ChatGPT

Sharks from Space – Challenge Solution Blueprint

Challenge Brief

- **NASA's Ask:** Develop a **mathematical framework** to identify sharks and predict their foraging habitats using NASA's satellite data, and propose a **new shark tag concept** that not only tracks location but also measures what sharks are eating, with **real-time data transmission** [1] [2]. In essence, we must turn ocean observations ("**from space**") into actionable insight on **where sharks hunt** and imagine better **tags** to capture feeding events.

- **Judging Criteria → Our Checklist:** We will meet and exceed each criterion [3] [4]:

- **Influence:** Target a big problem (shark conservation) with broad impact (ecosystem and public interest). Our solution will help protect top predators and inform marine spatial planning (global conservation relevance).
- **Creativity:** Introduce an innovative fusion of NASA data (satellite oceanography) with shark biology, plus a novel multi-sensor tag design (unique cross-disciplinary approach).
- **Validity:** Base methods on peer-reviewed science (habitat models, eddy detection) and validate on real shark tracking data [5] [6] (scientifically sound, real-world tested).
- **Relevance:** Directly answer the challenge with a working model for shark hotspots *and* a tag concept. We'll deliver a functional pipeline and demo story in 48 hours (on-target and technically feasible).

- **Presentation:** Tell a compelling story of "sharks from space" with clear visuals – an interactive map of predicted shark foraging zones, and a concise pitch hitting all rubric points (engaging storytelling). We will also aim for **Global Award** categories like *Best Use of Data* (leveraging 6+ NASA datasets), *Best Use of Science* (oceanographic methods), and *Galactic Impact* (protecting ocean health) [7] [8].

- **Our Strategy in a Nutshell:** Use NASA Earth observations (ocean color, temperature, height, salinity, currents, rainfall) to map ocean features – e.g. **thermal fronts, eddies, productive zones** – known to attract prey. Overlay **shark tag data** (e.g. open-source tracks) to label where sharks forage, then train an **ML model** that learns to predict shark presence from those features. Simultaneously, design a **smart tag** with sensors (temperature, depth, etc.) that can detect feeding events (e.g. stomach temperature drops when prey is ingested [9]) and transmit those events via satellite in near real-time. Finally, present a polished demo: a live map of "shark hotspots" with a 90-second narrative that ticks all judging boxes.

What to do next: Assemble key scientific references on shark habitat drivers; confirm NASA data availability and access; outline major tasks and team roles for data, modeling, and presentation.

Landscape Review (Scientific Background)

Sharks are challenging to track, but decades of research link their movements to oceanographic features. Below we summarize 10+ relevant findings to guide our model features:

1. **Fronts attract filter-feeders:** Basking sharks strongly **associate with thermal and chlorophyll fronts**, where zooplankton prey concentrate [5] [10]. Surface frontal activity was a significant predictor of basking shark presence in the NE Atlantic [5]. This highlights that **temperature gradients** and **productivity gradients** are key foraging cues.

2. **Productivity as a proxy for prey:** Chlorophyll-*a* (phytoplankton concentration) and sea surface temperature (SST) are proxies for ocean productivity and thus prey availability for pelagic sharks [11]. Higher chlorophyll means more plankton and fish; sharks (even top predators) often frequent **mesotrophic** (moderate productivity) areas where the food web is active [11]. Extremely low (oligotrophic) or very high (bloom) chlorophyll can both be less favorable.

3. **Eddies and prey concentration:** Mesoscale eddies alter local ecosystems. *Cyclonic* (cold-core) eddies can upwell nutrients, boosting phytoplankton; *anticyclonic* (warm-core) eddies can aggregate prey internally or along their periphery [12] [13]. Notably, mature white sharks were observed spending significant time **inside warm-core (anticyclonic) eddies**, diving ~1000 m to feed on deep prey – the warm eddy waters likely **reduced their energy costs** in cold depths [6] [14]. This suggests eddies (especially warm-core) can create profitable foraging grounds by making deep prey more accessible.

4. **Sharks follow temperature preferences:** Many sharks have preferred SST ranges. For example, great white sharks are often found in ~15–22 °C surface waters (a "Goldilocks zone"), though they can tolerate ~7–27 °C [15]. Juvenile white sharks stick to 14–24 °C shallows [16]. Temperature influences shark distribution both directly (comfort/physiology) and indirectly (via prey distribution). **We will incorporate SST and identify thermal fronts** to capture these preferences.

5. **Salinity and coastal inputs:** Some sharks show affinity for lower salinity waters near estuaries or river plumes where prey congregate. In the Gulf of Mexico, blacktip shark occurrence was highest in **waters of lower salinity (river-influenced)** and with high chlorophyll near wetlands [17]. Tiger sharks in Shark Bay, Australia also track salinity gradients (preferring certain brackish ranges) [18]. **Sea surface salinity (SSS)** can thus indicate river outflow or frontal zones that attract prey (and sharks).

6. **Currents and convergence zones:** Surface currents can create convergence zones that aggregate nutrients and organisms. For instance, convergence at fronts or eddy edges can trap plankton or fish eggs, forming "feeding hotspots." Strong currents can also transport sharks or their prey. We'll use surface current data (e.g. **OSCAR currents**) to derive features like current **speed** (turbulent vs calm waters) and **convergence** (from current divergence).

7. **Rainfall and runoff (indirect):** Large rainfall events can create nutrient-rich runoff plumes in coastal areas, stimulating local productivity (e.g. river plumes) [19]. While precipitation itself may not directly attract sharks, the **secondary effects** (lower salinity, higher turbidity and nutrients) could influence

shark foraging in coastal regions. We consider using recent **GPM IMERG precipitation** as a contextual feature for coastal foraging, especially for species known to enter estuaries.

8. **Multi-factor habitat models:** Shark habitat models often use **multiple environmental predictors**. Birkmanis *et al.* (2020) modeled global shark richness with predictors including *chlorophyll-a, SST, salinity, depth, and human impact* [20]. They found that environmental variables significantly explained shark occurrences [20]. This supports our multi-variable approach (thermal fronts + productivity + physical ocean features).

9. **Seasonal and climate effects:** Seasonal changes (temperature, productivity cycles) shift shark distributions [21]. E.g., juvenile white sharks off California move seasonally as SST and eddy kinetic energy (EKE) change [22]. Climate trends like warming and deoxygenation are also driving sharks poleward and shallower [23] [24]. Our model will include time (month/season) implicitly via data to capture seasonal habitat shifts.

10. **Tracking data insights:** Tagging studies reveal fine-scale behavior. White sharks repeatedly visiting **frontal zones** and **eddies** [25] [6]; whale sharks aggregating in **plankton-rich warm waters** [26]; and reef sharks clustering in **specific temperature-salinity niches** [18]. These patterns guide our feature engineering: e.g. a **front index**, **eddy proximity**, and deviations of local conditions from mean (anomalies) to flag unusual productive events that might trigger feeding.

Together, these studies paint a picture: **sharks forage where and when oceanographic conditions favor their prey**. Key features are **productive waters (chlorophyll)**, **appropriate thermal range (SST)**, **dynamic structures (fronts, eddies)**, and sometimes **lower salinity coastal inputs**. We will use these insights to design our data features and model.

What to do next: Finalize which features to derive (e.g. SST fronts, eddy presence, etc.) based on these findings; obtain shark tracking data for our chosen species/region to validate these relationships; ensure access to cited papers for deeper parameter values (e.g. front detection thresholds, optimal ranges).

Dataset Inventory & Suitability Matrix

To predict shark habitats from space, we prioritize **NASA and partner datasets** that capture the ocean features above. Table 1 summarizes the candidate datasets – all are operationally accessible via NASA Distributed Active Archive Centers (DAACs):

Table 1. Candidate Datasets for "Sharks from Space" (NASA prioritized). Each dataset's characteristics and usage notes are listed:

Dataset (DAAC)	Variables & Units	Spatial / Temporal Res	Coverage & Latency	Access (API/ID)	Quality / Artifacts
PACE OCI L2 (OB.DAAC)	Chlorophyll-*a* (mg m^{-3}); spectral Rrs (1 nm bands)	~1 km pixel; ~2-day revisit (sun-synchronous)	Global ocean between clouds; **Latency** ~1-2 days for L2 [27] [28]	**CMR** ShortName: `PACE_OC_L2` (Provisional); OceanColor file API	Cloud-obstructed; glint near equator; quality flags for clouds, straylight (ghosting corrected in reproc v3.1 [29])
MODIS Aqua / VIIRS (OB.DAAC) *(Fallback)*	Chlorophyll-*a* (mg m^{-3})	4 km (MODIS) / 1 km (VIIRS); daily global	Global since 2002; Latency ~days	CMR Coll: `MODISA_L3_CHL_Daily` etc.	Clouds, sun glint; well-characterized calibration
GHRSST MUR SST (PO.DAAC)	Sea surface temp (°C) – foundation (night)	**0.011° (~1 km)** grid; **Daily** composites [30]	Global ice-free oceans; Latency ~1–2 days (real-time v4.1) [31]	**Harmony**/CMR: ShortName `MUR-JPL-L4-GLOB-v4.1` [32] (DOI:10.5067/GHGMR-4FJ04)	Gap-filled analysis (uses IR, microwave data); smoothed – may miss sharp fronts >0.5°C/km; some coastal bias and no-data near ice

Dataset (DAAC)	Variables & Units	Spatial / Temporal Res	Coverage & Latency	Access (API/ID)	Quality / Artifacts
SWOT Altimetry L2 (PO.DAAC)	Sea surface height (SSH, m); anomaly (SSHA)	~0.05° along-track; 120 km swath; 21-day repeat	Select regions (swaths); June 2023 availability; Latency weeks (science data)	**Earthdata** (requires login; CMR concept ID for SWOT L2)	High-resolution but **sparse** in time/space; calibration phase issues; contains flags for land/ice, bad data
JPL Merged Altimetry (MEaSUREs PO.DAAC)	Sea surface height anomaly (m)	**1/6° (~18 km)** grid; **5-day** interval [33]	Global 66°S–66°N (multi-satellite, 1992–present); Latency ~10 days (interim) [34]	CMR: ShortName `SEA_SURFACE_HEIGHT_ALT_INTERIM` (Example)	5-day smoothing; **no data near coast/ice**; 18 km res misses small eddies <50 km; CMEMS/AVISO altimetry similar
OSCAR Ocean Currents (PO.DAAC)	Zonal (u) & meridional (v) surface current (m/s)	**0.33° (~37 km)** grid; **5-day** averages [35]	Global ± 90° (gap at poles); available from 1992–present; ~2-day latency [36]	CMR: ShortName `OSCAR_L4_OC_third-deg` (v2)	Derived from SSH, winds, Ekman – accuracy ±0.1 m/s; smoothed flow, no tides; coastal currents not resolved

Dataset (DAAC)	Variables & Units	Spatial / Temporal Res	Coverage & Latency	Access (API/ID)	Quality / Artifacts
SMAP Sea Surface Salinity (PO.DAAC)	Salinity (PSU) at ~1 cm depth	**0.25° (~25 km)** grid; **8-day running mean** [37] [38]	Global ±70° lat (tropics & mid-latitudes; reduced quality >60°); Latency ~7–10 days	CMR: ShortName `JPL_SMAP_SSS_L3_8DAY_V5` (CAP v5.0)	**RF interference noise** in some regions; higher error in cold water; gaps near coasts; bias ~±0.2 PSU
GPM IMERG Precipitation (GES DISC)	Rain rate (mm/hr) or daily total (mm)	**0.1° (~11 km)** grid; 30-min temporal (monthly climatology avail.)	Global 60°S–60°N (full coverage); **Final run latency ~3 months**, Near-Real-Time (Early/Late runs 4–14 hrs) [39] [40]	GES DISC API: e.g. collection `GPM_3IMERGDL` (Daily) V06	Satellite est. error ~10–30%; detection misses very light rain; coastal heavy rain uncertainties

Key Dataset Suitability Considerations:

- *Spatial Resolution:* We target a common **0.1° grid** (~11 km) for modeling, which balances the finer detail of some datasets (PACE 1 km, MUR 1 km) with the coarser ones (altimetry 18 km, OSCAR 37 km). At 0.1° we can still resolve major fronts/eddies of interest. High-res datasets will be **downscaled** (averaged) to 0.1°, while coarse ones may be **interpolated** to 0.1° for alignment.

