03_ukraine_missiles_final

December 11, 2025

1 Ukraine missiles dataset

De **Massive Missile Attacks on Ukraine dataset** bevat gedetailleerde informatie over gelanceerde en neergeschoten raketten en drones tijdens Russische massale aanvallen op Oekraïne.

1.1 Data inladen en basisverkenning

[5]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import folium
from folium.plugins import HeatMap
import ipywidgets as widgets
from IPython.display import display, clear_output
from pyspark.sql import SparkSession
from pyspark.sql import functions as F
from pyspark.sql.functions import (
    col, year, month, date_format, to_date,
    sum as spark_sum, avg, count, round as spark_round
)
```

[6]:
```
# 1. PySpark sessie en data inladen

spark = SparkSession.builder.appName("UkraineMissileAttacks").getOrCreate()
sns.set_theme(style="whitegrid")
```

[7]:
```
csv_file = "/home/jovyan/work/drone-project/missile_attacks_daily.csv"
df_missiles = spark.read.csv(csv_file, header=True, inferSchema=True)

print("Schema van de Ukraine Missile Attacks dataset:")
df_missiles.printSchema()
df_missiles.show(5, truncate=False)
```

```
Schema van de Ukraine Missile Attacks dataset:
root
 |-- time_start: timestamp (nullable = true)
 |-- time_end: timestamp (nullable = true)
 |-- model: string (nullable = true)
```

```
 |-- launch_place: string (nullable = true)
 |-- target: string (nullable = true)
 |-- target_main: string (nullable = true)
 |-- launched: double (nullable = true)
 |-- destroyed: double (nullable = true)
 |-- not_reach_goal: double (nullable = true)
 |-- still_attacking: double (nullable = true)
 |-- border_crossing: string (nullable = true)
 |-- is_shahed: double (nullable = true)
 |-- num_hit_location: double (nullable = true)
 |-- num_fall_fragment_location: double (nullable = true)
 |-- carrier: string (nullable = true)
 |-- turbojet: double (nullable = true)
 |-- turbojet_destroyed: double (nullable = true)
 |-- affected region: string (nullable = true)
 |-- destroyed_details: string (nullable = true)
 |-- launched_details: string (nullable = true)
 |-- launch_place_details: string (nullable = true)
 |-- source: string (nullable = true)

+-------------------+-------------------+---------------+----------------------
----------------------------------------------------------------------------
--+-------+-----------+--------+---------+--------------+---------------+-------
--------+---------+----------------+-------------------------+-------+--------
+------------------+-----------------------------------------------------------
----------------------------------------------------------------------------
----------+-------------------------------------------------------------------
+----------------+--------------------+---------------------------------------
-----------------------------------------+
|time_start         |time_end           |model           |launch_place
|target |target_main|launched|destroyed|not_reach_goal|still_attacking|border_cr
ossing|is_shahed|num_hit_location|num_fall_fragment_location|carrier|turbojet|tu
rbojet_destroyed|affected region
|destroyed_details
|launched_details|launch_place_details|source
|
+-------------------+-------------------+---------------+----------------------
----------------------------------------------------------------------------
--+-------+-----------+--------+---------+--------------+---------------+-------
--------+---------+----------------+-------------------------+-------+--------
+------------------+-----------------------------------------------------------
----------------------------------------------------------------------------
----------+-------------------------------------------------------------------
+----------------+--------------------+---------------------------------------
-----------------------------------------+
|2025-12-06 18:00:00|2025-12-07 09:00:00|X-47 Kinzhal    |Tambov oblast
|Ukraine|NULL       |3.0     |2.0      |NULL          |NULL           |{}
|NULL     |NULL            |NULL                      |MiG-31K|NULL    |NULL
```

```
|NULL
|NULL                                                                    |NULL
|NULL                |kpszsu/posts/pfbid0d2Rx2p1iSxS4i874x54NCZzg9fcsBvdA5SGa8s5
3UAmBLQsePqC2TyKm8KhpvAARl|
|2025-12-06 18:00:00|2025-12-07 09:00:00|Shahed-136/131  |Primorsko-Akhtarsk and
Chauda, Crimea and Kursk oblast and Oryol oblast and Millerovo
|Ukraine|NULL       |241.0   |175.0    |0.0           |NULL           |{}
|150.0    |14.0            |0.0                        |NULL   |NULL    |NULL
|NULL
|{'south': 8, 'north': NaN, 'east': NaN}                                 |NULL
|NULL                |kpszsu/posts/pfbid0d2Rx2p1iSxS4i874x54NCZzg9fcsBvdA5SGa8s5
3UAmBLQsePqC2TyKm8KhpvAARl|
|2025-12-06 18:00:00|2025-12-07 09:00:00|Iskander-M/KN-23|Kursk oblast
|Ukraine|NULL       |2.0     |2.0      |NULL          |NULL           |{}
|NULL     |NULL            |NULL                       |NULL   |NULL    |NULL
|NULL
|NULL                                                                    |NULL
|NULL                |kpszsu/posts/pfbid0d2Rx2p1iSxS4i874x54NCZzg9fcsBvdA5SGa8s5
3UAmBLQsePqC2TyKm8KhpvAARl|
|2025-12-05 18:00:00|2025-12-06 10:00:00|Shahed-136/131  |Primorsko-Akhtarsk and
Chauda, Crimea and Kursk oblast and Bryansk oblast and Oryol oblast and
Millerovo|Ukraine|NULL       |653.0   |585.0    |0.0           |NULL
|{}             |300.0    |29.0            |3.0                        |NULL
|NULL    |NULL              |['Lviv oblast', 'Ivano-Frankivsk oblast', 'Lutsk
oblast', 'Kyiv oblast', 'Odesa oblast', 'Dnipropetrovsk oblast', 'Donetsk
oblast', 'Chernihiv oblast']|{'west': 39, 'south': 28, 'north': NaN, 'east':
NaN, 'center': NaN}|NULL            |NULL                |kpszsu/posts/pfbid08V2
L3Y6pPBmVG3bXDkQWysJ4fhRpdpuUoPaD3BmYkJ5d4GWwKjzCjys9NmEZ7vPNl|
|2025-12-05 18:00:00|2025-12-06 10:00:00|Iskander-M/KN-23|Crimea and Bryansk
oblast and Rostov oblast and Krasnodar Krai
|Ukraine|NULL       |14.0    |1.0      |NULL          |NULL           |{}
|NULL     |NULL            |NULL                       |NULL   |NULL    |NULL
|NULL
|NULL                                                                    |NULL
|NULL                |kpszsu/posts/pfbid08V2L3Y6pPBmVG3bXDkQWysJ4fhRpdpuUoPaD3Bm
YkJ5d4GWwKjzCjys9NmEZ7vPNl|
+--------------------+--------------------+---------------+----------------------
------------------------------------------------------------------------------
--+-------+-----------+--------+---------+-------------+--------------+-------
--------+---------+---------------+-------------------------+-------+--------
+------------------+-----------------------------------------------------------------
------------------------------------------------------------------------------
-----------+-------------------------------------------------------------------
+---------------+--------------------+---------------------------------------
-----------------------------------------+
only showing top 5 rows
```

1.2 Data cleaning

[8]:
```
# --- BLOK 2: DATA CLEANING & TIJD-KOLOMMEN ---

