Data Driven Traffic Insights

Enhancing road safety through analytics

Sebastian Wegstein

Data Scientist
GitHub: TheHessianBo

LinkedIn

Milos Mirkovic

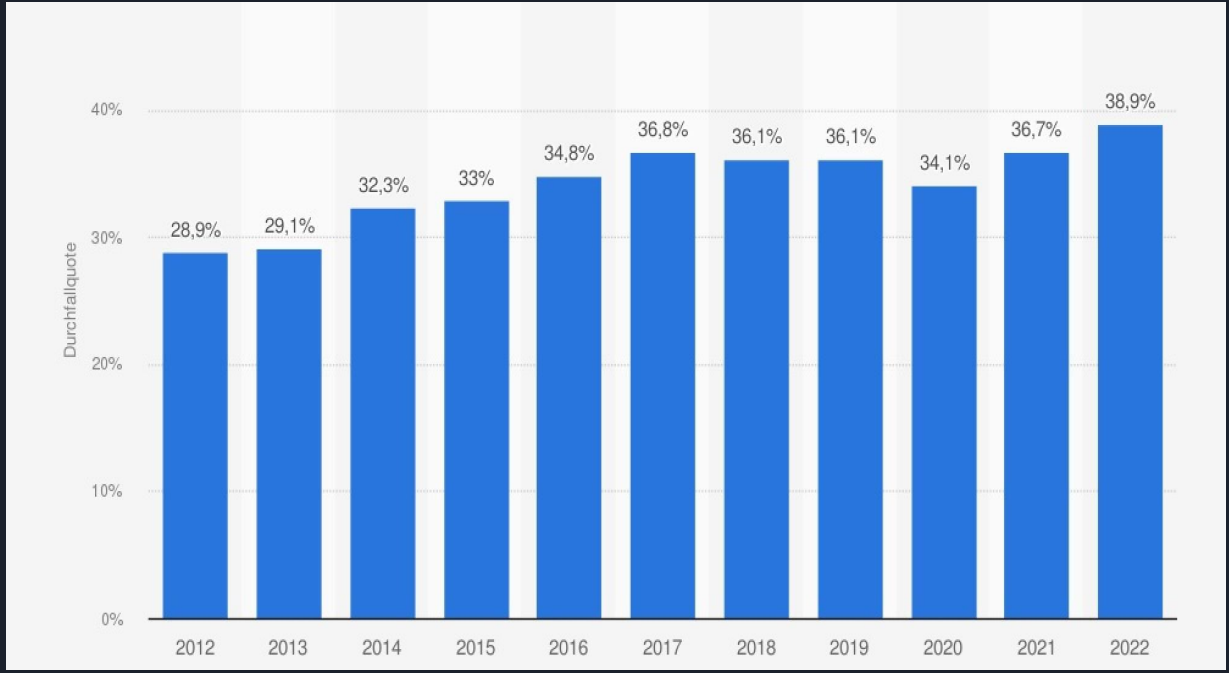
Data Analyst
GitHub: Milos191405

LinkedIn

Agenda

1. Motivation
2. Stakeholder Information
3. Dataset Introduction
4. EDA
5. Time Series Analysis
6. Takeaways
7. Recommendations

Motivation


Durchfallquote
40%
30%
20%
10%
0%
28,9%
29,1%
32,3%
33%
34,8%
36,8%
36,1%
36,1%
34,1%
36,7%
38,9%
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022

Stakeholder

Bundesanstalt für Straßen und Verkehrswesen

- traffic safety
- road construction and technology
- bridges and tunnel construction
- digitalization and automation
- legal and standardization work

Dataset Introduction

- Provided by DSTATIS
- Covers the years 2011 - November 2024
- Location of the accident (freeway, built up areas, outside of town)
- Type of accident (personal injury, material damage, intoxication)

