# CA3 Integrated

In [1]:
```
import sys, pyspark
print(sys.executable)
print(pyspark.__version__)
```

```
/Users/endreasgard/miniconda3/envs/personlig311/bin/python
3.5.1
```

In [1]:
```
from pathlib import Path
import sys
import warnings
import subprocess

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.model_selection import GridSearchCV, TimeSeriesSplit
from statsmodels.tools.sm_exceptions import ConvergenceWarning

warnings.filterwarnings("ignore", category=FutureWarning)
warnings.filterwarnings("ignore", category=ConvergenceWarning)


def locate_root() -> Path:
    cwd = Path.cwd().resolve()
    candidates = [cwd, cwd.parent]
    for candidate in candidates:
        if (candidate / "common").exists() and (candidate / "data").exists():
            return candidate
    return cwd


PROJECT_ROOT = locate_root()
if str(PROJECT_ROOT) not in sys.path:
    sys.path.insert(0, str(PROJECT_ROOT))

from common.ca_shared import (
    add_region_column,
    best_window_lag,
    create_cassandra_connection,
    create_spark_session,
    ensure_keyspace,
    fit_sarimax_holdout,
    load_settings,
    read_cassandra_table_with_spark,
    write_spark_df_to_cassandra,
)
```

```
DATA_DIR = PROJECT_ROOT / "data"
OUTPUT_DIR = PROJECT_ROOT / "CA3" / "outputs"
OUTPUT_DIR.mkdir(exist_ok=True)

java_version = subprocess.check_output(["java", "-version"], stderr=subprocess
# print("Java:", java_version)
# print("Project root:", PROJECT_ROOT)
```

In [2]:

```
settings = load_settings()
TARGET_YEAR = settings["general"]["target_year"]
KEYSPACE = settings["cassandra"]["keyspace"]

cluster, session = create_cassandra_connection(settings)
ensure_keyspace(session, KEYSPACE)

try:
    spark = create_spark_session(settings, app_name="CA3CassandraSpark")
except Exception as exc:
    cluster.shutdown()
    raise RuntimeError(
        "Spark failed to start. Check version compatibility first: "
        "pyspark must be <=3.5.x for Cassandra connector support, and Java sho
        f"Root cause: {exc}"
    ) from exc

print("Connected Cassandra keyspace:", KEYSPACE)
print("Spark version:", spark.version)
```

```
:: loading settings :: url = jar:file:/Users/endreasgard/miniconda3/envs/person
lig311/lib/python3.11/site-packages/pyspark/jars/ivy-2.5.1.jar!/org/apache/ivy/
core/settings/ivysettings.xml
```

```
Ivy Default Cache set to: /Users/endreasgard/.ivy2/cache
The jars for the packages stored in: /Users/endreasgard/.ivy2/jars
com.datastax.spark#spark-cassandra-connector_2.12 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-f026f071-a45
a-487f-aa78-d8b32800e767;1.0
        confs: [default]
        found com.datastax.spark#spark-cassandra-connector_2.12;3.5.1 in centra
l
        found com.datastax.spark#spark-cassandra-connector-driver_2.12;3.5.1 in
central
        found org.scala-lang.modules#scala-collection-compat_2.12;2.11.0 in cen
tral
        found org.apache.cassandra#java-driver-core-shaded;4.18.1 in central
        found com.datastax.oss#native-protocol;1.5.1 in central
        found com.datastax.oss#java-driver-shaded-guava;25.1-jre-graal-sub-1 in
central
        found com.typesafe#config;1.4.1 in central
        found org.slf4j#slf4j-api;1.7.26 in central
        found io.dropwizard.metrics#metrics-core;4.1.18 in central
        found org.hdrhistogram#HdrHistogram;2.1.12 in central
        found org.reactivestreams#reactive-streams;1.0.3 in central
        found org.apache.cassandra#java-driver-mapper-runtime;4.18.1 in central
        found org.apache.cassandra#java-driver-query-builder;4.18.1 in central
        found org.apache.commons#commons-lang3;3.10 in central
        found com.thoughtworks.paranamer#paranamer;2.8 in central
        found org.scala-lang#scala-reflect;2.12.19 in central
:: resolution report :: resolve 205ms :: artifacts dl 6ms
        :: modules in use:
        com.datastax.oss#java-driver-shaded-guava;25.1-jre-graal-sub-1 from cen
tral in [default]
        com.datastax.oss#native-protocol;1.5.1 from central in [default]
        com.datastax.spark#spark-cassandra-connector-driver_2.12;3.5.1 from cen
tral in [default]
        com.datastax.spark#spark-cassandra-connector_2.12;3.5.1 from central in
[default]
        com.thoughtworks.paranamer#paranamer;2.8 from central in [default]
        com.typesafe#config;1.4.1 from central in [default]
        io.dropwizard.metrics#metrics-core;4.1.18 from central in [default]
        org.apache.cassandra#java-driver-core-shaded;4.18.1 from central in [de
fault]
        org.apache.cassandra#java-driver-mapper-runtime;4.18.1 from central in
[default]
        org.apache.cassandra#java-driver-query-builder;4.18.1 from central in
[default]
        org.apache.commons#commons-lang3;3.10 from central in [default]
        org.hdrhistogram#HdrHistogram;2.1.12 from central in [default]
        org.reactivestreams#reactive-streams;1.0.3 from central in [default]
        org.scala-lang#scala-reflect;2.12.19 from central in [default]
        org.scala-lang.modules#scala-collection-compat_2.12;2.11.0 from central
in [default]
        org.slf4j#slf4j-api;1.7.26 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
```

```
        ---------------------------------------------------------------------
        |      default     |   16  |   0   |   0   |   0   ||   16  |   0   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent-f026f071-a45a-487f-aa78-d
8b32800e767
        confs: [default]
        0 artifacts copied, 16 already retrieved (0kB/5ms)
26/02/21 14:43:16 WARN NativeCodeLoader: Unable to load native-hadoop library f
or your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLev
el(newLevel).
Connected Cassandra keyspace: fishhealth
Spark version: 3.5.1
```