# Datumkolommen omzetten + jaar/maand toevoegen
df_clean = (df_missiles
            .withColumn("date", to_date(col("time_start")))
            .withColumn("year", year(col("time_start")))
            .withColumn("month", month(col("time_start")))
            .withColumn("year_month", date_format(col("time_start"),␣
 ↪"yyyy-MM")))

# Kolommen trimmen en NULLs opvullen voor numerieke kolommen
numeric_cols = ["launched", "destroyed", "not_reach_goal", "is_shahed",
                "turbojet", "turbojet_destroyed"]

for c in numeric_cols:
    if c in df_clean.columns:
        df_clean = df_clean.fillna(0.0, subset=[c])

# Controleer
print(f"Aantal geldige rijen: {df_clean.count()}")
df_clean.select("date", "year", "year_month", "model", "launched", "destroyed").
 ↪show(10)
```

```
Aantal geldige rijen: 3174
+----------+----+----------+--------------------+--------+---------+
|      date|year|year_month|               model|launched|destroyed|
+----------+----+----------+--------------------+--------+---------+
|2025-12-06|2025|   2025-12|        X-47 Kinzhal|     3.0|      2.0|
|2025-12-06|2025|   2025-12|      Shahed-136/131|   241.0|    175.0|
|2025-12-06|2025|   2025-12|    Iskander-M/KN-23|     2.0|      2.0|
|2025-12-05|2025|   2025-12|      Shahed-136/131|   653.0|    585.0|
|2025-12-05|2025|   2025-12|    Iskander-M/KN-23|    14.0|      1.0|
|2025-12-05|2025|   2025-12|X-101/X-555 and K…|    34.0|     29.0|
|2025-12-06|2025|   2025-12|        X-47 Kinzhal|     3.0|      0.0|
|2025-12-06|2025|   2025-12|                   |     5.0|      5.0|
|2025-12-04|2025|   2025-12|      Shahed-136/131|   137.0|     80.0|
|2025-12-05|2025|   2025-12|                   |     5.0|      5.0|
+----------+----+----------+--------------------+--------+---------+
only showing top 10 rows
```

1.3 Data Aggregaties

1.3.1 Totaal gelanceerde vs neergeschoten per dag

[9]:
```
# --- AGGREGATIE 1: Dagelijkse totalen

daily_agg = (df_clean
             .groupBy("date")
             .agg(
                 spark_sum("launched").alias("total_launched"),
                 spark_sum("destroyed").alias("total_destroyed")
             )
             .withColumn("success_rate",
                         spark_round(col("total_destroyed") /␣
 ↪col("total_launched"), 3))
             .orderBy("date"))

daily_agg.show(10)
```

```
+----------+--------------+---------------+------------+
|      date|total_launched|total_destroyed|success_rate|
+----------+--------------+---------------+------------+
|2022-09-28|           8.0|            7.0|       0.875|
|2022-09-29|           7.0|            5.0|       0.714|
|2022-09-30|           3.0|            3.0|         1.0|
|2022-10-01|           1.0|            1.0|         1.0|
|2022-10-02|           7.0|            5.0|       0.714|
|2022-10-05|           8.0|            8.0|         1.0|
|2022-10-06|          17.0|           17.0|         1.0|
|2022-10-07|           3.0|            3.0|         1.0|
|2022-10-08|           3.0|            3.0|         1.0|
|2022-10-09|          22.0|            1.0|       0.045|
+----------+--------------+---------------+------------+
only showing top 10 rows
```

1.3.2 Success rate per model

[10]:
```
# --- AGGREGATIE 2: Success-rate per wapenmodel (cruise, ballistic, drones) ---

weapon_agg = (df_clean
              .groupBy("model")
              .agg(
                  spark_sum("launched").alias("total_launched"),
                  spark_sum("destroyed").alias("total_destroyed")
              )
              .withColumn("success_rate",
                          spark_round(col("total_destroyed") /␣
 ↪col("total_launched"), 3))
```

