Machine Learning Final Project

Crime Prediction of Category in San Francisco

工管四乙 109370210 黃鈺凱
工管四乙 109370211 陳國誌

Introduction

San Francisco, once home to the notorious Alcatraz prison from 1934 to 1963, now thrives as a hub of technological innovation. Despite its transformation into a tech mecca, the city continues to grapple with significant social challenges. Rising wealth inequality, severe housing shortages, and the ubiquitous presence of costly digital gadgets contribute to an ongoing tapestry of urban crime.

In this analysis, we explore a dataset encompassing nearly 12 years of crime reports across San Francisco's diverse neighborhoods

Data

Training Set

- Dates
- Category
- Descript
- PdDistrict
- Resolution
- Address
- X Y coordinate

The training set consists of 878,049 entries

Test Set

- ID
- Dates
- DayOfWeek
- Address
- X Y coordinate

The test set also contains 878,049 entries

Similar to the training set.
But without the Category, Descript, and Resolution columns.

Exploratory Data Analysis (EDA)

To facilitate analysis and modeling, we need to convert categorical variables into numerical values. Given the high number of unique values in some columns (e.g., Category with 39 unique values), using one-hot encoding is impractical. Instead, we use count encoding for categorical variables and mapping for ordinal variables.

Hypotheses

Hypothesis 1:

Crimes are more likely to occur on weekdays than weekends.

Hypothesis 2:

Crimes are more likely to occur during late night/early morning hours compared to other times of the day.

Hypothesis 3:

Crimes of similar types are more likely to occur in close proximity to each other.