Dataset Introduction

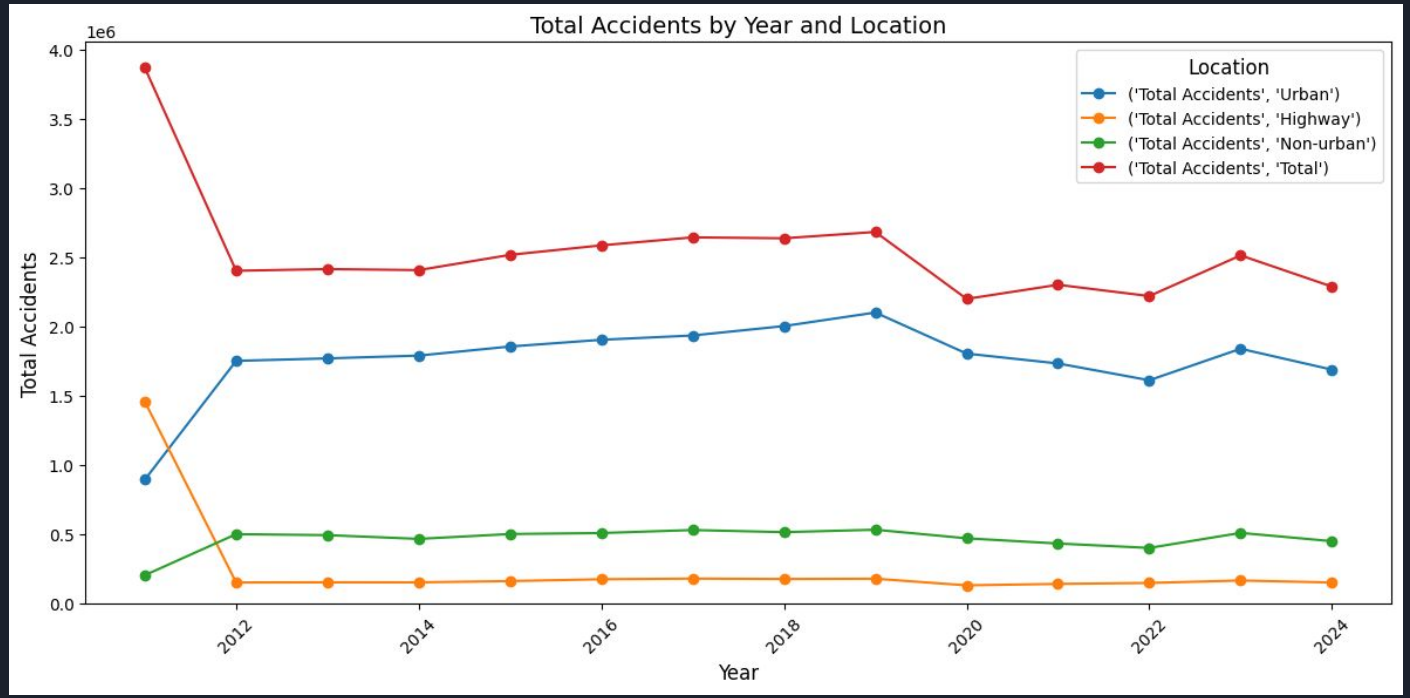
Statistics of road traffic accidents

Germany

Accidents (recorded by the police) (number)

Accident category	Area	2011							
		January	February	March	April	May	June	July	August
Personal injury accidents	Inside built-up areas	10,595	11,008	15,402	18,853	22,913	19,974	19,058	19,584
	Outside built-up areas (excl. motorways/freeways)	4,995	4,405	5,353	6,791	7,236	7,137	7,014	7,274
	On motorways/freeways	1,250	1,166	1,314	1,460	1,586	1,688	1,668	1,725
	Total	16,840	16,579	22,069	27,104	31,735	28,799	27,740	28,583
Serious acc. invol. mat.damage in the nearer sense	Inside built-up areas	3,893	3,548	3,871	3,701	3,894	3,806	3,900	3,713
	Outside built-up areas (excl. motorways/freeways)	2,137	1,728	1,462	1,409	1,549	1,604	1,656	1,532
	On motorways/freeways	1,094	925	651	707	758	1,024	1,139	1,075
	Total	7,124	6,201	5,984	5,817	6,201	6,434	6,695	6,320
Other accidents involving intoxication	Inside built-up areas	970	889	1,000	1,110	1,154	1,242	1,245	1,252
	Outside built-up areas (excl. motorways/freeways)	181	164	126	135	141	192	163	161
	On motorways/freeways	45	37	45	53	56	59	59	58
	Total	1,196	1,090	1,171	1,298	1,351	1,493	1,467	1,471
Other accidents involving material damage	Inside built-up areas	114,745	105,064	120,772	120,344	128,768	119,026	120,585	119,769
	Outside built-up areas (excl. motorways/freeways)	31,161	26,721	26,923	30,040	33,007	27,642	29,794	27,814
	On motorways/freeways	8,985	7,995	9,046	9,845	10,747	10,405	10,885	10,566
	Total	154,891	139,780	156,741	160,229	172,522	157,073	161,264	158,149
Total	Inside built-up areas	130,203	120,509	141,045	144,008	156,729	144,048	144,788	144,318
	Outside built-up areas (excl. motorways/freeways)	38,474	33,018	33,864	38,375	41,933	36,575	38,627	36,781
	On motorways/freeways	11,374	10,123	11,056	12,065	13,147	13,176	13,751	13,424
	Total	180,051	163,650	185,965	194,448	211,809	193,799	197,166	194,523

