```
pip install polyline
```

```
Collecting polyline
  Downloading polyline-2.0.2-py3-none-any.whl.metadata (6.4 kB)
Downloading polyline-2.0.2-py3-none-any.whl (6.0 kB)
Installing collected packages: polyline
Successfully installed polyline-2.0.2
```

```
# Import Libraries
# ------------------------------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import random
import itertools
import os

from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error

from IPython.display import display
import requests
import polyline
import folium
import plotly.graph_objects as go

import folium
from IPython.display import display, clear_output

# Suppress all warnings
import warnings
warnings.filterwarnings("ignore")
```

```
df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/EDUNET COLAB IMPLEMENTATION/ICBP Project Data/all_route_variants_with_emissions.cs
df.head(5)
```

	Route ID	Origin	Destination	Route_Summary	Route_Distance_km	Route_Type	Traffic	Weather	Cargo_Weight_kg	Fuel_Efficiency
0	Ahmedabad-Surat-Route-1-NE 1 and NE 4	Ahmedabad	Surat	NE 1 and NE 4	311.67	Highway	Low	Clear	17619.74	
1	Ahmedabad-Surat-Route-1-NE 1 and NE 4	Ahmedabad	Surat	NE 1 and NE 4	311.67	Urban	Low	Clear	15224.30	
2	Ahmedabad-Surat-Route-1-NE 1 and NE 4	Ahmedabad	Surat	NE 1 and NE 4	311.67	Mixed	Low	Clear	15637.58	
3	Ahmedabad-Surat-Route-1-NE 1 and NE 4	Ahmedabad	Surat	NE 1 and NE 4	311.67	Highway	Low	Rainy	9175.74	
4	Ahmedabad-Surat-Route-1-NE 1 and NE 4	Ahmedabad	Surat	NE 1 and NE 4	311.67	Urban	Low	Rainy	8957.47	

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
numeric_columns = ['Route_Distance_km', 'Cargo_Weight_kg', 'Fuel_Efficiency_kmpl',
                   'Adjusted_Efficiency', 'Fuel_Used_Litres', 'CO2_Emissions_kg']

# Calculate min and max values for numeric columns
min_values = df[numeric_columns].min()
```

```
max_values = df[numeric_columns].max()

# Combine them in a DataFrame for easy viewing
min_max_values = pd.DataFrame({'Min': min_values, 'Max': max_values})

# Display the min and max values for each column
print(min_max_values)
```

```
                           Min       Max
Route_Distance_km       82.02    596.69
Cargo_Weight_kg       2000.26  19997.85
Fuel_Efficiency_kmpl     2.50      4.50
Adjusted_Efficiency      1.59      4.50
Fuel_Used_Litres        19.88    369.88
CO2_Emissions_kg        53.29    991.28
```

```
df.columns
```

```
Index(['Route ID', 'Origin', 'Destination', 'Route_Summary',
       'Route_Distance_km', 'Route_Type', 'Traffic', 'Weather',
       'Cargo_Weight_kg', 'Fuel_Efficiency_kmpl', 'Adjusted_Efficiency',
       'Fuel_Used_Litres', 'CO2_Emissions_kg'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17190 entries, 0 to 17189
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Route ID              17190 non-null  object
 1   Origin                17190 non-null  object
 2   Destination           17190 non-null  object
 3   Route_Summary         17190 non-null  object
 4   Route_Distance_km     17190 non-null  float64
 5   Route_Type            17190 non-null  object
 6   Traffic               17190 non-null  object
 7   Weather               17190 non-null  object
 8   Cargo_Weight_kg       17190 non-null  float64
 9   Fuel_Efficiency_kmpl  17190 non-null  float64
 10  Adjusted_Efficiency   17190 non-null  float64
 11  Fuel_Used_Litres      17190 non-null  float64
 12  CO2_Emissions_kg      17190 non-null  float64
dtypes: float64(6), object(7)
memory usage: 1.7+ MB
```

```
# Features and target
X = df.drop(columns=[
    'CO2_Emissions_kg',   # Target
    'Route ID',           # Identifier
    'Origin',
    'Destination',
    'Route_Summary'       # Contains text like 'NE 1 and NH 48'
])
y = df['CO2_Emissions_kg']

# Categorical and numeric features
categorical_cols = ['Route_Type', 'Traffic', 'Weather']
numeric_cols = [col for col in X.columns if col not in categorical_cols]

