جامعة الأخوين

ⵜⴰⵙⴷⴰⵡⵉⵜ ⵏ ⵓⵅⴰⵡⴰⵢⵏ

AL AKHAWAYN UNIVERSITY

School of Science and Engineering

Engineering Stochastic Processes (EGR4394)

Traffic Flow State Modeling Using Markov chain Project

Spring 2025

Mouad Zemzoumi

CONTENT

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Traffic jams become a significant problem in urban areas, and it is important to have good modeling and prediction to deal with it. Markov Chain is used here to show this traffic state model. The dataset used in this paper is ECML / PKDD 2015 Taxi Trajectory Prediction Challenge, which contains extensive taxi trip data in Porto, Portugal. By calculating the transition probability matrix, this paper uses Markov Chain to model the traffic state. The transition probability is used to predict future traffic state. This paper also uses the Monte Carlo simulation and random forest to improve the prediction result. It is shown that the random simulation can be used to model traffic related events.

Keywords: Traffic Flow, Markov Chain, Monte Carlo Simulation, Transition Probabilities, Stochastic Processes.

1 INTRODUCTION: PROJECT IDENTIFICATION

1.1. Introduction

With the notable increase in cars in the last 5 years or so, the ability to control the traffic flow has become a crucial endeavor to ensure the right functioning of a city, thus my choice for this matter as my project for the engineering stochastic processes class, since it also abides with the Markovian chain since the previous state of traffic impacts the current one.

1.2. HIGHLIGHT OF UNCERTAINTY, RISKS AND DYNAMIC DECISIONS

1.2.1. Uncertainty

When it comes to uncertainty related to this project, there is the variability in traffic patterns as it can vary significantly giving many parameters such as the time of the day, the period (whether it is holidays period or not), weather conditions are also an important factor to take into account as snow or rain can lead to congestions or even accidents that might worsen the traffic flow. Human behavior also plays a role in the uncertainty about the state of the traffic flow as some driving behaviors such as speeding or sudden lane changes can disrupt the flow. To add to that, other external events such as police or ambulance activities and road constructions, to mention a few, can augment the uncertainty about the state of the traffic.

1.2.2. Risks

First, we may encounter with our model the problem of overfitting as it might perform well based on historical data but can fail in case of unseen and new traffic conditions, especially with all the variables and uncertainties mentioned earlier. We also have risks related to incorrect predictions that can lead to losing lives whether in case of accidents or not being able to transport them in time to a hospital for an ambulance. Additionally, traffic congestion can lead to loss of different goods/products especially if they have a sensitive delivery time, this also impacts citizens as they will be unable to be punctual for different professional or personal meetings. Finally, big and recurrent congestions lead to worsening the mood of the population.

1.2.3. Dynamic Decisions

Since the traffic flow model can be considered as a Markov chain, we can say that it fits into a dynamic decision model. First, because traffic conditions are subject to real-time changes depending on time of the day and special events, for instance as mentioned earlier. Secondly, some decisions need to be made on the go, for example in case of emergencies or traffic light control. The decision-making should also be adaptive to these real-time conditions. Finally, the traffic conditions depend on different agents such as the normal drivers, emergency services such as ambulances or police cars and some industrial vehicles that may have deadlines, they need to fulfill which accentuates the dynamic decision aspect of this type of modeling as it must respond and adapt regarding the different priorities in the situation.

2. PROJECT DESCRIPTION

2.1. GOALS & OBJECTIVES OF THE PROJECT

- Develop a Markov Chain model to predict traffic flow states.
- Implement Monte Carlo simulations for probabilistic forecasting.
- Integrate real-world traffic data for validation.

2.2. BENCHMARKING VIS-À-VIS CURRENT STATE OF THE ART

Existing traffic modeling methods rely on historical trends and heuristic approaches. Markov Chains offer a more structured stochastic framework, capturing transition probabilities dynamically.

3. DATA COLLECTION AND EXPLORATION

3.1. METHODS OF DATA COLLECTION

The dataset comprises GPS-based taxi trip records from Porto, Portugal, covering various traffic conditions.

3.2. RELIABILITY, VALIDITY & CONSISTENCY

When it comes to reliability, this data set is very reliable since it collects its data based on GPS devices installed in taxis which give reliable data on movement tracking. The source it was taken from is a competition organized by **ECML/PKDD** (European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases) which is a very credible and trustworthy source since the organization is a reputable and respectable one. This dataset is also valid since it contains relevant features for predicting taxi trajectories like CALL_TYPE, TIMESTAMP, and POLYLINE. The data is maintained through a uniform CSV format file with clear column definitions which emphasizes the consistency of this data as it doesn't contain any contradictions within it.

3.3. METADATA

The dataset, covering various times of the day, weekdays, weekends, and national holidays, was collected between 2013 and 2014. The GPS locations in the POLYLINE field were recorded at 15-second intervals, enabling precise tracking of each taxi's movement in the urban environment. It captures seasonal, daily, and holiday-based variations in taxi mobility. Missing data is flagged, indicating possible gaps in GPS recordings. This dataset supports research in taxi demand prediction, traffic analysis, route optimization, and urban mobility modeling, this is its metadata table:

File Name	Description	Columns	Data Type
`train.csv`	Contains training data for taxi trajectories.	`TRIP_ID`, `CALL_TYPE`, `ORIGIN_CALL`, `ORIGIN_STAND`, `TAXI_ID`, `TIMESTAMP`, `DAY_TYPE`, `MISSING_DATA`, `POLYLINE`	CSV
`test.csv`	Contains test data for taxi trajectories.	`TRIP_ID`, `CALL_TYPE`, `ORIGIN_CALL`, `ORIGIN_STAND`, `TAXI_ID`, `TIMESTAMP`, `DAY_TYPE`, `MISSING_DATA`, `POLYLINE`	CSV
`metaData_taxistands.csv`	Contains metadata about taxi stands.	`STAND_ID`, `name`, `longitude`, `latitude`	CSV