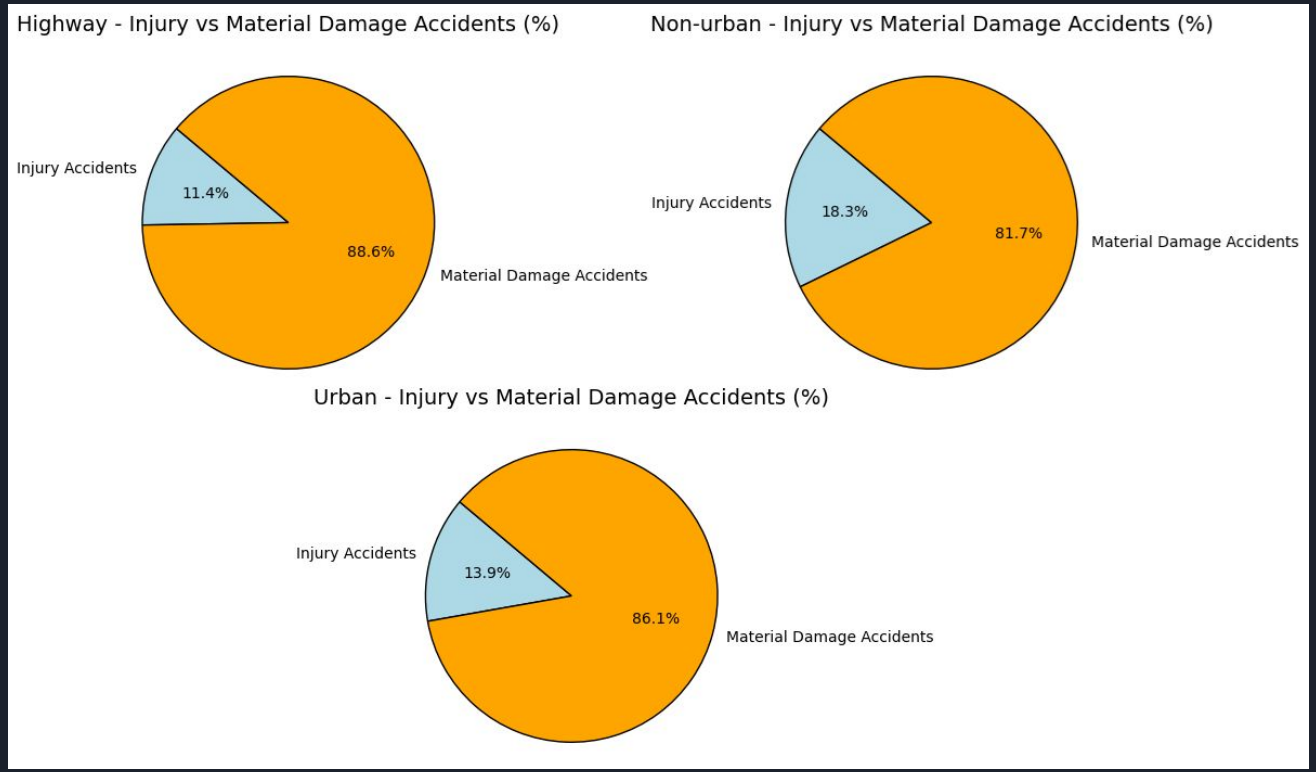
EDA


Total Accidents by Year and Location
1e6
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Total Accidents
2012
2014
2016
2018
2020
2022
2024
Year
Location
('Total Accidents', 'Urban')
('Total Accidents', 'Highway')
('Total Accidents', 'Non-urban')
('Total Accidents', 'Total')

EDA

- Highway and urban are constant
- Urban accidents slowly increasing

EDA - Accident Types



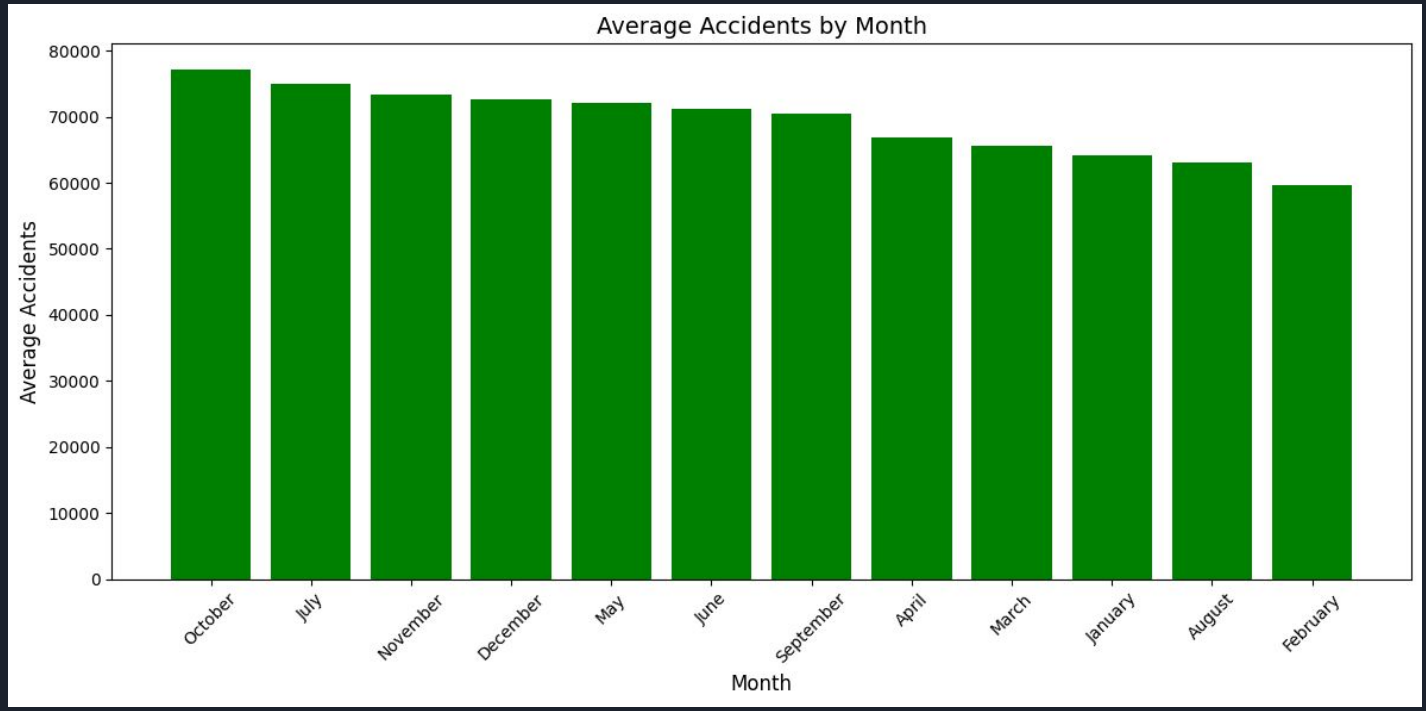
EDA - Accident Types

- Highest amount of personal injuries in non urban areas
- Highway and urban areas show higher material damage

