

MEASURE ENERGY CONSUMPTION FOR DEVELOPMENT PART-2

To measure energy consumption for development, you can follow these steps:

Define Your Metrics: Clearly define the metrics you want to measure. This might include electricity consumption, server load, or other relevant energy-related metrics.

Use Monitoring Tools: Implement monitoring tools or services like Prometheus, Grafana, or specialized energy monitoring systems to track energy consumption in your development environment.

Measure Hardware: If you're developing on physical hardware, use energy meters or smart plugs to measure the power consumption of your servers, workstations, and other equipment.

Measure Software: Use software tools to monitor energy consumption within your code. Profiling tools can help you identify energy-intensive code segments.

Benchmark and Optimize: Create benchmarks to compare energy usage before and after optimizations. Optimize code, hardware configurations, and infrastructure to reduce energy consumption.

Analyze Results: Analyze the data collected to identify energy-hungry components and areas for improvement.

Implement Energy-Efficient Practices: Encourage energy-efficient coding practices, such as optimizing algorithms, reducing unnecessary computations, and minimizing server idle times.

Regular Reporting: Set up a system for regular reporting and review to track your progress



in reducing energy consumption during development.

Educate Your Team: Make sure your development team is aware of the importance of energy efficiency and how their actions impact it.

Sustainability Initiatives: Consider broader sustainability initiatives within your organization to reduce the environmental impact of development.

Remember that energy consumption for development is a multifaceted issue, and measuring it requires a combination of hardware and software monitoring, along with ongoing efforts to reduce energy usage.

PROJECT CODING:

Supercharging Sustainability: Harnessing Superconductors for Renewable Energy Innovation

Part 1: Introduction

The looming danger of intensified climate change, driven by the relentless emission of greenhouse gases from fossil fuel-based electricity generation, underscores the urgent need to transition towards sustainable energy sources swiftly. The potential discovery of a room-temperature superconductor, LK-99, by South Korean researchers provides the opportunity for Superconducting Magnetic Energy Storage [SMES] which was invented in 1970. This technology is meant to effectively store energy through induction where electrons flow continuously within a superconducting coil wrapped around a core usually made from a niobium-titanium alloy embedded in a copper matrix. The primary benefits of SMES over other methods include nearly instantaneous power availability with a minimal time delay during charge and discharge, low power loss due to negligible electrical resistance, and enhanced reliability as its stationary components contribute to system stability. However, SMES systems can face increased complexity and operational cost due to their dependence on cryogenic cooling to maintain the superconducting state of the superconducting coil. The chance of LK-99's use in such a system remains low due to current research indicating a limited critical current of 250 mA (the maximum current where $R=0$) and a constrained maximum magnetic field threshold for maintaining superconductivity.

The primary objective of this notebook is to identify the plausibility of creating a solar battery that, when combined with a solar panel installation or any other renewable energy



source, can sustain the energy consumption of an ideal American city. In order to do this, the data analysis algorithms will be employed for prediction and optimization

Part 1.1 Aurora City: Pioneering Sustainability in a Vibrant Metropolis

Let's begin by imagining we are in Aurora City with a population of 1,024,144 people, and is a bustling metropolis known for its vibrant cultural scene, modern architecture, and diverse population. The city boasts a mix of historic neighborhoods and futuristic skyscrapers, offering a unique blend of old-world charm and contemporary innovation. With a strong emphasis on sustainability, Aurora City is home to lush parks, efficient public transportation, and a commitment to renewable energy sources. Its world-class universities, thriving tech industry, and rich culinary scene make it a hub of creativity and opportunity. In line with that commitment to sustainability, the city's government has pledged to move towards 100% solar electricity generation within the next 10 years.

```
!pip install prophet
```

```
!pip install cplex
```

```
!pip install stable_baselines3
```

```
Requirement already satisfied: prophet in /opt/conda/lib/python3.10/site-packages (1.1.1)
```

```
Requirement already satisfied: cmdstanpy>=1.0.4 in /opt/conda/lib/python3.10/site-packages (from prophet) (1.1.0)
```

```
Requirement already satisfied: numpy>=1.15.4 in /opt/conda/lib/python3.10/site-packages (from prophet) (1.23.5)
```

```
Requirement already satisfied: pandas>=1.0.4 in /opt/conda/lib/python3.10/site-packages (from prophet) (1.5.3)
```

```
Requirement already satisfied: matplotlib>=2.0.0 in /opt/conda/lib/python3.10/site-packages (from prophet) (3.7.1)
```

```
Requirement already satisfied: LunarCalendar>=0.0.9 in /opt/conda/lib/python3.10/site-packages (from prophet) (0.0.9)
```

```
Requirement already satisfied: convertdate>=2.1.2 in /opt/conda/lib/python3.10/site-packages (from prophet) (2.4.0)
```

```
Requirement already satisfied: holidays>=0.14.2 in /opt/conda/lib/python3.10/site-packages (from prophet) (0.24)
```



Requirement already satisfied: setuptools>=42 in /opt/conda/lib/python3.10/site-packages (from prophet) (59.8.0)

Requirement already satisfied: setuptools-git>=1.2 in /opt/conda/lib/python3.10/site-packages (from prophet) (1.2)

Requirement already satisfied: python-dateutil>=2.8.0 in /opt/conda/lib/python3.10/site-packages (from prophet) (2.8.2)

Requirement already satisfied: tqdm>=4.36.1 in /opt/conda/lib/python3.10/site-packages (from prophet) (4.65.0)

Requirement already satisfied: wheel>=0.37.0 in /opt/conda/lib/python3.10/site-packages (from prophet) (0.40.0)

Requirement already satisfied: pymeeus<=1,>=0.3.13 in /opt/conda/lib/python3.10/site-packages (from convertdate>=2.1.2->prophet) (0.5.12)

Requirement already satisfied: hijri-converter in /opt/conda/lib/python3.10/site-packages (from holidays>=0.14.2->prophet) (2.3.1)

Requirement already satisfied: korean-lunar-calendar in /opt/conda/lib/python3.10/site-packages (from holidays>=0.14.2->prophet) (0.3.1)

Requirement already satisfied: ephem>=3.7.5.3 in /opt/conda/lib/python3.10/site-packages (from LunarCalendar>=0.0.9->prophet) (4.1.4)

Requirement already satisfied: pytz in /opt/conda/lib/python3.10/site-packages (from LunarCalendar>=0.0.9->prophet) (2023.3)

Requirement already satisfied: contourpy>=1.0.1 in /opt/conda/lib/python3.10/site-packages (from matplotlib>=2.0.0->prophet) (1.1.0)

Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.10/site-packages (from matplotlib>=2.0.0->prophet) (0.11.0)

Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.10/site-packages (from matplotlib>=2.0.0->prophet) (4.40.0)

Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/lib/python3.10/site-packages (from matplotlib>=2.0.0->prophet) (1.4.4)

Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.10/site-packages (from matplotlib>=2.0.0->prophet) (21.3)

Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.10/site-packages (from matplotlib>=2.0.0->prophet) (9.5.0)




Requirement already satisfied: pyparsing>=2.3.1 in /opt/conda/lib/python3.10/site-packages (from matplotlib>=2.0.0->prophet) (3.0.9)

Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.10/site-packages (from python-dateutil>=2.8.0->prophet) (1.16.0)

Collecting cplex

Downloading cplex-22.1.1.0-cp310-cp310-manylinux1_x86_64.whl (44.2 MB)


0:00:00  44.2/44.2 MB 33.4 MB/s eta

Installing collected packages: cplex

Successfully installed cplex-22.1.1.0


Collecting stable_baselines3

Downloading stable_baselines3-2.1.0-py3-none-any.whl (178 kB)

0:00:00  178.7/178.7 kB 4.6 MB/s eta

Collecting gymnasium<0.30,>=0.28.1 (from stable_baselines3)

Downloading gymnasium-0.29.0-py3-none-any.whl (953 kB)

0:00:00  953.8/953.8 kB 25.9 MB/s eta

Requirement already satisfied: numpy>=1.20 in /opt/conda/lib/python3.10/site-packages (from stable_baselines3) (1.23.5)