```
                .filter(col("total_launched") > 5)  # filter rare wapens eruit
                .orderBy(col("total_launched").desc()))

weapon_agg.show(20)
```

```
+--------------------+--------------+---------------+------------+
|               model|total_launched|total_destroyed|success_rate|
+--------------------+--------------+---------------+------------+
|      Shahed-136/131|       65488.0|        44384.0|       0.678|
|         X-101/X-555|        1808.0|         1535.0|       0.849|
|         Unknown UAV|         855.0|          841.0|       0.984|
|X-101/X-555 and K…|         643.0|          508.0|        0.79|
|              Kalibr|         464.0|          368.0|       0.793|
|          Iskander-M|         411.0|           69.0|       0.168|
|              Lancet|         355.0|          355.0|         1.0|
|    Iskander-M/KN-23|         352.0|           88.0|        0.25|
|               C-300|         345.0|            2.0|       0.006|
|  Reconnaissance UAV|         335.0|          334.0|       0.997|
|                ZALA|         309.0|          308.0|       0.997|
|           X-59/X-69|         274.0|          181.0|       0.661|
|                X-59|         269.0|          166.0|       0.617|
|            Orlan-10|         264.0|          263.0|       0.996|
|            Supercam|         252.0|          251.0|       0.996|
|                    |         231.0|          231.0|         1.0|
|        X-47 Kinzhal|         212.0|           64.0|       0.302|
|          Iskander-K|         187.0|          117.0|       0.626|
|                X-22|         151.0|            3.0|        0.02|
|         C-300/C-400|         141.0|            0.0|         0.0|
+--------------------+--------------+---------------+------------+
only showing top 20 rows
```

[11]:
```
# --- AGGREGATIE 3: Shahed-drones vs andere ---

shahed_agg = (df_clean
              .groupBy("is_shahed")
              .agg(
                  spark_sum("launched").alias("total_launched"),
                  spark_sum("destroyed").alias("total_destroyed"),
                  count("*").alias("num_attacks")
              )
              .withColumn("success_rate",
                          spark_round(col("total_destroyed") /␣
 ↪col("total_launched"), 3)))

shahed_agg.show()
```

```
+---------+--------------+---------------+-----------+------------+
|is_shahed|total_launched|total_destroyed|num_attacks|success_rate|
```

```
+---------+-------------+--------------+-----------+------------+
|     70.0|         888.0|         683.0|          8|       0.769|
|      0.0|       59410.0|       38677.0|       3085|       0.651|
|    255.0|         400.0|         198.0|          1|       0.495|
|    350.0|         596.0|         558.0|          1|       0.936|
|     29.0|          35.0|          11.0|          1|       0.314|
|     75.0|         122.0|         104.0|          1|       0.852|
|     35.0|          62.0|          39.0|          1|       0.629|
|     80.0|        1931.0|        1426.0|         14|       0.738|
|    250.0|        1780.0|        1596.0|          4|       0.897|
|    300.0|        2017.0|        1838.0|          4|       0.911|
|     25.0|          96.0|          78.0|          2|       0.813|
|     85.0|         138.0|         114.0|          1|       0.826|
|    140.0|         223.0|         206.0|          1|       0.924|
|     50.0|        1003.0|         770.0|         12|       0.768|
|    400.0|         653.0|         592.0|          1|       0.907|
|     60.0|         521.0|         414.0|          5|       0.795|
|     59.0|          64.0|          36.0|          1|       0.563|
|    200.0|        1396.0|        1051.0|          4|       0.753|
|     40.0|         469.0|         328.0|          7|       0.699|
|     30.0|          54.0|          33.0|          1|       0.611|
+---------+-------------+--------------+-----------+------------+
only showing top 20 rows
```

1.3.3 Massale aanvallen (meer dan 50 projectielen)

[13]:
```
# --- SQL QUERY: Massale aanvallen (zoals jouw "Civilian Ratio per Land" SQL␣
  ↪query) ---

df_clean.createOrReplaceTempView("missiles")

massive_attacks = spark.sql("""
    SELECT date, year_month, model, launched, destroyed,
           `affected region`, target_main,
           ROUND(destroyed / launched, 3) AS success_rate
    FROM missiles
    WHERE launched > 50
    ORDER BY launched DESC
""")