Possibly related to

- Fewer traffic controls
- Lack of lane barriers

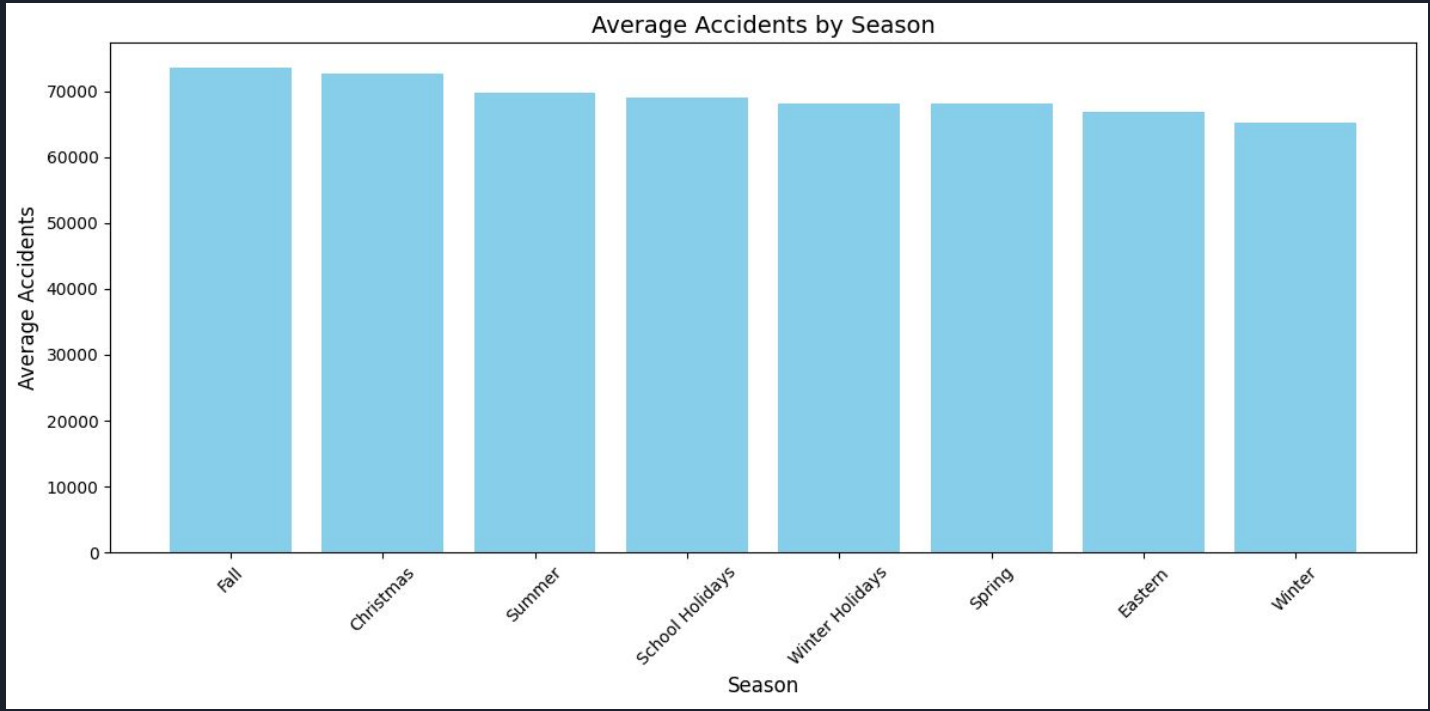
EDA - Seasonality



EDA - Seasonality

- October has the highest chance of an accident
- February lowest chance of an accident happening
- Second half of the year has a higher chance of accidents happening