Requirement already satisfied: torch>=1.13 in /opt/conda/lib/python3.10/site-packages (from stable_baselines3) (2.0.0+cpu)

Requirement already satisfied: cloudpickle in /opt/conda/lib/python3.10/site-packages (from stable_baselines3) (2.2.1)

Requirement already satisfied: pandas in /opt/conda/lib/python3.10/site-packages (from stable_baselines3) (1.5.3)

Requirement already satisfied: matplotlib in /opt/conda/lib/python3.10/site-packages (from stable_baselines3) (3.7.1)

Requirement already satisfied: typing-extensions>=4.3.0 in /opt/conda/lib/python3.10/site-packages (from gymnasium<0.30,>=0.28.1->stable_baselines3) (4.6.3)



Collecting farama-notifications>=0.0.1 (from gymnasium<0.30,>=0.28.1->stable_baselines3)

Downloading Farama_Notifications-0.0.4-py3-none-any.whl (2.5 kB)

Requirement already satisfied: filelock in /opt/conda/lib/python3.10/site-packages (from torch>=1.13->stable_baselines3) (3.12.2)

Requirement already satisfied: sympy in /opt/conda/lib/python3.10/site-packages (from torch>=1.13->stable_baselines3) (1.12)

Requirement already satisfied: networkx in /opt/conda/lib/python3.10/site-packages (from torch>=1.13->stable_baselines3) (3.1)

Requirement already satisfied: jinja2 in /opt/conda/lib/python3.10/site-packages (from torch>=1.13->stable_baselines3) (3.1.2)

Requirement already satisfied: contourpy>=1.0.1 in /opt/conda/lib/python3.10/site-packages (from matplotlib->stable_baselines3) (1.1.0)

Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.10/site-packages (from matplotlib->stable_baselines3) (0.11.0)

Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.10/site-packages (from matplotlib->stable_baselines3) (4.40.0)

Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/lib/python3.10/site-packages (from matplotlib->stable_baselines3) (1.4.4)

Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.10/site-packages (from matplotlib->stable_baselines3) (21.3)

Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.10/site-packages (from matplotlib->stable_baselines3) (9.5.0)

Requirement already satisfied: pyparsing>=2.3.1 in /opt/conda/lib/python3.10/site-packages (from matplotlib->stable_baselines3) (3.0.9)

Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python3.10/site-packages (from matplotlib->stable_baselines3) (2.8.2)

Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.10/site-packages (from pandas->stable_baselines3) (2023.3)

Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.10/site-packages (from python-dateutil>=2.7->matplotlib->stable_baselines3) (1.16.0)

Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/lib/python3.10/site-



packages (from jinja2->torch>=1.13->stable_baselines3) (2.1.3)

Requirement already satisfied: mpmath>=0.19 in /opt/conda/lib/python3.10/site-packages (from sympy->torch>=1.13->stable_baselines3) (1.3.0)

Installing collected packages: farama-notifications, gymnasium, stable_baselines3

Attempting uninstall: gymnasium

Found existing installation: Gymnasium 0.26.3

Uninstalling Gymnasium-0.26.3:

Successfully uninstalled Gymnasium-0.26.3

Successfully installed farama-notifications-0.0.4 gymnasium-0.29.0 stable_baselines3-2.1.0

```
import numpy as np
```

```
import pandas as pd
```

```
import random
```

```
from datetime import datetime, timedelta
```

```
import math
```

```
import tensorflow as tf
```

```
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import LSTM, Dense
```

```
from prophet import Prophet
```

```
from scipy.optimize import minimize
```

```
import pulp
```

```
import matplotlib.pyplot as plt
```

```
import statistics
```

```
import sys
```

```
from stable_baselines3 import PPO
```

```
from stable_baselines3.common.env_util import make_vec_env
```

```
import gymnasium as gym
```




```
from gymnasium import spaces
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy  
version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5
```

```
warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}")
```

```
/opt/conda/lib/python3.10/site-packages/tensorflow_io/python/ops/__init__.py:98:  
UserWarning: unable to load libtensorflow_io_plugins.so: unable to open file:  
libtensorflow_io_plugins.so, from paths: ['/opt/conda/lib/python3.10/site-  
packages/tensorflow_io/python/ops/libtensorflow_io_plugins.so']
```

```
caused by: ['/opt/conda/lib/python3.10/site-  
packages/tensorflow_io/python/ops/libtensorflow_io_plugins.so: undefined symbol:  
_ZN3tsl6StatusC1EN10tensorflow5error4CodeESt17basic_string_viewIcSt11char_traitsIcEENS_14SourceLocationE']
```

```
warnings.warn(f"unable to load libtensorflow_io_plugins.so: {e}")
```

```
/opt/conda/lib/python3.10/site-packages/tensorflow_io/python/ops/__init__.py:104:  
UserWarning: file system plugins are not loaded: unable to open file: libtensorflow_io.so,  
from paths: ['/opt/conda/lib/python3.10/site-  
packages/tensorflow_io/python/ops/libtensorflow_io.so']
```

```
caused by: ['/opt/conda/lib/python3.10/site-  
packages/tensorflow_io/python/ops/libtensorflow_io.so: undefined symbol:  
_ZTVN10tensorflow13GcsFileSystemE']
```

```
warnings.warn(f"file system plugins are not loaded: {e}")
```

```
def get_readings_for_date(csv_file_path, target_date, time_column, energy_column):
```

```
    # Read the CSV file into a DataFrame
```

```
    df = pd.read_csv(csv_file_path)
```

```
    df[time_column] = pd.to_datetime(df[time_column])
```

```
    df['filter_date'] = df[time_column].dt.date
```

```
    filtered_df = df[df['filter_date'] == target_date]
```

```
    return filtered_df[[time_column, energy_column]]
```

```
def generate_random_date(delim):
```




```
start_date = datetime.strptime('2015-9-1', '%Y-%m-%d')
```

```
end_date = datetime.strptime('2018-8-2', '%Y-%m-%d')
```

```
days_difference = (end_date - start_date).days
```

```
random_days = random.randint(0, days_difference)
```

```
random_date = (start_date + timedelta(days=random_days)).date()
```

```
return f"{random_date.year}{delim}{random_date.month}{delim}{random_date.day}"
```

```
target_date = pd.to_datetime(generate_random_date("/"))
```

```
month = target_date.month
```

```
target_date
```

```
Timestamp('2018-06-03 00:00:00')
```

Part 2: Data Cleaning & Analysis

Part 2.1: Energizing Change: The Quest for Sustainable Power in Aurora City

To pave the way for a sustainable energy transformation, the city government assembles a team of experts, Solar Nexus Taskforce (SNT), tasked with crafting a viable strategy. Their objective was to orchestrate the reduction of the city's conventional power plants while simultaneously bolstering the deployment of solar panels and SMES solar batteries.

With determination, SNT promptly swings into action. The first step on their journey is to gather 2-3 years worth of energy consumption data from the city's power grid and energy production data from an existing solar field that will be undergoing a substantial expansion in the coming months.

```
csv_file_path = '/kaggle/input/solar-energy-production/Solar_Energy_Production.csv' #  
Solar energy production data sourced from solar installations in Calgary, Alberta
```

```
def get_production(date):
```



```

    production_readings_for_date = get_readings_for_date(csv_file_path,
date.date(), 'date', 'kWh').groupby('date')['kWh'].sum().reset_index()

    production_readings_for_date["kWh"] = production_readings_for_date["kWh"]*1000 #
To account for a major scale-up in functionality

    return production_readings_for_date

production_readings_for_date = get_production(target_date)

consumption_csv_file_path = '/kaggle/input/hourly-energy-consumption/AEP_hourly.csv'

# Electrical energy consumption data from American Electric Power Company, Inc (AEP)
which is among the nation's largest generators of electricity, owning nearly 38,000
megawatts of generating capacity in the U.S and serving nearly 5 million customers.

def get_consumption(date):

    consumption_readings_for_date = get_readings_for_date(consumption_csv_file_path,
date.date(), 'Datetime', 'AEP_MW').sort_values(by='Datetime')

consumption_readings_for_date["AEP_MW"] = consumption_readings_for_date["AEP_MW"]
*200

    # converts MW to kW and also scales down the data by a factor of 5 to make more
representative of the demands of a city the size of Auroa City

    # Set the 'datetime' column as the index

    consumption_readings_for_date.set_index('Datetime', inplace=True)

    return consumption_readings_for_date

consumption_readings_for_date = get_consumption(target_date)

production_readings_for_date.set_index("date", inplace=True)

# Adding in zeroes for the times not included with the assumption that those were times
when the sun was not up

new_row = {"kWh":0}

for date in consumption_readings_for_date.index:

    if date not in production_readings_for_date.index:

```



```
production_readings_for_date.loc[date] = new_row
```

```
production_readings_for_date = production_readings_for_date.sort_values(by='date')
```

```
production_readings_for_date
```

```
kWh
```

```
date
```

```
2018-06-03 00:00:00 0.0
```

```
2018-06-03 01:00:00 0.0
```

```
2018-06-03 02:00:00 0.0
```

```
2018-06-03 03:00:00 0.0
```

```
2018-06-03 04:00:00 0.0
```

```
2018-06-03 05:00:00 5364.0
```

```
2018-06-03 06:00:00 89479.0
```

```
2018-06-03 07:00:00 243391.0
```

```
2018-06-03 08:00:00 506163.0
```

```
2018-06-03 09:00:00 840801.0
```

```
2018-06-03 10:00:00 1022944.0
```

```
2018-06-03 11:00:00 1124456.0
```

```
2018-06-03 12:00:00 1052887.0
```

```
2018-06-03 13:00:00 662972.0
```

```
2018-06-03 14:00:00 1063243.0
```

```
2018-06-03 15:00:00 1084414.0
```

```
2018-06-03 16:00:00 895397.0
```

```
2018-06-03 17:00:00 491990.0
```

```
2018-06-03 18:00:00 343648.0
```



2018-06-03 19:00:00 176110.0

2018-06-03 20:00:00 80177.0

2018-06-03 21:00:00 4680.0

2018-06-03 22:00:00 0.0

2018-06-03 23:00:00 0.0

```
energy_demand = ((consumption_readings_for_date['AEP_MW']-  
production_readings_for_date["kWh"]).mean())/1e6
```

```
total = (consumption_readings_for_date['AEP_MW'].mean())/1e6
```

```
f"Energy Demand Unfilled by Solar Installation on {target_date.date()}:  
{round(energy_demand,2)} GWh ({round(energy_demand/total,3)*100}% of demand)"
```

```
'Energy Demand Unfilled by Solar Installation on 2018-06-03: 2.36 GWh  
(85.39999999999999% of demand)'
```

```
import matplotlib.pyplot as plt
```

```
# Plot datetime against energy consumption
```

```
plt.figure(figsize=(10, 6)) # Adjust the figure size as needed
```

```
plt.plot(consumption_readings_for_date.index, consumption_readings_for_date['AEP_MW'],  
production_readings_for_date.index, production_readings_for_date["kWh"])
```

```
plt.xlabel('Time (mm-dd hh)')
```

```
plt.ylabel('Power (kW)')
```

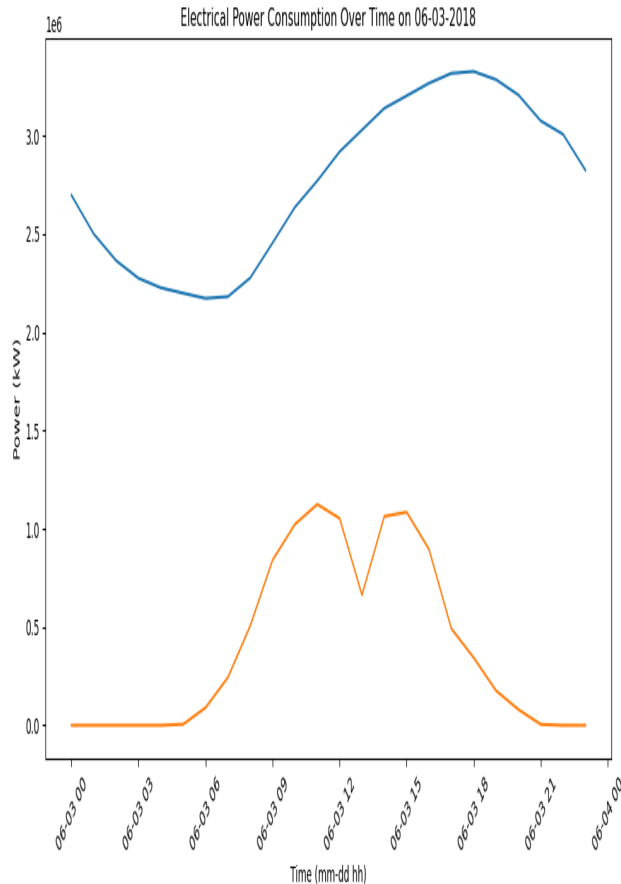
```
plt.title(f'Electrical Power Consumption Over Time on {datetime.strftime(target_date,"%m  
-%d-%Y")}'))
```

```
plt.xticks(rotation=45) # Rotate x-axis labels for better visibility
```

```
plt.tight_layout()
```

```
plt.show()
```





Supercharging Sustainability: Harnessing Superconductors for Renewable Energy Innovation

Part 1: Introduction

The looming danger of intensified climate change, driven by the relentless emission of greenhouse gases from fossil fuel-based electricity generation, underscores the urgent need to transition towards sustainable energy sources swiftly. The potential discovery of a room-temperature superconductor, LK-99, by South Korean researchers provides the opportunity for Superconducting Magnetic Energy Storage [SMES] which was invented in 1970. This technology is meant to effectively store energy through induction where electrons flow continuously within a superconducting coil wrapped around a core usually made from a niobium-titanium alloy embedded in a copper matrix. The primary benefits of SMES over other methods include nearly instantaneous power availability with a minimal time delay during charge and discharge, low power loss due to negligible electrical resistance, and enhanced reliability as its stationary components contribute to system stability. However, SMES systems can face increased complexity and operational cost due



to their dependence on cryogenic cooling to maintain the superconducting state of the superconducting coil. The chance of LK-99's use in such a system remains low due to current research indicating a limited critical current of 250 mA (the maximum current where $R=0$) and a constrained maximum magnetic field threshold for maintaining superconductivity.

The primary objective of this notebook is to identify the plausibility of creating a solar battery that, when combined with a solar panel installation or any other renewable energy source, can sustain the energy consumption of an ideal American city. In order to do this, the data analysis algorithms will be employed for prediction and optimization

Part 1.1 Aurora City: Pioneering Sustainability in a Vibrant Metropolis

Let's begin by imagining we are in Aurora City with a population of 1,024,144 people, and is a bustling metropolis known for its vibrant cultural scene, modern architecture, and diverse population. The city boasts a mix of historic neighborhoods and futuristic skyscrapers, offering a unique blend of old-world charm and contemporary innovation. With a strong emphasis on sustainability, Aurora City is home to lush parks, efficient public transportation, and a commitment to renewable energy sources. Its world-class universities, thriving tech industry, and rich culinary scene make it a hub of creativity and opportunity. In line with that commitment to sustainability, the city's government has pledged to move towards 100% solar electricity generation within the next 10 years.

```
!pip install prophet
```

```
!pip install cplex
```

```
!pip install stable_baselines3
```

```
import numpy as np
```

```
import pandas as pd
```

```
import random
```

```
from datetime import datetime, timedelta
```

```
import math
```

```
import tensorflow as tf
```



```

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

from prophet import Prophet

from scipy.optimize import minimize

import pulp

import matplotlib.pyplot as plt

import statistics

import sys

from stable_baselines3 import PPO

from stable_baselines3.common.env_util import make_vec_env

import gymnasium as gym

from gymnasium import spaces

/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy
version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5

    warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}")

/opt/conda/lib/python3.10/site-packages/tensorflow_io/python/ops/__init__.py:98:
UserWarning: unable to load libtensorflow_io_plugins.so: unable to open file:
libtensorflow_io_plugins.so, from paths: ['/opt/conda/lib/python3.10/site-
packages/tensorflow_io/python/ops/libtensorflow_io_plugins.so']

caused by: ['/opt/conda/lib/python3.10/site-
packages/tensorflow_io/python/ops/libtensorflow_io_plugins.so: undefined symbol:
_ZN3tsl6StatusC1EN10tensorflow5error4CodeESt17basic_string_viewIcSt11char_traitsIcEENS_14SourceLocationE']

    warnings.warn(f"unable to load libtensorflow_io_plugins.so: {e}")

/opt/conda/lib/python3.10/site-packages/tensorflow_io/python/ops/__init__.py:104:
UserWarning: file system plugins are not loaded: unable to open file: libtensorflow_io.so,
from paths: ['/opt/conda/lib/python3.10/site-
packages/tensorflow_io/python/ops/libtensorflow_io.so']

caused by: ['/opt/conda/lib/python3.10/site-
packages/tensorflow_io/python/ops/libtensorflow_io.so: undefined symbol:

```




```
_ZTVN10tensorflow13GcsFileSystemE']
```

```
warnings.warn(f"file system plugins are not loaded: {e}")
```

```
def get_readings_for_date(csv_file_path, target_date, time_column, energy_column):
```

```
    # Read the CSV file into a DataFrame
```

```
    df = pd.read_csv(csv_file_path)
```

```
    df[time_column] = pd.to_datetime(df[time_column])
```

```
    df['filter_date'] = df[time_column].dt.date
```

```
    filtered_df = df[df['filter_date'] == target_date]
```

```
    return filtered_df[[time_column, energy_column]]
```

```
def generate_random_date(delim):
```

```
    start_date = datetime.strptime('2015-9-1', '%Y-%m-%d')
```

```
    end_date = datetime.strptime('2018-8-2', '%Y-%m-%d')
```

```
    days_difference = (end_date - start_date).days
```

```
    random_days = random.randint(0, days_difference)
```

```
    random_date = (start_date + timedelta(days=random_days)).date()
```

```
    return f"{random_date.year}{delim}{random_date.month}{delim}{random_date.day}"
```

```
target_date = pd.to_datetime(generate_random_date("/"))
```

```
month = target_date.month
```

```
target_date
```

```
Timestamp('2018-06-03 00:00:00')
```

Part 2: Data Cleaning & Analysis



Part 2.1: Energizing Change: The Quest for Sustainable Power in Aurora City

To pave the way for a sustainable energy transformation, the city government assembles a team of experts, Solar Nexus Taskforce (SNT), tasked with crafting a viable strategy. Their objective was to orchestrate the reduction of the city's conventional power plants while simultaneously bolstering the deployment of solar panels and SMES solar batteries.

With determination, SNT promptly swings into action. The first step on their journey is to gather 2-3 years worth of energy consumption data from the city's power grid and energy production data from an existing solar field that will be undergoing a substantial expansion in the coming months.

```
csv_file_path = '/kaggle/input/solar-energy-production/Solar_Energy_Production.csv' #  
Solar energy production data sourced from solar installations in Calgary, Alberta
```

```
def get_production(date):
```

```
    production_readings_for_date = get_readings_for_date(csv_file_path,  
date.date(), 'date', 'kWh').groupby('date')['kWh'].sum().reset_index()
```

```
    production_readings_for_date["kWh"] = production_readings_for_date["kWh"]*1000 #  
To account for a major scale-up in functionality
```

```
    return production_readings_for_date
```

```
production_readings_for_date = get_production(target_date)
```

```
consumption_csv_file_path = '/kaggle/input/hourly-energy-consumption/AEP_hourly.csv'
```

```
# Electrical energy consumption data from American Electric Power Company, Inc (AEP)  
which is among the nation's largest generators of electricity, owning nearly 38,000  
megawatts of generating capacity in the U.S and serving nearly 5 million customers.
```

```
def get_consumption(date):
```

```
    consumption_readings_for_date = get_readings_for_date(consumption_csv_file_path,  
date.date(), 'Datetime', 'AEP_MW').sort_values(by='Datetime')
```

```
consumption_readings_for_date["AEP_MW"] = consumption_readings_for_date["AEP_MW"]  
*200
```

```
# converts MW to kW and also scales down the data by a factor of 5 to make more
```



representative of the demands of a city the size of Auroa City

```
# Set the 'datetime' column as the index
```

```
consumption_readings_for_date.set_index('Datetime', inplace=True)
```

```
return consumption_readings_for_date
```

```
consumption_readings_for_date = get_consumption(target_date)
```

```
production_readings_for_date.set_index("date", inplace=True)
```

```
# Adding in zeroes for the times not included with the assumption that those were times  
when the sun was not up
```

```
new_row = {"kWh":0}
```

```
for date in consumption_readings_for_date.index:
```

```
    if date not in production_readings_for_date.index:
```

```
        production_readings_for_date.loc[date] = new_row
```

```
production_readings_for_date = production_readings_for_date.sort_values(by='date')
```

```
production_readings_for_date
```

```
kWh
```

```
date
```

```
2018-06-03 00:00:00  0.0
```

```
2018-06-03 01:00:00  0.0
```

```
2018-06-03 02:00:00  0.0
```

```
2018-06-03 03:00:00  0.0
```

```
2018-06-03 04:00:00  0.0
```

```
2018-06-03 05:00:00 5364.0
```

```
2018-06-03 06:00:00 89479.0
```

```
2018-06-03 07:00:00 243391.0
```



```
2018-06-03 08:00:00 506163.0
2018-06-03 09:00:00 840801.0
2018-06-03 10:00:00 1022944.0
2018-06-03 11:00:00 1124456.0
2018-06-03 12:00:00 1052887.0
2018-06-03 13:00:00 662972.0
2018-06-03 14:00:00 1063243.0
2018-06-03 15:00:00 1084414.0
2018-06-03 16:00:00 895397.0
2018-06-03 17:00:00 491990.0
2018-06-03 18:00:00 343648.0
2018-06-03 19:00:00 176110.0
2018-06-03 20:00:00 80177.0
2018-06-03 21:00:00 4680.0
2018-06-03 22:00:00 0.0
2018-06-03 23:00:00 0.0
```

```
energy_demand = ((consumption_readings_for_date['AEP_MW']-
production_readings_for_date["kWh"]).mean())/1e6
```

```
total = (consumption_readings_for_date['AEP_MW'].mean())/1e6
```

```
f"Energy Demand Unfilled by Solar Installation on {target_date.date()}:
{round(energy_demand,2)} GWh ({round(energy_demand/total,3)*100}% of demand)"
```

```
'Energy Demand Unfilled by Solar Installation on 2018-06-03: 2.36 GWh
(85.39999999999999% of demand)'
```

```
import matplotlib.pyplot as plt
```

```
# Plot datetime against energy consumption
```

```
plt.figure(figsize=(10, 6)) # Adjust the figure size as needed
```

```
plt.plot(consumption_readings_for_date.index, consumption_readings_for_date['AEP_MW'],
```



```
production_readings_for_date.index, production_readings_for_date["kWh"])

plt.xlabel('Time (mm-dd hh)')

plt.ylabel('Power (kW)')

plt.title(f'Electrical Power Consumption Over Time on {datetime.strftime(target_date,"%m-%d-%Y")}')

plt.xticks(rotation=45) # Rotate x-axis labels for better visibility

plt.tight_layout()

plt.show()
```

Part 2.2: Unveiling Challenges: The City's Energy Quest Confronts Shadows

Amid caffeine-fueled calculations and simulations, a stark truth emerges from the data scientists' endeavors. A formidable challenge looms over the team. Within the conference room, a somber air settles as the lead data scientist projects slide after slide of graphs and statistics that each echoes a disheartening tale, such as those above. Despite the expansion of solar installation, the city's energy demands seem poised to surpass the solar yield. This disconcerting disparity casts a shadow over the team's ambitions.

With sleeves rolled up, the team ventures deep into the night, engrossed in brainstorming sessions. Yet, they soon end up going in circles. Recognizing the need for clarity, the director makes a call to dismiss the team so they can rest and return with renewed creativity. With crushed and weary spirits, they go home uncertain about the fate of a project that had once shown so much promise.

