# Air Quality Analysis in Tamil Nadu

**Phase3 Development Part1:**

Analyzing air quality in Tamil Nadu is crucial for both environmental and public health reasons. The state of Tamil Nadu, located in southern India, has experienced rapid industrialization and urbanization in recent years, which has led to various air quality challenges. To undertake a comprehensive air quality analysis for development, it is important to consider several key aspects and steps.

**1. Data Collection:**

Monitoring Stations: Establish a network of air quality monitoring stations across Tamil Nadu. These stations should be strategically located in urban, industrial, and rural areas to capture a representative sample of air quality conditions.

- **Parameters:** Measure various air quality parameters, including particulate matter (PM2.5 and PM10), nitrogen dioxide (NO2), sulfur dioxide (SO2), carbon monoxide (CO), ozone (O3), and volatile organic compounds (VOCs).
- **Meteorological Data:** Collect meteorological data, such as temperature, humidity, wind speed, and wind direction, as these factors can influence air quality.
- **Historical Data:** Gather historical air quality data to establish trends and identify areas with chronic air quality problems.

**2. Data Analysis:**

Air Quality Index (AQI): Calculate the AQI for different locations in Tamil Nadu to provide a clear and understandable representation of air quality to the public.

- **Identify Hotspots:** Identify areas with consistently poor air quality, such as major cities or industrial zones, and pinpoint the key pollutants responsible.
- **Seasonal Trends:** Analyze seasonal variations in air quality, as well as the factors contributing to these variations, such as agricultural burning, weather conditions, or industrial activity.

**3. Pollution Sources:**

Industrial Emissions: Examine emissions from industrial facilities, such as factories and power plants, and assess compliance with emission standards.

- **Vehicle Emissions:** Evaluate the impact of vehicular emissions on air quality, considering the prevalence of different types of vehicles and fuel types.
- **Agricultural Practices:** Investigate the role of agriculture in air quality, including the use of pesticides and burning of crop residues.
- **Waste Management:** Assess waste disposal practices and their impact on air quality, especially in urban areas.

**4. Health Impact Assessment:**

Collaborate with healthcare institutions to study the health effects of poor air quality on the population of Tamil Nadu.

Identify vulnerable groups, such as children, the elderly, and individuals with pre-existing respiratory conditions, and assess their exposure and health outcomes.

**5. Policy and Regulation:**

Review existing air quality regulations and policies in Tamil Nadu to identify gaps or areas for improvement.

Develop or update regulations to control emissions from various sources, and enforce strict compliance measures.

**6. Public Awareness:**

Launch public awareness campaigns to educate residents about the health risks associated with poor air quality and ways to protect themselves.

Provide real-time air quality information through websites, apps, and public displays.

**7. Mitigation Strategies:**

Implement pollution control technologies in industries and encourage the use of cleaner fuels.

Promote sustainable urban planning, public transportation, and green spaces to reduce vehicle emissions and enhance air quality.

Encourage agricultural practices that minimize burning and promote sustainable waste management.

**8. International Cooperation:**

Collaborate with neighboring states and countries to address transboundary air pollution issues, especially during cross-border events like crop burning.

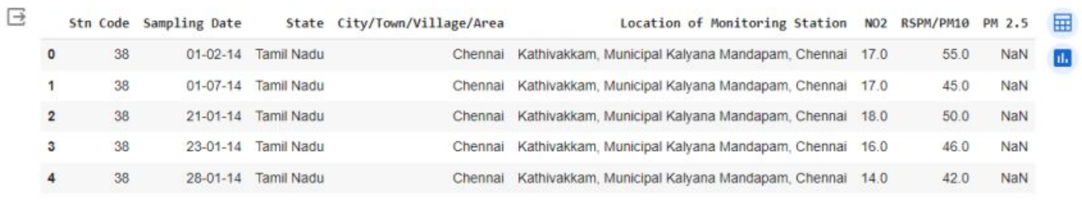
This air quality analysis is the first part of a comprehensive strategy to improve air quality in Tamil Nadu. It is essential to monitor progress over time and adjust strategies as needed to ensure cleaner air for the people and the environment.

