Styling

```
from IPython.core.display import HTML
HTML("""
<style>
.usecase-title, .usecase-duration, .usecase-section-header {
    padding-left: 15px;
    padding-bottom: 10px;
    padding-top: 10px;
    padding-right: 15px;
    background-color: #0f9295;
    color: #fff;
}

.usecase-title {
    font-size: 1.7em;
    font-weight: bold;
}

.usecase-authors, .usecase-level, .usecase-skill {
    padding-left: 15px;
    padding-bottom: 7px;
    padding-top: 7px;
    background-color: #baeaeb;
    font-size: 1.4em;
    color: #121212;
}

.usecase-level-skill  {
    display: flex;
}

.usecase-level, .usecase-skill {
    width: 50%;
}

.usecase-duration, .usecase-skill {
    text-align: right;
    padding-right: 15px;
    padding-bottom: 8px;
    font-size: 1.4em;
}

.usecase-section-header {
    font-weight: bold;
    font-size: 1.5em;
}
.usecase-subsection-header, .usecase-subsection-blurb {
    font-weight: bold;
    font-size: 1.2em;
    color: #121212;
}

.usecase-subsection-blurb {
    font-size: 1em;
    font-style: italic;
}
</style>
""")
```

Renewable Energy Optimization
Authored by: Sinan Kilci

Duration: 300 mins
Level: Intermediate
Pre-requisite Skills: Python

The main goal of this project is to use historical environmental data to improve the installation and operation of renewable energy sources like wind turbines and solar panels. The aim is to make energy production more efficient and sustainable by studying environmental factors that affect energy production. The results will help us understand the best places to put renewable energy sources and how to operate them more effectively, using detailed environmental information to boost efficiency and sustainability.

At the end of this use case I will:

- be able to perform explanatory analysis and interpret complex data sets.
- be upskilled in creating visual representation of data to communicate insights effectively.
- be proficient in data wrangling techniques.
- be capable to build and validate simple predictive models using historical data.

Introduction

Optimizing Renewable Energy Deployment Using Environmental Sensor Data

In recent years, the global need for renewable energy has been increased as we strive to reduce carbon emissions and reduce the effects of climate change. Renewable energy sources, such as wind turbines and solar panels, play a critical role in this transition. However, the efficiency and effectiveness of these installations are heavily influenced by environmental conditions. Placing a wind turbine in an area with inconsistent wind patterns or a solar panel where sunlight is obstructed can lead to weak performance and increased costs.

This study aims to address these challenges by leveraging detailed environmental data from microclimate and meshed sensor networks. By analyzing factors such as wind speed, solar radiation, temperature, and humidity, we can identify optimal locations and operational strategies for renewable energy installations. The goal is to maximize energy generation, improve cost-efficiency, and reduce environmental impact.

The datasets used in this study are gathered from City of Melbourne Open Data Platform (https://data.melbourne.vic.gov.au) and provide extensive insights into local environmental conditions in Melbourne City. They include:

- Weather Stations Data (ATMOS 41): Historical data collected by weather stations installed in Argyle Square, capturing wind speed, solar radiation, and atmospheric conditions. This information is crucial for assessing site suitability for renewable energy projects.
- Microclimate Sensors Data: Contains climate readings from sensors located within the city, updated every fifteen minutes. This dataset includes ambient air temperature, relative humidity, atmospheric pressure, wind speed and direction, gust wind speed, particulate matter 2.5, particulate matter 10, and noise. It is essential for understanding microclimate variations throughout the day.
- Microclimate Sensor Locations: Provides the historical location and description for each microclimate sensor device installed throughout the city. This data is vital for understanding the spatial distribution of sensors and any relocations that may affect historical data interpretation.
- Microclimate Sensor Readings: Offers environmental readings updated every hour in fifteen-minute increments. It includes data on ambient air temperature, relative humidity, barometric pressure, particulate matter 2.5, particulate matter 10, and average wind speed. This dataset also aligns with EPA Victoria's air quality data, helping correlate microclimate conditions with broader environmental insights.

Requesting the datasets from their sources (Download)

```
import requests

# URLs for the datasets
urls = {
    "microclimate_sensor_readings": "https://data.melbourne.vic.gov.au/api/explore/v2.1/c
    "microclimate_sensors_data": "https://data.melbourne.vic.gov.au/api/explore/v2.1/cata
    "meshed_sensor_type_1": "https://data.melbourne.vic.gov.au/api/explore/v2.1/catalog/d
}

# Function to download a dataset
def download_dataset(url, filename):
    response = requests.get(url)
    if response.status_code == 200:
        with open(filename, 'wb') as file:
            file.write(response.content)
        print(f"Downloaded {filename} successfully.")
    else:
        print(f"Failed to download {filename}. Status code: {response.status_code}")

# Download each dataset
for name, url in urls.items():
    download_dataset(url, f"{name}.csv")
```

```
Downloaded microclimate_sensor_readings.csv successfully.
Downloaded microclimate_sensors_data.csv successfully.
Downloaded meshed_sensor_type_1.csv successfully.
```

