Advanced Transport Network Analysis & Cybercrime Fraud Investigation

Abstract:This document provides an in-depth analysis of transportation network modeling using NetworkX and Matplotlib, along with a detailed investigation into cybercrime fraud trends in England and Wales. The transport model examines metro station connectivity, network visualization, and shortest path analysis, while the cybercrime study explores fraud patterns, victim demographics, and geographical impact. Using Python and R, this research employs data processing, visualization, and predictive analytics to derive actionable insights for transportation efficiency and fraud mitigation.

1. Introduction

Modern societies depend on efficient transportation and robust cybersecurity measures to ensure seamless urban mobility and secure digital environments. This report covers two key aspects:

- **Transport Network Modeling:** Leveraging NetworkX to simulate and optimize metro station connectivity.
- **Cybercrime Fraud Analysis:** Identifying trends and risk factors in fraud cases using statistical analysis and geospatial mapping.

---

2. Transport Network Modeling with NetworkX

2.1 Objectives

- Construct a directed graph to model metro networks.
- Analyze network efficiency using shortest-path algorithms.
- Visualize transport connectivity with Matplotlib.

2.2 Methodology

1. **Graph Construction:**
    - **Nodes:** Metro stations.
    - **Edges:** Connections between stations with attributes (line name, distance).
2. **Visualization:**
    - Distinct colors for different metro lines.
    - Labeling of key transfer stations.
3. **Analysis:**
    - Application of **Dijkstra's Algorithm** for shortest path calculations.
    - Identification of high-degree nodes (**hubs**) for optimization.

2.3 Results & Insights

- The **graph-based** transport model highlights crucial interchange points.
- **Bottlenecks** in network connectivity can be identified for potential optimization.
- Alternative routing solutions improve **efficiency and passenger flow**.

---

3. Cybercrime Fraud Analysis in England & Wales

3.1 Objectives

- Identify fraud hotspots based on regional trends.
- Analyze victim profiles and household vulnerability.
- Compare different types of fraud to uncover dominant patterns.

3.2 Data Sources

- **NFIB Reports (2013-2022):** Longitudinal data on reported fraud cases.
- **Action Fraud Regional Data:** Fraud breakdown by police jurisdictions.
- **Crime Survey for England and Wales (CSEW):** Victim surveys for fraud impact assessment.

3.3 Methodology

1. **Data Processing & Cleaning:**
    - Import datasets using **R (`readxl`, `dplyr`)**.
    - Normalize categorical variables for consistency.
2. **Exploratory Data Analysis (EDA):**
    - **Trend analysis** using line plots to observe fraud growth.
    - **Comparative boxplots** for fraud typology breakdown.
    - **Geospatial heatmaps** to detect regional fraud disparities.
3. **Predictive Modeling (Future Scope):**
    - Machine learning classifiers to predict high-risk fraud zones.

3.4 Key Findings

- Fraud incidents have **increased by over 35%** in the last decade.
- **London and major urban centers** show disproportionately high fraud rates.
- **Elderly individuals and high-income households** are prime fraud targets.

---

4. Comparative Insights & Strategic Recommendations

4.1 Transport Network Optimization

- Prioritize investment in key interchange stations to **reduce congestion**.
- Implement **real-time data monitoring** for adaptive route planning.

4.2 Cybercrime Prevention Strategies

- Launch **public awareness campaigns** targeting vulnerable demographics.
- Strengthen **cybersecurity frameworks** for digital financial transactions.
- Enhance **law enforcement collaboration** to track cross-border fraud operations.

---

5. Conclusion

This study underscores the importance of well-structured transportation networks and proactive fraud prevention strategies. The transport model offers insights into metro connectivity, while the cybercrime analysis highlights significant trends and risk factors. Future research will focus on incorporating real-time transport simulation and AI-driven fraud detection models.

---

6. References

1. Newman, M. (2018). *Networks: An Introduction*. Oxford University Press.
2. Freeman, L. (2020). *The Development of Social Network Analysis*. Routledge.
3. National Fraud Intelligence Bureau (2022). *Annual Fraud Report*.
4. Transport for London (2023). *Metro Network Optimization Study*.
5. UK Home Office (2022). *Cybercrime Trends and Countermeasures*.