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Preprocessing pipeline
preprocessor = ColumnTransformer([
    ('num', StandardScaler(), numeric_cols),
    ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_cols)])

# Modeling pipeline
model_pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('model', RandomForestRegressor(n_estimators=100, random_state=42))])
```

```
# Train the model
model_pipeline.fit(X_train, y_train)

# Predict
y_pred = model_pipeline.predict(X_test)


# Basic Metrics
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)  # Manually compute RMSE
mape = np.mean(np.abs((y_test - y_pred) / y_test)) * 100  # % Error

# Output the results
print(f"Evaluation Metrics:")
print(f"MAE  (Mean Absolute Error): {mae:.2f} kg CO2")
print(f"MSE  (Mean squared Error): {mse:.2f} kg CO2")
print(f"RMSE (Root Mean Squared Error): {rmse:.2f} kg CO2")
print(f"MAPE (Mean Absolute Percentage Error): {mape:.2f}%")
```

```
Evaluation Metrics:
MAE  (Mean Absolute Error): 0.05 kg CO2
MSE  (Mean squared Error): 0.05 kg CO2
RMSE (Root Mean Squared Error): 0.22 kg CO2
MAPE (Mean Absolute Percentage Error): 0.02%
```

```
def generate_routes_and_predict(user_input, model, route_data):

    """
    Generates all possible routes for a given origin and destination,
    computes fuel usage, and predicts CO2 emissions using a provided model.

    Parameters:
        user_input (dict): User input including Origin, Destination, Traffic, Weather,Cargo Weight, and Fuel Efficiency.
        model (sklearn): A trained model with a .predict() method.
        route_data (DataFrame): Contains predefined route options and features.

    Returns:
        List[dict]: List of route IDs and predicted CO2 emissions.
    """

    origin = user_input['Origin']
    destination = user_input['Destination']
    traffic = user_input['Traffic']
    weather = user_input['Weather']
    cargo_weight = user_input['Cargo_Weight_kg']
    fuel_efficiency = user_input['Fuel_Efficiency_kmpl']

    possible_routes = []

    # Get all unique route summaries for the given origin-destination pair
    unique_route_summaries = route_data['Route_Summary'].unique()

    for summary in unique_route_summaries:
        route_subset = route_data[route_data['Route_Summary'] == summary]

        if route_subset.empty:
            continue  # Skip if no data for this summary

        # Take the first matching route (or choose based on a better rule)
        selected_route = route_subset.iloc[0]

        distance_km = selected_route['Route_Distance_km']
        route_type = selected_route['Route_Type']

        fuel_used = distance_km / fuel_efficiency

        # Prepare input features for prediction
        features = {
            'Route_Distance_km': distance_km,
            'Route_Type': route_type,
            'Traffic': traffic,
            'Weather': weather,
            'Cargo_Weight_kg': cargo_weight,
            'Fuel_Efficiency_kmpl': fuel_efficiency,
            'Adjusted_Efficiency': fuel_efficiency,  # You could modify this dynamically if needed
```

```
            'Fuel_Used_Litres': fuel_used
        }

        input_df = pd.DataFrame([features])
        predicted_emission = model.predict(input_df)[0]

        possible_routes.append({
            'Route ID': f"{origin}-{destination}-{summary}",
            'Predicted_CO2_Emissions': predicted_emission
        })

    return possible_routes



def display_all_routes_emissions(route_predictions, origin, destination):
    # Convert the list of predictions into a DataFrame
    df = pd.DataFrame(route_predictions)

    # Sort the DataFrame by predicted emissions
    df_sorted = df.sort_values(by='Predicted_CO2_Emissions').reset_index(drop=True)

    # Set pandas display options to show more rows and columns if needed
    pd.set_option('display.max_rows', None)  # No row limit
    pd.set_option('display.max_columns', None)  # No column limit
    pd.set_option('display.width', 1000)  # Wide enough to show data

    # Display the sorted DataFrame
    print(f"All Routes and Their Predicted CO₂ Emissions: {origin} --> {destination}")
    display(df_sorted)  # For a clean display in Jupyter Notebooks

    return df_sorted  # Return the DataFrame for further use


def plot_emission_spectrum_with_route_id(route_predictions, origin, destination):
    df = pd.DataFrame(route_predictions)
    df_sorted = df.sort_values(by='Predicted_CO2_Emissions').reset_index(drop=True)