massive_attacks.show(15, truncate=False)
```

```
+----------+----------+-------------+--------+---------+-----------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
-------------------------------------------------------------------------------+-------
-----------------------------------+------------+
|date      |year_month|model        |launched|destroyed|affected region
```

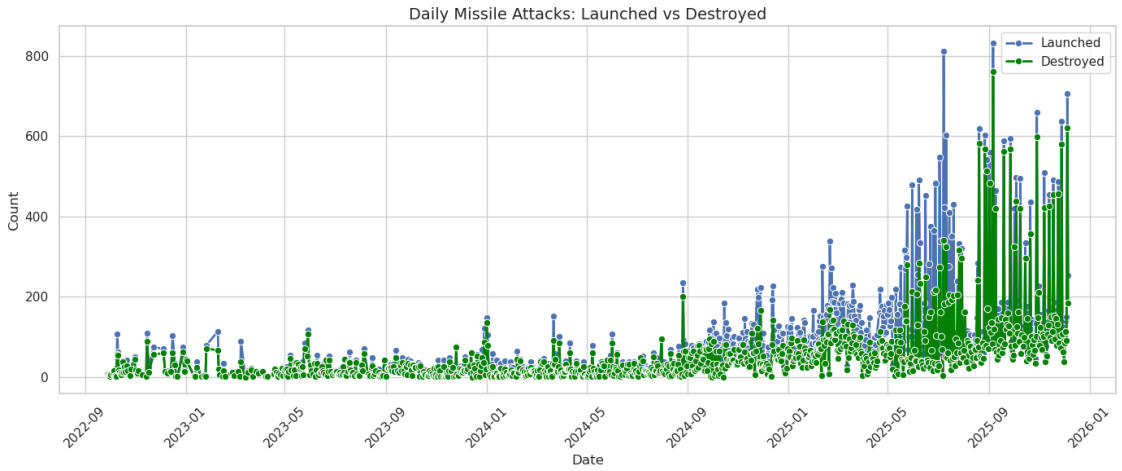
```
|target_main                                    |success_rate|
+----------+----------+--------------+--------+---------+-----------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------+-------
-------------------------------------+------------+
|2025-09-06|2025-09   |Shahed-136/131|810.0   |747.0    |['Kyiv oblast',
'Dnipropetrovsk oblast', 'Zaporizhzhia oblast', 'Poltava oblast', 'Odesa
oblast', 'Khmelnytskyi oblast', 'Kharkiv oblast', 'Zhytomyr oblast', 'Sumy
oblast']
|NULL                                           |0.922       |
|2025-07-08|2025-07   |Shahed-136/131|728.0   |296.0    |NULL
|Lutsk                                          |0.407       |
|2025-12-05|2025-12   |Shahed-136/131|653.0   |585.0    |['Lviv oblast', 'Ivano-
Frankivsk oblast', 'Lutsk oblast', 'Kyiv oblast', 'Odesa oblast',
'Dnipropetrovsk oblast', 'Donetsk oblast', 'Chernihiv oblast']
|NULL                                           |0.896       |
|2025-10-29|2025-10   |Shahed-136/131|653.0   |592.0    |['Lviv oblast', 'Ivano-
Frankivsk oblast', 'Zaporizhzhia oblast', 'Dnipropetrovsk oblast', 'Donetsk
oblast', 'Vinnytsia oblast', 'Khmelnytskyi oblast', 'Odesa oblast', 'Sumy
oblast', 'Chernihiv oblast', 'Poltava oblast', 'Kirovohrad oblast', 'Kyiv
oblast']|NULL                                                 |0.907       |
|2025-08-27|2025-08   |Shahed-136/131|598.0   |563.0    |['Kyiv oblast',
'Zaporizhzhia oblast', 'Khmelnytskyi oblast', 'Vinnytsia oblast', 'Chernihiv
oblast']
|Kyiv oblast                                    |0.941       |
|2025-07-11|2025-07   |Shahed-136/131|597.0   |319.0    |['Lviv oblast', 'Volyn
oblast', 'Chernivtsi oblast', 'Kyiv oblast', 'Kharkiv oblast']
|NULL                                           |0.534       |
|2025-11-28|2025-11   |Shahed-136/131|596.0   |558.0    |NULL
|Kyiv oblast                                    |0.936       |
|2025-09-27|2025-09   |Shahed-136/131|593.0   |566.0    |['Kyiv oblast', 'Sumy
oblast', 'Donetsk oblast', 'Odesa oblast']
|Kyiv                                           |0.954       |
|2025-09-19|2025-09   |Shahed-136/131|579.0   |552.0    |['Kyiv oblast',
'Dnipropetrovsk oblast', 'Donetsk oblast', 'Zaporizhzhia oblast', 'Odesa
oblast', 'Mykolaiv oblast', 'Kirovohrad oblast', 'Chernihiv oblast']
|NULL                                           |0.953       |
|2025-08-20|2025-08   |Shahed-136/131|574.0   |546.0    |['Lviv oblast', 'Rivne
oblast', 'Volyn oblast', 'Zakarpattia oblast', 'Zaporizhzhia oblast',
'Dnipropetrovsk oblast', 'Donetsk oblast', 'Sumy oblast']
|NULL                                           |0.951       |
|2025-07-03|2025-07   |Shahed-136/131|539.0   |268.0    |['Kyiv', 'Kyiv oblast']
|Kyiv                                           |0.497       |
|2025-08-29|2025-08   |Shahed-136/131|537.0   |510.0    |['Dnipropetrovsk
oblast', 'Zaporizhzhia oblast', 'Khmelnytskyi oblast', 'Volyn oblast', 'Sumy
oblast', 'Kyiv oblast', 'Cherkasy oblast', 'Zhytomyr oblast']
|Dnipropetrovsk oblast and Zaporizhzhia oblast|0.95        |
```

```
|2025-09-02|2025-09   |Shahed-136/131|502.0   |430.0    |['Volyn oblast', 'Lviv
oblast', 'Ivano-Frankivsk oblast', 'Donetsk oblast', 'Kirovohrad oblast',
'Khmelnytskyi oblast', 'Sumy oblast', 'Kyiv oblast', 'Odesa oblast']
|NULL                                          |0.857       |
|2025-10-04|2025-10   |Shahed-136/131|496.0   |439.0    |['Lviv oblast', 'Ivano-
Frankivsk oblast', 'Rivne oblast', 'Odesa oblast', 'Zhytomyr oblast', 'Sumy
oblast', 'Kharkiv oblast', 'Chernihiv oblast', 'Zaporizhzhia oblast']
|Lviv oblast                                   |0.885       |
|2025-06-08|2025-06   |Shahed-136/131|479.0   |277.0    |NULL
|NULL                                          |0.578       |
+----------+----------+--------------+--------+---------+----------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------+-------
--------------------------------------+------------+
only showing top 15 rows
```

1.4 Data Visualisaties

1.4.1 Tijdsreeks: gelanceerd vs neergeschoten

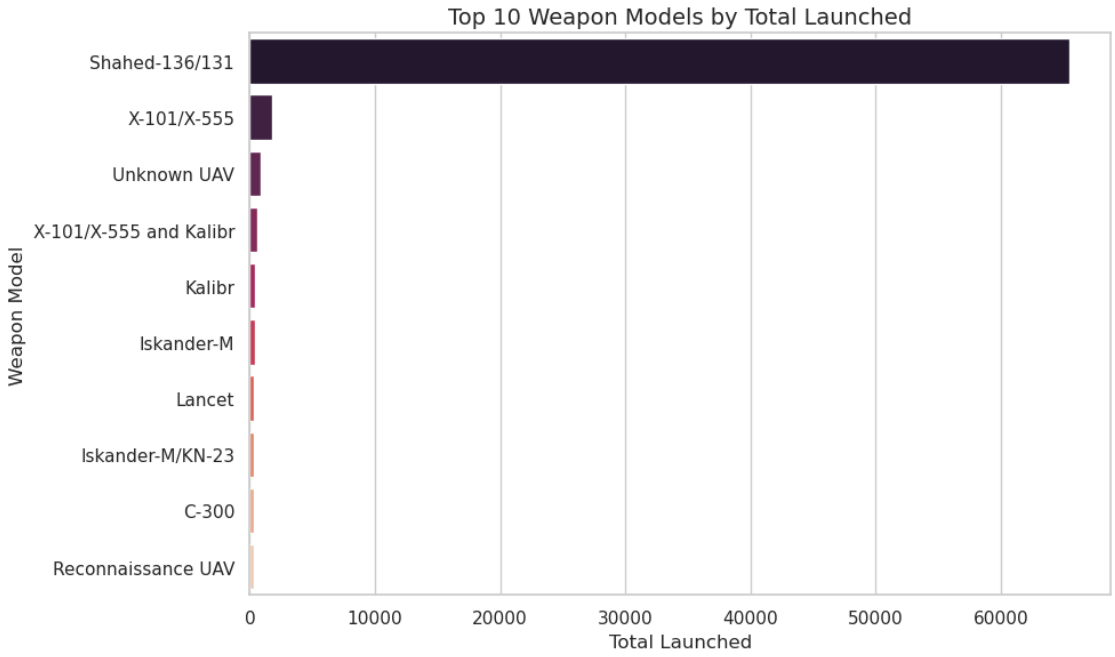
[14]:
```python
# --- VISUALISATIE 1: Tijdreeks launched vs destroyed (zoals "Slachtoffers over tijd per land") ---

