Take-Home Assignment: Year-Over-Year Growth Analysis for Internet Providers

Overview

You will analyze year-over-year growth opportunities for internet service providers (ISPs) in a simplified single-state market. This assignment involves data manipulation, strategic prioritization, capacity-constrained allocation, and time-series analysis using Python.

Time Estimate: 4-6 hours

Background

Geographic Definitions

For candidates unfamiliar with U.S. geography:

- **Location ID**: A specific serviceable location or address where internet service can be provided (e.g., a home, business, or apartment building). Think of this as an individual service point.

- **Census Block (block_geoid)**: A statistical geographic area defined by the U.S. Census Bureau, typically containing multiple locations. Census blocks are the smallest geographic unit for which census data is collected, roughly equivalent to a city block in urban areas but can be larger in rural areas.

In this analysis, you'll work with locations nested within census blocks, where providers may have varying levels of presence across both geographic levels.

Problem Statement

You are tasked with building a **multi-year expansion forecast model** that determines:

1. Which locations each provider should prioritize for network upgrades or new builds
2. How many locations each provider can build per year given capacity constraints

3. A year-over-year breakdown of growth by provider and growth type

---

Data Structure

You will work with a dataset containing the following key fields:

- location_id: Unique identifier for each serviceable location

- block_geoid: Census block identifier (locations roll up to blocks)

- provider_id: Unique identifier for each internet service provider

- technology: Integer code representing the technology type (see below)

- max_advertised_download_speed: Maximum download speed in Mbps

- max_advertised_upload_speed: Maximum upload speed in Mbps

Technology Codes

```
TECH_MAPPING = {

    10: "Copper",

    40: "Cable",

    50: "Fiber",

}
```

We are only analyzing **wired technologies** for this exercise (codes 10, 40, 50).

Technology Classification

Technologies must be classified based on both the technology type and download speed using the following logic:

```
def calculate_technology_classification(df):
    """

    Classifies technologies into: Slow Copper, Fast Copper, Slow Cable,

    Fast Cable, Fiber, or Other



    Classification Rules:

    - Technology 10 (Copper) with download speed < 50 Mbps → "Slow Copper"

    - Technology 10 (Copper) with download speed ≥ 50 Mbps → "Fast Copper"

    - Technology 40 (Cable) with download speed < 500 Mbps → "Slow Cable"

    - Technology 40 (Cable) with download speed ≥ 500 Mbps → "Fast Cable"

    - Technology 50 → "Fiber"

    - All others → "Other"

    """

    # Your implementation here
```

Gigabit-Capable Provider Definition

A **gigabit-capable provider** is defined as a provider that delivers internet service via:

- **Fiber** (technology 50), OR
- **Cable with speeds ≥ 500 Mbps** (technology 40 with download speed ≥ 500, i.e., "Fast Cable")

Market Structure Constraints

Location-Level Capacity Constraint

Each location can support:

- **Any number** of slower technology providers (Slow Copper, Fast Copper, Slow Cable)

- **Maximum 2 gigabit-capable providers** (Fiber or Fast Cable)

Available spots for new gigabit builds at a location:

Available Spots = 2 - (Current Number of Gigabit-Capable Providers)

Examples:

- Location with 0 gigabit providers → 2 available spots
- Location with 1 gigabit provider → 1 available spot
- Location with 2 gigabit providers → 0 available spots (saturated)

Important: If a location already has more than 2 gigabit providers in the dataset, it is considered "grandfathered" and does not need to change.

Growth Opportunity Types

There are **two types** of growth opportunities:

1. **Upgrades** (Higher Priority)

A provider can upgrade their technology at a location where they currently have presence with:

- **Slow Copper** (technology 10, speed < 50 Mbps), OR
- **Slow Cable** (technology 40, speed < 500 Mbps)

The provider can upgrade these technologies to:

- **Fiber** (technology 50), OR
- **Fast Cable** (technology 40, speed ≥ 500 Mbps)

2. **Edge-Out Near-Net** (Lower Priority)

A provider can build at a location where:

- They have **no current presence** at that specific location
- BUT they **do have presence** somewhere else within the same census block

---

Prioritization Logic

Step 1: Determine Eligible Contenders

For each location:

1. **Upgrade Contenders**: Providers with Slow Copper or Slow Cable at that location
2. **Edge-Out Near-Net Contenders**: Providers present elsewhere in the census block but not at this location

Step 2: Priority Ranking

Upgrades always have priority over Edge-Out Near-Net builds.

