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: 11th December 2025

Introduction

TransConnect is a smart AI web-based transit prediction platform designed to provide:

- Predicted-time bus movement tracking
- Predictive travel times using machine learning
- More accurate ETA (Estimated Time of Arrival) predictions
- The system improves commuter experience by reducing uncertainty and wait time

Project Objectives

- Provide accurate travel time predictions for public transport
- Reduce commuter waiting time and uncertainty
- Offer traffic-adjusted ETA during rush hours
- Improve the efficiency of the (Kabuga–Nyabugogo route) case study
- Use machine learning to support data-driven transport planning

Background

Public transport users in Rwanda often experience:

- Unpredictable delays
- Lack of predicted-time transit information
- Traffic-based inconsistencies, especially in mornings and evenings

- The Kabuga–Nyabugogo case study route is a major corridor, making it ideal for predictive analytics

Rationale

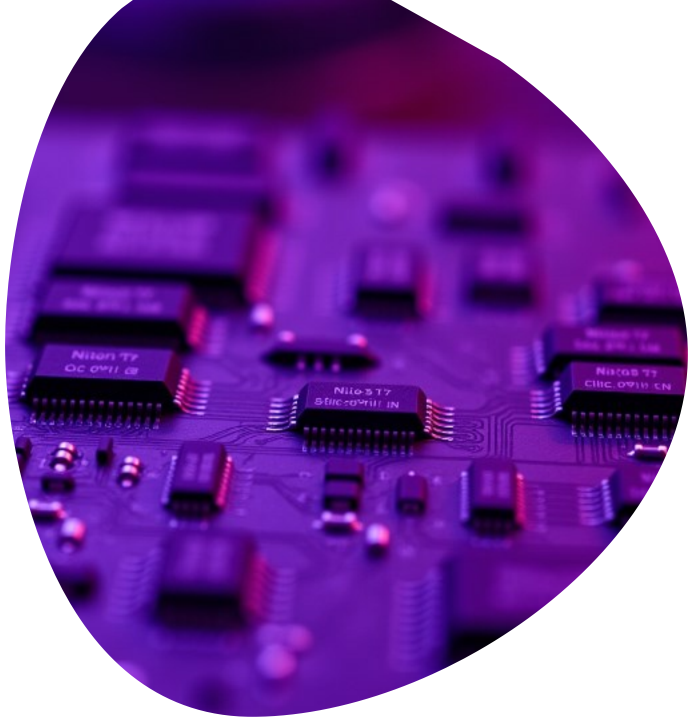
Why TransConnect?

- Passengers need accurate ETA to plan their journeys
- Transport operators need better route visibility
- Machine Learning can reduce uncertainty and enhance mobility
- Real-time predictions = better satisfaction + optimized transport flow

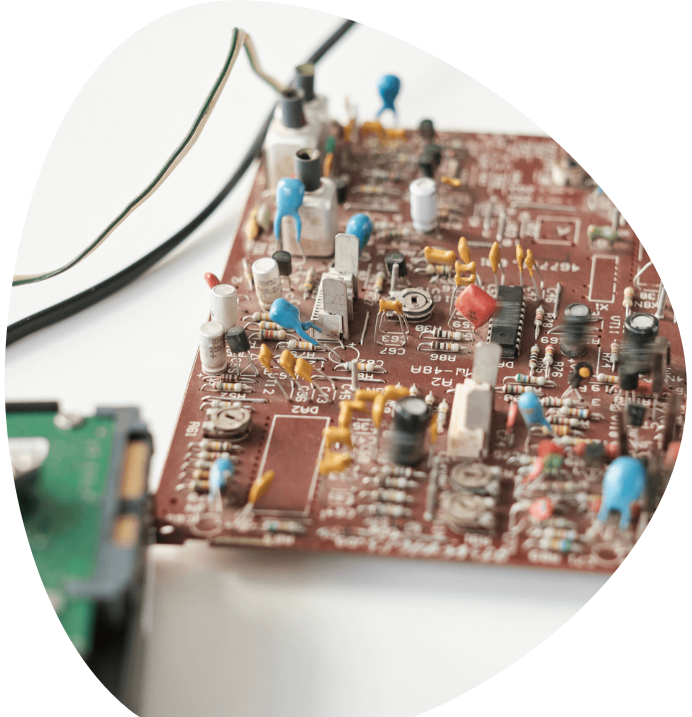
System Features

- Real-time GPS updates (to be implemented later)
- Distance-to-destination calculation
- ML-based ETA prediction (Random Forest, Gradient Boosting, Regression & LSTM).
- Traffic condition classification (normal, rush hour)
- Remaining distance + arrival time display

Machine Learning Models



- Models evaluated:
  - Linear Regression
  - Decision Tree
  - Random Forest
  - Gradient Boosting
  - Long Short-Term Memory
- Best model is automatically selected based on highest R^2 and lowest RMSE



Expected Impact

- Improved commuter planning
- Reduced delays and congestion
- Increased operational transparency
- Smart city readiness for public transport

Conclusion

TransConnect demonstrates how Machine Learning can transform public transportation by:

- Providing real-time predictive insights
- Enhancing user satisfaction
- Supporting data-driven decision making
- Contributing to Rwanda's smart mobility vision

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 (coordina

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 Coordinat

Route Details

File Edit View Insert Format Data Tools Extensions Help

Share

100% $ % .0 .00 123 Arial 11 B I A

H16 fx

Stops	Latitude	Longitude
Kabuga Bus Park1	S1.97922	E30.22352
Masaka Hospital	S1.98699	E30.21955
Masaka Bus Terminal	S1.99804	E30.19164
Masaka Rd_2	S1.99587	E30.19155
Masaka Health Centre	S1.99302	E30.19111
Masaka Rd_1	S1.99001	E30.19285
At 19_Inyange1	S1.98532	E30.19108
KK 3 Rd_15	S1.97920	E30.18248
Kibaya_Minagri	S1.97749	E30.18150
Bus Route 107_3	S1.97365	E30.17860
Mulindi1	S1.97129	E30.17572
KK 3 Rd_11	S1.96891	E30.16975
At 15 Nderal	S1.96715	E30.16319
SEZ B2	S1.95801	E30.15010
Kigali Parents School2	S1.95409	E30.13899
La Parisse1	S1.95748	E30.13211
Ku Cyamutzig	S1.96272	E30.12460
Prince House2	S1.96011	E30.11533
Alpha Palace Hotel	S1.96120	E30.11216
Good Year2	S1.96281	E30.10898
Sonatubes4	S1.96820	E30.10170
Bralirwa3	S1.96891	E30.09799
Amasezerano	S1.96807	E30.09426
Rwandex3	S1.96918	E30.08676
Kwa Mironko2	S1.96752	E30.08285
KN 3 Rd_6	S1.96489	E30.08039
Volta Super1	S1.96249	E30.07761
Ku Mazi1	S1.95730	E30.07608
Chez Rasta	S1.95026	E30.07484

Kabuga - Nyabugogo | Nyabugogo - Kabuga | Rubirizi - Downtown | Downtown - Rubirizi | Kibaya - Kanombe | Kanombe - Kibaya | MUSAVE - REMERA

DATA PREPOCESSING STEPS

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)

```
# 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')
    }
```

Impact:

- Efficient multi-route management

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
```

HAVERSINE FORMULA

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

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
```