**Input:**

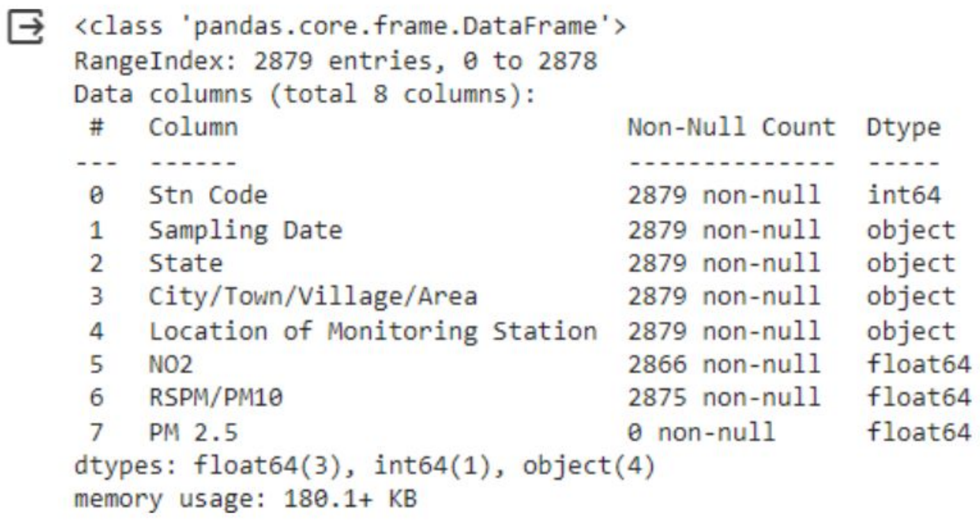
```
import os
import glob
import time
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_theme()
from sklearn.ensemble import (
    RandomForestRegressor,
    GradientBoostingRegressor,
    AdaBoostRegressor,
    HistGradientBoostingRegressor
)
from sklearn.metrics import (
   r2_score,
   mean_squared_error,
   mean_absolute_error,
   mean_absolute_percentage_error
)
from sklearn.model_selection import (
   cross_val_score,
   TimeSeriesSplit,
   RandomizedSearchCV
)

import xgboost as xgb
from IPython.display import clear_output

N_JOBS = -1
RANDOM_STATE = 18
DATASET_SRC = '/content/cpcb_dly_aq_tamil_nadu-2014.csv'
df_states = pd.read_csv('/content/cpcb_dly_aq_tamil_nadu-2014.csv')
df_states.drop(columns=['Agency', 'Type of Location', 'SO2'], inplace=True)
df_states.head()

df_states.head()
unique_states = df_states['State'].unique()
unique_states
def combine_state_df(State):
   '''
   Combine all state files into a single dataframe and attaching the city information.
   Parameters
   -----------
      state_name (str): The name of the state
   Return
   -------
      df (DataFrame): The combined dataframe from all files of a specific state
```

```
    '''
    state_code = df_states[df_states['State'] == state_name]['file_name'].iloc[0][:2]
    state_files = glob.glob('/content/cpcb_dly_aq_tamil_nadu-2014.csv')
    print(f'Combining a total of {len(state_files)} files...\n')
    combined_df = []
    for state_file in state_files:
        file_name = state_file.split(f'{DATASET_SRC}/')[1][0:-4]
        file_df = pd.read_csv(state_file)
        file_df['City'] = df_states[df_states['file_name'] == file_name]['City'].values[0]
        file_df['city'] = file_df['City'].astype('string')
        combined_df.append(file_df)
    return pd.concat(combined_df)
df_states.info()

def create_dt_index(dataframe):
    dataframe = dataframe.drop(columns='To Date')
    dataframe['From Date'] = pd.to_datetime(dataframe['From Date'])
    dataframe = dataframe.rename(columns={'From Date': 'datetime'})
    return dataframe.set_index('datetime')

df_states.head(2)

df_states.tail(2)
def plot_feature_similarities(dataframe, feature_groups, columns=2):
    rows = int((len(feature_groups)/columns)//1)
    fig, axes = plt.subplots(rows, columns, figsize=(13, 4*rows))
    fig.tight_layout(pad=3.0)

    row_num = 0
    col_num = 0
    for pos, group in enumerate(feature_groups):
        # Move to new row
        if pos % columns == 0 and pos != 0:
            row_num += 1
            col_num = 0
        for feature in feature_groups[group]:
            df_feature = dataframe[dataframe[feature].notnull()][feature]
            df_feature = df_feature.groupby([df_feature.index.year]).mean(numeric_only=True)
            sns.lineplot(data=df_feature, label=feature, ax=axes[row_num, col_num])
        axes[row_num, col_num].set_title(group)
