How Does Urban Environment Shape Crime Rates

April 29, 2025

1 How Does Urban Environment Shape Crime Rates

1.1 Preparation

- https://github.com/YULI61/DSSS_PR/tree/main
- Number of words: 1534
- Runtime: 4 minutes (*Memory 32 GB, Intel(R) Core(TM) i7-14700HX 2.10 GHz*)
- Coding environment: SDS Docker
- License: this notebook is made available under the Creative Commons Attribution license (or other license that you like).

1.2 Table of contents

1.3 Introduction

[go back to the top]

Environmental factors strongly shape urban crime. Governments at all levels treat crime prevention as a key policy goal. Although environmental ecology has become an important area in crime prevention and urban planning research(Lai et al., 2025), scholars still disagree about the specific ways built environment features influence crime(He and Li, 2022). Existing studies show that factors like the permeability of built environments, the mix of land uses, and road connectivity interact in complex ways with community collective efficacy and crime risk(Anderson et al., 2013).

This study aims to use spatial data from London to test the relationship between built environment features and crime rates. We will control for social and demographic factors and apply nonlinear modeling methods. We will explore how different features of the built environment affect property and non-property crimes, and how these effects vary across space.

1.3.1 Requirements to run the analysis

An overview of packages used to run the analysis.

[1]:
```
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn.model_selection import train_test_split, GridSearchCV,␣
↪validation_curve
from sklearn.metrics import root_mean_squared_error
import numpy as np
from tabulate import tabulate

# CART
from sklearn.tree import DecisionTreeRegressor

# random forest
from sklearn.ensemble import RandomForestRegressor

# feature importance
import rfpimp

# xgboost
import xgboost
from xgboost import XGBRegressor

from linearmodels.panel import PanelOLS
import statsmodels.api as sm
```

1.3.2 Literature review

Many studies have explored how the built environment affects crime, but there is no clear answer. For example, residential areas often have lower crime than commercial or mixed-use zones. In addition, crime tends to fall after commercial spaces are turned into housing Anderson et al., 2013 .

New Urbanism promotes compact, walkable, and mixed-use areas to support sustainable cities. But some criminologists question its safety. Some studies show mixed results on how the built environment affects property crime. Also, they often overlook the effects of nearby social disadvantage.(He and Li, 2022). In addition, Butts and others pointed out that there is a "nonlinear relationship" between population density and crime(Butts et al., 2012). This shows that it is important to consider nonlinear effects.

Most current studies use linear methods and do not fully capture the possible nonlinear relationships between environmental factors and crime rates. Given the complex causes of crime, it is necessary to explore nonlinear effects as a complement. Also, existing research mainly focuses on property and violent crimes, while giving less attention to crimes like disorder and drug offenses(He and Li, 2022).

(Groff and Lockwood, 2014) pointed out that property crimes are most common on street segments, but disorder crimes are much less frequent.Therefore, this study will look at both property and non-property crimes. It will use changes in crime rates as the main outcome variable. The goal is to better show how built environment features influence urban crime patterns.

1.4 Research questions

[go back to the top]

This study aims to explore whether crime in London shows spatial regularities and whether these patterns are linked to features of the built environment. It further examines if such relationships are consistent over time or potentially contribute to shifts in local crime risk.

This leads to the following research questions:

RQ1 | Do crime rates vary spatially across London? Are the patterns of property crime and non-property crime distributed in the same way?

RQ2 | If spatial differences exist, are they correlated with built environment features such as greenspace, road density, and transport accessibility?

RQ3 | If such relationships hold, can we identify whether built environment changes—like housing growth—have a measurable temporal effect on crime?

1.5 Data

[go back to the top]

1.5.1 Boundaries Data

Ward-level boundary data were taken from the UK Office for National Statistics GeoPortal. Only London wards were kept. Geometric areas were calculated in square kilometers. Wards in the City of London (LAD code: E09000001) were merged into one unit for consistency. It serves as the base for merging other data.

[2]:
```
# Load ward shapefiles
#ward_boundaties = gpd.read_file("data/ward_boundaties.geojson")
ward_boundaties = gpd.read_file("https://raw.githubusercontent.com/YULI61/
↪DSSS_PR/main/_data/ward_boundaties.geojson")


ward_boundaties = ward_boundaties[ward_boundaties['LAD24CD'].str.
↪startswith('E09')].copy()
ward_boundaties['ward_area'] = ward_boundaties['geometry'].area / 1e6

ward = ward_boundaties[["WD24CD", "WD24NM", "LAD24CD", "geometry", "ward_area"]]
```

```
city_of_london_wards = ward[ward['LAD24CD'] == 'E09000001']
city_of_london_merged = city_of_london_wards.dissolve()
city_of_london_merged['WD24CD'] = 'E09000001'
city_of_london_merged['WD24NM'] = 'City of London'
city_of_london_merged['LAD24CD'] = 'E09000001'
city_of_london_merged['ward_area'] = city_of_london_wards['ward_area'].sum()  #
↪

city_of_london_merged = city_of_london_merged.reset_index(drop=True)

ward = ward[ward['LAD24CD'] != 'E09000001']
ward = pd.concat([ward, city_of_london_merged], ignore_index=True)

ward.head()
```

[2]:

```
      WD24CD                    WD24NM    LAD24CD  \
0  E05009317       Bethnal Green East  E09000030
1  E05009318  Blackwall & Cubitt Town  E09000030
2  E05009319                 Bow East  E09000030
3  E05009320                 Bow West  E09000030
4  E05009321            Bromley North  E09000030

                                            geometry  ward_area
0  POLYGON ((536274.997 182418.503, 536012.696 18…   1.209072
1  POLYGON ((538731.44 178760.081, 538618.804 178…   1.352522
2  POLYGON ((537638.7 184578.685, 537673.048 1844…   1.868249
3  POLYGON ((537375.704 182857.797, 536746.103 18…   1.358472
4  POLYGON ((538302.301 182600.76, 538298.586 182…   0.600058
```

1.5.2 Crime data

Crime data were sourced from the London Datastore, specifically the Metropolitan Police Service ward-level datasets. Two datasets were used:

- **Recent 24-month crime data**: for spatial correlation and ratio analysis.
- **Historical monthly crime data (2001–2022)**: for long-term panel analysis of temporal trends.

In the recent dataset, monthly crime counts were aggregated by ward to calculate total crime volume. The classification follows the FBI Uniform Crime Reporting (UCR) scheme, where property crimes include burglary, larceny-theft, arson, and motor vehicle theft at the census block group level (He and Li, 2022).