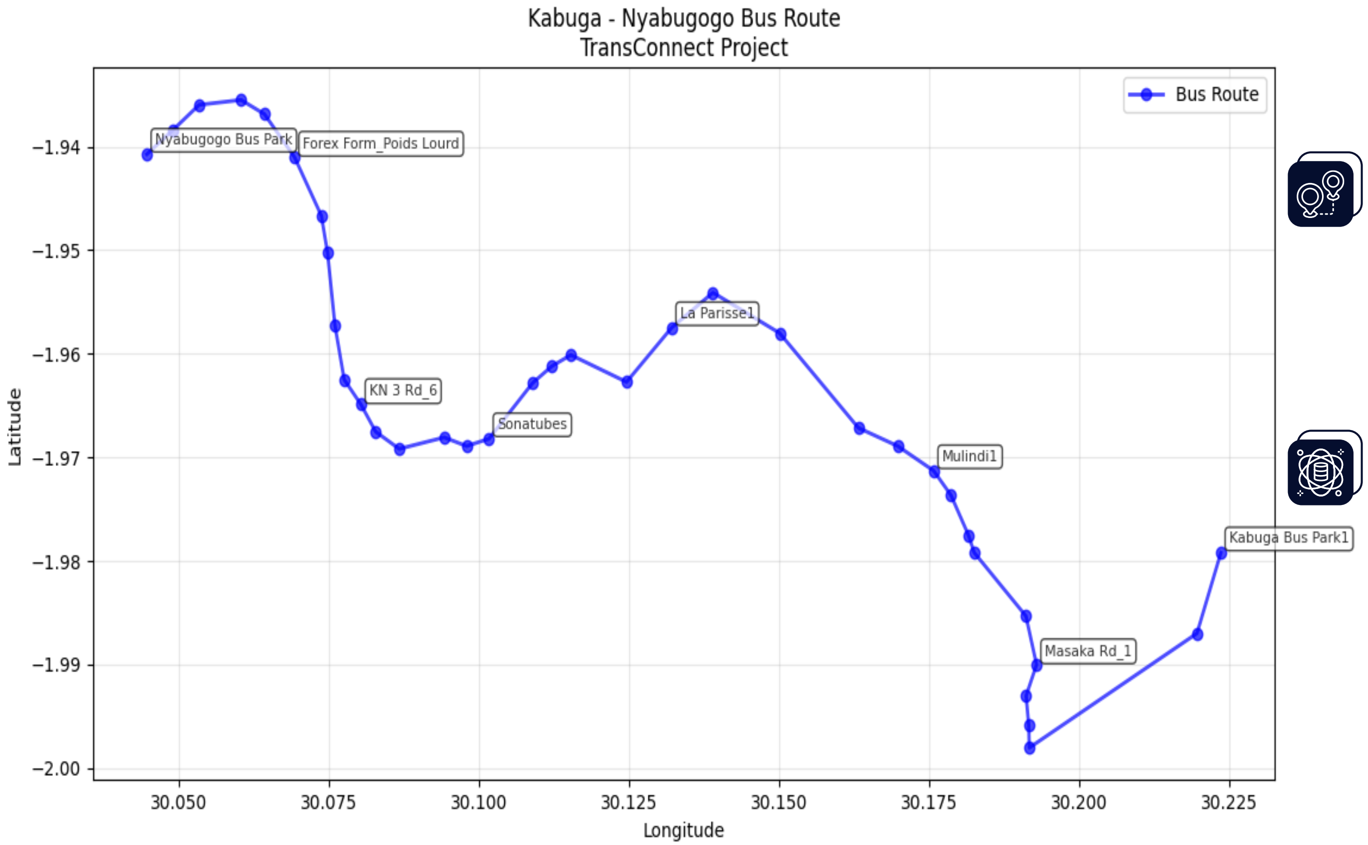
Data Augmentation:

- Generate synthetic trips with varying travel times
- Simulate different conditions: rush hour, weather, events
- 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

BEFORE – DATA PREPROCESSING

transconnect_clean_route * | Route Details - Kabuga - Nyabugogo *

Stops	Latitude	Longitude
Kabuga Bus Park1	S1.97922	E30.22352
Masaka Hospital	S1.98699	E30.21955
Masaka Bus Terminal	S1.99804	E30.19164
Masaka Rd_2	S1.99587	E30.19155
Masaka Health Centre	S1.99302	E30.19111
Masaka Rd_1	S1.99001	E30.19285
At 19_Inyange1	S1.98532	E30.19108
KK 3 Rd_15	S1.97920	E30.18248
Kibaya_Minagri	S1.97749	E30.18150
Bus Route 107_3	S1.97365	E30.17860
Mulindi1	S1.97129	E30.17572
KK 3 Rd_11	S1.96891	E30.16975
At 15 Ndera1	S1.96715	E30.16319
SEZ B2	S1.95801	E30.15010
Kigali Parents School2	S1.95409	E30.13899
La Parisse1	S1.95748	E30.13211
Ku Cyamutzig	S1.96272	E30.12460
Prince House2	S1.96011	E30.11533
Alpha Palace Hotel	S1.96120	E30.11216
Good Year2	S1.96281	E30.10898
Sonatubes4	S1.96820	E30.10170
Bralirwa3	S1.96891	E30.09799
Amasezerano	S1.96807	E30.09426
Rwandex3	S1.96918	E30.08676
Kwa Mironko2	S1.96752	E30.08285
KN 3 Rd_6	S1.96489	E30.08039
Volta Super1	S1.96249	E30.07761
Ku Mazi1	S1.95730	E30.07608
Chez Rasta	S1.95026	E30.07484
One Love	S1.94667	E30.07381
Forex Form_Poids Lourd	S1.94101	E30.06934
Kigali Gaz_Kinamba	S1.93686	E30.06433

Route Details - Kabuga - Nyabug

DATA PREPOCESSING ACTIONS

```
=== DATA ASSESSMENT ===
Dataset shape: (36, 3)
Columns: ['Stops', 'Latitude', 'Longitude']

First 5 rows:
                 Stops  Latitude  Longitude
0      Kabuga Bus Park1  S1.97922  E30.22352
1       Masaka Hospital  S1.98699  E30.21955
2   Masaka Bus Terminal  S1.99804  E30.19164
3           Masaka Rd_2  S1.99587  E30.19155
4  Masaka Health Centre  S1.99302  E30.19111

Missing values:
Stops        0
Latitude     0
Longitude    0
dtype: int64
```

```
=== DATA CLEANING ===
Original stops: 36
Cleaned stops: 36
Sample cleaned data:
                 Stops  latitude  longitude
0      Kabuga Bus Park1  -1.97922   30.22352
1       Masaka Hospital  -1.98699   30.21955
2   Masaka Bus Terminal  -1.99804   30.19164
3           Masaka Rd_2  -1.99587   30.19155
4  Masaka Health Centre  -1.99302   30.19111
```

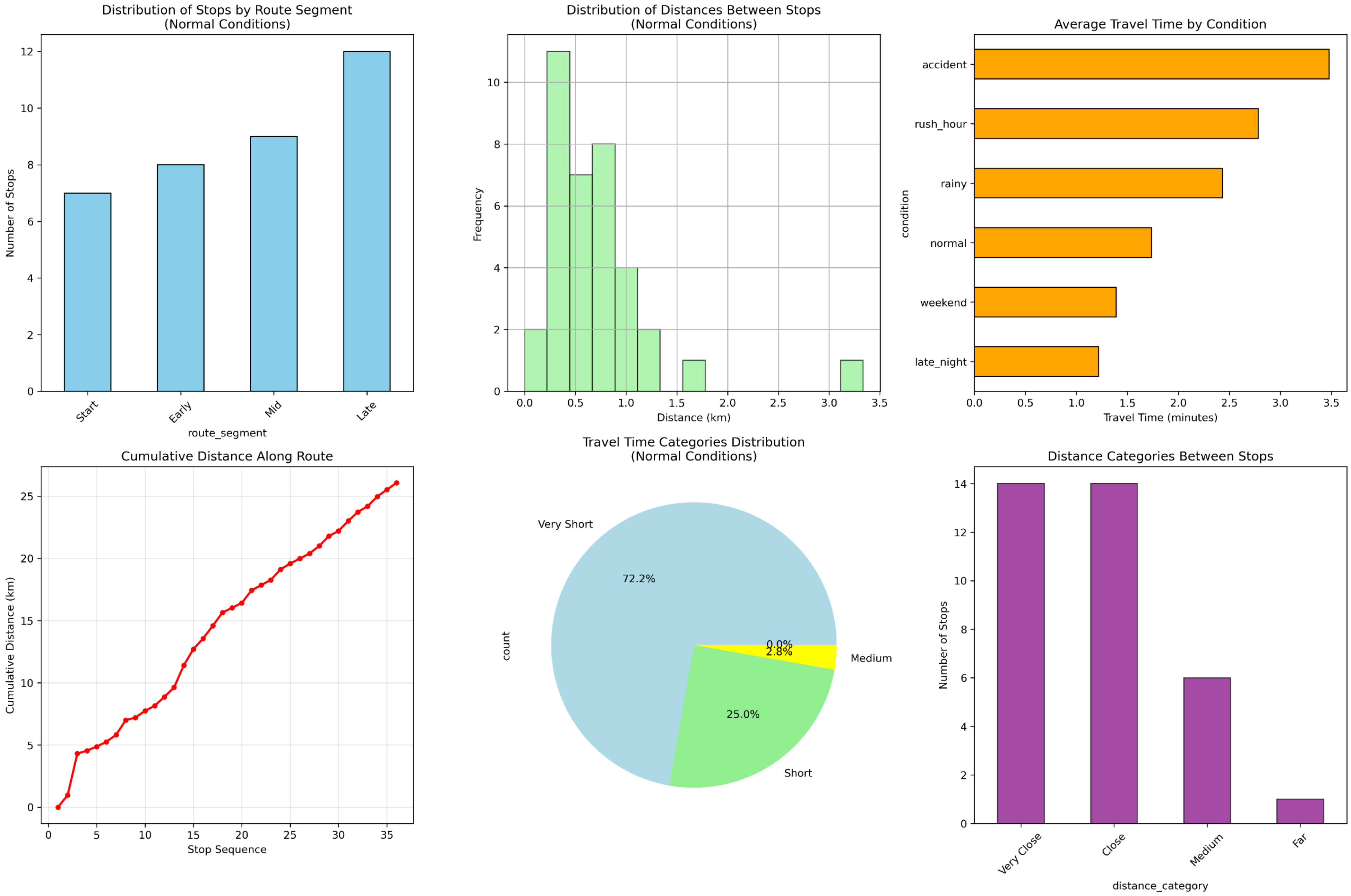
```
=== DATA INTEGRATION ===
Integrated dataset shape: (36, 9)
Routes processed: 1
Stops in route: 36
```

```
=== DATA REDUCTION ===
Columns removed: {'Longitude', 'Latitude'}
Reduced dataset shape: (36, 7)
Final features: ['route_id', 'route_name', 'stop_id', 'stop_sequence', 'latitude', 'longitude', 'stop_name']

=== FEATURE ENGINEERING ===
Processing route: Kabuga_Nyabugogo_001 with 36 stops
Feature engineering completed:
- Total route distance: 26.07 km
- Total estimated time: 62.6 minutes
- Average distance between stops: 0.72 km
- Average time between stops: 1.7 min
```