# 1) Prepare Cassandra tables from local project data

In [4]:

```
session.execute(
    """
    CREATE TABLE IF NOT EXISTS ca3_localities (
        week int,
        localityno int,
        year int,
        avgadultfemalelice double,
        haspd boolean,
        hasila boolean,
        lat double,
        lon double,
        region text,
        PRIMARY KEY ((week), localityno, year)
    )
    """
)

session.execute(
    """
    CREATE TABLE IF NOT EXISTS ca3_weather_lice (
        week int,
        seatemperature double,
        avgadultfemalelice double,
        avgmobilelice double,
        avgstationarylice double,
        mean_air_temperature double,
        mean_relative_humidity double,
        mean_wind_speed double,
        sum_precipitation_amount double,
        mean_air_temperature_lag1 double,
        mean_relative_humidity_lag1 double,
        mean_wind_speed_lag1 double,
        sum_precipitation_amount_lag1 double,
```

```
        PRIMARY KEY (week)
    )
    """
)

loc_count = session.execute("SELECT count(*) FROM ca3_localities").one().count
feat_count = session.execute("SELECT count(*) FROM ca3_weather_lice").one().cc
print("Existing rows in ca3_localities:", loc_count)
print("Existing rows in ca3_weather_lice:", feat_count)

if loc_count == 0:
    locations_path = DATA_DIR / "locations_df.csv"
    if not locations_path.exists():
        raise FileNotFoundError(f"Missing file: {locations_path}")

    loc_pd = pd.read_csv(locations_path)
    loc_pd = loc_pd.rename(
        columns={
            "localityNo": "localityno",
            "avgAdultFemaleLice": "avgadultfemalelice",
            "hasPd": "haspd",
            "hasIla": "hasila",
        }
    )

    loc_pd = add_region_column(loc_pd, lat_col="lat", target_col="region")

    if "year" in loc_pd.columns:
        filtered = loc_pd[loc_pd["year"] == TARGET_YEAR]
        if not filtered.empty:
            loc_pd = filtered

    loc_pd = loc_pd[[
        "week", "localityno", "year", "avgadultfemalelice",
        "haspd", "hasila", "lat", "lon", "region"
    ]].copy()

    loc_pd["week"] = pd.to_numeric(loc_pd["week"], errors="coerce").astype("In
    loc_pd["localityno"] = pd.to_numeric(loc_pd["localityno"], errors="coerce"
    loc_pd["year"] = pd.to_numeric(loc_pd["year"], errors="coerce").astype("In

    loc_pd = loc_pd.dropna(subset=["week", "localityno", "year", "lat", "lon"]
    loc_pd["week"] = loc_pd["week"].astype(int)
    loc_pd["localityno"] = loc_pd["localityno"].astype(int)
    loc_pd["year"] = loc_pd["year"].astype(int)

    loc_spark = spark.createDataFrame(loc_pd)
    write_spark_df_to_cassandra(loc_spark, KEYSPACE, "ca3_localities", mode="a

    loc_count = session.execute("SELECT count(*) FROM ca3_localities").one().c
    print("Rows inserted into ca3_localities:", loc_count)

if feat_count == 0:
```

```
    feature_path = DATA_DIR / "sar_weather_lice.csv"
    if not feature_path.exists():
        raise FileNotFoundError(f"Missing file: {feature_path}")

    feat_pd = pd.read_csv(feature_path)

    needed = [
        "week",
        "seatemperature",
        "avgadultfemalelice",
        "avgmobilelice",
        "avgstationarylice",
        "mean_air_temperature",
        "mean_relative_humidity",
        "mean_wind_speed",
        "sum_precipitation_amount",
        "mean_air_temperature_lag1",
        "mean_relative_humidity_lag1",
        "mean_wind_speed_lag1",
        "sum_precipitation_amount_lag1",
    ]

    missing = [c for c in needed if c not in feat_pd.columns]
    if missing:
        raise ValueError(f"Missing columns in sar_weather_lice.csv: {missing}"

    feat_pd = feat_pd[needed].copy()
    feat_pd["week"] = pd.to_numeric(feat_pd["week"], errors="coerce").astype("
    feat_pd = feat_pd.dropna(subset=["week"])
    feat_pd["week"] = feat_pd["week"].astype(int)

    feat_spark = spark.createDataFrame(feat_pd)
    write_spark_df_to_cassandra(feat_spark, KEYSPACE, "ca3_weather_lice", mode

    feat_count = session.execute("SELECT count(*) FROM ca3_weather_lice").one(
    print("Rows inserted into ca3_weather_lice:", feat_count)
```

```
Existing rows in ca3_localities: 89084
Existing rows in ca3_weather_lice: 26
```

