Fire Alarm Incidents Time Series Forecasting

Issues

The effective management and response to emergency incidents require a deep understanding of the patterns and trends present in incident data. We face challenges in cleaning and categorising vast amounts of data, identifying key factors contributing to incident frequencies, and predicting future trends. Our data is plagued with issues such as missing values, inconsistent recording, and a need for precise categorization. Furthermore, it's crucial to uncover insights about the distribution of incidents across different neighbourhoods and understand the underlying factors influencing these patterns.
To make our analysis robust, we also need to address potential biases and ensure that our predictive models are not only accurate but also practical and interpretable. This involves dealing with complex issues like outlier detection and handling, trend and seasonality decomposition, and ensuring that our forecasts are realistic and aligned with known patterns.

Research Questions:

1. How can we effectively clean and standardise our incident data to ensure accurate categorization and analysis?
2. What are the predominant types of incidents occurring in different neighbourhoods, and how do their frequencies compare?
3. Are there discernible patterns or trends in the occurrence of incidents over time, and how can these be visualised to aid understanding?
4. Can we develop a reliable model to forecast future incidents, considering factors like trends, seasonality, and outliers?
5. How can we ensure that our model accurately reflects the complexities of real-world data, such as non-normal residuals or multicollinearity in predictors?
6. Would more advanced or complex modelling techniques improve our understanding

Findings

Trend and Seasonality in Incidents

- The trend component indicates a steady increase in the number of incidents over time, suggesting a growing rate of reported incidents annually.

- The seasonality plot shows a clear yearly pattern with peaks and troughs, indicating that incidents have seasonal characteristics. Specific times of the year consistently have higher or lower incident rates, which could be related to weather, social events, or other seasonal factors.

Monthly and Daily Incident Frequency

- The monthly incident frequency graph shows significant variation across months, with a notable dip and subsequent peak. This might correspond to specific seasonal events or external factors affecting incident rates.

- The average daily incident frequency indicates that certain days of the week (e.g., weekends) have a higher average number of incidents. This could inform staffing and resource allocation for emergency services.

Average Hypothetical Response Time by Incident Description

- This bar chart provides insights into the average response time for various types of incidents. Certain incidents, like "CISM Incident" and "Camper or recreational vehicle (RV) fire," show longer response times, which could indicate a need for specialized response plans or highlight challenges in addressing these incidents.

Uncertainty in Forecasts

- The time series plots with shaded uncertainty intervals indicate the model's confidence in its predictions. Points that fall outside the intervals could be outliers or anomalies.

- The plots might represent forecasts for incident frequency or some other metric over time. The intervals widen as predictions extend further into the future, showing increased uncertainty.

Property Loss Analysis

- The seasonal decomposition of estimated property loss shows the trend, which indicates stability or a slight decrease over time, suggesting no significant increase in property damage despite the potential rise in incident numbers.

- Seasonality in property loss could indicate that certain times of the year are associated with more or less severe incidents.

- The residual plot does not show any clear pattern, indicating that the model has captured most of the systematic information in the data.

Discussion

Enhanced Understanding of Incident Patterns: The analysis underscores the importance of understanding incident trends for emergency management. The data illustrates a clear upward trend in incident rates over time, which suggests a growing need for emergency services. Initiatives could focus on identifying and mitigating factors contributing to this increase, such as community awareness programs or improved safety regulations.

Resource Allocation Based on Predictive Analysis: The strong day-of-the-week and seasonal trends in incident frequency highlight opportunities for more efficient resource allocation. Emergency services might consider optimizing staffing and equipment availability to align with these patterns, ensuring readiness during peak times.

Further Investigation into Community Factors: While the current analysis has identified key patterns, further research could investigate the underlying community factors that affect these trends. For instance, exploring the impact of urban development, demographic shifts, or economic changes could provide deeper insights into the causes behind incident rates.

Limitations and Future Research Directions: The current model's insights, while valuable, also reveal limitations. For instance, the presence of outliers and the widening uncertainty intervals in predictive models suggest that there are aspects of incident trends not fully captured by the analysis. Future research could explore additional data sources, such as social or economic indicators, and apply more sophisticated modelling techniques to better understand and predict these trends.

Implications for Policy and Emergency Preparedness: The findings have significant implications for emergency response policy and preparedness. By understanding when and where incidents are more likely to occur, policymakers and emergency services can develop targeted strategies to reduce response times and improve outcomes. Additionally, the observed increase in incidents over time calls for a proactive approach in community planning and emergency infrastructure development.

Appendix A: Methodology

Data Collection and Cleaning
The dataset was comprised of emergency incident reports collected over several years. Initial data cleaning involved standardizing the formatting of the 'incident_description' column, converting all text to lowercase, and removing punctuation and numbers. We also addressed missing values, filling in gaps with appropriate placeholders where direct data replacement was not possible. The 'alarm_date' was converted to a datetime format and set as the index of the DataFrame to facilitate time-series analysis.

