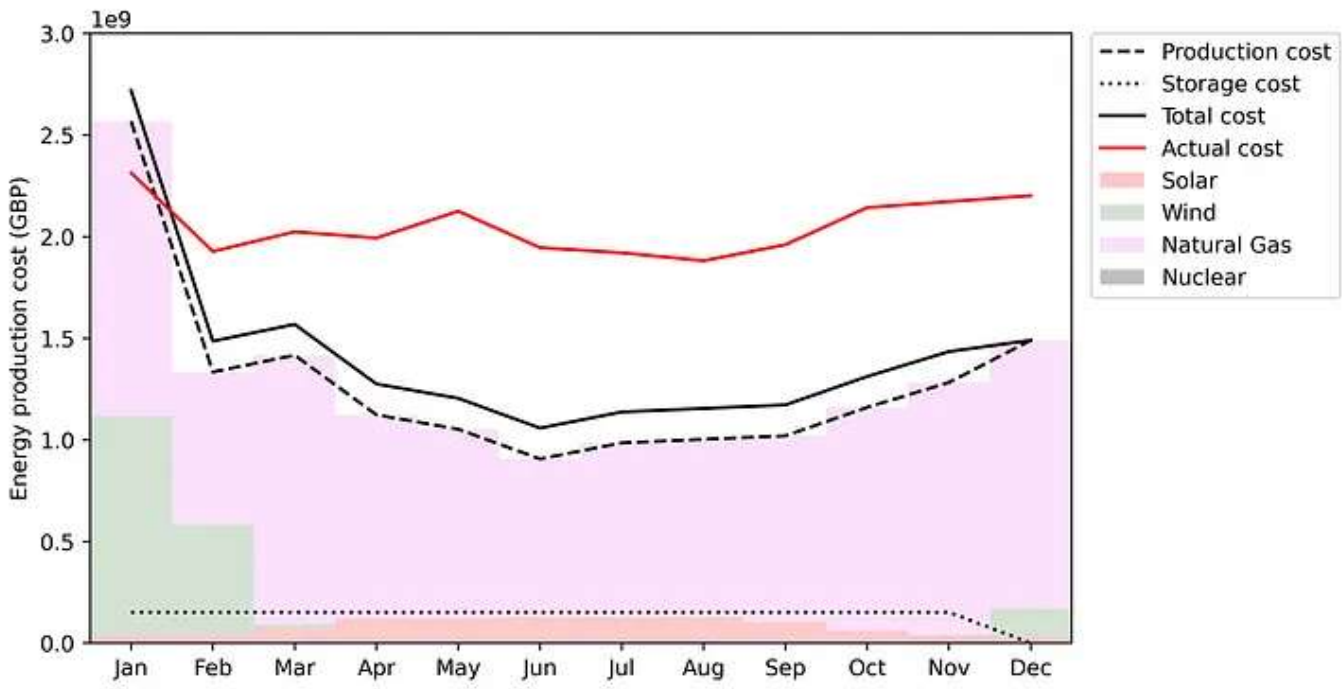
For storage capacities, we use a proxy of 10% of the EU's gas storage capacity, equivalent to approximately 9500 GWh per month, to represent natural gas storage. We allocate a storage capacity of about 2000 GWh per month for other energy sources, presuming they are stored in large energy storage facilities.

Finally, we approximate our production cost figures on the real costs of these energy sources in 2022. For storage, we adopt the physical storage cost of natural gas, which is about £16,000 per GWh. However, for other energy sources, we assume they are stored using lithium energy packs, which usually come at an exceptionally high cost compared to the physical storage cost of natural gas. To show this substantial cost difference, we set the storage cost for other energy sources at £80,000,000 per GWh.



A pie chart of the production cost in GBP per GWh. Source: image by the author.

Figure below shows the energy production and storage costs throughout the year by dashed and dotted lines, respectively. Given that the storage cost of natural gas is exceptionally inexpensive in comparison to other energy sources, and it can even be lower than the production cost of some energy sources, the optimal strategy is to store natural gas consistently throughout the year. Consequently, the storage cost is only attributed to natural gas storage. The only exception is December, where no storage occurs. This is due to our analysis being designed to conclude in December, and therefore, there is no incentive to store energy beyond our analysis period.