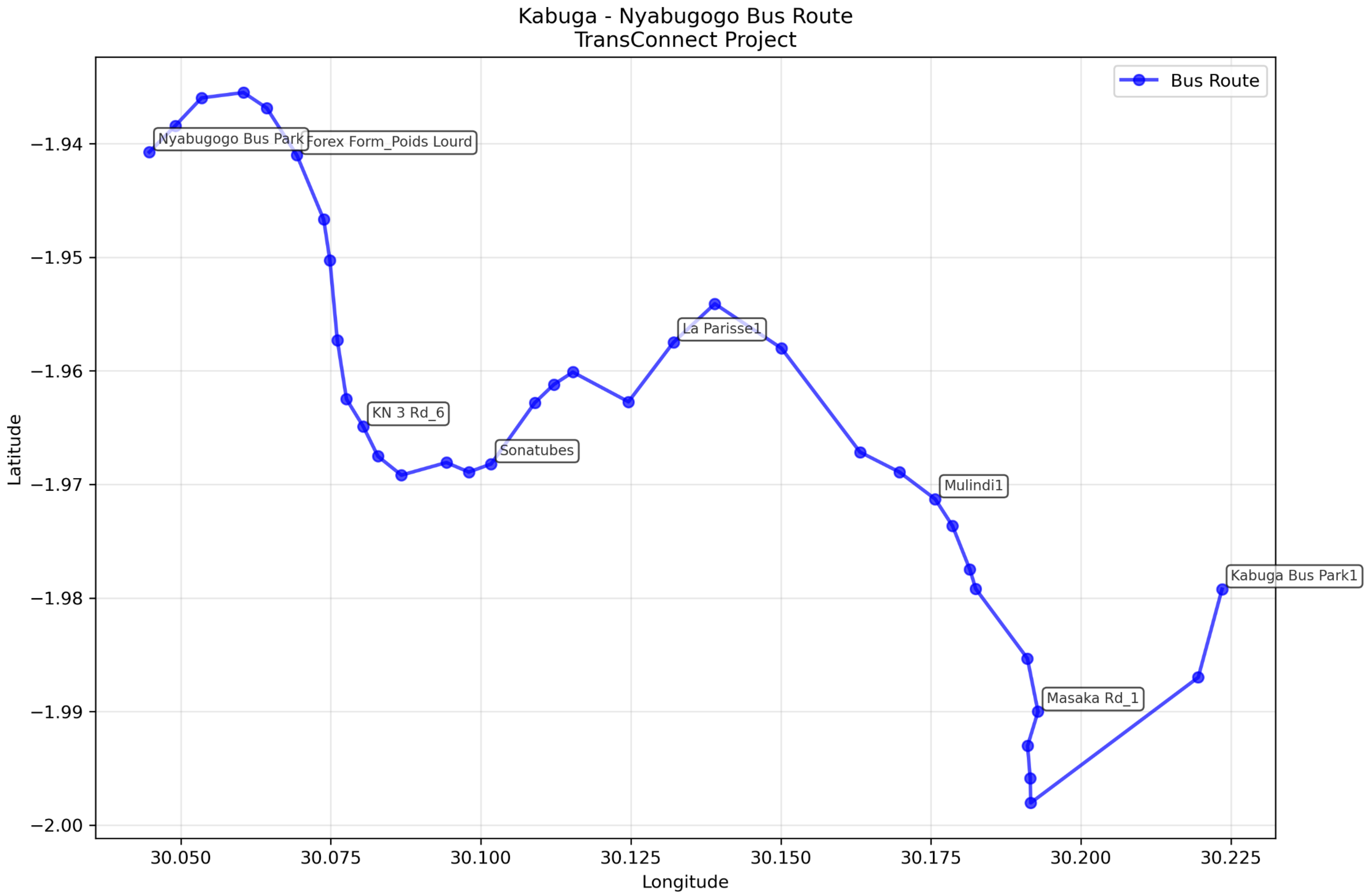
```
=== DATA DISCRETIZATION ===
Discretization completed:

Route segments distribution:
route_segment
Start      7
Early      8
Mid        9
Late      12
Name: count, dtype: int64

Travel time categories:
travel_time_category
Very Short    26
Short          9
Medium         1
Long           0
Name: count, dtype: int64
```

```
Route segments distribution:
route_segment
Start    42
Early    48
Mid      59
Late     67
Name: count, dtype: int64

Travel time categories:
travel_time_category
Very Short    135
Short          68
Medium         10
Long            3
Name: count, dtype: int64
```

VISUALIZATION



Distribution of Stops by Route Segment
(Normal Conditions)
Number of Stops
route_segment
Start
Early
Mid
Late
Distribution of Distances Between Stops
(Normal Conditions)
Frequency
Distance (km)
Average Travel Time by Condition
condition
accident
rush_hour
rainy
normal
weekend
late_night
Travel Time (minutes)
Cumulative Distance Along Route
Cumulative Distance (km)
Stop Sequence
Travel Time Categories Distribution
(Normal Conditions)
count
Very Short
72.2%
Short
25.0%
Medium
2.8%
0.0%
Distance Categories Between Stops
Number of Stops
distance_category
Very Close
Close
Medium
Far



Kabuga - Nyabugogo Bus Route
TransConnect Project
Bus Route
Nyabugogo Bus Park
Forex Form_Poids Lourd
La Parisse1
KN 3 Rd_6
Sonatubes
Mulindi1
Kabuga Bus Park1
Masaka Rd_1
Latitude
Longitude
−1.94
−1.95
−1.96
−1.97
−1.98
−1.99
−2.00
30.050
30.075
30.100
30.125
30.150
30.175
30.200
30.225