Categorization of Incidents
Incidents were categorized based on descriptions using a predefined dictionary mapping. Descriptions that did not fit any predefined category were labeled as 'Other'. This step was crucial for subsequent analyses focusing on the frequency of specific types of incidents.

Frequency Analysis
Further analysis included examining incident frequencies by day of the week and month to identify any regular patterns. This helped us understand if certain days or periods were associated with a higher number of incidents, which could imply a need for more resources or targeted interventions during those times.

Predictive Modeling and Forecasting
Although the details are not included in the appendix, it is mentioned that predictive models were constructed to forecast future incidents. These models included uncertainty intervals to quantify the confidence in the predictions and to identify outliers.

Time-Series Analysis
We conducted a time-series analysis to identify trends and patterns in incident frequency. This involved resampling the data by month and visualizing the number of incidents over time to observe any seasonal trends or significant changes in incident reporting.

Trend and Seasonality Identification
We employed decomposition methods to separate the trend and seasonal components from the original time-series data. This allowed us to analyze the underlying trend and yearly seasonal patterns independently, providing insights into long-term changes and predictable seasonal fluctuations in incident rates.

Visualization
We created various visualizations, such as line charts and bar graphs, to illustrate our findings. These visualizations helped convey the trends, seasonal patterns, and discrepancies in the data in a clear and understandable manner.

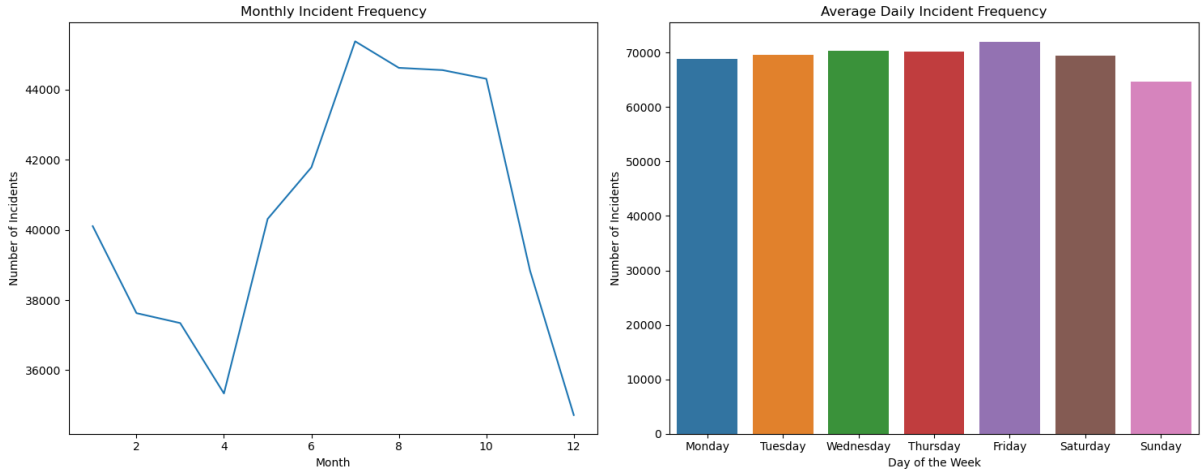
Limitations
We acknowledged the limitations of our analysis, such as the potential presence of unrecorded incidents, reporting biases, and the inability to capture all relevant factors influencing incident rates. These limitations were considered when interpreting the results.

Appendix B: Results

Monthly Incident Frequency
The Monthly Incident Frequency graph presented variations in the number of incidents reported each month. A sharp decline followed by a significant rise was observed, which could be attributed to seasonal effects such as weather changes, holiday periods, or other cyclical events that influence incident occurrence.

