Synthetic Data Validation Report: Singapore Check-ins

Basic Dataset Stats

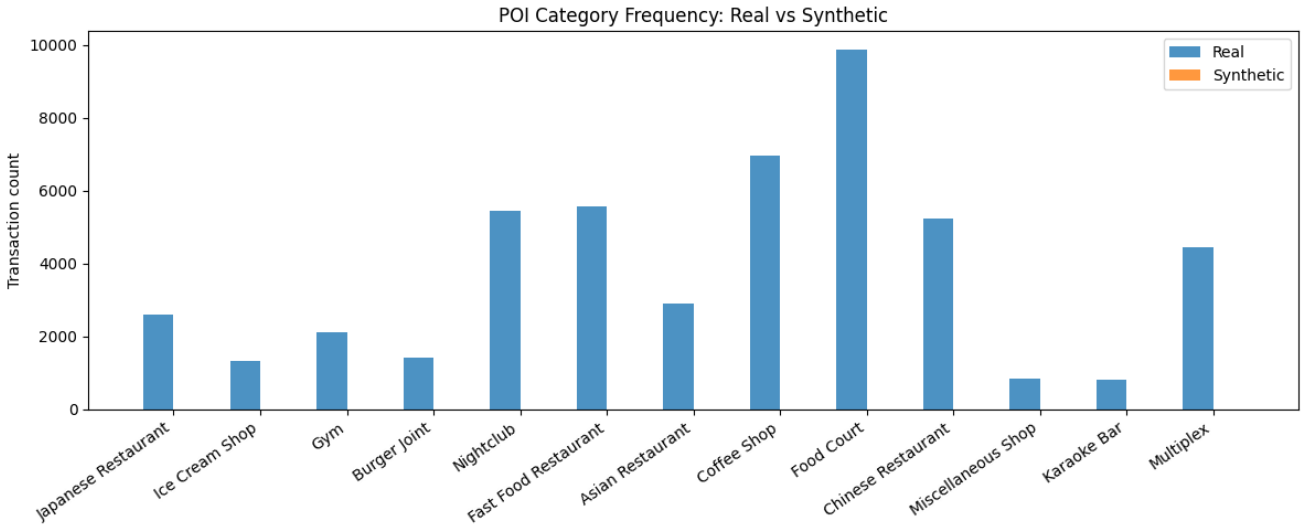
Real dataset: 2339 users, 24012 POIs, 261698 check-ins

Synthetic dataset: 50 users, 82 POIs, 94 transactions

POI Category Distribution

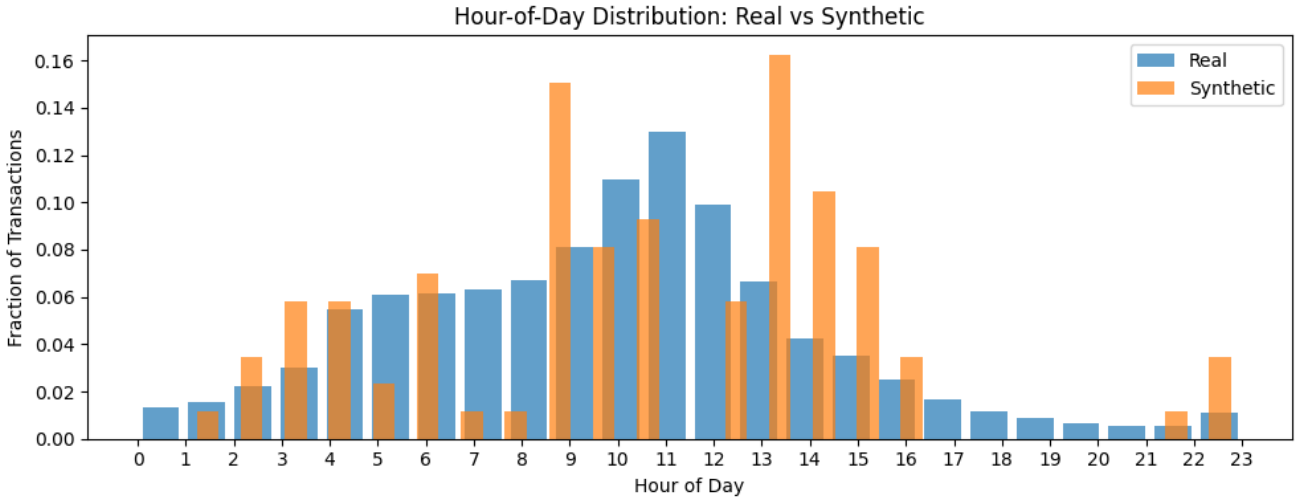
Top categories: Japanese Restaurant, Ice Cream Shop, Gym, Burger Joint, Nightclub, Fast Food Restaurant, Asian Restaurant, Coffee Shop, Food Court, Chinese Restaurant, Miscellaneous Shop, Karaoke Bar, Multiplex