- *Temporal Resolution:* We will use **daily** intervals for the model. Datasets provided as multi-day composites (5-day OSCAR, 8-day SMAP) will be interpolated or held constant across their interval (e.g. a 5-day mean applied to each day in that window) to integrate with daily features. Our time window of interest (e.g. June–Aug 2023 by default) is short enough to assume environmental

conditions don't need monthly climatology adjustments, but we may incorporate **lagged features** (discussed later).

- *Coverage and Gaps:* Optical data (PACE, MODIS) have **cloud gaps**. We mitigate this by using daily composites or multi-day averages (e.g. an 8-day PACE L3 if available) and by supplementing with cloud-insensitive data (SST, SSH) when optical is missing. Altimetry has **no data very close to coasts and within ~10 km of land** – we will mask those coastal pixels in all datasets to avoid false signals. SMAP SSS similarly excludes ~40 km from coasts and is noisy at high latitudes; for our NW Atlantic example (mid-latitude open ocean) it is suitable. IMERG covers our latitude range (e.g. up to ~50°N for Cape Cod) with no gaps.

- *Data Volume:* All data are subset to our Region of Interest (ROI) and time. For example, NW Atlantic (roughly 30–50°N, 80–50°W) at 0.1° has ~200x300 grid = 60k pixels. Daily SST (MUR) at 1 km global is ~2.6 GB per day raw; subset to ROI (~2% of globe) and downsample to 0.1° yields ~1–2 MB/day. Similarly, PACE L2 files can be large (~200 MB granule covering an orbit); but we will request small spatial areas via API. Total data volume for ~90 days and all layers might be on the order of a few GB – manageable for local processing with xarray/dask. We will **limit memory use** by reading data in chunks (time or space) and using cloud-optimized formats (NetCDF4 with compression, or Zarr) where possible.

- *Auth & Access:* All datasets require **NASA Earthdata login** (free). We will use a shared authentication approach (token or netrc file) for scripting downloads. Many are available through the **Harmony API** (PO.DAAC and OB.DAAC have Harmony endpoints) which allow us to request only the subset we need (reducing volume significantly). We note any exceptions: e.g., GPM IMERG from GES DISC can be accessed via direct HTTPS with a token.

- *Licensing:* All NASA data are **open access** for use (U.S. Government public domain), requiring only attribution. OCEARCH shark track data (to be used for labels) is publicly viewable; we'll double-check their terms (likely requiring credit to OCEARCH, which we will provide). Our code and derived products will be under an open-source license (e.g. MIT) in the GitHub repo, to encourage reuse.

What to do next: Acquire sample granules from each dataset to test format and content; generate a small example (e.g. one day, one location) to confirm units and ranges (e.g. verify that MUR SST is in Kelvin or Celsius, check PACE chlorophyll scale); finalize the ROI and time range (default NW Atlantic Summer 2023) or adjust if track data suggests a different region.

Data Quality & Interoperability Audit

Integrating multi-mission data requires careful QA and standardization. We identify potential data quality issues and define a common framework for **coordinates, units, and grids**:

- **Clouds and Gaps:** PACE (and MODIS/VIIRS) optical data have missing pixels due to clouds and sunglint. **Detection:** Each L2 pixel has quality flags; we will mask out cloud-contaminated values (e.g. use the provided cloud mask product [41]). **Mitigation:** For daily compositing, we can fill small gaps by spatial interpolation or carry forward the last valid observation for a few days. Where persistent clouds exist, we rely on SST and SSH features (which are gap-free in L4 analyses). In polar night or

very low sun angle conditions (not an issue in our summer ROI, but generally), optical data would be absent – again SST/SSH serve as proxies in those cases.

- **Striping or Sensor Artifacts:** Early PACE versions had a known **ghosting artifact** in certain bands [29] (since corrected in reprocessing v3.1). MODIS historically had striping in some thermal channels. **Detection:** We monitor histograms of the data for abnormal stripes or systematic offsets. **Mitigation:** Use the latest reprocessed data (PACE v3.1+), and apply median filtering to remove striping noise if needed. For our use (broad features), any minor striping is unlikely to affect results significantly.

- **Land/Ice Masks:** We will apply a consistent **land mask** across datasets. Many products already flag land/ice (e.g., altimetry grids have fill over land, OSCAR currents have no data on land, SMAP SSS excludes land by design). We'll use a high-resolution coastline to mask out any grid cells that contain land if any slip through (especially at 0.1°, a coastal cell might be mixed water/land). Similarly, exclude data where sea-ice concentration is high (not likely in our warm-season ROI, but a consideration if extended). **Order of masking:** First, apply each product's own mask/flags (clouds, ice), then apply a unified land mask to all feature layers so that any land/coastal artifact is removed before analysis.

- **Unit Conventions:** We standardize all units:

- SST in **°C** (convert from Kelvin if necessary, e.g. some GHRSST are in Kelvin – MUR is typically in Kelvin; we will subtract 273.15 to get °C).
- Chlorophyll-*a* in **mg m^{-3}** (generally that's how OB.DAAC provides it). We will likely **log-transform chlorophyll** for modeling, given its log-normal distribution [42] (the front detection will use linear values, but modeling features will use log(Chl) and perhaps anomalies).
- SSH anomalies in **meters** (already in m). If we compute Okubo-Weiss (which involves derivatives of velocity in s^{-2}), we will be careful to compute in consistent SI units.
- Currents in m/s (OSCAR provides m/s).
- Salinity is unitless (PSU).

- Rain in mm (we might use mm/day).

- **Time Datum and Alignment:** We use **UTC** for all timestamps. Daily data will be referenced to the same 24h window (likely midnight-to-midnight GMT). Some products label times differently (IMERG daily file might be labeled by the file start at 00Z; MUR daily SST often labeled at 12Z or as an average). We will treat each "day" of data as covering that calendar day UTC. Shark tracking data timestamps will be converted to UTC and then matched to the nearest daily data (if a tag ping is at 15Z on Aug 1, 2023, we use Aug 1 daily data; if a ping is late night, it's still that day's environment – this introduces up to <12h mismatch which is acceptable given daily resolution). We allow ±1 day tolerance in matching if needed (e.g. if a shark ping has no data that day due to a gap, we can use the previous or next day's environmental data as a proxy).

- **Coordinate Reference System:** All data are in latitude/longitude with WGS84 datum (EPSG:4326) by default, which we will use. We note that some data (like model output) might use a 0–360° longitude range. We'll convert everything to **-180° to 180°** range for consistency with our ROI definition and ease of integration (Harmony subset requests support both). Our region spans the dateline? (No, NW

Atlantic is well within -180 to -50, so no dateline issues). We will ensure lat/lon arrays match in orientation (some datasets may provide descending lat order; xarray handles that if we are careful). We will use xarray's `.interp` or `.reindex` to put everything on a shared coordinate grid.

- **Common Grid & Regridding:** Target grid is **0.1° x 0.1°, daily**. For each dataset:

- If higher-res (PACE, MUR): we will **aggregate** to 0.1°. Preferred method is area-weighted binning (e.g. using xESMF or `rioxarray` to reproject to 0.1° grid with **conservative** resampling for continuous fields). For simplicity and speed, we might do a simple block average if the high-res grid aligns (MUR 0.01° aligns exactly into 0.1°, 10x10 block average). PACE swath data (L2) is not on a fixed grid, so we will bin those points into our 0.1° cells (e.g. averaging all valid pixels within each 0.1° cell).
- If lower-res (altimetry 1/6°, OSCAR 1/3°): we will **interpolate** to 0.1°. For altimetry, bilinear interpolation of SSH is reasonable as long as we don't create spurious new extrema. For currents, bilinear as well. We could also choose to keep altimetry at 0.167° and currents at 0.333° and upsample those just for matching with track points (point-sampling), but a unified grid simplifies the model input generation.
- The regridding will be done in a **lat-lon coordinate system** (we aren't dealing with projections like polar stereographic, since our ROI is mid-latitude).

- We will define the exact grid edges to ensure alignment (probably lat bins centered on 0.05° increments, e.g. 30.05, 30.15, ..., and lon accordingly). Using xarray's `.coarsen` or `.interp` functionalities ensures CF conventions (grid cell boundaries) are followed.

- **Time Aggregation:** The daily target means some data (5-day OSCAR, 8-day SSS) need time interpolation. We will likely keep those static for the days they cover (e.g. the 5-day OSCAR velocity for Aug 1–5 is applied to each of those days). Alternatively, we could linearly interpolate daily – but since ocean currents change gradually, using step changes every 5 days is fine for our short-term analysis. We will document this choice. Similarly for 8-day SSS, we'll use the same value for all days in that window (or linearly interpolate if needed to avoid sharp jumps on the boundary of the composite window).

- **Naming Schema:** We will assign standardized names to each data layer to avoid confusion:

- e.g. `SST` (sea_surface_temperature), `SST_grad` (front strength), `SSH_anom` (sea_surface_height_anomaly), `EKE` (eddy kinetic energy if derived), `Chl` (chlorophyll-a, possibly log10), `Chl_grad`, `Eddy_flag` (boolean inside eddy), `Cur_speed`, `SSS`, `Rain`. We will use CF standard names where possible and document in metadata. For features we compute (fronts, eddies), we'll include attributes describing how they were derived.

- **Quality Flags & Missing Data Handling:** Each dataset's QC flags will be respected:

- PACE/OC: use flags to exclude bad retrievals (e.g. high aerosols, straylight).
- MUR SST: being L4, it doesn't have per-pixel QC, but we know polar regions might be less reliable – in our ROI not an issue.
- Altimetry: use the provided "mapped_error" or "num_satellites" if available to mask low-confidence areas (like near land or if only one altimeter available).

- OSCAR: no explicit QC per vector; we might mask where input data density was low, but likely fine.
- SMAP SSS: use their provided error estimate or flag (RSS and JPL CAP products often have a "SSS uncertainty" or a mask for RFI contamination). We'll exclude SSS where uncertainty > threshold (e.g. >0.5 PSU).

- IMERG: use the "precipitationQualityIndex" if available, or note early vs final runs. For hack we probably use Late run (which has reasonable quality ~12hr latency).

- **Data Integration Checks:** After assembly, we will perform a quick sanity check: pick a known strong feature (e.g. Gulf Stream front on a date) and see if SST_grad and Chl_grad both highlight it, see if an eddy in SSH corresponds with a swirl in currents, etc. This ensures our regridding and alignment didn't distort the data.

By addressing these interoperability and quality points, we ensure the input features to our model are **consistent and reliable** across datasets.

What to do next: Script the regridding pipeline for one sample date to verify alignment (output a multi-layer NetCDF for, say, 2023-07-01 in ROI; check min/max and spatial plots for reasonableness); adjust any thresholds for masking (e.g. determine a sensible chlorophyll cloud-mask strategy like requiring OCI quality >0.5); finalize the list of output feature names and units for the modeling stage.

Retrieval & Subsetting Plan

We will use NASA's APIs and tools for efficient data access. Below we outline how to search and subset each dataset, with one working example per dataset. We emphasize **automation** – minimal manual downloading – to keep the pipeline agentic and reproducible:

- **Authentication:** We create a `.netrc` file or use an Earthdata **token** for scripted access. For example, using the `earthaccess` Python library:

```
import earthaccess
auth = earthaccess.login()  # uses .netrc or prompts for credentials
```

This will store a token for subsequent API calls [43]. For command-line (curl/wget), we include `-u USER:PASS` or better, `-H "Authorization: Bearer <token>"`.