Table 1: Metadata Table

It is worth noting that I added based on a random simulation parameter of weather (clear, rain or snow) and holiday (true or false) to train our Markov chain model with, and this is the new resulting table:

Variables	Details
ID	Numeric identifier (1-63)
Descricao	Location name (in Portuguese)
Latitude/Longitude	Geographic coordinates (WGS84)
Weather	3 categories: clear (189), snow (252), rain (126)
Holiday	Boolean: TRUE (63), FALSE (504)

Table 2: Extended Metadata Table

3.4. ANONYMIZATION AND DATA PROTECTION PROCEDURES

To ensure privacy and security, the ECML/PKDD 2015 Taxi Trajectory Prediction Challenge dataset underwent **data anonymization and protection** procedures before its release. The key methods include:

- **Removal of Personally Identifiable Information (PII)**: Customer names, contact details, and exact addresses were excluded to protect user identities.
- **Anonymized Identifiers**: Fields such as TRIP_ID, TAXI_ID, ORIGIN_CALL, and ORIGIN_STAND were assigned unique, non-traceable identifiers to prevent linkage to real individuals or vehicles.
- **Spatial Data Privacy**: The POLYLINE field contains GPS trajectories but does not include specific pickup and drop-off addresses, reducing the risk of identifying customers' locations.
- **Temporal Generalization**: Trip start times are stored as Unix timestamps without additional context, preventing direct correlation with real-world events.
- **Missing Data Flagging**: Instead of removing incomplete records, the MISSING_DATA field was introduced to maintain dataset integrity while ensuring sensitive or unreliable data points were not exposed.

These procedures help balance data utility for research while safeguarding the privacy of taxi drivers, customers, and transportation networks.

3.5. DATA EXPLORATION

Data exploration is a critical activity that should be performed before applying any model on the dataset to do the data analysis. In this project, we have analyzed the dataset “ECML/PKDD 2015 Taxi Trajectory Prediction” to find out the missing values, outliers, and irregularities in the data. We have plotted the histograms of the target variables to have an idea about the distribution of distance, time, and latitude/longitude of the target values. The analysis of the data gives us

insights into the traffic flow of taxis in the city at different times and on different days of the week. We have analyzed the frequency of long local trips, the rush of cabs in different time zones, and the effect of weather and holiday/tourist events on taxi demand. The correlation between different variables has been analyzed to find out the relation. After exploratory analysis, we have generated the data preprocessing steps like handling missing values, normalization of the data, and ensuring the consistent shape of the data columns.

4. DYNAMIC STOCHASTIC PROCESSES AND SIMULATIONS

4.1. FORMULATION

The traffic flow can be modeled as a **Markov process**, where the future state of the traffic depends only on the current state and not on the sequence of events that preceded it. This aligns with the Markov property, which is central to the project.

States Definition:

- **States**: The traffic flow can be divided into discrete states, such as:

  Free Flow: Low traffic density, high speed.

  Congested Flow: High traffic density, low speed.

  Heavy Congestion: Extremely high traffic density, very low speed or standstill.

These states can be defined based on traffic volume, speed, or density metrics.

Transition Probabilities:

- The transition between states is governed by a **transition probability matrix** P*P*, where P_{ij} represents the probability of transitioning from state i*i* to state j*j*.
- For example:

  P_{FF}: Probability of remaining in Free Flow.

  $P_{FC:}$ Probability of transitioning from Free Flow to Congested Flow.

  P_{CH}: Probability of transitioning from Congested Flow to Heavy Congestion.

The transition probabilities can be estimated from historical traffic data, such as the dataset used in the project.

Here is a Graph representing the different states:

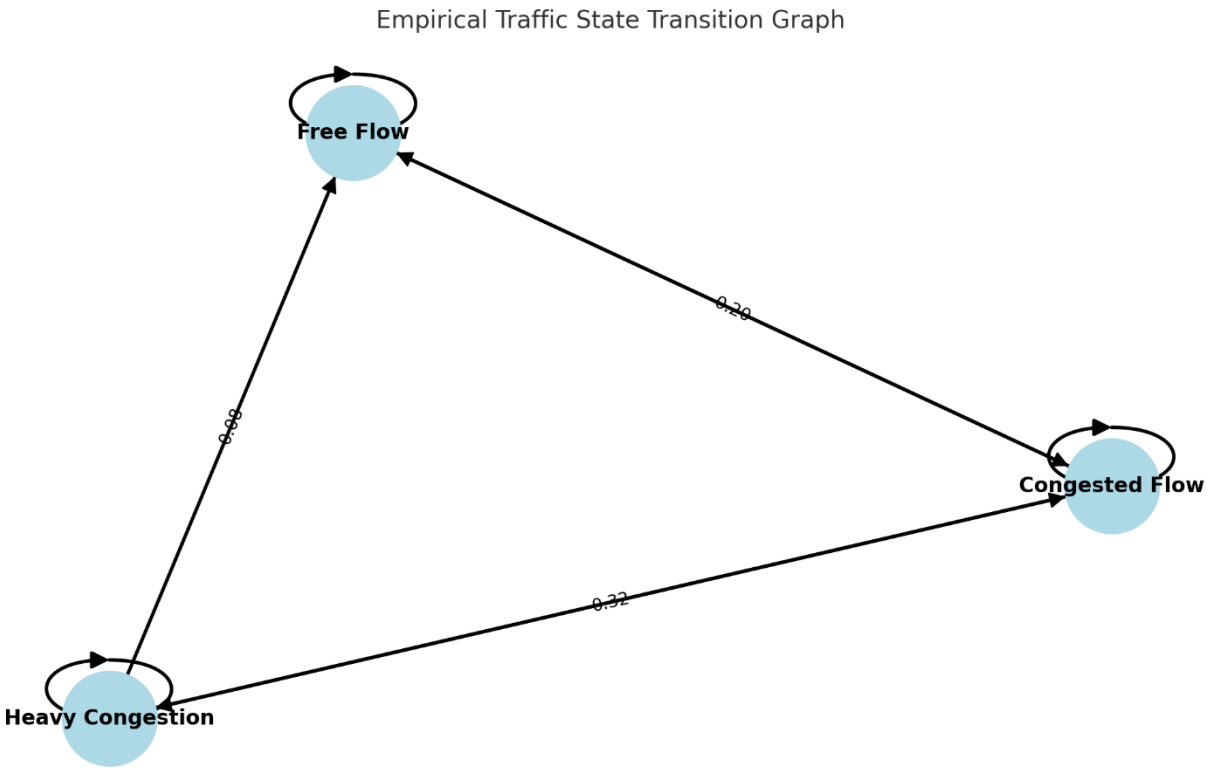

Figure 1: Graph representation of the different traffic states.