Recent 24-month crime data

[3]:

```
# Load data
crime = pd.read_csv("data/MPS Ward Level Crime (most recent 24 months).csv")
```

```python
month_cols = crime.columns[5:]
crime["total_crime_count"] = crime[month_cols].sum(axis=1)

# Define property crime categories
property_major_categories = [
    "ARSON AND CRIMINAL DAMAGE",
    "BURGLARY",
    "ROBBERY",
    "THEFT"
]
property_minor_categories_vehicle = [
    "THEFT FROM A MOTOR VEHICLE",
    "THEFT OR TAKING OF A MOTOR VEHICLE"
]

# Create property crime flag
crime["is_property_crime"] = (
    (crime["MajorText"].isin(property_major_categories)) |
    ((crime["MajorText"] == "VEHICLE OFFENCES") & (crime["MinorText"].
 ↪isin(property_minor_categories_vehicle)))
)

# Group by
crime_summary = crime.groupby('WardCode').agg(
    total_crimes=("total_crime_count", "sum"),
    property_crimes=("is_property_crime", lambda x: (x * crime.loc[x.index,␣
 ↪"total_crime_count"]).sum()),
    WardName=("WardName", "first")
).reset_index()

# Calculate property crime ratio
crime_summary["property_crime_ratio"] = crime_summary["property_crimes"] /␣
 ↪crime_summary["total_crimes"]
crime_summary["non_property_crime_ratio"] = 1 -␣
 ↪crime_summary["property_crime_ratio"]

# Merge crime summary with WARD shapefile
ward = pd.merge(ward, crime_summary,left_on='WD24CD', right_on='WardCode',␣
 ↪how='left')

#Calculate crime density (only where crime exists)
ward['crime_density'] = ward['total_crimes'] / ward['ward_area']
```

Historical monthly crime data (2001–2022)

[4]:
```python
# 1. Load crime data
crime = pd.read_csv("data/MPS Ward Level Crime (Historical).csv")
```

```python
ward_lookup = ward_boundaties[["WD24CD", "LAD24CD"]]  # WD24CD = WardCode,␣
↪LAD24CD = BoroughCode

# 3. Merge crime data with borough code
crime = crime.merge(ward_lookup, how="left", left_on="WardCode",␣
↪right_on="WD24CD")

# 4. Filter needed month columns (e.g., 201004 to 202203)
month_cols = [col for col in crime.columns if col.isdigit()]
month_cols = [col for col in month_cols if 201004 <= int(col) <= 202203]

# 5. Reshape into long format
crime_long = crime.melt(id_vars=["LAD24CD"], value_vars=month_cols,␣
↪var_name="Month", value_name="Crime_Count")

# 6. Convert month into proper datetime
crime_long["Month"] = pd.to_datetime(crime_long["Month"], format='%Y%m')

# 7. Create Fiscal Year
crime_long["Fiscal_Year"] = crime_long["Month"].apply(
    lambda x: f"{x.year}-{x.year+1}" if x.month >= 4 else f"{x.year-1}-{x.year}"
)

# 8. Group by borough and fiscal year
crime_fy = crime_long.groupby(["LAD24CD", "Fiscal_Year"])["Crime_Count"].sum().
↪reset_index()

# 9. Pivot to wide table
crime_fy_wide = crime_fy.pivot(index="LAD24CD", columns="Fiscal_Year",␣
↪values="Crime_Count").fillna(0).reset_index()

crime_fy_wide.head()
```

[4]:

```
Fiscal_Year    LAD24CD  2010-2011  2011-2012  2012-2013  2013-2014  2014-2015  \
0            E09000002      18656      18269      16803      15648      15811
1            E09000003      25135      25729      24956      22461      23091
2            E09000004      13418      11975      12190      11673      12117
3            E09000005      28068      30148      26381      23819      24724
4            E09000006      21457      21355      19861      19117      20181

Fiscal_Year  2015-2016  2016-2017  2017-2018  2018-2019  2019-2020  2020-2021  \
0                17168      17158      18488      18227      19214      17858
1                24779      25378      26497      28698      30345      25902
2                12538      13367      14593      15869      17619      14494
3                25857      27649      29611      30248      29089      27480
4                21068      21199      22463      23396      24534      20833
```

```
Fiscal_Year  2021-2022
0                19166
1                27413
2                15376
3                28315
4                23078
```

1.5.3 Built Environment Data

This section uses spatial data from OS OpenMap to describe the built environment in each London ward. The layers include greenspace areas, roads, buildings, bus stops, and railway stations. Each feature was matched to a ward using a spatial join.

Four density indicators were created:

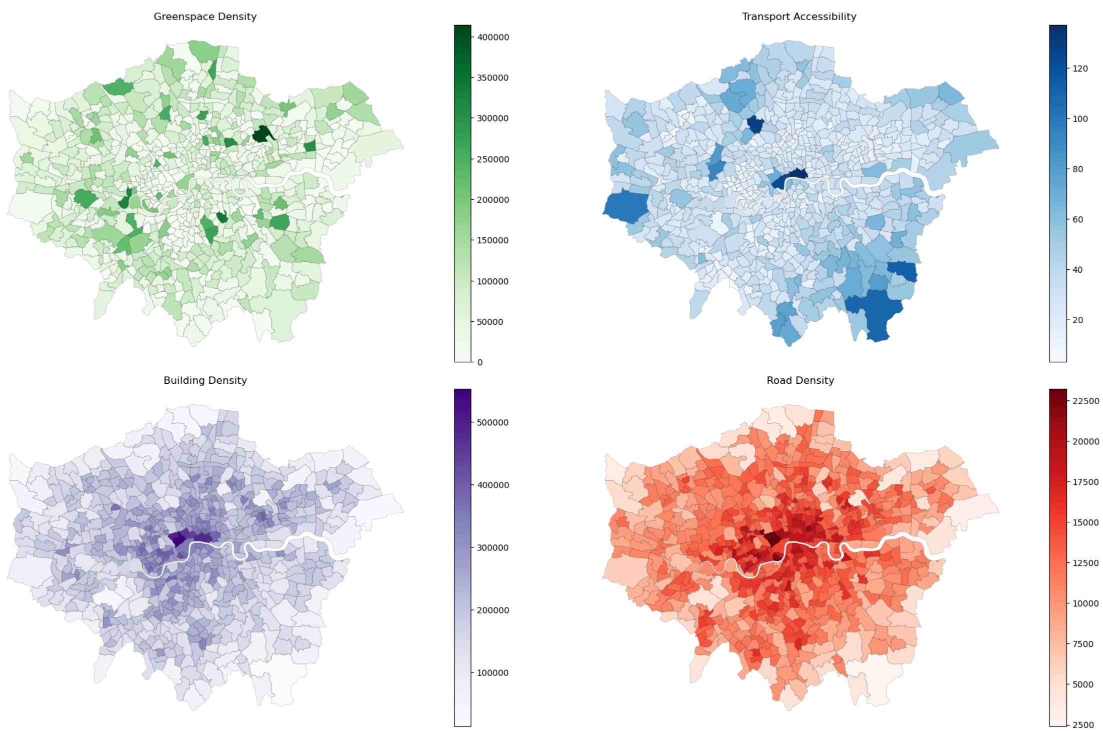
- **Building density**: total building area divided by ward area
- **Road density**: total road length divided by ward area
- **Greenspace density**: total green area divided by ward area
- **Public transport density**: number of bus stops and stations divided by ward area