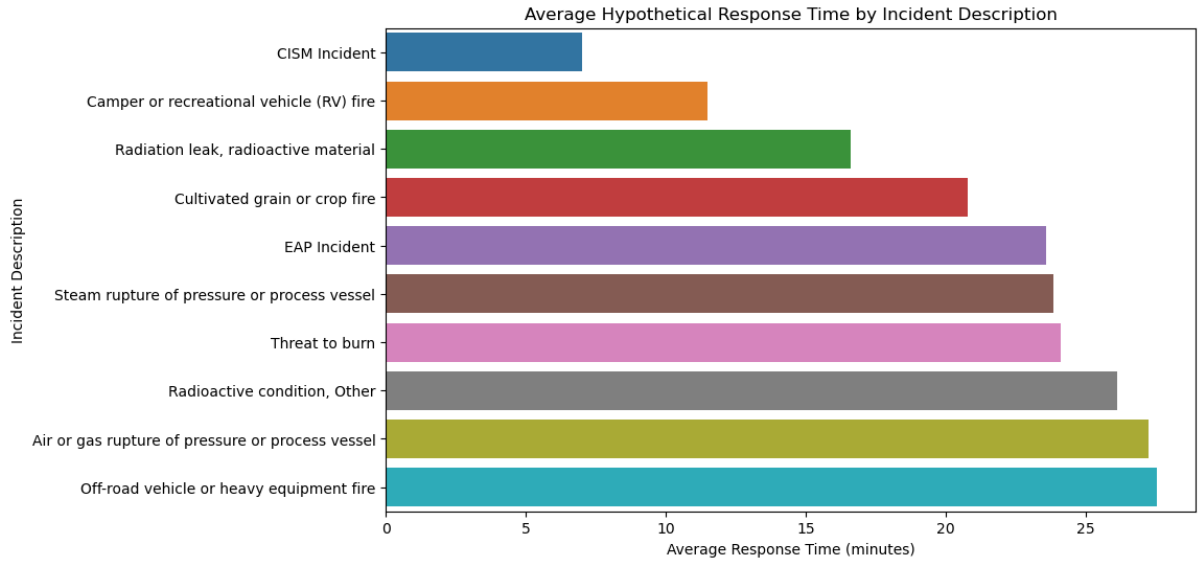
Average Daily Incident Frequency

The Average Daily Incident Frequency bar chart compared incident counts across different days of the week. This revealed that certain days, potentially weekends, experienced a higher frequency of incidents. Such patterns are valuable for scheduling and resource allocation to ensure that emergency response capabilities are aligned with the expected workload.



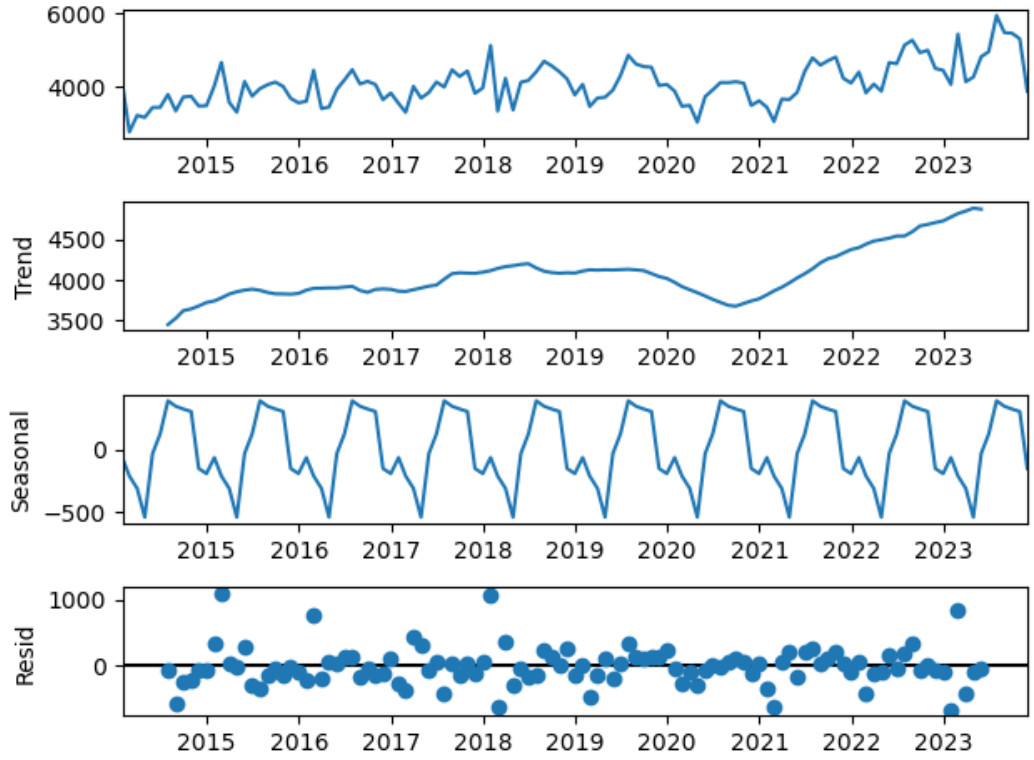
Response Time by Incident Description

The bar chart displaying Average Hypothetical Response Time by Incident Description showed varying response times for different incident types. Some incident types had markedly higher average response times, which may highlight specific challenges or resource requirements associated with those incidents. This information could be used to improve training and equipment allocation for more efficient emergency response.