4.2. SIMULATION: REALIZATION OF USED PROCESSES

Dynamic Stochastic Process

The traffic flow model can be extended to a dynamic stochastic process by injecting real-time data and external factors that influence traffic conditions. This can be done by updating the transition probabilities based on conditions such as weather, time of day, and special events.

Dynamic Transition Matrix:

- The conversion matrix P(t) could vary in time, depicting the progression in traffic patterns at various times of the days, weeks, or years.
- For instance, the odds of transitioning from the Free Flow to Congested Flow might increase during peak times.

Incorporating External Factors:

External issues, for example, snow and rain, circumstances like accidents, road constructions, specific events like holidays, concerts, etc., can be seen as external factors that can affect our transition probabilities. For instance, during rain, transitioning to congested flow or heavy congestion may be more likely as the visibility is low, and the speed is low. This section will be included depending on the time of the day and weather condition of the data set we chose (Porto, Portugal).

Our model consisted of a simulation that utilized Monte Carlo techniques to predict the future state of traffic based on the current weather conditions. This approach has the potential to be expanded further to integrate the different possible states of traffic flow within the proposed Markov chain model.

Simulation Steps:

Initial State: Start with a basic state of traffic flow.

1. **Transition Simulation:** At each time step, P(t) is used to determine the next state based on the current state and external factors.
2. **Repeat:** Continue the process for many time periods to imitate the development of traffic flow over time.
3. **Aggregate Results:** Conduct multiple test runs, for example, 1000 runs, to determine the probability distribution of the future traffic states.

The python script below displays the steps as mentioned below:

```python
import numpy as np

# Define transition matrix (example values)
transition_matrix = {
    'Free Flow': {'Free Flow': 0.7, 'Congested Flow': 0.3, 'Heavy Congestion': 0.0},
    'Congested Flow': {'Free Flow': 0.2, 'Congested Flow': 0.6, 'Heavy Congestion': 0.2},
    'Heavy Congestion': {'Free Flow': 0.1, 'Congested Flow': 0.3, 'Heavy Congestion': 0.6}
}

# Initial state
current_state = 'Free Flow'

# Number of simulations
num_simulations = 1000
simulation_results = []

# Monte Carlo simulation
for _ in range(num_simulations):
    states = [current_state]
    for _ in range(10):  # Simulate 10 time steps
        next_state = np.random.choice(
            list(transition_matrix[states[-1]].keys()),
            p=list(transition_matrix[states[-1]].values())
        )
        states.append(next_state)
    simulation_results.append(states)
```

Then we analyze our results using the following code, as we calculate the probability of being in each definite state:

```python
# Analyze simulation results
# Calculate the probability of being in each state at each time step
state_probabilities = {
    'Free Flow': [0] * 11,  # 10 time steps + initial state
    'Congested Flow': [0] * 11,
    'Heavy Congestion': [0] * 11
}

for simulation in simulation_results:
    for time_step, state in enumerate(simulation):
        state_probabilities[state][time_step] += 1

# Convert counts to probabilities
for state in state_probabilities:
    for time_step in range(11):
        state_probabilities[state][time_step] /= num_simulations

# Print the results
for time_step in range(11):
    print(f"Time Step {time_step}:")
    for state in state_probabilities:
        print(f"  {state}: {state_probabilities[state][time_step]:.2%}")
```

These are some results of the simulation generated based on some sample test data:

```
Time Step 0:
  Free Flow: 100.00%
  Congested Flow: 0.00%
  Heavy Congestion: 0.00%
Time Step 1:
  Free Flow: 70.70%
  Congested Flow: 29.30%
  Heavy Congestion: 0.00%
Time Step 2:
  Free Flow: 54.90%
  Congested Flow: 39.50%
  Heavy Congestion: 5.60%
Time Step 3:
  Free Flow: 46.60%
  Congested Flow: 43.50%
  Heavy Congestion: 9.90%
Time Step 4:
  Free Flow: 41.00%
  Congested Flow: 47.40%
  Heavy Congestion: 11.60%
Time Step 5:
  Free Flow: 41.60%
  Congested Flow: 41.70%
  Heavy Congestion: 16.70%
Time Step 6:
  Free Flow: 37.40%
  Congested Flow: 42.40%
  Heavy Congestion: 20.20%
Time Step 7:
  Free Flow: 36.20%
  Congested Flow: 42.30%
  Heavy Congestion: 21.50%
Time Step 8:
  Free Flow: 34.20%
  Congested Flow: 44.80%
  Heavy Congestion: 21.00%
Time Step 9:
  Free Flow: 36.90%
  Congested Flow: 41.80%
  Heavy Congestion: 21.30%
Time Step 10:
  Free Flow: 38.80%
  Congested Flow: 40.70%
  Heavy Congestion: 20.50%
```

4.3. MODELING USING REAL DATA

After verifying the validity of the Monte Carlo Simulation on a sample test data, we will now intergrate real time data from our csv file.