[5]:
```python
# Load built environment datasets
greenspace = gpd.read_file("data/BE_simplified/greenspace_simplified.geojson")
road = gpd.read_file("data/BE_simplified/road_simplified.geojson")
building = gpd.read_file("data/BE_simplified/building_simplified.geojson")

bus = pd.read_csv("data/Bus_Stops.csv")
bus = gpd.GeoDataFrame(
    bus,
    geometry=gpd.points_from_xy(bus['X'], bus['Y']),
    crs="EPSG:27700"
)

RailwayStation = pd.read_csv("data/Overground_Stations.csv")
RailwayStation = gpd.GeoDataFrame(
    RailwayStation,
    geometry=gpd.points_from_xy(RailwayStation['X'], RailwayStation['Y']),
    crs="EPSG:27700"
)

# Reproject all to LSOA CRS first
greenspace = greenspace.to_crs(ward.crs)
RailwayStation = RailwayStation.to_crs(ward.crs)
road = road.to_crs(ward.crs)
building = building.to_crs(ward.crs)
bus = bus.to_crs(ward.crs)
```

[6]:
```python
# Spatial join
building_with_ward = gpd.sjoin(building, ward, how="inner", predicate="within")
road_with_ward = gpd.sjoin(road, ward, how="inner", predicate="within")
greenspace_with_ward = gpd.sjoin(greenspace, ward, how="inner",␣
↪predicate="within")
railway_with_ward = gpd.sjoin(RailwayStation, ward, how="inner",␣
↪predicate="within")
bus_with_ward = gpd.sjoin(bus, ward, how="inner", predicate="within")

# Groupby and aggregate
building_area = building_with_ward.groupby("WD24CD")["geometry"].apply(lambda x:
↪ x.area.sum())
road_length = road_with_ward.groupby("WD24CD")["geometry"].apply(lambda x: x.
↪length.sum())
greenspace_area = greenspace_with_ward.groupby("WD24CD")["geometry"].
↪apply(lambda x: x.area.sum())
railway_counts = railway_with_ward.groupby("WD24CD").size()
bus_counts = bus_with_ward.groupby("WD24CD").size()

# Merge back
ward["lsoa_area"] = ward.geometry.area
ward["building_area"] = ward["WD24CD"].map(building_area)
ward["road_length"] = ward["WD24CD"].map(road_length)
ward["greenspace_area"] = ward["WD24CD"].map(greenspace_area)
ward['bus_counts'] = ward['WD24CD'].map(bus_counts).fillna(0)
ward['railway_counts'] = ward['WD24CD'].map(railway_counts).fillna(0)
ward["public_transport_counts"] = ward["bus_counts"] + ward["railway_counts"]

ward = ward.fillna(0)

# Calculate density variables
ward["building_density"] = ward["building_area"] / ward["ward_area"]
ward["road_density"] = ward["road_length"] / ward["ward_area"]
ward["greenspace_density"] = ward["greenspace_area"] / ward["ward_area"]
ward["public_transport_density"] = ward["public_transport_counts"] /␣
↪ward["ward_area"]
```

[7]:
```python
fig, axes = plt.subplots(2, 2, figsize=(20, 12))

# Greenspace Density
ward.plot(column="greenspace_density", cmap="Greens", ax=axes[0,0], legend=True)
ward.boundary.plot(ax=axes[0,0], color="black", linewidth=0.1)
axes[0,0].set_title("Greenspace Density")

# Transport Accessibility
ward.plot(column="public_transport_counts", cmap="Blues", ax=axes[0,1],␣
↪legend=True)
```

```
ward.boundary.plot(ax=axes[0,1], color="black", linewidth=0.1)
axes[0,1].set_title("Transport Accessibility ")

# Building Density
ward.plot(column="building_density", cmap="Purples", ax=axes[1,0], legend=True)
ward.boundary.plot(ax=axes[1,0], color="black", linewidth=0.1)
axes[1,0].set_title("Building Density")

# Road Density
ward.plot(column="road_density", cmap="Reds", ax=axes[1,1], legend=True)
ward.boundary.plot(ax=axes[1,1], color="black", linewidth=0.1)
axes[1,1].set_title("Road Density")

# Turn off axis lines
for ax in axes.flat:
    ax.axis('off')

plt.tight_layout()
plt.show()
```



These maps reveal clear spatial disparities in built environment features across London.

Building and road densities are highest in central areas, which also tend to have higher crime rates. In contrast, greenspace is more common in outer wards, where crime is generally lower. Public transport access shows mixed patterns, with high densities both in central boroughs and specific peripheral zones.

These differences suggest that built environment factors may shape crime distribution by influencing accessibility, surveillance, and urban activity intensity—key aspects relevant to crime opportunity and deterrence.

1.5.4 Social Environment Data

Social variables were collected from multiple sources on the London Datastore, including census and deprivation data. The key indicators include:

- **Male ratio**: share of male residents

- **Working-age ratio**: residents aged 16–64, follows the age band definition used in the Index of Multiple Deprivation (IMD)

- **Population density**: population divided by ward area

- **Ethnicity ratios**: share of residents in major groups (White, Black, Asian, Mixed)

- **Average deprivation**: weighted index across four household deprivation dimensions

Prior research highlights that property crime is often linked to community-level characteristics such as income level, deprivation, and ethnic composition (He and Li, 2022). Controlling for these factors allows for a clearer assessment of how the built environment shapes crime independently.

[8]:
```
deprivation = pd.read_csv("data/Household deprivation.csv")
age = pd.read_csv("data/Five year age bands.csv")
usual_residents = pd.read_csv("data/2021_usual_residents.csv")
ethnic = pd.read_csv("data/2021_ethnic_group.csv")

#Merge age data
work_age_cols = age.columns[8:17]
age['working_age_population'] = age[work_age_cols].sum(axis=1)
age['working_age_ratio'] = age['working_age_population'] / age["All usual␣
↪residents"]
age = age[['ward code', 'working_age_ratio']]

ward = pd.merge(ward, age, left_on='WD24CD', right_on='ward code', how='left')
ward = ward.drop(columns='ward code')

#Merge gender data
usual_residents["male_ratio"] = usual_residents["Males"] /␣
↪(usual_residents["All usual residents"])
usual_residents["population"] = usual_residents["All usual residents"]

ward = ward.merge(
    usual_residents[["ward code", "population", "male_ratio"]],
    how="left", left_on="WD24CD", right_on="ward code"
)
```

```
ward["population_density"] = ward["population"] / (ward["ward_area"])
ward = ward.drop(columns=["ward code"])

#Merge ethnicity data
ethnic["white_ratio"] = ethnic.iloc[:, 5:10].sum(axis=1) / ethnic.iloc[:, 4].
↪astype(float)
ethnic["mixed_ratio"] = ethnic.iloc[:, 10:14].sum(axis=1) / ethnic.iloc[:, 4].
↪astype(float)
ethnic["asian_ratio"] = ethnic.iloc[:, 14:19].sum(axis=1) / ethnic.iloc[:, 4].
↪astype(float)
ethnic["black_ratio"] = ethnic.iloc[:, 19:23].sum(axis=1) / ethnic.iloc[:, 4].
↪astype(float)

# Merge ethnicity ratios into ward
ward = ward.merge(
    ethnic[["ward code", "white_ratio", "mixed_ratio", "asian_ratio",␣
↪"black_ratio"]],
    how="left",
    left_on="WD24CD",
    right_on="ward code"
)

ward = ward.drop(columns="ward code")

# Fill missing values if any
ward = ward.fillna(0)

#Merge deprivation data
deprivation['average_deprivation'] = (
    (1 * deprivation['1 dimension']) +
    (2 * deprivation['2 dimensions']) +
    (3 * deprivation['3 dimensions']) +
    (4 * deprivation['4 dimensions'])
) / deprivation['All Households']

ward = ward.merge(
    deprivation[["ward code", "average_deprivation"]],
    how="left", left_on="WD24CD", right_on="ward code"
)

ward = ward.drop(columns='ward code')
```

1.5.5 Housing Data

Housing data were obtained from the Greater London Authority's dataset on net additional dwellings. It reports the number of new homes completed each year in every London borough from 2001 to 2022. This variable is used to explore whether changes in housing supply relate to trends in crime levels.