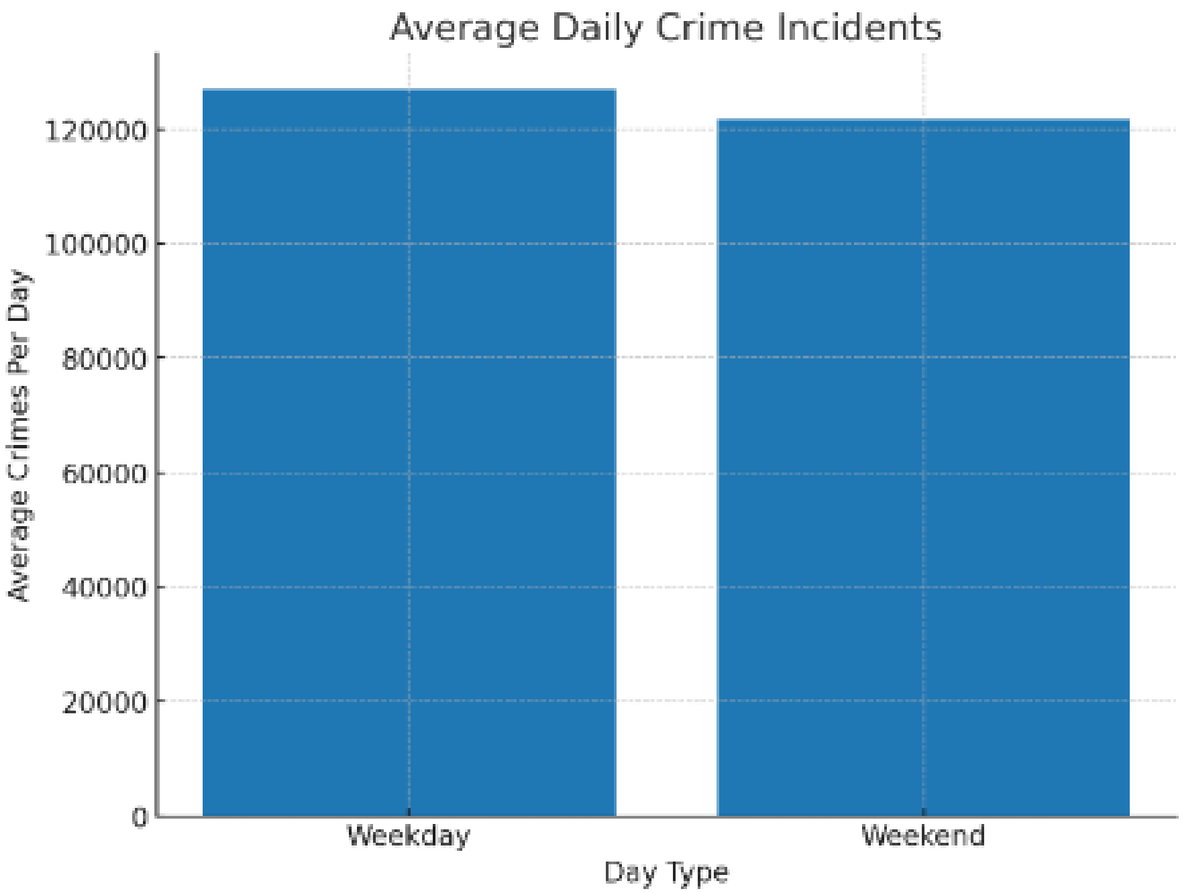
Hypothesis 1

Weekday vs. Weekend Crimes

Result

Average weekday crimes per day: **126,906.40**
Average weekend crimes per day: **121,758.50**

Weekday > Weekend

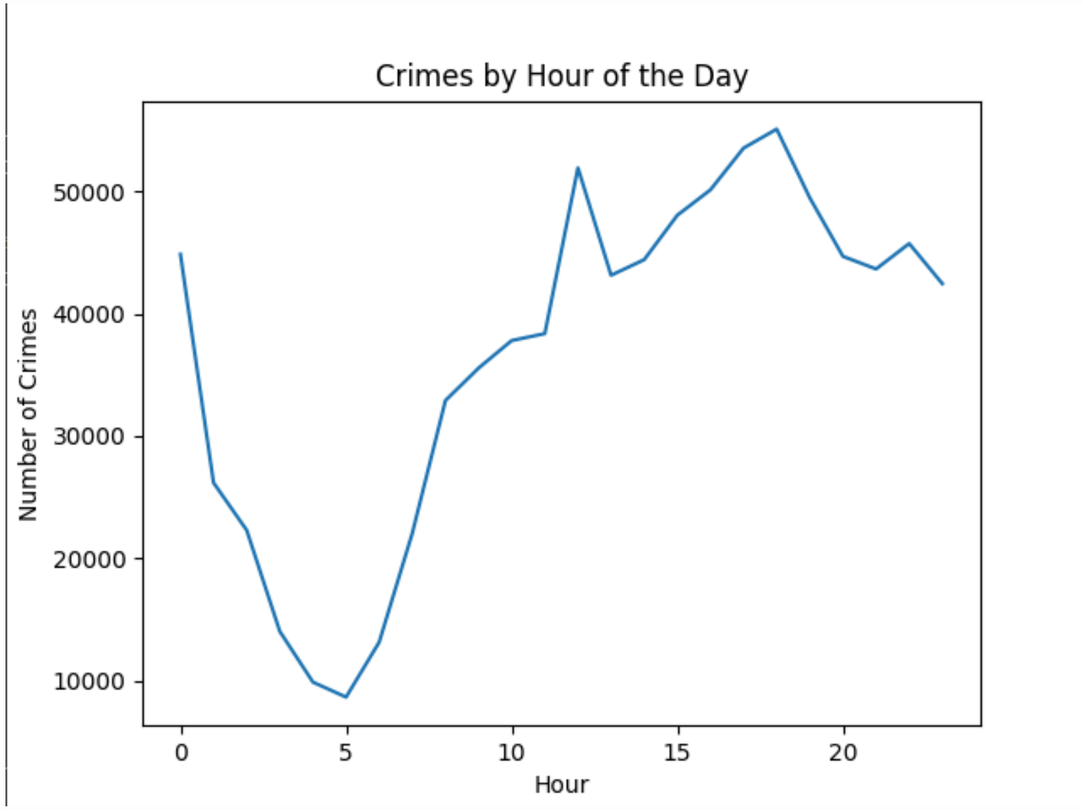


Hypothesis 2

Morning: 6 AM - 12 PM
Noon: 12 PM - 1 PM
Afternoon: 1 PM - 6 PM
Evening: 6 PM - 10 PM
Night: 10 PM - 6 AM