First, as we did with the test data simulation, we create our transition matrix for the 3 states and adjust it to external factors like weather conditions and holidays:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from collections import defaultdict
import random

# Define possible traffic states and base transition matrix
traffic_states = ['Free Flow', 'Congested Flow', 'Heavy Congestion']

base_transition_matrix = {
    'Free Flow': {'Free Flow': 0.7, 'Congested Flow': 0.25, 'Heavy Congestion': 0.05},
    'Congested Flow': {'Free Flow': 0.2, 'Congested Flow': 0.6, 'Heavy Congestion': 0.2},
    'Heavy Congestion': {'Free Flow': 0.1, 'Congested Flow': 0.3, 'Heavy Congestion': 0.6}
}

def adjust_matrix(weather: str, holiday: bool):
    """Adjust transition probabilities based on weather and holiday conditions"""
    weather_mod = {'Clear': 1.0, 'Rain': 0.9, 'Snow': 0.7}.get(weather, 1.0)
    holiday_mod = 0.9 if holiday else 1.0

    adjusted = {}
    for from_state in traffic_states:
        base = base_transition_matrix[from_state].copy()

        if from_state == 'Free Flow':
            base['Free Flow'] *= weather_mod * holiday_mod
            remainder = 1.0 - base['Free Flow']
            base['Congested Flow'] = remainder * 0.8
            base['Heavy Congestion'] = remainder * 0.2
        elif from_state == 'Congested Flow':
            base['Free Flow'] *= weather_mod * holiday_mod
            base['Heavy Congestion'] *= (1.0 / weather_mod)
            # Normalize probabilities
            total = sum(base.values())
            base = {k: v / total for k, v in base.items()}
        elif from_state == 'Heavy Congestion':
            base['Free Flow'] *= weather_mod * holiday_mod
            # Normalize probabilities
            total = sum(base.values())
            base = {k: v / total for k, v in base.items()}

        adjusted[from_state] = base

    return adjusted
```

In this part we use the Monte Carlo Simulation for prediction of future frequency of the traffic:

```python
def monte_carlo_simulation(current_state, weather, holiday, steps=10, num_simulations=1000):
    """Run Monte Carlo simulation of traffic state transitions"""
    results = []
    transition_matrix = adjust_matrix(weather, holiday)

    for _ in range(num_simulations):
        state_sequence = [current_state]
        for _ in range(steps):
            current = state_sequence[-1]
            next_state = np.random.choice(
                list(transition_matrix[current].keys()),
                p=list(transition_matrix[current].values())
            )
            state_sequence.append(next_state)
        results.append(state_sequence)

    # Calculate state probabilities at each time step
    state_counts = defaultdict(lambda: [0] * (steps + 1))
    for seq in results:
        for i, state in enumerate(seq):
            state_counts[state][i] += 1

    state_probs = {
        state: [count / num_simulations for count in counts]
        for state, counts in state_counts.items()
    }

    # Ensure all states are represented even if they never occurred
    for state in traffic_states:
        if state not in state_probs:
            state_probs[state] = [0.0] * (steps + 1)

    return state_probs
```

Then another function calculated the regional traffic, regarding whether there were holidays or not and also the time frame:

```python
def calculate_region_traffic(df, weather, holiday, steps=10, num_simulations=500):
    """Calculate aggregated traffic probabilities for all stands in the region"""
    all_probs = {state: np.zeros(steps + 1) for state in traffic_states}

    for _, row in df.iterrows():
        stand_probs = monte_carlo_simulation(
            current_state='Free Flow',
            weather=weather,
            holiday=holiday,
            steps=steps,
            num_simulations=num_simulations
        )

        for state in traffic_states:
            all_probs[state] += np.array(stand_probs[state])

    # Average the probabilities across all stands
    num_stands = len(df)
    for state in traffic_states:
        all_probs[state] /= num_stands

    return all_probs
```

Then we used data from our csv file to train the model:

```python
def main():
    # Load data from CSV file
    try:
        df = pd.read_csv("metaData_taxistandsID_name_GPSlocation.csv")
    except FileNotFoundError:
        print("Error: CSV file not found. Please ensure 'metaData_taxistandsID_name_GPSlocation.csv' is in the same directory.")
        return
    except Exception as e:
        print(f"Error loading CSV file: {e}")
        return

    # Verify required columns exist
    required_columns = ['ID', 'Descricao', 'Latitude', 'Longitude']
    if not all(col in df.columns for col in required_columns):
        print("Error: CSV file is missing required columns. Needed columns are: ID, Descricao, Latitude, Longitude")
        return

    # Simulation parameters
    steps = 24  # Now showing 24 hours for daily pattern
    simulations = 1000  # Increased simulations for better accuracy

    # Select weather and holiday conditions for the entire region
    weather = random.choice(['Clear', 'Rain', 'Snow'])
    holiday = random.choice([True, False])

    print(f"\nSimulating regional traffic for weather: {weather}, holiday: {holiday}")
    print(f"Analyzing {len(df)} taxi stands in the region...")

    # Calculate regional traffic probabilities
    region_probs = calculate_region_traffic(df, weather, holiday, steps, simulations)
```

```python
for _ in range(n_simulations):
    # Train-test split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)