EDA - Seasonality



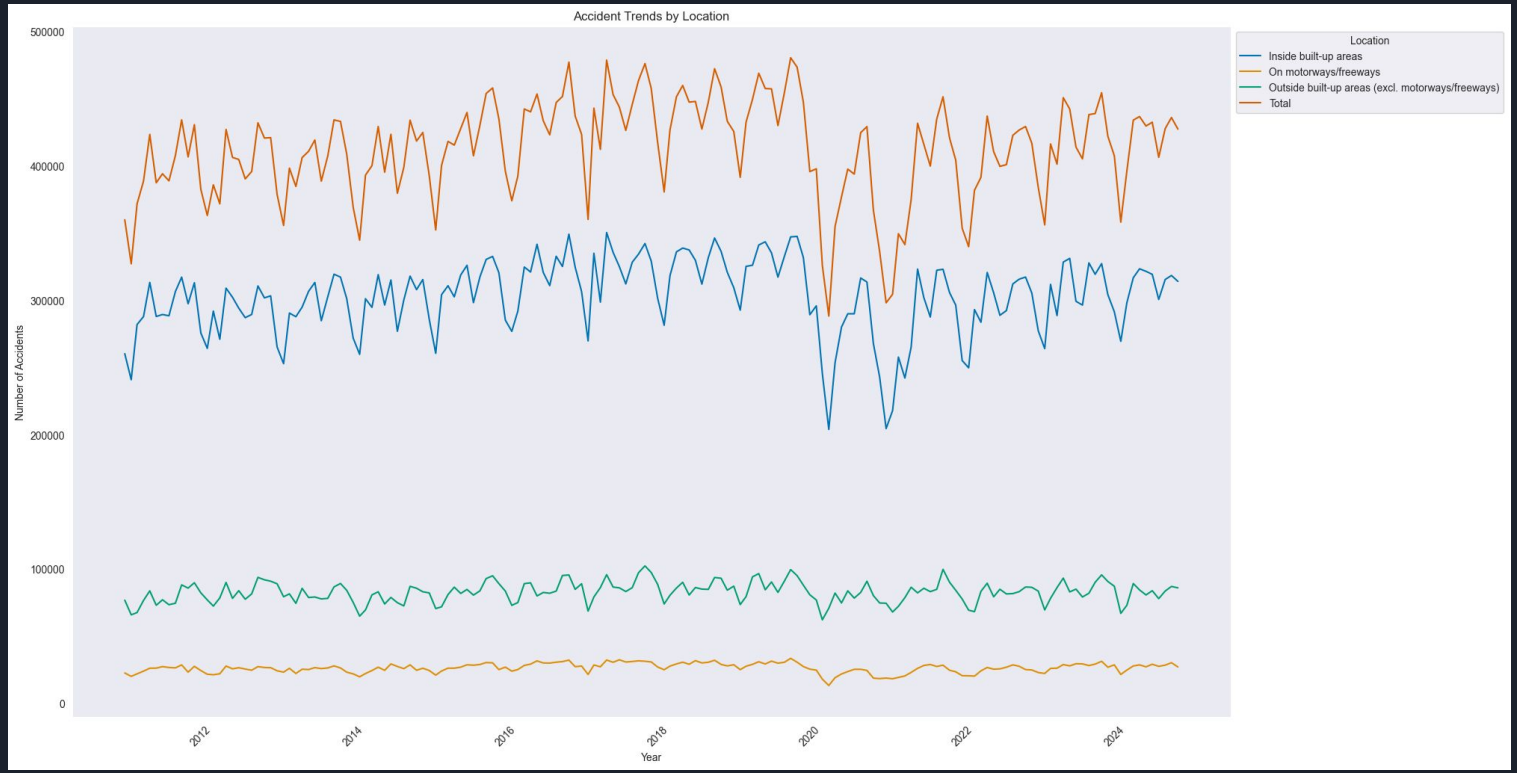
EDA - Seasonality

- Highest number of accidents during fall
- Winter has the lowest number of accidents
- Christmas second highest chance
- No big difference between summer and winter holidays

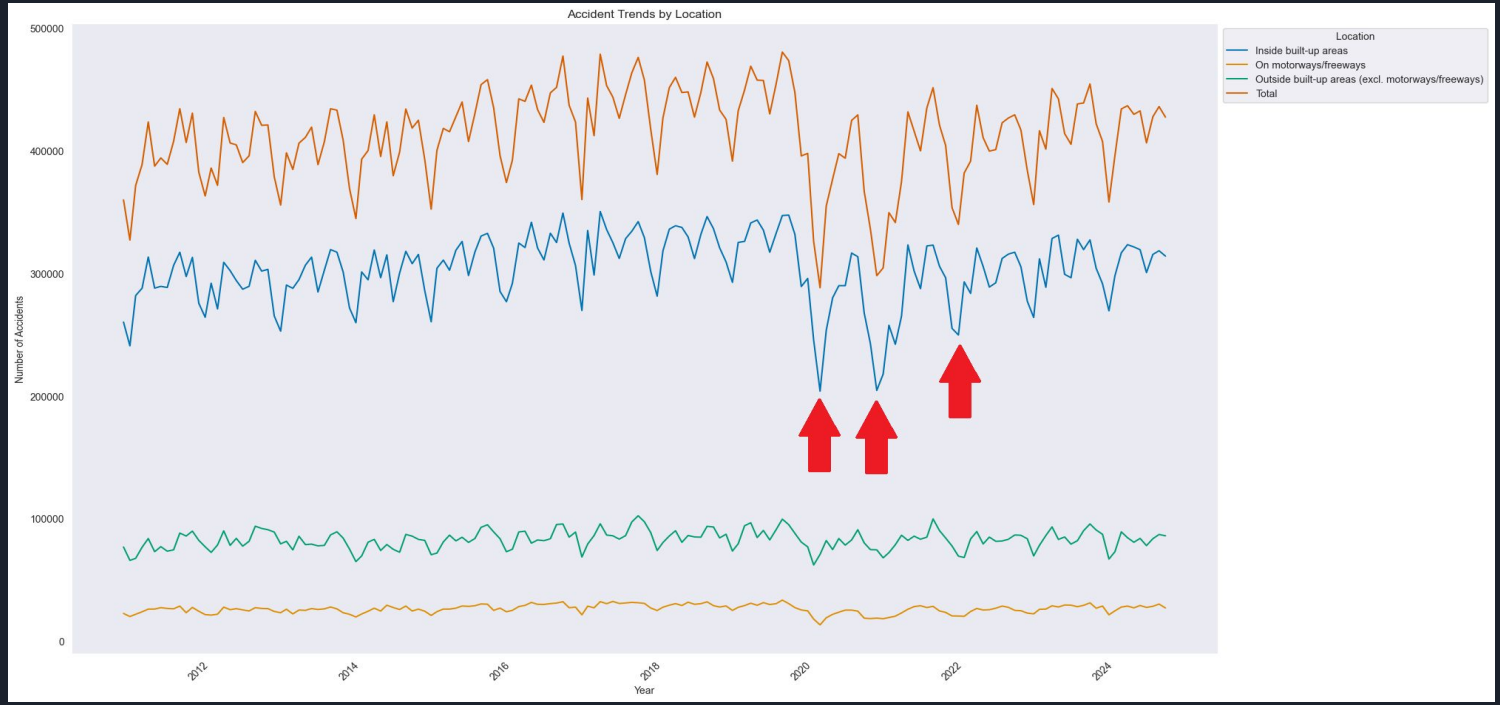
Machine Learning

- Time Series Analysis
- Comparison pre- and post-Covid
- Prediction for 2025 and 2026