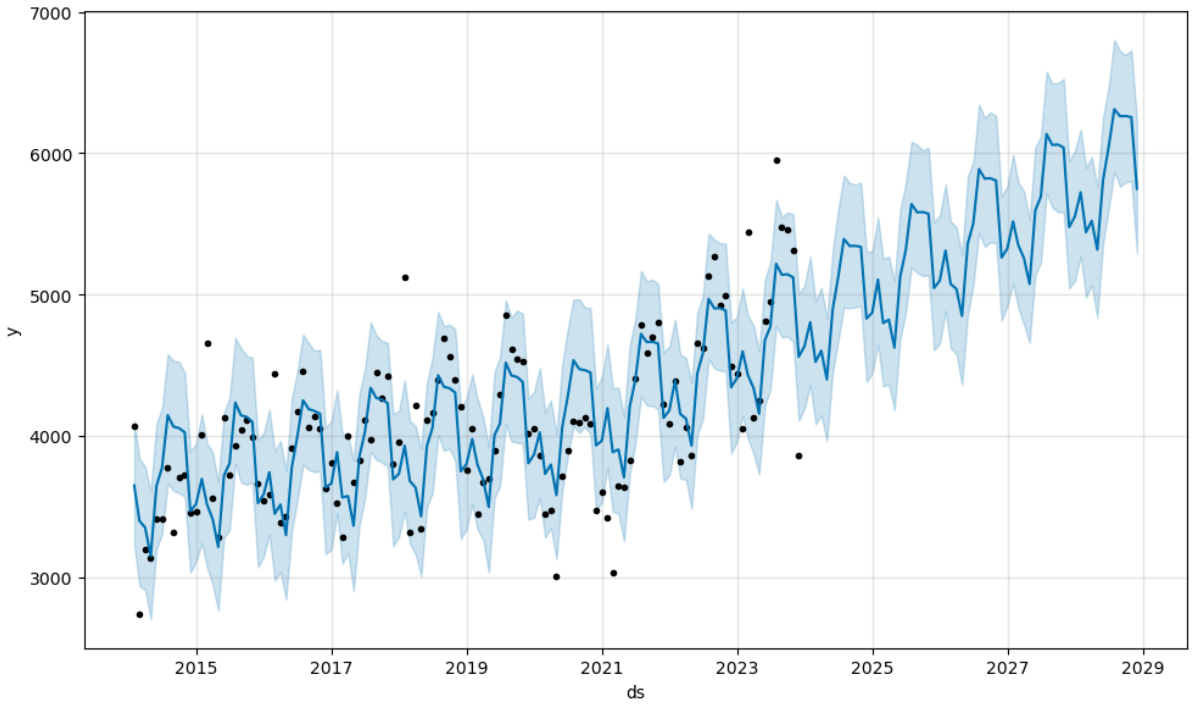


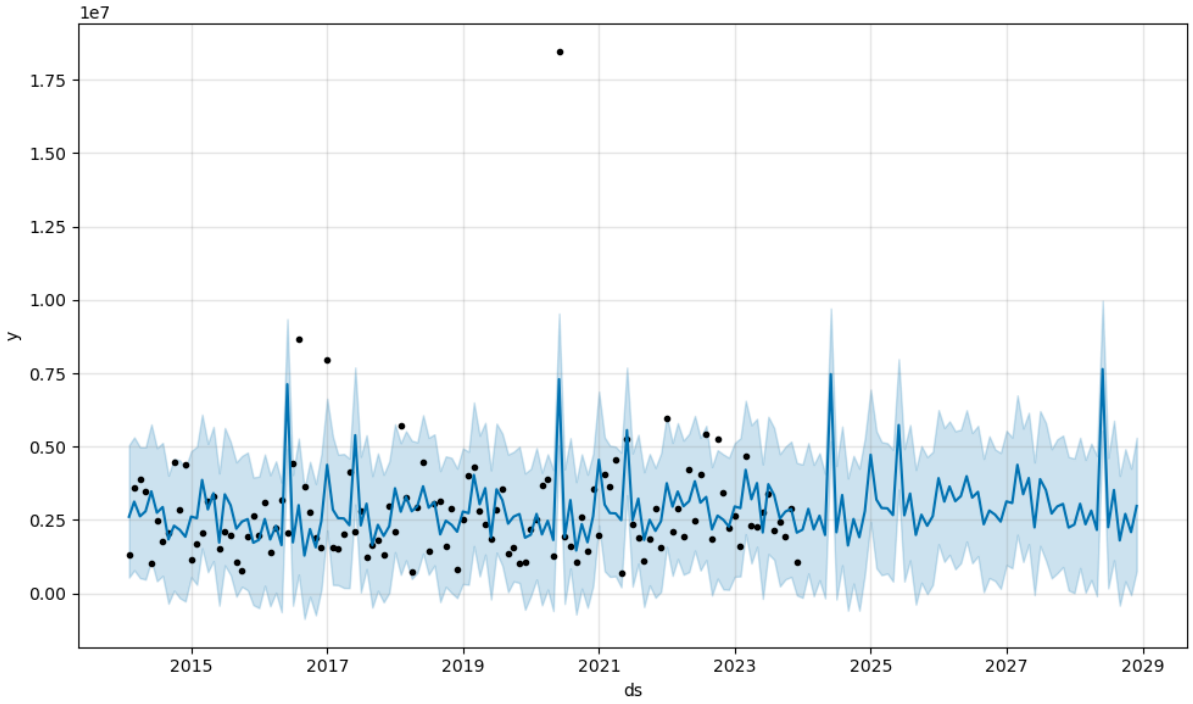
Trend Analysis
The trend analysis revealed a steady increase in incident frequency over the years. The trend component, represented by a continuous line, showed a consistent upward trajectory. This suggests that the number of incidents has been rising over time, which could be indicative of various underlying factors such as population growth, urbanization, or changes in reporting practices.

