UNIVERSITY OF RWANDA

COLLEGE OF SCIENCE AND TECHNOLOGY
Y3 CSE

Group 4: TransConnect
Members: MUSAZA PATRICK (223019061)
MANZI NSENGA IVAN (223004392)

Module: Computing Intelligence & Applications
Date: 28th October 2025

#3

DATASET OVERVIEW

Source: Bus route information for Kigali public transport

Data Description:

- Tabular data with bus routes, stop names, and coordinates
- Multiple routes with sequential stops
- Mixed data types: text (stop names) and numeric (coordinates

Initial Data Quality Issues:

- Inconsistent stop naming conventions
- Mixed coordinate formats S139282 E30.28382)
- Potential missing values and duplicates
- Non-standardized data structure across routes
- Nature: Spatial-Temporal Data with Geographic Coordinates

1. DATA CLEANING

Purpose: Handle missing, incorrect, and inconsistent data

Application to TransConnect:

- Identify missing values
- Clean coordinate formats: Remove ‘E’, ‘S’ characters
- Convert to standard decimal degrees: -1.39282, 30.28382
- Handle missing stop names or coordinates
- Handle duplicates

```
    # Load the route details CSV file
    df = pd.read_csv('Route Details - Kabuga - Nyabugogo.csv’)

    # Clean coordinate formats - convert S/E to negative/positive
    df_clean['latitude'] = df_clean['Latitude'].str.replace('S', '-
').astype(float)
    df_clean['longitude'] = df_clean['Longitude'].str.replace('E',
'').astype(float)

    # Handle missing values
    df_clean = df_clean.dropna(subset=['latitude', 'longitude',
'Stops'])

    # Remove duplicates based on stop name and coordinates
    df_clean = df_clean.drop_duplicates(subset=['Stops',
'latitude', 'longitude'])
```

2. DATA INTEGRATION

Data Integration:

- Merge multiple route datasets into unified structure
- Create stops table (unique stops with IDs)
- Create routes table (route sequences referencing stops)

Impact:

- Efficient multi-route management

```
# Create unique stop IDs
    df_clean['stop_id'] = range(1, len(df_clean) + 1)

    # Assign route information
    df_clean['route_id'] = 'Kabuga_Nyabugogo_001'
    df_clean['route_name'] = 'Kabuga - Nyabugogo'

    # Create route-stop sequence mapping
    route_stop_sequence = {
        'Kabuga_Nyabugogo_001': df_clean[['stop_id', 'stop_sequence',
'Stops']].sort_values('stop_sequence').to_dict('records')
    }
```

3. DATA REDUCTION

```
    # Select key features for modeling
    essential_features = [
        'route_id', 'route_name', 'stop_id', 'stop_sequence',
        'latitude', 'longitude', 'Stops'
    ]

    df_reduced = df_integrated[essential_features].copy()

    # Rename columns for consistency
    df_reduced = df_reduced.rename(columns={'Stops': 'stop_name'})

    # Remove irrelevant columns that were in raw data
    columns_removed = set(df_integrated.columns) -
set(essential_features)
```

Data Reduction:

- ❑ Remove irrelevant columns: "Route Price"
- ❑ Focus on core features: stop_id, latitude, longitude
- ❑ Deduplicate identical stops across routes

Impact:

- Faster processing for route optimization
- Foundation for transfer point identification

4. DATA TRANSFORMATION

Purpose: Convert data into model-ready format

Key Transformations for TransConnect:

- Coordinate validation and normalization
- Feature engineering using Haversine formula:
- Distance between consecutive stops
- Estimated travel time between stops
- Create sequential ordering of stops

Impact::
Raw coordinates actionable features for AI model

```
R = 6371  # Earth radius in km

    # Convert degrees to radians
    lat1, lon1, lat2, lon2 = map(radians, [lat1, lon1,
lat2, lon2])
    dlat = lat2 - lat1
    dlon = lon2 - lon1

    # Haversine formula
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) *
sin(dlon/2)**2
    c = 2 * atan2(sqrt(a), sqrt(1-a))
```

```
 # Update distance to next stop
                df_reduced.loc[current_idx,
'distance_to_next'] = distance

                # Estimate travel time (assuming
average speed 25 km/h in urban areas)
                travel_time = (distance / 25) * 60  #
Convert to minutes
                df_reduced.loc[current_idx,
'estimated_travel_time_min'] = travel_time
```