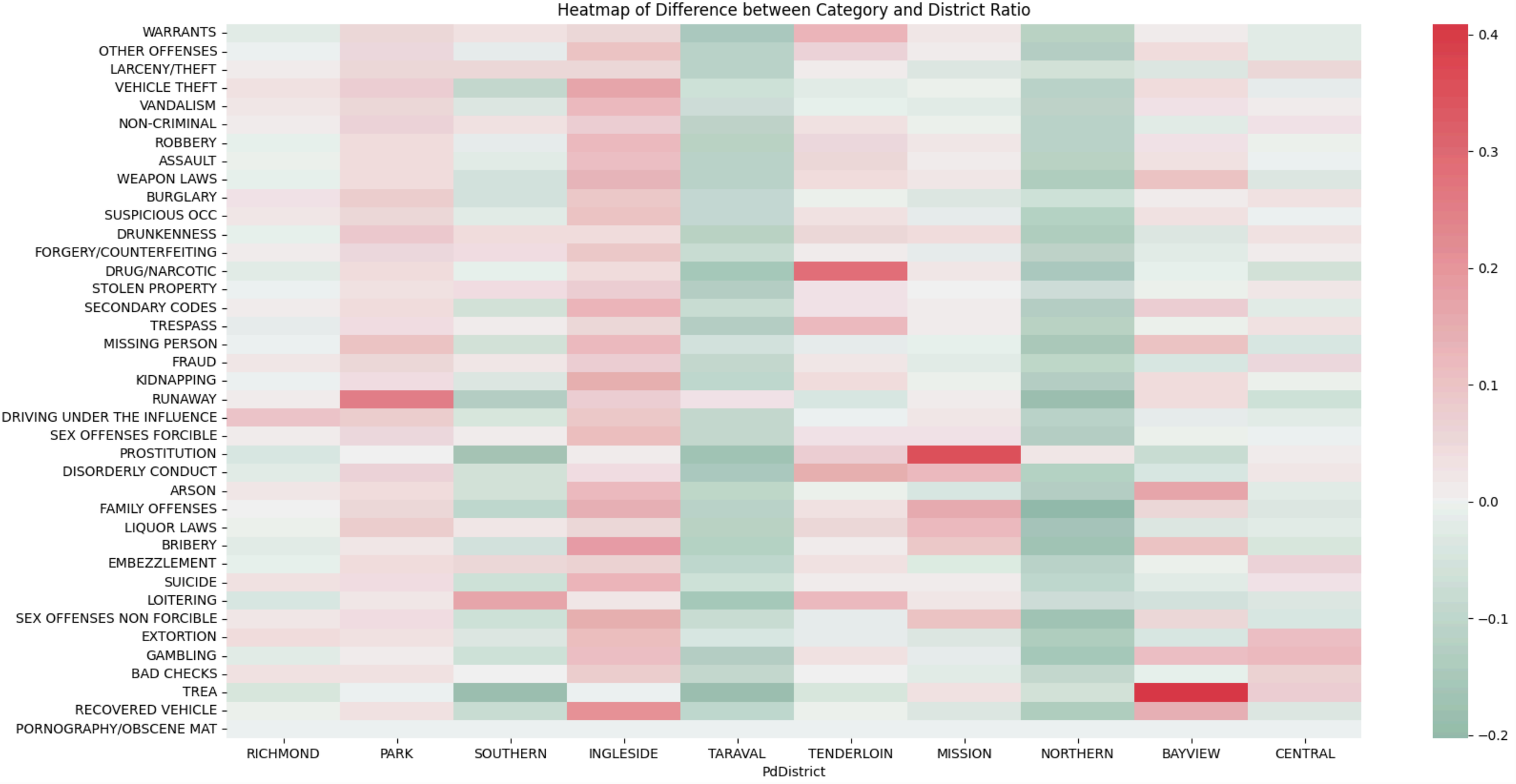
Findings

Evening (6 PM - 10 PM)
Noon (12 PM)