        axes[row_num, col_num].set(xlabel=None)
        col_num += 1

    plt.plot()

df_states.isnull().sum().sort_values(ascending=False)

df_states = df_states.dropna(how='all')
df_states = df_states.dropna(how='all', axis='columns')

def get_null_info(dataframe):
    null_vals = dataframe.isnull().sum()

    df_null_vals = pd.concat({'Null Count': null_vals,
                              'Percent Missing (%)': round(null_vals * 100 / len(dataframe), 2)}, axis=1)
    return df_null_vals.sort_values(by=['Null Count'], ascending=False)

df_null_info = get_null_info(df_states)

plt.figure(figsize=(8, 10))
sns.barplot(data=df_null_info, x='Percent Missing (%)', y=df_null_info.index, orient='h',
color='steelblue')
plt.show()

def get_overall_ds_info():
    features = {}
    total_records = 0

    for i, state_name in enumerate(unique_states):
        clear_output(wait=False)
        print(f"Processing state of {state_name} ({i+1}/{len(unique_states)})")

        temp_df = combine_state_df(state_name) # Get combined state dataframe
        temp_df = create_dt_index(temp_df)     # Create datetime index
        temp_df = temp_df.dropna(how='all')    # Drop empty rows

        comparisons = get_null_info(temp_df)

        total_records += df.shape[0]

        for feature in comparisons.index:
            if feature in features:
                features[feature] += comparisons.loc[[feature]]['Null Count'].values[0]
            else:
                features[feature] = comparisons.loc[[feature]]['Null Count'].values[0]

    ds_null_info = pd.DataFrame.from_dict(features, orient='index', columns=['Null Count'])
    ds_null_info['Percent Missing (%)'] = round(ds_null_info['Null Count'] * 100 / total_records, 2)
    ds_null_info = ds_null_info.sort_values(by=['Null Count'], ascending=False)
    return ds_null_info

threshold = 0.6
df_states = df_states.dropna(thresh=df_states.shape[0]*threshold, axis=1)
get_null_info(df_states)

get_null_info(df)def plot_features_by_group(features, df_states):
    for feature in features:
        fig, ax = plt.subplots(1, 1, figsize=(12, 4))
        fig.suptitle(feature)

        labels = []
        for i, (group, group_df) in enumerate(slice_groups.items()):
            data_slice = group_df[group_df.columns.intersection(pollutants[feature])]

            # Keep only the NOx feature, as it combines both NO (Nitrogen Oxide) and NO2 (Nitrogen
        Dioxide)
        if feature == "Nitrogen Compounds":
          data_slice = data_slice.drop(['NO (ug/m3)', 'NO2 (ug/m3)'], axis=1)

        data_slice.plot(kind="line", ax=ax)
        for column in data_slice.columns:
          labels.append(f'{column} [{group}]')
      ax.set(xlabel=None)
      ax.legend(labels)
      plt.plot()
get_null_info(df_states)
df = df_states.interpolate(method='pad')
df = df_states.fillna(df_states.mean())
df.info()
 df_states.plot.bar()
```