5. DATA DISCRETIZATION

```
 # Discretize cumulative distance into route segments
    max_distance = df_transformed['cumulative_distance'].max()
    distance_bins = [0, max_distance*0.25, max_distance*0.5,
max_distance*0.75, max_distance + 0.1]
    distance_labels = ['Start', 'Early', 'Mid', 'Late']

    df_transformed['route_segment'] = pd.cut(
        df_transformed['cumulative_distance'],
        bins=distance_bins,
        labels=distance_labels,
        include_lowest=True
    )
```

```
# Discretize travel time estimates - handle NaN values
    time_bins = [0, 2, 5, 10, float('inf')]
    time_labels = ['Very Short', 'Short', 'Medium', 'Long']

    df_transformed['travel_time_category'] = pd.cut(
        df_transformed['estimated_travel_time_min'],
        bins=time_bins,
        labels=time_labels
    )
```

Data Discretization:

Purpose: Convert continuous data into categories for simpler analysis.

Convert continuous "distance_from_start" into categories:

- Bin 1: "Start" (0–2 km)
- Bin 2: "Early" (2–5 km)
- Bin 3: "Mid" (5–10 km)
- Bin 4: "Late" (10+ km)

Enables traffic pattern analysis by route segment

Impact: Helps analyze congestion patterns or route efficiency per time segment.

6. DATA AUGMENTATION

```
    # Add condition column to original
data first
    df_processed = df_processed.copy()
    df_processed['condition'] = 'normal'
    augmented_data = [df_processed]  #
Start with original data
```

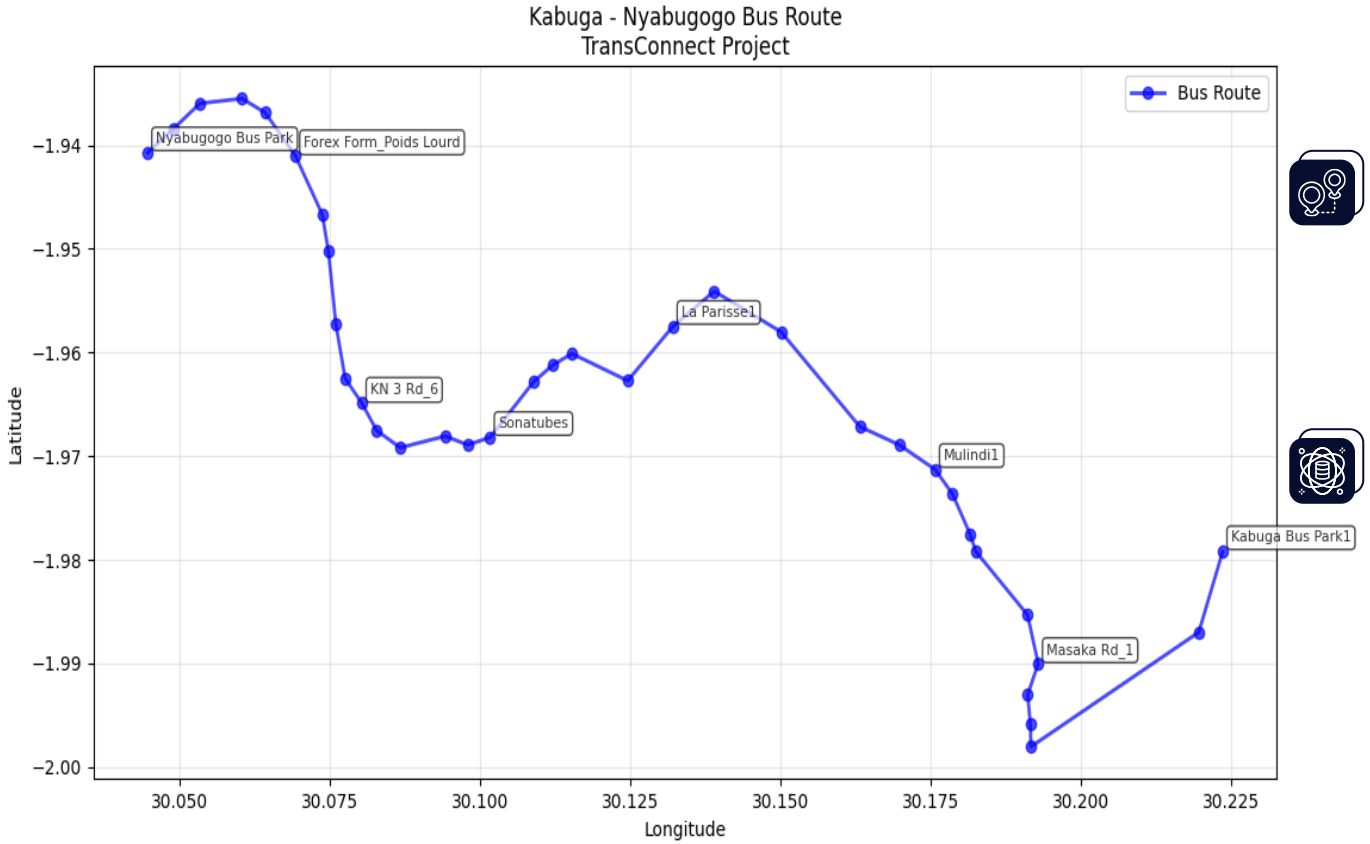
Data Augmentation:

- Generate synthetic trips with varying travel times
- Simulate different conditions: rush hour, weather, e
- Add temporal features: time_of_day, day_of_week

Purpose: Enrich the dataset with synthetic variations to improve prediction accuracy.

Impact:
More robust and reliable system, even under uncertain or missing data conditions.

5. VISUALIZATION & IMPACT



Before Preprocessing:

- Messy, overlapping points on map
- Inconsistent stop representations
- Unreliable spatial relationships

After Preprocessing:

- Clean, sequential route visualization
- Accurate stop positioning
- Calculated distances and travel times

Impact on Model Readiness:

- Clean spatial data for route optimization
- Engineered features for arrival prediction
- Structured data for CI algorithms (Genetic Algorithms, ML)
- Foundation for real-time intelligence system

CHALLENGES & LESSONS LEARNED

Challenges Encountered:

- Interpreting mixed coordinate formats.
- Standardizing stop names without official references.
- Handling incomplete route information.
- Ensuring spatial accuracy for distance calculations

Lessons Learned:

- Data preprocessing is crucial (80% of data science work)
- Clean, structured data enables effective CI applications
- Domain knowledge essential for data interpretation
- Quality preprocessing directly impacts model performance

Conclusion: The preprocessed dataset is now ready for: Machine learning model training, Route optimization algorithms, Real-time arrival prediction systems, Intelligent transportation analytics.

TRANSCONNECT DATA PREPROCESSING SUMMARY
==========
Original stops: 36
Cleaned stops: 36
Total route distance: 26.07 km
Total travel time: 62.6 minutes
Augmented dataset size: 216 rows
Conditions simulated: 6

Summary for Transconnect for Route ***"Kabuga – Nyabugogo"***

CODE DEMONSTRATION STRUCTURE

Python Script Sections:

- Data Loading & Initial Assessment
- Data Cleaning Operations
- Data Integration & Deduplication
- Feature Engineering (Distance Calculations)
- Data Transformation & Discretization
- Data Augmentation for Model Training
- Visualization of Results

Tools & Libraries:

- Pandas for data manipulation
- NumPy for numerical operations
- Scikit-learn for preprocessing
- Haversine for distance calculations
- Matplotlib/Plotly for visualization
- Folium for map-based displays

Codes:
https://drive.google.com/file/d/120uvRQOfj2I6KP-9buP23hwFub0JSyPO/view?usp=sharing
Dataset:
https://docs.google.com/spreadsheets/d/1nQSVadk_r9I_5k6FVnBV3_UW04ITMIJ-7orBFDDlBv8/edit?usp=sharing

END