[9]:
```
# Load data and skip 1st and 3rd rows
housing = pd.read_csv("data/net-additional-dwellings-total-stock-borough.csv",␣
↪skiprows=[0, 2])

# Rename LAD24CD
housing = housing.rename(columns={housing.columns[0]: "LAD24CD"})

# Keep only LAD24CD + years columns
cols_to_keep = ["LAD24CD"] + [col for col in housing.columns if "-" in col or "/
↪" in col]
housing = housing[cols_to_keep]

# Fix fiscal year column names
new_columns = []
for col in housing.columns:
    if "-" in col or "/" in col:
        col = col.replace("/", "-")  # replace "/" with "-"
        start_year = col.split("-")[0]
        end_year = col.split("-")[1].zfill(2)
        full_year = f"{start_year}-20{end_year}"
        new_columns.append(full_year)
    else:
        new_columns.append(col)
housing.columns = new_columns

# Check
housing.head()
```

[9]:
```
     LAD24CD 2001-2002 2002-2003 2003-2004 2004-2005 2005-2006 2006-2007  \
0  E09000001       108         9       137       111        -1         1
1  E09000002       228       312        62       362       268       307
2  E09000003       286     1,083       973     1,047     1,039       488
3  E09000004       302       316       727       271       213       313
4  E09000005     1,401     1,018       608       639     1,674     1,202

  2007-2008 2008-2009 2009-2010  … 2012-2013 2013-2014 2014-2015 2015-2016  \
0        46        46        -8  …        35       437       226        77
1       716       288       108  …       506       731       514       732
2     1,252     1,089       848  …     1,374     1,113     1,324     1,458
3       333       293       428  …       418       528       810      -132
4     1,067     1,207     1,170  …       662       734     1,559     1,051

  2016-2017 2017-2018 2018-2019 2019-2020 2020-2021 2021-2022
0         7       138       351       297       206       432
1       596       413       906       741     1,048     1,206
```

```
2     1,799     2,208     2,209     1,990     2,250       206
3       764       277       486       208       632       710
4     1,364       694     1,741     2,433     2,404     3,574

[5 rows x 22 columns]
```

1.5.6 Summary of Correlation Variables

[10]:
```python
crime_value = ward[['crime_density','property_crime_ratio','building_density',
       'road_density', 'greenspace_density', 'public_transport_density',
       'working_age_ratio', 'male_ratio', 'population_density',
       'white_ratio', 'mixed_ratio', 'asian_ratio', 'black_ratio',
       'average_deprivation']]

independent_vars = [
    'building_density', 'road_density', 'greenspace_density',␣
 ↪'public_transport_density',
    'working_age_ratio', 'male_ratio', 'population_density',
    'white_ratio', 'mixed_ratio', 'asian_ratio', 'black_ratio',
    'average_deprivation','property_crime_ratio'
]
dependent_vars = ['crime_density']

summary_stats = crime_value.describe().T[['mean', '50%', 'min', 'max', 'std']]
summary_stats = summary_stats.rename(columns={'50%': 'median'})
summary_stats = summary_stats.round(2)
summary_stats.index.name = 'Variable'
summary_stats = summary_stats.reset_index()

summary_stats['Type'] = summary_stats['Variable'].apply(
    lambda x: 'Dependent' if x in dependent_vars else 'Independent'
)

summary_stats['Type_order'] = summary_stats['Type'].map({'Independent': 0,␣
 ↪'Dependent': 1})
summary_stats = summary_stats.sort_values(by=['Type_order', 'Variable']).
 ↪drop(columns='Type_order')


summary_stats = summary_stats[['Type', 'Variable', 'mean', 'median', 'min',␣
 ↪'max', 'std']]
print(tabulate(summary_stats, headers='keys', tablefmt='github',␣
 ↪showindex=False))
```

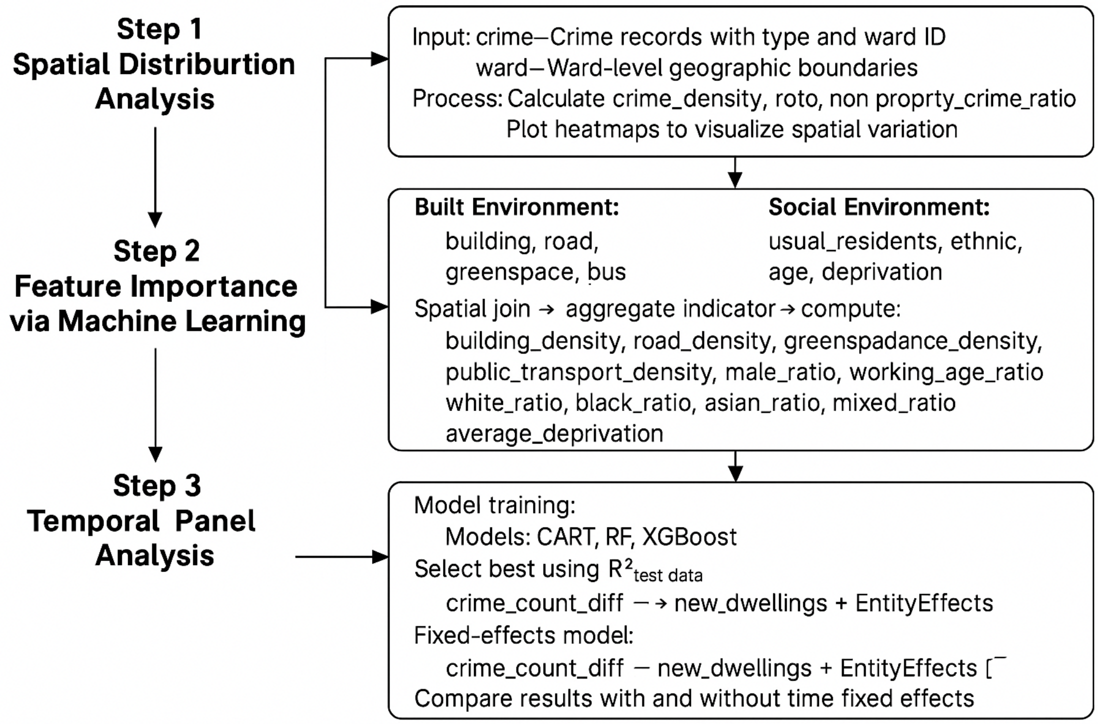
```
| Type        | Variable                  |      mean |    median |      min |
max |      std |
|-------------|---------------------------|-----------|-----------|----------|---
```