AFTER – CLEAN PREPROCESSING

transconnect_clean_route

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	route_i	route_name	stop_id	stop_se	latitude	longituc	stop_name	distance_to_	cumulative_dist	estimated_travel_tir	route_segment	travel_time_cat	distance_category	condition
2	Kabuga_	Kabuga - Nyabu	1	1	-1.979	30.224	Kabuga Bus P	0.970107559	0	2.328258142	Start	Short	Close	normal
3	Kabuga_	Kabuga - Nyabu	2	2	-1.987	30.22	Masaka Hospi	3.33608671	0.970107559	8.006608105	Start	Medium	Far	normal
4	Kabuga_	Kabuga - Nyabu	3	3	-1.998	30.192	Masaka Bus T	0.24150018	4.306194269	0.579600431	Start	Very Short	Very Close	normal
5	Kabuga_	Kabuga - Nyabu	4	4	-1.996	30.192	Masaka Rd_2	0.320655506	4.547694449	0.769573213	Start	Very Short	Very Close	normal
6	Kabuga_	Kabuga - Nyabu	5	5	-1.993	30.191	Masaka Healtl	0.386537039	4.868349955	0.927688894	Start	Very Short	Very Close	normal
7	Kabuga_	Kabuga - Nyabu	6	6	-1.99	30.193	Masaka Rd_1	0.5573654	5.254886994	1.33767696	Start	Very Short	Close	normal
8	Kabuga_	Kabuga - Nyabu	7	7	-1.985	30.191	At 19_Inyange	1.173229832	5.812252394	2.815751596	Start	Short	Medium	normal
9	Kabuga_	Kabuga - Nyabu	8	8	-1.979	30.182	KK 3 Rd_15	0.219123292	6.985482226	0.525895901	Early	Very Short	Very Close	normal
10	Kabuga_	Kabuga - Nyabu	9	9	-1.977	30.182	Kibaya Minagr	0.534957457	7.204605518	1.283897896	Early	Very Short	Close	normal
11	Kabuga_	Kabuga - Nyabu	10	10	-1.974	30.179	Bus Route 107	0.413880807	7.739562975	0.993313937	Early	Very Short	Very Close	normal
12	Kabuga_	Kabuga - Nyabu	11	11	-1.971	30.176	Mulindi1	0.714276414	8.153443782	1.714263393	Early	Very Short	Close	normal
13	Kabuga_	Kabuga - Nyabu	12	12	-1.969	30.17	KK 3 Rd_11	0.754819859	8.867720195	1.811567663	Early	Very Short	Close	normal
14	Kabuga_	Kabuga - Nyabu	13	13	-1.967	30.163	At 15 Ndera1	1.774549572	9.622540055	4.258918973	Early	Short	Medium	normal
15	Kabuga_	Kabuga - Nyabu	14	14	-1.958	30.15	SEZ B2	1.309339478	11.39708963	3.142414747	Early	Short	Medium	normal
16	Kabuga_	Kabuga - Nyabu	15	15	-1.954	30.139	Kigali Parents	0.852447952	12.7064291	2.045875085	Early	Short	Close	normal
17	Kabuga_	Kabuga - Nyabu	16	16	-1.957	30.132	La Parisse1	1.017854079	13.55887706	2.442849791	Mid	Short	Medium	normal
18	Kabuga_	Kabuga - Nyabu	17	17	-1.963	30.125	Ku Cyamutzig	1.070272592	14.57673114	2.568654221	Mid	Short	Medium	normal
19	Kabuga_	Kabuga - Nyabu	18	18	-1.96	30.115	Prince House2	0.372548431	15.64700373	0.894116234	Mid	Very Short	Very Close	normal
20	Kabuga_	Kabuga - Nyabu	19	19	-1.961	30.112	Alpha Palace I	0.396151284	16.01955216	0.950763083	Mid	Very Short	Very Close	normal
21	Kabuga_	Kabuga - Nyabu	20	20	-1.963	30.109	Good Year2	1.006840164	16.41570344	2.416416393	Mid	Short	Medium	normal
22	Kabuga_	Kabuga - Nyabu	21	21	-1.968	30.102	Sonatubes	0.419780487	17.42254361	1.007473168	Mid	Very Short	Very Close	normal
23	Kabuga_	Kabuga - Nyabu	22	22	-1.969	30.098	Bralirwa3	0.42490554	17.84232409	1.019773296	Mid	Very Short	Very Close	normal
24	Kabuga_	Kabuga - Nyabu	23	23	-1.968	30.094	Amasezerano	0.842559119	18.26722963	2.022141885	Mid	Short	Close	normal
25	Kabuga_	Kabuga - Nyabu	24	24	-1.969	30.087	Rwandex	0.472096312	19.10978875	1.133031148	Mid	Very Short	Very Close	normal
26	Kabuga_	Kabuga - Nyabu	25	25	-1.968	30.083	Kwa Mironko2	0.400322989	19.58188507	0.960775173	Late	Very Short	Very Close	normal
27	Kabuga_	Kabuga - Nyabu	26	26	-1.965	30.08	KN 3 Rd_6	0.408243289	19.98220805	0.979783893	Late	Very Short	Very Close	normal
28	Kabuga_	Kabuga - Nyabu	27	27	-1.962	30.078	Volta Super1	0.601627875	20.39045134	1.443906901	Late	Very Short	Close	normal
29	Kabuga_	Kabuga - Nyabu	28	28	-1.957	30.076	Ku Mazi1	0.794848627	20.99207922	1.907636704	Late	Very Short	Close	normal
30	Kabuga_	Kabuga - Nyabu	29	29	-1.95	30.075	Chez Rasta	0.415276558	21.78692784	0.99666374	Late	Very Short	Very Close	normal
31	Kabuga_	Kabuga - Nyabu	30	30	-1.947	30.074	One Love	0.801787986	22.2022044	1.924291166	Late	Very Short	Close	normal
32	Kabuga_	Kabuga - Nyabu	31	31	-1.941	30.069	Forex Form_P	0.723142135	23.00399239	1.735541123	Late	Very Short	Close	normal
33	Kabuga_	Kabuga - Nyabu	32	32	-1.937	30.064	Kigali Gaz_Kin	0.45730886	23.72713452	1.097541265	Late	Very Short	Very Close	normal

transconnect_clean_route

CHALLENGES & LESSONS LEARNED

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

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

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

AI MODELS

Selected Models for TransConnect

1. Linear Regression – simple baseline model
2. Random Forest – stronger non-linear model

Linear Regression – What Was Completed

Provides a simple baseline for ETA prediction

Key Features:

- Simple, Fast, and Computationally Efficient
- Highly Interpretable
- Works Well on Linearly Related Data (travel time, distance, time of day, road features)

- **Advantages**: easy, fast, interpretable
- **Disadvantages**: weak with non-linear route patterns

Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
from sklearn.preprocessing import StandardScaler
```

Set Model

```
def __init__(self):
    self.model = LinearRegression()
    self.results = {}
    self.scaler = StandardScaler()
```

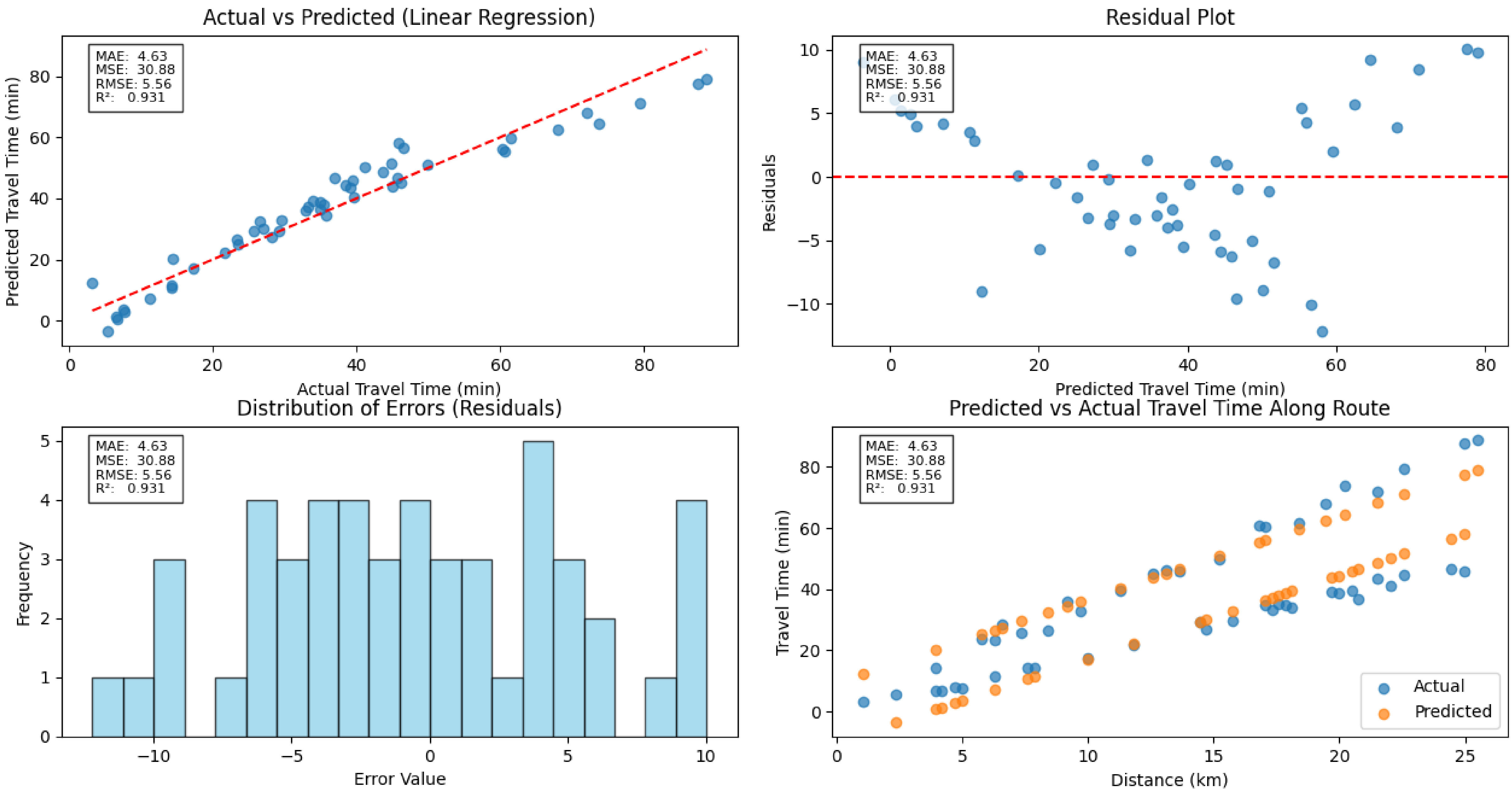
Train Dataset

```
X_train, X_test, y_train, y_test =
 train_test_split(X, y, test_size=0.2, random_state=42)

X_train_scaled = self.scaler.fit_transform(X_train)
X_test_scaled = self.scaler.transform(X_test)

self.model.fit(X_train_scaled, y_train)
y_pred = self.model.predict(X_test_scaled)
```

Performance Metric