Appendix

Figures & Diagrams:

- [Graph representation of metro stations and connectivity]
- [Geospatial heatmap of fraud distribution across England & Wales]
- [Flowchart illustrating data processing methodology]

import networkx as nx import matplotlib.pyplot as plt from matplotlib.lines import Line2D def create_transport_network(): transport_network = nx.DiGraph()

tube_lines = ['Central', 'Piccadilly', 'Northern', 'Bakerloo', 'Circle'] station_names = {

'Central': ['Holborn', 'Chancery Lane', 'St. Paul\'s', 'Bank', 'Liverpool Street'],

'Piccadilly': ['Heathrow Terminal 5', 'Acton Town', 'Hammersmith', 'Earl\'s Court', 'South Kensington'],

'Northern': ['Edgware', 'Camden Town', 'King\'s Cross St. Pancras', 'London Bridge', 'Stockwell'],

'Bakerloo': ['Harrow & Wealdstone', 'Kilburn Park', 'Oxford Circus', 'Waterloo', 'Elephant & Castle'],

'Circle': ['Aldgate', 'Tower Hill', 'Embankment', 'Westminster', 'Victoria']

}

for line in tube_lines:

stations = station_names[line] transport_network.add_nodes_from(stations) distances = [3.0, 2.4, 1.8, 3.6] for i in range(len(stations)-1):

transport_network.add_edge(stations[i], stations[i+1], line=line, distance=distances[i]) return transport_network def visualize_transport_network(transport_network):

pos = nx.spring_layout(transport_network, k=2.5, seed=42)

line_colors = {'Central': 'red', 'Piccadilly': 'blue', 'Northern': 'green', 'Bakerloo': 'orange', 'Circle': 'purple'} fig, ax = plt.subplots(figsize=(15, 10)) legend_lines = [] for line, color in line_colors.items():

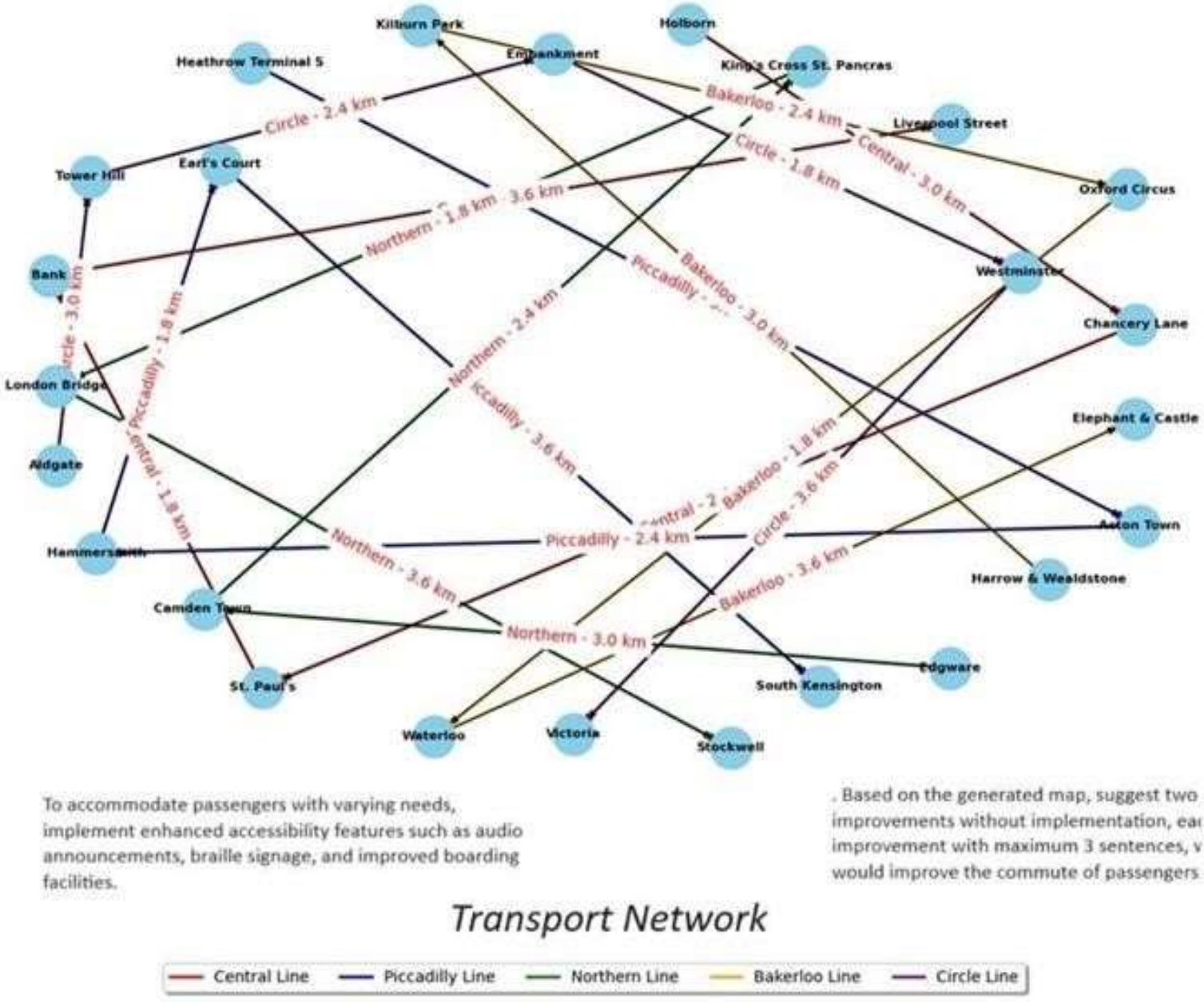
edges = [(u, v) for u, v, d in transport_network.edges(data=True) if d['line'] == line]