Independent	asian_ratio	0.2	0.15	0.02	0.8	0.15
Independent	average_deprivation	0.76	0.75	0.38	1.29	0.17
Independent	black_ratio	0.15	0.12	0.02	0.5	0.09
Independent	building_density	204815	197981	13573.4	552319	78454
Independent	greenspace_density	63730	48764.7	0	414253	58066.4
Independent	male_ratio	0.48	0.48	0.44	0.55	0.01
Independent	mixed_ratio	0.06	0.06	0.01	0.1	0.02
Independent	population_density	8700.4	7748.33	163.61	26189.8	4855.42
Independent	property_crime_ratio	0.42	0.41	0	0.85	0.1
Independent	public_transport_density	17.26	15.9	1.67	73.66	8.69
Independent	road_density	12350.5	12291.1	2410.91	23223.9	3557.75
Independent	white_ratio	0.55	0.57	0.05	0.93	0.17
Independent	working_age_ratio	0.65	0.65	0.52	0.85	0.06
Dependent	crime_density	1931.93	1244.13	0	32771.6	2234.26

1.6 Methodology

[go back to the top]

This study uses a three-step approach to understand how built and social environments affect crime in London:



1.6.1 Spatial Pattern Analysis

We first mapped crime ratios across wards using choropleth maps. This helped show where property and non-property crimes are more common and revealed possible spatial clusters.

1.6.2 Crime Correlation Modeling

Training CART, Random Forest, and XGBoost models to predict crime density using multiple variables. The model with the highest R^2 was chosen, and we used permutation importance to find the most important features.

[11]:
```
random_state_split = 100
train_x, test_x, train_y, test_y = train_test_split(crime_value.
 ↪drop(['crime_density'], axis = 1),
                                                    crime_value.crime_density,␣
 ↪random_state=random_state_split)

print(train_x.shape)
print(train_y.shape)
print(test_x.shape)
print(test_y.shape)

# check the index of train_x and train_y - they should be identical. The index␣
 ↪indicates which rows from the original data.

print(train_x.index.identical(train_y.index))
```

```
print(test_x.index.identical(test_y.index))
```

```
(510, 13)
(510,)
(170, 13)
(170,)
True
True
```

[12]:
```
# Train a default CART model
cart_default = DecisionTreeRegressor(random_state=0)
cart_default.fit(train_x, train_y)

# Get the default tree depth
default_depth = cart_default.get_depth()
print(f"Default tree depth is: {default_depth}")
```

```
Default tree depth is: 18
```

[13]:
```
# Define hyperparameter grid based on the observed depth
param_grid = {
    'max_depth': [5,10,15,20,25,30,35],
    'min_samples_split': [6,8,10,12,14]
}

# Perform Grid Search
randomState_dt = 10000
dt = DecisionTreeRegressor(random_state=randomState_dt)
clf = GridSearchCV(dt, param_grid, cv=5)
clf.fit(train_x, train_y)

# Print best parameters
print("Best parameters:", clf.best_params_)
print("Best score:", clf.best_score_)
```

```
Best parameters: {'max_depth': 10, 'min_samples_split': 10}
Best score: 0.5913885412148272
```

```
/opt/conda/lib/python3.11/site-packages/numpy/ma/core.py:2820: RuntimeWarning:
invalid value encountered in cast
  _data = np.array(data, dtype=dtype, copy=copy,
```

[14]:
```
dt_final = DecisionTreeRegressor(max_depth=clf.best_params_['max_depth'],␣
 ↪min_samples_split=clf.best_params_['min_samples_split'],␣
 ↪random_state=randomState_dt)
dt_final.fit(train_x, train_y)
```

[14]:
```
DecisionTreeRegressor(max_depth=10, min_samples_split=10, random_state=10000)
```

[15]:
```
# values of max_depth and min_samples_split
hyperparameters = {'max_depth': [5,10,15,20,25,30,35],
    'min_samples_split': [4,6,8,10,12,14]
}

randomState_dt = 10000
rf = RandomForestRegressor(random_state=randomState_dt)

# cv=5 by default, which means 5-fold cross-validation
clf = GridSearchCV(rf, hyperparameters)

clf.fit(train_x, train_y)

# we can query the best parameter value and its accuracy score
print ("The best parameter value is: ")
print (clf.best_params_)
print ("The best score is: ")
print (clf.best_score_)
```

```
/opt/conda/lib/python3.11/site-packages/numpy/ma/core.py:2820: RuntimeWarning:
invalid value encountered in cast
  _data = np.array(data, dtype=dtype, copy=copy,
```

```
The best parameter value is:
{'max_depth': 15, 'min_samples_split': 6}
The best score is:
0.7771690607576976
```

[16]:
```
rf_final = RandomForestRegressor(max_depth=clf.best_params_['max_depth'],␣
 ↪min_samples_split=clf.best_params_['min_samples_split'],␣
 ↪random_state=randomState_dt)
rf_final.fit(train_x, train_y)
```

[16]:
```
RandomForestRegressor(max_depth=15, min_samples_split=6, random_state=10000)
```

[17]:
```
# values of max_depth and min_samples_split
hyperparameters = {'max_depth':[10,20,30,40,50], 'n_estimators':
 ↪[50,100,150,200,250]}

randomState_xgb = 125
xgb = XGBRegressor(random_state=randomState_xgb)

# cv=5 by default, which means 5-fold cross-validation
gscv_xgb = GridSearchCV(xgb, hyperparameters)

gscv_xgb.fit(train_x, train_y)

# we can query the best parameter value and its accuracy score
```

```
print ("The best parameter value is: ")
print (gscv_xgb.best_params_)
print ("The best score is: ")
print (gscv_xgb.best_score_)
```

```
The best parameter value is:
{'max_depth': 20, 'n_estimators': 250}
The best score is:
0.7477040574311402
```

[18]:
```
xgb_final = XGBRegressor(max_depth=gscv_xgb.best_params_['max_depth'],␣
↪n_estimators=gscv_xgb.best_params_['n_estimators'],␣
↪random_state=randomState_xgb)
xgb_final.fit(train_x, train_y)
```

[18]:
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=20, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=250, n_jobs=None,
             num_parallel_tree=None, random_state=125, …)
```

1.6.3 Temporal Effect Estimation

We then used a panel fixed effects model to test if yearly housing growth changes crime over time. By comparing models with and without time effects, we checked whether the results were driven by shared yearly events.

[19]:
```
# 1. Crime reshape
crime_long = crime_fy_wide.melt(
    id_vars="LAD24CD",
    var_name="fiscal_year",
    value_name="crime_count"
)

# 2. Housing reshape
housing_long = housing.melt(
    id_vars="LAD24CD",
    var_name="fiscal_year",
    value_name="new_dwellings"
)