```
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
```

VISUALIZATION



Actual vs Predicted (Linear Regression)
MAE: 4.63
MSE: 30.88
RMSE: 5.56
R²: 0.931
Predicted Travel Time (min)
Actual Travel Time (min)
Residual Plot
Residuals
Predicted Travel Time (min)
Distribution of Errors (Residuals)
Frequency
Error Value
Predicted vs Actual Travel Time Along Route
Travel Time (min)
Distance (km)
Actual
Predicted

Random Forest – What Was Completed

Handles non-linear transit patterns well

Key Features:

- Features do not need normalization or standardization (unlike Linear Regression)
- Provides Feature Importance
- High Accuracy on Tabular Data

- **Advantages**: high accuracy, robust
- **Disadvantages**: slower, harder to interpret

CODE SNIPPET

Library

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
```

Model Fitting

```python
def __init__(self):
    self.model = RandomForestRegressor(
        n_estimators=150,
        random_state=42,
        n_jobs=-1
    )

    self.results = {}
```

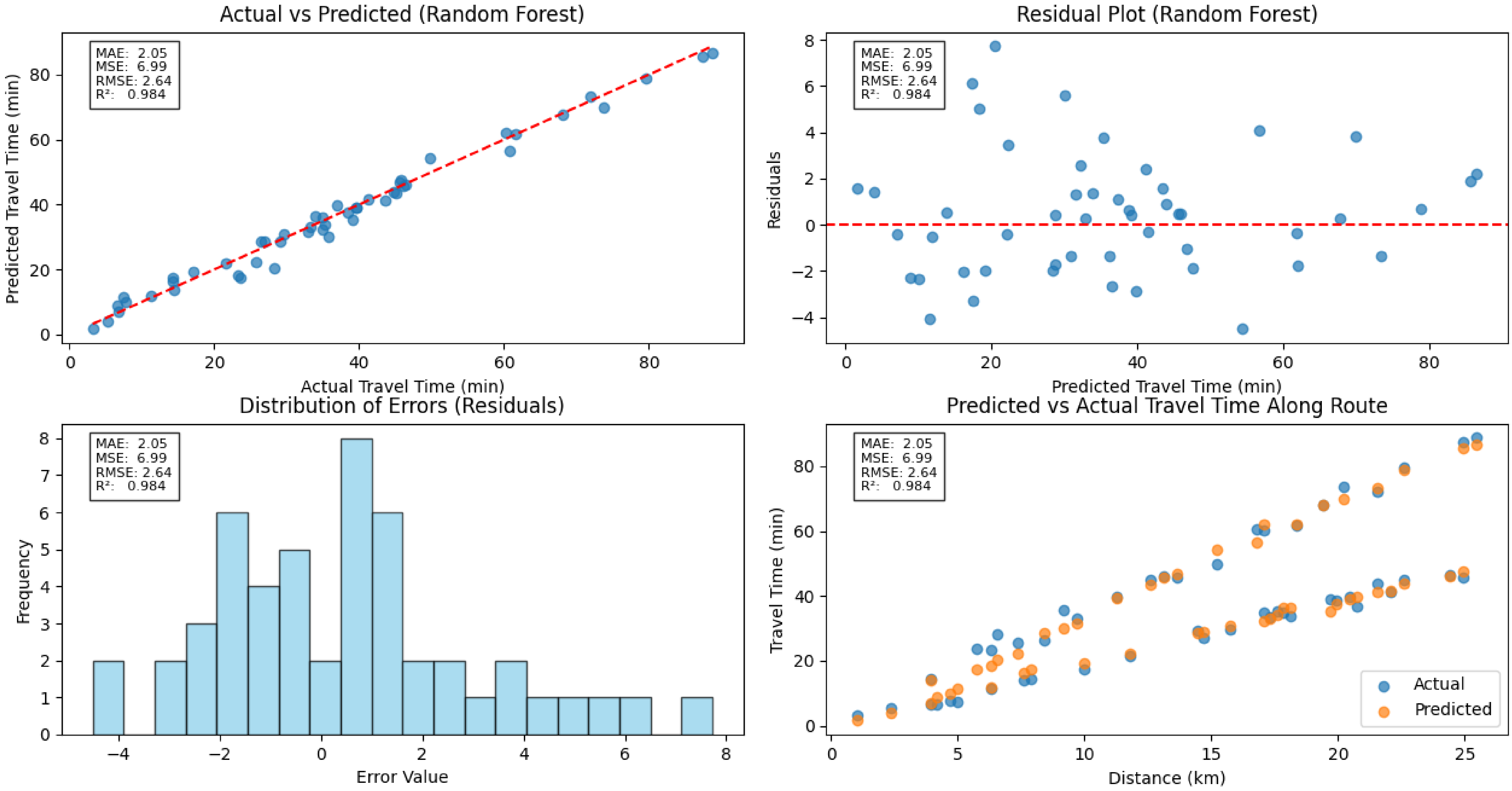
Train Model

```python
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

self.model.fit(X_train, y_train)
y_pred = self.model.predict(X_test)
```

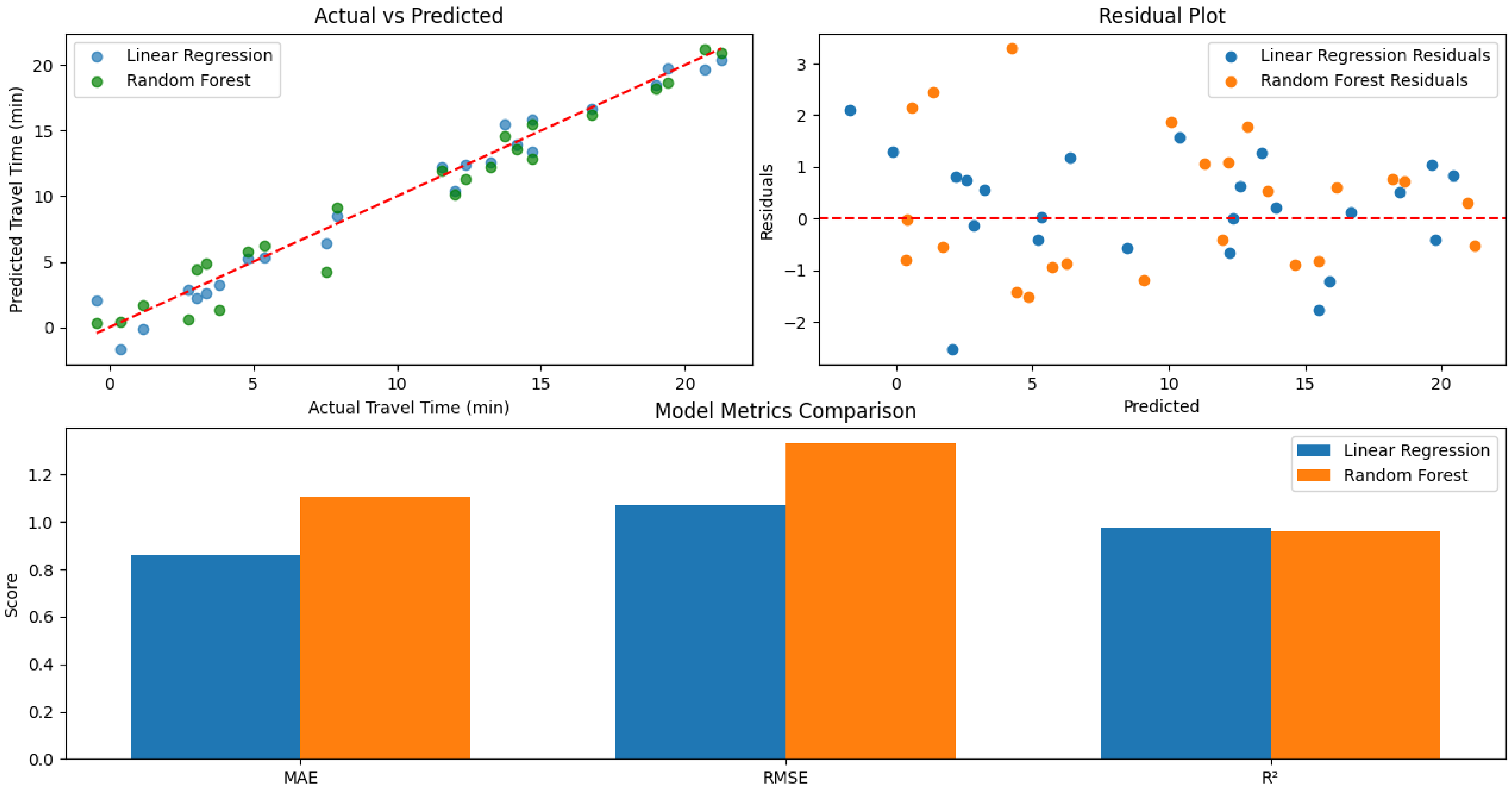
Performance Metric

```python
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
```

VISUALIZATION


Actual vs Predicted (Random Forest)
MAE: 2.05
MSE: 6.99
RMSE: 2.64
R²: 0.984
Predicted Travel Time (min)
Actual Travel Time (min)
Residual Plot (Random Forest)
MAE: 2.05
MSE: 6.99
RMSE: 2.64
R²: 0.984
Residuals
Predicted Travel Time (min)
Distribution of Errors (Residuals)
MAE: 2.05
MSE: 6.99
RMSE: 2.64
R²: 0.984
Frequency
Error Value
Predicted vs Actual Travel Time Along Route
MAE: 2.05
MSE: 6.99
RMSE: 2.64
R²: 0.984
Travel Time (min)
Distance (km)
Actual
Predicted

LINEAR REGRESSION
VS
RANDOM FOREST



Actual vs Predicted
Linear Regression
Random Forest
Predicted Travel Time (min)
Actual Travel Time (min)
Residual Plot
Linear Regression Residuals
Random Forest Residuals
Residuals
Predicted
Model Metrics Comparison
Linear Regression
Random Forest
Score
MAE
RMSE
R²

Cont...