# 2) Read data back through Spark from Cassandra

In [5]:

```
locations_spark = read_cassandra_table_with_spark(spark, KEYSPACE, "ca3_locali
features_spark = read_cassandra_table_with_spark(spark, KEYSPACE, "ca3_weather

locations_df = locations_spark.toPandas().sort_values(["week", "localityno"])
feature_df = features_spark.toPandas().sort_values("week")

print("locations_df shape:", locations_df.shape)
print("feature_df shape:", feature_df.shape)
locations_df.head(5)
```

```
locations_df shape: (89084, 9)
feature_df shape: (26, 13)
```

Out[5]:

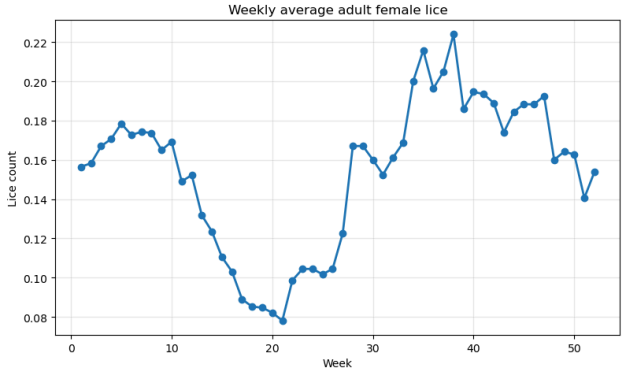
| | week | localityno | year | avgadultfemalelice | hasila | haspd | lat | |
|---|---|---|---|---|---|---|---|---|
| **32553** | 1 | 10029 | 2022 | NaN | False | False | 59.371235 | 5.2 |
| **32554** | 1 | 10037 | 2022 | NaN | False | False | 60.339880 | 4.9 |
| **32555** | 1 | 10040 | 2022 | NaN | False | False | 59.880650 | 5.1 |
| **32556** | 1 | 10041 | 2022 | 0.38 | False | False | 60.054317 | 5.0 |
| **32557** | 1 | 10044 | 2022 | NaN | False | False | 59.862167 | 5.1 |