pdf_daily = daily_agg.toPandas().set_index("date")

plt.figure(figsize=(14, 6))
sns.lineplot(data=pdf_daily, x=pdf_daily.index, y="total_launched",
             label="Launched", marker='o', linewidth=2)
sns.lineplot(data=pdf_daily, x=pdf_daily.index, y="total_destroyed",
             label="Destroyed", marker='o', linewidth=2, color='green')
plt.title("Daily Missile Attacks: Launched vs Destroyed", fontsize=14)
plt.xlabel("Date")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



1.4.2 Barplot: Aantal aanvallen per wapenmodel

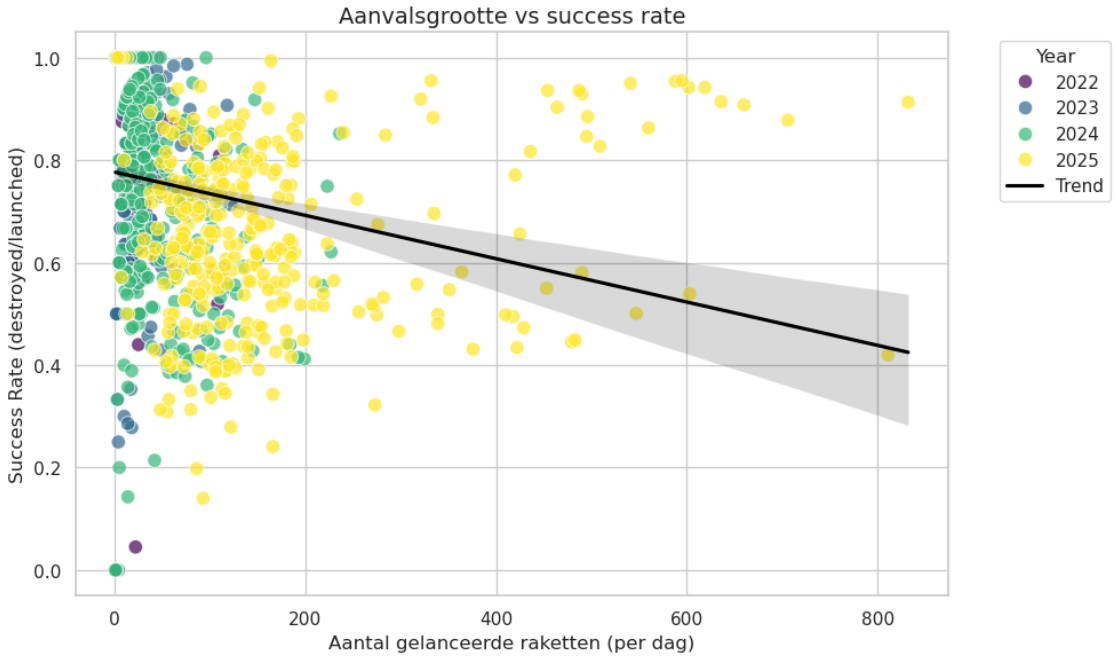
[15]:
```
# --- VISUALISATIE 2: Totaal aantal gelanceerd per wapenmodel (top 10) ---

pdf_weapon = weapon_agg.limit(10).toPandas()

plt.figure(figsize=(10, 6))
sns.barplot(data=pdf_weapon, y="model", x="total_launched", palette="rocket")
plt.title("Top 10 Weapon Models by Total Launched", fontsize=14)
plt.xlabel("Total Launched")
plt.ylabel("Weapon Model")
plt.tight_layout()
plt.show()
```

```
/tmp/ipykernel_63111/88954114.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(data=pdf_weapon, y="model", x="total_launched", palette="rocket")
```