Selected Models for TransConnect

3. Gradient Boosting Regression – Ensemble model of many decision trees

4. Decision Tree – supervised non-linear model

CASE STUDY

ROUTE

Kabuga - Nyabugogo

Gradient Boosting – What Was Completed

- Learns sequentially: each tree corrects previous errors
- Very powerful for structured data

Key Features:

High-accuracy ETA prediction

Modeling traffic effects on travel time

Works Well on Linearly Related Data (travel time, distance, time of day, road features)

Advantages:

- High prediction accuracy
- Handles complex relationships
- Great for time-based forecasting

Disadvantages:

- Slower training
- Sensitive to hyperparameters
- Can overfit if not tuned

Code Snippet

Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score
```

Set Model

```
model = GradientBoostingRegressor(
    n_estimators=300,
    learning_rate=0.05,
    max_depth=4,
    random_state=42
)
```

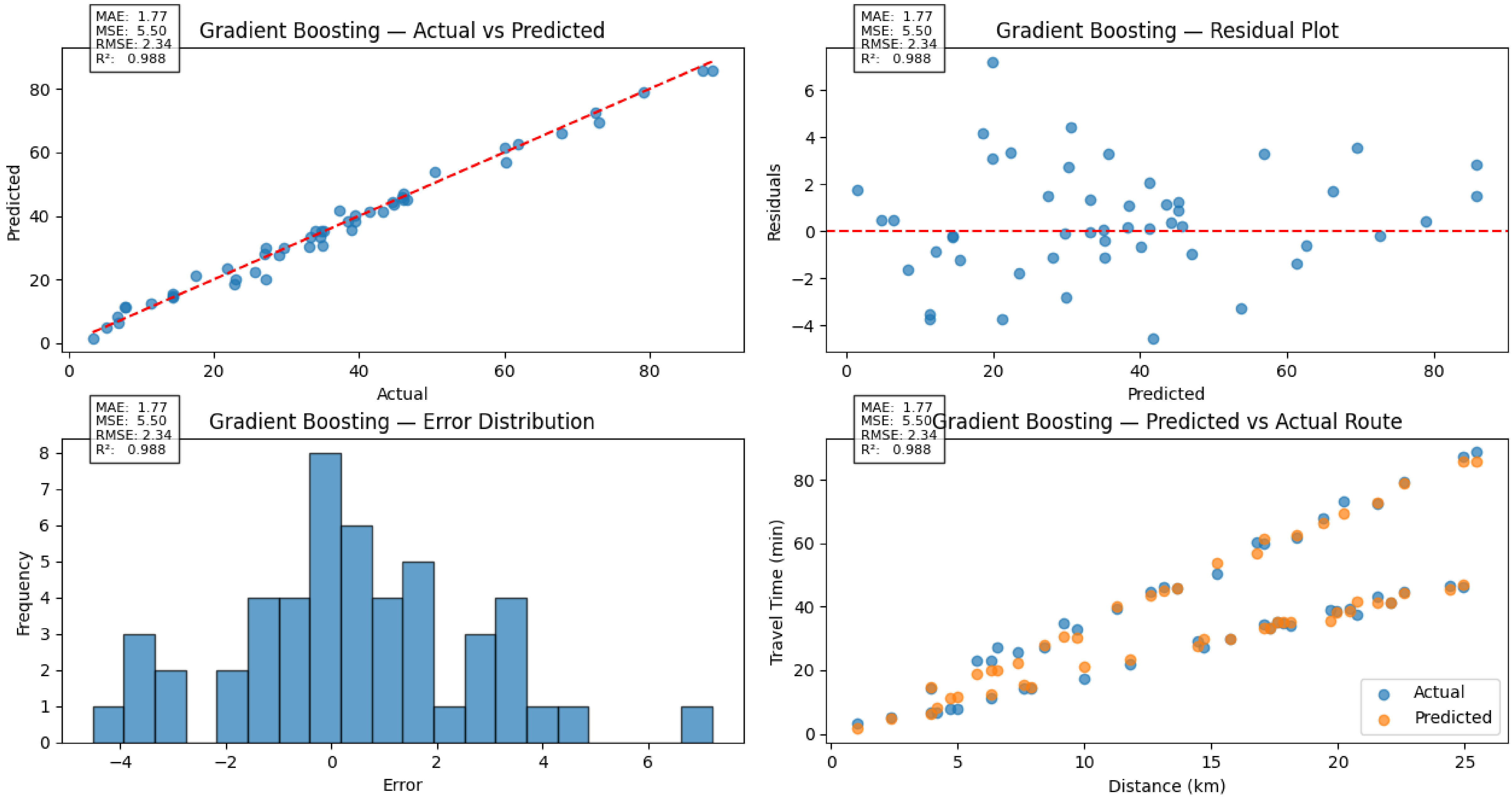
Train Dataset

```
X = df[["distance", "is_traffic"]]
y = df["travel_time"]


X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=42)


model.fit(X_train, y_train)
y_pred = model.predict(X_test)
residuals = y_test - y_pred
```

Performance Metric

```
mae  = mean_absolute_error(y_test, y_pred)
mse  = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2   = r2_score(y_test, y_pred)
```

VISUALIZATION


MAE: 1.77
MSE: 5.50
RMSE: 2.34
R²: 0.988
Gradient Boosting — Actual vs Predicted
Predicted
Actual
Gradient Boosting — Residual Plot
Residuals
Predicted
Gradient Boosting — Error Distribution
Frequency
Error
Gradient Boosting — Predicted vs Actual Route
Travel Time (min)
Distance (km)
Actual
Predicted

Decision Tree – What Was Completed

- Supervised non-linear model
- Splits data into rules based on thresholds
- Easy to interpret and visualize

Key Features:

- Predicting travel time by condition (traffic vs normal)
- Detecting unusual stops or delays
- Fast to train, good for quick predictions

- **Advantages**:
  - Simple and interpretable
  - Handles non-linear data
  - Works well with small datasets

- **Disadvantages**:
  - Can overfit without pruning
  - Unstable with small changes in data

CODE SNIPPET

Library

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score
```

Model Fitting

```python
model = DecisionTreeRegressor(max_depth=6,
                              random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
residuals = y_test - y_pred
```