    # Select 5 routes across the spectrum
    indices = [0, int(len(df)*0.25), int(len(df)*0.5), int(len(df)*0.75), len(df)-1]
    spectrum_df = df_sorted.iloc[indices].copy()

    # Plot
    plt.figure(figsize=(10, 6))
    sns.barplot(x='Route ID', y='Predicted_CO2_Emissions', data=spectrum_df, palette='Spectral')

    plt.title(f"CO₂ Emission Spectrum for Routes: {origin} → {destination}")
    plt.xlabel("Route ID")
    plt.ylabel("Predicted CO₂ Emissions (kg)")
    plt.xticks(rotation=90, ha='right')
    plt.tight_layout()
    plt.show()

    return spectrum_df  # return selected routes for inspection



df.columns
```

```
Index(['Route ID', 'Origin', 'Destination', 'Route_Summary',
       'Route_Distance_km', 'Route_Type', 'Traffic', 'Weather',
       'Cargo_Weight_kg', 'Fuel_Efficiency_kmpl', 'Adjusted_Efficiency',
       'Fuel_Used_Litres', 'CO2_Emissions_kg'],
      dtype='object')
```

```
# To get range of all variation affect on Co2 Emission

numeric_columns = ['Route_Distance_km', 'Cargo_Weight_kg', 'Fuel_Efficiency_kmpl','Adjusted_Efficiency', 'Fuel_Used_Litres', 'CO2_Emissions

# Calculate min and max values for numeric columns
min_values = df[numeric_columns].min()
max_values = df[numeric_columns].max()

# Combine them in a DataFrame for easy viewing
min_max_values = pd.DataFrame({'Min': min_values, 'Max': max_values})

# Display the min and max values for each column
```

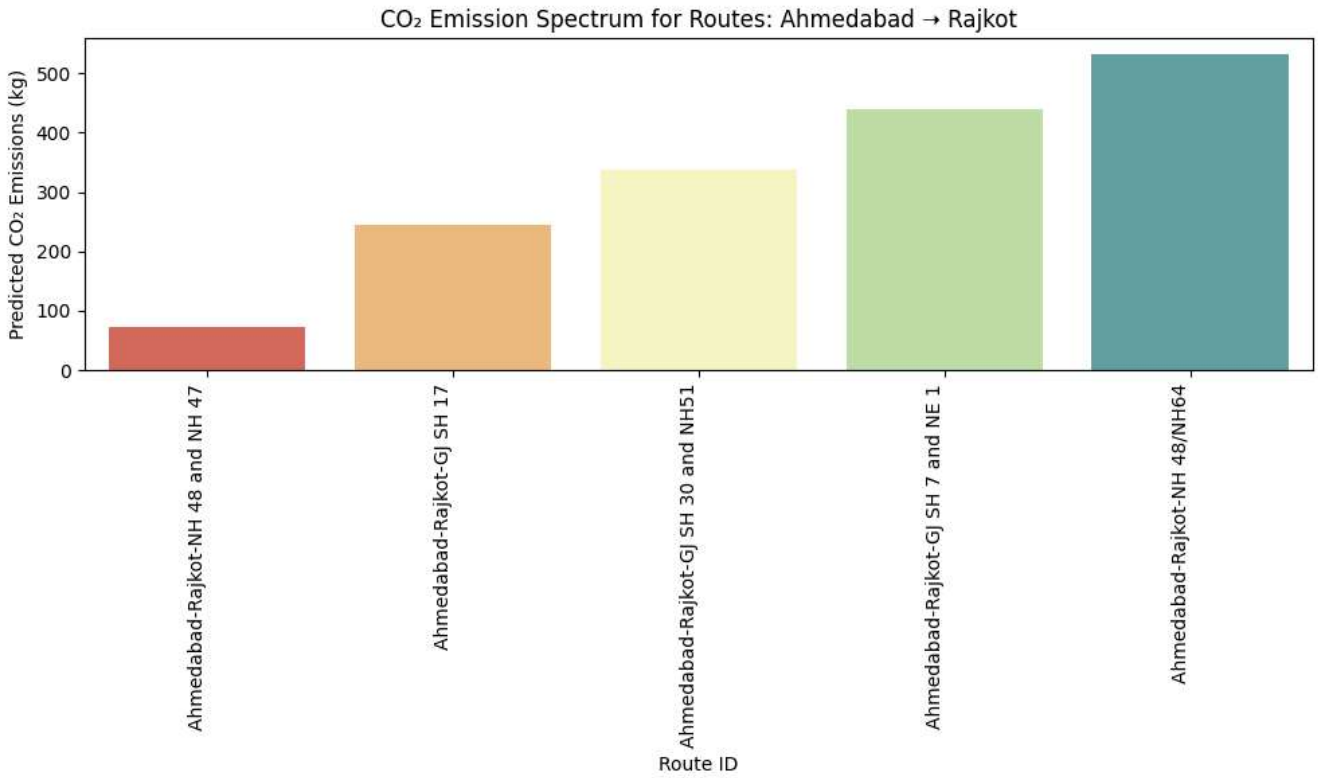
```
print(min_max_values)
```