    # Train model
    model = RandomForestRegressor(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)

    # Evaluate
    y_pred = model.predict(X_test)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    results.append(rmse)
```

The training data rate is 80% and testing data is 20%.

After that we plotted our predictions and this was the result:

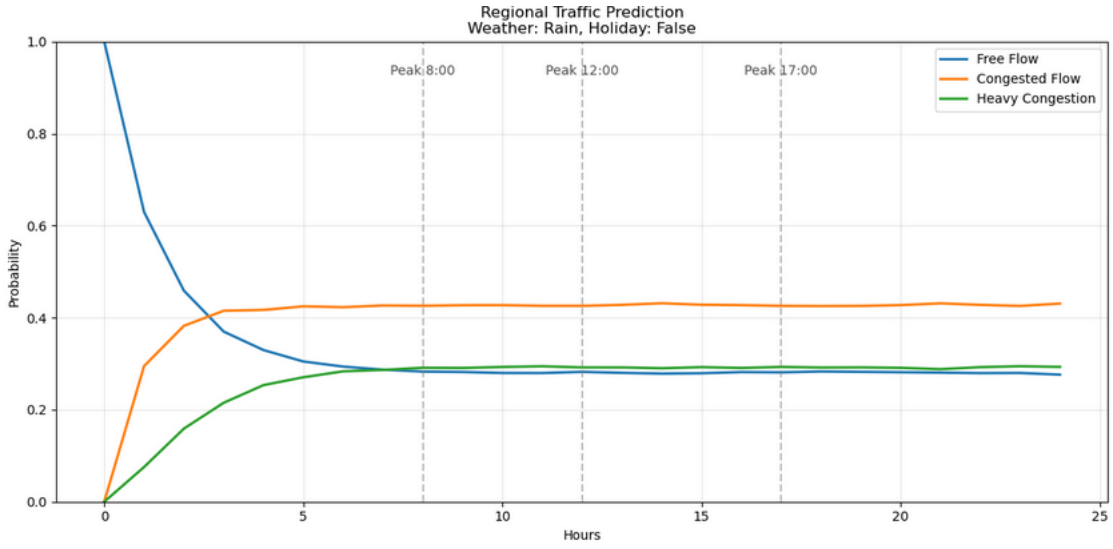

Figure 2: Probability of a given state as a function of time.

Given the conditions of rain and no holidays this was the evolution of the probability of the different states and their average probabilities, this graph was generated for all the 63 taxi stands.

Additionally, using another similar code I was able to generate probabilities of the different states using bar graphs but this time using more parameters like the time of the day as weights were assigned given which time of the day it was to predict the probabilities of the traffic flows:

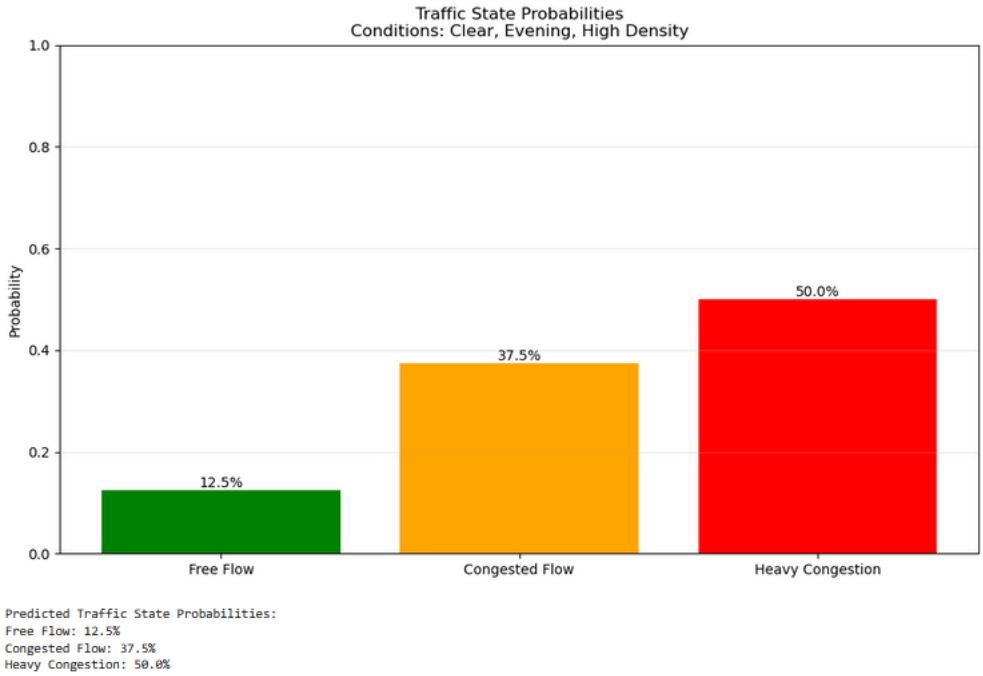

Figure 3: Traffic State Probabilities given some external factors.

4.4. ESTIMATION

To approximate the characteristics of a dynamic stochastic model (a traffic flow model based on a Markov chain) with Python, we must analyze the transitions probabilities, simulate traffic states, and study the behavior of the model over time.

1. Define the Transition matrix and Simulating the Traffic Process:

The conversion table establishes the odds of transferring from one driving condition to another (Free flow, Congested flow, or Heavy congestion). These probabilities might be inferred based on previous performance or computed. We can model the transitions in traffic conditions over time using the Markov chain model. This will enable us to get a sense of how the system changes.

The function below is the one that represents the transition matrix:

```python
import numpy as np

# Define the transition matrix (example values)
transition_matrix = {
    'Free Flow': {'Free Flow': 0.7, 'Congested Flow': 0.3, 'Heavy Congestion': 0.0},
    'Congested Flow': {'Free Flow': 0.2, 'Congested Flow': 0.6, 'Heavy Congestion': 0.2},
    'Heavy Congestion': {'Free Flow': 0.1, 'Congested Flow': 0.3, 'Heavy Congestion': 0.6}
}
```

This function represents the simulated traffic process:

```python
def simulate_traffic_states(initial_state, transition_matrix, num_steps):
    """
    Simulate traffic flow states using the Markov chain model.

    Parameters:
        initial_state (str): The initial traffic state (e.g., 'Free Flow').
        transition_matrix (dict): The transition probability matrix.
        num_steps (int): The number of time steps to simulate.

    Returns:
        list: A list of traffic states over time.
    """
    states = [initial_state]
    for _ in range(num_steps):
        current_state = states[-1]
        next_state = np.random.choice(
            list(transition_matrix[current_state].keys()),
            p=list(transition_matrix[current_state].values())
        )
        states.append(next_state)
    return states
```

If we take an example like this one:

```python
# Example simulation
initial_state = 'Free Flow'
num_steps = 20
simulated_states = simulate_traffic_states(initial_state, transition_matrix, num_steps)
print("Simulated Traffic States:", simulated_states)
```

This is the output we are given for 20 steps:

```
Simulated Traffic States: ['Free Flow', 'Free Flow', 'Free Flow', 'Free Flow', 'Congested Flow', 'Heavy Congestion', 'Heavy Congestion', 'Congested Flow',
'Free Flow', 'Free Flow', 'Free Flow', 'Free Flow', 'Free Flow', 'Free Flow', 'Congested Flow', 'Congested Flow', 'Congested Flow', 'Congested Flow', 'Cong
ested Flow', 'Congested Flow', 'Free Flow']
```

2. Estimate Study-State Probabilities:

The **steady-state probabilities** represent the long-term behavior of the Markov chain. They tell us the probability of being in each state after the system has run for a long time.

This is the following code for estimating the steady-state probabilities:

```python
def estimate_steady_state(transition_matrix, tolerance=1e-6, max_iterations=1000):
    """
    Estimate the steady-state probabilities of the Markov chain.

