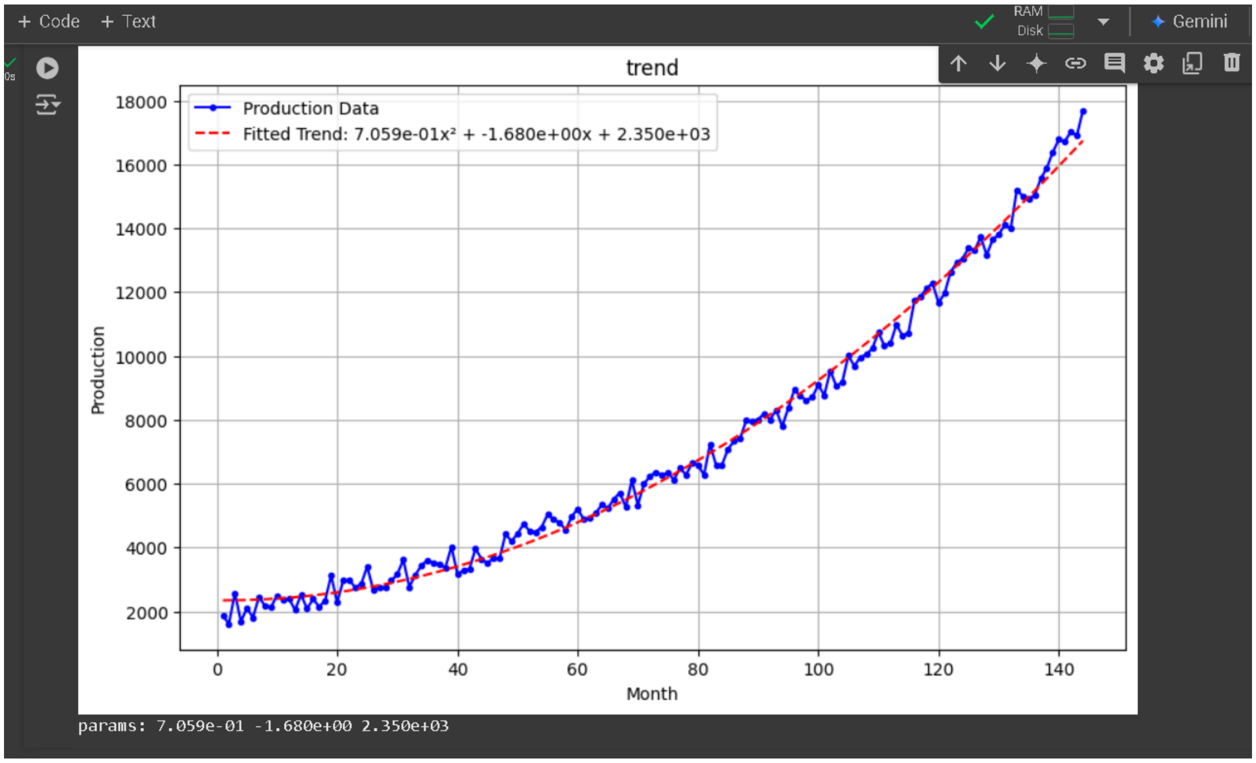
NIM: 2702259632

Nama: Vincentius Jonathan Tanujaya

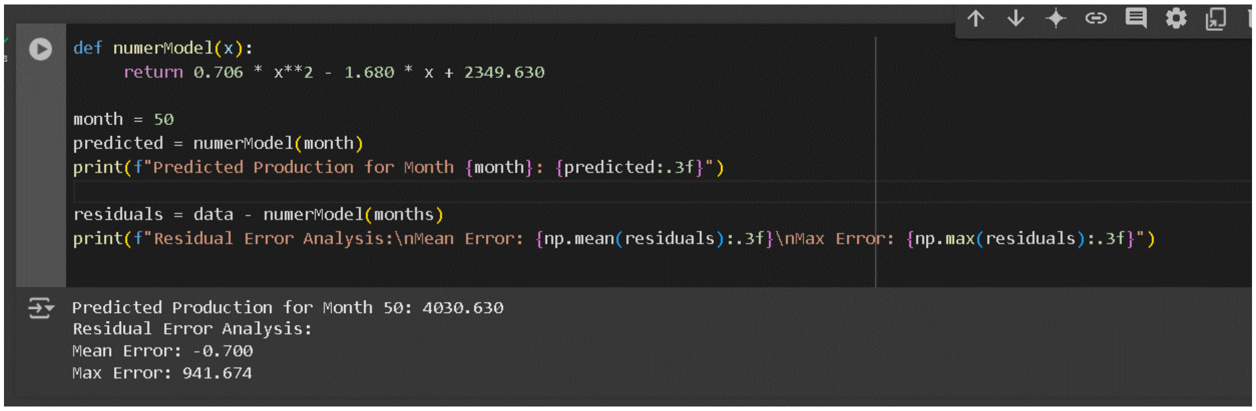
1. For this question I am tasked to find the trend of the data, but because I need to avoid using linear approach, I used a quadratic approach, because the use of quadratic model, can capture data that involves accelaration and deceleration over time, I also chose the quadratic model because it can model curves, showing upward trends that is seen in the data and also the quadratic model is flexible enough.

$$P(x) = ax2 + bx + c$$



```
params: 7.059e-01 -1.680e+00 2.350e+03
```

2. For this question, I converted the formula to its numerical form based on question no 1, and to measure the conversion is still accurate, I checked the mean error and max error of the conversion, and the mean error is just a -0.7 which is manageable, and the max error is 941 which is not a lot considering the dataset. Which means the model is accurate enough.