JS Divergence: 0.24



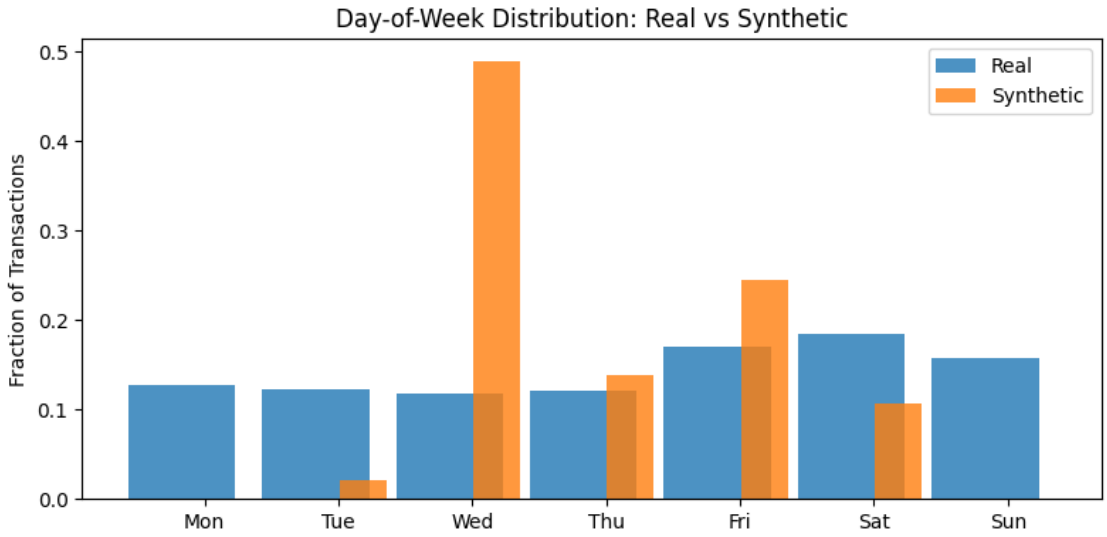
Hour-of-Day Distribution

KS statistic: 0.17, p-value: 0.0084



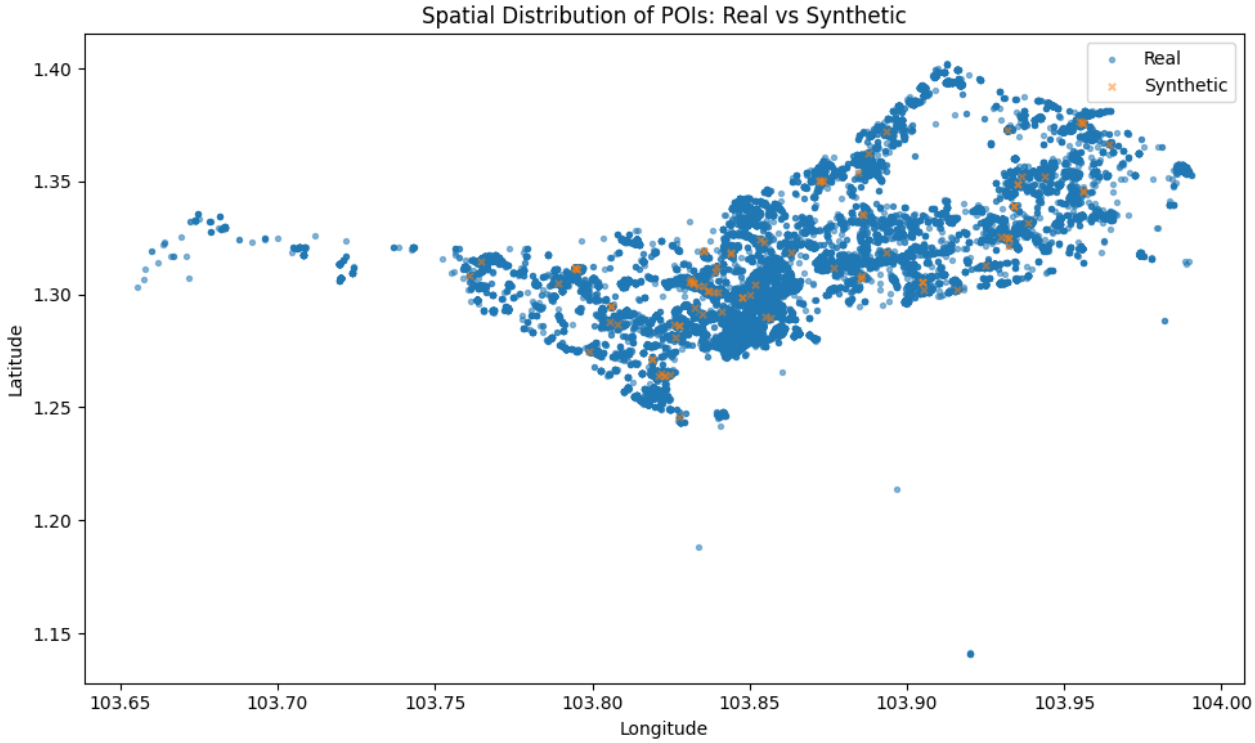