Within each growth type, rank contenders by:

1. **Existing presence at the location** (Upgrades only - this is implicit since upgrades require presence)
2. **Number of gigabit-capable locations** the provider has within the census block (descending)
3. **Total number of locations** the provider has within the census block (descending)

This ensures:

- Providers already at the location get first priority (upgrades)
- Among edge-out contenders, providers with larger gigabit footprints get priority
- Ties are broken by general provider size in the census block

---

Scoring Methodology

Once you have determined the eligible contenders and their priorities for each location, calculate a **composite score** for each location to determine the order in which locations should be built across years.

The composite score has **three equally-weighted components**:

1. Initial Competition Score (Location-Level)

Competition Score = 1 - (Number of Existing Gigabit Providers / 2)

- Location with 0 gigabit providers: Score = 1.0
- Location with 1 gigabit provider: Score = 0.5
- Location with 2+ gigabit providers: Score = 0.0

2. Gigabit Coverage Score (Census Block-Level, Provider-Specific)

GB Coverage Score = (Provider's Gigabit Locations in Block) / (Total Locations in Block)

After calculating for each provider-block combination, **normalize to 0-1 across the entire dataset**.

This rewards providers who already have gigabit presence in the census block (momentum/clustering effect).

3. Market Size Score (Census Block-Level)

Market Size Score = Total Unique Locations in Census Block

Normalize to 0-1 across the entire dataset.

This rewards larger markets.

Composite Score

Final Score = (Competition Score + Normalized GB Coverage Score + Normalized Market Size Score) / 3

Locations should be ranked in **descending order** by this final score.

Capacity Constraints

Provider-Level Annual Capacity

Each provider can build at most **4% of their existing unique footprint** per year.

Provider Annual Capacity = 0.04 × (Provider's Existing Unique Location Count at Start of Year)

Important:

- The footprint is the provider's current presence (all locations they serve), not their gigabit-capable presence.

- Capacity is calculated **at the start of each year** based on the provider's existing footprint (prior to that year's builds).
- As providers expand their footprint through builds, their capacity **increases in subsequent years**.

State-Level Annual Capacity

The total number of new gigabit builds across all providers in the state cannot exceed **2% of total unique locations** per year.

State Annual Capacity = 0.02 × (Total Unique Locations in State at Start of Year)

Important:

- Capacity is calculated **at the start of each year** based on the total unique locations in the state.
- State capacity remains constant year-over-year (2% of total unique locations), providing a stable ceiling on total annual builds.

Year-Over-Year Allocation Process

Annual Build Process

For each year:

1. **Rank all eligible locations** by the composite score (descending)
2. **Iterate through locations** from highest to lowest score
3. For each location:
    - Identify the highest-priority contender (upgrades first, then edge-out)
    - Check if the provider has remaining capacity for the year
    - Check if the state has remaining capacity for the year
    - If both checks pass:
        - Allocate the build to that provider
        - Decrement provider capacity by 1
        - Decrement state capacity by 1
        - Mark the location as having one additional gigabit provider
    - If the provider is at capacity, move to the next eligible contender for that location
    - If the state is at capacity, stop processing for the year
4. **Update the state** for the next year:
    - Provider footprints increase based on completed builds (both upgrades and edge-out builds add to footprint)

    - Locations that received builds now have increased gigabit provider counts
    - **Recalculate capacities** at the start of the next year based on the updated footprints:
        - Provider capacities: 4% of each provider's expanded footprint (has grown from prior year)
        - State capacity: 2% of total unique locations (remains constant)
5. **Repeat** for subsequent years until no more builds are possible

Note: Because provider capacities are based on prior footprints that grow with each year's builds, provider capacities increase over time, enabling accelerating growth potential.

Deliverables

Please provide the following:

1. Python Code

- Well-commented, production-quality Python code (preferably using **pandas** or **polars**) or a Jupyter notebook with functions to organize your code
- Modular functions with clear docstrings

2. Year-Over-Year Summary Table

A summary table showing for each year and provider:

year	provider_id	upgrade_builds	edge_out_builds	total_builds
2026	PROV_077	150	200	350
2026	PROV_425	120	180	300

...

3. Analysis Report

A brief written summary (a sentence on each of the following) addressing:

- Which providers are growing the fastest and why?
- Which growth type (upgrades vs. edge-out) dominates in early years vs. later years?
- What happens to the state-level and provider-level capacity constraints over time?

Evaluation Criteria

You will be evaluated on:

1. **Correctness**: Does your code implement the specified logic accurately?
2. **Code Quality**: Is your code clean, readable, and well-organized?
3. **Analytical Thinking**: Do you understand the business logic and constraints?
4. **Communication**: Can you clearly explain your approach and findings?