- **Dataset Search (CMR):** NASA's CMR (Common Metadata Repository) allows query by short name, time, and spatial bounding box. Each dataset has a **collection ID** or short name:

- *Example (MUR SST)*: Short name = `MUR-JPL-L4-GLOB-v4.1`. We can search granules for July 2023 in our ROI via HTTP:

```
curl -H "Authorization: Bearer $EDL_TOKEN" -G \
  -d "short_name=MUR-JPL-L4-GLOB-v4.1" \
  -d "version=4.1" \
```

```
  -d "temporal=2023-07-01T00:00:00Z,2023-07-31T23:59:59Z" \
  -d "bounding_box=-80,30,-60,45" \
  -d "page_size=100" \
  https://cmr.earthdata.nasa.gov/search/granules.json
```

This returns a JSON listing granules matching the criteria (which are daily files) including their URLs. We could also use `earthaccess.search_data()` in Python to get these URLs.

- **Direct Download vs Harmony:** For many datasets on NASA's cloud, the files are in AWS S3 buckets. We prefer **Harmony** for on-the-fly subsetting and reformatting:

- Harmony can subset by bbox and time and even reformat (e.g. GeoTIFF, NetCDF, or Zarr). We will request NetCDF for familiarity.

- *Example (MUR SST via Harmony)*: The Harmony endpoint for PO.DAAC:

```
curl -L -H "Authorization: Bearer $EDL_TOKEN" \
  "https://harmony.earthdata.nasa.gov/harmony/PO.DAAC/MUR-JPL-L4-GLOB-v4.1
   ?bbox=-80,30,-60,45
   &datetime=2023-07-10T00:00:00Z,2023-07-10T23:59:59Z
   &interpolation=nearneighbor&format=netcdf"
  -o mur_sst_20230710.nc
```

This will submit a Harmony job to extract the 10-July-2023 SST for our box. We include `interpolation=nearneighbor` just as an example (Harmony might default to something appropriate for L4 data) and request output as NetCDF. The `-L` follows redirects (Harmony jobs give a status URL and then eventually the file URL). The output is a NetCDF only covering our ROI, dramatically smaller than the full global file.

- **Note:** If using Python, `harmony-py` client can simplify this [44] [45]. For example:

```
from harmony import BBox, Client, Collection, Request
client = Client()  # uses netrc for auth
request = Request(
   collection=Collection(id="MUR-JPL-L4-GLOB-v4.1"),
   spatial=BBox(-80, 30, -60, 45),
   temporal={'start': "2023-07-10T00:00:00Z", 'stop':
"2023-07-10T23:59:59Z"}
)
resp = client.submit(request).wait_for_processing()
urls = list(client.result_urls(resp))
# then download from urls[0] or read via s3 if using cloud environment
```

This yields the same subset file [46] [47]. We will incorporate these snippets in our notebooks for each data source.

- **PACE Ocean Color:** PACE L2 data are accessed via OB.DAAC. Currently, Earthdata Search or the OceanColor file API can be used. We might use the OB.DAAC File Search API with parameters for date, area, etc. Alternatively, if PACE L3 gridded products are available (8-day composite), we might take those for convenience:

- *Example:* Use CMR or direct:

```
curl -H "Authorization: Bearer $TOKEN" \
  "https://oceandata.sci.gsfc.nasa.gov/api/file_search?
mission=PACE&product=L2_OC&start_date=2023-07-01&end_date=2023-07-01&bbox=-80,30,-60,45"
```

  This would return matching file names. Then we can download via HTTPS (the data are in NASA's OceanColor HTTP archive, which requires the token).

- If we get L2 swath files, we will subset them after download (since Harmony might not yet support OB.DAAC subsetting for L2). We can do this with e.g. `gdalwarp` or a small Python routine to filter lat/lon arrays in the file.

- **Altimetry (MEaSUREs SSH):** The 5-day gridded SSH is provided as NetCDF with global coverage. We can download only needed time steps:

- PO.DAAC's OPeNDAP could be used (if available for this dataset) to slice by lon/lat/time. Or use Harmony if a service exists: *Example:*

```
curl -L -H "Authorization: Bearer $TOKEN" \
  "https://harmony.earthdata.nasa.gov/harmony/PO.DAAC/
GMSL_MEA_SSH_5DAY_VHR_v2205
   ?bbox=-80,30,-60,45
   &startDate=2023-07-01&endDate=2023-07-31
   &variables=adt,ugos,vgos
   &format=netcdf"
  -o ssh_5day_July2023.nc
```

  (Here `adt` might be the anomaly variable name; `ugos/vgos` could be optional u/v geostrophic currents if available). If Harmony doesn't support this collection, alternative is to download monthly files (which might be ~300 MB each) and subset locally with xarray. We will try Harmony first.

- **SWOT L2 (if used):** We would use CMR to find SWOT passes over our region/dates. However, integrating swath data is complex for a short hack – we likely rely on the gridded altimetry instead. If time permits, we might incorporate a SWOT pass by reading its Ka-band SSH swath and extracting eddy signals.

- **OSCAR Currents:** Available via PO.DAAC, the *third-degree, 5-day* product:

- Use Harmony similarly:

```
curl -L -H "Authorization: Bearer $TOKEN" \
  "https://harmony.earthdata.nasa.gov/harmony/PO.DAAC/oscar_L4_OC_third-deg
   ?bbox=-80,30,-60,45
   &startDate=2023-06-01&endDate=2023-08-31
   &format=netcdf"
  -o oscar_currents_JJA2023.nc
```

We'd confirm the correct collection ID for OSCAR. Alternatively, use the PO.DAAC DRIVE (an HTTPS directory) to fetch the needed NetCDF files (they are typically named by date ranges).

- **SMAP SSS:** Also at PO.DAAC (JPL CAP 8-day product):

- Use Harmony:

```
curl -L -H "Authorization: Bearer $TOKEN" \
  "https://harmony.earthdata.nasa.gov/harmony/PO.DAAC/
JPL_SMAP_L3_SSS_CAP_V5
   ?bbox=-80,30,-60,45
   &startDate=2023-06-01&endDate=2023-08-31
   &format=netcdf"
  -o smap_sss_JJA2023.nc
```

(We might need the exact collection ID from CMR). Or use direct download from PO.DAAC's AWS bucket if known.

- **GPM IMERG:** At GES DISC, we can use their subset API or OPeNDAP:

- The daily IMERG Final V06 is in an HDF5 or NetCDF with global 0.1° grid. GES DISC offers a "Subsetter" service via URL parameters. For example:

```
curl -J -L -b "urs_userid=...; urs_password=..." \
  "https://gpm1.gesdisc.eosdis.nasa.gov/data/GPM_L3/IMERGDM.06/2023/07/3B-
DAY.MS.MRG.3IMERG.20230710-S000000-E235959.V06.nc4
   ?subset=lat(30:45)&subset=lon(-80:-60)"
  -o imerg_20230710.nc4
```

Here we directly access the file with a query string to subset lat 30–45N and lon -80–-60 [48]. The `-b` sends Earthdata cookies (or we use `Authorization` header similarly). We'd loop this for each day or use OPeNDAP with a constraint expression across days (maybe concatenation might require a different approach though). Given IMERG's latency, we might use the "Late run" (near-real-time) data for up-to-date coverage if needed, with similar access.

- **Automation & Agents:** These commands will be orchestrated by our **Data-Scout agent** script. It will:

- Query CMR for each dataset to get relevant granule IDs or direct URLs.
- Either call Harmony API for each (preferred) or fall back to parallel downloads + local subset if needed.
- Save subsets in a structured directory (e.g. `data/raw/SST_20230710.nc`, etc).

We will implement **retry logic** and verification (e.g. check file sizes) to avoid silent failures. For example, if a Harmony request fails (HTTP error), the agent will log it and try again or switch to direct download. By providing the agents with clearly defined requests (bounding box, date window, variable of interest) and an environment with valid credentials, we reduce errors. Each retrieval will be first tested on a single day to ensure the format is as expected.

Finally, we'll document example commands like those above in our README for transparency. This ensures anyone (judges included) can reproduce data access with minimal fuss.

What to do next: Write the Python scripts or notebook sections to perform these queries (using `requests` or `harmony-py` as appropriate); test each on a small temporal slice; store the Earthdata credentials securely (in `.env` or netrc) for the runtime environment; ensure the outputs are correctly subset (open one file and plot to see that it indeed covers the ROI and has valid data).

Feature-Engineering Playbook

With our data in hand, we derive a set of environmental **features** that likely correlate with shark foraging activity. Here we define each feature, how it's calculated (formula/algorithm), and reasoning:

1. Sea Surface Temperature (SST) – and Thermal Fronts

- **SST (°C):** Direct value from MUR analysis (after conversion to °C). Useful as sharks prefer certain temperature ranges (e.g. whites ~15–22 °C [15]). We will also include **SST anomaly** if possible (difference from seasonal mean, to highlight unusual warm/cold events).

- **SST Gradient Magnitude (°C/km):** Captures thermal fronts. We compute horizontal gradient on the 0.1° grid: $$G_{SST} = \sqrt{\left(\frac{\partial T}{\partial x}\right)^2 + \left(\frac{\partial T}{\partial y}\right)^2},$$ where $\partial T/\partial x$ and $\partial T/\partial y$ are differences between neighboring grid cells (taking care with lat scaling). We convert gradient to °C per km (at our latitude ~1° ~ 85 km).

Front Detection: We define a binary **Front Flag** = 1 if G_{SST} exceeds a threshold. Literature suggests using ~0.25–0.3 °C per km to detect significant fronts [49]. For example, Belkin & O'Reilly's front detection used ~0.05 °C per 0.25 km (~0.2 °C/km) for strong fronts. We will start with **0.3 °C/km** as threshold [49] and adjust if needed (this would mark sharp Gulf Stream edges, etc.). We may refine by requiring a certain spatial extent (e.g. at least 2 adjacent pixels exceed threshold, to avoid single-pixel noise).

Output features: `sst` (value), `sst_anom` (if computed), `sst_front` (flag 0/1), and possibly `sst_grad` (continuous gradient strength for use in ML model as well).

2. Chlorophyll-*a* – and Chlorophyll Fronts

- **Chlorophyll-*a* (mg m^{-3}):** from PACE or MODIS. We will likely take **$\log_{10}(\text{Chl})`** as a feature, since predator presence often correlates with moderate log-chl (too low = little food, too high = maybe murky/low visibility or unbalanced ecosystem) [11]. For instance, we expect sharks in productive shelf waters typically where Chl ~0.2–2.0 mg/m^3 (log10 ~ -0.7 to 0.3). We will standardize or clip extreme values (if any algal bloom >10 mg/m^3, we cap it or treat as outlier).

- **Chlorophyll Gradient:** Similarly to SST, compute gradient magnitude of log-chl or chl. Fronts in chlorophyll can indicate boundaries of productivity (e.g. upwelling fronts). Basking sharks feed along chl fronts [50] [42], and generally, chl fronts often coincide with thermal fronts but can also occur from river plumes etc. We threshold a **Chl Front Flag** if gradient > some value (we might calibrate from data distribution – say a quantile like top 5% gradient). Example threshold might be ~50% change over 10 km. The basking shark study noted association with "chl-a front gradient density" measures [51]. For simplicity, we'll generate a binary mask of top X% chl gradients as front zones.

- Additionally, we can compute a **chlorophyll anomaly** (current value vs a monthly climatology). However, given PACE is new, we may not have a climatology. Alternatively, use MODIS climatology for baseline. This could highlight bloom anomalies that might attract prey.

Output: `chl_log`, `chl_front` (flag), and possibly `chl_grad`.

3. Sea Level Anomaly & Eddy Indicators

- **Sea Surface Height Anomaly (SSHA, m):** from gridded altimetry (e.g. relative to mean sea level). Eddies show up as +SSHA (anticyclonic, clockwise in N. hemisphere) or -SSHA (cyclonic) circulations. We include SSHA itself as a feature; sharks might favor one eddy polarity (e.g. whites in warm anticyclones [6]).