# 3. Merge
```

```
merged = crime_long.merge(housing_long, on=["LAD24CD", "fiscal_year"],␣
↪how="left")
merged = merged.dropna()
```

[20]:
```
merged['new_dwellings'] = merged['new_dwellings'].str.replace(',', '').
↪astype(float)
merged['crime_count'] = merged['crime_count'].astype(float)

merged['crime_count_diff'] = merged.groupby('LAD24CD')['crime_count'].diff()
merged = merged.dropna(subset=['crime_count_diff', 'new_dwellings'])
```

1.7 Results and discussion

[go back to the top]

1.7.1 Spatial Distribution of Crime

The maps show clear differences in crime across London. Crime density is highest in the center, showing a strong center-to-edge pattern. The City of London appears blank, as its data was excluded.

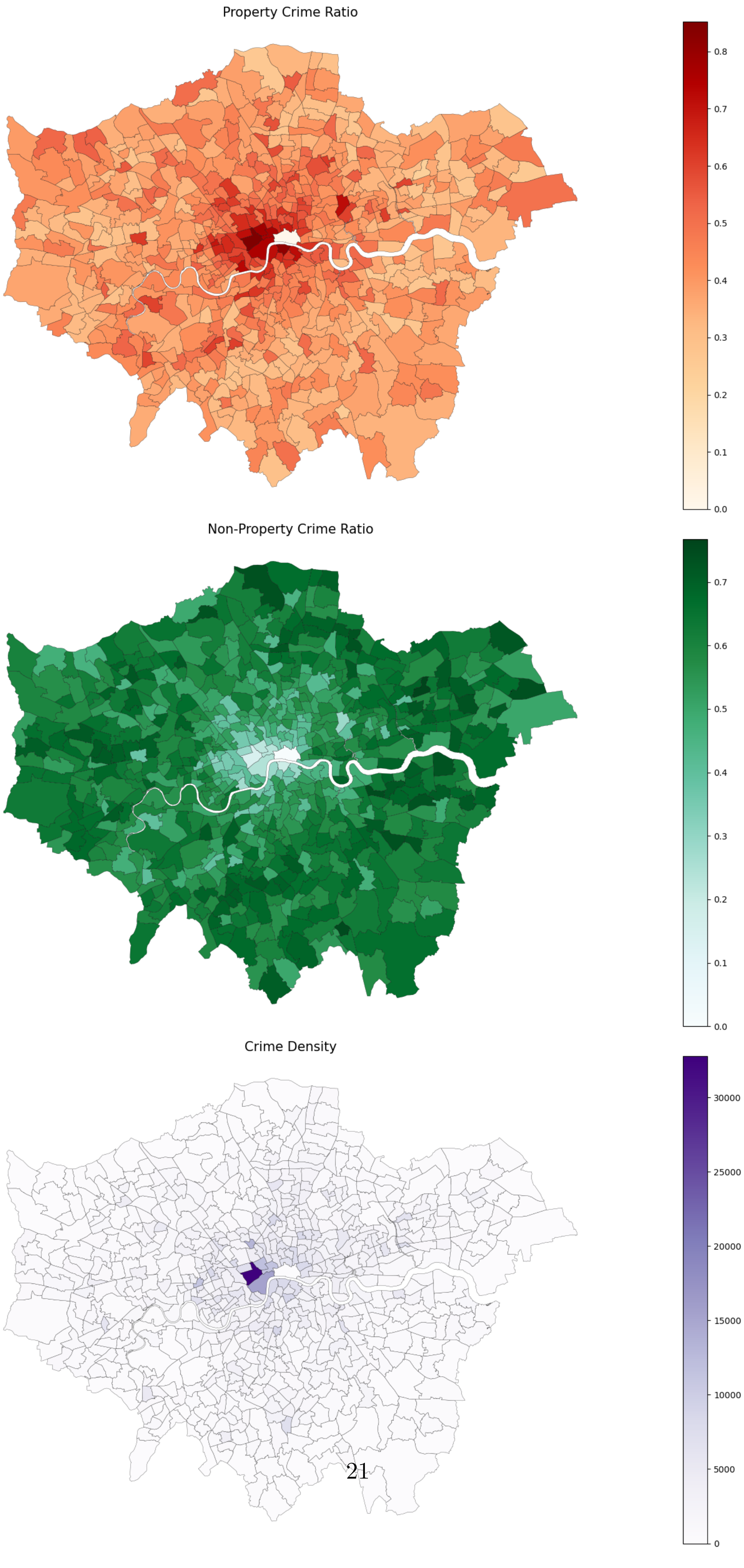
Property crime is most common in central areas like Westminster, where busy streets and shops may increase theft. This supports the idea that crime rises where more people gather.

In contrast, non-property crime is more common in outer or mixed residential areas like Newham, Southwark, and Croydon. This suggests that different types of crime follow different spatial patterns.

Overall, these patterns may relate to land use or local conditions.

[21]:
```
# Set up 1 row, 3 columns of subplots
fig, axes = plt.subplots(3, 1, figsize=(25, 25))

# Plot property crime ratio
ward.plot(
    column="property_crime_ratio",
    cmap="OrRd",
    legend=True,
    ax=axes[0],
    missing_kwds={"color": "lightgrey"},
    edgecolor="black",
    linewidth=0.2
)
axes[0].set_title("Property Crime Ratio", fontsize=15)
axes[0].axis("off")

# Plot non-property crime ratio
ward.plot(
    column="non_property_crime_ratio",
```

```
    cmap="BuGn",
    legend=True,
    ax=axes[1],
    missing_kwds={"color": "lightgrey"},
    edgecolor="black",
    linewidth=0.2
)
axes[1].set_title("Non-Property Crime Ratio", fontsize=15)
axes[1].axis("off")

# Plot crime density
ward.plot(
    column="crime_density",
    cmap="Purples",
    legend=True,
    ax=axes[2],
    missing_kwds={"color": "lightgrey"},
    edgecolor="black",
    linewidth=0.2
)
axes[2].set_title("Crime Density", fontsize=15)
axes[2].axis("off")