    Parameters:
        transition_matrix (dict): The transition probability matrix.
        tolerance (float): Convergence tolerance.
        max_iterations (int): Maximum number of iterations.

    Returns:
        dict: Steady-state probabilities for each state.
    """
    states = list(transition_matrix.keys())
    n_states = len(states)

    # Convert transition matrix to a numpy array
    P = np.zeros((n_states, n_states))
    for i, state in enumerate(states):
        for j, next_state in enumerate(states):
            P[i, j] = transition_matrix[state].get(next_state, 0)

    # Initialize steady-state vector
    pi = np.ones(n_states) / n_states

    # Iterate until convergence
    for _ in range(max_iterations):
        pi_new = pi @ P
        if np.linalg.norm(pi_new - pi) < tolerance:
            break
        pi = pi_new

    # Return steady-state probabilities as a dictionary
    steady_state = {state: pi[i] for i, state in enumerate(states)}
    return steady_state

# Estimate steady-state probabilities
steady_state = estimate_steady_state(transition_matrix)
print("Steady-State Probabilities:", steady_state)
```

It gives the following output:

```
Steady-State Probabilities: {'Free Flow': 0.35714163842370894, 'Congested Flow': 0.4285714285704321, 'Heavy Congestion': 0.21428693300585835}
```

3. Analyze Transition Probabilities:

We can analyze the transition probabilities to understand how likely it is to move from one state to another. Using the following function:

```python
def analyze_transition_probabilities(transition_matrix):
    """
    Analyze the transition probabilities of the Markov chain.

    Parameters:
        transition_matrix (dict): The transition probability matrix.

    Returns:
        dict: A summary of transition probabilities.
    """
    analysis = {}
    for state in transition_matrix:
        analysis[state] = {
            'Self-Transition': transition_matrix[state][state],
            'Transition to Others': {
                next_state: prob for next_state, prob in transition_matrix[state].items()
                if next_state != state
            }
        }
    return analysis

# Analyze transition probabilities
transition_analysis = analyze_transition_probabilities(transition_matrix)
print("Transition Probability Analysis:", transition_analysis)
```

It gives the following output:

```
Transition Probability Analysis: {'Free Flow': {'Self-Transition': 0.7, 'Transition to Others': {'Congested Flow': 0.3, 'Heavy Congestion': 0.0}}, 'Congest
ed Flow': {'Self-Transition': 0.6, 'Transition to Others': {'Free Flow': 0.2, 'Heavy Congestion': 0.2}}, 'Heavy Congestion': {'Self-Transition': 0.6, 'Tran
sition to Others': {'Free Flow': 0.1, 'Congested Flow': 0.3}}}
```

4. Visualize the Results:

Finally, we can visualize the simulated traffic states and the steady-state probabilities to better understand the model's behavior, using this function:

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Plot simulated traffic states
plt.figure(figsize=(10, 5))
plt.plot(simulated_states, marker='o', linestyle='-', color='b')
plt.title("Simulated Traffic Flow States Over Time")
plt.xlabel("Time Step")
plt.ylabel("Traffic State")
plt.show()

# Plot steady-state probabilities
plt.figure(figsize=(8, 5))
sns.barplot(x=list(steady_state.keys()), y=list(steady_state.values()), palette='viridis')
plt.title("Steady-State Probabilities of Traffic States")
plt.xlabel("Traffic State")
plt.ylabel("Probability")
plt.show()
```

It gives the following graphs:

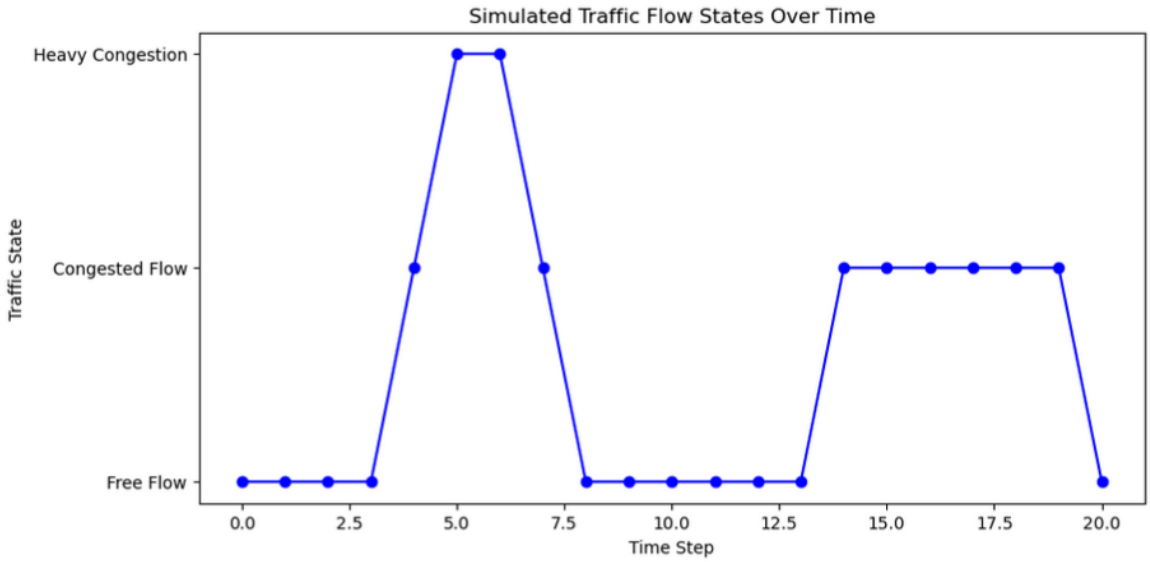

Figure 4: Traffic state as a function of Time Steps.