1.4.3 Scatterplot: correlatie aanvalsgrootte vs successrate

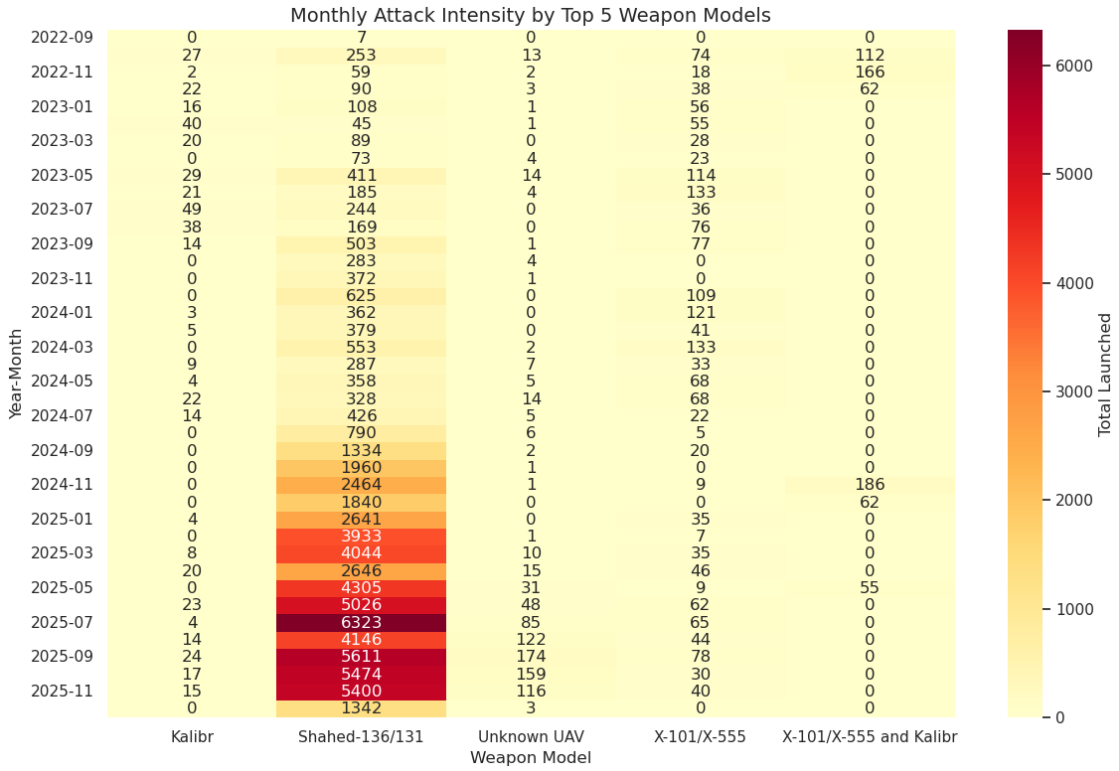
[18]:
```
# --- VISUALISATIE 3: Correlatie tussen aanvalsgrootte en success-rate ---

pdf_scatter = daily_agg.filter(col("total_launched") > 0).toPandas()

# Ensure date column is datetime type
pdf_scatter['date'] = pd.to_datetime(pdf_scatter['date'])

plt.figure(figsize=(10, 6))
sns.scatterplot(data=pdf_scatter, x="total_launched", y="success_rate",
                hue=pdf_scatter["date"].dt.year, palette="viridis", s=80,␣
 ↪alpha=0.7)
sns.regplot(data=pdf_scatter, x="total_launched", y="success_rate",
            scatter=False, color="black", label="Trend")
plt.title("Aanvalsgrootte vs success rate", fontsize=14)
plt.xlabel("Aantal gelanceerde raketten (per dag)")
plt.ylabel("Success Rate (destroyed/launched)")
plt.legend(title="Year", bbox_to_anchor=(1.05, 1), loc="upper left")
plt.tight_layout()
plt.show()
```



1.4.4 Heatmap: intensiteit per maand x wapentype

[19]:
```
# --- VISUALISATIE 4: Heatmap per maand en wapenmodel (top 5 modellen) ---

# Filter top 5 modellen voor leesbaarheid
top5_models = [row["model"] for row in weapon_agg.limit(5).collect()]

heatmap_data = (df_clean
                .filter(col("model").isin(top5_models))
                .groupBy("year_month", "model")
                .agg(spark_sum("launched").alias("total"))
                .orderBy("year_month")
                .toPandas())

# Pivot voor heatmap: rijen = maand, kolommen = model
matrix_heat = heatmap_data.pivot(index="year_month", columns="model",␣
 ↪values="total").fillna(0)

plt.figure(figsize=(12, 8))
sns.heatmap(matrix_heat, annot=True, fmt="g", cmap="YlOrRd",
            cbar_kws={"label": "Total Launched"})
plt.title("Monthly Attack Intensity by Top 5 Weapon Models", fontsize=14)
plt.xlabel("Weapon Model")
plt.ylabel("Year-Month")
plt.tight_layout()
```

```
plt.show()
```



1.5 Folium Geografische Visualisaties

1.5.1 Voorbereiding: Samenvoegen per regio (aggregatie) + geocoding

[24]:
```
# --- GEO AGGREGATIE: Regio's uitsplitsen (explode) en dan aggregeren ---

import ast

# Parse de list-strings naar echte lijsten en explode
def parse_region_list(region_str):
    """Parse string representation of list to actual list"""
    if isinstance(region_str, str):
        try:
            return ast.literal_eval(region_str)
        except:
            return [region_str]
    return region_str if isinstance(region_str, list) else [region_str]

# Eerst naar Pandas, parse, explode
```

```
temp_pdf = df_clean.select("affected region", "launched", "destroyed").
 ↪toPandas()
temp_pdf["region"] = temp_pdf["affected region"].apply(parse_region_list)
temp_pdf = temp_pdf.explode("region")

# Nu aggregeren per individuele regio
region_agg = (temp_pdf
              .groupby("region")
              .agg({
                  "launched": "sum",
                  "destroyed": "sum",
                  "region": "count"  # aantal aanvallen
              })
              .rename(columns={"region": "num_attacks",
                               "launched": "total_launched",
                               "destroyed": "total_destroyed"})
              .reset_index()
              .rename(columns={"region": "affected_region"})
)

# Success rate toevoegen
region_agg["success_rate"] = (region_agg["total_destroyed"] /␣
 ↪region_agg["total_launched"]).round(3)

region_agg.head(10)
```

[24]:

	affected_region	total_launched	total_destroyed	num_attacks	success_rate
0	Cherkasy oblast	4650.0	3098.0	29	0.666
1	Chernihiv oblast	7709.0	5672.0	37	0.736
2	Chernivtsi oblast	772.0	410.0	2	0.531
3	Dnipropetrovsk oblast	16890.0	11414.0	93	0.676
4	Donetsk oblast	13829.0	9182.0	83	0.664
5	Ivano-Frankivsk oblast	4204.0	3384.0	11	0.805
6	Kharkiv oblast	19210.0	11850.0	122	0.617
7	Kherson oblast	80.0	15.0	1	0.188
8	Khmelnytskyi oblast	5730.0	4225.0	17	
9	Kirovohrad oblast	3507.0	2530.0	16	

```
8          0.737
9          0.721
```

[21]:
```
!pip install geopy
```

```
Collecting geopy
  Downloading geopy-2.4.1-py3-none-any.whl.metadata (6.8 kB)
Collecting geographiclib<3,>=1.52 (from geopy)
  Downloading geographiclib-2.1-py3-none-any.whl.metadata (1.6 kB)
Downloading geopy-2.4.1-py3-none-any.whl (125 kB)
Downloading geographiclib-2.1-py3-none-any.whl (40 kB)
Installing collected packages: geographiclib, geopy
                           2/2 [geopy]
Successfully installed geographiclib-2.1 geopy-2.4.1
```

