DD and PD Calculation Using Accounting Data (Book Values)

This notebook walks through the `dd_pd_accounting.py` script step by step.
We will:

1. Install dependencies

2. Set up imports and file paths

3. Load and inspect the accounting data

4. Standardize column names, extract years

5. Load and merge market cap and equity volatility

6. Define the Accounting-Based Merton Solver
7. Run the Accounting-Based Solver and Compute DDa/PDa
8. Compute Distance to Default (DDa) and Probability of Default (PDa)

9. Export results and append diagnostics to log

10. Summarize next steps

1. Install dependencies

2. Imports and File Paths

Import libraries and define all paths relative to the project root.

3. Load and Inspect Accounting Data

- Read `Book2_clean.csv` which must contain `total_assets` and `debt_total`.

- Display row and unique `(instrument, year)` counts.

4. Standardize Column Names & Extract Year

- Lowercase and replace spaces/dashes with underscores

- If `date` exists, parse it and extract year; otherwise ensure `year` is integer.

5. Load and Merge Market Cap & Equity Volatility

- Load annual market cap, standardize tickers, merge on `(symbol, Year)`
- Load equity volatility, standardize, merge on `(ticker_prefix, Year)`

6. Define the Accounting-Based Merton Solver

We treat **net equity** $E = A - F$ (assets minus debt) as a call option on the firm's assets. Given:

- A: total assets (book value)
- F: total debt (book value)
- σ_E: observed equity volatility
- r_f: risk-free rate
- T: time horizon (1 year)

we solve for:

- V: total asset value
- σ_V: asset volatility

by enforcing the two Merton equations:

1. **Option-pricing relation**

$$V\Phi(d_1) - Fe^{-r_f T}\Phi(d_2) - (A - F) = 0$$

2. **Volatility link**

$$\sigma_E - \frac{V}{(A-F)}\Phi(d_1)\sigma_V = 0$$

with

$$d_1 = \frac{\ln(V/F) + (r_f + \frac{1}{2}\sigma_V^2)T}{\sigma_V\sqrt{T}}, \quad d_2 = d_1 - \sigma_V\sqrt{T}$$

We will use `scipy.optimize.root` to find (V, σ_V) that makes both expressions zero, similarly as the calculation using market data

7. Run the Accounting-Based Solver and Compute DDa/PDa

In this step we will:

1. Apply `merton_solver_accounting` to every row of `df` to get
   - `asset_value` (V)
   - `asset_vol` (_V)
   - `merton_status` (convergence flag)
   - `dd_pd_tag` (e.g. “no_debt”)
2. Compute **Distance to Default** (DDa):

$$\text{DDa} = \frac{\ln(V/F)\ +\ (0 - \frac{1}{2}\,\sigma_V^2)\,T}{\sigma_V\,\sqrt{T}}$$

3. Compute **Probability of Default** (PDa):

$$\text{PDa} = \Phi(-\text{DDa})$$

4. Set both to `NaN` when `dd_pd_tag == 'no_debt'`.

8. Export Results & Append Diagnostics

In this final step we will:

1. **Save** the full DataFrame (including `DDa` and `PDa`) to CSV at `output_fp`.
2. **Append** a diagnostics summary to the accounting log file (`log_fp`), including:
   - Total rows processed
   - Solver status counts
   - Summary statistics for `DDa` and `PDa`
   - Counts of missing or failed estimates

9. Summary and Next Steps

What we’ve accomplished
- Loaded and cleaned the accounting data
- Merged in market caps, equity volatilities, and risk-free rates
- Defined and ran an accounting-based Merton solver
- Computed annual **DDa** and **PDa**
- Exported results and logged diagnostics

Next steps

1. **Review the log file** (`dd_pd_accounting_log.txt`) for any convergence warnings or missing inputs.
2. **Compare** accounting-based metrics (`DDa/PDa`) with market-based (`DDm/PDm`) to identify discrepancies.
3. **Visualize** the distributions of `DDa` and `PDa` across firms and years.
4. **Incorporate** these default-risk measures into your credit models or presentations for Professor Abel.