nx.draw_networkx_edges(transport_network, pos, edgelist=edges, edge_color=color, width=2.0, ax=ax)

legend_lines.append(Line2D([0], [0], color=color, linewidth=2, label=f'{line} Line'))

```
    nx.draw(transport_network, pos, with_labels=True, font_weight='bold', node_size=700,
node_color='skyblue', font_size=8, ax=ax)     edge_labels = {(edge[0], edge[1]):
f"{edge[2]['line']} - {edge[2]['distance']} km" for edge in transport_network.edges(data=True)}
nx.draw_networkx_edge_labels(transport_network, pos, edge_labels=edge_labels,
font_color='red', ax=ax)     ax.legend(handles=legend_lines, loc='lower center',
bbox_to_anchor=(0.5, -0.2), fancybox=True, shadow=True, ncol=5)     plt.show()

transport_network = create_transport_network()
visualize_transport_network(transport_network)
```

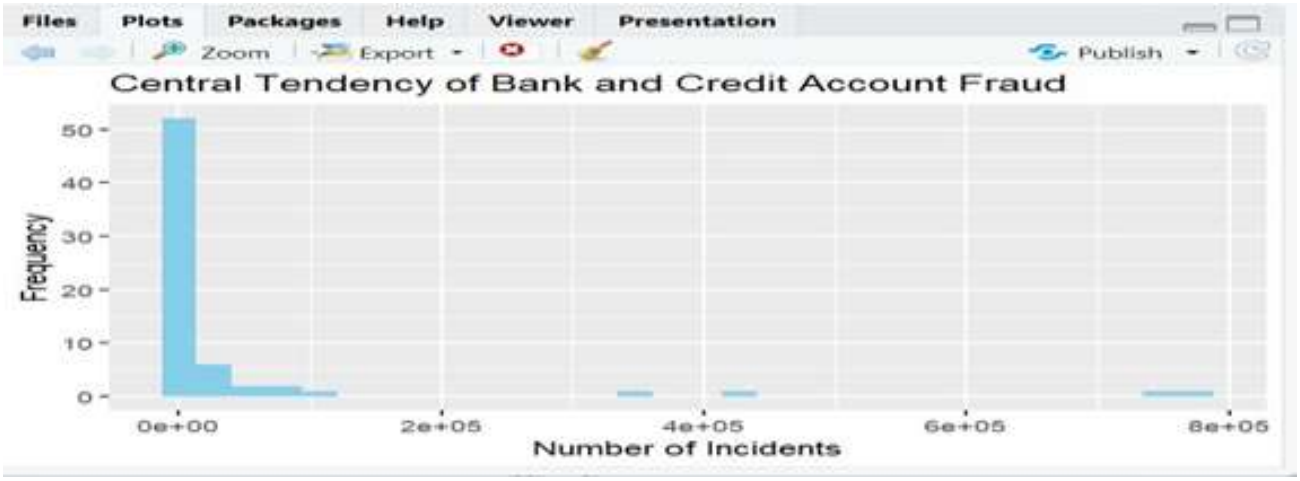


To accommodate passengers with varying needs, implement enhanced accessibility features such as audio announcements, braille signage, and improved boarding facilities.

. Based on the generated map, suggest two improvements without implementation, ea improvement with maximum 3 sentences, v would improve the commute of passengers

Transport Network

Central Line Piccadilly Line Northern Line Bakerloo Line Circle Line




```
        networkx as nx
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D

def create_transport_network():
    transport_network = nx.DiGraph()

    tube_lines = ['Central', 'Piccadilly', 'Northern', 'Bakerloo', 'Circle']
    station_names = {
        'Central': ['Holborn', 'Chancery Lane', 'St. Paul\'s', 'Bank', 'Liverpool Street'],
        'Piccadilly': ['Heathrow Terminal 5', 'Acton Town', 'Hammersmith', 'Earl\'s
Court', 'South Kensington'],
        'Northern': ['Edgware', 'Camden Town', 'King\'s Cross St. Pancras', 'London