Day-of-Week Distribution

JS Divergence: 0.43



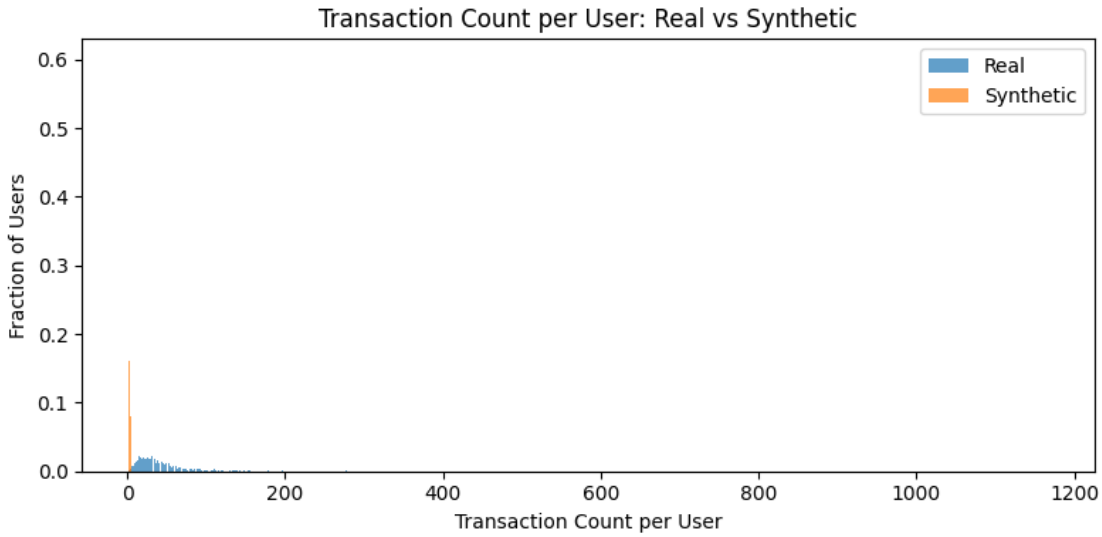
Spatial Distribution

KS Latitude: 0.17 (p=0.01), KS Longitude: 0.13 (p=0.072)