```
                           Min       Max
    Route_Distance_km       82.02    596.69
    Cargo_Weight_kg       2000.26  19997.85
    Fuel_Efficiency_kmpl     2.50      4.50
    Adjusted_Efficiency      1.59      4.50
    Fuel_Used_Litres        19.88    369.88
    CO2_Emissions_kg        53.29    991.28
```

```
# Collect user inputs
user_input = {
    'Origin': input("Enter Origin City: "),
    'Destination': input("Enter Destination City: "),
    'Traffic': input("Enter Traffic Condition (Low/Medium/High): "),
    'Weather': input("Enter Weather Condition (Clear/Rainy/Foggy/Summer/Storm): "),
    'Cargo_Weight_kg': float(input("Enter Cargo Weight (kg): ")),
    'Fuel_Efficiency_kmpl': float(input("Enter Fuel Efficiency (km/l): "))
}

# Generate route predictions
route_predictions = generate_routes_and_predict(user_input, model_pipeline, df)
print(route_predictions)

# Show best route recommendation
best_route = min(route_predictions, key=lambda x: x['Predicted_CO2_Emissions'])
print("\nRecommended Route:")
print(best_route)
```

```
Enter Origin City: Ahmedabad
Enter Destination City: Rajkot
Enter Traffic Condition (Low/Medium/High): Medium
Enter Weather Condition (Clear/Rainy/Foggy/Summer/Storm): Rainy
Enter Cargo Weight (kg): 8000
Enter Fuel Efficiency (km/l): 3
[{'Route ID': 'Ahmedabad-Rajkot-NE 1 and NE 4', 'Predicted_CO2_Emissions': np.float64(278.4738000000001)}, {'Route ID': 'Ahmedabad-Rajk

Recommended Route:
{'Route ID': 'Ahmedabad-Rajkot-NH 48 and NH 47', 'Predicted_CO2_Emissions': np.float64(73.28789999999998)}
```

```
# Assuming route_predictions and user_input['Origin'] and user_input['Destination'] are defined
plot_emission_spectrum_with_route_id(route_predictions, user_input['Origin'], user_input['Destination'])
```



	Route ID	Predicted_CO2_Emissions
0	Ahmedabad-Rajkot-NH 48 and NH 47	73.2879
27	Ahmedabad-Rajkot-GJ SH 17	244.5281
55	Ahmedabad-Rajkot-GJ SH 30 and NH51	338.3915
83	Ahmedabad-Rajkot-GJ SH 7 and NE 1	438.6926
110	Ahmedabad-Rajkot-NH 48/NH64	531.6969

```
# IDentify al possibe routes between origin and destination with their carbon emission rate in ascending order