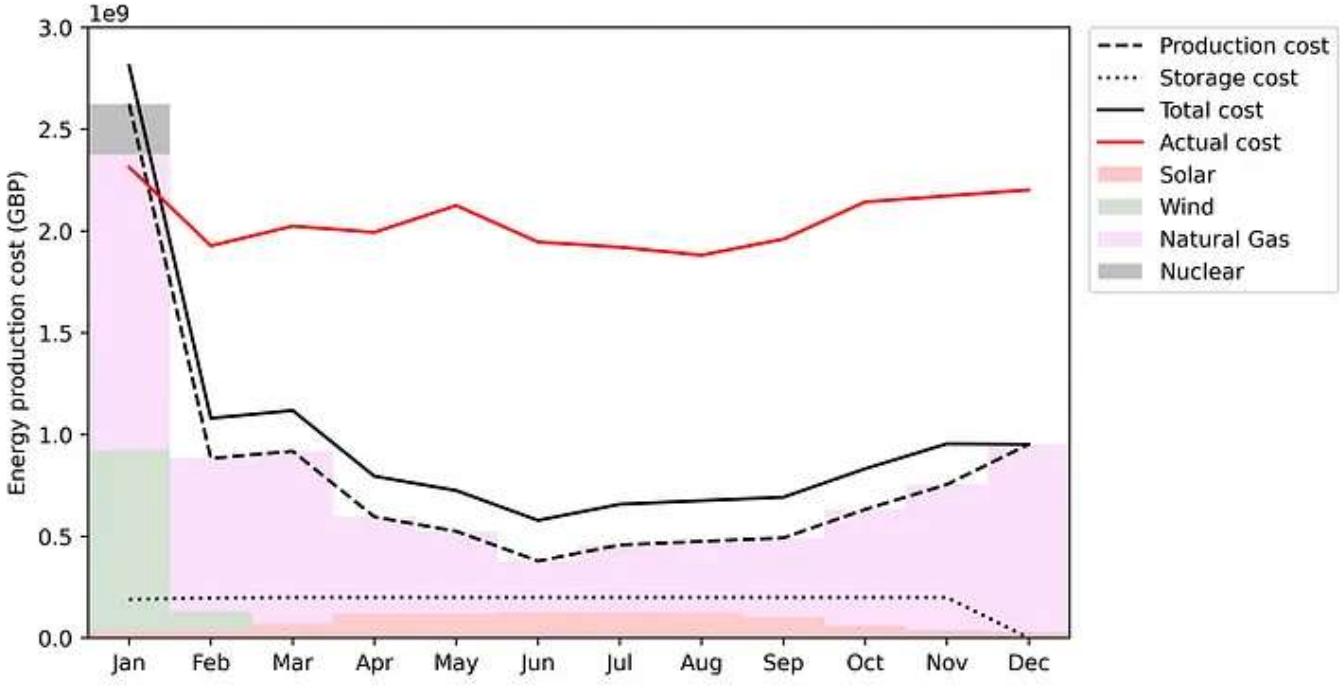
A stack bar chart of energy production per month, and the plots of costs. Source: image by the author.

Furthermore, our analysis reveals that natural gas production emerges as the most cost-effective option due to its lower production cost compared to other energy sources. In the case of wind and solar, we observe fluctuations in production over the course of the year, likely influenced by seasonal variations in production capabilities (for instance, maximum sunlight and peak solar power generation typically occur during the mid-year). This fluctuation is evident from the bar chart of production limits.

When we combine the monthly production and storage costs, the result is an optimised annual cost of £17.01 billion.

It's worth noting that we have utilised the same energy demand figures as the actual demand in the UK for 2022. This suggests that our optimised energy supply and storage allocation outperform the actual scenario. This conclusion becomes even more apparent when considering that countries like the UK often export surplus energy to other nations in exchange for revenue. The export of excess energy is not factored into our model but represents an additional potential benefit in a real-world context.

In the spirit of experimentation, we have reduced the storage cost for all energy sources except natural gas by 1000 times to simulate a hypothetical situation in which large-scale battery storage becomes significantly affordable. The results are depicted in the figure below.



A stack bar chart of energy production per month, and the plots of costs, given the revised storage costs. Source: image by the author.

In this new scenario, we can see an increase in energy storage, while production costs generally decrease because it becomes more economical to store energy rather than produce it. Interestingly, we also notice an actual need to produce nuclear energy, although only in January.

In terms of cost, the total annual cost calculated by the model in this new scenario is £11.87 billion, approximately half of the actual annual cost. This demonstrates the significant impact of storage costs on the overall total annual cost.

5. Summary

We have showcased the application of mathematical optimisation in solving a simple energy supply and demand problem, demonstrating its effectiveness in determining optimal energy production and storage quantities to minimise the total annual cost.