Transaction Count per User

KS statistic: 0.97, p-value: 1.5e-70



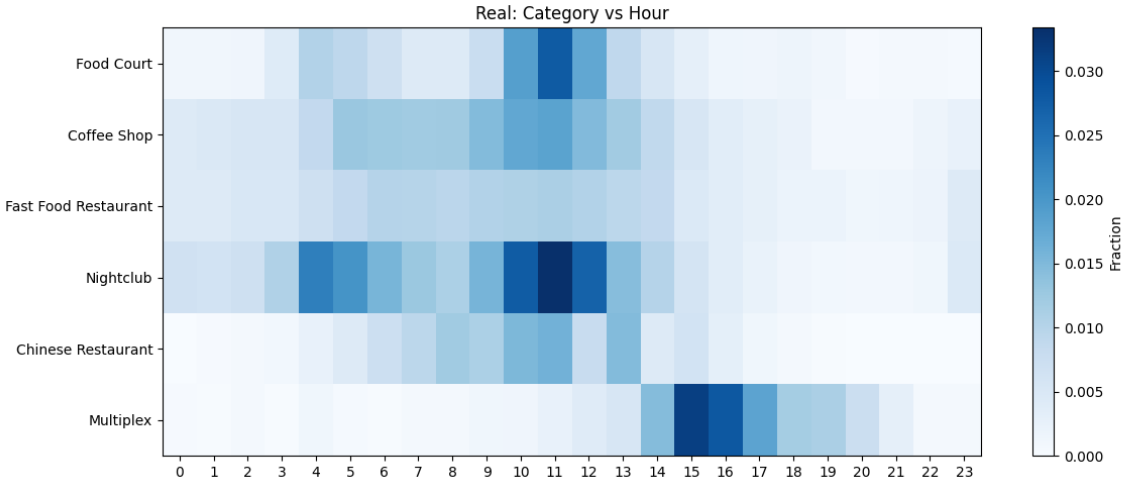
User Journey Coherence

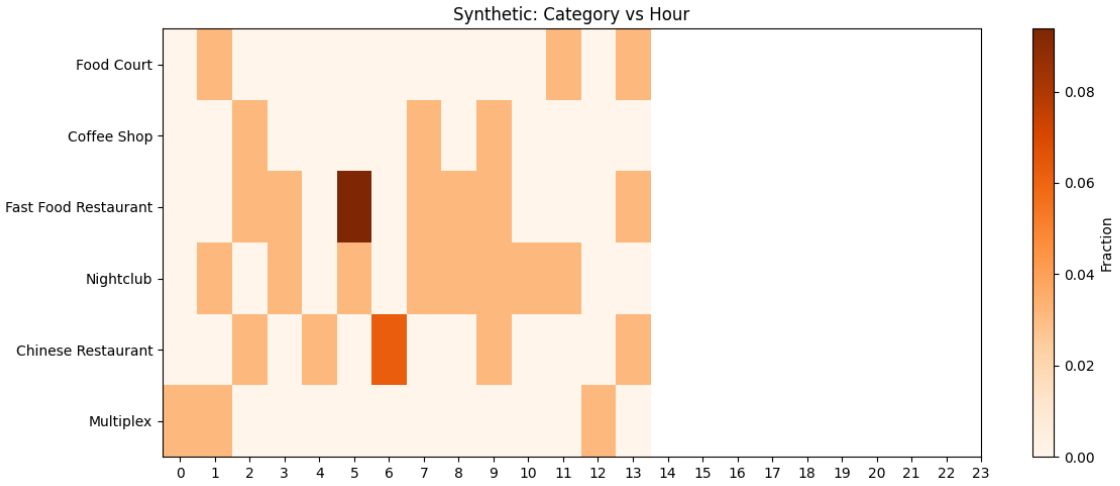
Real mean step: 4157 m, median: 2499 m, std: 4618 m

Synthetic mean step: 1921 m, median: 1989 m, std: 1106 m

Joint Distribution (Category vs Hour)

JS Divergence: 0.61 (joint structure preservation)





Model Utility (RF, hour+lat+lon -> category, top 6)