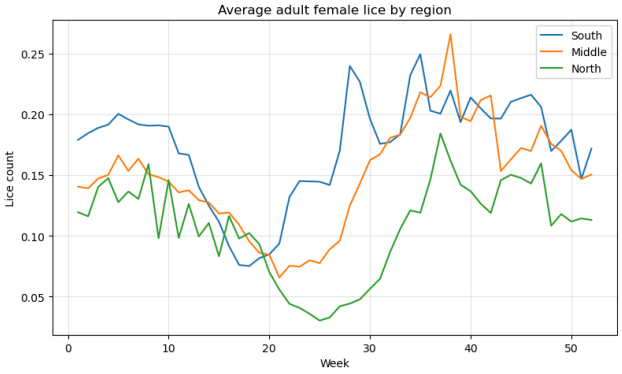
# 3) Pivoting and regional analysis

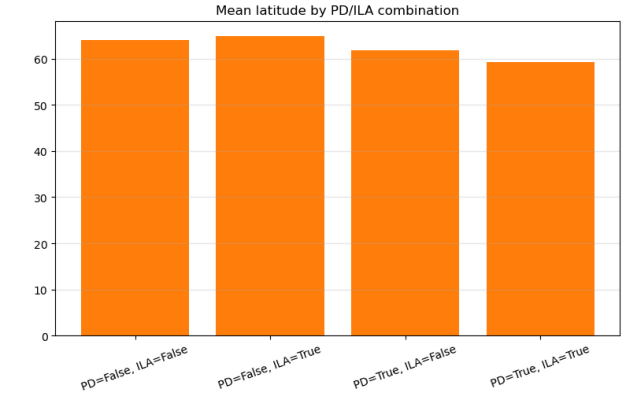
In [6]:

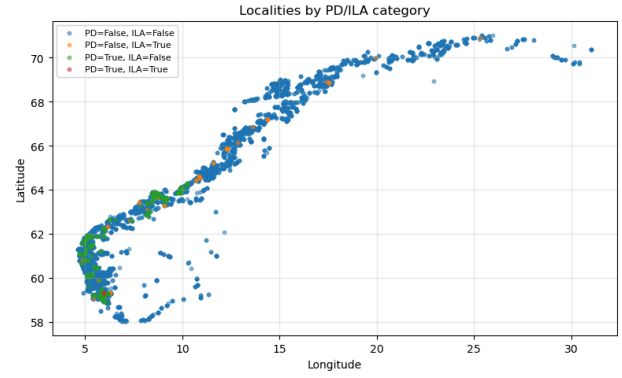
```
weekly_avg = (
    locations_df.groupby("week", dropna=True)["avgadultfemalelice"]
    .mean()
    .reset_index(name="avgadultfemalelice")
)

pivot_table_pd_ila = (
    locations_df.pivot_table(
        index=["haspd", "hasila"],
        values="lat",
        aggfunc="mean",
    )
    .reset_index()
)

pivot_table_week_region = (
    locations_df.pivot_table(
        index="week",
        columns="region",
        values="avgadultfemalelice",
        aggfunc="mean",
    )
    .sort_index()
)

fig, axes = plt.subplots(2, 2, figsize=(16, 10))

axes[0, 0].plot(weekly_avg["week"], weekly_avg["avgadultfemalelice"], marker="
axes[0, 0].set_title("Weekly average adult female lice")
axes[0, 0].set_xlabel("Week")
axes[0, 0].set_ylabel("Lice count")
axes[0, 0].grid(alpha=0.3)

for region in [c for c in ["South", "Middle", "North"] if c in pivot_table_wee
    axes[0, 1].plot(pivot_table_week_region.index, pivot_table_week_region[reg
```

```
axes[0, 1].set_title("Average adult female lice by region")
axes[0, 1].set_xlabel("Week")
axes[0, 1].set_ylabel("Lice count")
axes[0, 1].legend()
axes[0, 1].grid(alpha=0.3)

tmp = pivot_table_pd_ila.copy()
tmp["label"] = "PD=" + tmp["haspd"].astype(str) + ", ILA=" + tmp["hasila"].ast
axes[1, 0].bar(tmp["label"], tmp["lat"], color="tab:orange")
axes[1, 0].set_title("Mean latitude by PD/ILA combination")
axes[1, 0].tick_params(axis="x", rotation=20)
axes[1, 0].grid(axis="y", alpha=0.3)

sample_map = locations_df[["lon", "lat", "haspd", "hasila"]].dropna().copy()
if len(sample_map) > 5000:
    sample_map = sample_map.sample(5000, random_state=42)
sample_map["category"] = "PD=" + sample_map["haspd"].astype(str) + ", ILA=" +
for category, group in sample_map.groupby("category"):
    axes[1, 1].scatter(group["lon"], group["lat"], s=10, alpha=0.5, label=cate
axes[1, 1].set_title("Localities by PD/ILA category")
axes[1, 1].set_xlabel("Longitude")
axes[1, 1].set_ylabel("Latitude")
axes[1, 1].legend(fontsize=8)
axes[1, 1].grid(alpha=0.3)

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



# 4) Sliding-window correlations (sea temperature vs lice)

In [7]:
```
temp = feature_df["seatemperature"].astype(float)
series_map = {
    "avgadultfemalelice": feature_df["avgadultfemalelice"].astype(float),
    "avgmobilelice": feature_df["avgmobilelice"].astype(float),
    "avgstationarylice": feature_df["avgstationarylice"].astype(float),
}