Train Model

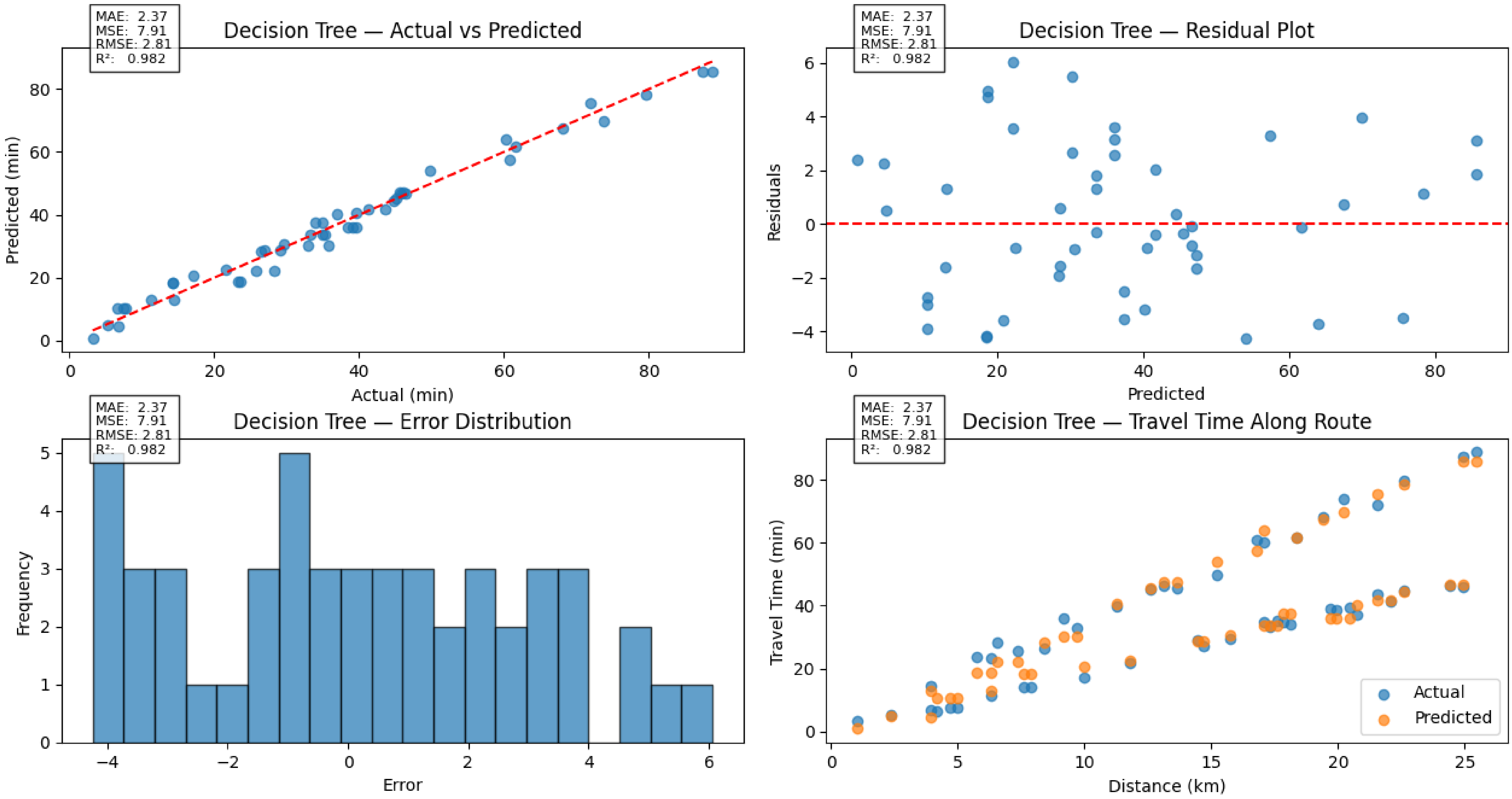
```python
X = df[["distance", "is_traffic"]]
y = df["travel_time"]
```

```python
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=42)
```

Performance Metric

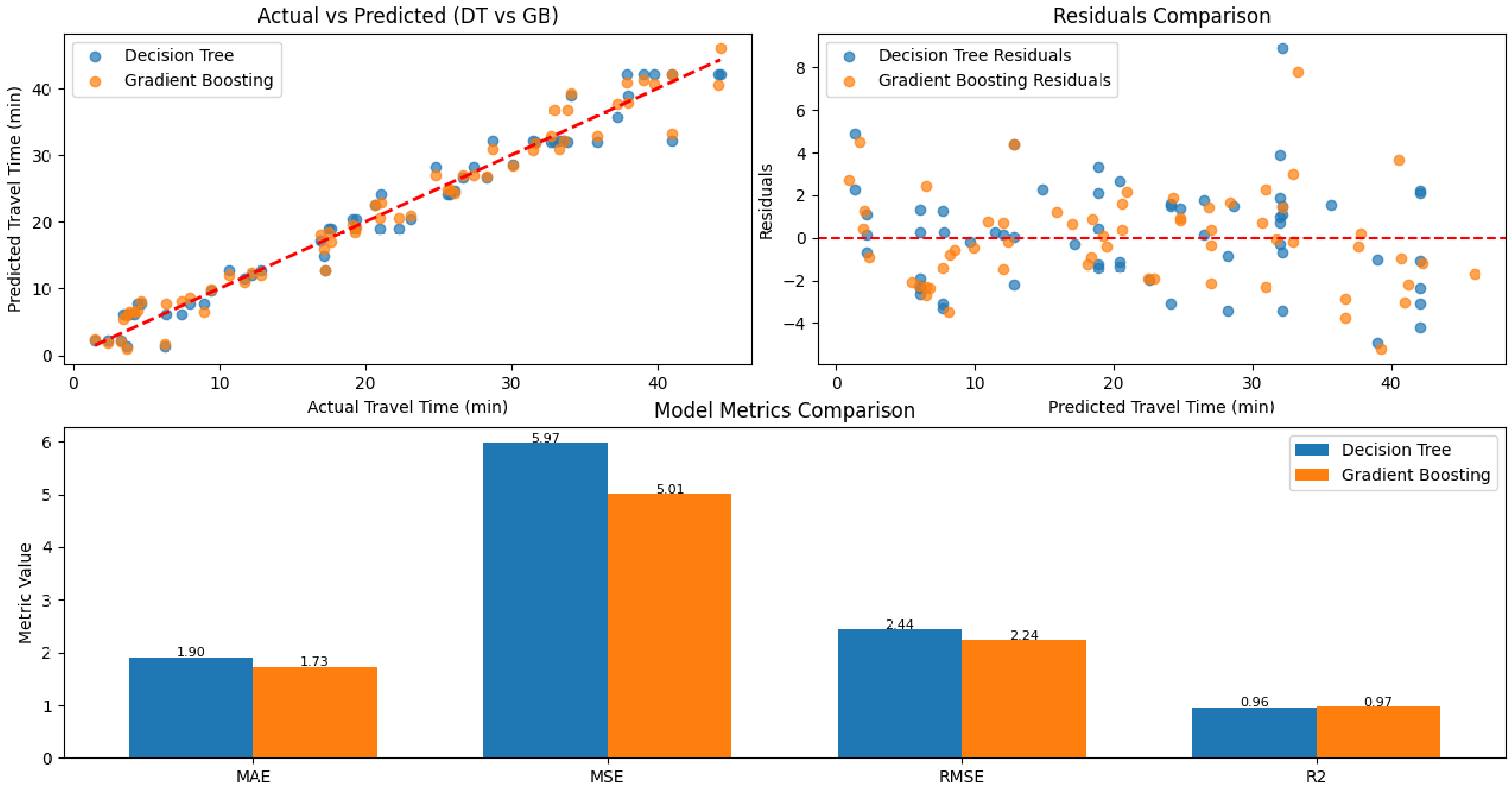
```python
mae  = mean_absolute_error(y_test, y_pred)
mse  = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2   = r2_score(y_test, y_pred)
```

VISUALIZATION



MAE: 2.37
MSE: 7.91
RMSE: 2.81
R²: 0.982
Decision Tree — Actual vs Predicted
Predicted (min)
Actual (min)
Decision Tree — Residual Plot
Residuals
Predicted
Decision Tree — Error Distribution
Frequency
Error
Decision Tree — Travel Time Along Route
Travel Time (min)
Distance (km)
Actual
Predicted

GRADIENT BOOSTING REGRESSION
VS
DECISION TREE



Actual vs Predicted (DT vs GB)
Decision Tree
Gradient Boosting
Predicted Travel Time (min)
Actual Travel Time (min)
Residuals Comparison
Decision Tree Residuals
Gradient Boosting Residuals
Residuals
Predicted Travel Time (min)
Model Metrics Comparison
Decision Tree
Gradient Boosting
Metric Value
1.90
1.73
5.97
5.01
2.44
2.24
0.96
0.97
MAE
MSE
RMSE
R2

SUMMARY

Decision Tree & Gradient Boosting

- Good for quick rule-based predictions
- Easy to interpret

- Best accuracy
- Ideal for real-time ETA predictions

RNN Model for TransConnect

5. **LSTM** (Long Short-Term Memory)
deep learning model for sequential data

Why LSTM?

