# **Data Mining Project Report: NYC Taxi Dataset Preprocessing**

**Instructor**: Dr. Shamal Taha  
**Course:** Data Mining  
**Assignment:** Dataset Selection and Preprocessing

**Group 5:** Omar Safar, Ahmed Bahdin, Aya Firas, and Shelan Mohammedtahir.

## **1. Introduction**

The objective of this project is to gather, examine, and prepare a real-world dataset for data mining activities. The NYC Yellow Taxi trip data for October, November, and December 2023 is the selected dataset. It includes comprehensive records of taxi journeys, including times, distances, rates, passenger counts, and extra fees.

Reason for Dataset Selection:

1. Size and Complexity: With millions of rows, the dataset is appropriate for real-world preprocessing, cleaning, and exploratory data analysis practice.
2. Rich Features: Provides both categorical and numerical data, enabling a variety of analytics (passenger behavior, fare prediction, travel patterns).
3. Availability: Reproducible and made publicly available by the NYC Taxi & Limousine Commission.
4. Real-world Uses: Beneficial for financial analytics, anomaly identification, traffic analysis, and predictive modeling.

## **2. Dataset Description**

The combined dataset consists of **over 9.8 million rows** and **19 columns**, including:

* **Trip Information:** tpep\_pickup\_datetime, tpep\_dropoff\_datetime, trip\_distance
* **Financial Information:** fare\_amount, total\_amount, extra, tip\_amount, tolls\_amount, mta\_tax, improvement\_surcharge, congestion\_surcharge, Airport\_fee
* **Passenger Information:** passenger\_count, payment\_type
* **Location Information:** PULocationID, DOLocationID
* **Other:** VendorID, RatecodeID, store\_and\_fwd\_flag

This dataset provides a strong foundation for exploring both **numerical trends** (distances, fares) and **categorical patterns** (payment types, rate codes, locations).

## **3. Preprocessing and Cleaning Steps**

### **3.1 Combining Datasets**

To build a single, cohesive dataset, the three monthly CSV files were loaded one at a time and then joined using pandas.concat().

Reasoning:

* Consistency is ensured, and a thorough examination of patterns over time is made possible by combining all months.
* Combining datasets expedites preprocessing and prevents code duplication.

**Code Example:**

import pandas as pd  
  
# Load each month  
df\_oct = pd.read\_csv("NYC\_Taxi\_Dataset\_October.csv")  
df\_nov = pd.read\_csv("NYC\_Taxi\_Dataset\_November.csv")  
df\_dec = pd.read\_csv("NYC\_Taxi\_Dataset\_December.csv")  
  
# Combine into a single DataFrame  
df = pd.concat([df\_oct, df\_nov, df\_dec], ignore\_index=True)

### **3.2 Converting Dates**

Using pd.to\_datetime(), the timestamps for pickup and dropoff were transformed into datetime objects.

reasoning:

* ensures that time-related features (such as trip duration and peak hours) are handled correctly for further analysis.
* enables feature engineering and filtering, such as isolating day, hour, or weekday.

**Code Example:**

df["tpep\_pickup\_datetime"] = pd.to\_datetime(df["tpep\_pickup\_datetime"], errors='coerce')  
df["tpep\_dropoff\_datetime"] = pd.to\_datetime(df["tpep\_dropoff\_datetime"], errors='coerce')

### **3.3 Inspecting Missing and Zero Values**

Percentage of missing (NaN) and zero values was calculated for all columns to identify critical issues:

|  |  |
| --- | --- |
| **Column** | **% NaN + 0** |
| tolls\_amount | 91.74 |
| Airport\_fee | 87.63 |
| extra | 41.10 |
| tip\_amount | 22.29 |
| congestion\_surcharge | 6.47 |
| payment\_type | 3.04 |
| mta\_tax | 0.75 |
| passenger\_count | 0.0045 |
| improvement\_surcharge | 0.0034 |
| trip\_distance, fare\_amount, total\_amount | 0 |
| Other columns | 0 |

**Reasoning:**

* Critical columns for analysis are trip\_distance, fare\_amount, total\_amount, and passenger\_count. These have minimal missing or zero values.
* Optional columns such as tolls\_amount and Airport\_fee have high missing percentages and will be kept or filled as needed.

