Set up Dependencies

In [ ]:
```
import numpy as np
from scipy.stats import linregress
from scipy.optimize import curve_fit
import sklearn as sk
```

Input Data

Source: https://ir.netflix.net/financials/sec-filings/default.aspx

- 2025 10-K Annual Report
- 2023 10-K Annual Report

In [26]:
```
YEARS = np.array([0, 1, 2, 3, 4])
LATAM = np.array([7.45, 7.73, 8.48, 8.66, 8.24])
UCAN = np.array([13.32, 14.56, 15.86, 16.28, 17.20])
EMEA = np.array([10.72, 11.63, 10.99, 10.87, 10.96])
APAC = np.array([9.12, 9.56, 8.50, 7.64, 7.29])
```

Set up handlers

In [27]:
```
def exponential_func(x, a, b):
    return a * np.exp(b * x)


def fitted_exponential_func(popt):
    a_fit, b_fit = popt
    return lambda x: a_fit * np.exp(b_fit * x)


def fitted_exponential_func_str(popt):
    a_fit, b_fit = popt
    return f"y={a_fit:.4f}*e^({b_fit:.4f}x)"


def fitted_poly_func_str(model: np.poly1d):
    a_fit, b_fit, c_fit = model.coefficients
    return f"y={a_fit:.4f}x^2+{b_fit:.4f}x+{c_fit:.4f}"


def fitted_lin_func_str(result):
    return f"y={result.slope:.4f}x+{result.intercept:.4f}"
```

```
def r_squared_exp(popt, y_data, f=exponential_func):
    residuals = y_data - f(YEARS, *popt)
    ss_res = np.sum(residuals**2)
    ss_tot = np.sum((y_data-np.mean(y_data))**2)

    r_squared = 1 - (ss_res / ss_tot)
    return r_squared


def test_exp(y_data):
    popt, _pcov = curve_fit(exponential_func, YEARS, y_data)
    r_squared = r_squared_exp(popt, y_data)
    return r_squared, fitted_exponential_func_str(popt)


def test_poly(y_data):
    model = np.poly1d(np.polyfit(YEARS, y_data, 2))
    r_squared = sk.metrics.r2_score(y_data, model(YEARS))
    return r_squared, fitted_poly_func_str(model)


def test_lin(y_data):
    result = linregress(YEARS, y_data, alternative='greater')
    r_squared = np.square(result.rvalue)
    return r_squared, fitted_lin_func_str(result)


def test(y_data, verbose=False):
    regressions = dict()

    r1, eq1 = test_lin(y_data)
    regressions[eq1] = r1
    r2, eq2 = test_poly(y_data)
    regressions[eq2] = r2
    r3, eq3 = test_exp(y_data)
    regressions[eq3] = r3

    if verbose:
        print(regressions)

    return max(regressions.items(), key=lambda x: x[1])
```

Procedure

We compare linear, exponential, and quadratic regression outputs and pick which performs the best in context, factoring in R^2 value and how the parameter has performed in the past.

Line one of the output is a dictionary of regression outputs mapped to their corresponding R^2 values. Line two of the output is the regression output with the

highest R^2 output.

> R^2 is by no means a definitive metric of the fit of each model. Outputs are analyzed in context. Sometimes, the best model will not have the greater R^2 value.

Note

Each regression output's parameters has a standard error. The regression outputs of this study assume regression parameters under a 95% Confidence Interval are valid ($n = 5$, $df = 2$).

In [28]:
```
test(UCAN, True)
```

```
{'y=0.9480x+13.5480': np.float64(0.9717483824089135), 'y=-0.1086x^2+1.3
823x+13.3309': 0.9895925499364828, 'y=13.6279*e^(0.0607x)': np.float64(
0.9611083110345044)}
```

Out[28]:
```
('y=-0.1086x^2+1.3823x+13.3309', 0.9895925499364828)
```

In [29]:
```
test(EMEA, True)
```

```
{'y=-0.0280x+11.0900': np.float64(0.01606162419077287), 'y=-0.0800x^2+
0.2920x+10.9300': 0.19962304351389049, 'y=11.0895*e^(-0.0025x)': np.flo
at64(0.015899942460122718)}
```

Out[29]:
```
('y=-0.0800x^2+0.2920x+10.9300', 0.19962304351389049)
```

In [30]:
```
test(LATAM, True)
```

```
{'y=0.2510x+7.6100': np.float64(0.6079534488748214), 'y=-0.1407x^2+0.81
39x+7.3286': 0.8754556132098883, 'y=7.6299*e^(0.0302x)': np.float64(0.5
932595369360112)}
```

Out[30]:
```
('y=-0.1407x^2+0.8139x+7.3286', 0.8754556132098883)
```

In [31]:
```
test(APAC, True)
```

```
{'y=-0.5580x+9.5380': np.float64(0.8458036335187757), 'y=-0.0986x^2+-0.
1637x+9.3409': 0.8827550665606991, 'y=9.5525*e^(-0.0650x)': np.float64(
0.830434683778757)}
```

Out[31]:
```
('y=-0.0986x^2+-0.1637x+9.3409', 0.8827550665606991)
```

Results

Parameters are included in Confidence Interval ($\alpha = 0.05$)

UCAN

$$y = 12.9279 * e^{(0.0607x)}$$

EMEA

$$y = 0.0280x + 11.090$$

LATAM

$$y = -0.1207x^2 + 0.9139x + 6.7286$$

APAC

$$y = -0.5580x + 10.180$$