plt.tight_layout()
plt.show()
```


Property Crime Ratio
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Non-Property Crime Ratio
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Crime Density
0
5000
10000
15000
20000
25000
30000

1.7.2 Association Between Built Environment and Crime

The results show that the built environment explains much of the difference in crime across London. Among all models, XGBoost had the highest accuracy, showing that the layout of an area is closely linked to its crime level.In particular, places with high building and population density had more crime. This supports the idea that dense areas may create more crime opportunities.

On the other hand, social and demographic factors had little effect once the built environment was considered. This means that how a city is built may matter more than who lives there.

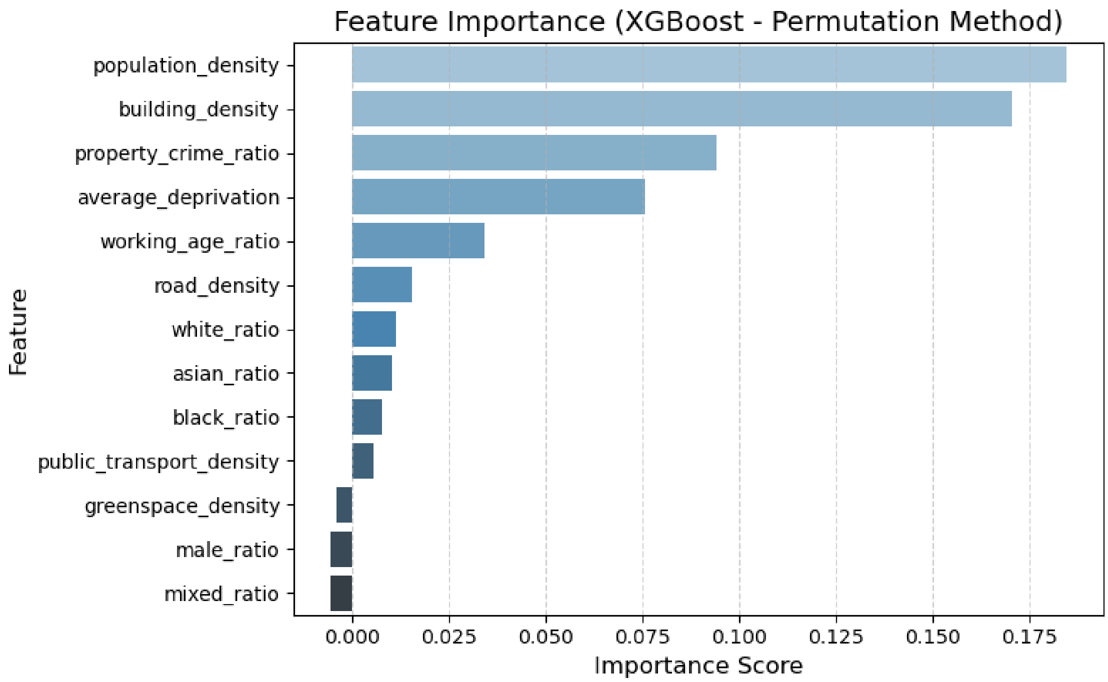
[22]:
```python
# CART
r2_train_cart = dt_final.score(train_x, train_y)
r2_test_cart = dt_final.score(test_x, test_y)
r2_diff_cart = r2_train_cart - r2_test_cart

# RF
r2_train_rf = rf_final.score(train_x, train_y)
r2_test_rf = rf_final.score(test_x, test_y)
r2_diff_rf = r2_train_rf - r2_test_rf

# XGB
r2_train_xgb = xgb_final.score(train_x, train_y)
r2_test_xgb = xgb_final.score(test_x, test_y)
r2_diff_xgb = r2_train_xgb - r2_test_xgb

results = pd.DataFrame({
    'R2_train_data': [r2_train_cart, r2_train_rf, r2_train_xgb],
    'R2_test_data': [r2_test_cart, r2_test_rf, r2_test_xgb],
    'R2_diff': [r2_diff_cart, r2_diff_rf, r2_diff_xgb]
}, index=['CART', 'RF', 'XGBoost'])

display(results)
```

	R2_train_data	R2_test_data	R2_diff
CART	0.952051	0.522458	0.429593
RF	0.942956	0.766560	0.176396
XGBoost	1.000000	0.772070	0.227930

[23]:
```python
imp = rfpimp.importances(xgb_final, test_x, test_y) # permutation
print(imp)

# Sort importance values in descending order and prepare data
imp_sorted = imp.sort_values(by='Importance', ascending=False).copy()
imp_sorted['Feature'] = imp_sorted.index  # Add 'Feature' column for use with␣
↪hue
```

```python
# Plot feature importances using seaborn
plt.figure(figsize=(8, 5))
sns.barplot(
    x='Importance',
    y='Feature',
    hue='Feature',            # Assign hue to avoid future warnings
    data=imp_sorted,
    dodge=False,
    palette='Blues_d',        # Use a blue gradient color palette
    legend=False              # Disable redundant legend
)

# Add labels and formatting
plt.title('Feature Importance (XGBoost - Permutation Method)', fontsize=14)
plt.xlabel('Importance Score', fontsize=12)
plt.ylabel('Feature', fontsize=12)
plt.grid(True, axis='x', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```

```
                          Importance
Feature
population_density          0.184451
building_density            0.170331
property_crime_ratio        0.093930
average_deprivation         0.075874
working_age_ratio           0.034163
road_density                0.015661
white_ratio                 0.011535
asian_ratio                 0.010370
black_ratio                 0.008010
public_transport_density    0.005790
greenspace_density         -0.003882
male_ratio                 -0.005246
mixed_ratio                -0.005279
```



1.7.3 Temporal Dynamics of Housing Growth and Crime

In 2020, crime in London dropped sharply due to the pandemic. Westminster alone accounted for nearly one-third of this decline. This shows that sudden events can disrupt long-term crime trends, so time-related shocks should be considered in analysis.

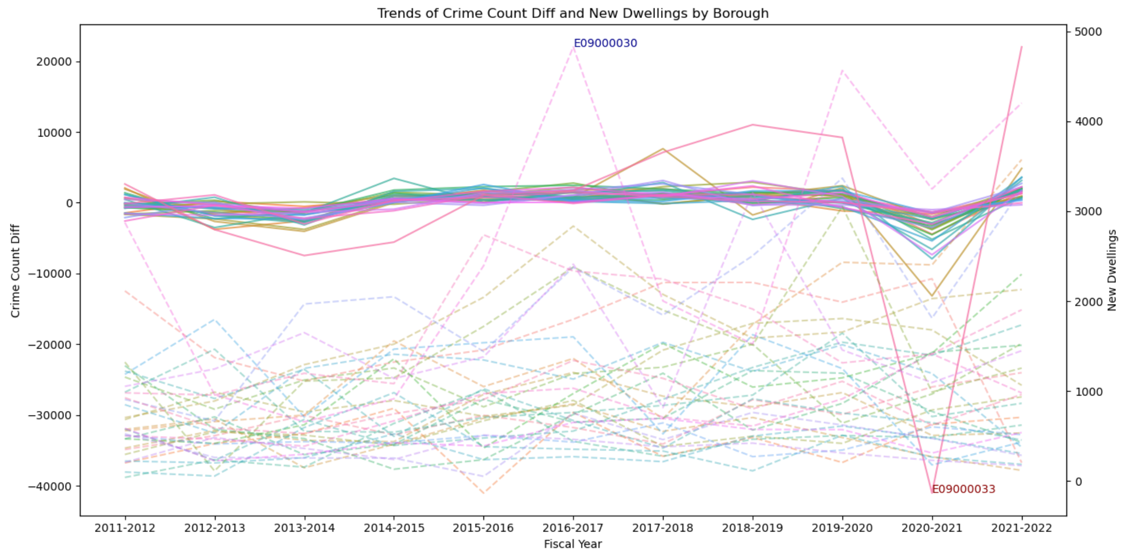
Earlier findings showed that building density is the strongest predictor of crime. This raises a key planning question: if higher density is linked to more crime, should new housing projects consider safety impacts?

To test this, we used a panel fixed effects model to study the link between yearly housing growth and crime. Before controlling for time, housing growth showed a weak but significant link to crime. After adding time fixed effects, this link disappeared.

This suggests that housing growth does not directly cause crime changes. Instead, stable features like building density have a stronger and more consistent impact.