Time Series Analysis



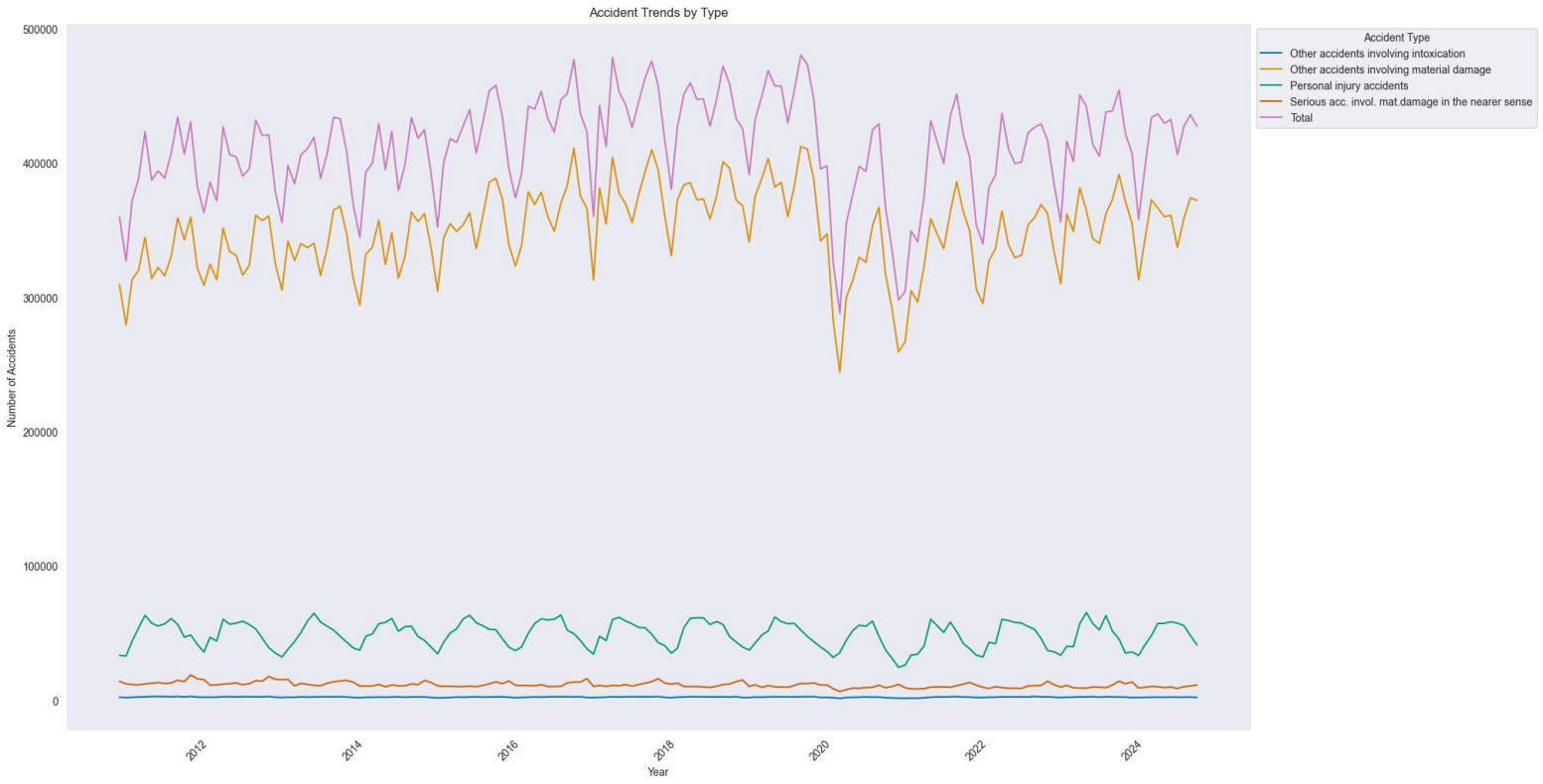
Time Series Analysis



Time Series Analysis

- Accidents slowly increase over time
- Drop in Accidents during the Covid Lockdowns
- Only built up areas affected

Time Series Analysis



Time Series Analysis

- Lesser personal injuries
- Lesser material damage
- Lesser accidents involving intoxication
- Accidents are slowly nearing pre covid times again

Takeaways

- Most accidents happen in built up areas
- Even with lower traffic accidents outside of cities stay constant
- The pandemic had a big influence on total number of accidents
- Most accidents happen over christmas
- Most personal injuries happen outside of cities