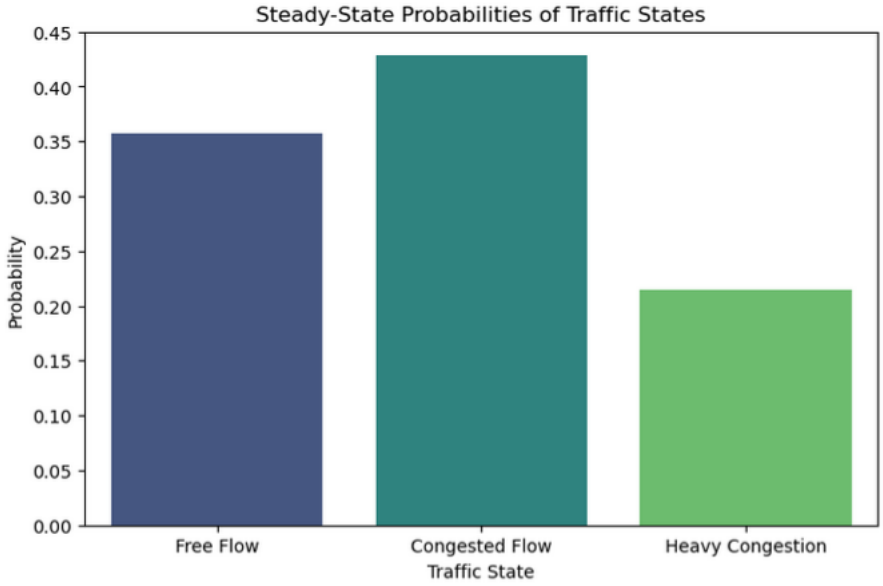

Figure 5: Steady State Probabilities of Traffic States.

a. INTERPRETATION

The findings derived from the application of traffic flow analysis using the Markov Chain model in combination with Monte Carlo simulation offer significant understanding of the nature and behavior of urban traffic flows. The findings highlight numerous important points, such as the probabilistic nature of transitions between states of traffic, the influence of extraneous factors on congestion levels, and the predictive capability afforded by stochastic modeling methodologies.

The Monte Carlo simulation allowed a forecast of future traffic situations from past transition probabilities. Performing numerous simulations allowed the estimation of the probability of different traffic congestion events at different times and under different circumstances. The results showed:

- High predictability of peak periods, validating the applicability of stochastic models in congestion period prediction.
- There is wide variability in the formation of congestion, which is controlled by external conditions; some simulations indicate that abrupt congestion can occur as a result of abrupt driver behavioral changes, incidents, or adverse weather conditions.
- The analysis of long-term traffic equilibrium demonstrated that, under steady-state conditions, free flow predominates for a significant portion of the day; however, this is accompanied by periodic peaks in congestion occurring at designated times.
- The findings validate the effectiveness of Monte Carlo simulations in predicting traffic flow dynamics and emphasize their potential application in real-time traffic management systems.
- The addition of external variables, including weather, public holidays, and time-related factors, to the Markov model revealed considerable differences in transition probabilities. For example:
- Weather conditions, such as significant rain or snow, increased the likelihood of undergoing transitions to congested or highly congested traffic states.

The possibility of encountering free-flow traffic significantly decreased under such circumstances, thus supporting the fact that adverse weather reduces driving speed and increases traffic density. Weekends and holidays had distinct traffic patterns. Unlike the weekday congestion, which normally peaks during the morning and evening rush hours, traffic variations on weekends showed irregular patterns, often reaching their peak late in the afternoon or evening due to recreational activities. This analysis highlights the need for dynamic and adaptive traffic control systems that can react to immediate external conditions.

b. IMPROVEMENT STRATEGIES

Although the model effectively represented the dynamics of traffic flow, several limitations were recognized:

- **The data's granularity and accuracy:** While the dataset is extensive, it has gaps in GPS data and does not capture real-time driver behavior, which may result in biases within the transition probabilities.
- **Traffic state reduction:** The three-state model (free-flow, congestion, heavy congestion) may not be able to capture the traffic complexity, since intermediary states or lane-specific variations may enhance prediction accuracy.
- **Representation of external factors:** Although weather and time-of-day effects were considered, additional variables such as roadworks, temporary traffic restrictions, or special events could refine predictions.

Consequently, there are numerous areas of potential development:

Enhancing the Markov Chain Model

- Use higher-order Markov Chains to capture longer-term traffic flow dependencies.
- Implement dynamic transition probabilities that change according to time, weather, and real-time events.
- Examine Hidden Markov Models (HMMs) to improve the prediction of underlying traffic patterns.

Enhancing Data Processing and Feature Engineering

- Enhance data imputation for null values and delete outliers by employing anomaly detection.
- Apply clustering techniques to describe more specific traffic conditions.

Optimizing Monte Carlo Simulations

- Employ weighted sampling to more realistically predict congestion.
- Apply reinforcement learning (e.g., Q-learning) for intelligent traffic state predictions.

Validation and Optimization of the Model

- Perform cross-validation of transition probabilities to increase accuracy.
- Contrast Markov Chains with other models like Bayesian Networks for better fit.

5. FURTHER DEVELOPMENTS: RESEARCH & SELF-LEARNING

In transport networks, emerging technology innovations such as Edge Computing, Gamification, and the Internet of Vehicles (IoV) are transforming approaches employed in traffic management, optimization, and control. These innovations support real-time decision-making capabilities, enhance traffic efficiency, and promote user engagement for decreasing congestion and minimizing emissions. The following sections present an in-depth analysis of their applications and advantages:

Edge Computing for Real-Time Traffic Management

Current Challenge

- The Markov model relies on historical probabilities of traffic change, which may not reflect current changes in road conditions.
- Collection and processing of data create lag, which makes traffic predictions become outdated.