**Code Example:**

# Calculate percentage of NaN + zeros per column  
nulls = ((df.isna() | (df==0)).mean() \* 100).sort\_values(ascending=False)  
print(nulls)

### **3.4 Cleaning Strategy**

**Crucial Columns**:

* Use NaN in place of zeros.
* Remove rows that have zero or missing values.
* Eliminate anomalies such as trip\_distance > 300 miles, negative fares, and negative total sums.

**Columns that are optional:**

* To preserve as much information as possible, keep missing or zero values (which can be filled with 0 or mode).

**Reasoning:**

* guarantees that the dataset contains relevant, high-quality records for analysis.
* minimizes noise from erroneous or excessive numbers without needlessly eliminating a lot of data.

**Code Example:**

import numpy as np  
  
# Replace 0 with NaN in critical columns  
critical\_cols = ["trip\_distance", "fare\_amount", "total\_amount", "passenger\_count"]  
df[critical\_cols] = df[critical\_cols].replace(0, np.nan)  
  
# Drop rows with NaN in critical columns  
df\_clean = df.dropna(subset=critical\_cols)  
  
# Remove unrealistic values  
df\_clean = df\_clean[(df\_clean["trip\_distance"] < 300) &   
 (df\_clean["fare\_amount"] > 0) &  
 (df\_clean["total\_amount"] > 0)]

### **3.5 Saving the Cleaned Dataset**

Both **compressed** and **uncompressed** versions were saved for efficient storage and sharing:

# Save cleaned dataset  
df\_clean.to\_csv("NYC\_Taxi\_Cleaned.csv", index=False)  
df\_clean.to\_csv("NYC\_Taxi\_Cleaned.csv.gz", index=False, compression="gzip")

## **4. Descriptive Statistics**

### **4.1 Numerical Columns**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Column** | **Count** | **Mean** | **Std** | **Min** | **25%** | **50%** | **75%** | **Max** |
| trip\_distance | 9,813,848 | 3.87 | 163.6 | 0.01 | 1.06 | 1.78 | 3.39 | 205,544 |
| fare\_amount | 9,813,848 | 19.73 | 18.5 | -1087 | 9.3 | 14.2 | 22.6 | 2,320 |
| total\_amount | 9,813,848 | 28.86 | 23.5 | -1094 | 16.1 | 21.48 | 31.56 | 2,372 |
| passenger\_count | 9,813,848 | 1.15 | 0.55 | 0 | 1 | 1 | 1 | 4 |

### **4.2 Categorical Columns**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Count** | **Unique** | **Top** | **Freq** |
| RatecodeID | 9,813,848 | 14 | 1 | 8,917,170 |
| store\_and\_fwd\_flag | 9,813,848 | 3 | N | 9,468,628 |
| congestion\_surcharge | 9,813,848 | 10 | 2.5 | 8,699,559 |
| Airport\_fee | 9,813,848 | 8 | 0 | 8,599,598 |

## **5. Potential Further Analysis**

Numerous data mining activities are possible with this dataset, including:

**Analysis of the Trip:**

Peak times, most-traveled routes, and distribution of trip distances

**Analyzing finances:**

Trends in fare and tipping, revenue per day and month

**Finding Anomalies:**

Unusual lengthy journeys or negative fares

**Learning Machines:**

Estimate fare or gratuity amounts based on the specifics of the trip.

Organize travel by time and place to gain traffic insights.

**Visualization:**

Heatmaps showing the locations of pickup and drop-off

Fare, journey distance, and tip histograms

## **6. Conclusion**

Critical columns were cleaned, outliers were eliminated, and the dataset was prepared for analysis thanks to the preprocessing. High missing value optional columns were kept for possible additional analysis.

The goals of this Data Mining course project have been met since the cleaned dataset is now appropriate for exploratory analysis, predictive modeling, and visualization.