Part 3: Energy Forecasting & Optimization

Part 3.1: Renewed Resolve: Crafting a Dynamic Energy Strategy for Aurora City's Future

With the dawn of a new day comes hours of deliberation, leading to a consensus that they will stay the course, but with a more powerful battery to bridge the gap and the traditional power plants will still be able to pick up any remaining slack. Overcoming that challenge renews the team's faith in the project. They are ready to move to the next stage of the process which is working out how the solar installation and battery will handle the gradual buildup in electrical power demand that will result as the city phases out electricity from traditional power plants. The team begins working on a machine learning model to take into



account the season, time of day in hours, and battery's charge level to create a dynamic strategy to handle the load of the city's electricity demands.

Part 3.2: Entangled Choices: Navigating Neural Network Dilemmas in Energy Prediction

Regrettably, the team encounters another impasse, this time stemming from a difference in opinion about which tool to use. On one side are the data scientists, who were in favor of employing a technology known as Long Short-Term Memory (LSTM) neural networks. They believe these networks held the key to comprehending the complex puzzle of electricity distribution patterns.

The data scientists argue that LSTMs are like seasoned detectives of time. They can piece together patterns across various time frames – whether it is the daily rhythm, changing seasons, or even the subtle dance between different time intervals. This skill makes LSTMs an ideal choice to predict electricity distribution, as it depends on a delicate balance of many variables, like the time of day, the battery's energy, and the electricity demand.

On the opposing side, the software developers advocate for Gated Recurrent Unit (GRU) neural networks. They see these networks as intelligent and efficient navigators of data. "GRUs are like streamlined drivers that ensured a smooth ride even through intricate datasets, helping to keep the project on track and within budget" explained an impassioned junior developer.

As spirited debates ensue, the team's director steps in, recognizing merit in both arguments. After careful reflection, the director leans towards the LSTM approach, appreciating its prowess in capturing time-driven intricacies. Believing that LSTMs could equip the team with superior insights for accurate electricity distribution forecasts, the director's decision was made. The software developers, albeit begrudgingly, collaborate with the data science team to construct and train the LSTM neural network.

Yet, just as discussions settled, a new voice emerged. Inspired by a recent tech seminar, one of the team's interns proposes harnessing the predictive power of Prophet from Facebook. The intern explains that Prophet is a forecasting tool that excels at predicting time-series data with multiple patterns and trends. Its ability to automatically detect seasonality and handle missing data makes it well-suited for predicting electricity



distribution patterns. The director, open to innovation, reweighs the options and ultimately embraces the intern's suggestion. With fresh hope, the team embarks on a new phase – navigating the uncharted waters of energy forecasting using a novel tool.

```
df = pd.read_csv(consumption_csv_file_path)
```

```
df
```

```
Datetime      AEP_MW
0      2004-12-31 01:00:00  13478.0
1      2004-12-31 02:00:00  12865.0
2      2004-12-31 03:00:00  12577.0
3      2004-12-31 04:00:00  12517.0
4      2004-12-31 05:00:00  12670.0
...      ...      ...
121268 2018-01-01 20:00:00  21089.0
121269 2018-01-01 21:00:00  20999.0
121270 2018-01-01 22:00:00  20820.0
121271 2018-01-01 23:00:00  20415.0
121272 2018-01-02 00:00:00  19993.0
121273 rows × 2 columns
```

```
consumption_df = pd.DataFrame()
```

```
consumption_df["ds"]=df["Datetime"]
```

```
consumption_df["y"]=df["AEP_MW"]*200
```

```
consumption_df["y"].max()
```

```
5139000.0
```

```
p_df = pd.read_csv(csv_file_path)
```




```
production_df = pd.DataFrame()
production_df["ds"] = p_df["date"]
production_df["y"] = p_df["kWh"]*1000
```

production_df

```
ds      y
0      2017/09/11 08:00:00 AM    1130.0
1      2017/09/11 09:00:00 AM    2340.0
2      2017/09/11 10:00:00 AM    3656.0
3      2017/09/11 11:00:00 AM    4577.0
4      2017/09/11 12:00:00 PM    6506.0
...     ...     ...
258418 2023/03/12 03:00:00 PM    201285.0
258419 2023/03/12 04:00:00 PM    162582.0
258420      2023/03/12 05:00:00 PM    107060.0
258421 2023/03/12 06:00:00 PM     43074.0
258422      2023/03/12 07:00:00 PM     1788.0
258423 rows × 2 columns
```

```
c_model = Prophet()
c_model.fit(consumption_df)
future = c_model.make_future_dataframe(periods=131424, freq="H", include_history=False)
```

future

21:26:27 - cmdstanpy - INFO - Chain [1] start processing

21:27:58 - cmdstanpy - INFO - Chain [1] done processing

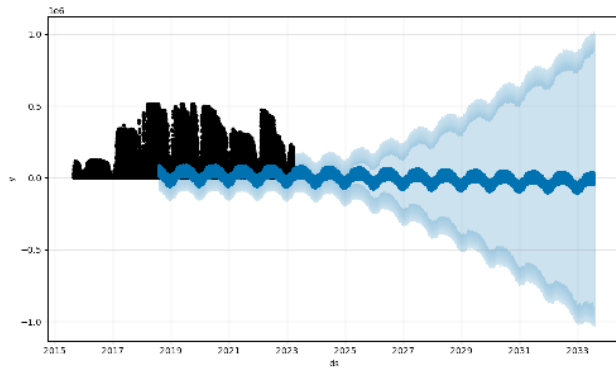
ds



```
0      2018-08-03 01:00:00
1      2018-08-03 02:00:00
2      2018-08-03 03:00:00
3      2018-08-03 04:00:00
4      2018-08-03 05:00:00
...
131419 2033-07-30 20:00:00
131420 2033-07-30 21:00:00
131421 2033-07-30 22:00:00
131422 2033-07-30 23:00:00
131423 2033-07-31 00:00:00
131424 rows × 1 columns
```

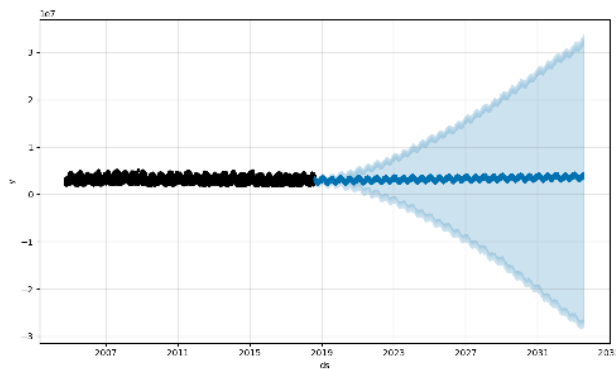
```
p_model = Prophet()
p_model.fit(production_df)
21:28:44 - cmdstanpy - INFO - Chain [1] start processing
21:29:58 - cmdstanpy - INFO - Chain [1] done processing
<prophet.forecaster.Prophet at 0x7ae6b0cceb0>
production_forecast=p_model.predict(future)
fig2 = p_model.plot(production_forecast)
```





```
forecast = c_model.predict(future)
```

```
fig1 = c_model.plot(forecast)
```



Supercharging Sustainability: Harnessing Superconductors for Renewable Energy Innovation

Part 1: Introduction

The looming danger of intensified climate change, driven by the relentless emission of greenhouse gases from fossil fuel-based electricity generation, underscores the urgent need to transition towards sustainable energy sources swiftly. The potential discovery of a room-temperature superconductor, LK-99, by South Korean researchers provides the opportunity for Superconducting Magnetic Energy Storage [SMES] which was invented in 1970. This technology is meant to effectively store energy through induction where electrons flow continuously within a superconducting coil wrapped around a core usually made from a niobium-titanium alloy embedded in a copper matrix. The primary benefits of SMES over other methods include nearly instantaneous power availability with a minimal time delay during charge and discharge, low power loss due to negligible electrical resistance, and enhanced reliability as its stationary components contribute to system stability. However, SMES systems can face increased complexity and operational cost due to their dependence on cryogenic cooling to maintain the superconducting state of the



superconducting coil. The chance of LK-99's use in such a system remains low due to current research indicating a limited critical current of 250 mA (the maximum current where $R=0$) and a constrained maximum magnetic field threshold for maintaining superconductivity.