Loading datasets

```
import pandas as pd

microclimate_sensor_readings = pd.read_csv('microclimate_sensor_readings.csv')
microclimate_sensors_data = pd.read_csv('microclimate_sensors_data.csv')
meshed_sensor_type_1 = pd.read_csv('meshed_sensor_type_1.csv')
```

Converting time related columns to datetime

```
microclimate_sensor_readings['Local_Time'] = pd.to_datetime(microclimate_sensor_readings[
microclimate_sensors_data['Time'] = pd.to_datetime(microclimate_sensors_data['Time'], err
meshed_sensor_type_1['date_measure'] = pd.to_datetime(meshed_sensor_type_1['date_measure'
```

Check for missing values

```
print("Total rows in wrangled_microclimate_sensor_readings:", len(microclimate_sensor_rea
print("Total rows in wrangled_microclimate_sensors_data:", len(microclimate_sensors_data)
print("Total rows in wrangled_meshed_sensor_type_1:", len(meshed_sensor_type_1))

print("Missing Values in Microclimate Sensor Readings:\n", microclimate_sensor_readings.i
print("Missing Values in Microclimate Sensors Data:\n", microclimate_sensors_data.isnull(
print("Missing Values in Meshed Sensor Type 1:\n", meshed_sensor_type_1.isnull().sum())
```

```
Total rows in wrangled_microclimate_sensor_readings: 56
Total rows in wrangled_microclimate_sensors_data: 50338
Total rows in wrangled_meshed_sensor_type_1: 119724
Missing Values in Microclimate Sensor Readings:
 Local_Time      0
ID              0
Site_ID         0
Sensor_ID       0
Value           0
Type            0
Units           0
Gatewayhub_ID   0
Site_Status     0
dtype: int64
Missing Values in Microclimate Sensors Data:
 Device_id                 0
Time                      0
SensorLocation          803
LatLong                 803
MinimumWindDirection   6573
AverageWindDirection     20
MaximumWindDirection   6573
MinimumWindSpeed       6573
AverageWindSpeed         20
GustWindSpeed          6573
AirTemperature           20
RelativeHumidity         20
AtmosphericPressure      20
PM25                   4555
PM10                   4555
Noise                  4555
dtype: int64
Missing Values in Meshed Sensor Type 1:
 dev_id                   0
date_measure             0
RTC                   2677
battery               2088
solarPanel            2196
command               2129
solar                 2783
precipitation         2784
strikes               2784
windSpeed             2199
windDirection         2200
gustSpeed             2200
vapourPressure        2784
atmosphericPressure   2200
relativeHumidity      2200
airTemp               2783
Lat Long              2677
Sensor Name           2677
dtype: int64
```

Handling the missing values

```
# Handle missing values (drop or fill)

# Identify existing Device_id to SensorLocation and LatLong mappings
location_mapping = microclimate_sensors_data.dropna(subset=['SensorLocation']).set_index('D
latlong_mapping = microclimate_sensors_data.dropna(subset=['LatLong']).set_index('Device_ic

# Fill missing SensorLocation values based on Device_id
microclimate_sensors_data['SensorLocation'] = microclimate_sensors_data.apply(
    lambda row: location_mapping.get(row['Device_id'], row['SensorLocation']), axis=1
)

# Fill missing LatLong values based on Device_id
microclimate_sensors_data['LatLong'] = microclimate_sensors_data.apply(
    lambda row: latlong_mapping.get(row['Device_id'], row['LatLong']), axis=1
)

# Check if the missing values in SensorLocation and LatLong are filled
print("Missing Values in SensorLocation after filling:\n", microclimate_sensors_data['Senso
print("Missing Values in LatLong after filling:\n", microclimate_sensors_data['LatLong'].is

microclimate_sensors_data.dropna(inplace=True)
meshed_sensor_type_1.dropna(inplace=True)
```

```
     Missing Values in SensorLocation after filling:
      0
     Missing Values in LatLong after filling:
      0
```

Merge the Microclimate data into one dataset

```
# Merge datasets where applicable



# Feature Engineering (example: calculate the difference in readings over time)
# Calculate hourly changes for readings
# merged_data['hourly_change'] = merged_data.groupby('Site_ID')['Value'].diff()

# Save the wrangled datasets
microclimate_sensor_readings.to_csv('wrangled_microclimate_sensor_readings.csv', index=Fals
microclimate_sensors_data.to_csv('wrangled_microclimate_sensors_data.csv', index=False)
meshed_sensor_type_1.to_csv('wrangled_meshed_sensor_type_1.csv', index=False)
#merged_data.to_csv('merged_and_wrangled_data.csv', index=False)

print("Total rows in wrangled_microclimate_sensor_readings:", len(microclimate_sensor_readi
print("Total rows in wrangled_microclimate_sensors_data:", len(microclimate_sensors_data))
print("Total rows in wrangled_meshed_sensor_type_1:", len(meshed_sensor_type_1))
```

```
Total rows in wrangled_microclimate_sensor_readings: 56
Total rows in wrangled_microclimate_sensors_data: 39230
Total rows in wrangled_meshed_sensor_type_1: 116940
```

- I decided to remove the dataset "Microclimate Sensor"
- I'll work on the "microclimate_sensors_data" about missing values because 20% of data will be lost if I just dropna.