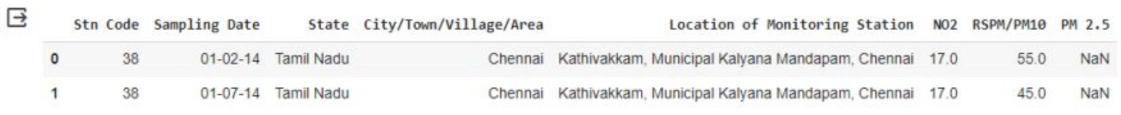
**Output:**



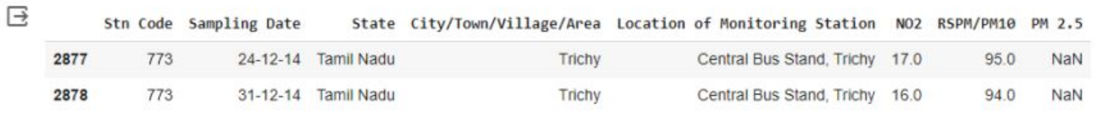
| | Stn Code | Sampling Date | State | City/Town/Village/Area | Location of Monitoring Station | NO2 | RSPM/PM10 | PM 2.5 |
|---|---|---|---|---|---|---|---|---|
| 0 | 38 | 01-02-14 | Tamil Nadu | Chennai | Kathivakkam, Municipal Kalyana Mandapam, Chennai | 17.0 | 55.0 | NaN |
| 1 | 38 | 01-07-14 | Tamil Nadu | Chennai | Kathivakkam, Municipal Kalyana Mandapam, Chennai | 17.0 | 45.0 | NaN |
| 2 | 38 | 21-01-14 | Tamil Nadu | Chennai | Kathivakkam, Municipal Kalyana Mandapam, Chennai | 18.0 | 50.0 | NaN |
| 3 | 38 | 23-01-14 | Tamil Nadu | Chennai | Kathivakkam, Municipal Kalyana Mandapam, Chennai | 16.0 | 46.0 | NaN |
| 4 | 38 | 28-01-14 | Tamil Nadu | Chennai | Kathivakkam, Municipal Kalyana Mandapam, Chennai | 14.0 | 42.0 | NaN |



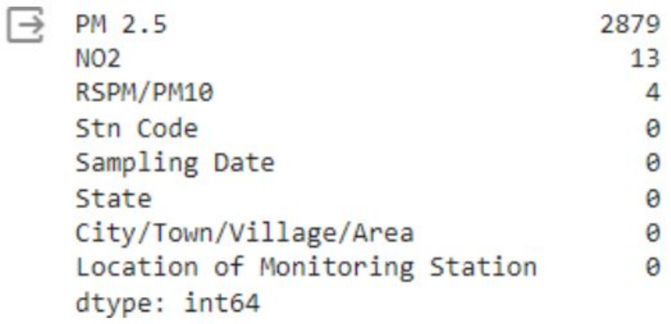
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2879 entries, 0 to 2878
Data columns (total 8 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   Stn Code                        2879 non-null   int64  
 1   Sampling Date                   2879 non-null   object 
 2   State                           2879 non-null   object 
 3   City/Town/Village/Area          2879 non-null   object 
 4   Location of Monitoring Station  2879 non-null   object 
 5   NO2                             2866 non-null   float64
 6   RSPM/PM10                       2875 non-null   float64
 7   PM 2.5                          0 non-null      float64
dtypes: float64(3), int64(1), object(4)
memory usage: 180.1+ KB
```



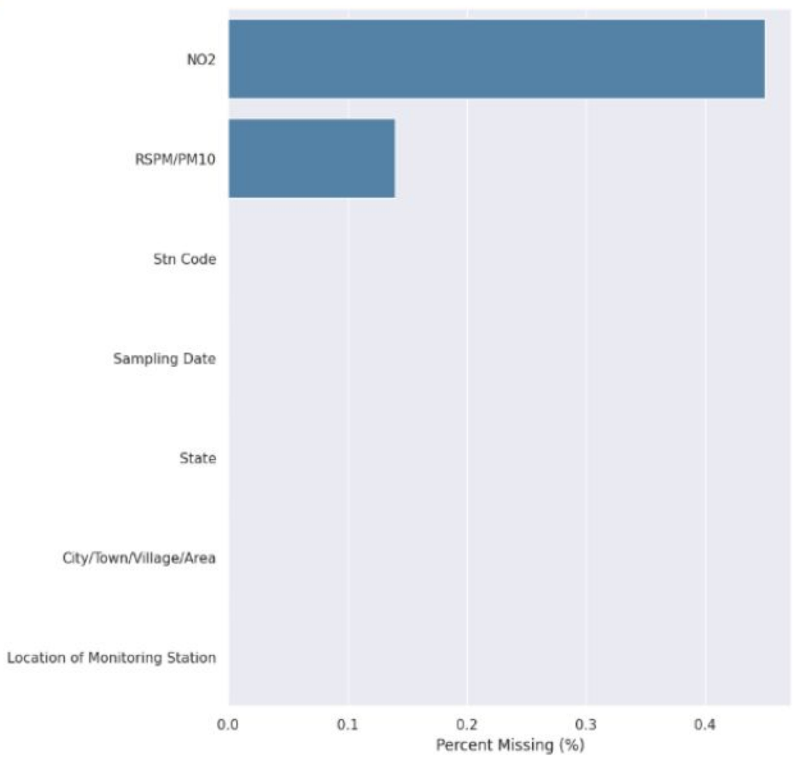
| | Stn Code | Sampling Date | State | City/Town/Village/Area | Location of Monitoring Station | NO2 | RSPM/PM10 | PM 2.5 |
|---|---|---|---|---|---|---|---|---|
| 0 | 38 | 01-02-14 | Tamil Nadu | Chennai | Kathivakkam, Municipal Kalyana Mandapam, Chennai | 17.0 | 55.0 | NaN |
| 1 | 38 | 01-07-14 | Tamil Nadu | Chennai | Kathivakkam, Municipal Kalyana Mandapam, Chennai | 17.0 | 45.0 | NaN |



| | Stn Code | Sampling Date | State | City/Town/Village/Area | Location of Monitoring Station | NO2 | RSPM/PM10 | PM 2.5 |
|---|---|---|---|---|---|---|---|---|
| 2877 | 773 | 24-12-14 | Tamil Nadu | Trichy | Central Bus Stand, Trichy | 17.0 | 95.0 | NaN |
| 2878 | 773 | 31-12-14 | Tamil Nadu | Trichy | Central Bus Stand, Trichy | 16.0 | 94.0 | NaN |