windows = [4, 6, 8, 10]
lags = [0, 1, 2, 3]

best_cfg = {name: best_window_lag(temp, series, windows, lags) for name, serie

best_cfg_df = pd.DataFrame([
    {
        "series": name,
        "best_window": cfg["window"],
        "best_lag": cfg["lag"],
        "mean_abs_correlation": cfg["score"],
    }
    for name, cfg in best_cfg.items()
    if cfg is not None
])

best_cfg_df
```

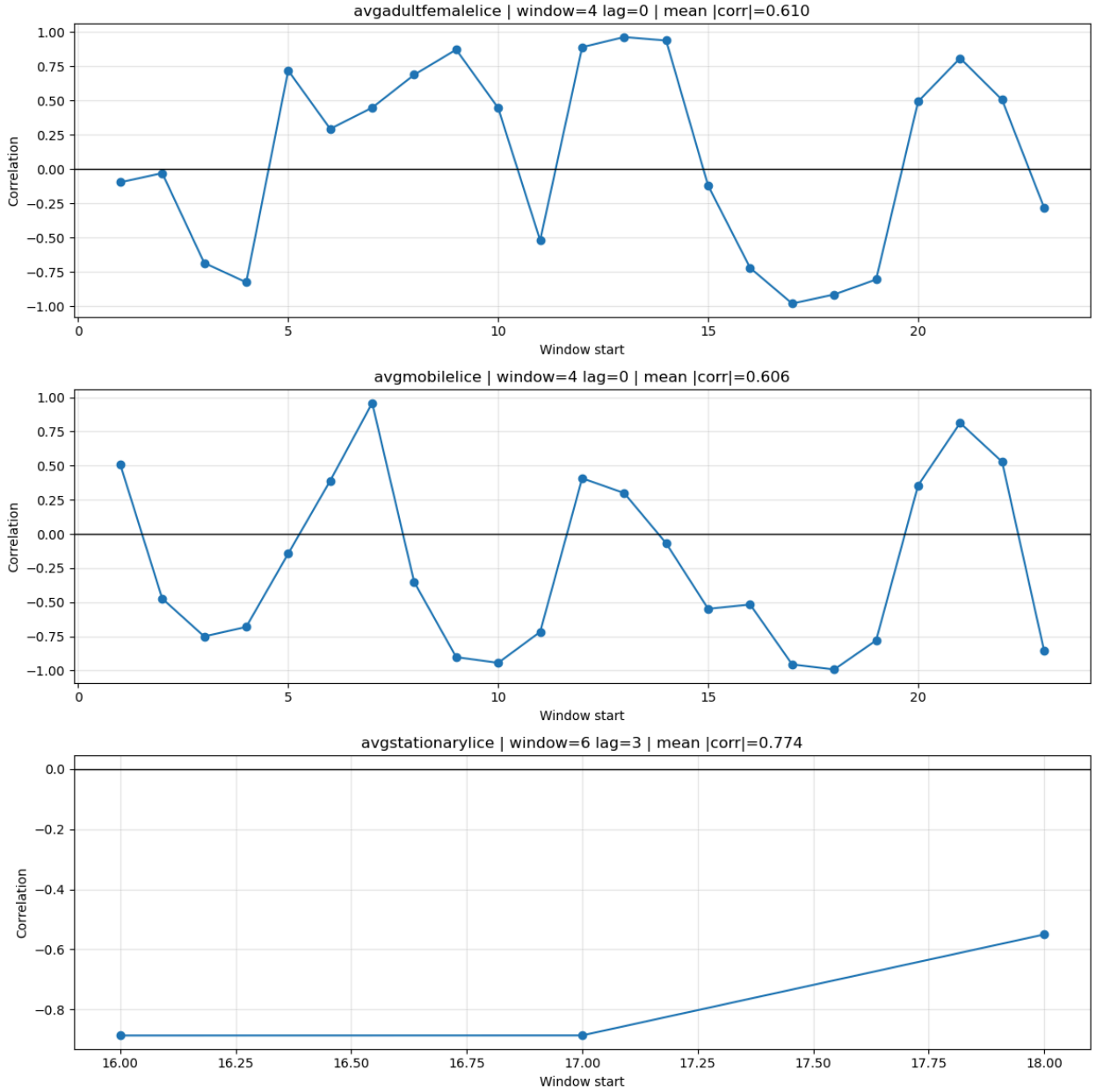
```
/Users/endreasgard/miniconda3/envs/personlig311/lib/python3.11/site-packages/nu
mpy/lib/_function_base_impl.py:3065: RuntimeWarning: invalid value encountered
in divide
  c /= stddev[:, None]
/Users/endreasgard/miniconda3/envs/personlig311/lib/python3.11/site-packages/nu
mpy/lib/_function_base_impl.py:3066: RuntimeWarning: invalid value encountered
in divide
  c /= stddev[None, :]
```

Out[7]:

| | series | best_window | best_lag | mean_abs_correlation |
|---|---|---|---|---|
| **0** | avgadultfemalelice | 4 | 0 | 0.610241 |
| **1** | avgmobilelice | 4 | 0 | 0.606241 |
| **2** | avgstationarylice | 6 | 3 | 0.774091 |

In [8]:
```
fig, axes = plt.subplots(3, 1, figsize=(12, 12))
for ax, (name, cfg) in zip(axes, best_cfg.items()):
    if cfg is None:
        ax.set_title(f"{name}: no valid config")
        continue
    s = cfg["series"]
    ax.plot(s.index, s.values, marker="o")
```