Recommendations

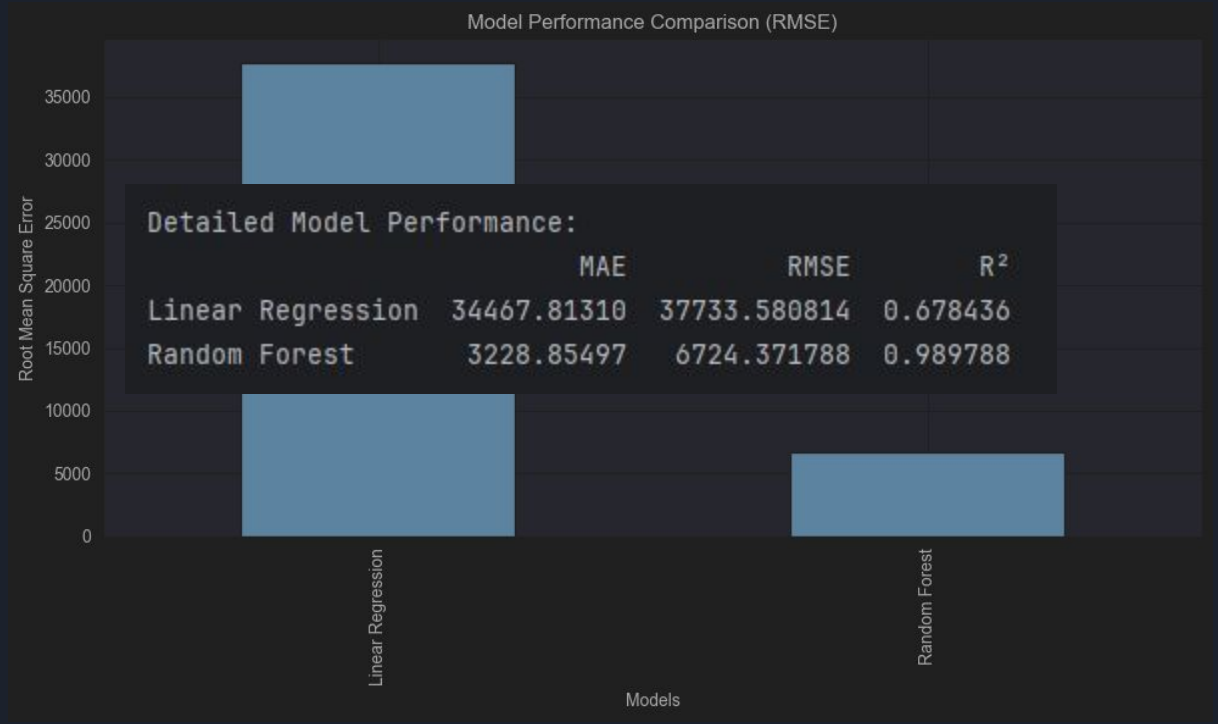
- Enforce more traffic checks outside of cities
- Better education in school for traffic inside cities
- Stricter speed limit outside of cities
- Build more lane limits/barriers outside of cities

Predicting 2025 and 2026

- Decided against using ARIMA

  -> Data not stationary, even after multiple attempts
- Compared Random Forest and Linear Regression
- Random Forest performed better

Model Comparison



Detailed Model Performance:

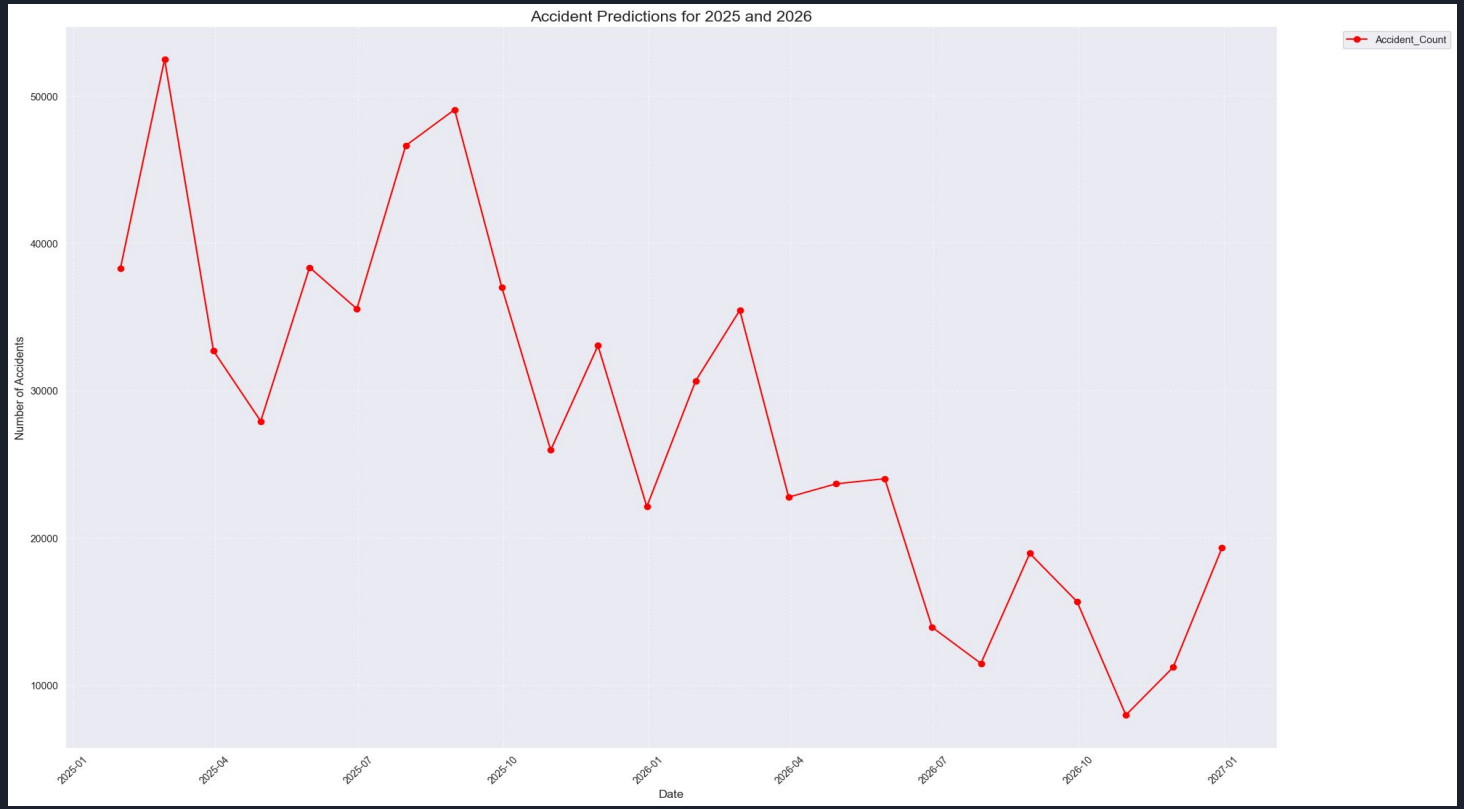
	MAE	RMSE	R^2
Linear Regression	34467.81310	37733.580814	0.678436
Random Forest	3228.85497	6724.371788	0.989788

Model Comparison

- Lower Mean Absolute Error
- Lower Root Mean Square Error
- Higher Coefficient of Determination

-> Random Forest is the superior model

Predicting 2025 and 2026



Predicting 2025 and 2026

- 2025 is predicted to be close to the same as 2023
- 2026 shows a significant drop
  - Humans will drive better/less
  - Not enough data
  - New pandemic predicted

Questions?