```
PM 2.5                            2879
NO2                                 13
RSPM/PM10                            4
Stn Code                             0
Sampling Date                        0
State                                0
City/Town/Village/Area               0
Location of Monitoring Station       0
dtype: int64
```

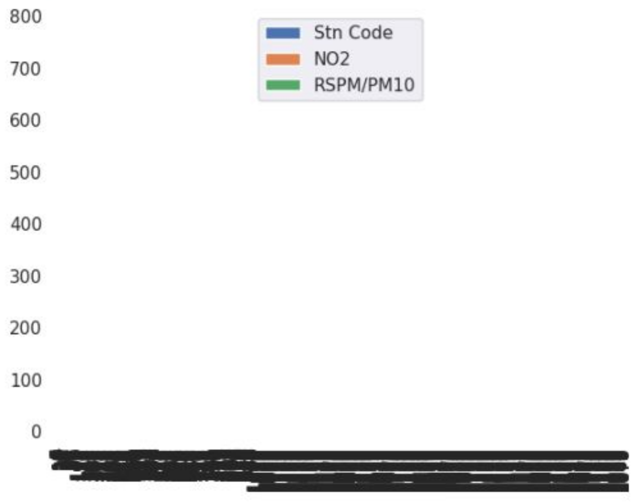


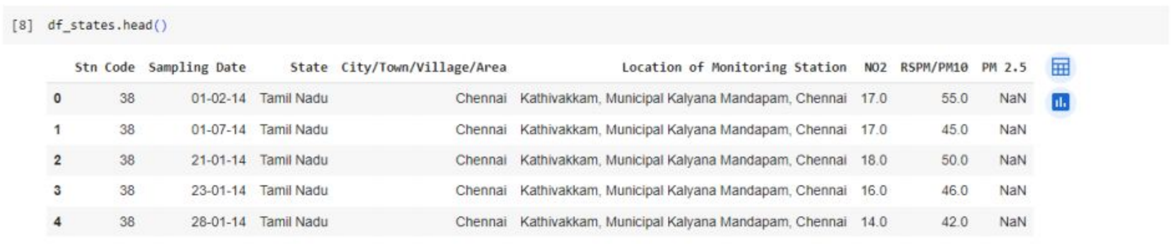
| | Null Count | Percent Missing (%) |
|---|---|---|
| **NO2** | 13 | 0.45 |
| **RSPM/PM10** | 4 | 0.14 |
| **Stn Code** | 0 | 0.00 |
| **Sampling Date** | 0 | 0.00 |
| **State** | 0 | 0.00 |
| **City/Town/Village/Area** | 0 | 0.00 |
| **Location of Monitoring Station** | 0 | 0.00 |

| | Null Count | Percent Missing (%) |
|---|---|---|
| **NO2** | 13 | 0.45 |
| **RSPM/PM10** | 4 | 0.14 |
| **Stn Code** | 0 | 0.00 |
| **Sampling Date** | 0 | 0.00 |
| **State** | 0 | 0.00 |
| **City/Town/Village/Area** | 0 | 0.00 |
| **Location of Monitoring Station** | 0 | 0.00 |

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2879 entries, 0 to 2878
Data columns (total 7 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Stn Code                        2879 non-null   int64
 1   Sampling Date                   2879 non-null   object
 2   State                           2879 non-null   object
 3   City/Town/Village/Area          2879 non-null   object
 4   Location of Monitoring Station  2879 non-null   object
 5   NO2                             2879 non-null   float64
 6   RSPM/PM10                       2879 non-null   float64
dtypes: float64(2), int64(1), object(4)
memory usage: 157.6+ KB
<ipython-input-110-4697573c63a4>:2: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a future version, it will defaul
  df = df_states.fillna(df_states.mean())
```