Assumptions and Clarifications

- Assume the dataset provided contains all locations in the state
- Each location has at least one provider serving it (100% coverage by at least one provider)
- However, providers may serve only a subset of locations within a census block where they have presence
- This creates natural edge-out opportunities: providers can expand to new locations within blocks where they already serve some locations
- If a location has no gigabit-capable providers and no eligible contenders (upgrades or edge-out), it remains unserved
- You may assume the data is clean (no nulls, valid technology codes, etc.) unless otherwise noted
- For tie-breaking in scoring, you may use any consistent method (e.g., provider_id order)
- You can simulate for 5-10 years or until no more builds are possible
- Round capacity calculations to the nearest integer (round down for conservative estimates)

Submission Instructions

Please submit:

1. Your Python code (`.py` files or a Jupyter notebook)
2. The year-over-year summary table (CSV format)
3. A README with instructions on how to run your code

Sample Data Generation (Optional Helper)

If you'd like to test your code logic, here's a function to generate sample data. This version is designed to create both upgrade and edge-out opportunities:

```
import numpy as np

import pandas as pd


def generate_sample_data(n_blocks=1000, n_providers=15, locations_per_block=100):

    """

    Generates sample internet provider data for testing.

    Creates edge-out opportunities by:

    1. Assigning providers to blocks first

    2. Then assigning providers to only a subset of locations within each block

    3. This creates the pattern: provider has presence in block but not at all locations

    Returns a DataFrame with columns:

    - location_id

    - block_geoid

    - provider_id

    - technology

    - max_advertised_download_speed

    - max_advertised_upload_speed

    """

    np.random.seed(42)
    data = []

    location_counter = 1

    for block in range(1, n_blocks + 1):

        block_geoid = f"42{block:013d}"  # Pennsylvania FIPS code is 42, block code is 13 after
the state FIPS code

        n_locations = np.random.randint(20, locations_per_block + 1)

        # First, assign which providers serve this block (2-6 providers per block)

        n_providers_in_block = np.random.randint(2, min(7, n_providers + 1))

        providers_in_block = np.random.choice(

            [f"PROV_{i:03d}" for i in range(1, n_providers + 1)],

            size=n_providers_in_block,

            replace=False,

        )

        # Create locations

        for _ in range(n_locations):

            location_id = f"LOC_{location_counter:06d}"

            location_counter += 1

            # Each location gets a subset of the providers in this block

            # Coverage ranges from 40% to 80% to create edge-out opportunities

            coverage_pct = np.random.uniform(0.4, 0.8)

            n_providers_at_loc = max(1, int(len(providers_in_block) * coverage_pct))

            providers_at_loc = np.random.choice(
    providers_in_block,
    size=min(n_providers_at_loc, len(providers_in_block)),
    replace=False,
)
for provider in providers_at_loc:
    # Random technology - reduced slow tech to create edge-out opportunities
    # Strategy: Make slow tech rare so upgrade opportunities are limited
    # This allows the model to demonstrate edge-out builds once upgrades are exhausted
    technology = np.random.choice([10, 40, 50], p=[0.05, 0.25, 0.7])
    # Speed based on technology
    if technology == 10:  # Copper
        # All slow copper to maximize upgrade opportunities where copper exists
        download = 25  # Always slow
        upload = download // 10
    elif technology == 40:  # Cable
        # Mostly fast cable (80% fast) to minimize upgrade opportunities
        download = np.random.choice([200, 600, 1000], p=[0.2, 0.5, 0.3])
        upload = download // 10
    else:  # Fiber
        download = np.random.choice([500, 1000, 2000], p=[0.3, 0.5, 0.2])
        upload = download
    data.append(
                {
                    "location_id": location_id,
                    "block_geoid": block_geoid,
                    "provider_id": provider,
                    "technology": technology,
                    "max_advertised_download_speed": download,
                    "max_advertised_upload_speed": upload,
                }
            )
    return pd.DataFrame(data)


# Example usage:
# df = generate_sample_data(n_blocks=1000, n_providers=15, locations_per_block=100)
```

Note: This data generation approach creates both upgrade and edge-out opportunities. Providers are assigned to blocks first, then serve only a subset of locations within each block (40-80% coverage). This structure ensures that providers have presence in blocks but not at all locations, creating natural edge-out opportunities. The reduced proportion of slow technologies (slow copper and slow cable) limits upgrade opportunities, allowing edge-out builds to appear in later years once upgrades are exhausted.

Questions?

If you have any questions about the assignment, please don't hesitate to reach out to data-analytics@newstreetresearch.com. Good luck!