The primary objective of this notebook is to identify the plausibility of creating a solar battery that, when combined with a solar panel installation or any other renewable energy source, can sustain the energy consumption of an ideal American city. In order to do this, the data analysis algorithms will be employed for prediction and optimization

Part 1.1 Aurora City: Pioneering Sustainability in a Vibrant Metropolis

Let's begin by imagining we are in Aurora City with a population of 1,024,144 people, and is a bustling metropolis known for its vibrant cultural scene, modern architecture, and diverse population. The city boasts a mix of historic neighborhoods and futuristic skyscrapers, offering a unique blend of old-world charm and contemporary innovation. With a strong emphasis on sustainability, Aurora City is home to lush parks, efficient public transportation, and a commitment to renewable energy sources. Its world-class universities, thriving tech industry, and rich culinary scene make it a hub of creativity and opportunity. In line with that commitment to sustainability, the city's government has pledged to move towards 100% solar electricity generation within the next 10 years.

```
!pip install prophet
```

```
!pip install cplex
```

```
!pip install stable_baselines3
```

```
import numpy as np
```

```
import pandas as pd
```

```
import random
```

```
from datetime import datetime, timedelta
```

```
import math
```

```
import tensorflow as tf
```

```
from tensorflow.keras.models import Sequential
```



```

from tensorflow.keras.layers import LSTM, Dense

from prophet import Prophet

from scipy.optimize import minimize

import pulp

import matplotlib.pyplot as plt

import statistics

import sys

from stable_baselines3 import PPO

from stable_baselines3.common.env_util import make_vec_env

import gymnasium as gym

from gymnasium import spaces

/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy
version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5

    warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}")

/opt/conda/lib/python3.10/site-packages/tensorflow_io/python/ops/__init__.py:98:
UserWarning: unable to load libtensorflow_io_plugins.so: unable to open file:
libtensorflow_io_plugins.so, from paths: ['/opt/conda/lib/python3.10/site-
packages/tensorflow_io/python/ops/libtensorflow_io_plugins.so']

caused by: ['/opt/conda/lib/python3.10/site-
packages/tensorflow_io/python/ops/libtensorflow_io_plugins.so: undefined symbol:
_ZN3tsl6StatusC1EN10tensorflow5error4CodeESt17basic_string_viewIcSt11char_traitsIcEENS_14SourceLocationE']

    warnings.warn(f"unable to load libtensorflow_io_plugins.so: {e}")

/opt/conda/lib/python3.10/site-packages/tensorflow_io/python/ops/__init__.py:104:
UserWarning: file system plugins are not loaded: unable to open file: libtensorflow_io.so,
from paths: ['/opt/conda/lib/python3.10/site-
packages/tensorflow_io/python/ops/libtensorflow_io.so']

caused by: ['/opt/conda/lib/python3.10/site-
packages/tensorflow_io/python/ops/libtensorflow_io.so: undefined symbol:
_ZTVN10tensorflow13GcsFileSystemE']

```



```
warnings.warn(f"file system plugins are not loaded: {e}")

def get_readings_for_date(csv_file_path, target_date,time_column,energy_column):
    # Read the CSV file into a DataFrame
    df = pd.read_csv(csv_file_path)
    df[time_column] = pd.to_datetime(df[time_column])
    df['filter_date'] = df[time_column].dt.date
    filtered_df = df[df['filter_date'] == target_date]

    return filtered_df[[time_column,energy_column]]

def generate_random_date(delim):

    start_date = datetime.strptime('2015-9-1', '%Y-%m-%d')
    end_date = datetime.strptime('2018-8-2', '%Y-%m-%d')

    days_difference = (end_date - start_date).days
    random_days = random.randint(0, days_difference)

    random_date = (start_date + timedelta(days=random_days)).date()
    return f"{random_date.year}{delim}{random_date.month}{delim}{random_date.day}"
target_date = pd.to_datetime(generate_random_date("/"))
month = target_date.month
target_date
Timestamp('2018-06-03 00:00:00')
```

Part 2: Data Cleaning & Analysis

Part 2.1: Energizing Change: The Quest for Sustainable Power in Aurora City



To pave the way for a sustainable energy transformation, the city government assembles a team of experts, Solar Nexus Taskforce (SNT), tasked with crafting a viable strategy. Their objective was to orchestrate the reduction of the city's conventional power plants while simultaneously bolstering the deployment of solar panels and SMES solar batteries.

With determination, SNT promptly swings into action. The first step on their journey is to gather 2-3 years worth of energy consumption data from the city's power grid and energy production data from an existing solar field that will be undergoing a substantial expansion in the coming months.

```
csv_file_path = '/kaggle/input/solar-energy-production/Solar_Energy_Production.csv' #  
Solar energy production data sourced from solar installations in Calgary, Alberta
```

```
def get_production(date):
```

```
    production_readings_for_date = get_readings_for_date(csv_file_path,  
date.date(), 'date', 'kWh').groupby('date')['kWh'].sum().reset_index()
```

```
    production_readings_for_date["kWh"] = production_readings_for_date["kWh"]*1000 #  
To account for a major scale-up in functionality
```

```
    return production_readings_for_date
```

```
production_readings_for_date = get_production(target_date)
```

```
consumption_csv_file_path = '/kaggle/input/hourly-energy-consumption/AEP_hourly.csv'
```

```
# Electrical energy consumption data from American Electric Power Company, Inc (AEP)  
which is among the nation's largest generators of electricity, owning nearly 38,000  
megawatts of generating capacity in the U.S and serving nearly 5 million customers.
```

```
def get_consumption(date):
```

```
    consumption_readings_for_date = get_readings_for_date(consumption_csv_file_path,  
date.date(), 'Datetime', 'AEP_MW').sort_values(by='Datetime')
```

```
consumption_readings_for_date["AEP_MW"] = consumption_readings_for_date["AEP_MW"]  
*200
```

```
# converts MW to kW and also scales down the data by a factor of 5 to make more  
representative of the demands of a city the size of Auroa City
```




```

# Set the 'datetime' column as the index
consumption_readings_for_date.set_index('Datetime', inplace=True)

return consumption_readings_for_date

consumption_readings_for_date = get_consumption(target_date)
production_readings_for_date.set_index("date", inplace=True)

# Adding in zeroes for the times not included with the assumption that those were times
when the sun was not up

new_row = {"kWh":0}

for date in consumption_readings_for_date.index:
    if date not in production_readings_for_date.index:
        production_readings_for_date.loc[date] = new_row

production_readings_for_date = production_readings_for_date.sort_values(by='date')
production_readings_for_date

kWh
date
2018-06-03 00:00:00 0.0
2018-06-03 01:00:00 0.0
2018-06-03 02:00:00 0.0
2018-06-03 03:00:00 0.0
2018-06-03 04:00:00 0.0
2018-06-03 05:00:00 5364.0
2018-06-03 06:00:00 89479.0
2018-06-03 07:00:00 243391.0
2018-06-03 08:00:00 506163.0

```



```

2018-06-03 09:00:00 840801.0
2018-06-03 10:00:00 1022944.0
2018-06-03 11:00:00 1124456.0
2018-06-03 12:00:00 1052887.0
2018-06-03 13:00:00 662972.0
2018-06-03 14:00:00 1063243.0
2018-06-03 15:00:00 1084414.0
2018-06-03 16:00:00 895397.0
2018-06-03 17:00:00 491990.0
2018-06-03 18:00:00 343648.0
2018-06-03 19:00:00 176110.0
2018-06-03 20:00:00 80177.0
2018-06-03 21:00:00 4680.0
2018-06-03 22:00:00 0.0
2018-06-03 23:00:00 0.0

```

```

energy_demand = ((consumption_readings_for_date['AEP_MW']-
production_readings_for_date["kWh"]).mean())/1e6

```

```

total = (consumption_readings_for_date['AEP_MW'].mean())/1e6

```

```

f"Energy Demand Unfilled by Solar Installation on {target_date.date()}:
{round(energy_demand,2)} GWh ({round(energy_demand/total,3)*100}% of demand)"

```

```

'Energy Demand Unfilled by Solar Installation on 2018-06-03: 2.36 GWh
(85.39999999999999% of demand)'