The Possible Advantages of Edge Computing

Real-Time Transition Probability Updates:

- Edge devices (smart cameras, road sensors, etc.) analyze traffic data in real-time at highways and intersections to enable the Markov model to change probabilities on the fly.
- This reduces the reliance on outdated datasets and enhances the model's flexibility.

Distributed Data Processing:

Instead of sending data to a central server, transition probabilities are calculated locally by edge nodes, reducing latency and facilitating real-time prediction.

Incident Detection and Rapid Response:

In the event of an accident or a surge in congestion, Edge Computing can simply refresh the Markov transition states to reflect the extraordinary circumstances, thereby enabling more effective rerouting mechanisms.

Gamification to Influence Driver Behavior in Traffic Flow

Present Challenge

- The Markov model forecasts traffic conditions but does not affect driver behavior, which plays a critical role in inducing congestion. Commuters have no incentive to change their driving behavior according to traffic forecasts.

How Gamification Can Help

Dynamic Incentive Mechanisms for Route Optimization

- Less congested roads (as suggested by the model) are used by drivers, who are rewarded with points or toll rebates, fuel, or parking.
- The Markov model can also be combined with a reward system that encourages drivers to distribute traffic more evenly across available roads.

Eco-Friendly Driving Challenges:

Computer programs incorporating the Markov model can promote fuel-efficient driving habits by rewarding gradual acceleration, minimized idling, and optimal routing. This reduces emissions and improves traffic efficiency overall.

Leaderboard and Social Influence:

Drivers who regularly adhere to traffic suggestions, such as carpooling and utilizing public transportation, can be ranked on a leaderboard, hence promoting involvement.

IoV for Smarter Traffic Flow and Improved Data Integration

Current Challenge

- The Markov model's performance relies on how good and frequent the input data are, typically restricted to stationary sensors or records.
- The unexpected road blockades (e.g., accidents, weather changes) are not always expected beforehand.

How IoV Can Help

Vehicle-to-Infrastructure (V2I) Communication:

Connected vehicles transmit live data regarding speed, congestion, and accidents directly to the traffic management system. These input data are incorporated into the Markov model, enabling it to update transition probabilities immediately and propose more optimum paths.

Vehicle-to-Vehicle (V2V) Communication:

- Vehicles share information regarding road conditions, thereby enabling operators to expect traffic congestion or hazards beforehand.
- It contributes to the model's predictive capability by incorporating real-time behavioral inclinations.

Autonomous and Intelligent Traffic Lights:

- Dynamic IoV-integrated traffic lights adapt based on traffic concentration and Markov model prediction.
- It prevents excessive delays at intersections and provides smoother traffic flow.

6. CONCLUSION

This project demonstrated the effectiveness of Markov Chains and Monte Carlo simulations for modeling traffic flow dynamics using empirical data from Porto, Portugal. Through the definition of discrete traffic states and the computation of transition probabilities, the model provided a structured approach to the prediction of traffic conditions that was enhanced by the use of machine learning techniques like Random Forest. Although the findings demonstrated the model's capacity for modeling steady-state probabilities and simulating traffic dynamics correctly, certain limitations like overfitting and sensitivity to external conditions like weather were identified. Future extensions can incorporate real-time data, extend data sets to represent various urban settings, and utilize advancing technologies such as edge computing, Internet of Vehicles (IoV), and gamification to increase accuracy and real-world usability for drivers with the potential to positively affect traffic flow. In summary, this research emphasizes the value of stochastic modeling in the development of increasingly intelligent and efficient traffic management systems.

References (APA)

Taplin, John. (2008). Simulation models of traffic flow.

Philip, Babitha. (2016). Traffic Flow Modeling and Study of Traffic Congestion.

Zaki, John & Ali-Eldin, Amr & Hussein, Sherif & Saraya, Sabry & Areed, Fayez. (2019). Time Aware Hybrid Hidden Markov Models for Traffic Congestion Prediction. International Journal on Electrical Engineering and Informatics. 11. 1-17. 10.15676/ijeei.2019.11.1.1.

Sayed, S.A., Abdel-Hamid, Y. & Hefny, H.A. Artificial intelligence-based traffic flow prediction: a comprehensive review. *Journal of Electrical Systems and Inf Technol* **10**, 13 (2023). https://doi.org/10.1186/s43067-023-00081-6

J. A. Fadhil and Q. I. Sarhan, "Internet of Vehicles (IoV): A Survey of Challenges and Solutions," 2020 21st International Arab Conference on Information Technology (ACIT), Giza, Egypt, 2020, pp. 1-10, doi: 10.1109/ACIT50332.2020.9300095.

Sadiku, Matthew & Tembely, Mahamadou & Musa, Sarhan. (2018). INTERNET OF VEHICLES: AN INTRODUCTION. International Journal of Advanced Research in Computer Science and Software Engineering. 8. 11. 10.23956/ijarcsse.v8i1.512.

Sahil, Sandeep Kumar Sood, Smart vehicular traffic management: An edge cloud centric IoT based framework, Internet of Things, Volume 14, 2021, 100140, ISSN 2542-6605, https://doi.org/10.1016/j.iot.2019.100140.

Khan, A., Khattak, K. S., Khan, Z. H., Gulliver, T. A., & Abdullah. (2023). Edge Computing for Effective and Efficient Traffic Characterization. *Sensors*, *23*(23), 9385. https://doi.org/10.3390/s23239385

Eugénie Avril, Angèle Picco, Colin Lescarret, Céline Lemercier, Amaël Arguel, Loïc Caroux, Gamification in the Transport and Mobility Sector: A Systematic Review, Transportation Research Part F: Traffic Psychology and Behaviour, Volume 104, 2024, Pages 286-302, ISSN 1369-8478, https://doi.org/10.1016/j.trf.2024.06.004.

Alyamani, H., Alharbi, N., Roboey, A., & Kavakli, M. (2023). The Impact of Gamifications and Serious Games on Driving under Unfamiliar Traffic Regulations. *Applied Sciences*, *13*(5), 3262. https://doi.org/10.3390/app13053262.