However, it's important to acknowledge some potential limitations and challenges associated with mathematical optimisation. One such challenge is the need to simplify the problem during the formulation of the mathematical model, which can lead to an incomplete representation of the real-world problem. Attempting to incorporate more complexities can result in a more intricate formulation, making it harder to comprehend and solve.

Furthermore, solving large-scale mathematical optimisation problems often demands substantial computational resources. To address this issue, various strategies can be employed. Heuristics offer a practical approach to quickly obtain approximate solutions that may be close to the optimal solution. Additionally, metaheuristics, such as genetic algorithm (which I explain in details here), provide higher-level techniques for iteratively searching for solutions that are sufficiently close to the optimal solution. These

approaches are valuable tools for tackling complex optimisation problems and can lead to practical solutions in scenarios where exact optimisation is challenging or computationally intensive.

Further reading

K. Y. Liow, Optimizing Object Avoidance With Genetic Algorithm in Python (2023), Towards AI @ Medium

References and data sources

[1] Monthly Electricity Statistics (2023), International Energy Agency

[2] T. Stehly and P. Duffy, 2021 Cost of Wind Energy Review (2022), National Renewable Energy Laboratory

[3] M. Roser, Why did renewables become so cheap so fast? (2020), Our World In Data

[4] EU reaches 90% gas storage target ahead of winter (2023), European Commission

[5] GlobalData, Top five energy storage projects in the UK (2023), Power Technology

[6] J. Witwer, The Role of Methane Storage in Achieving Renewable Energy Goals and Enhancing Energy Resiliency (2022), Onboard Dynamics

Optimization Energy Mathematics Gurobi Data Science

Written by Kong You Liow

Follow

19 Followers · Writer for Towards Data Science

Data Scientist | PhD in Astrophysics | https://www.linkedin.com/in/kong-you-liow/

More from Kong You Liow and Towards Data Science

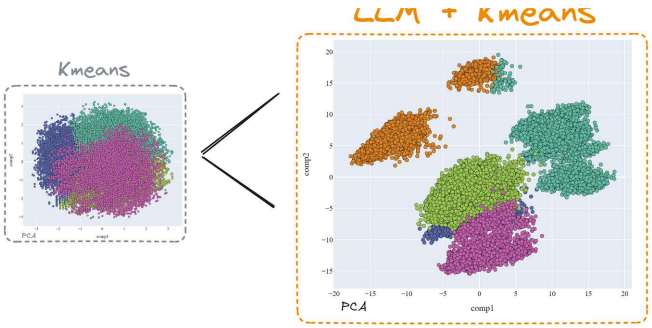
Kong You Liow in Towards AI

Optimizing Object Avoidance With Genetic Algorithm in Python

Inspiring optimization through evolutionary genetic principles

10 min read · Jun 3

25



Damian Gil in Towards Data Science

Mastering Customer Segmentation with LLM

Unlock advanced customer segmentation techniques using LLMs, and improve your…

23 min read · Sep 27

2.2K 22

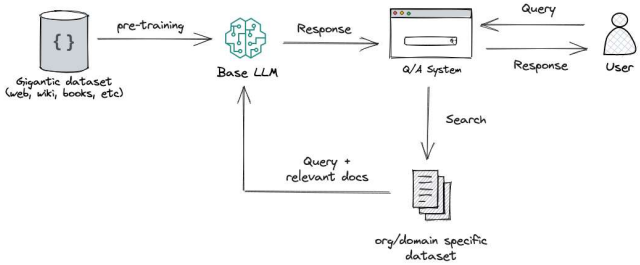
Giuseppe Scalamogna in Towards Data Science

New ChatGPT Prompt Engineering Technique: Program Simulation

A potentially novel technique for turning a ChatGPT prompt into a mini-app.

9 min read · Sep 4

1.7K 17



Heiko Hotz in Towards Data Science

RAG vs Finetuning — Which Is the Best Tool to Boost Your LLM...