Hypothesis 3

Proximity to Crime Category Hotspots Influences Crime Likelihood


Heatmap of Difference between Category and District Ratio
WARRANTS
OTHER OFFENSES
LARCENY/THEFT
VEHICLE THEFT
VANDALISM
NON-CRIMINAL
ROBBERY
ASSAULT
WEAPON LAWS
BURGLARY
SUSPICIOUS OCC
DRUNKENNESS
FORGERY/COUNTERFEITING
DRUG/NARCOTIC
STOLEN PROPERTY
SECONDARY CODES
TRESPASS
MISSING PERSON
FRAUD
KIDNAPPING
RUNAWAY
DRIVING UNDER THE INFLUENCE
SEX OFFENSES FORCIBLE
PROSTITUTION
DISORDERLY CONDUCT
ARSON
FAMILY OFFENSES
LIQUOR LAWS
BRIBERY
EMBEZZLEMENT
SUICIDE
LOITERING
SEX OFFENSES NON FORCIBLE
EXTORTION
GAMBLING
BAD CHECKS
TREA
RECOVERED VEHICLE
PORNOGRAPHY/OBSCENE MAT
RICHMOND
PARK
SOUTHERN
INGLESIDE
TARAVAL
TENDERLOIN
MISSION
NORTHERN
BAYVIEW
CENTRAL
PdDistrict
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2

Hypothesis 3

Proximity to Crime Category Hotspots Influences Crime Likelihood

Findings

Larceny/Theft is highly concentrated in the **Southern district.**

Drug/Narcotic incidents are heavily concentrated in the **Tenderloin district.**

Vehicle Theft is most prevalent in the Ingleside and **Bayview districts**.

DATASET PREPROCESSING STEPS

We outlines the steps taken to preprocess the San Francisco crime dataset, which includes encoding categorical features, handling temporal data, and preparing the data for training and testing.

1. Converting Dates to Timestamps

```python
def target_encode_proportion(df, column_name):

    # Create a copy of the input DataFrame      Gary, 36 minutes ago • add trining.py
    df_encoded = df.copy()

    # Initialize the CountEncoder
    count_encoder = ce.CountEncoder()

    # Encode the specified column
    new_column_name = f"{column_name}_encoded"
    df_encoded[new_column_name] = count_encoder.fit_transform(df_encoded[column_name])

    return df_encoded

df_eda = target_encode_proportion(df_eda, 'Category')
df_eda = target_encode_proportion(df_eda, 'PdDistrict')
df_eda = target_encode_proportion(df_eda, 'Resolution')
df_eda.head()

def encode_day_of_week(df, column_name):
    day_map = {
        'Monday': 1,
        'Tuesday': 2,
        'Wednesday': 3,
        'Thursday': 4,
        'Friday': 5,
        'Saturday': 6,
        'Sunday': 7
    }

    new_column_name = f"{column_name}_encoded"
    df[new_column_name] = df[column_name].map(day_map)

    return df
```

To handle the temporal aspect of the data, we converted the 'Dates' column into a numeric format (timestamps), which represents the number of seconds since January 1, 1970.

2. Feature Engineering

```python
def add_weekday_weekend_pattern(df):
    category_column = 'Category'
    day_column = 'DayOfWeek'

    category_weekday_counts = df.groupby([category_column, day_column])[category_column].count().unstack(fill_value=0)

    def check_pattern(row):
        weekdays = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
        weekends = ['Saturday', 'Sunday']

        weekday_avg = row[weekdays].mean()
        weekend_avg = row[weekends].mean()

        if weekday_avg == 0 and weekend_avg == 0:
            return "?"
        elif weekday_avg > weekend_avg:
            return 1
        elif weekday_avg < weekend_avg:
            return 0
        else:
            return "?"

    pattern_column = 'Weekday>Weekend_Pattern'
    patterns = category_weekday_counts.apply(check_pattern, axis=1)

    df[pattern_column] = df[category_column].map(patterns)

    return df

df_eda = add_weekday_weekend_pattern(df_eda)
df_eda.describe()
```

We performed several feature engineering steps to prepare the dataset for modeling. These include encoding categorical features and adding new features based on temporal and spatial data.