- **Okubo–Weiss (OW) Parameter (s^{-2}):** an Eulerian metric to identify eddies [52]. We compute OW = $s_n^2 + s_s^2 - \omega^2$ [53] [52], where $s_n = \partial u/\partial x - \partial v/\partial y$ (normal strain), $s_s = \partial v/\partial x + \partial u/\partial y$ (shear strain), and $\omega = \partial v/\partial x - \partial u/\partial y$ (vorticity) [54]. For a geostrophic flow from SSH, we can derive u, v (approximately, $u \approx -\frac{g}{f} \partial(\text{SSH})/\partial y$, $v \approx \frac{g}{f} \partial(\text{SSH})/\partial x$ in mid-latitudes). Then we calculate OW on the grid.

In eddy cores, vorticity dominates (ω large) and OW becomes negative; in strain-dominated areas (background flow), OW positive [52]. We will generate an **Eddy Core Flag** = 1 for contiguous areas where OW < -W0 (negative threshold). Common practice is to choose W0 such that area of OW<0 matches typical eddy sizes, or use one standard deviation: e.g. W0 = -0.2 * (std of OW) or similar criterion [55] [56]. We'll experiment to ensure that known eddies (like the warm-core rings north of Gulf Stream) are captured. We also classify **cyclonic vs anticyclonic**: we can use the sign of SSHA at the OW-core pixel (neg SSHA likely cyclonic, pos SSHA anticyc in our region) or compute vorticity sign directly. We may label each eddy pixel with +1 (anticyclonic eddy), -1 (cyclonic eddy), 0 (no eddy).

- **Closed Contour Eddy Detection:** As a cross-check or simpler approach, we can identify eddies via closed SSH contours [57]. We find local extrema in the SSH field (peaks and lows above some

amplitude, say >10 cm deviation) and flood-fill outward until contour curvature changes. Due to time, we likely rely on OW or even a provided eddy dataset (if we find one) rather than implementing full contour tracking.

- **Eddy Distance/Edge:** We can derive a feature "distance to nearest eddy edge" or a boolean "inside eddy vs outside." Since some studies indicate sharks might forage at eddy **edges** (convergence zones of eddies can collect biomass) [12], we should capture that. Using the eddy flag from OW, we can erode/dilate the regions: e.g., define eddy *core* as OW<<0 area, and *edge* as a surrounding ring perhaps where OW is just turning positive or max vorticity at edge. For simplicity in features:

- `eddy_flag` (1 if inside any eddy core).
- `eddy_cyc` (1 if inside cyclonic eddy, else 0).
- `eddy_anti` (1 if inside anticyclonic).
- `dist_eddy_edge` (km to nearest eddy boundary) – optional continuous feature.

- **Eddy Kinetic Energy (EKE, m^2/s^2):** Also computable from altimetry: $EKE = 0.5*(u'^2 + v'^2)$ for velocity anomalies. But since our model already has eddy flags and current speeds, EKE might be redundant. We might include it if easily derived.

Outputs: `ssh_anom`, `eddy_flag` (and/or separate cyc/anti flags), possibly `OW_val` for ML continuous input (though highly correlated with eddy_flag essentially).

4. Surface Currents & Derived Features

- **Current Speed (m/s):** Calculate $speed = \sqrt{u^2 + v^2}$ from OSCAR currents. This indicates how dynamic the water is. Some species might avoid strong currents to save energy, others may use them as highways. We also consider:
- **Current direction** relative to, say, coastline or front orientation. But representing direction in a model is tricky (we could use sine/cosine of heading).

- Perhaps simpler: split into `u_current` and `v_current` as features (so the model can learn if say northward flow correlates with sharks in our region migrating north etc.).

- **Current Divergence (1/s):** Compute $\nabla \cdot \mathbf{u} = \partial u/\partial x + \partial v/\partial y$. Convergent zones (negative divergence) can accumulate floating matter (and maybe prey). We include `curr_divergence` – likely most useful to identify features like converging fronts or eddy edges. Oscar's coarse resolution might make this noisy; we will smooth it a bit or just use sign.

- **Relative vorticity:** $\omega = \partial v/\partial x - \partial u/\partial y$ from OSCAR could also be computed (in fact, we will compute it for OW anyway). Vorticity indicates local rotation (small eddies, shears). Could include as feature if not redundant with OW/eddy.

Because OSCAR is 5-day, these features will be somewhat smoothed. But major structures (like the Gulf Stream core – strong u, or eddy circulations – high vorticity) will appear.

Outputs: `curr_speed`, `curr_div`, `u_current`, `v_current`.

5. Sea Surface Salinity (SSS)

- **SSS (PSU):** Direct from SMAP. Useful for identifying freshwater plumes. E.g., a low SSS reading might indicate a river plume with high nutrients and productivity (feeding ground). We might also derive:
- **SSS anomaly** (difference from a regional mean or climatology).

- **SSS front/gradient** (like at a river plume boundary). However, SMAP resolution is coarse and errors can be large, so a salinity gradient feature may be noisy. We might try it if time permits, focusing near known plume areas (e.g., mouth of Chesapeake or St. Lawrence if in ROI).

- We will treat SSS primarily as a stable indicator: e.g., whether the location is within a low-salinity water mass. Many sharks (bull sharks, some coastal spp.) tolerate very low salinity, but white sharks generally stay oceanic (full salinity). So for our default species (white shark), this may not be highly predictive except perhaps indirectly (for instance, a low-salinity patch could indicate a productive estuarine outflow area with fish – maybe juvenile whites could be near? We'll see).

Output: `sss` (and maybe `sss_anom`).

6. Precipitation / Recent Rainfall

- **7-day Rain Accumulation (mm):** Sum of IMERG rain over the past week at each location. The idea: a high value could signal a major rain event -> possible river runoff soon after. This is a bit speculative for open-ocean sharks, but relevant for coastal/estuarine usage. If focusing on white sharks, they don't go far into estuaries, so rain might have minor effect. We include it as an optional exploratory feature or for other species (like bull sharks which go upriver, or if modeling coastal shark hotspots for generality).

- If not using continuous rain, we could simply tag if an area had >50 mm in last week (heavy rain flag).

- We must be cautious as this adds another dimension of potential noise. It's optional and we'll evaluate correlation; we might drop it in final model if negligible.

Output: `rain_7d` (or `rain_anom`).

7. Temporal Features (Lags and Averages)

Shark foraging may respond to conditions with some lag (e.g., a bloom happens, prey fish increase a few days later, then sharks arrive). To capture this, we will generate **lagged features**: - For each variable above that can change significantly in days (SST, Chl, maybe currents): include values from **3 days before, 7 days before, 14 days before** (0, 3, 7, 14-day lag as suggested). This means if a shark is at time *t*, we append e.g. SST(t-3) as a feature. This can help the model detect trends (cooling vs warming, blooming vs declining productivity). - We'll also try a **7-day moving average** for some fields (Chl 7-day mean, SST 7-day mean) to smooth out short-term noise. - We must align these carefully with shark observation times (e.g., if a shark

location is on July 10, for a 7-day lag we need data from July 3). Our data access plan will fetch continuous series, so that's fine.

Potential features from this: - `chl_log_lag3, chl_log_lag7, chl_log_lag14`, similarly `sst_lagX`, etc. - We might not lag the binary front flags (those are very ephemeral signals); lagging continuous fields is more sensible.

8. Composite Indices and Habitat Preferences (for Baseline)

While the ML model will learn from the raw features above, we also plan a simple **rule-based index**. For that, we'll define some thresholds: - **Preferred SST range:** 15–22 °C (for white shark) [15]. We can make an index component that is 1 if SST is in that range (0 if not, or perhaps a linear scale if outside by a bit). - **Preferred Chl range:** say 0.2–1.5 mg/m³ (justified by typical productive coastal values without being a red tide). Index component = 1 if in range, 0 if outside some wider range (or decrease if very low/high). - **Front/Eddy proximity:** add points if `sst_front=1` or `chl_front=1` near the location. Also if `eddy_flag=1` (especially anticyclonic eddy for white shark based on Gaube) or within, say, 50 km of an eddy edge. - Weights: we might weight thermal fronts and moderate Chl highest (since literature shows strong influence [5]), eddy presence next, then SST range.

For example, a simple **Habitat Suitability Index (HSI)**:

$$\text{HSI} = w_1 \cdot f_{\text{SST}} + w_2 \cdot f_{\text{Chl}} + w_3 \cdot f_{\text{Front}} + w_4 \cdot f_{\text{Eddy}} + w_5 \cdot f_{\text{Other}},$$

where each f is normalized 0–1. We'll set, say, $w_1=w_2=0.3$, $w_3=0.2$, $w_4=0.2$ as an initial guess (SST and Chl 30% each, fronts and eddies 20% each). We will define: - $f_{\text{SST}} = 1$ if SST in [15,22]°C, taper to 0 at 10°C and 28°C (outside likely range) – basically a triangular membership. - $f_{\text{Chl}} = 1$ if log10(Chl) in [-0.7, 0.2] (~0.2–1.6 mg/m³), taper to 0 at -1 (0.1 mg) and 0.7 (5 mg). - $f_{\text{Front}} = 1$ if any front flag =1 within, say, 0.2° of location (to allow slight mismatch), else 0. - $f_{\text{Eddy}} = 1$ if inside favorable eddy zone: For white sharks, based on evidence, that might specifically be **inside a warm-core anticyclonic eddy** [14]. So we set 1 if `eddy_anti=1`, else 0. (For generality, we could make it 0.5 if cyclonic to not exclude, but leaning on Gaube's finding). - f_{Other} could include depth if relevant (e.g., sharks might prefer shelf depth <200m for foraging). If our ROI includes deep ocean vs shelf, we might incorporate bathymetry (from a static dataset). Given interest in coastal vs pelagic zones, we may add a binary for "on continental shelf (depth < 200m)" since many foraging events of white sharks occur near shelf edge. We can fetch depth from e.g. ETOPO1 static grid.

This HSI will be used as a baseline to compare to ML.

We will cite values as needed (like SST range from NOAA: 59–72°F ~ 15–22°C [15], which we did). The specific weights and ranges we might refine with a few iterations or based on literature if available.

Citations within feature logic: - Basking shark & front association [5] (justifies front usage). - White shark & warm eddy [14] (justifies focusing on warm-core eddies). - Preferred SST [15].

Feature Summary: After processing, each shark observation will have columns for each feature: e.g. `sst, sst_front, chl_log, chl_front, eddy_flag (cyclone vs anti separate),`

`curr_speed, curr_div, sss, rain7d, plus lagged versions`. This rich feature set (likely 15–25 features) will feed the ML model.

What to do next: Implement the computation of each feature in code – likely using `xarray` for gradients (e.g. `xr.apply_ufunc` with `np.gradient` on SST), OpenCV or skimage if needed for edge detection (but Sobel can be done via convolution filters in numpy too). Test on a known image (e.g., does our front flag align with known front locations?). Prepare visual examples for documentation – e.g., map of SST with front contours, map of SSH with eddy flags – to ensure the methods are working properly. Decide on final threshold values and document them in the code for transparency (with citations as comments where applicable).

Label Strategy & Evaluation Data

To train and evaluate our model, we need **ground truth** of shark presence in space and time. We will leverage open **shark tracking data** and construct a labeled dataset with careful attention to quality, balance, and ethical use:

- **Tracking Data Source:** We plan to use **OCEARCH** open-source tagged shark tracks for our target species (default: *Carcharodon carcharias*, great white shark). OCEARCH provides timestamps and coordinates of tag pings on their website. For example, mature female white sharks like "Lydia" have well-known tracks in the NW Atlantic (from Cape Cod to open ocean). If available, we'll download CSVs of those tracks. Alternatively, the **Movebank** or other research databases might have shared tracks (with permission). Since OCEARCH data is public (they display it on a map), we assume we can use it for hackathon purposes, crediting OCEARCH.

- **Label Definition:** Each track location is a **positive instance** (shark present & presumably foraging-capable) at a certain time and place. We consider that as a "foraging opportunity" label = 1. However, not every presence means active foraging – but we assume if a shark is in an area, it's at least *searching* for food.

