Data Preprocessing Task Guide - Detailed

Overview

Input Datasets:

- `ttc-subway-delay-2024.xlsx` (26,467 records)
- `ttc-subway-delay-data-since-2025.csv` (25,713 records)

Required Libraries:

```
pandas
openpyxl
numpy
```

Task 1: Standardize Line Names

Dataset: Both 2024.xlsx and 2025.csv

Current Problem:

- Line 1 has 20+ variations: YU, YU/BD, YU / BD, YU/ BD, LINE 1, ONGE-UNIVERSITY AND BL, etc.
- Line 2 has similar issues: BD, BLOOR DANFORTH, BD/YU, BD / YU, etc.
- Invalid entries: 109 RANEE, 20 CLIFFSIDE, TRACK LEVEL ACTIVITY

Subtasks:

1. Create mapping dictionary for all 22 unique Line values to 4 canonical categories:
    - Line 1 (all YU variations)
    - Line 2 (all BD variations)
    - Line 4 (all SHP variations)
    - Other (SRT, etc.)
2. Handle multi-line entries (YU/BD, YU / BD, etc.) - decide whether to:
    - Keep as "Line 1/2" (combined category), OR
    - Drop these records, OR
    - Assign to primary line only
3. Apply mapping to both datasets

4. Verify results: Check that Line column contains only expected values

Libraries: pandas

Output: Both datasets with standardized Line column

Task 2: Remove Invalid Records

Dataset: Cleaned data from Task 1

Records to Remove:

- ~44 records from 2024 with missing/null Line values
- ~67 records from 2025 with missing/null Line values
- ~5 erroneous entries (109 RANEE, 20 CLIFFSIDE, TRACK LEVEL ACTIVITY)

Subtasks:

1. Filter out rows where Line is null/NaN
2. Filter out specific invalid entries using not-in list
3. Log number of records removed from each dataset
4. Verify total remaining: ~52,064 records

Libraries: pandas

Output: Clean dataset with valid Line values only

Task 3: Parse Time Column

Dataset: Data from Task 2

Current Format: Text column with mixed formats

- Standard: "02:00", "06:00", "22:00" (most common)
- With minutes: "08:46", "17:05", "18:07" (20-30%)

Subtasks:

1. Create function to parse Time strings:
    - Split on ":" character
    - Extract first component as hour (integer 0-23)

- Handle any parsing errors

2. Create new column "hour" with values 0-23
3. Validate hour range:
    - Min should be 0, Max should be 23
    - No null values after parsing
    - Report any parsing failures
4. Drop original Time column (no longer needed)

Libraries: pandas

Output: Dataset with new "hour" column (integer), original Time column removed

---

Task 4: Parse Date Column

Dataset: Data from Task 3

Current Format:

- 2024 file: datetime format (2024-01-01)
- 2025 file: text format "2025-01-01"

Subtasks:

1. Ensure both datasets have Date as datetime type:
    - Convert 2025 CSV date strings to datetime
    - Verify 2024 Excel dates are already datetime
2. Extract temporal features into new columns:
    - `year`: Extract year (int)
    - `month`: Extract month 1-12 (int)
    - `week`: Extract week number (int)
3. Validate date range:
    - 2024 data: Jan 1 2024 - Dec 31 2024
    - 2025 data: Jan 1 2025 - Dec 31 2025
4. Check for any invalid/null dates and remove if found

Libraries: pandas

Output: Dataset with Date as datetime, plus year/month/week columns

---

Task 5: Define Target Variable

Dataset: Data from Task 4

Decision Required: What constitutes "delayed"?

Subtasks:

1. Analyze Min Delay distribution:
    - Calculate percentiles (25th, 50th, 75th)
    - Review mean, median, max values
    - Plot histogram
2. Choose threshold. Options:
    - **5 minutes** (recommended): Balanced class distribution
    - **10 minutes**: Stricter definition
    - **Custom**: Based on stakeholder input
3. Create binary column `is_delayed`:
    - If Min Delay >= threshold, then 1 (delayed)
    - Else 0 (on-time)
4. Check class distribution:
    - Count records in each class
    - Calculate percentage split (should be roughly 70/30 or 75/25)
    - Log results
5. Verify no null values in new column

Libraries: pandas, numpy

Output: Dataset with new `is_delayed` binary column (0 or 1)

Task 6: Create Day-of-Week Numeric

Dataset: Data from Task 5

Current Format: Text column "Day" with values: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday

Subtasks:

1. Create mapping dictionary:
    - Monday → 0
    - Tuesday → 1
    - Wednesday → 2
    - Thursday → 3
    - Friday → 4
    - Saturday → 5
    - Sunday → 6
2. Create new column `day_of_week` with numeric values 0-6
3. Verify mapping is correct by spot-checking random rows
4. Ensure no null values
5. Keep original Day column or drop (your choice)

Libraries: pandas

Output: Dataset with new `day_of_week` numeric column

Task 7: Create Weekday Indicator

Dataset: Data from Task 6

Subtasks:

1. Create binary column `is_weekend`:
    - If day_of_week >= 5 (Saturday or Sunday), then 1
    - Else 0 (Monday-Friday)
2. Verify values are only 0 or 1
3. Spot-check: Ensure Saturdays/Sundays → 1, weekdays → 0

Libraries: pandas

Output: Dataset with new `is_weekend` binary column

Task 8: Calculate Historical Frequencies

Dataset: Data from Task 7

Purpose: Create features showing historical delay patterns for modeling

Subtasks:

1. **Route-level delay frequency:**
    - Group by Line
    - Calculate proportion of delays: count(is_delayed=1) / total count
    - Create mapping: Line → delay_frequency
    - Map back to all rows in new column `route_delay_frequency`
2. **Route-Hour delay frequency:**
    - Create composite key: Line + "_" + hour
    - Group by this key
    - Calculate proportion delayed
    - Map to new column `route_hour_delay_frequency`
3. **Route-Day-Hour delay frequency:**
    - Create composite key: Line + "" + *day_of_week* + "" + hour
    - Group by this key
    - Calculate proportion delayed
    - Map to new column `route_day_hour_delay_frequency`
4. **Validation:**
    - Check that values are between 0 and 1
    - No null values
    - Verify a few manual calculations
5. **Documentation:**
    - Note which time period these frequencies cover (full 2-year dataset)
    - These will be used as features in Phase 1 modeling

Libraries: pandas

Output: Dataset with three new frequency columns

Task 9: Handle Missing Bound Column

Dataset: Data from Task 8

Current Status:

- ~36-37% of rows missing Bound values
- Bound represents direction (N, S, E, W)

Decision Point - Choose ONE approach:

Option A: Drop Column (Simpler)

- Remove Bound column entirely
- Rationale: Not required for Phase 1, adds processing complexity

Option B: Keep and Flag Missing

- Create column `has_bound`: 1 if Bound exists, 0 if missing
- Leave null values as-is
- Rationale: May be useful in Phase 2

Option C: Attempt Inference (Complex)

- Try to infer direction from Station + Line combination
- Requires additional analysis/lookup table
- Only if critical for Phase 1

Recommended: Option A (drop) for efficiency

Subtasks (if Option A):

1. Remove Bound column
2. Document that direction information was not included in Phase 1

Subtasks (if Option B):

1. Create `has_bound` indicator column
2. Keep Bound column with nulls

Libraries: pandas

Output: Dataset with Bound column either removed or flagged

Task 10: Final Validation

Dataset: Data from Task 9 (pre-export)

Subtasks:

1. **Check for null values:**
    - Generate null count per column
    - Ensure no nulls in: Line, hour, day_of_week, is_delayed, is_weekend
    - Acceptable nulls: None
2. **Verify data types:**
    - Line: object (string)
    - hour: int (0-23)
    - day_of_week: int (0-6)
    - is_weekend: int (0 or 1)
    - is_delayed: int (0 or 1)
    - Date: datetime
    - year, month, week: int
    - Min Delay: int (original measurement)
    - route_delay_frequency, route_hour_delay_frequency, route_day_hour_delay_frequency: float (0-1)
3. **Check value ranges:**
    - hour: 0-23 only
    - day_of_week: 0-6 only
    - is_weekend: 0-1 only
    - is_delayed: 0-1 only
    - month: 1-12 only
    - All frequency columns: 0-1 range
4. **Record counts:**
    - Total records after all cleaning: ~52,000-52,064
    - Log breakdown by line (Line 1, Line 2, Line 4)
5. **Data quality report:**
    - Generate summary table with column names, types, null counts, value ranges
    - Confirm all checks passed

Libraries: pandas

Output: Validation report confirming data readiness for export

---

Task 11: Export Clean Data

Dataset: Validated data from Task 10

Subtasks:

1. **Select columns to keep:**
    - Date, Line, hour, day_of_week, is_weekend, month, week, year
    - Min Delay, is_delayed
    - route_delay_frequency, route_hour_delay_frequency, route_day_hour_delay_frequency
    - Code (delay reason, useful for exploration)
    - Station (optional, for analysis)
2. **Export to CSV:**
    - Filename: `cleaned_ttc_delay_data.csv`
    - Index: False
    - Encoding: utf-8
    - Delimiter: comma
3. **Create metadata file** (optional):
    - Record preprocessing date/time
    - Total records: count
    - Date range covered
    - Columns included
    - Any notes (e.g., threshold used for is_delayed)
4. **Verification:**
    - Read exported CSV back in
    - Verify row count matches export count
    - Spot-check 5-10 random rows
    - Confirm no corruption during export

Libraries: pandas

Output:

- `cleaned_ttc_delay_data.csv` (ready for modeling)
- Optional: `preprocessing_metadata.json` (documentation)

Summary of Columns in Final Dataset

Column	Type	Values	Purpose
Date	datetime	2024-2025	Record timestamp
Line	string	Line 1, Line 2, Line 4	Route identifier
hour	int	0-23	Time of day feature
day_of_week	int	0-6	Day feature (0=Mon)
is_weekend	int	0, 1	Weekday flag
month	int	1-12	Month for seasonality
week	int	1-53	Week number
year	int	2024, 2025	Year
Min Delay	int	0-900	Original delay in minutes
is_delayed	int	0, 1	**TARGET VARIABLE**
route_delay_frequency	float	0-1	Historical delay rate by route
route_hour_delay_frequency	float	0-1	Historical delay rate by route+hour
route_day_hour_delay_frequency	float	0-1	Historical delay rate by route+day+hour
Code	string	125 codes	Delay reason (exploratory)
Station	string	456 stations	Location (exploratory)

Dependencies Summary

Python Libraries:

```
pandas>=1.3.0
openpyxl>=3.0.0
numpy>=1.20.0
```

Installation:

```bash
pip install pandas openpyxl numpy
```

Checkpoint Checklist

After each task, verify:

- ✓ No unexpected data loss (record counts reasonable)
- ✓ Output data types correct
- ✓ No null values in critical columns
- ✓ Value ranges as expected

Before moving to Task 11, confirm all Tasks 1-10 complete without errors.