The definitive guide for choosing the right method for your use case

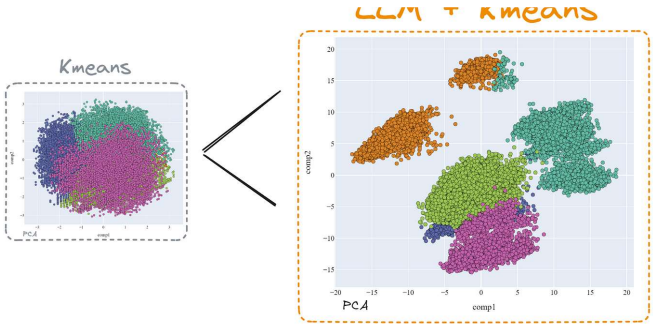
· 19 min read · Aug 25

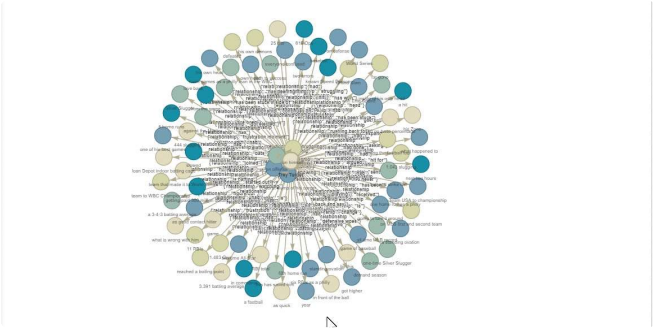
2.3K 17

See all from Kong You Liow

See all from Towards Data Science

Recommended from Medium





Damian Gil in Towards Data Science

Mastering Customer Segmentation with LLM

Unlock advanced customer segmentation techniques using LLMs, and improve your...

23 min read · Sep 27

2.2K 22

Wenqi Glantz in Better Programming

7 Query Strategies for Navigating Knowledge Graphs With...

Exploring NebulaGraph RAG Pipeline with the Philadelphia Phillies

· 17 min read · 4 days ago

501 4

Lists

Predictive Modeling w/ Python

20 stories · 452 saves

New_Reading_List

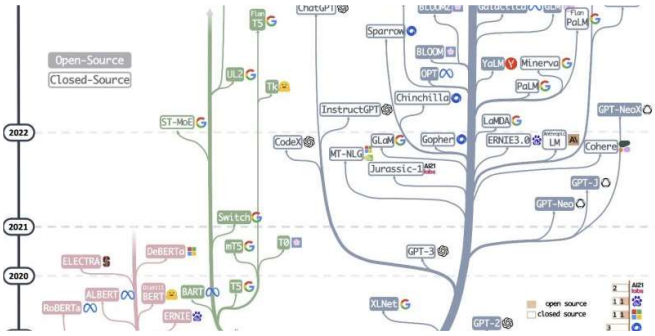
174 stories · 133 saves

Practical Guides to Machine Learning

10 stories · 519 saves

Coding & Development

11 stories · 200 saves



Haifeng Li

A Tutorial on LLM

Generative artificial intelligence (GenAI), especially ChatGPT, captures everyone's...

Jim the AI Whisperer in The Generator

The 19 tell-tale signs an article was written by AI

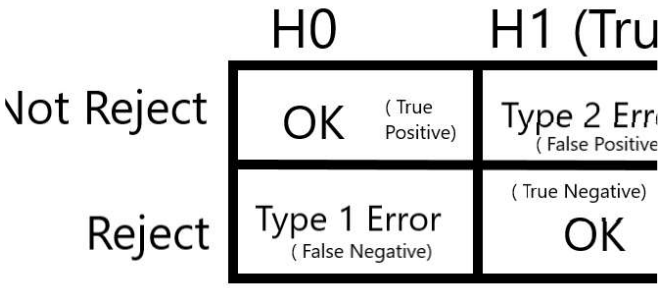
Avoid these AI cliches (or use them to detect AI writing in the wild)

15 min read · Sep 14

374

✨ · 16 min read · 5 days ago

2.4K 55



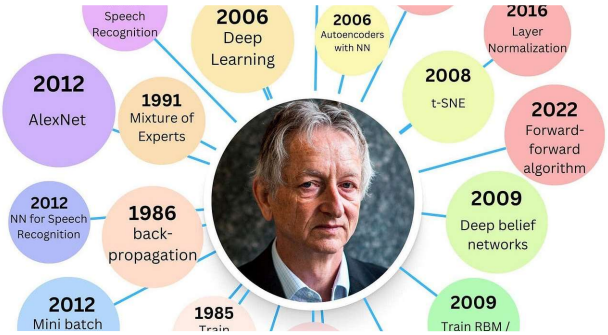
Kamna Sinha in Data At The Core !

Hypothesis Test And All About P Value,T test,Chi Square Test, Ano...

Hypothesis Test

4 min read · 6 days ago

22



Coursesteach

Deep Learning (Part 1)

📚Chapter1: Introduction of Deep learning:Understanding Basic Neural...

12 min read · Aug 6

126 2

See more recommendations