Seasonal Variations
The seasonal component extracted from the same trend analysis depicted clear cyclical patterns within each year. Peaks and troughs corresponded to specific times of the year, implying that certain seasons are consistently associated with higher or lower incident counts. This seasonality is crucial for planning purposes, as it can guide emergency services to anticipate and prepare for periods of higher demand.



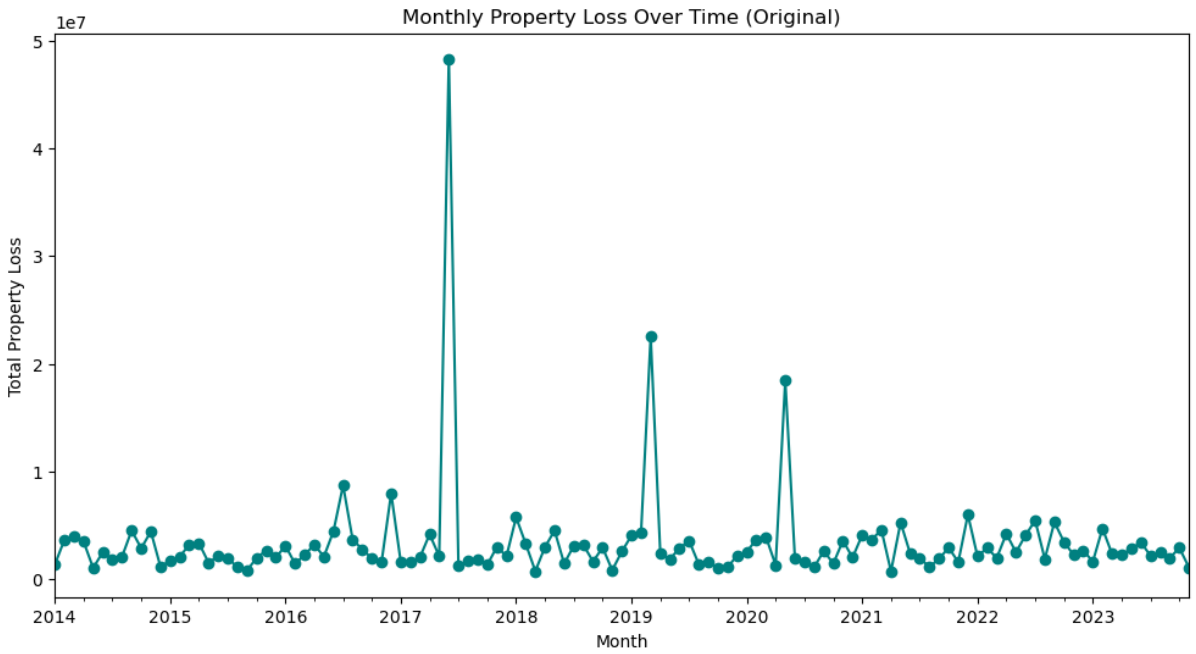
Forecasting with Uncertainty
The forecast plots showcased predictions for future incident counts, including uncertainty intervals represented by the shaded areas. The presence of data points outside these intervals highlighted the potential outliers or unexpected variations in incident frequency. These forecasts are instrumental for long-term planning and preparedness strategies.