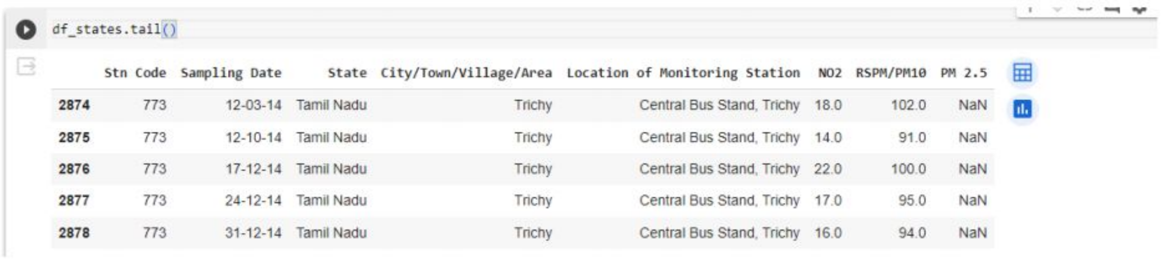
```
<Axes: >
```





```
[8] df_states.head()
```

| | Stn Code | Sampling Date | State | City/Town/Village/Area | Location of Monitoring Station | NO2 | RSPM/PM10 | PM 2.5 |
|---|---|---|---|---|---|---|---|---|
| **0** | 38 | 01-02-14 | Tamil Nadu | Chennai | Kathivakkam, Municipal Kalyana Mandapam, Chennai | 17.0 | 55.0 | NaN |
| **1** | 38 | 01-07-14 | Tamil Nadu | Chennai | Kathivakkam, Municipal Kalyana Mandapam, Chennai | 17.0 | 45.0 | NaN |
| **2** | 38 | 21-01-14 | Tamil Nadu | Chennai | Kathivakkam, Municipal Kalyana Mandapam, Chennai | 18.0 | 50.0 | NaN |
| **3** | 38 | 23-01-14 | Tamil Nadu | Chennai | Kathivakkam, Municipal Kalyana Mandapam, Chennai | 16.0 | 46.0 | NaN |
| **4** | 38 | 28-01-14 | Tamil Nadu | Chennai | Kathivakkam, Municipal Kalyana Mandapam, Chennai | 14.0 | 42.0 | NaN |



```
df_states.tail()
```

| | Stn Code | Sampling Date | State | City/Town/Village/Area | Location of Monitoring Station | NO2 | RSPM/PM10 | PM 2.5 |
|---|---|---|---|---|---|---|---|---|
| **2874** | 773 | 12-03-14 | Tamil Nadu | Trichy | Central Bus Stand, Trichy | 18.0 | 102.0 | NaN |
| **2875** | 773 | 12-10-14 | Tamil Nadu | Trichy | Central Bus Stand, Trichy | 14.0 | 91.0 | NaN |
| **2876** | 773 | 17-12-14 | Tamil Nadu | Trichy | Central Bus Stand, Trichy | 22.0 | 100.0 | NaN |
| **2877** | 773 | 24-12-14 | Tamil Nadu | Trichy | Central Bus Stand, Trichy | 17.0 | 95.0 | NaN |
| **2878** | 773 | 31-12-14 | Tamil Nadu | Trichy | Central Bus Stand, Trichy | 16.0 | 94.0 | NaN |