[24]:
```
palette = sns.color_palette("husl", n_colors=len(merged['LAD24CD'].unique()))

fig, ax1 = plt.subplots(figsize=(14, 7))

# crime_count_diff
for i, borough in enumerate(merged['LAD24CD'].unique()):
    data = merged[merged['LAD24CD'] == borough].sort_values('fiscal_year')
    ax1.plot(data['fiscal_year'], data['crime_count_diff'], label=borough,␣
 ↪color=palette[i], alpha=0.7)
```

```
ax1.set_ylabel('Crime Count Diff', color='black')
ax1.tick_params(axis='y', labelcolor='black')
ax1.set_xlabel('Fiscal Year')

# new_dwellings
ax2 = ax1.twinx()
for i, borough in enumerate(merged['LAD24CD'].unique()):
    data = merged[merged['LAD24CD'] == borough].sort_values('fiscal_year')
    ax2.plot(data['fiscal_year'], data['new_dwellings'], linestyle='--',␣
↪color=palette[i], alpha=0.4)

ax2.set_ylabel('New Dwellings', color='black')
ax2.tick_params(axis='y', labelcolor='black')


min_crime_diff_row = merged.loc[merged['crime_count_diff'].idxmin()]
max_dwellings_row = merged.loc[merged['new_dwellings'].idxmax()]

borough_min_crime = min_crime_diff_row['LAD24CD']
borough_max_dwell = max_dwellings_row['LAD24CD']

ax1.text(min_crime_diff_row['fiscal_year'],␣
↪min_crime_diff_row['crime_count_diff'],
         f"{borough_min_crime}", color='darkred', fontsize=10)

ax2.text(max_dwellings_row['fiscal_year'], max_dwellings_row['new_dwellings'],
         f"{borough_max_dwell}", color='darkblue', fontsize=10)


plt.title('Trends of Crime Count Diff and New Dwellings by Borough')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[25]:
```
merged = merged.copy()
merged['fiscal_year'] = merged['fiscal_year'].astype(str)
merged['fiscal_year_num'] = merged['fiscal_year'].str[:4].astype(int)
merged = merged.set_index(['LAD24CD', 'fiscal_year_num'])
```

[26]:
```
# Run Fixed Effects Regression using formula
model = PanelOLS.from_formula('crime_count_diff ~ 1 + new_dwellings +␣
↪EntityEffects', data=merged).fit()
print(model)
```

```
                          PanelOLS Estimation Summary
================================================================================
Dep. Variable:       crime_count_diff   R-squared:                        0.0143
Estimator:                   PanelOLS   R-squared (Between):             -2.6870
No. Observations:                 352   R-squared (Within):               0.0143
Date:                Tue, Apr 29 2025   R-squared (Overall):              0.0001
Time:                        22:35:34   Log-likelihood                   -3348.5
Cov. Estimator:            Unadjusted
                                        F-statistic:                      4.6321
Entities:                          32   P-value                           0.0321
Avg Obs:                       11.000   Distribution:                   F(1,319)
Min Obs:                       11.000
Max Obs:                       11.000   F-statistic (robust):             4.6321
                                        P-value                           0.0321
Time periods:                      11   Distribution:                   F(1,319)
Avg Obs:                       32.000
Min Obs:                       32.000
Max Obs:                       32.000
```

```
                        Parameter Estimates
================================================================================
=
                Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper
CI
--------------------------------------------------------------------------------
-
Intercept         -737.44     403.97    -1.8255     0.0689     -1532.2
57.345
new_dwellings      0.7916     0.3678     2.1522     0.0321      0.0680
1.5153
================================================================================
=

F-test for Poolability: 0.1231
P-value: 1.0000
Distribution: F(31,319)

Included effects: Entity
```

[27]:

```python
model = PanelOLS.from_formula(
    'crime_count_diff ~ 1 + new_dwellings + EntityEffects + TimeEffects',
    data=merged
).fit()
print(model)
```

```
                          PanelOLS Estimation Summary
================================================================================
Dep. Variable:       crime_count_diff   R-squared:                        0.0017
Estimator:                   PanelOLS   R-squared (Between):             -0.1529
No. Observations:                 352   R-squared (Within):               0.0074
Date:                Tue, Apr 29 2025   R-squared (Overall):              0.0066
Time:                        22:35:34   Log-likelihood                   -3279.6
Cov. Estimator:            Unadjusted
                                        F-statistic:                      0.5247
Entities:                          32   P-value                           0.4694
Avg Obs:                       11.000   Distribution:                   F(1,309)
Min Obs:                       11.000
Max Obs:                       11.000   F-statistic (robust):             0.5247
                                        P-value                           0.4694
Time periods:                      11   Distribution:                   F(1,309)
Avg Obs:                       32.000
Min Obs:                       32.000
Max Obs:                       32.000

                        Parameter Estimates
================================================================================
=
```

```
                Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper
CI
--------------------------------------------------------------------------------
-
Intercept         -200.59     362.38    -0.5535     0.5803     -913.65
512.46
new_dwellings      0.2431     0.3356     0.7244     0.4694     -0.4172
0.9034
================================================================================
=

F-test for Poolability: 3.7455
P-value: 0.0000
Distribution: F(41,309)

Included effects: Entity, Time
```

1.8 Conclusion

[go back to the top]

This study shows that crime in London varies widely across space and is closely tied to features of the built environment. Building density is the most consistent factor linked to crime, while short-term housing growth has little clear effect over time.

Crime is not just a social issue—it is also shaped by space. Features like density, land-use mix, and how people move through the city affect where crime happens. Planners should focus less on how many homes are built each year and more on how city spaces are designed. Thinking about crime prevention in urban design could help build safer, more sustainable cities.

1.8.1 Limitations

- **Transport effects are mixed** Station closures often come with other changes, so their impact on crime is hard to separate.(Phillips and Sandler, 2015).

- **Underreporting of crime** Not all crimes are reported, so actual levels may be higher than the data shows.

- **Limited spatial detail** Borough-level data may miss neighborhood-level patterns.

1.9 References

[go back to the top]

An Empirical Study of Los Angeles', University of Pennsylvania Law Review, 161(3), pp. 699–756.

Butts, C.T. et al. (2012) 'Geographical variability and network structure', Social Networks, 34(1), pp. 82–100. Available at: https://doi.org/10.1016/j.socnet.2011.08.003.

Groff, E.R. and Lockwood, B. (2014) 'Criminogenic Facilities and Crime across Street Segments in Philadelphia: Uncovering Evidence about the Spatial Extent of Facility Influence', Journal of Research in Crime and Delinquency, 51(3), pp. 277–314. Available at: https://doi.org/10.1177/0022427813512494.

He, Q. and Li, J. (2022) 'The roles of built environment and social disadvantage on the geography of property crime', Cities, 121, p. 103471. Available at: https://doi.org/10.1016/j.cities.2021.103471.

Lai, J. et al. (2025) 'Exploring the built environment impacts on Online Car-hailing waiting time: An empirical study in Beijing', Computers, Environment and Urban Systems, 115, p. 102205. Available at: https://doi.org/10.1016/j.compenvurbsys.2024.102205.

Phillips, D.C. and Sandler, D. (2015) 'Does public transit spread crime? Evidence from temporary rail station closures', Regional Science and Urban Economics, 52, pp. 13–26. Available at: https://doi.org/10.1016/j.regsciurbeco.2015.02.001.