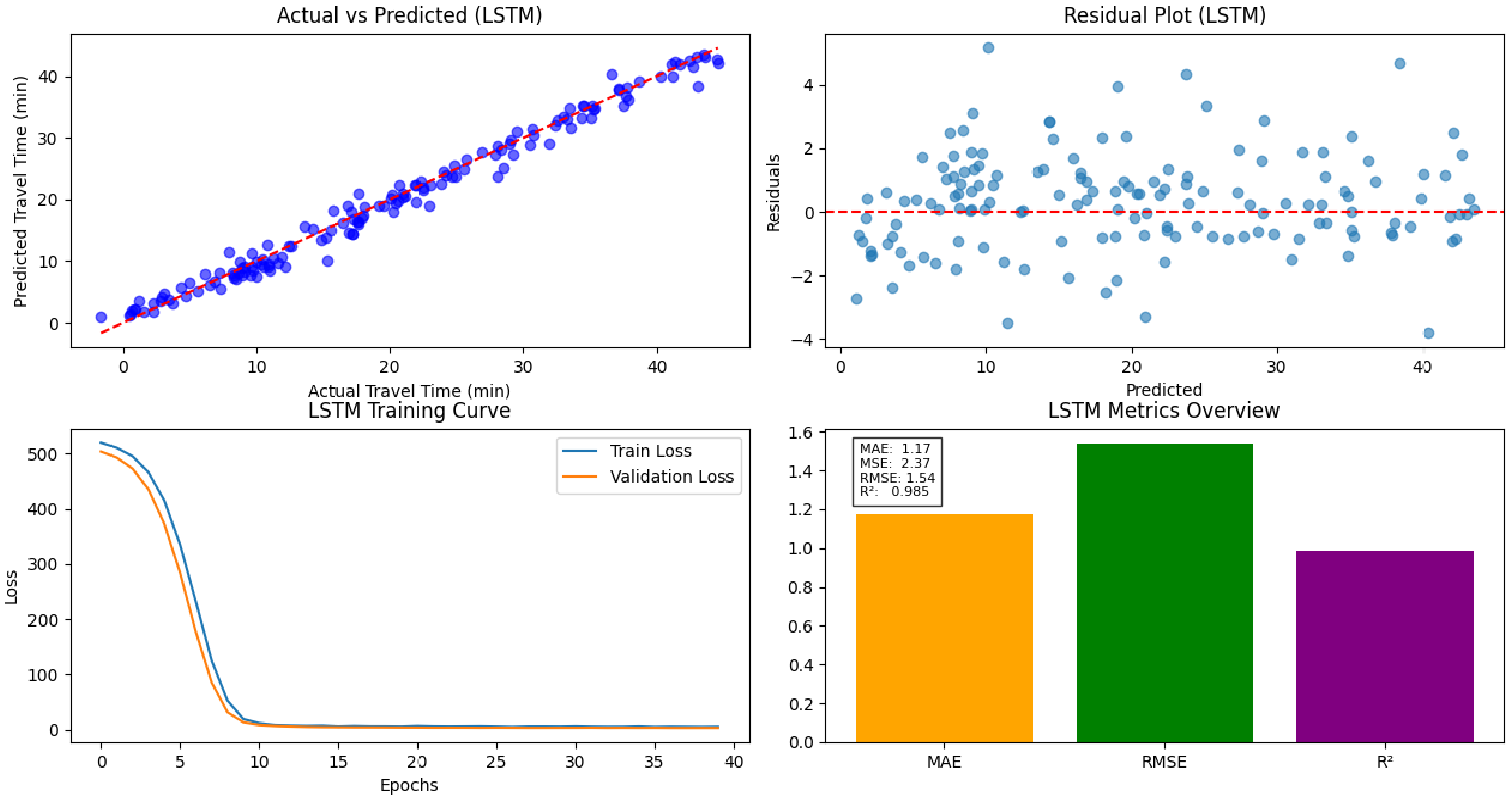
- Designed for sequential data
- Learns traffic patterns over time
- Captures rush hour vs normal variations
- More accurate than standard ML models

- **Model Structure**:
  - **Input** (distance, is_traffic, hour)
  - LSTM **Layer** (64 units) – learns sequential behavior
  - **Dropout** (0.2) – prevents overfitting
  - Dense Layer (32 neurons)
  - Output Layer: 1 node (Predicted travel time)

Key Features:

- Handles time-dependent patterns
- Smooth prediction curve instead of noisy output
- Works well with irregular travel speeds
- Adjust predictions using context (distance, traffic, hour)

VISUALIZATION



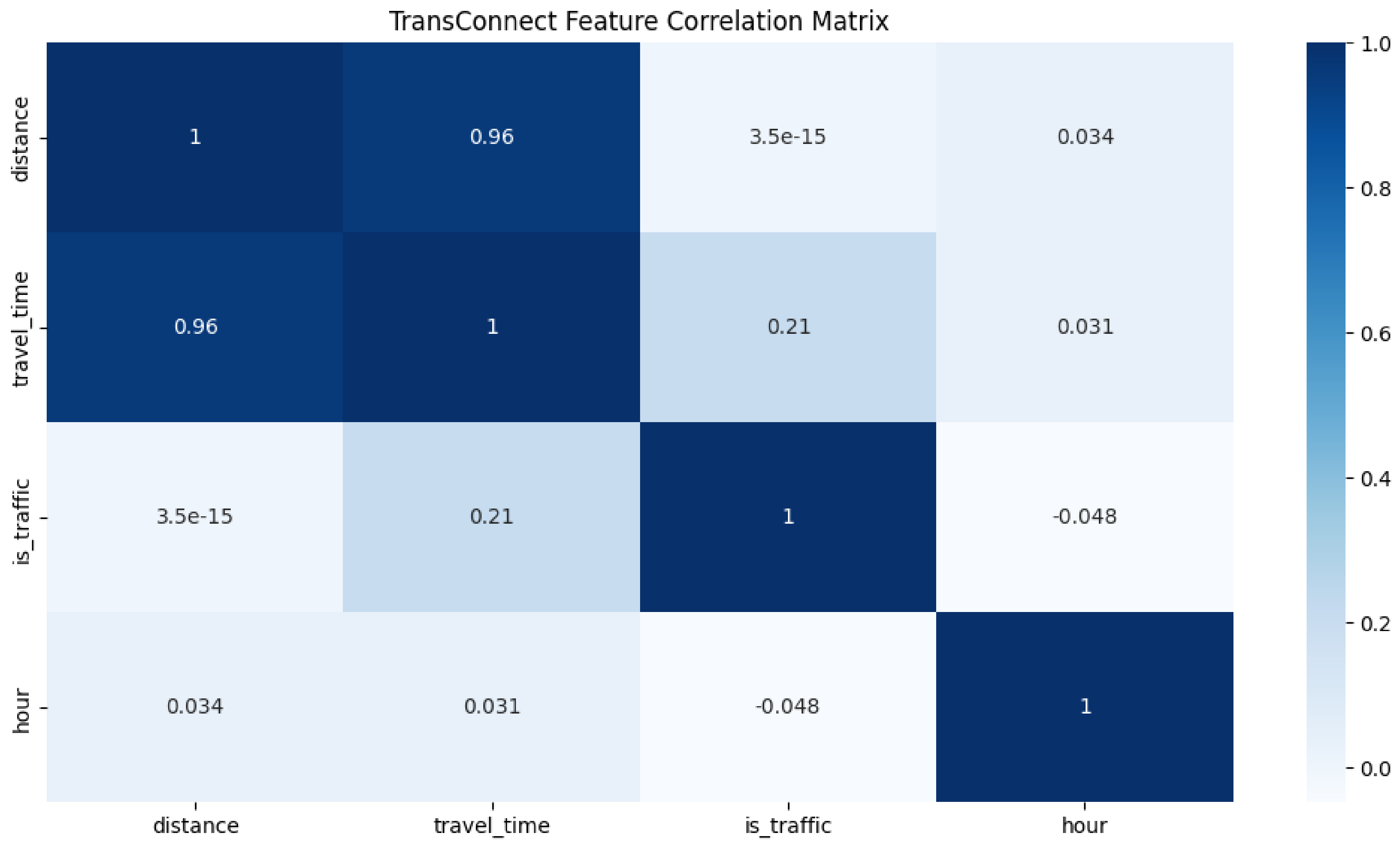
Actual vs Predicted (LSTM)
Predicted Travel Time (min)
Actual Travel Time (min)
Residual Plot (LSTM)
Residuals
Predicted
LSTM Training Curve
Train Loss
Validation Loss
Loss
Epochs
LSTM Metrics Overview
MAE: 1.17
MSE: 2.37
RMSE: 1.54
R²: 0.985
MAE
RMSE
R²

BEST MODEL – AMONG ALL

LSTM

best_transconnect_model.h5

MORE COMPARISON


TransConnect Feature Correlation Matrix
distance
travel_time
is_traffic
hour
1
0.96
3.5e-15
0.034
0.96
1
0.21
0.031
3.5e-15
0.21
1
-0.048
0.034
0.031
-0.048
1
1.0
0.8
0.6
0.4
0.2
0.0

Model Performance Summary

Model	MAE	MSE	RMSE	R^2
Linear Regression	1.6956	4.5832	2.1408	0.97095
Decision Tree	1.5216	4.0942	2.0234	0.97405
Random Forest	1.4514	3.5948	1.8960	0.97721
Gradient Boosting	1.4090	3.2546	1.8040	0.97937
LSTM	**1.2463**	**2.7779**	**1.6667**	**0.98239**

- Highest R^2
- lowest RMSE
- learns sequential traffic patterns

Challenges

- Limited historical data
- GPS noise and inconsistent-format coordinates
- Few features available for prediction
- Hard to validate ETA accuracy in real time

Mitigation Measures

- Collect more real bus timing logs (according to passengers, like us)
- Apply smoothing & cleaning to GPS streams
- Add engineered features (traffic, rush hour, holi
- Test across multiple routes for reliability

END