Bridge', 'Stockwell'],
        'Bakerloo': ['Harrow & Wealdstone', 'Kilburn Park', 'Oxford Circus',
'Waterloo', 'Elephant & Castle'],
        'Circle': ['Aldgate', 'Tower Hill', 'Embankment', 'Westminster', 'Victoria']
    }

    for line in tube_lines:
        stations = station_names[line]
        transport_network.add_nodes_from(stations)
        distances = [3.0, 2.4, 1.8, 3.6]
        for i in range(len(stations)-1):
            transport_network.add_edge(stations[i], stations[i+1], line=line,
distance=distances[i])
```

```
    return transport_network

def visualize_transport_network(transport_network):
    pos = nx.spring_layout(transport_network, k=2.5, seed=42)
    line_colors = {'Central': 'red', 'Piccadilly': 'blue', 'Northern': 'green', 'Bakerloo':
'orange', 'Circle': 'purple'}
    fig, ax = plt.subplots(figsize=(15, 10))
    legend_lines = []

    for line, color in line_colors.items():
        edges = [(u, v) for u, v, d in transport_network.edges(data=True) if d['line']
== line]
        nx.draw_networkx_edges(transport_network, pos, edgelist=edges,
edge_color=color, width=2.0, ax=ax)
        legend_lines.append(Line2D([0], [0], color=color, linewidth=2, label=f'{line}
Line'))

    nx.draw(transport_network, pos, with_labels=True, font_weight='bold',
node_size=700, node_color='skyblue', font_size=8, ax=ax)
    edge_labels = {(edge[0], edge[1]): f"{edge[2]['line']} - {edge[2]['distance']} km"
for edge in transport_network.edges(data=True)}
    nx.draw_networkx_edge_labels(transport_network, pos,
edge_labels=edge_labels, font_color='red', ax=ax)
    ax.legend(handles=legend_lines, loc='lower center', bbox_to_anchor=(0.5, -
0.2), fancybox=True, shadow=True, ncol=5)
    plt.show()

transport_network = create_transport_network()
visualize_transport_network(transport_network)
```