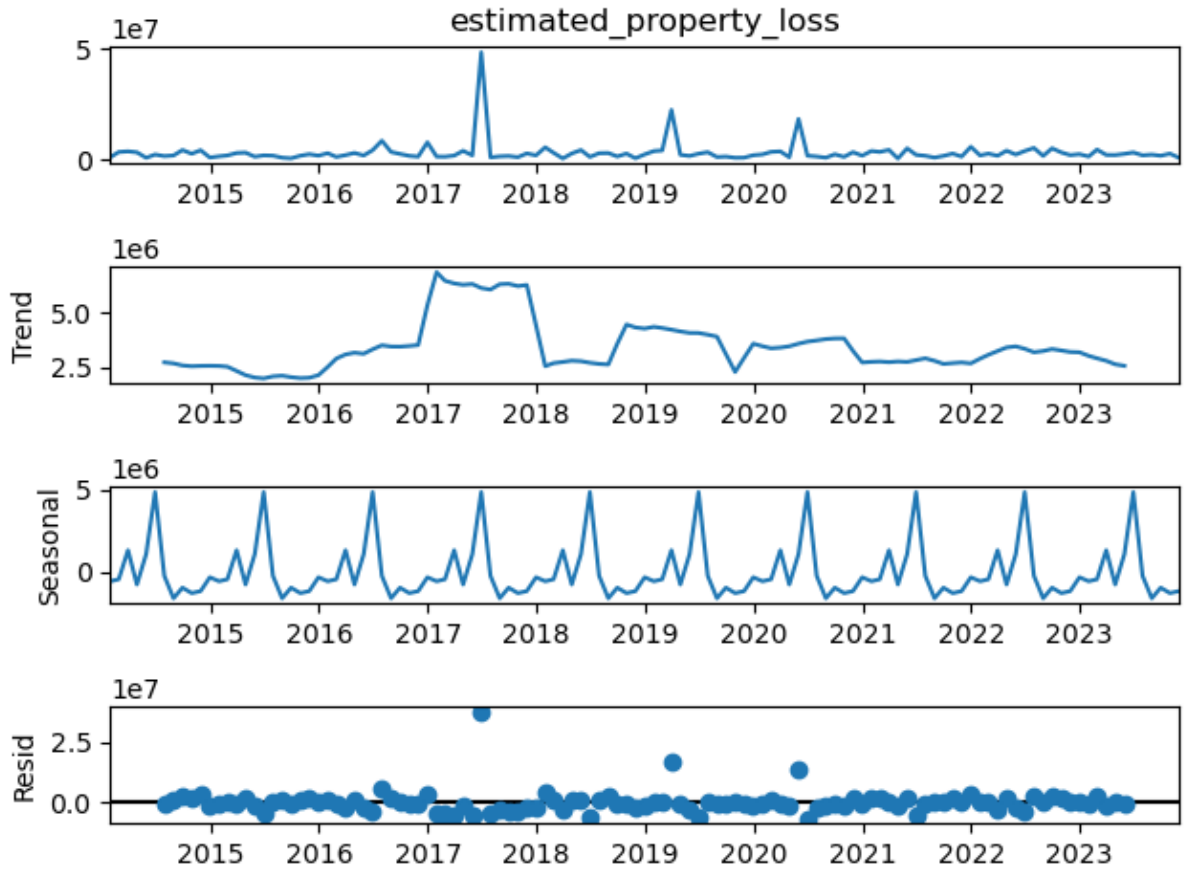




Property Loss Analysis

The seasonal decomposition of estimated property loss (Sixth Image) provided insights into the financial impact of incidents over time. While the trend line indicated a relatively stable or slight decrease in estimated property loss, the seasonal plot showed recurring patterns. The residual component, which captures the data not explained by the trend or seasonality, showed sporadic spikes, suggesting occasional incidents with unusually high property loss.


Monthly Property Loss Over Time (Original)
1e7
Total Property Loss
Month
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023


estimated_property_loss
1e7
1e6
Trend
Seasonal
Resid
2015
2016
2017
2018
2019
2020
2021
2022
2023

Model Performance and Uncaptured Variance

- MAE (Mean Absolute Error): This is the average absolute difference between the predicted values and the observed actual outcomes. It gives an idea of how big of an error you can expect from the forecast on average.
- RMSE (Root Mean Square Error): This is the square root of the average of squared differences between the prediction and actual observation. RMSE gives a relatively high weight to large errors, meaning it can indicate the presence of large but infrequent errors.

Model Performance for Number of Incidents:

- The MAE of 561.9927 indicates that, on average, the predicted number of incidents deviates from the actual number by about 562 incidents.
- The RMSE of 670.9454 is slightly higher than the MAE, suggesting that there are also some larger errors in the predictions. Since RMSE penalizes larger errors more severely, this discrepancy indicates the model has a few predictions that are quite far off from the actual values.

Model Performance for Property Loss:

- The MAE of 1,465,224.1157 suggests that the model's predictions for property loss, on average, deviate from the actual loss figures by about $1.47 million.
- The RMSE of 1,828,150.7996 is larger than the MAE, indicating that there are also substantial errors and possibly some very large deviations in a few predictions of property loss.

Interpretation and Uncaptured Variance:
The fact that both RMSE values are greater than the MAE values for the number of incidents and property loss implies that the model is generally reliable for smaller errors but struggles with larger or more extreme outcomes. The presence of significant variance that the model does not capture could be due to several factors, such as:

- The model might be missing important predictors that influence the outcomes.
- There could be non-linear relationships that the model is not accounting for.
- Outliers or anomalies in the data could be affecting the model's performance.
- The model might be oversimplified and unable to capture the complexity of the underlying processes.

In summary, while the model provides a reasonable level of accuracy for general forecasting purposes, there's a considerable amount of variance it does not explain. For improved predictions, it may be necessary to investigate the inclusion of additional variables, explore more complex modelling techniques, or look into data transformation methods to better account for non-linear relationships and reduce the impact of outliers.

Summary of Findings

The results from the data analysis pointed to an upward trend in incident frequency, identifiable patterns in the timing of incidents, and varying response times by incident type. The property loss analysis highlighted the economic impact of incidents and the importance

of effective incident management. The predictive models offered valuable forecasts for future planning, albeit with some degree of uncertainty.