- **Negative (Background) Sampling:** To train a binary classifier (presence vs absence), we need examples of locations/times with no shark. We will create pseudo-absence points:

- We define a spatial-temporal domain (e.g., NW Atlantic, June–Aug 2023). Within that, for each day or each track point, we will sample random locations (and times) where no tracked shark was present. This gives negative examples.
- We must avoid sampling "absences" in places/times where another tagged shark was actually present (to not label a presence as absence). So we will compile all track points from possibly multiple sharks, then sample from the gaps.
- We will likely sample, say, 5× more negatives than positives (to have enough, but not too imbalanced). If we have ~1000 shark positions, we might sample ~5000 background points across the region and period.
- **Spatial constraints:** We should constrain negatives to the overall area sharks *could* plausibly roam. For example, we won't sample on land (obviously) or in completely unrelated oceans. Also, if our species is coastal, we'd focus negative samples in coastal areas as well, to challenge the model. For white sharks, they roam both coastal and pelagic, so we can sample anywhere within the general

bounding box (maybe excluding too shallow estuaries if they never go there – but white sharks sometimes do go into brackish bays chasing fish, so maybe include).

- **Temporal constraints:** Use similar date distribution. If our shark data spans June–August, we'll also sample absences throughout that period, not outside it (so the environment conditions are comparable).

- **Deduplication & Temporal resolution:** Many shark tags ping irregularly (some every hour when at surface, some once per day). We will **resample the track to daily** to match our environmental data frequency. If a shark has multiple pings in one day within our ROI, we might use only one (or an average location) to avoid over-representing that day. Also, multiple sharks in same area/time – we could keep those as separate positives (since they indicate good habitat if more sharks are there), but careful if they cluster (it could overly weight that event). We might down-weight or filter if 10 pings come from effectively same spot/time (e.g., a cluster around a feeding event).

- **Data Schema:** We will create a table: each row has `timestamp, latitude, longitude, species, shark_id, label`. Label = 1 for actual track point, 0 for a sampled background point. Additional fields for group splits (like fold number, region label) will be added for cross-validation.

- **Cross-Validation Strategy:** To robustly evaluate, we avoid training and testing on points that are too similar:

- **Spatial CV:** We will partition data by region. For example, one fold might treat coastal shelf points as test and offshore points as train, and vice versa. Or partition by latitude bands or use clustering (k-means on coords to define groups). This checks model's ability to generalize to new areas.
- **Temporal CV:** We could do train on June–July, test on August (though environment conditions shift, but could test short-term predictive power).
- **Individual-based CV:** Perhaps most important – ensure that if multiple individuals are tracked, we test on sharks not seen in training. This prevents leakage of individual behavior patterns. For example, use Shark A's track as test and Shark B/C's as train, then rotate. This is feasible if we have a handful of individuals. If only one shark's data, we rely on spatial/temporal splits.

- We will likely use a combination: e.g., 5-fold CV where folds are defined by individual sharks (if ≥ 5 sharks), or by clusters of track segments.

- **Evaluation Metrics:** We will use **AUC-ROC** and **PR AUC** for classification performance (given class imbalance, PR AUC is informative). Also **accuracy at optimal threshold** and perhaps **True Skill Statistic (TSS)** which is common in habitat modeling. We'll track performance for each fold and overall.

- **Ethical & Licensing Checks:** We ensure that using the tracking data does not violate any terms:

- OCEARCH data will be attributed; it's collected from wild sharks with minimal harm (they tag and release). We need to make sure not to misuse it (e.g., we won't publicly pinpoint sensitive locations like breeding grounds beyond what OCEARCH itself publishes).
- If using any other source (e.g., academic dataset), we ensure it's open or we have permission (for the hack, we'd lean on publicly posted data).

- We also anonymize or aggregate if necessary. For instance, if some track data have super fine resolution that could be sensitive (like revealing exact near-shore locations where sharks frequent beaches), we might aggregate to 0.1° grid to obscure exact coordinates. Since we are anyways modeling at 0.1°, we are not giving pinpoint locations beyond that scale.

- **Label Balance & Weighting:** We will maintain a reasonable balance in training (we can upsample minority if needed or give class weight). Likely positives will be << negatives in count. We'll handle this in model training by using appropriate class weights or focusing on ROC/AUC which is insensitive to imbalance.

- **Alignment with Environment:** We align each shark point's time to nearest daily environmental day (as mentioned in interop). If a shark ping time is midday, we use that day's satellite data (which might correspond to a composite of that day). If a shark moved tens of km within a day, our daily environment may not capture the fine-scale movement, but that's accepted noise.

- **Background label sanity:** A challenge: we don't actually know the shark *wasn't* in some background sample area unless a tag was there to say no. We assume with wide-ranging animals that unobserved locations are likely absence given limited population. There is potential presence of untagged sharks. But since we only have tagged individuals, we treat it as pseudo-absence. This is standard in species distribution modeling (presence-only or presence-background approaches). We may consider using methods like MaxEnt (which use presence vs background differently), but for now we treat as a binary classification with pseudo-absences.

- **Augmenting labels:** If available, we could also bring in **other species** to increase data volume (e.g., tracks of tiger sharks or makos in the Atlantic). But mixing species complicates labels because each has different preferences. We prefer focusing on one species for clear signal. Another approach: label not just binary presence, but perhaps "foraging event" vs other movement. If we had high-resolution tags with accelerometers (maybe not in OCEARCH data), we could label actual feeding events. But that's beyond our scope due to data limitations. So binary presence in suitable habitat is our proxy.

- **External Validation:** If possible, we'll test the model predictions qualitatively against known shark hotspots (e.g., does our prediction map highlight Cape Cod bay in summer – known white shark hotspot – and shelf break areas where seal colonies or fish schools occur?). If we have any independent sightings data or fisheries catch data, we could compare, but likely not within hack timeframe.

In summary, we create a labeled dataset of shark presence and pseudo-absence with careful partitioning to evaluate generalization. This will allow us to train the model and assess performance under realistic conditions (predicting for new areas or new times).

What to do next: Obtain the actual track data. Clean it (remove obvious errors or telemetry gaps). Create the negative samples (write a script to randomly generate lat-lon-date within bounds, ensuring not within X km of any shark point). Merge with environmental features by date and location (spatial join – likely nearest grid cell to the shark point). Split into folds. Double-check that each fold's positives are spatially separated if that's the plan (visualize fold assignment on a map). This sets the stage for modeling.

Modeling Blueprint

We will pursue a two-pronged modeling approach: a **rule-based baseline** for interpretability and an **ML model** (gradient boosting) for predictive power. Additionally, we outline possible stretch models if time permits.

Baseline Rule-Based Model

Our baseline is a transparent **Habitat Suitability Index (HSI)** as described. This isn't a learned model but a weighted sum of boolean/normalized rules:

1. **Define Criteria:** Using the thresholds in the Feature Playbook:
2. SST in [15, 22] °C → ideal (score 1), gradually to 0 outside [10, 28] °C [15] .
3. Log10(Chl) in [-0.7, 0.2] (~0.2–1.6 mg/m^3) → ideal (score 1), 0 below -1 (0.1 mg) or above 0.7 (5 mg) (sharks avoid ultra-oligotrophic and maybe very eutrophic waters).
4. Front present (SST or Chl) → add bonus (score 1 if any front within ~20 km).
5. Inside anticyclonic eddy or within ~1 eddy radius of one → add bonus (score 1).
6. Depth < 200 m (on continental shelf) → for white sharks, likely positive (many feeding areas near shelf). But also some go deep ocean (e.g., Sargasso) for other prey. We might skip depth or include lightly.
7. **Weighting:** As suggested: SST (0.3), Chl (0.3), Front (0.2), Eddy (0.2). Depth if included maybe 0.1 taken from others.

The HSI for location *i*:
$$HSI_i = 0.3 \cdot f_{\text{SST},i} + 0.3 \cdot f_{\text{Chl},i} + 0.2 \cdot f_{\text{Front},i} + 0.2 \cdot f_{\text{Eddy},i} \,,$$
where each f ∈ [0,1] as defined above. This yields 0–1. We could map that to a binary by choosing a threshold (e.g., HSI > 0.5 = likely shark presence).

1. **Calibration:** We can calibrate on part of the data: e.g., ensure that known presence points have higher HSI on average than random points. If not, adjust weights. This baseline will likely be far from perfect, but it sets an intuition (and we can present a quick example: "Our index gives high score in areas where sharks were indeed found, and low where they weren't, roughly").
2. **Use:** We will compare the ML model's performance gain over this baseline. Also, HSI can be visualized as a continuous map of "habitat suitability", which is great for storytelling (even without ML).

Machine Learning Model (Gradient Boosting on Tabular Features)

We choose an ensemble tree method (e.g. **XGBoost** or **LightGBM**) for the main predictive model because: - It handles mixed data types and missing values gracefully, - It's fast to train and easy to interpret via feature importance or SHAP, - With our feature count (~20–30), a tree model is appropriate (not too high-dimensional for deep learning, and dataset likely small – a few thousand points).

Features Input: All features described (continuous and binary flags). We will also add interactions implicitly by tree splits. No manual polynomial features needed.

Data Preparation: - **Normalization:** Tree-based methods don't require standard scaling. We might still scale some features for uniformity in feature importances plotting, but it's optional. We will, however, take log of highly skewed ones (Chl already log, rain maybe log if used). - **Missing data:** If an environmental feature is missing (e.g., no optical data due to cloud), XGBoost by default can handle missing by learning a separate route. We will enable that. Alternatively, we can fill with a mean or flag. A robust approach: create a parallel feature indicating "Chl data available" (1/0) and fill missing chl with mean. Then tree can use the flag. But given we will composite away most missing, and XGBoost's split can send missing values one way, we might not need explicit flags. We will still be mindful e.g. if 20% of points lack Chl due to clouds, we keep that info (the model might effectively learn to interpret missing as average conditions).

- **Training Procedure:**
- Use cross-validation as defined (spatial or by shark). Possibly we do a 5-fold CV and take average metrics. If time, do a grid search of hyperparameters within CV.
- We'll likely use XGBoost with default-ish params initially:
  - `max_depth ~ 5`, `n_estimators ~ 100` (we can increase to 300 if quick), `learning_rate ~ 0.1`.
  - Possibly `subsample ~ 0.8` and `colsample_bytree ~ 0.8` for regularization.
  - Use `scale_pos_weight` or sample weights to account for class imbalance (e.g., if 1:5 ratio, set weight=5 for positive class to balance).

- Optimize on AUC or Brier score. We might simply monitor validation AUC.

- **Validation:** Evaluate on each fold's held-out set:

- Compute ROC curve, AUC.
- Compute confusion matrix at chosen probability threshold (maybe threshold that gives a certain sensitivity or maximizes TSS).
- Compute PR curve and AP (average precision).

- If multiple models (like if we test LightGBM vs XGBoost or different feature sets), use the same CV to compare.

- **Ablation Experiments:** To demonstrate NASA data value:

- We will train multiple versions:
  1. Full model with all features.
  2. Minus ocean color (no Chl features).
  3. Minus SST (no temp/front features).
  4. Minus SSH/eddy features.
  5. Minus currents.
  6. Minus salinity/precip (if included, though these might already show up as low importance).
- Measure drop in AUC or increase in error for each. E.g., "Removing chlorophyll data caused AUC to drop from 0.85 to 0.75" – a significant drop highlighting the importance of NASA's PACE mission data in predicting shark habitats (a great justification for *Best Use of Data*).

- We'll tabulate these results or at least note the largest drops. Likely expect SST/Chl removal to hurt most, maybe eddy removal too if sharks indeed follow eddies.

- **Interpretability:** We will apply **SHAP (SHapley Additive exPlanations)** values on the trained model to quantify each feature's contribution:

- Rank features by mean |SHAP|. We anticipate seeing maybe SST or Chl as top, or front flags if they are very informative.
- We'll create partial dependence or SHAP dependence plots for key features to see direction of effect (e.g., how probability of presence changes with SST – likely a peak in the middle range).
- This helps validate that the model learned reasonable patterns (and we can compare with our assumptions: e.g., if model says extremely high Chl gives low probability, that matches expectation that sharks avoid algal bloom anoxic areas).