```

```

import matplotlib.pyplot as plt

```

```

# Plot datetime against energy consumption

```

```

plt.figure(figsize=(10, 6)) # Adjust the figure size as needed

```

```

plt.plot(consumption_readings_for_date.index, consumption_readings_for_date['AEP_MW'],
production_readings_for_date.index, production_readings_for_date["kWh"])

```



```
plt.xlabel('Time (mm-dd hh)')  
  
plt.ylabel('Power (kW)')  
  
plt.title(f'Electrical Power Consumption Over Time on {datetime.strftime(target_date,"%m-%d-%Y")}')  
  
plt.xticks(rotation=45) # Rotate x-axis labels for better visibility  
  
plt.tight_layout()  
  
plt.show()
```

Part 2.2: Unveiling Challenges: The City's Energy Quest Confronts Shadows

Amid caffeine-fueled calculations and simulations, a stark truth emerges from the data scientists' endeavors. A formidable challenge looms over the team. Within the conference room, a somber air settles as the lead data scientist projects slide after slide of graphs and statistics that each echoes a disheartening tale, such as those above. Despite the expansion of solar installation, the city's energy demands seem poised to surpass the solar yield. This disconcerting disparity casts a shadow over the team's ambitions.

With sleeves rolled up, the team ventures deep into the night, engrossed in brainstorming sessions. Yet, they soon end up going in circles. Recognizing the need for clarity, the director makes a call to dismiss the team so they can rest and return with renewed creativity. With crushed and weary spirits, they go home uncertain about the fate of a project that had once shown so much promise.

Part 3: Energy Forecasting & Optimization

Part 3.1: Renewed Resolve: Crafting a Dynamic Energy Strategy for Aurora City's Future

With the dawn of a new day comes hours of deliberation, leading to a consensus that they will stay the course, but with a more powerful battery to bridge the gap and the traditional power plants will still be able to pick up any remaining slack. Overcoming that challenge renews the team's faith in the project. They are ready to move to the next stage of the process which is working out how the solar installation and battery will handle the gradual buildup in electrical power demand that will result as the city phases out electricity from traditional power plants. The team begins working on a machine learning model to take into account the season, time of day in hours, and battery's charge level to create a dynamic strategy to handle the load of the city's electricity demands.



Part 3.2: Entangled Choices: Navigating Neural Network Dilemmas in Energy Prediction

Regrettably, the team encounters another impasse, this time stemming from a difference in opinion about which tool to use. On one side are the data scientists, who were in favor of employing a technology known as Long Short-Term Memory (LSTM) neural networks. They believe these networks held the key to comprehending the complex puzzle of electricity distribution patterns.

The data scientists argue that LSTMs are like seasoned detectives of time. They can piece together patterns across various time frames – whether it is the daily rhythm, changing seasons, or even the subtle dance between different time intervals. This skill makes LSTMs an ideal choice to predict electricity distribution, as it depends on a delicate balance of many variables, like the time of day, the battery's energy, and the electricity demand.

On the opposing side, the software developers advocate for Gated Recurrent Unit (GRU) neural networks. They see these networks as intelligent and efficient navigators of data. "GRUs are like streamlined drivers that ensured a smooth ride even through intricate datasets, helping to keep the project on track and within budget" explained an impassioned junior developer.

As spirited debates ensue, the team's director steps in, recognizing merit in both arguments. After careful reflection, the director leans towards the LSTM approach, appreciating its prowess in capturing time-driven intricacies. Believing that LSTMs could equip the team with superior insights for accurate electricity distribution forecasts, the director's decision was made. The software developers, albeit begrudgingly, collaborate with the data science team to construct and train the LSTM neural network.

Yet, just as discussions settled, a new voice emerged. Inspired by a recent tech seminar, one of the team's interns proposes harnessing the predictive power of Prophet from Facebook. The intern explains that Prophet is a forecasting tool that excels at predicting time-series data with multiple patterns and trends. Its ability to automatically detect seasonality and handle missing data makes it well-suited for predicting electricity distribution patterns. The director, open to innovation, reweighs the options and ultimately embraces the intern's suggestion. With fresh hope, the team embarks on a new phase – navigating the uncharted waters of energy forecasting using a novel tool.



```
df = pd.read_csv(consumption_csv_file_path)
```

```
df
```

```
Datetime    AEP_MW
```

```
0    2004-12-31 01:00:00  13478.0
```

```
1    2004-12-31 02:00:00  12865.0
```

```
2    2004-12-31 03:00:00  12577.0
```

```
3    2004-12-31 04:00:00  12517.0
```

```
4    2004-12-31 05:00:00  12670.0
```

```
...
```

```
121268 2018-01-01 20:00:00  21089.0
```

```
121269 2018-01-01 21:00:00  20999.0
```

```
121270 2018-01-01 22:00:00  20820.0
```

```
121271 2018-01-01 23:00:00  20415.0
```

```
121272 2018-01-02 00:00:00  19993.0
```

```
121273 rows × 2 columns
```

```
consumption_df = pd.DataFrame()
```

```
consumption_df["ds"] = df["Datetime"]
```

```
consumption_df["y"] = df["AEP_MW"] * 200
```

```
consumption_df["y"].max()
```

```
5139000.0
```

```
p_df = pd.read_csv(csv_file_path)
```

```
production_df = pd.DataFrame()
```

```
production_df["ds"] = p_df["date"]
```



```
production_df["y"] = p_df["kWh"]*1000
```

```
production_df
```

```
ds      y
0      2017/09/11 08:00:00 AM    1130.0
1      2017/09/11 09:00:00 AM    2340.0
2      2017/09/11 10:00:00 AM    3656.0
3      2017/09/11 11:00:00 AM    4577.0
4      2017/09/11 12:00:00 PM    6506.0
...      ...      ...
258418 2023/03/12 03:00:00 PM    201285.0
258419 2023/03/12 04:00:00 PM    162582.0
258420      2023/03/12 05:00:00 PM    107060.0
258421 2023/03/12 06:00:00 PM     43074.0
258422      2023/03/12 07:00:00 PM     1788.0
258423 rows × 2 columns
```

```
c_model = Prophet()
```

```
c_model.fit(consumption_df)
```

```
future = c_model.make_future_dataframe(periods=131424, freq="H", include_history=False)
```

```
future
```

```
21:26:27 - cmdstanpy - INFO - Chain [1] start processing
```

```
21:27:58 - cmdstanpy - INFO - Chain [1] done processing
```

```
ds
```

```
0      2018-08-03 01:00:00
1      2018-08-03 02:00:00
```



```
2      2018-08-03 03:00:00
3      2018-08-03 04:00:00
4      2018-08-03 05:00:00
...      ...
131419 2033-07-30 20:00:00
131420 2033-07-30 21:00:00
131421 2033-07-30 22:00:00
131422 2033-07-30 23:00:00
131423 2033-07-31 00:00:00
131424 rows × 1 columns
```

```
p_model = Prophet()
p_model.fit(production_df)
21:28:44 - cmdstanpy - INFO - Chain [1] start processing
21:29:58 - cmdstanpy - INFO - Chain [1] done processing
<prophet.forecaster.Prophet at 0x7ae6b0cceb0>
production_forecast=p_model.predict(future)
fig2 = p_model.plot(production_forecast)

forecast = c_model.predict(future)
fig1 = c_model.plot(forecast)
```

Part 3.3: Optimizing Power Flow: Bridging Solar, SMES, and the City's Grid

The team has finished the predictions for solar installation energy production and the city's energy demand for the next decade. Now they are one step closer to implementation. However, first, they need to use their predictions to determine the optimum battery energy capacity and power flow among the solar installation, the SMES, and the city's power



grid.