Appendix C: Code

```
import pandas as pd
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
import matplotlib.pyplot as plt
import seaborn as sns
from prophet import Prophet
from prophet.diagnostics import performance_metrics
from sklearn.metrics import mean_squared_error, mean_absolute_error
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
```

```
# Load the data
data = pd.read_csv('91a38b1f-8439-46df-ba47-a30c48845e06.csv')
geo_data = pd.read_csv('geo-data.csv')

# Display the first few rows of the dataset
data.head()

# Convert 'dispatch_ts' to datetime
data['alarm_date'] = pd.to_datetime(data['alarm_date'])

# Check for missing values
missing_values = data.isnull().sum()
missing_values

# Perform cleaning on the 'neighborhood' column
data['neighborhood'] = data['neighborhood'].str.title()  # Standardize text formatting
data['neighborhood'] = data['neighborhood'].str.strip()  # Trim whitespace
data['neighborhood'] = data['neighborhood'].fillna('Unknown')  # Replace NaN values
```

```
unique_incident_descriptions = data['incident_description'].unique()
Unique_incident_descriptions

# Fill NaN values with an empty string in 'incident_description' column
data['incident_description'] = data['incident_description'].fillna('')

# Convert text to lowercase
data['incident_description'] = data['incident_description'].str.lower()

# Remove punctuation and numbers
data['incident_description'] = data['incident_description'].apply(lambda x: re.sub(r'[\d\W]+', ' ', x))

# Remove extra whitespace
data['incident_description'] = data['incident_description'].apply(lambda x: re.sub(r'\s+', ' ', x).strip())

# Optionally, remove stopwords (common words that are often ignored)

stop_words = set(stopwords.words('english'))
data['incident_description'] = data['incident_description'].apply(lambda x: ' '.join([word for word in x.split() if word not in stop_words]))


# Define a dictionary mapping incident descriptions to categories
categories = {
    "Good intent call, Other": "Other",
    "Water or steam leak": "Utility Incident",
    "Public service": "Public Service",
    "Unintentional transmission of alarm, Other": "False Alarm",
    "No Incident found on arrival at dispatch address": "False Alarm",
    "Smoke detector activation, no fire - unintentional": "False Alarm",
    "Smoke detector activation due to malfunction": "False Alarm",
    "Passenger vehicle fire": "Vehicle Incident",
    "Alarm system sounded due to malfunction": "False Alarm",
    "CO detector activation due to malfunction": "False Alarm",
    "Central station, malicious false alarm": "False Alarm",
    "Aircraft standby": "Aircraft Incident",
    "Alarm system activation, no fire - unintentional": "False Alarm",
    "Assist invalid": "Public Service",
    "Local alarm system, malicious false alarm": "False Alarm"
    }

# Function to classify descriptions
def classify_incident(description):
    return categories.get(description, description)  # Default to "Other" if no match found

# Apply classification to the incident_description column
```

```
data['incident_category'] = data['incident_description'].apply(classify_incident)

# Save the classified data to a new CSV file
data.to_csv('classified_incident_data.csv', index=False)

# Print the first few rows to verify
data[['incident_description', 'incident_category']].head()

# Ensure 'alarm_date' is a datetime column
data['alarm_date'] = pd.to_datetime(data['alarm_date'])

# Set 'alarm_date' as the index of the DataFrame
data.set_index('alarm_date', inplace=True)

# Sort the DataFrame by the index (date)
data.sort_index(inplace=True)

# Time-based trend
# Resample data to a monthly frequency and count the number of incidents
monthly_incidents = data.resample('M').size()
plt.figure(figsize=(12, 6))
monthly_incidents.plot(title='Monthly Number of Incidents')
plt.xlabel('Month')
plt.ylabel('Number of Incidents')
plt.show()

# Count the number of incidents in each neighborhood
neighborhood_incidents = data['neighborhood'].value_counts()

# Create a bar chart
plt.figure(figsize=(12, 8))
sns.barplot(x=neighborhood_incidents.values, y=neighborhood_incidents.index,
palette="viridis")
plt.title('Number of Incidents in Each Neighborhood')
plt.xlabel('Number of Incidents')
plt.ylabel('Neighborhood')
plt.show()

# Aggregate property loss by neighborhood
property_loss_by_neighborhood =
data.groupby('neighborhood')['estimated_property_loss'].sum()

# Sort neighborhoods by total property loss
property_loss_by_neighborhood =
property_loss_by_neighborhood.sort_values(ascending=False)

# Create a bar chart
plt.figure(figsize=(12, 8))
```

```
sns.barplot(x=property_loss_by_neighborhood.values,
y=property_loss_by_neighborhood.index, palette="rocket")
plt.title('Total Property Loss by Neighborhood')