[26]:
```
# --- GEOCODING: Regio's omzetten naar coördinaten (eenmalig uitvoeren) ---

from geopy.geocoders import Nominatim
from geopy.extra.rate_limiter import RateLimiter

geolocator = Nominatim(user_agent="ukraine_missile_project")
geocode = RateLimiter(geolocator.geocode, min_delay_seconds=1)

def geocode_region(name):
    """Geocodeer Oekraïense oblast/regio namen"""
    try:
        # Probeer eerst exacte naam
        location = geocode(f"{name}, Ukraine")
        if location:
            return pd.Series({"lat": location.latitude, "lon": location.
 ↪longitude})
        # Fallback: zonder "oblast" suffix
        name_clean = name.replace(" oblast", "").replace(" region", "")
        location = geocode(f"{name_clean}, Ukraine")
        if location:
            return pd.Series({"lat": location.latitude, "lon": location.
 ↪longitude})
    except Exception as e:
        print(f"Fout bij geocoding {name}: {e}")
    return pd.Series({"lat": None, "lon": None})

# Geocodeer unieke regio's (run 1x en sla op in CSV)
geo_df = region_agg.copy()
geo_df[["lat", "lon"]] = geo_df["affected_region"].apply(geocode_region)
geo_df.to_csv("/home/jovyan/work/drone-project/ukraine_regions_geocoded_v2.
 ↪csv", index=False)
```

```
# Later: geo_df = pd.read_csv("ukraine_regions_geocoded.csv")
geo_df.head()
```

[26]:

	affected_region	total_launched	total_destroyed	num_attacks	success_rate	lat	lon
0	Cherkasy oblast	4650.0	3098.0	29	0.666	49.146017	31.227174
1	Chernihiv oblast	7709.0	5672.0	37	0.736	51.272593	31.741723
2	Chernivtsi oblast	772.0	410.0	2	0.531	48.381079	26.108167
3	Dnipropetrovsk oblast	16890.0	11414.0	93	0.676	48.662589	34.950172
4	Donetsk oblast	13829.0	9182.0	83	0.664	47.921291	37.780983

1.5.2 Folium Kaart: Circlemarkers per regio

[30]:

```
# --- FOLIUM KAART 1: CircleMarkers per regio ---

# Kaart centreren op Oekraïne
m = folium.Map(location=[49.0, 31.5], zoom_start=6, tiles="CartoDB positron")

# Kleurcodering op basis van success-rate
for _, row in geo_df.dropna(subset=["lat", "lon"]).iterrows():
    rate = row["success_rate"]

    # Kleur: groen = hoog, oranje = medium, rood = laag
    if rate >= 0.7:
        color = "green"
    elif rate >= 0.4:
        color = "orange"
    else:
        color = "red"

    # Radius proportioneel aan aantal aanvallen (zoals velostation)
    radius = row["total_launched"] / 150.0

    popup_html = f"""
    <b>{row['affected_region']}</b><br>
    Total launched: {int(row['total_launched'])}<br>
    Total destroyed: {int(row['total_destroyed'])}<br>
    Success rate: {rate:.2%}<br>
    Num attacks: {int(row['num_attacks'])}
    """
```

```
    folium.CircleMarker(
        location=[row["lat"], row["lon"]],
        radius=radius,
        color=color,
        fill=True,
        fill_opacity=0.7,
        popup=folium.Popup(popup_html, max_width=250)
    ).add_to(m)

# Voeg legenda toe (handmatig)
legend_html = '''
<div style="position: fixed; bottom: 50px; left: 50px; width: 200px;
            background-color: white; border:2px solid grey; z-index:9999;
            font-size:14px; padding: 10px">
<p><b>Success Rate</b></p>
<p><i class="fa fa-circle" style="color:green"></i> >= 70%</p>
<p><i class="fa fa-circle" style="color:orange"></i> 40-70%</p>
<p><i class="fa fa-circle" style="color:red"></i> < 40%</p>
</div>
'''
m.get_root().html.add_child(folium.Element(legend_html))

m.save("/home/jovyan/work/drone-project/ukraine_missile_attacks_map_v4.html")
m
```

[30]:
```
<folium.folium.Map at 0x7a7308944510>
```

1.5.3 HeatMap: Intensiteit aanvallen

[31]:
```
# --- FOLIUM KAART 2: HeatMap van aanvalsintensiteit ---

# Per aanval een punt maken (gebruik individuele records met locatie)
# Aanname: je wilt elke aanval als punt, niet per regio aggregeren
# Voor dit voorbeeld maken we een heatmap per regio met "gewicht" =␣
 ↪total_launched

heat_data = [
    [row["lat"], row["lon"], row["total_launched"]]
    for _, row in geo_df.dropna(subset=["lat", "lon"]).iterrows()
]

m2 = folium.Map(location=[49.0, 31.5], zoom_start=6, tiles="CartoDB␣
 ↪dark_matter")

HeatMap(heat_data, radius=25, blur=20, name="Attack Density").add_to(m2)

folium.LayerControl().add_to(m2)
```

```
m2.save("/home/jovyan/work/drone-project/ukraine_missile_heatmap.html")
m2
```

[31]: <folium.folium.Map at 0x7a7326219370>

1.6 Jupyter Widgets Demo

1.6.1 Dropdown filter om jaar te selecteren

[ ]:
```
# --- WIDGET 1: Jaar-filter voor tijdreeks (zoals "Land filter" bij Dronewars)
---

pdf_full = daily_agg.toPandas()
pdf_full["date"] = pd.to_datetime(pdf_full["date"])  # Zet om naar datetime
pdf_full["year"] = pdf_full["date"].dt.year

@widgets.interact(year=sorted(pdf_full["year"].unique()))
def plot_year_trend(year):
    """Plot tijdreeks voor geselecteerd jaar"""
    subset = pdf_full[pdf_full["year"] == year]

    plt.figure(figsize=(12, 5))
    sns.lineplot(data=subset, x="date", y="total_launched", label="Launched",
marker='o')
    sns.lineplot(data=subset, x="date", y="total_destroyed", label="Destroyed",
marker='o', color='green')
    plt.title(f"Missile Attacks in {year}", fontsize=14)
    plt.xlabel("Date")
    plt.ylabel("Count")
    plt.xticks(rotation=45)
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

