07_ingestion_drift_simulation_nyc

Simulate ingestion drift (missing partition, partial ingestion, delayed ingestion) on NYC Taxi monthly partitions and analyze partition-level metrics and downstream regression impact.

In [1]:
```
import os
import glob
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import ks_2samp
import warnings
warnings.filterwarnings('ignore')
sns.set(style='whitegrid')
```

In [2]:
```
# 1) Collect monthly files (partitions)
files = sorted(glob.glob(os.path.join('..','data','nyc_taxi','*.csv')) )
print('Found partitions:', files)
# deterministic sampling fraction
SAMPLE_FRAC = 0.01
RANDOM_STATE = 42
def load_partition(path, sample_frac=SAMPLE_FRAC):
    # memory-efficient deterministic sample
    df = pd.read_csv(path, low_memory=False)
    if sample_frac < 1.0:
        df = df.sample(frac=sample_frac, random_state=RANDOM_STATE)
    # attach partition id from filename
    pid = os.path.basename(path).replace('.csv','')
    df['partition'] = pid
    return df
```

```
Found partitions: ['../data/nyc_taxi/yellow_tripdata_2015-01.csv', '../data/nyc_taxi/yellow_tripdata_2016-01.csv',
'../data/nyc_taxi/yellow_tripdata_2016-02.csv', '../data/nyc_taxi/yellow_tripdata_2016-03.csv']
```

In [3]:
```
# Load all partitions with deterministic sampling
parts = []
for p in files:
    try:
        parts.append(load_partition(p))
    except Exception as e:
        print('Error loading', p, e)
df_all = pd.concat(parts, ignore_index=True) if parts else pd.DataFrame()
print('Concatenated shape:', df_all.shape)
```

```
Concatenated shape: (472489, 21)
```

In [4]:
```
# Basic ingestion metrics per partition before simulation
metric_cols = []
for c in ['trip_distance','fare_amount']:
    if c in df_all.columns:
        metric_cols.append(c)
grouped = df_all.groupby('partition').agg(row_count=('partition','size'))
for c in metric_cols:
    grouped[c + '_mean'] = df_all.groupby('partition')[c].mean()
grouped = grouped.reset_index()
grouped.head()
```

Out[4]:

	partition	row_count	trip_distance_mean	fare_amount_mean
0	yellow_tripdata_2015-01	127490	2.794001	11.911037
1	yellow_tripdata_2016-01	109069	2.903486	12.506083
2	yellow_tripdata_2016-02	113820	2.846650	12.361499
3	yellow_tripdata_2016-03	122110	9.481320	12.649050

Ingestion Drift Scenarios

We will simulate Missing Partition, Partial Ingestion and Delayed Ingestion and compare partition-level metrics before and after.

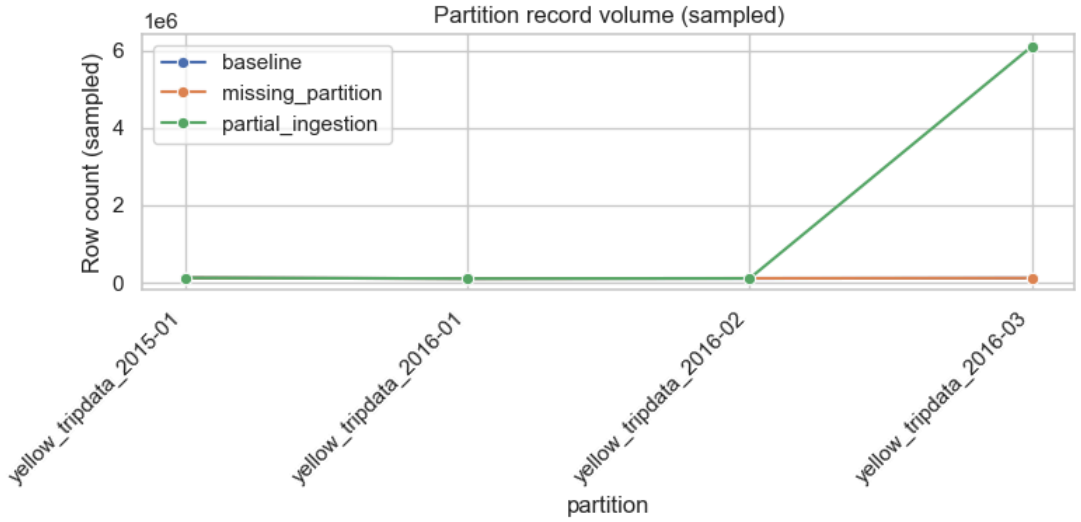
In [5]:
```
# Scenario A: Missing Partition — remove a specific month if available (e.g., contains '2016-02')
target_month = None
for f in files:
    if '2016-02' in f:
        target_month = f; break
if target_month is None and files:
    # fallback: choose middle partition
    target_month = files[len(files)//2]
print('Simulating missing partition:', os.path.basename(target_month))
parts_missing = [load_partition(p) for p in files if p != target_month]
df_missing = pd.concat(parts_missing, ignore_index=True)
group_missing = df_missing.groupby('partition').agg(row_count=('partition','size'))
for c in metric_cols:
    group_missing[c + '_mean'] = df_missing.groupby('partition')[c].mean()
group_missing = group_missing.reset_index()
```