Code Snippets

```python
# Load the CSV file
df = pd.read_csv("data/46241-0002_en_flat.csv", delimiter=";")
print(df.isnull().sum())

# Select relevant columns and rename them for clarity
df_cleaned = df.loc[:, [
    "time",  # Year
    "1_variable_attribute_label",  # Month
    "4_variable_attribute_label",  # Accident location
    "3_variable_attribute_label",  # Accident type
    "value"  # Number of accidents
]].rename(columns={
    "time": "Year",
    "1_variable_attribute_label": "Month",
    "4_variable_attribute_label": "Location",
    "3_variable_attribute_label": "Accident_Type",
    "value": "Accident_Count"
}).copy()  # Ensure we work on a separate DataFrame

# Forward-fill missing year values
df_cleaned["Year"] = df_cleaned["Year"].ffill()
```

Code Snippets

```python
# Combine "Year" and "Month" into a single "Date" column
month_mapping = {
    "Januar": "January", "Februar": "February", "März": "March", "April": "April",
    "Mai": "May", "Juni": "June", "Juli": "July", "August": "August",
    "September": "September", "Oktober": "October", "November": "November", "Dezember": "December"
}
df_cleaned["Month"] = df_cleaned["Month"].replace(month_mapping)

# Create a new Date column and convert it to datetime format
df_cleaned["Date"] = pd.to_datetime(df_cleaned["Year"].astype(str) + "-" + df_cleaned["Month"], format="%Y-%B", errors="coerce")

# Categorize accident types into broader categories
def categorize_accident(acc_type):
    if "Personal injury" in acc_type:
        return "Personal Injury"
    if "mat.damage" in acc_type:
        return "Material Damage"
    if "intoxication" in acc_type:
        return "Intoxication"
    if "Other accidents involving material damage" in acc_type:
        return "Other Material Damage"
    if "Total" in acc_type:
        return "Total"
    return "Unknown"

df_cleaned["Accident_Category"] = df_cleaned["Accident_Type"].apply(categorize_accident)

# Drop original "Year" and "Month" columns, since "Date" now replaces them
df_cleaned = df_cleaned.drop(columns=["Year", "Month"])

# Save the cleaned dataset to a new CSV file
df_cleaned.to_csv("cleaned_traffic_accidents_with_separate_year.csv", index=True)
```

Code Snippets

```python
def train_and_evaluate_models(X, y, preprocessor):
    """
    Train multiple regression models and evaluate their performance.
    """
    # Split data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Define models with preprocessing pipeline
    models = {
        "Linear Regression": Pipeline([
            ('preprocessor', preprocessor),
            ('regressor', LinearRegression())
        ]),
        "Random Forest": Pipeline([
            ('preprocessor', preprocessor),
            ('regressor', RandomForestRegressor(n_estimators=100, random_state=42))
        ])
    }

    # Train and evaluate models
    results = {}
    for name, model in models.items():
        # Fit the model
        model.fit(X_train, y_train)

        # Make predictions
        y_pred = model.predict(X_test)

        # Calculate performance metrics
        results[name] = {
            "MAE": mean_absolute_error(y_test, y_pred),
            "RMSE": np.sqrt(mean_squared_error(y_test, y_pred)),
            "R²": r2_score(y_test, y_pred)
        }

    return results
```

```
Detailed Model Performance:
                            MAE          RMSE        R²
Linear Regression  34467.81310  37733.580814  0.678436
Random Forest       3228.85497   6724.371788  0.989788
```

Code Snippets

```python
def train_random_forest_models(features):
    """Trains a Random Forest model for each feature column."""
    models = {}

    for column, X, y in features:
        if len(X) == 0:  # Skip if not enough data
            continue

        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

        scaler = StandardScaler()
        X_train_scaled = scaler.fit_transform(X_train)
        X_test_scaled = scaler.transform(X_test)

        rf = RandomForestRegressor(n_estimators=100, random_state=42)
        rf.fit(X_train_scaled, y_train)

        y_pred = rf.predict(X_test_scaled)

        models[column] = {
            'model': rf,
            'scaler': scaler,
            'metrics': {
                'MAE': mean_absolute_error(y_test, y_pred),
                'MSE': mean_squared_error(y_test, y_pred),
                'R2': r2_score(y_test, y_pred)
            }
        }

    return models
```


```python
def train_random_forest_models(features):
    """Trains a Random Forest model for each feature column."""
    models = {}

    for column, X, y in features:
        if len(X) == 0:  # Skip if not enough data
            continue

        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

        scaler = StandardScaler()
        X_train_scaled = scaler.fit_transform(X_train)
        X_test_scaled = scaler.transform(X_test)

        rf = RandomForestRegressor(n_estimators=100, random_state=42)
        rf.fit(X_train_scaled, y_train)

        y_pred = rf.predict(X_test_scaled)

        models[column] = {
            'model': rf,
            'scaler': scaler,
            'metrics': {
                'MAE': mean_absolute_error(y_test, y_pred),
                'MSE': mean_squared_error(y_test, y_pred),
                'R2': r2_score(y_test, y_pred)
            }
        }

    return models
```