Train real, test synthetic: 0.91

Train synthetic, test real: 0.33

Appendix: Full Python Code

```
import pandas as pd
import json
import matplotlib.pyplot as plt
import numpy as np
from scipy.spatial.distance import jensenshannon
from scipy.stats import ks_2samp
from geopy.distance import geodesic
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from fpdf import FPDF

# --- 1. DATA LOADING ---
real_df = pd.read_csv("/home/rubesh/Desktop/sweta/google_review_collection_POI_task/singapore_foursquare_dataset_analysis/Code_Freeze_Changes/c1_data/c0_original/singapore_checkins_filtered_with_locations_coord.txt", sep="\t", header=None,
                      names=["user_id", "place_id", "timestamp", "unknown", "latitude", "longitude"])
with open("/home/rubesh/Desktop/sweta/google_review_collection_POI_task/singapore_foursquare_dataset_analysis/Code_Freeze_Changes/all_results/singapore_synthetic_user_profiles_full.json", "r") as f:
    synthetic_profiles = json.load(f)
cat_map = pd.read_csv("/home/rubesh/Desktop/sweta/google_review_collection_POI_task/singapore_foursquare_dataset_analysis/Code_Freeze_Changes/c1_data/c0_original/sg_place_id_to_category.csv")
rel_poi = pd.read_excel("/home/rubesh/Desktop/sweta/google_review_collection_POI_task/singapore_foursquare_dataset_analysis/Code_Freeze_Changes/c1_data/c0_original/Relevant_POI_category.xlsx")

# --- 2. BASIC STATS ---
real_user_count = real_df['user_id'].nunique()
real_poi_count = real_df['place_id'].nunique()
real_row_count = len(real_df)
synthetic_user_count = len(synthetic_profiles)
synthetic_txn_rows = sum([len(u['interaction']['transactions']) for u in synthetic_profiles])
synthetic_poi_ids = set()
for u in synthetic_profiles:
    synthetic_poi_ids.update([p['poiId'] for p in u['pois']])
synthetic_poi_count = len(synthetic_poi_ids)

# --- 3. CATEGORY DISTRIBUTION ---
real_merged = pd.merge(real_df, cat_map, left_on="place_id", right_on="place_id", how="left")
real_merged = real_merged.rename(columns={"category": "poi_category"})
relevant_cats = rel_poi[rel_poi.iloc[:, 2].str.lower() == 'yes']['POI Category in Singapore'].str.strip().tolist()
real_merged = real_merged[real_merged['poi_category'].isin(relevant_cats)]
real_cat_counts = real_merged['poi_category'].value_counts().sort_values(ascending=False)
synthetic_txn_rows_list = []
for u in synthetic_profiles:
    for t in u['interaction']['transactions']:
        synthetic_txn_rows_list.append({
            "poiId": t['poiId'],
            "category": t['poiCategories'][0] if t['poiCategories'] else "Unknown",
            "timestamp": t['timestamp'] if 'timestamp' in t else None
        })
synthetic_txns = pd.DataFrame(synthetic_txn_rows_list)
synthetic_cat_counts = synthetic_txns['category'].value_counts().sort_values(ascending=False)
top_categories = list(set(real_cat_counts.head(8).index).union(synthetic_cat_counts.head(8).index))
real_dist = real_cat_counts.reindex(top_categories, fill_value=0).values
real_dist = real_dist / real_dist.sum()
synth_dist = synthetic_cat_counts.reindex(top_categories, fill_value=0).values
synth_dist = synth_dist / synth_dist.sum()
js_divergence = jensenshannon(real_dist, synth_dist)
plt.figure(figsize=(12,5))
bar_width = 0.35
index = range(len(top_categories))
plt.bar(index, real_cat_counts.reindex(top_categories, fill_value=0).values, bar_width, label='Real', alpha=0.8)
plt.bar([i+bar_width for i in index], synthetic_cat_counts.reindex(top_categories, fill_value=0).values, bar_width, label='Synthetic', alpha=0.8)
```

```
plt.xticks([i+bar_width/2 for i in index], top_categories, rotation=35, ha='right')
plt.ylabel('Transaction count')
plt.title('POI Category Frequency: Real vs Synthetic')
plt.legend()
plt.tight_layout()
plt.savefig("/home/rubesh/Desktop/sweta/google_review_collection_POI_task/singapore_foursquare_dataset_analysis/Code_Freeze_Changes/all_results/plot_cat_freq.png")
plt.close()

# --- 4. HOUR OF DAY ---
real_hours = pd.to_datetime(real_merged['timestamp']).dt.hour
synthetic_hours = pd.to_datetime(synthetic_txns['timestamp']).dt.hour
ks_hour = ks_2samp(real_hours, synthetic_hours)
plt.figure(figsize=(10,4))
plt.hist(real_hours, bins=24, alpha=0.7, label='Real', density=True, rwidth=0.8)