```
    ax.axhline(0, color="black", linewidth=1)
    ax.set_title(f"{name} | window={cfg['window']} lag={cfg['lag']} | mean |cc
    ax.set_xlabel("Window start")
    ax.set_ylabel("Correlation")
    ax.grid(alpha=0.3)

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




# 5) Forecasting (ARIMAX + ML baseline)

In [9]:
```
base_cols = [
    "mean_air_temperature",
    "mean_relative_humidity",
```

```
    "mean_wind_speed",
    "sum_precipitation_amount",
]
lag_cols = [
    "mean_air_temperature_lag1",
    "mean_relative_humidity_lag1",
    "mean_wind_speed_lag1",
    "sum_precipitation_amount_lag1",
]

base_cols = [c for c in base_cols if c in feature_df.columns]
lag_cols = [c for c in lag_cols if c in feature_df.columns]

arimax_a = fit_sarimax_holdout(feature_df, "seatemperature", base_cols, "ARIMA
arimax_b = fit_sarimax_holdout(feature_df, "seatemperature", base_cols + lag_c

params_b = arimax_b["result"].params.drop(labels=["sigma2"], errors="ignore")
coef_b = params_b[[c for c in params_b.index if c in (base_cols + lag_cols)]]
least_influential = coef_b.abs().sort_values().index[0]

reduced_cols = [c for c in (base_cols + lag_cols) if c != least_influential]
arimax_c = fit_sarimax_holdout(feature_df, "seatemperature", reduced_cols, f"A

metrics_arimax = pd.DataFrame([
    {"model": arimax_a["label"], "aic": arimax_a["aic"], "rmse": arimax_a["rms
    {"model": arimax_b["label"], "aic": arimax_b["aic"], "rmse": arimax_b["rms
    {"model": arimax_c["label"], "aic": arimax_c["aic"], "rmse": arimax_c["rms
]).sort_values("rmse")

metrics_arimax
```

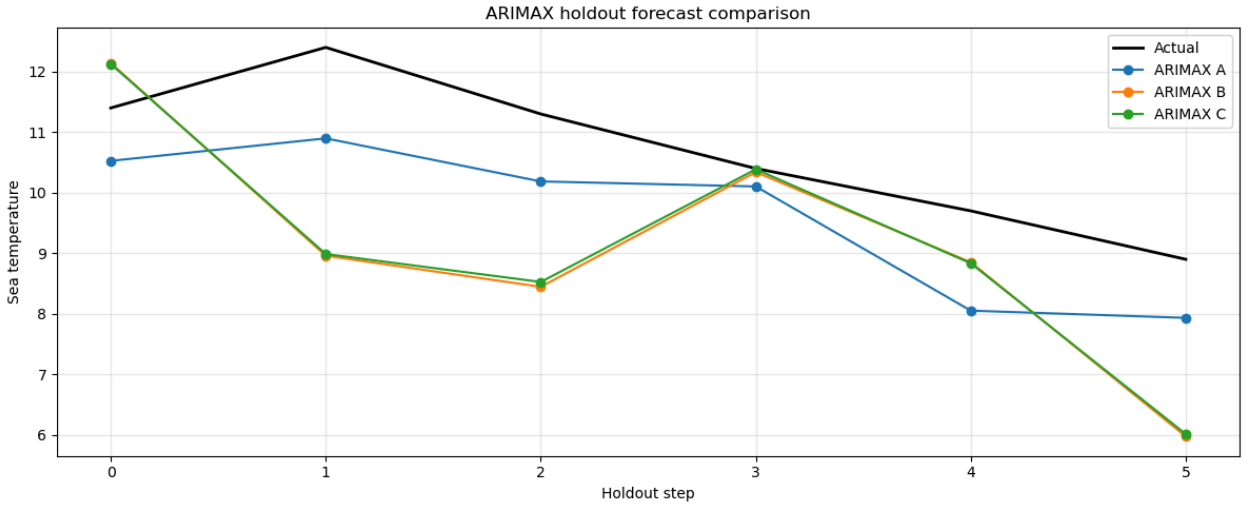
Out[9]:

| | model | aic | rmse | mae |
|---|---|---|---|---|
| **0** | ARIMAX_A_base | 91.542874 | 1.154612 | 1.066847 |
| **2** | ARIMAX_C_reduced_minus_mean_relative_humidity_... | 71.648238 | 2.196574 | 1.779561 |
| **1** | ARIMAX_B_base_plus_lag | 73.636473 | 2.229038 | 1.811506 |

In [10]:

```
plt.figure(figsize=(12, 5))

x_idx = np.arange(len(arimax_a["actual"]))
plt.plot(x_idx, arimax_a["actual"], color="black", linewidth=2, label="Actual"
plt.plot(x_idx, arimax_a["predicted"], marker="o", label="ARIMAX A")
plt.plot(x_idx, arimax_b["predicted"], marker="o", label="ARIMAX B")
plt.plot(x_idx, arimax_c["predicted"], marker="o", label="ARIMAX C")

plt.title("ARIMAX holdout forecast comparison")
plt.xlabel("Holdout step")
plt.ylabel("Sea temperature")
plt.grid(alpha=0.3)
plt.legend()
plt.tight_layout()
```

```
plt.show()
```




In [11]:

```
ml_df = feature_df[["seatemperature"] + (base_cols + lag_cols)].dropna().reset
X = ml_df[base_cols + lag_cols].astype(float)
y = ml_df["seatemperature"].astype(float)

split_idx = max(12, len(ml_df) - 6)
X_train, X_test = X.iloc[:split_idx], X.iloc[split_idx:]
y_train, y_test = y.iloc[:split_idx], y.iloc[split_idx:]

ts_cv = TimeSeriesSplit(n_splits=3)
param_grid = {
    "n_estimators": [100, 200, 300],
    "learning_rate": [0.03, 0.05, 0.1],
    "max_depth": [2, 3, 4],
    "subsample": [0.8, 1.0],
}

search = GridSearchCV(
    estimator=GradientBoostingRegressor(random_state=42),
    param_grid=param_grid,
    cv=ts_cv,
    scoring="neg_mean_squared_error",
    n_jobs=-1,
)
search.fit(X_train, y_train)

best_gb = search.best_estimator_
gb_pred = best_gb.predict(X_test)

gb_rmse = float(np.sqrt(mean_squared_error(y_test, gb_pred)))
gb_mae = float(mean_absolute_error(y_test, gb_pred))

metrics_ml = pd.DataFrame([
    {
        "model": "GradientBoosting_best_cv",
        "rmse": gb_rmse,
        "mae": gb_mae,
```

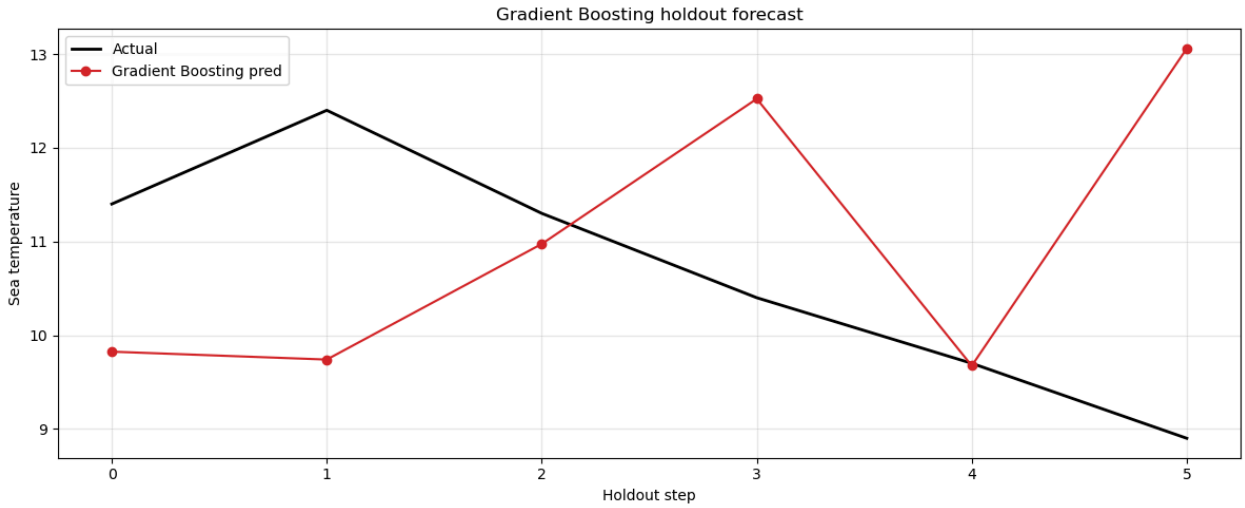
```
        "best_params": str(search.best_params_),
    }
])

metrics_ml
```