3. Creating a Preprocessing Pipeline

```python
def create_preprocessing_pipeline():
    # 定義 ColumnTransformer
    categorical_features = ['DayOfWeek', 'PdDistrict', 'Resolution', 'Category']
    ordinal_features = ['DayOfWeek']
    numerical_features = ['X', 'Y']
    columns_to_drop = ['Dates', 'Descript', 'DayOfWeek', 'PdDistrict', 'Resolution', 'Category','Address']

    preprocessor = ColumnTransformer(
        transformers=[
            ('count_encode', CountEncoder(), categorical_features),
            ('ord_encode', OrdinalEncoder(), ordinal_features),
            ('date_encode', FunctionTransformer(convert_dates_to_numeric), []),
            ('pattern_recog', FunctionTransformer(add_weekday_weekend_pattern), []),
            ('time_of_day', FunctionTransformer(extract_time_of_day), []),
            ('temp_level', FunctionTransformer(add_temp_level), []),
            ('num_pass', 'passthrough', numerical_features),
            ('drop_cols', 'drop', columns_to_drop)
        ]
    )

    preprocessor.set_output(transform='pandas') # 設定結果以 pandas 輸出

    # 建立 Pipeline
    pipeline = Pipeline([
        ('preprocess', preprocessor)
    ])

    # 返回 Pipeline 和 ColumnTransformer
    return pipeline, preprocessor

pipeline, preprocessor = create_preprocessing_pipeline()
# 進行資料預處理
X = pipeline.fit_transform(df_train)
```

We utilized '**ColumnTransformer**' and '**Pipeline**' from scikit-learn to streamline the preprocessing steps. This approach ensures that all necessary transformations are applied consistently to both training and testing datasets.

4. Mapping Encoded Features to Original Categories

```
# 遍歷每個犯罪類別
for category in df_eda['Category'].unique():
    # 選擇該犯罪類別的資料子集
    category_data = df_eda[df_eda['Category'] == category]

    # 計算該犯罪類別的總數
    category_total = len(category_data)

    # 初始化該類別的字典
    category_district_ratios[category] = {district: 0.0 for district in all_districts}

    # 遍歷每個警區
    for district in category_data['PdDistrict'].unique():
        # 計算該警區的犯罪數量
        district_count = len(category_data[category_data['PdDistrict'] == district])

        # 計算該警區犯罪數量佔該類別總數的比例
        ratio = district_count / category_total

        # 將該比例寫入字典
        category_district_ratios[category][district] = ratio

# 建立一個包含所有警區的列表
all_districts = set()
for district_ratios in category_district_ratios.values():
    all_districts.update(district_ratios.keys())
```

For the '**Category**' feature, we need to retain the original category names as we will predict crime categories rather than numerical codes. We create a mapping to revert the encoded values back to their original form.

5. Mapping Encoded Features to Original Categories

```
# 分割資料為訓練集和測試集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

We separated the target variable (Category) from the feature set and divided the data into training and testing subsets.

MODLE BUILDING

1.Random Forest Classifier
2. XGBoost Classifier

Random Forest Classifier

```
# 建立隨機森林模型
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

# 訓練模型
rf_model.fit(X_train, y_train)

from sklearn.metrics import accuracy_score

# 在測試集上進行預測
y_pred = rf_model.predict(X_test)

# 計算準確率
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

XGBoost (Extreme Gradient Boosting) Classifier

```
# 建立 XGBoost 分類模型
xgb_model = xgb.XGBClassifier(n_estimators=100, random_state=42, objective='multi:softmax', num_class=39)

# 訓練模型
xgb_model.fit(X_train, y_train_encoded)

# 在測試集上進行預測
y_pred_encoded = xgb_model.predict(X_test)

# 將預測結果轉換回原始類別標籤
y_pred = label_encoder.inverse_transform(y_pred_encoded)

# 計算準確率
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

Result: Predict Accuracy

Random Forest Classifier

```
Accuracy: 0.26352713399900917
```

XGBoost Classifier

```
Accuracy: 0.2740219805250271
```

FUTURE WORKS

While the Random Forest and XGBoost models provided a starting point, their performance suggests room for improvement.

Future Works

Enhanced Feature Engineering

- Incorporate Resolution Outcomes
- Temporal Aggregation

Advanced Modeling Techniques

Implementing models that explicitly account for **spatial** and **temporal** dependencies, such as spatial-temporal autoregressive models or geographically weighted regression.

THANK YOU!