```
interactive(children=(Dropdown(description='year', options=(np.int32(2022), np.
int32(2023), np.int32(2024), np…
```

1.6.2 Multi filter met regio en wapenmodel

[ ]:
```
# --- WIDGET 2: Gekoppelde filters regio + model ---

# Data voorbereiden
weapon_region_data = (df_clean
                      .filter(col("affected region").isNotNull())
                      .filter(col("model").isNotNull())
                      .select("date", "year_month", "affected region", "model",
```

```
                                "launched", "destroyed")
                        .toPandas())

regions = sorted(weapon_region_data["affected region"].dropna().unique())
models = sorted(weapon_region_data["model"].dropna().unique())

@widgets.interact(
    region=widgets.Dropdown(options=["ALL"] + regions, description="Regio:"),
    weapon=widgets.Dropdown(options=["ALL"] + models, description="Wapen:")
)
def plot_region_weapon(region, weapon):
    """Plot tijdreeks voor geselecteerde regio en wapen"""
    subset = weapon_region_data.copy()

    if region != "ALL":
        subset = subset[subset["affected region"] == region]
    if weapon != "ALL":
        subset = subset[subset["model"] == weapon]

    if subset.empty:
        print(f"Geen data voor {region} + {weapon}")
        return

    # Aggregeren per maand
    agg = subset.groupby("year_month")[["launched", "destroyed"]].sum().
 ↪reset_index()

    plt.figure(figsize=(12, 5))
    sns.lineplot(data=agg, x="year_month", y="launched", label="Launched",␣
 ↪marker='o')
    sns.lineplot(data=agg, x="year_month", y="destroyed", label="Destroyed",␣
 ↪marker='o', color='green')
    plt.title(f"Attacks: {region} | {weapon}", fontsize=14)
    plt.xlabel("Year-Month")
    plt.ylabel("Count")
    plt.xticks(rotation=45)
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

```
interactive(children=(Dropdown(description='Regio:', options=('ALL', "['Cherkasy␣
 ↪oblast', 'Kharkiv oblast', 'P…
```

1.6.3 Extra Word count doelwitten

[35]:
```
# --- RDD WORD COUNT: Meest getroffen doelwit-types ---

# Stopwords definiëren (pas aan op basis van je data)
stopwords = {"ukraine", "oblast", "region", "city", "and", "the", "of", "in",␣
 ↪"on", "at"}

# RDD maken van target kolom
rdd_targets = (df_clean
               .select("target")
               .filter(col("target").isNotNull())
               .rdd
               .flatMap(lambda row: [word.strip().lower() for word in row[0].
 ↪split(",")]))

# Word count
target_counts = (rdd_targets
                 .map(lambda word: word.strip(".,!?"))
                 .filter(lambda word: len(word) > 3 and word not in stopwords)
                 .map(lambda word: (word, 1))
                 .reduceByKey(lambda a, b: a + b)
                 .sortBy(lambda x: x[1], ascending=False))

print("Top 15 Meest Genoemde Doelwitten:")
for word, count in target_counts.take(15):
    print(f"  {word}: {count}")
```

```
Top 15 Meest Genoemde Doelwitten:
  south: 798
  odesa oblast: 196
  kherson oblast: 155
  dnipropetrovsk oblast: 126
  mykolaiv oblast: 111
  east: 91
  kharkiv oblast: 68
  dnipro raion: 34
  kryvyi rih raion: 27
  kyiv oblast: 25
  zaporizhzhia oblast: 24
  poltava oblast: 23
  donetsk oblast: 20
  sumy oblast: 18
  odesa oblast and mykolaiv oblast: 16
```

1.6.4 Extra: Word count per wapenmodel

[36]:
```python
# Word count per wapenmodel

# --- RDD WORD COUNT: Top 5 Meest Gebruikte Wapenmodellen ---

# 1. Selecteer de modelkolom en filter NULLs
rdd_models = (df_clean
              .select("model")
              .filter(col("model").isNotNull())
              .rdd)

# 2. MapReduce: Split op '/' om gecombineerde namen te scheiden
#    Bijv: "Iskander-M/KN-23" wordt ["Iskander-M", "KN-23"]
model_counts = (rdd_models
                .flatMap(lambda row: row.model.split('/'))
                .map(lambda word: word.strip())          # Spaties verwijderen
                .filter(lambda word: len(word) > 1)      # Lege strings/
 ↪afkortingen filteren
                .map(lambda word: (word, 1))             # Tuple (woord, 1) maken
                .reduceByKey(lambda a, b: a + b)         # Optellen per sleutel
                .sortBy(lambda x: x[1], ascending=False)) # Sorteren

# 3. Resultaat ophalen en printen
print("Top 5 Meest Gebruikte Wapenmodellen (op basis van vermeldingen):")
top5_models = model_counts.take(5)

for model, count in top5_models:
    print(f"  {model}: {count}")

# 4. Optioneel: Visualisatie van de top 5
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

pdf_top5 = pd.DataFrame(top5_models, columns=["Model", "Mentions"])

plt.figure(figsize=(10, 5))
sns.barplot(data=pdf_top5, x="Mentions", y="Model", palette="viridis")
plt.title("Top 5 Weapon Models by Frequency of Attacks")
plt.xlabel("Number of Mentions (Attack Waves)")
plt.ylabel("Weapon Model")
plt.tight_layout()
plt.show()
```

```
Top 5 Meest Gebruikte Wapenmodellen (op basis van vermeldingen):
  Shahed-136: 856
  131: 855
```

```
  Iskander-M: 308
  X-59: 254
  Orlan-10: 199
```

```
/tmp/ipykernel_63111/3347067966.py:36: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.
```

```
  sns.barplot(data=pdf_top5, x="Mentions", y="Model", palette="viridis")
```