HW-1

February 28, 2025

0.0.1 Hripsime Soghomonyan - Bass Model Homework

1. 2024 innovation: https://time.com/7094854/sakuu-kavian/
2. I have chosen from 2024 innovations Sakuu's Kavian platform's 3D printing electrodes, and as a similar innovation from past graphite electrodes. Graphite electrodes are made by mixing graphite with binder and making it into solid blocks. This process requires heating at extremely high temperatures and toxic chemicals, which makes it harmful for the environment. 3D printed electrodes are made by layered printing technique, without using any liquid, which is faster, less energy consuming and more eco-friendly. Graphite electrodes are used in steel production, especially in electric arc and ladle furnaces, while 3D printed ones are used in lithium-ion battery production, which itself is used in the electric vehicle and other portable electronic devices' production.
3. Consumption volume of graphite electrodes worldwide 2014-2019: https://www.statista.com/statistics/1235825/graphite-electrode-consumption-volume-worldwide/

[2]:
```python
import numpy as np
import pandas as pd
import scipy.optimize as opt
import matplotlib.pyplot as plt
```

[3]:
```python
data = pd.read_excel('data/electrodes_data.xlsx')
print(data)
```

```
   Year  Volume
0  2014  1000.5
1  2015   949.5
2  2016   886.9
3  2017   856.4
4  2018   875.6
5  2019   892.0
```

[4]:
```python
years = data["Year"].values
volume = data["Volume"].values
```

[5]:
```python
#4
def bass_model(t, p, q, M):
```

```
    return (M * (p + q) ** 2 * np.exp(-(p + q) * t)) / (p + q * np.exp(-(p + q)␣
↪* t)) ** 2
```

[6]:
```
years -= years.min()
```

[7]:
```
params, _ = opt.curve_fit(bass_model, years, volume, p0=[0.03, 0.38, 16000])
p, q, M = params
print(f"Estimated Parameters: p={p:.4f}, q={q:.4f}, M={M:.2f}")
```

```
Estimated Parameters: p=0.0134, q=-0.0135, M=970.80
```

[8]:
```
# 5
def predict_diffusion(p, q, M, start_year=2020, end_year=2040):
    future_years = np.arange(start_year, end_year)
    adopters = np.zeros(len(future_years))
    for i in range(1, len(future_years)):
        adopters[i] = (p + q * (sum(adopters[:i]) / M)) * (M - sum(adopters[:
↪i]))
    return future_years, np.cumsum(adopters)

graphite_years, graphite_adoption = predict_diffusion(p, q, M)

graphite_predictions_df = pd.DataFrame({"Year": graphite_years,␣
↪"Predicted_Adopters": graphite_adoption})

print(graphite_predictions_df)
```

```
    Year  Predicted_Adopters
0   2020            0.000000
1   2021           13.004856
2   2022           25.662875
3   2023           37.987809
4   2024           49.992690
5   2025           61.689877
6   2026           73.091099
7   2027           84.207494
8   2028           95.049646
9   2029          105.627621
10  2030          115.950996
11  2031          126.028888
12  2032          135.869985
13  2033          145.482568
14  2034          154.874533
15  2035          164.053418
16  2036          173.026420
17  2037          181.800414
18  2038          190.381971
19  2039          198.777377
```

6. I am going to analyze the diffusion worldwide, as my data is about worldwide consumption, and not country specific. Country specific analysis is not compatible with the topic, as electrodes are mainly consumed during steel production, which is not country specific thing, therefore it needs global overview.

[10]:
```
# 7
def estimate_adopters_by_period(p, q, M, start_year=2020, end_year=2040):
    future_years = np.arange(start_year, end_year)
    new_adopters = np.zeros(len(future_years))
    cumulative_adopters = np.zeros(len(future_years))

    for i in range(1, len(future_years)):
        new_adopters[i] = (p + q * (cumulative_adopters[i - 1] / M)) * (M -␣
 ↪cumulative_adopters[i - 1])
        cumulative_adopters[i] = cumulative_adopters[i - 1] + new_adopters[i]

    return future_years, new_adopters, cumulative_adopters

adoption_years, adopters_per_period, cumulative_adopters =␣
 ↪estimate_adopters_by_period(p, q, M)

adopters_df = pd.DataFrame({
    "Year": adoption_years,
    "New_Adopters": adopters_per_period,
    "Cumulative_Adopters": cumulative_adopters
})

print(adopters_df)
```

```
    Year  New_Adopters  Cumulative_Adopters
0   2020      0.000000             0.000000
1   2021     13.004856            13.004856
2   2022     12.658019            25.662875
3   2023     12.324934            37.987809
4   2024     12.004881            49.992690
5   2025     11.697187            61.689877
6   2026     11.401222            73.091099
7   2027     11.116395            84.207494
8   2028     10.842152            95.049646
9   2029     10.577975           105.627621
10  2030     10.323375           115.950996
11  2031     10.077893           126.028888
12  2032      9.841097           135.869985
13  2033      9.612582           145.482568
14  2034      9.391965           154.874533
15  2035      9.178885           164.053418
16  2036      8.973002           173.026420
17  2037      8.773994           181.800414
```

18	2038	8.581557	190.381971
19	2039	8.395405	198.777377