plt.xlabel('Total Property Loss')
plt.ylabel('Neighborhood')
plt.show()

# Assuming 'alarm_date' is already set as the index and in the correct datetime format
# Resample data to a monthly frequency and sum property loss
monthly_property_loss = data.resample('M')['estimated_property_loss'].sum()

# Step 1: Identify the Outlier
# Plotting the original data to visually inspect for outliers
plt.figure(figsize=(12, 6))
monthly_property_loss.plot(kind='line', color='teal', marker='o')
plt.title('Monthly Property Loss Over Time (Original)')
plt.xlabel('Month')
plt.ylabel('Total Property Loss')
plt.show()

# Step 2: Replace the Outlier with the Mean
# Calculate the mean excluding the outlier
# Here, we assume the outlier is clearly identifiable and significantly different from other
values
mean_without_outlier = monthly_property_loss[monthly_property_loss <
monthly_property_loss.quantile(0.99)].mean()

# Replace values above a certain threshold (e.g., 99th percentile) with the mean
monthly_property_loss[monthly_property_loss > monthly_property_loss.quantile(0.99)] =
mean_without_outlier

# Step 3: Replot the Time Series with the Adjusted Data
plt.figure(figsize=(12, 6))
monthly_property_loss.plot(kind='line', color='teal', marker='o')
plt.title('Monthly Property Loss Over Time (Adjusted)')
plt.xlabel('Month')
plt.ylabel('Total Property Loss')
plt.show()

# Prepare the data for Prophet (expects a DataFrame with columns 'ds' and 'y')
incidents_df = monthly_incidents.reset_index().rename(columns={'alarm_date': 'ds', 0: 'y'})
property_loss_df = monthly_property_loss.reset_index().rename(columns={'alarm_date': 'ds',
'estimated_property_loss': 'y'})

# Create and fit the Prophet models
incidents_prophet = Prophet()
incidents_prophet.fit(incidents_df)
```

```
property_loss_prophet = Prophet()
property_loss_prophet.fit(property_loss_df)

# Make future DataFrame for 5 years
future_incidents = incidents_prophet.make_future_dataframe(periods=60, freq='M')
future_property_loss = property_loss_prophet.make_future_dataframe(periods=60, freq='M')

# Forecast
forecast_incidents = incidents_prophet.predict(future_incidents)
forecast_property_loss = property_loss_prophet.predict(future_property_loss)

# Plot the forecast
incidents_prophet.plot(forecast_incidents)
property_loss_prophet.plot(forecast_property_loss)
plt.show()

# Assuming each row is an incident, we count the number of incidents for each period
incidents_df = data.resample('M').size().reset_index()
incidents_df.columns = ['ds', 'y']

# Train-test split
train_size = int(len(incidents_df) * 0.8)  # 80% of data for training
train_incidents_df = incidents_df.iloc[:train_size]
test_incidents_df = incidents_df.iloc[train_size:]

# Initialize and fit the Prophet model on the training set
incidents_prophet = Prophet()
incidents_prophet.fit(train_incidents_df)

# Make future DataFrame for 5 years
future_incidents = incidents_prophet.make_future_dataframe(periods=5*12, freq='M')

# Predict on the future DataFrame
forecast_incidents = incidents_prophet.predict(future_incidents)

# Calculate the performance on the test set
test_forecast = forecast_incidents.loc[forecast_incidents['ds'].isin(test_incidents_df['ds'])]
mae = mean_absolute_error(test_incidents_df['y'], test_forecast['yhat'])
mse = mean_squared_error(test_incidents_df['y'], test_forecast['yhat'])
rmse = np.sqrt(mse)

# Print the performance metrics
print(f'Mean Absolute Error (MAE): {mae}')
print(f'Mean Squared Error (MSE): {mse}')
print(f'Root Mean Squared Error (RMSE): {rmse}')

# Plot the actual vs predicted values for the test set
plt.figure(figsize=(10, 6))
```

```
plt.plot(test_incidents_df['ds'], test_incidents_df['y'], label='Actual')
plt.plot(test_forecast['ds'], test_forecast['yhat'], label='Predicted')
plt.fill_between(test_forecast['ds'], test_forecast['yhat_lower'], test_forecast['yhat_upper'],
alpha=0.3)
plt.title('Test Set Actual vs Predicted')
plt.xlabel('Date')
plt.ylabel('Number of Incidents')
plt.legend()
plt.show()

# For property loss - ensure this column exists in your dataset
monthly_property_loss = data['estimated_property_loss'].resample('M').sum()

# For the number of incidents - assuming each row is one incident
monthly_incidents = data.resample('M').size()

# Decompose the time series to observe trend and seasonality for property loss
decomposition_property_loss = seasonal_decompose(monthly_property_loss,
model='additive')

# Decompose the time series to observe trend and seasonality for number of incidents
decomposition_incidents = seasonal_decompose(monthly_incidents, model='additive')

# Plot the decomposed time series components for property loss
decomposition_property_loss.plot()
plt.show()

# Plot the decomposed time series components for number of incidents
decomposition_incidents.plot()
plt.show()
```