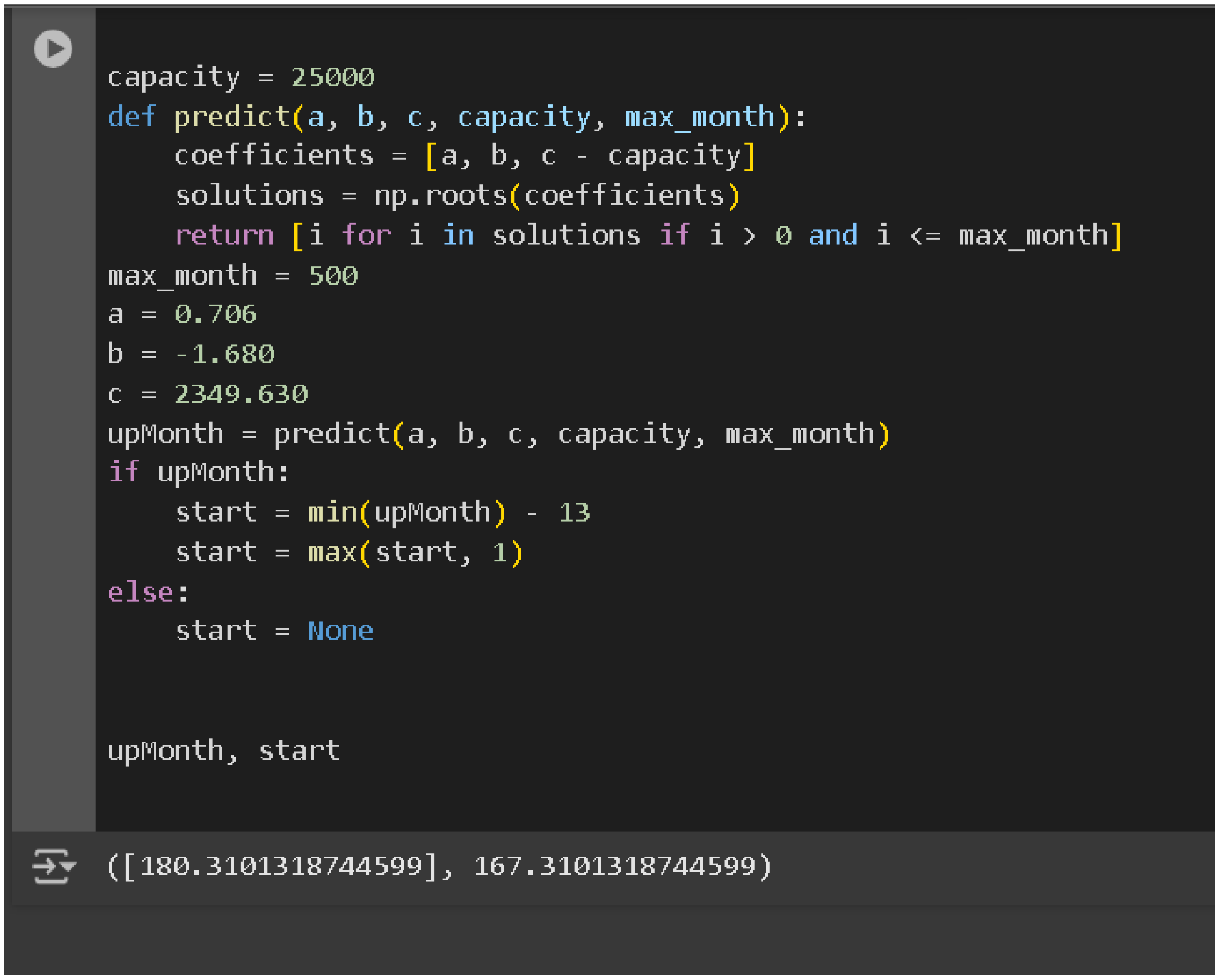
```
def numerModel(x):
    return 0.706 * x**2 - 1.680 * x + 2349.630

month = 50
predicted = numerModel(month)
print(f"Predicted Production for Month {month}: {predicted:.3f}")

residuals = data - numerModel(months)
print(f"Residual Error Analysis:\nMean Error: {np.mean(residuals):.3f}\nMax Error: {np.max(residuals):.3f}")
```

```
Predicted Production for Month 50: 4030.630
Residual Error Analysis:
Mean Error: -0.700
Max Error: 941.674
```

3. For this question, we need to check when does production for a month exceeds the capacity of 25000, and when does the EGIER need to start production before the overflow, for the function below we use root function , and see that when does 1 month exceed it, and see at what month that It starts to overflow, and from that month subtract 13 from that because EGIER needs 13 months to start ware house production. And from the code, below, we can see that it starts to overflow coming into month 181, so EGIER needs to start production for a new warehouse by 181-13 which is month 168.



```
capacity = 25000
def predict(a, b, c, capacity, max_month):
    coefficients = [a, b, c - capacity]
    solutions = np.roots(coefficients)
    return [i for i in solutions if i > 0 and i <= max_month]
max_month = 500
a = 0.706
b = -1.680
c = 2349.630
upMonth = predict(a, b, c, capacity, max_month)
if upMonth:
    start = min(upMonth) - 13
    start = max(start, 1)
else:
    start = None


upMonth, start
```

```
([180.3101318744599], 167.3101318744599)
```

4.link collab

https://colab.research.google.com/drive/1NB2KydoGmn7ZaE-mz1x3i5ILnT1ySbaq?usp=sharing