routes_emissions_df = display_all_routes_emissions(route_predictions, user_input['Origin'], user_input['Destination'])
```

	Route ID	Predicted_CO2_Emissions
0	Ahmedabad-Rajkot-NH 48 and NH 47	73.2879
1	Ahmedabad-Rajkot-NH147	85.2356
2	Ahmedabad-Rajkot-Gujarat State Highway 71 and ...	87.1574
3	Ahmedabad-Rajkot-NE 1 and Ahmedabad - Palanpur...	88.4451
4	Ahmedabad-Rajkot-NH 751 and NE 4	88.6029
5	Ahmedabad-Rajkot-Chikhli-Atgam Rd and NH 48	88.9254
6	Ahmedabad-Rajkot-NH341	108.2440
7	Ahmedabad-Rajkot-GJ SH 31	110.5410
8	Ahmedabad-Rajkot-NE 4 and NH 48	119.4277
9	Ahmedabad-Rajkot-NH 754K and NE 4	139.8073
10	Ahmedabad-Rajkot-NH 47 and NH 751D	146.1792
11	Ahmedabad-Rajkot-NE 1, NE 4 and NH 48	170.9567
12	Ahmedabad-Rajkot-NH 47 and NH 27	173.6348
13	Ahmedabad-Rajkot-NH 751D and NH 751	174.0994
14	Ahmedabad-Rajkot-Surat - Navsari Rd	175.1708
15	Ahmedabad-Rajkot-NH 754K and NH 27	177.9475
16	Ahmedabad-Rajkot-GJ SH 1	178.5135
17	Ahmedabad-Rajkot-GJ SH 19	195.6200
18	Ahmedabad-Rajkot-NE 1 and NH 47	199.7569
19	Ahmedabad-Rajkot-NH 27 and NH 754K	201.3828
20	Ahmedabad-Rajkot-GJ SH 18 and NH 754K	203.4314
21	Ahmedabad-Rajkot-NH51 and Bhavnagar - Rajkot Rd	204.2440
22	Ahmedabad-Rajkot-NH 754K and Radhanpur Rd	217.5163
23	Ahmedabad-Rajkot-Jasdan - Ahmedabad Hwy	217.9407
24	Ahmedabad-Rajkot-NH64 and Ahmedabad - Palanpur...	227.0905
25	Ahmedabad-Rajkot-NH 27 and Bhuj - Bhachau Hwy	237.5768
26	Ahmedabad-Rajkot-NH53 and NH 48	238.7897
27	Ahmedabad-Rajkot-GJ SH 17	244.5281
28	Ahmedabad-Rajkot-NH 48, NE 4 and NE 1	246.5004
29	Ahmedabad-Rajkot-Bhavnagar - Rajkot Rd	247.1407
30	Ahmedabad-Rajkot-NH151 and NH 27	252.5402
31	Ahmedabad-Rajkot-NH64 and NH 751	258.6810
32	Ahmedabad-Rajkot-Gandhinagar Link Rd	259.6815
33	Ahmedabad-Rajkot-NH 27 and Rajkot - Morbi Hwy	259.8543
34	Ahmedabad-Rajkot-NH 47 and NE 1	261.0933
35	Ahmedabad-Rajkot-Ahmedabad - Palanpur Highway ...	265.0921
36	Ahmedabad-Rajkot-NH 27 and NH151	268.0362
37	Ahmedabad-Rajkot-NH 751 and NH 751D	268.3349
38	Ahmedabad-Rajkot-GJ SH 31 and Bhavnagar - Rajk...	269.6285
39	Ahmedabad-Rajkot-NH64 and NH 47	276.1320
40	Ahmedabad-Rajkot-NE 1 and NE 4	278.4738
41	Ahmedabad-Rajkot-GJ SH 7 and Morbi - Halvad Rd	280.1800
42	Ahmedabad-Rajkot-NH64 and NE 4	284.9099
43	Ahmedabad-Rajkot-NH 48 and NE 4	287.0216
44	Ahmedabad-Rajkot-NH 48 and GJ SH 195/GJ SH 196	292.8612

45	Ahmedabad-Rajkot-Radhanpur Rd and NH 754K	298.9829
46	Ahmedabad-Rajkot-NE 1 and NH 48	311.4174
47	Ahmedabad-Rajkot-NE 4 and NH 48/NH64	315.5280
48	Ahmedabad-Rajkot-Ahmedabad - Palanpur Highway ...	318.3340
49	Ahmedabad-Rajkot-NH64	322.0885
50	Ahmedabad-Rajkot-Gandhinagar - Ahmedabad Rd an...	325.4815
51	Ahmedabad-Rajkot-NE 4 and NH 751	327.9474
52	Ahmedabad-Rajkot-NH 47 and NH 48	327.9597
