
31/03/2024 - Statistics for Data Science

Activity Structure: Each activity is designed for 30 minutes:

- 10 minutes: Learning content/research
- 10 minutes: Python coding
- 10 minutes: Writing a 1-4 page MS Word output document

Activity 1: Understanding the Sigmoid Function

Duration: 30 minutes
Objective: Learn how the sigmoid function turns raw scores into probabilities using **e** (**Euler's number ≈ 2.718**) and apply it to classify outcomes.

Explanation:
The sigmoid function is a mathematical tool used in data science to convert any number (called a "score" or z) into a probability between 0 and 1. This is useful for binary classification, like deciding if someone will repay a loan (Class 1) or default (Class 0). The function uses e, a special number that helps "squish" values into the 0-1 range.

For example, when z = 0, the probability is exactly 0.5 (a coin flip), and when z gets larger (like 5), the probability approaches 1.

- 10-Minute Learning: Research the sigmoid function and its formula:
  Probability = 1 / (1 + e^(-z))
  Understand that if the probability is 0.5 or higher, we predict Class 1; otherwise, Class 0. Explore why e is used (it's from exponential growth) and how it ensures probabilities stay between 0 and 1.
- 10-Minute Coding: Write Python code to calculate probabilities for z = 0 (using e^0) and z = 5 (using e^5).
- 10-Minute Writing: Document your findings in Word, explaining what the sigmoid does, the results for z = 0 and z = 5, and why the output makes sense.

Math and Python (End of Activity):

- Formula: Probability = 1 / (1 + e^(-z))
  - For z = 0: e^0 = 1, so Probability = 1 / (1 + 1) = 0.5
  - For z = 5: e^(-5) ≈ 0.0067, so Probability = 1 / (1 + 0.0067) ≈ 0.993

Python Code:

```python
import math

def sigmoid(z):
    return 1 / (1 + math.exp(-z))

for z in [0, 5]:
    p = sigmoid(z)
    print(f"Score {z} -> Probability {p:.3f} -> Class {1 if p >= 0.5 else 0}")
```

Output:
Score 0 -> Probability 0.500 -> Class 1
Score 5 -> Probability 0.993 -> Class 1

Activity 2: Loan Default Prediction

Duration: 30 minutes
Objective: Use a logistic model to predict loan default probability based on balance, focusing on logit = 0 and logit = 5.

Explanation
In real-world data science, we often predict outcomes like loan defaults using a "logit" score, which is a linear combination of features (e.g., loan balance).

Here, we use the formula logit = -10 + 0.005 * balance to calculate a score, then pass it through the sigmoid function to get a probability. A balance of £2000 gives logit = 0 (50% chance of default), while a higher balance like £3000 gives logit = 5 (very high chance). This shows how features (balance) influence predictions.

- 10-Minute Learning: Study the logit formula:
  logit = -10 + 0.005 * balance
  Research how this connects to the sigmoid function and why a positive coefficient (0.005) increases risk as balance grows.
- 10-Minute Coding: Write Python code to compute default probabilities for balances of £2000 and £3000.
- 10-Minute Writing: In Word, explain the logit formula, report the probabilities, and interpret what they mean for loan risk.

Math and Python (End of Activity):

- **Formula:**
  logit = -10 + 0.005 * balance
  Probability = 1 / (1 + e^(-logit))
    - For balance = £2000: logit = -10 + 0.005 * 2000 = 0
      Probability = 1 / (1 + e^0) = 1 / (1 + 1) = 0.5
    - For balance = £3000: logit = -10 + 0.005 * 3000 = 5
      Probability = 1 / (1 + e^(-5)) ≈ 1 / (1 + 0.0067) ≈ 0.993

Python Code:

```
python

import math

balance = 2000
logit = -10 + 0.005 * balance
p = 1 / (1 + math.exp(-logit))
print(f"Balance £{balance} -> Default Probability: {p:.3f}")

balance = 3000
logit = -10 + 0.005 * balance
p = 1 / (1 + math.exp(-logit))
print(f"Balance £{balance} -> Default Probability: {p:.3f}")
```

Output:
Balance £2000 -> Default Probability: 0.500
Balance £3000 -> Default Probability: 0.993

Summary

- **Sigmoid Function: Turns scores into probabilities using e^x.**
- **Loan Prediction: Uses logit scores to estimate real-world risks like defaults.**
- **Next Steps: Try this with other scenarios (e.g., customer churn).**