plt.hist(synthetic_hours, bins=24, alpha=0.7, label='Synthetic', density=True, rwidth=0.5)
plt.xticks(range(24))
plt.xlabel('Hour of Day')
plt.ylabel('Fraction of Transactions')
plt.title('Hour-of-Day Distribution: Real vs Synthetic')
plt.legend()
plt.tight_layout()
plt.savefig("/home/rubesh/Desktop/sweta/google_review_collection_POI_task/singapore_foursquare_dataset_analysis/Code_Freeze_Changes/all_results/plot_hour.png")
plt.close()

# --- 5. DAY OF WEEK ---
real_dow = pd.to_datetime(real_merged['timestamp']).dt.dayofweek
synthetic_dow = pd.to_datetime(synthetic_txns['timestamp']).dt.dayofweek
real_dow_counts = real_dow.value_counts().sort_index()
synthetic_dow_counts = synthetic_dow.value_counts().sort_index()
all_dows = list(sorted(set(real_dow_counts.index).union(synthetic_dow_counts.index)))
real_dow_dist = real_dow_counts.reindex(all_dows, fill_value=0).values / real_dow_counts.sum()
synthetic_dow_dist = synthetic_dow_counts.reindex(all_dows, fill_value=0).values / synthetic_dow_counts.sum()
js_dow = jensenshannon(real_dow_dist, synthetic_dow_dist)
dow_labels = ['Mon','Tue','Wed','Thu','Fri','Sat','Sun']
plt.figure(figsize=(8,4))
plt.bar(range(7), real_dow_counts.reindex(range(7), fill_value=0)/real_dow_counts.sum(), alpha=0.8, label='Real')
plt.bar([x+0.35 for x in range(7)], synthetic_dow_counts.reindex(range(7), fill_value=0)/synthetic_dow_counts.sum(), width=0.35, alpha=0.8, label='Synthetic')
plt.xticks([x+0.17 for x in range(7)], dow_labels)
plt.ylabel('Fraction of Transactions')
plt.title('Day-of-Week Distribution: Real vs Synthetic')
plt.legend()
plt.tight_layout()
plt.savefig("/home/rubesh/Desktop/sweta/google_review_collection_POI_task/singapore_foursquare_dataset_analysis/Code_Freeze_Changes/all_results/plot_dow.png")
plt.close()

# --- 6. SPATIAL DISTRIBUTION ---
real_lat = real_merged['latitude']
real_lon = real_merged['longitude']
synthetic_lat = pd.to_numeric(synthetic_txns['poiId'].map(
    {p['poiId']: p['location']['latitude'] for u in synthetic_profiles for p in u['pois']}
), errors='coerce')
synthetic_lon = pd.to_numeric(synthetic_txns['poiId'].map(
    {p['poiId']: p['location']['longitude'] for u in synthetic_profiles for p in u['pois']}
), errors='coerce')
ks_lat = ks_2samp(real_lat, synthetic_lat)
ks_lon = ks_2samp(real_lon, synthetic_lon)
plt.figure(figsize=(10,6))
plt.scatter(real_lon, real_lat, s=8, alpha=0.5, label='Real', marker='o')
plt.scatter(synthetic_lon, synthetic_lat, s=14, alpha=0.5, label='Synthetic', marker='x')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Spatial Distribution of POIs: Real vs Synthetic')
plt.legend()
```

```
plt.tight_layout()
plt.savefig("/home/rubesh/Desktop/sweta/google_review_collection_POI_task/singapore_foursquare_dataset_analysis/Code_Freeze_Changes/all_results/plot_spatial.png")
plt.close()

# --- 7. TXN COUNT PER USER ---
real_user_txn_counts = real_merged['user_id'].value_counts()
synthetic_counts = [len(u['interaction']['transactions']) for u in synthetic_profiles]
ks_txn_count = ks_2samp(real_user_txn_counts.values, synthetic_counts)
plt.figure(figsize=(8,4))
plt.hist(real_user_txn_counts.values, bins=range(1, max(real_user_txn_counts.max(), max(synthetic_counts))+2),
         alpha=0.7, label='Real', density=True, rwidth=0.85)
plt.hist(synthetic_counts, bins=range(1, max(real_user_txn_counts.max(), max(synthetic_counts))+2),
         alpha=0.7, label='Synthetic', density=True, rwidth=0.5)
plt.xlabel('Transaction Count per User')
plt.ylabel('Fraction of Users')
plt.title('Transaction Count per User: Real vs Synthetic')
plt.legend()
plt.tight_layout()
plt.savefig("/home/rubesh/Desktop/sweta/google_review_collection_POI_task/singapore_foursquare_dataset_analysis/Code_Freeze_Changes/all_results/plot_txn_count.png")
plt.close()

# --- 8. USER JOURNEY COHERENCE ---
def avg_user_step_distance_real(df):
    step_distances = []
    for uid, group in df.groupby("user_id"):
        group = group.sort_values("timestamp")
        coords = list(zip(group['latitude'], group['longitude']))
        if len(coords) > 1:
            step_dists = [geodesic(coords[i], coords[i+1]).meters for i in range(len(coords)-1)]
            step_distances.extend(step_dists)