53	Ahmedabad-Rajkot-NH51	331.8315
54	Ahmedabad-Rajkot-NH 47 and NE 4	336.0823
55	Ahmedabad-Rajkot-GJ SH 30 and NH51	338.3915
56	Ahmedabad-Rajkot-NE 4, NE 1 and NH 48	340.0875
57	Ahmedabad-Rajkot-NH 751 and NH64	344.0765
58	Ahmedabad-Rajkot-NH151	344.1005
59	Ahmedabad-Rajkot-NH53	346.5629
60	Ahmedabad-Rajkot-NE 4 and NH 751D	351.1687
61	Ahmedabad-Rajkot-NH 48	351.4999
62	Ahmedabad-Rajkot-NE 1	353.8919
63	Ahmedabad-Rajkot-NH 27 and NH 47	363.0840
64	Ahmedabad-Rajkot-NH 48 and NH51	377.7060
65	Ahmedabad-Rajkot-GJ SH 36	377.8351
66	Ahmedabad-Rajkot-Gujarat State Highway 71	381.1058
67	Ahmedabad-Rajkot-GJ SH 22 and Rajkot - Morbi Hwy	382.0077
68	Ahmedabad-Rajkot-NH 754K	386.4338
69	Ahmedabad-Rajkot-NH51 and NH 48	391.1783
70	Ahmedabad-Rajkot-NH 47	394.3338
71	Ahmedabad-Rajkot-NH 47 and NH64	400.3008
72	Ahmedabad-Rajkot-Ahmedabad - Patan Highway Rd ...	404.4577
73	Ahmedabad-Rajkot-Ahmedabad - Palanpur Highway ...	406.3725
74	Ahmedabad-Rajkot-NH 48 and GJ SH 7	406.6138
75	Ahmedabad-Rajkot-NH 48 and Surat - Bardoli Rd	406.6483
76	Ahmedabad-Rajkot-Gandhinagar - Ahmedabad Rd	408.3616
77	Ahmedabad-Rajkot-NH 48 and GJ SH 10	411.8181
78	Ahmedabad-Rajkot-NE 4 and NH64	416.4059
79	Ahmedabad-Rajkot-NH 27	419.1690
80	Ahmedabad-Rajkot-NH 48/NH64 and NE 1	424.8500
81	Ahmedabad-Rajkot-Ahmedabad - Palanpur Highway ...	432.8024
82	Ahmedabad-Rajkot-Ahmedabad - Patan Highway Rd	438.2291
83	Ahmedabad-Rajkot-GJ SH 7 and NE 1	438.6926
84	Ahmedabad-Rajkot-NE 4 and NH 47	444.3852
85	Ahmedabad-Rajkot-NH 48/NH64 and NE 4	445.4137
86	Ahmedabad-Rajkot-NH51 and GJ SH 31	446.6950
87	Ahmedabad-Rajkot-NE 1, NE 4 and NH 48/NH64	448.9242
88	Ahmedabad-Rajkot-Bhuj - Bhachau Hwy	461.7330
89	Ahmedabad-Rajkot-NE 1 and NH 48/NH64	462.1830
90	Ahmedabad-Rajkot-NH 754K and GJ SH 18	465.6211
91	Ahmedabad-Rajkot-GJ SH 7	470.1948

91	Ahmedabad-Rajkot-GJ SH 7	470.1948
92	Ahmedabad-Rajkot-NH 754K and NE 1	471.1500
93	Ahmedabad-Rajkot-NH 751	471.6366
94	Ahmedabad-Rajkot-Morbi - Halvad Rd and GJ SH 7	472.5549
95	Ahmedabad-Rajkot-GJ SH 30	475.8953
96	Ahmedabad-Rajkot-NH 27 and GJ SH 7	480.7225
97	Ahmedabad-Rajkot-NH 27 and NH341	482.8531
98	Ahmedabad-Rajkot-NH 48/NH64 and NH 48	486.5903
99	Ahmedabad-Rajkot-NE 1 and NH147	489.3492
100	Ahmedabad-Rajkot-NE 4	496.9899
101	Ahmedabad-Rajkot-NE 1 and NH 754K	502.3472
102	Ahmedabad-Rajkot-NE 4 and NH 754K	505.1503
103	Ahmedabad-Rajkot-GJ SH 181 and NH 48	510.8043
104	Ahmedabad-Rajkot-NE 4 and NE 1	513.3828
105	Ahmedabad-Rajkot-GJ SH 30 and NE 4	515.9801
106	Ahmedabad-Rajkot-Ahmedabad - Dholera Expy and ...	518.7728
107	Ahmedabad-Rajkot-NH 48 and NH53	520.1221
108	Ahmedabad-Rajkot-NH64 and NH 48/NH64	526.9104
109	Ahmedabad-Rajkot-Rajkot - Morbi Hwy	531.4864
110	Ahmedabad-Rajkot-NH 48/NH64	531.6969