- If some weird feature shows up (like say rain_7d high SHAP meaning, which might indicate something spurious), we can investigate and mention or drop it.

- **Edge-case prediction:** We will test the model on certain known scenarios as sanity check:

- Predict on a grid for a given day: e.g., map of predicted probability on August 1, 2023 across ROI. Does it highlight expected areas (say shelf edge, Gulf Stream ring boundaries)? We plan to use these maps in the demo.

- Check specific track: feed one shark's path to model and see if model would have predicted those points high. Conversely, pick a random path and ensure low prediction (for trust).

- **Stretch Goal – Spatiotemporal ML:** If time allows, consider using a Convolutional Neural Network or LSTM on raster time series:

- We could frame it as a sequence labeling or segmentation on a map. But given limited time, we focus on tabular GBDT. We'll mention CNN as a future idea (like using U-Net on daily maps to directly segment favorable areas, or a 3D CNN on time stack).
- Also mention possibly using a **spatiotemporal GBDT** by including lag features (which we are doing).

- If extremely ambitious, we might try a simple neural net on sequential data for one shark (but probably unnecessary).

- **Model Deployment Consideration:** The final model could be saved (pickle or Booster file) and used to predict on new data (e.g., we can feed tomorrow's forecast or yesterday's satellite data to predict where sharks might be – a potential real-world use for fisheries or public safety). While we won't actually integrate forecasts, we mention the capability (leading to potential *Galactic Impact* – mitigating shark-human conflict or aiding conservation).

- **Metrics & Expectations:** We expect baseline HSI might get AUC ~0.6–0.7 (better than random). ML might achieve AUC ~0.8–0.9 if environment factors strongly dictate presence (as some papers have found high explanatory power in environment [20]). We should be careful not to overfit – hence cross-validation and using spatial folds which is a stringent test (likely yielding more modest performance, but realistic). We will report metrics per fold and average to show robustness.

- **Calibration:** We might calibrate the probability output (via Platt scaling or isotonic) if needed to interpret as probability of presence. But relative ranking is more important for now.

Summary: Our ML model will output a probability that a given location/day is a shark foraging habitat. We'll compare it to actual presence points to measure success. We will also show that **features derived from NASA data indeed drive the model** (and thus shark behavior), meeting the challenge's core objective.

What to do next: Implement the XGBoost training in a notebook: load feature table, split into folds, train and evaluate. Use scikit-learn or XGBoost's built-in CV. After initial run, generate SHAP values (using `shap.TreeExplainer`). Plot feature importance. Save the model. Run ablations by dropping columns and retraining quickly (since dataset not huge, this is fine). Gather results in a concise form for reporting. Validate model predictions vs some intuitive expectations as final check.

Robustness & Edge-Case Plan

Our solution should be robust to data issues and unexpected conditions. We enumerate potential risks and how we mitigate them, ensuring the system (and its agents) can handle edge cases gracefully. Table 2 summarizes the major risks:

Table 2. Risks, Causes, Detection, Mitigation, and Demo Fallbacks

Risk (Impact)	**Cause** (Why it happens)	**Detection** (How we know)	**Mitigation** (What we do about it)	**Demo Fallback** (If all fails)
Optical data gaps – PACE chlorophyll missing (Moderate)	Clouds, high glint, or polar night cause no optical retrievals	Large NaN areas in Chl layer; quality flag indicates cloud [41]	*Mitigation:* Use SST & SSH features instead when Chl is missing (model still has those inputs). Fill small gaps with recent data (e.g. previous day's Chl) or a regional mean. We also composite 8-day Chl to reduce cloud gaps.	In demo maps, overlay SST front locations if Chl map is blank; explain that model defaults to SST/eddy cues in cloudy periods.
SWOT not covering ROI/ time (Low)	SWOT's orbit may not have passed our area on a given day	No SWOT granule for date in CMR; or track coverage map shows gap	*Mitigation:* Use gridded altimetry (MEaSUREs) for continuous coverage [33]. Our pipeline checks if SWOT data exists; if not, it automatically uses fallback dataset for SSH.	Skip showing SWOT-specific output; focus on altimetry-derived eddies which we have for all days.

Risk (Impact)	**Cause** (Why it happens)	**Detection** (How we know)	**Mitigation** (What we do about it)	**Demo Fallback** (If all fails)
Harmony API failure halting data fetch (High)	Network issues or service downtime; agent lacking proper format	Error codes from curl/ Python (non-200 HTTP or timeouts)	*Mitigation:* Implement retry logic with exponential backoff. If Harmony repeatedly fails, switch agent to direct S3 download via s3:// links using stored AWS creds (Earthdata provides temporary S3 creds [58]). Also pre-download critical data ahead of demo as cache.	Have a local copy of subset data as backup. For the demo, we'll note how it *would* fetch but use cached files to proceed if live fetch fails (so demo flow continues).
Misalignment of data grids affecting features (Moderate)	Regridding or coordinate mis-match (e.g., one dataset uses 0–360 lon)	Strange artifacts in feature maps (e.g., front lines shifted from SST map); low correlation between related features that should correlate	*Mitigation:* Use consistent grid definitions via xarray. We test alignment by plotting layers over each other. We also include assertions in code (e.g., lat arrays of all datasets match after regridding). If misaligned, apply shifting or use xarray `.interp` with dataset's native coords.	In presentation, if a feature looks off, we'll verbally clarify it's an alignment issue and focus on correctly aligned ones. (We'll avoid showing any obviously misaligned result by thorough testing beforehand.)

Risk (Impact)	**Cause** (Why it happens)	**Detection** (How we know)	**Mitigation** (What we do about it)	**Demo Fallback** (If all fails)
Model overfitting or strange predictions (Moderate)	Small training sample or data leakage between train/ test	Detected if training AUC >> test AUC, or if CV fold variance is huge; or if predictions in obviously unsuitable areas are high	*Mitigation:* Use spatial/ individual cross-validation to ensure independence. Limit model complexity (depth of trees, etc.). We will also inspect predictions – if model says sharks likely in absurd places (e.g. open ocean with 0 Chl and cold water), we'll know something's off. Retrain with adjusted features or regularization if needed.	If ML model seems unreliable, fall back to the simpler HSI index for demo (less accurate but interpretable). We can present the idea that ML would refine it, without depending on shaky predictions live.
Shark track data issues (Low)	Tag transmission errors or sparse data (e.g., long gaps, outliers)	Outlier points on land or abrupt 1000 km jumps; big time gaps seen in data	*Mitigation:* Pre-process tracks: remove impossible locations (speed > threshold), interpolate across short gaps or omit overly gap segments. For sparse data, we augment with other individuals if possible. Also, our model doesn't assume continuous coverage – it looks at point-wise environment, so irregular timing is okay as long as each point is valid.	If track data is too sparse to demo time alignment, we'll simulate a plausible track path on our map (e.g., plot a hypothetical shark path through high-HSI areas) to illustrate concept. Clearly state it's hypothetical if so.

Risk (Impact)	**Cause** (Why it happens)	**Detection** (How we know)	**Mitigation** (What we do about it)	**Demo Fallback** (If all fails)
Agent pipeline logic bugs (Moderate)	Agents (scripts) might have coding errors, e.g., index off-by-one, memory leaks	Would show up as crashes or clearly incorrect outputs (e.g., no data merged for certain dates) during test runs	*Mitigation:* Test each agent in isolation with sample input. Incorporate unit-test-like checks (e.g., after subsetting agent runs, verify the output file covers requested dates; after feature computation, verify no all-zero arrays, etc.). We also use small chunks to ensure no memory overflow. Logging each step's summary (e.g. "50 shark points processed for fold 1") to catch anomalies.	In the worst case, run parts of pipeline manually in a notebook during demo if an agent fails. (We'll have Jupyter ready to quickly do a step by code if needed, acting as a human agent backup.) And have precomputed results to visualize if live pipeline stops.
Unexpected environment scenario (Low)	Conditions outside training distribution, e.g. a hurricane cold wake or marine heatwave	Model might extrapolate weirdly (e.g., never saw SST 30°C, not sure how to respond)	*Detection:* If such an event occurred in our timeframe (we'll know from data exploration – e.g., was Summer 2023 unusually warm?), and see if model yields extreme probabilities.	*Mitigation:* Expand training data if possible to include more variability (maybe include multi-year or global data). Place reasonable bounds in the model (e.g., if SST > 28, cap the effect as very low suitability for whites – since beyond physiological range).

By anticipating these issues, we ensure our hackathon project runs smoothly and the results are reliable. Importantly, each agent (data fetcher, processor, etc.) is equipped with the "help" it needs – whether fallback code paths or pre-checked assumptions – to work **flawlessly with minimal human intervention**.

What to do next: Finalize implementation of each mitigation in code (e.g., write the retry mechanism for downloads, test the alignment checks). Perform a full "dry run" of the pipeline on a small date range to see if any of these risks manifest and adjust accordingly. Prepare backup data and visuals for the demo just in case.

Pipeline & Agents

We design our pipeline as a sequence of modular **agents** (automated scripts/notebooks), each responsible for a stage of the process. This modular design makes the solution *fully agentic* – each part can run with minimal human input, and if one fails, others can be adjusted or rerun independently. The pipeline and agent responsibilities are:

1. Data-Scout Agent (Search & Catalog):
- *Role:* Query NASA CMR for relevant data files. Essentially "scouts" which data to get.
- *Tasks:* For each dataset (PACE, MUR, etc.), use the API to list granules for our ROI/time. Filter by our date range and bounding box. Output a **catalog JSON** (or CSV) of granule IDs/URLs for each data source.
- *Example:* It finds daily MUR SST files for Jun–Aug 2023 (e.g., 92 files) and writes `MUR_files.txt`.
- *Agent Implementation:* Python script using `earthaccess` /requests, logging any missing dates.
- *Autonomy Aids:* Provided with query parameters (short_name, date range, bbox) in a config file, so it doesn't need human input. We supply Earthdata auth via environment (.netrc).
- *Output:* List of granule URLs per dataset for the next agent.

2. Subset-Fetcher Agent (Download & subset):
- *Role:* Retrieve the data from cloud and subset to our ROI to minimize processing load.
- *Tasks:* Reads the catalog from Data-Scout; for each entry, calls **Harmony API** (or direct download if needed). Monitors for success (status OK or file saved). Saves files in `data/raw/<dataset>/...` directory.
- *Example:* Downloads `sst_20230710.nc` containing SST for our region on that date, rather than global 2GB file.
- *Agent Implementation:* Could be a shell script with cURL commands or a Python loop with Harmony-Py. Uses parallelization (maybe 3-4 threads) to speed up.
- *Autonomy Aids:* It has error catching – if a download fails, it logs and continues (to try others). If certain file is crucial, it retries. It can skip existing files (so we can rerun it without duplicating work).
- *Output:* Subsetted data files ready for processing. Possibly also merges multiple days of some datasets into one file (e.g., combine all IMERG daily into one timeseries file for convenience).

3. Ocean-Features Agent (Feature computation):
- *Role:* Transform raw data into feature layers like fronts, eddies, etc.
- *Tasks:* Load the subset files (likely using xarray for multi-file datasets). Compute gradients for SST and Chl, run edge detection, compute Okubo-Weiss from SSH, etc., as described earlier. Also include any static data merge (like depth from a static file, if used).
- *Example:* Opens `sst_*.nc` files as one Dataset, computes gradient mag array, flags where > threshold, writes out `sst_front_daily.nc`. Similarly, processes others.
- *Agent Implementation:* Python (xarray, numpy, possibly OpenCV for edges). Could be one combined script or separate small ones per feature type. We likely do it in one notebook/script for convenience: load all data into memory if feasible (or iterate day by day if memory heavy) and compute features day-by-day. Dask can be employed if large.
- *Autonomy Aids:* We supply all formulas and threshold values in a config or at top of script, so it just applies them. It checks for existence of output (so we can skip recompute if needed). If data missing for a day, it either interpolates or marks feature as missing (which model can handle).
- *Output:* A consolidated **feature dataset** – perhaps a single NetCDF with dimensions (time, lat, lon, feature),

or a daily CSV of point features ready for join. We prefer a gridded format for map-making and an extracted table for model.