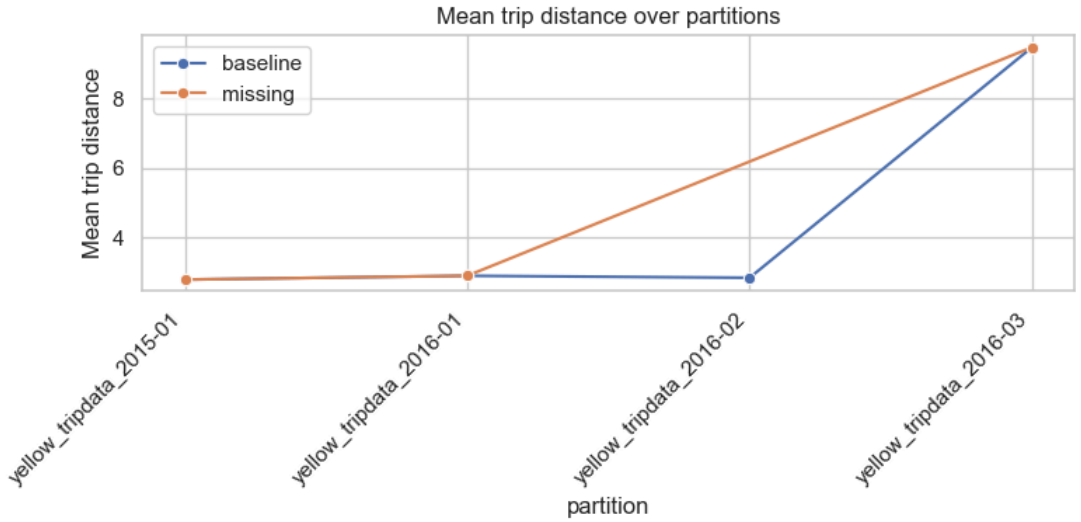
```
Simulating missing partition: yellow_tripdata_2016-02.csv
```

In [6]:
```
# Scenario B: Partial ingestion — retain only 50% rows for the last partition
if files:
    last = files[-1]
    parts_partial = [load_partition(p) for p in files[:-1]]
    parts_partial.append(load_partition(last, sample_frac=0.5))
    df_partial = pd.concat(parts_partial, ignore_index=True)
    group_partial = df_partial.groupby('partition').agg(row_count=('partition','size'))
    for c in metric_cols:
        group_partial[c + '_mean'] = df_partial.groupby('partition')[c].mean()
    group_partial = group_partial.reset_index()
else:
    group_partial = pd.DataFrame()
```

In [7]:
```
# Scenario C: Delayed ingestion — shuffle partition ordering to simulate late arrival of one partition
if len(files) > 1:
    delayed_idx = 0
    shuffled = files.copy()
    # move the chosen partition to the end to simulate delay
    p = shuffled.pop(delayed_idx)
    shuffled.append(p)
    parts_delayed = [load_partition(pp) for pp in shuffled]
    df_delayed = pd.concat(parts_delayed, ignore_index=True)
    group_delayed = df_delayed.groupby('partition').agg(row_count=('partition','size'))
    for c in metric_cols:
        group_delayed[c + '_mean'] = df_delayed.groupby('partition')[c].mean()
    group_delayed = group_delayed.reset_index()
else:
    group_delayed = pd.DataFrame()
```

In [8]:
```
# Compare metrics before and after and save tables/plots
os.makedirs(os.path.join('..','results','figures'), exist_ok=True)
os.makedirs(os.path.join('..','results','tables'), exist_ok=True)
grouped.to_csv(os.path.join('..','results','tables','07_baseline_partition_metrics.csv'), index=False)
group_missing.to_csv(os.path.join('..','results','tables','07_missing_partition_metrics.csv'), index=False)
group_partial.to_csv(os.path.join('..','results','tables','07_partial_partition_metrics.csv'), index=False)
group_delayed.to_csv(os.path.join('..','results','tables','07_delayed_partition_metrics.csv'), index=False)
# Plot record volume over time for baseline and scenarios
plt.figure(figsize=(8,4))
sns.lineplot(data=grouped, x='partition', y='row_count', marker='o', label='baseline')
sns.lineplot(data=group_missing, x='partition', y='row_count', marker='o', label='missing_partition')
sns.lineplot(data=group_partial, x='partition', y='row_count', marker='o', label='partial_ingestion')
plt.xticks(rotation=45, ha='right')
plt.ylabel('Row count (sampled)')
plt.title('Partition record volume (sampled)')
plt.tight_layout()
plt.savefig(os.path.join('..','results','figures','07_partition_volume.png'), dpi=300)
plt.show()
# Plot mean trip distance over time and annotate missing partition event
if 'trip_distance_mean' in grouped.columns:
    plt.figure(figsize=(8,4))
    sns.lineplot(data=grouped, x='partition', y='trip_distance_mean', marker='o', label='baseline')
    sns.lineplot(data=group_missing, x='partition', y='trip_distance_mean', marker='o', label='missing')
    plt.xticks(rotation=45, ha='right')
    plt.ylabel('Mean trip distance')
    plt.title('Mean trip distance over partitions')
    # annotate the removed partition if known
    removed = os.path.basename(target_month).replace('.csv','')
    plt.annotate('missing', xy=(removed, 0), xytext=(removed, grouped['trip_distance_mean'].max()), arrowprops=dict
    plt.tight_layout()
    plt.savefig(os.path.join('..','results','figures','07_trip_distance_over_time.png'), dpi=300)
    plt.show()
```





Research notes

- Ingestion drift changes the data available at the lake layer — missing partitions or partial ingestions cause downstream statistics and models to shift before any ML training occurs.
- Deterministic sampling preserves reproducibility for experiments while keeping memory use low.