Mathematical Modeling Equation Setup

```
def LCC(batt_cap):  
    return  
(batt_cap*unit_cap_price)+((coeff_op_main*(batt_cap*unit_cap_price))*calendric_life)  
optimal_capacity = 0  
battery_capacity = optimal_capacity  
min_soc = 0.1  
max_soc = 0.98  
calendric_life = 10 # years  
cycle_life = 100000 #cycles  
coeff_op_main = 0.03  
unit_cap_price=0.169 # $/kWh  
cost_batt = battery_capacity*unit_cap_price  
cost_op_m = (coeff_op_main*cost_batt)*8760*calendric_life  
a,b = int(4e6), int(6e6) #Initial Interval [a,b]  
r = (1-math.sqrt(5))/2 # Goldent Ratio  
tolerance = 1e-8 # Tolerance for convergence  
  
while abs((b-a)/b)>tolerance:  
    a1 = b-r*(b-a)  
    a2 = a+r*(b-a)  
    if LCC(a1)>LCC(a2):  
        a=a1  
    else:
```



$b=a^2$

`optimal_capacity = (a+b)/2`

`f"Optimal Battery Capacity: {optimal_capacity/1e6} GWh"`

`'Optimal Battery Capacity: 9.236067850615498 GWh'`

`forecast_df=pd.DataFrame()`

`forecast_df["Datetime"] = future["ds"][43775:]`

`forecast_df["P_load"] = forecast["yhat"][43775:]`

`forecast_df["P_PV"] = production_forecast["yhat_upper"][43775:]`

`forecast_df=forecast_df.reset_index()`

`forecast_df`

	index	Datetime	P_load	P_PV
0	43775	2023-08-01 00:00:00	3.533828e+06	124483.833097
1	43776	2023-08-01 01:00:00	3.383443e+06	122641.989298
2	43777	2023-08-01 02:00:00	3.269510e+06	118251.866090
3	43778	2023-08-01 03:00:00	3.204024e+06	93385.176095
4	43779	2023-08-01 04:00:00	3.196615e+06	83719.773769
...
87644	131419	2033-07-30 20:00:00	4.121297e+06	903719.560193
87645	131420	2033-07-30 21:00:00	4.082765e+06	904116.676289
87646	131421	2033-07-30 22:00:00	3.977992e+06	926797.067400
87647	131422	2033-07-30 23:00:00	3.822352e+06	933604.000761
87648	131423	2033-07-31 00:00:00	3.649970e+06	951770.016103
87649	rows × 4 columns			

Part 4: Empowering the Future: Implementing the Vision



Edit with WPS Office

After several long nights of meticulous planning, the team transitioned from predictions to action, orchestrating a seamless integration of the solar installation, the SMES, and the city's power grid. The skyline transformed as advanced solar arrays and the sleek SMES installation emerged, showcasing Aurora City's commitment to a sustainable future.

```
def calc_power_from_power_plant(params,ind):  
    charging_power, discharging_power = params  
  
    power = forecast_df["P_load"][ind]-forecast_df["P_PV"][ind]+charging_power-  
    discharging_power  
  
    if power>0:  
        return power[0]  
    else:  
        return 0  
  
class PowerOptimizationEnv(gym.Env):  
    def __init__(self):  
        super(PowerOptimizationEnv, self).__init__()  
  
        # Define simulation parameters  
        #self.total_time_steps = len(forecast_df["Datetime"])  
        self.battery_capacity = optimal_capacity  
        self.current_time = 0  
        self.power_plant_power = 0  
  
        self.optimal_power_charge_list = []  
        self.optimal_power_discharge_list = []  
        self.minimal_power_from_power_plant_list = [sys.maxsize]  
        self.battery_capacity_list = []
```



```

# Define observation space (state)

self.observation_space = spaces.Box(low=0, high=20000000, shape=(3,),
dtype=np.float64)


# Define action space (charge and discharge)

self.action_space = spaces.Box(low=self.battery_capacity/-3,
high=self.battery_capacity/3, shape=(1,), dtype=np.float64) # Continuous action


# Other environment-specific parameters

def calc_reward(self):

    power_reduction = self.minimal_power_from_power_plant_list[-2]-
self.minimal_power_from_power_plant_list[-1]

    if self.battery_capacity<0:

        return sys.maxsize*-1

    else:

        return float(power_reduction+self.battery_capacity)


def step(self, action):

    # Simulate environment dynamics and calculate reward

    self.battery_capacity+=(action[0]*1e6)

    if action >= 0:

        self.power_plant_power = calc_power_from_power_plant([action*1e6,
0],self.current_time)

        self.optimal_power_charge_list.append(action[0]*1e6)

        self.optimal_power_discharge_list.append(0)

        self.minimal_power_from_power_plant_list.append(self.power_plant_power)

```



```

        self.battery_capacity_list.append(self.battery_capacity)

    else:

        self.power_plant_power = calc_power_from_power_plant([0, action],
self.current_time)

        self.optimal_power_charge_list.append(0)

        self.optimal_power_discharge_list.append(action[0]*1e6)

        self.minimal_power_from_power_plant_list.append(self.power_plant_power)

        self.battery_capacity_list.append(self.battery_capacity)


    # Update self.current_time and other environment variables
    self.current_time+=1

    # Return next_observation, reward, done, info
    next_observation = np.array([self.battery_capacity, self.current_time,
self.power_plant_power], dtype=np.float64)

    reward = self.calc_reward()

    if self.current_time%24 == 0 :

        done = True

    else:

        done = False

    info = {'step': self.current_time}


    if self.current_time>=87647:

        truncated = True

    else:

        truncated = False

```



```

return next_observation, reward, done, truncated, info

def reset(self, seed=None):
    info = {'step': self.current_time}
    self.current_time = 0

    return np.array([self.battery_capacity, self.current_time, self.power_plant_power],
dtype=np.float64).astype(float), info

env = PowerOptimizationEnv()
model = PPO("MlpPolicy", env, verbose=1)
model.learn(total_timesteps=87600)
test_env = PowerOptimizationEnv()
obs = test_env.reset()[0].astype(float)
for i in range(87649):
    action, _states = model.predict(obs)
    obs, rewards, dones, truncated, info = test_env.step(action)
results_df = pd.DataFrame()
results_df["Charging Power"] = test_env.optimal_power_charge_list
results_df["Discharging Power"] = test_env.optimal_power_discharge_list
results_df["Power Plant Power Output"] =
test_env.minimal_power_from_power_plant_list[1:]
results_df["Battery Capacity"] = test_env.battery_capacity_list
results_df

```

Charging Power	Discharging Power	Power Plant Power Output	Battery Capacity
0	1.228953e+06	0.000000e+00	4.638297e+06 1.046502e+07



```

1      4.944454e+05 0.000000e+00 3.755247e+06 1.095947e+07
2      0.000000e+00 -1.957454e+06      3.151260e+06 9.002013e+06
3      0.000000e+00 -8.641661e+05 3.110639e+06 8.137847e+06
4      1.273549e+06 0.000000e+00 4.386444e+06 9.411396e+06
...      ...      ...      ...      ...
87644 0.000000e+00 -4.985948e+05      3.217578e+06 -2.059518e+08
87645 1.900344e+06 0.000000e+00 5.078992e+06 -2.040514e+08
87646 0.000000e+00 -2.335028e+05      3.051195e+06 -2.042849e+08
87647 0.000000e+00 -6.824750e+05      2.888749e+06 -2.049674e+08
87648 0.000000e+00 -4.046795e+05      2.698201e+06 -2.053721e+08
87649 rows x 4 columns

```

```

percent_change=(results_df["Power Plant Power Output"][87648]-results_df["Power Plant
Power Output"][0])/results_df["Power Plant Power Output"][0]

```

```

if percent_change<0:

```

```

    print(f"Auroa City has experienced a {percent_change*-100:.2f}% decrease in power
plant power over 10 years")

```

```

elif percent_change>0:

```

```

    print(f"Auroa City has experienced a {percent_change*100:.2f}% increase in power plant
power over over 10 years")

```

```

else:

```

```

    print("No change")

```

Auroa City has experienced a 41.83% decrease in power plant power over 10 years

Part 4 Cont.

As the system went live, a triumphant moment arrived – the city had significantly decreased its dependence on nonrenewable energy, and the SMES battery stood ready to bridge the gap after the sun sets. This success radiated beyond energy, inspiring schools to weave the tale into curricula and motivating other cities to pursue their sustainable



dreams. Amidst challenges, Aurora City's story demonstrated that a cleaner, brighter tomorrow is achievable through collaboration, innovation, and resilience, marking a new chapter for the city and the end of our story.