4. Label-Join Agent (Data assembly for ML):
- *Role:* Combine environmental features with shark presence/absence labels.
- *Tasks:* Read in the shark tracking data (CSV of points). For each point, sample the corresponding feature values: - If our features are in gridded form, we find the grid cell for that lat-lon (e.g., nearest neighbor or bilinear if needed) and extract feature values. - For absence points, do the same. Construct a pandas DataFrame of samples × features, with label column.
- *Agent Implementation:* Python with pandas & xarray. It will likely do something like: for each point in track, find nearest indices in lat, lon of the xarray Dataset of features at that date. This can be vectorized using xarray's `.sel` with method=nearest for all points at once (we might have to split by date for efficiency).
- *Autonomy Aids:* The agent needs the track data file path and the feature dataset path. Both will be known from config. It should handle if a point's date is outside feature date range (skip or warn).
- *Output:* `training_data.csv` containing all features and labels for each sample. Also, it could split into train/test folds here (adding a column fold or outputting separate files).

5. Modeler Agent (Training & Evaluation):
- *Role:* Train ML model(s) on the assembled data and evaluate performance.
- *Tasks:* Load `training_data.csv`. Split into folds (based on shark ID or precomputed assignment). Train XGBoost on training fold, predict on val fold, collect metrics. Do for all folds. Also train final model on all data (if deploying one). Additionally, run baseline HSI calculation on the same splits for comparison. And run ablation variants if scripted.
- *Agent Implementation:* Likely a Jupyter Notebook (to allow quick iteration and plotting) that we can also run as a script. Using scikit-learn or XGBoost's Python API. Might output metrics to console or a results JSON.
- *Autonomy Aids:* The process is straightforward given data; we hardcode CV logic or read fold definitions. The agent will output some plots (ROC curve, SHAP values) and save them to files, which we can display. It will also save the trained model (e.g. `model.xgb`).
- *Output:* `model.xgb` (or `.txt`), `performance_metrics.json`, and figure images (ROC.png, SHAP_summary.png, etc.) for the demo.

6. Cartographer Agent (Visualization & Demo Prep):
- *Role:* Create maps and interactive visualizations for communicating results.
- *Tasks:* Use the model to predict on a **grid** of environmental features to produce a habitat map. For a given day or average over period: - Feed the 0.1° gridded features into the model to get probability at each grid cell. - Produce a map (GeoTIFF or image) of these probabilities. - Alternatively, create a time series of maps (but due to time, we might pick a representative day or an average season map). Also, generate static maps of e.g. front locations vs shark positions, to illustrate correlation. Possibly prepare an interactive map (using Folium or kepler.gl) showing shark tracks over a layer of predicted hotspots. - *Agent Implementation:* Python with libraries like Matplotlib, Cartopy for static maps; Folium or deck.gl for interactive. We may generate a HTML or a small Dash app that can animate through days with a slider if ambitious (this might be out of scope, so likely static + perhaps an animated GIF). - *Autonomy Aids:* It uses outputs from earlier agents (model and feature grid). We provide coastline shapefiles or use Cartopy features for context. The agent has a config for which date to plot. - *Output:* Visual assets for the presentation: e.g., `shark_hotspots_Aug2023.png`, `shark_track_map.png`, `demo_map.html` (if interactive).

7. Explainer Agent (Report & Slides generation):
- *Role:* Compile results into presentation form.
- *Tasks:* Possibly auto-generate parts of the final README or slide deck. For example, use a Markdown template to populate with key metrics (like "AUC = 0.85") and embed the charts. If extremely automated, could even attempt to have an LLM agent write an executive summary (risky in hackathon; we likely do this part manually, but it's fun to mention). - *Agent Implementation:* This could be a simple script concatenating text and images into a PDF/markdown. Given time constraints, we might not automate slide creation fully; instead, we manual-curate slides using the outputs from above. However, documenting this as an agent shows we thought of pipeline end-to-end. - *Output:* The final **report (this document)** and a short slide deck. Possibly an auto-filled project page on Space Apps website with our results.

Each agent is designed to minimize human input – e.g., just run a script and it does the rest, using config files and built-in logic. We also design them to be run sequentially (or via a master script/Makefile) in case one output feeds the next.

Pipeline Orchestration: We will provide a **Makefile** with targets like `make data` (runs scout & fetcher), `make features` (runs feature agent), `make model` (runs modeler), `make maps` (runs cartographer), etc. This allows one-command execution of the whole pipeline (given prerequisites satisfied). We can also integrate with **GitHub Actions** or similar for continuous reproducibility, but likely not needed in hackathon timeframe.

For clarity, here's a simplified **flow diagram** of our pipeline:

```
Shark Tag CSV
    ↓ (Label-Join combines with Features)
Environmental Data → [Data-Scout] → [Subset-Fetcher] → Subset Files → [Ocean-
Features] → Feature Layers → [Label-Join] → Training Table → [Modeler] → Trained
Model & Metrics

Feature Layers (Grid) → [Cartographer] → Maps

Trained Model + Feature Grid → [Cartographer] → Predicted Hotspots
```

(Arrows indicate data flow; [Brackets] are agents. We see parallel path: features also go to Cartographer for map, not only through model.)

This architecture ensures each step is testable and replaceable. For instance, if we wanted to try a different region or species, we mainly swap the input data and rerun, with minimal code change, thanks to the configuration-driven design. This agentic pipeline not only runs the analysis end-to-end now, but is set up to be reused or expanded by others (e.g., test "sharks from space" in a different ocean or for a different predator).

Minimizing Human Elements: All key decisions (thresholds, region bounds, etc.) are parameterized. The human primarily needs to kick off the process (e.g. run `make all`). The agents have "help" in the form of detailed instructions (our code logic and comments serve as guidance to the agent – in effect, we codified

our domain knowledge into the pipeline). Because we've implemented error handling and sanity checks, the agents can adjust (like switching data sources or using default values if something's missing) rather than requiring human troubleshooting mid-run. This design addresses the user's goal of a fully automated solution with reliable, clean outputs.

What to do next: Finalize writing each agent script (some can be combined in notebooks during development, then separated); run the full pipeline on a small test (like 2 weeks of data) to verify the inter-agent connections; then run on full data; ensure the Makefile or orchestrator properly sequences dependencies (so features agent waits until data agent done, etc.). Prepare the environment file listing all needed Python packages for an easy run by judges.

Conceptual Tag Model (1-pager)

In addition to satellite detection, we propose a **next-generation shark tag** to directly monitor feeding in situ and link those events to satellite data. This conceptual design, inspired by state-of-the-art tags in research [59] [60], would equip tags with new sensors and communication abilities:

Tag Design & Sensors:
- **Form Factor:** A **pop-up archival tag (PAT)** combined with a real-time transmitter. Attaches to the shark's dorsal fin or is ingested (depending on sensor configuration). A hydrodynamic housing to minimize drag, with a corrodible release mechanism to detach after the mission (like existing PAT tags) [61]. Positive buoyancy and a small antenna ensure it surfaces to transmit when released [62].
- **Core Sensors:** - **Depth & Temperature:** Standard on many tags – logs dive profiles and ambient water temp. Depth can show hunting dives, temp stratification usage. - **Tri-axial Accelerometer & Gyroscope (IMU):** Captures fine-scale movement (tail beats, acceleration bursts) at, say, 5–10 Hz [63]. This can identify feeding-associated motions (rapid lunges, shaking). - **Light Sensor:** For geolocation (like classic PSAT tags use light for daily geo-estimates when no GPS). Also can detect when at surface (light increase). - **Stomach Temperature Pill:** A **small ingestible sensor** that sits in shark's stomach measuring internal temperature (and possibly pH) [64]. When a shark eats cold prey or ingests seawater during feeding, stomach temperature shows a distinct **drop**, followed by gradual rise as digestion produces heat [9]. This "thermal feeding signature" is a known indicator of a feeding event [59] [60]. Some studies have used acoustic stomach tags to detect feeding in white sharks and other species, proving feasibility [65] [64]. - *(Optional)* **Buccal Impedance or Pressure Sensor:** Mounted near jaws to detect jaw opening/closing or bites. Experimental, but could directly sense a bite action via changes in electrical impedance or pressure. - *(Optional)* **Hydrophone (Microphone):** To listen for sounds associated with predation (e.g., seal vocalizations or crunching on prey). This is power-intensive and data-heavy, so maybe not continuous, but could trigger recording if acceleration indicates a burst event. - *(Optional)* **Environmental DNA sampler:** This is quite futuristic, but a small chamber could periodically trap water and later be analyzed for DNA of prey species in area. (Probably not real-time though.)

- **Onboard Processing (Smart Tag):** A microprocessor on the tag runs an algorithm to detect probable **feeding events** in real-time from sensor data:
- It monitors for the pattern: a sudden drop in stomach temp (>= X°C) combined with a spike in acceleration (lunge) and perhaps depth change (quick up/down). When such a pattern occurs, the tag logs a "feeding event timestamp."

- This logic ensures it doesn't need to transmit continuous raw data (which is impossible underwater). It can store detailed data in memory for later, but for immediate needs it extracts key events.

- **Memory & Sampling:** High-capacity flash memory to log multi-sensor data for months (accelerometer 10 Hz, etc.). But to conserve, maybe duty-cycle high-freq sensors (e.g., record 1 min out of every 5, or triggered by certain conditions like nearing surface or known feeding times – dawn/dusk).

- Stomach pill can transmit to the main tag via acoustic link (like some existing pH/temperature pills do [66]).

- Expected battery life: perhaps 6–12 months if mostly logging and occasional transmit. Use lithium batteries and perhaps a small solar for surface recharging (if fin-mounted and shark spends time near surface by day).

- **Telemetry & Real-time Data:**

- Whenever the shark's fin/tag breaks the surface, the tag attempts to send data via **Argos satellite uplink** or newer **Iridium** network. Traditional Argos tags transmit a few bytes per uplink – enough for location and a summary. We'd program it to send:
    - Latest compressed track (recent positions via light/GPS if available).
    - **Feeding event alerts:** e.g., "Shark X had 3 feeding events in last 24h at times t1, t2, t3."
    - Basic vitals: current temp, battery, etc.
- If the tag is designed to detach at end of mission, it then uploads full dataset via Argos (which can take weeks with Argos due to limited bandwidth) or is physically recovered (as many PAT tags are – but recovery is rare unless you chase it).

- We can also imagine a near-real-time acoustic relay: e.g., if shark comes near a buoy or receiver, the tag could dump data via acoustic modem. This is used in some observatory systems but requires infrastructure.

- **Aligning with Satellite Data:** The **timestamps and locations** of feeding events from the tag are crucial. These can be compared to satellite-observed features:

- E.g., Tag says shark ate something on Aug 5, 2023, 10:00 UTC at ~40.1°N, 66.5°W. Our satellite data that day shows an eddy edge and SST front at that location. This validates our model's prediction and can help refine it.

- The conceptual system could even use satellite info in near-real time: if the tag surfaces and transmits position, we can check satellite data for that region and predict "high feeding probability" – if low but tag indicates feeding, interesting contradiction to study, etc. Over long term, these combined datasets (tag + satellites) allow training better models and also forecasting where sharks might feed next (by seeing where features are moving).

- **Practical Considerations:**

- **Attachment:** Fin-mount is typical for white sharks (a strong clamp or bolt through dorsal fin). Ingested stomach pill would pass naturally after some time (or stay until regurgitated, as in studies

where sharks did eventually regurgitate stomach loggers after days/weeks [67], which is okay as long as data was transmitted).

- **Ethics & Impact:** The tag should be designed to minimize harm: smooth edges, proper size (maybe ~15 cm device plus antenna). OCEARCH tags already involve capturing the shark – our extra sensors (stomach pill) could be administered during that handling. Endangered status (whites are vulnerable) means we want maximal data yield per animal – our multi-sensor approach does that, albeit at higher cost per tag.