Out[11]:

| | model | rmse | mae | best_params |
|---|---|---|---|---|
| **0** | GradientBoosting_best_cv | 2.290185 | 1.81152 | {'learning_rate': 0.1, 'max_depth': 2, 'n_esti... |

In [12]:

```
plt.figure(figsize=(12, 5))
plt.plot(np.arange(len(y_test)), y_test.to_numpy(), color="black", linewidth=2
plt.plot(np.arange(len(gb_pred)), gb_pred, marker="o", color="tab:red", label=
plt.title("Gradient Boosting holdout forecast")
plt.xlabel("Holdout step")
plt.ylabel("Sea temperature")
plt.grid(alpha=0.3)
plt.legend()
plt.tight_layout()
plt.show()
```




# 6) Save outputs

In [13]:

```
week_region_out = OUTPUT_DIR / "week_region_pivot.csv"
pd_ila_out = OUTPUT_DIR / "pd_ila_lat_pivot.csv"
feature_out = OUTPUT_DIR / "feature_df.csv"
sliding_out = OUTPUT_DIR / "sliding_window_best.csv"
arimax_metrics_out = OUTPUT_DIR / "arimax_metrics.csv"
ml_metrics_out = OUTPUT_DIR / "ml_metrics.csv"

pred_out = OUTPUT_DIR / "arimax_predictions_holdout.csv"
gb_pred_out = OUTPUT_DIR / "gb_predictions_holdout.csv"

pivot_table_week_region.reset_index().to_csv(week_region_out, index=False)
pivot_table_pd_ila.to_csv(pd_ila_out, index=False)
```

```
feature_df.to_csv(feature_out, index=False)
best_cfg_df.to_csv(sliding_out, index=False)
metrics_arimax.to_csv(arimax_metrics_out, index=False)
metrics_ml.to_csv(ml_metrics_out, index=False)

pd.DataFrame({
    "actual": arimax_a["actual"],
    "arimax_a_pred": arimax_a["predicted"],
    "arimax_b_pred": arimax_b["predicted"],
    "arimax_c_pred": arimax_c["predicted"],
}).to_csv(pred_out, index=False)

pd.DataFrame({
    "actual": y_test.to_numpy(),
    "gb_pred": gb_pred,
}).to_csv(gb_pred_out, index=False)

print(f"Saved: {week_region_out}")
print(f"Saved: {pd_ila_out}")
print(f"Saved: {feature_out}")
print(f"Saved: {sliding_out}")
print(f"Saved: {arimax_metrics_out}")
print(f"Saved: {ml_metrics_out}")
print(f"Saved: {pred_out}")
print(f"Saved: {gb_pred_out}")
```

```
Saved: /Users/endreasgard/NMBU/IND320/IND320/CA3/outputs/week_region_pivot.csv
Saved: /Users/endreasgard/NMBU/IND320/IND320/CA3/outputs/pd_ila_lat_pivot.csv
Saved: /Users/endreasgard/NMBU/IND320/IND320/CA3/outputs/feature_df.csv
Saved: /Users/endreasgard/NMBU/IND320/IND320/CA3/outputs/sliding_window_best.cs
v
Saved: /Users/endreasgard/NMBU/IND320/IND320/CA3/outputs/arimax_metrics.csv
Saved: /Users/endreasgard/NMBU/IND320/IND320/CA3/outputs/ml_metrics.csv
Saved: /Users/endreasgard/NMBU/IND320/IND320/CA3/outputs/arimax_predictions_hol
dout.csv
Saved: /Users/endreasgard/NMBU/IND320/IND320/CA3/outputs/gb_predictions_holdou
t.csv
```

In [14]:
```
spark.stop()
cluster.shutdown()
print("Closed Spark and Cassandra connections.")
```

```
Closed Spark and Cassandra connections.
```