    return np.mean(step_distances), np.median(step_distances), np.std(step_distances)
def avg_user_step_distance_synth(profiles):
    step_distances = []
    for u in profiles:
        txns = sorted(u['interaction']['transactions'], key=lambda t: t['timestamp'])
        coords = [(t['userLocation']['latitude'], t['userLocation']['longitude']) for t in txns]
        if len(coords) > 1:
            step_dists = [geodesic(coords[i], coords[i+1]).meters for i in range(len(coords)-1)]
            step_distances.extend(step_dists)
    return np.mean(step_distances), np.median(step_distances), np.std(step_distances)
real_avg, real_median, real_std = avg_user_step_distance_real(real_merged)
synth_avg, synth_median, synth_std = avg_user_step_distance_synth(synthetic_profiles)

# --- 9. JOINT DISTRIBUTION (category vs hour) ---
top6_cats = real_cat_counts.head(6).index.tolist()
def joint_cat_hour(df, cats):
    filtered = df[df['poi_category'].isin(cats)]
    hours = pd.to_datetime(filtered['timestamp']).dt.hour
    return pd.crosstab(filtered['poi_category'], hours, normalize='all')
real_joint = joint_cat_hour(real_merged, top6_cats)
synthetic_joint = pd.DataFrame()
if not synthetic_txns.empty:
    synthetic_txns = synthetic_txns.copy()
    synthetic_txns['hour'] = pd.to_datetime(synthetic_txns['timestamp']).dt.hour
    synthetic_joint = pd.crosstab(
        synthetic_txns[synthetic_txns['category'].isin(top6_cats)]['category'],
        synthetic_txns[synthetic_txns['category'].isin(top6_cats)]['hour'],
        normalize='all'
    )
real_joint_flat = real_joint.reindex(index=top6_cats, columns=range(24), fill_value=0).values.flatten()
synth_joint_flat = synthetic_joint.reindex(index=top6_cats, columns=range(24), fill_value=0).values.flatten()
js_joint = jensenshannon(real_joint_flat, synth_joint_flat)
plt.figure(figsize=(12,5))
plt.imshow(real_joint.values, aspect='auto', cmap='Blues')
plt.xticks(range(24), range(24))
```

```
plt.yticks(range(len(top6_cats)), top6_cats)
plt.colorbar(label="Fraction")
plt.title("Real: Category vs Hour")
plt.tight_layout()
plt.savefig("/home/rubesh/Desktop/sweta/google_review_collection_POI_task/singapore_foursquare_dataset_analysis/Code_Freeze_Changes/all_results/plot_joint_real.png")
plt.close()
plt.figure(figsize=(12,5))
plt.imshow(synthetic_joint.values, aspect='auto', cmap='Oranges')
plt.xticks(range(24), range(24))
plt.yticks(range(len(top6_cats)), top6_cats)
plt.colorbar(label="Fraction")
plt.title("Synthetic: Category vs Hour")
plt.tight_layout()
plt.savefig("/home/rubesh/Desktop/sweta/google_review_collection_POI_task/singapore_foursquare_dataset_analysis/Code_Freeze_Changes/all_results/plot_joint_synth.png")
plt.close()

# --- 10. MODEL UTILITY ---
def prepare_xy(df, cat_col='poi_category', cats=None):
    sub = df[df[cat_col].isin(cats)]
    x = pd.DataFrame({
        "hour": pd.to_datetime(sub['timestamp']).dt.hour,
        "lat": sub['latitude'],
        "lon": sub['longitude'],
    })
    y = sub[cat_col]
    return x, y
x_real, y_real = prepare_xy(real_merged, 'poi_category', top6_cats)
poi_lat_map = {p['poiId']: p['location']['latitude'] for u in synthetic_profiles for p in u['pois']}
poi_lon_map = {p['poiId']: p['location']['longitude'] for u in synthetic_profiles for p in u['pois']}
synthetic_txns['lat'] = synthetic_txns['poiId'].map(poi_lat_map)
synthetic_txns['lon'] = synthetic_txns['poiId'].map(poi_lon_map)
x_synth = synthetic_txns[synthetic_txns['category'].isin(top6_cats)][['timestamp', 'lat', 'lon']].copy()
x_synth['hour'] = pd.to_datetime(x_synth['timestamp']).dt.hour
x_synth = x_synth[['hour', 'lat', 'lon']]
y_synth = synthetic_txns[synthetic_txns['category'].isin(top6_cats)]['category']
clf_real = RandomForestClassifier(n_estimators=50, random_state=0)
clf_real.fit(x_real, y_real)
y_pred_synth = clf_real.predict(x_synth)
acc_real2synth = accuracy_score(y_synth, y_pred_synth)
clf_synth = RandomForestClassifier(n_estimators=50, random_state=0)
clf_synth.fit(x_synth, y_synth)
y_pred_real = clf_synth.predict(x_real)
acc_synth2real = accuracy_score(y_real, y_pred_real)

# --- PDF REPORT GENERATION ---
pdf = FPDF()
pdf.set_auto_page_break(auto=True, margin=15)
pdf.add_page()
pdf.set_font("Arial", 'B', 16)
pdf.cell(0, 10, "Synthetic Data Validation Report: Singapore Check-ins", ln=True)
pdf.set_font("Arial", size=12)
pdf.cell(0, 10, "Basic Dataset Stats", ln=True)
pdf.set_font("Arial", size=11)
pdf.multi_cell(0, 8, f"Real dataset: {real_user_count} users, {real_poi_count} POIs, {real_row_count} check-ins\n"
               f"Synthetic dataset: {synthetic_user_count} users, {synthetic_poi_count} POIs, {synthetic_txn_rows} transactions\n")

pdf.add_page()
pdf.set_font("Arial", 'B', 13)
pdf.cell(0, 10, "POI Category Distribution", ln=True)
pdf.set_font("Arial", size=11)
pdf.multi_cell(0, 7, f"Top categories: {', '.join(top_categories)}\nJS Divergence: {js_divergence:.2f}")
pdf.ln(2)
pdf.image("/home/rubesh/Desktop/sweta/google_review_collection_POI_task/singapore_foursquare_dataset_analysis/Code_Freeze_Changes/all_results/plot_cat_freq.png", w=175)
```

```
pdf.add_page()
pdf.set_font("Arial", 'B', 13)
pdf.cell(0, 10, "Hour-of-Day Distribution", ln=True)
pdf.set_font("Arial", size=11)
pdf.multi_cell(0, 7, f"KS statistic: {ks_hour.statistic:.2f}, p-value: {ks_hour.pvalue:.2g}")
pdf.ln(2)
pdf.image("/home/rubesh/Desktop/sweta/google_review_collection_POI_task/singapore_foursquare_dataset_analysis/Code_Freeze_Changes/all_results/plot_hour.png", w=175)

pdf.add_page()
pdf.set_font("Arial", 'B', 13)
pdf.cell(0, 10, "Day-of-Week Distribution", ln=True)
pdf.set_font("Arial", size=11)
pdf.multi_cell(0, 7, f"JS Divergence: {js_dow:.2f}")
pdf.ln(2)
pdf.image("/home/rubesh/Desktop/sweta/google_review_collection_POI_task/singapore_foursquare_dataset_analysis/Code_Freeze_Changes/all_results/plot_dow.png", w=150)

pdf.add_page()
pdf.set_font("Arial", 'B', 13)
pdf.cell(0, 10, "Spatial Distribution", ln=True)
pdf.set_font("Arial", size=11)
pdf.multi_cell(0, 7, f"KS Latitude: {ks_lat.statistic:.2f} (p={ks_lat.pvalue:.2g}), KS Longitude: {ks_lon.statistic:.2f} (p={ks_lon.pvalue:.2g})")
pdf.ln(2)
pdf.image("/home/rubesh/Desktop/sweta/google_review_collection_POI_task/singapore_foursquare_dataset_analysis/Code_Freeze_Changes/all_results/plot_spatial.png", w=170)

pdf.add_page()
pdf.set_font("Arial", 'B', 13)
pdf.cell(0, 10, "Transaction Count per User", ln=True)
pdf.set_font("Arial", size=11)
pdf.multi_cell(0, 7, f"KS statistic: {ks_txn_count.statistic:.2f}, p-value: {ks_txn_count.pvalue:.2g}")
pdf.ln(2)
pdf.image("/home/rubesh/Desktop/sweta/google_review_collection_POI_task/singapore_foursquare_dataset_analysis/Code_Freeze_Changes/all_results/plot_txn_count.png", w=150)

pdf.add_page()
pdf.set_font("Arial", 'B', 13)
pdf.cell(0, 10, "User Journey Coherence", ln=True)
pdf.set_font("Arial", size=11)
pdf.multi_cell(0, 7, f"Real mean step: {real_avg:.0f} m, median: {real_median:.0f} m, std: {real_std:.0f} m\n"
                     f"Synthetic mean step: {synth_avg:.0f} m, median: {synth_median:.0f} m, std: {synth_std:.0f} m")

pdf.add_page()
pdf.set_font("Arial", 'B', 13)
pdf.cell(0, 10, "Joint Distribution (Category vs Hour)", ln=True)
pdf.set_font("Arial", size=11)
pdf.multi_cell(0, 7, f"JS Divergence: {js_joint:.2f} (joint structure preservation)")
pdf.ln(2)
pdf.image("/home/rubesh/Desktop/sweta/google_review_collection_POI_task/singapore_foursquare_dataset_analysis/Code_Freeze_Changes/all_results/plot_joint_real.png", w=160)
pdf.ln(5)
pdf.image("/home/rubesh/Desktop/sweta/google_review_collection_POI_task/singapore_foursquare_dataset_analysis/Code_Freeze_Changes/all_results/plot_joint_synth.png", w=160)

pdf.add_page()
pdf.set_font("Arial", 'B', 13)
pdf.cell(0, 10, "Model Utility (RF, hour+lat+lon -> category, top 6)", ln=True)
pdf.set_font("Arial", size=11)
pdf.multi_cell(0, 7, f"Train real, test synthetic: {acc_real2synth:.2f}\nTrain synthetic, test real: {acc_synth2real:.2f}")

# -- APPENDIX: ALL CODE --
pdf.add_page()
pdf.set_font("Arial", 'B', 13)
pdf.cell(0, 10, "Appendix: Full Python Code", ln=True)
pdf.set_font("Arial", size=8)
with open(__file__, "r") as src:
```

```
    for line in src:
        pdf.multi_cell(0, 4, line.rstrip())

pdf.output("/home/rubesh/Desktop/sweta/google_review_collection_POI_task/singapore_foursquare_dataset_analysis/Code_Freeze_Changes/all_results/synthetic_data_validation_full_report.pdf")
```