- **Data Integration:**

- On the user side (scientists), a **dashboard** could receive real-time pings: mapping sharks with colored icons when a feeding event occurs (like "shark dinner bell"). They could overlay NASA data maps on that dashboard (similar to what we plan to demo) to see context.
- Over months, one could accumulate which oceanographic contexts correspond to feeding vs just roaming. This directly answers the Space Apps challenge by marrying tag and satellite perspectives.

Innovation & Benefits: This tag concept pushes beyond location-only tags to get at "what they are eating and when" (the challenge's phrasing [2]). By measuring **internal state (stomach temp)** and **fine motion**, we infer diet events – essentially giving the shark a voice to say "I ate now." Transmitting in real-time addresses the challenge of real-time data to users [68]. Such tags could drastically improve validation of satellite-based predictions (we can be *sure* a feeding happened at X time, to confirm if our model's "likely foraging habitat" was indeed used).

Finally, by knowing feeding success, we can improve conservation: e.g., if sharks aren't finding prey (few feeding events) in a usually productive area, that might signal a fish population decline or ocean change. Managers could respond (closing fisheries, etc.). Our conceptual tag thus helps close the loop between observation and actionable insight.

What to do next: Continue R&D on sensor fusion algorithms for feeding detection (perhaps simulate with captive sharks or using existing data [69]). Engage tag manufacturers (Wildlife Computers, etc.) to incorporate stomach pill comms. And test prototypes on smaller sharks or captive individuals before full deployment. This concept can be a part of our project story to show creativity and ambition.

Reproducible Repo Scaffold

We set up a clear repository structure to organize code, data, and documentation. This makes our project easy to navigate for judging and for anyone who wants to replicate or extend it. Below is our proposed directory tree and key files:

```
sharks-from-space/
├── README.md                    # High-level overview, setup & usage
instructions
├── environment.yml              # Conda environment file listing all
dependencies (Python 3.x, xarray, dask, xgboost, shap, folium, etc.)
├── .env.example                 # Template for sensitive configs (e.g.,
```

```
Earthdata username/password or token)
├── Makefile                        # Makefile to run pipeline stages (data,
features, model, viz)
├── config/
│   ├── params.yaml                 # Configurable parameters (ROI coords, date
range, thresholds for features, etc.)
│   └── earthdata_config.json       # e.g., Earthdata login credentials or token
(not tracked, user supplies or uses .env)
├── data/
│   ├── raw/                        # Raw downloaded subsets (git-ignored, since
potentially large)
│   │   ├── SST/ ...                # e.g., SST netCDF files
│   │   ├── SSH/ ...
│   │   └── ... (other datasets)
│   ├── interim/                    # Intermediate processed data
│   │   ├── features.nc             # Combined feature grids (NetCDF/Zarr
format)
│   │   └── training_data.csv       # Tabular data for modeling (features +
labels)
│   └── external/                   # External static data (bathymetry,
coastline for plots, etc.)
│       └── gebco_1min.nc (for depth, if used)
├── notebooks/
│   ├── 1_data_retrieval.ipynb      # Notebook (or script) demonstrating data
search & download
│   ├── 2_feature_engineering.ipynb # Notebook for computing fronts, eddies,
etc.
│   ├── 3_model_training.ipynb      # Notebook for training ML and evaluating
│   └── 4_visualization.ipynb       # Notebook for creating maps and plots
├── src/
│   ├── data_scout.py               # Script for CMR search (could also be
inside 1_data_retrieval.ipynb)
│   ├── fetch_data.py               # Uses Harmony or requests to download
subsets (maybe combined with data_scout)
│   ├── compute_features.py         # Script to process data into features (if
not using notebook for it)
│   ├── label_join.py               # Script to merge shark track with features
and output training table
│   ├── train_model.py              # Script to train model (could use CLI args
for hyperparams)
│   ├── evaluate_model.py           # Script to do CV and output metrics
│   └── make_maps.py                # Script to generate final maps and maybe
interactive output
├── docs/
│   ├── challenge_brief.md          # This detailed brief (for reference)
│   ├── landscape_refs.md           # Annotated bibliography of sources (10-15
items summarized)
```

```
│   ├── usage_guide.md              # Instructions on how to run the pipeline,
with examples
│   └── images/                     # Folder for images used in docs or README
│       ├── pipeline_diagram.png
│       ├── feature_example.png
│       └── ...
├── app/
│   ├── dashboard.py                # (Optional) Flask or Dash app for
interactive map of results
│   └── assets/                     # any web assets (if needed for app)
└── tests/
    └── test_feature_calc.py        # (Optional) small tests to verify
computations like OW or gradient thresholds
```

A few notes on this structure: - We separate raw data, processed data, and external data to avoid confusion. Raw data can be re-downloaded via the pipeline, so we don't version it. - Notebooks vs scripts: During development we use Jupyter notebooks (in `notebooks/`) for clarity and plotting. For automation, equivalent .py scripts in `src/` can run the same steps without manual intervention. We'll ensure any final critical step is also scripted (for headless run). - Config parameters in a central file (`config/params.yaml`) make it easy to change species or region later. - The Makefile might have recipes like: - `make data` -> run data_scout.py & fetch_data.py - `make features` -> run compute_features.py - `make train` -> run train_model.py - `make eval` -> run evaluate_model.py - `make maps` -> run make_maps.py - `make all` -> everything end-to-end. - `.env.example` will show environment variables we use, e.g., `EARTHDATA_USERNAME` and `EARTHDATA_PASSWORD` or a token, so the user can create their own `.env`. Our code will load those (using `python-dotenv` or os.environ).

- We will provide a **sample STAC/CMR search JSON** output (maybe in `data/interim`) so that even without hitting CMR, one can see what granules we intended. This was requested to demonstrate our search results for reproducibility.

- The repository will have a comprehensive README instructing:

- how to setup (install conda env from environment.yml, activate, put Earthdata creds in .env),
- how to run the pipeline (via make or running notebooks sequentially),

- where results will appear.

- We will version control all code and small sample outputs, but not large data. Possibly we include a small example dataset (like one day's data for quick test, under `data/external/` or a zip).

- **Reproducibility:** Anyone with Earthdata access should be able to clone the repo and run `make all` to regenerate everything. We lock package versions in environment.yml to avoid future incompatibilities.

- **Collaboration:** If this were a longer project, we'd integrate continuous integration (CI) to run tests (if tests are written for some functions). For hackathon, we just ensure things run on provided environment.

Given this scaffold, even after the hackathon, others can use it as a starting point (e.g., to track other marine animals with satellites).

What to do next: Initialize this repo structure on GitHub. Add the content we have (challenge brief, references, etc., to docs). Write the README skeleton. Fill in the environment file with exact versions (we'll gather those once code is done, e.g., xarray 2023.x, xgboost 1.6, shap 0.41, etc.). Prepare the .env.example with dummy values. Ensure .gitignore excludes data/raw.

We'll also prepare a short *"one-click deploy"* if possible: perhaps using Binder or similar to launch the notebooks with our environment (if data can be fetched on the fly). But given data volume, we might not fully one-click run everything, instead one-click to see the notebooks with outputs pre-filled (we can save notebook outputs or provide small sample runs).

Finally, double-check that our attachments (tables, figures) are in place as needed by guidelines (e.g. any CSV needed for judges to view tables, we might attach or embed simple ones like performance metrics).

Demo Plan

In the final demo (90 seconds pitch + supplementary material), we will present a compelling narrative focusing on problem, approach, and results. Here's our game plan:

Story Arc (90-second narrative):

1. **Hook (10s):** *"What if we could track sharks from space?"* Open with a striking visual: a map of the ocean with satellite data and a shark icon, to grab attention. Emphasize the challenge: Sharks are hard to follow, but NASA satellites see ocean features that might reveal where they feed. Mention the core ask: identify shark foraging habitats from space and design a better tag.

2. **Problem & Data (20s):** Show Slide 1 – split-screen: left side "Ocean from Space" (image of chlorophyll map with eddies [5]), right side "Shark Tracks" (image of a tagged shark track from OCEARCH). Explain: *"Top predators like great whites roam vast oceans. We have thousands of satellite images of the ocean's temperature, color, etc., and some sharks with tags. We need to connect the dots."* Highlight NASA data usage: mention we used **6 NASA datasets** (briefly list: PACE ocean color, satellite SST, altimetry for eddies, etc.) – judges love hearing multiple data sources.

3. **Solution Approach (30s):** Show Slide 2 – a diagram of our pipeline (agents flowchart) or an infographic:

    - Summarize approach: *"We created an AI pipeline that takes NASA imagery and finds hotspots – like thermal fronts and eddies – then checks if sharks go there."*
    - Mention key steps: detecting **fronts** (maybe an image with a front highlighted), detecting **eddies** (image of an eddy from SSH [6]), building a model.

- Emphasize the novelty: *"We even propose a new smart shark tag with a stomach sensor to verify if a shark actually ate when our model predicts it should – a fusion of space data and animal-borne data."* (Briefly show a sketch of the tag or list sensors).
- Judging criteria check: highlight creativity (AI + new tag), data use (NASA data heavy), science (built on real research). Possibly overlay small icons or checkmarks on this slide labeling criteria: e.g., a data icon with "NASA data ", a lightbulb icon "Innovation ".

4. **Results (20s):** Show Slide 3 – a heatmap of predicted shark foraging probability in our ROI, with actual shark locations overlaid as validation. For example, a map with colored areas (red = high habitat suitability) and black dots for actual shark pings – we hope many dots fall in red/orange zones (if our model did well).

    - Narrate: *"This map shows in red where our model predicts high shark foraging activity on July 2023. The black markers are actual tagged shark positions – see how they align with fronts and eddies in the Gulf Stream region, confirming our predictions."*
    - Mention performance: *"We achieved an AUC of ~0.85, meaning we can correctly identify shark hotspots ~85% of the time – a big improvement over random or static assumptions."* (If time, mention baseline vs model quickly: "Better than a static range map, which was ~0.6 AUC.")
    - If SHAP or feature importance was computed, verbally highlight: *"The model learned that chlorophyll and temperature fronts are the top indicators – matching marine biology theory* [5] *. This adds confidence in our model's validity."*

5. **Impact & Future (10s):** Conclude with how this helps and what's next:

    - *"Our project can help direct marine biologists and conservationists to feeding hotspots – kind of like finding wildlife refuges in the ocean – using open NASA data."*
    - *"The new tag concept could provide real-time alerts when a shark feeds, allowing immediate correlation with satellite data. This brings us closer to truly understanding ocean predators and protecting them."*
    - End with a visionary note: *"In short, we're giving sharks a 'voice' via satellites and sensors – a win for science and shark conservation."* and a final thank you.

Demo Visuals & Live Map:

- We plan to have a **live interactive map** (e.g., a Folium web map) as part of the demo for judges to explore:
- It could show the coastline and our predicted habitat index as a color overlay. We can add a time slider to animate through days (if using deck.gl or Folium plugin).
- Also plot shark track points (with popups for date perhaps).
- If internet or environment doesn't allow live, we'll have a local HTML or a short screen recording of the interaction.

- This fulfills the "one live map" requirement in the plan and impresses visually.

- **Screenshots fallback:** We will prepare static images of all critical visuals in case live demo fails. For instance, if Folium map doesn't display due to environment, we have a screenshot of it. Or if we intended to show a quick code output or metric, have a backup slide.

- **Judging Rubric Checklist Slide:** Possibly a final slide (Slide 4, only if time allows) or simply in our speaking, we will explicitly tick off criteria:

- Influence: *"Our project can influence conservation policy (Galactic/Local Impact)."*
- Creativity: *"We combined novel data and tag sensors (Innovation)."*
- Validity: *"Based on peer-reviewed methods and validated with real data (Best Use of Science)."*
- Data: *"Utilized 5 NASA datasets in synergy (Best Use of Data)."*
- Storytelling/Presentation: *"We told the story of Shark from Space connecting oceanography to ecology (Storytelling)."*

We can show a quick list with green check marks as a visual affirmation.

Rehearsal and Timing: We will practice to ensure we hit ~90 seconds. It's tight, so likely: - 10s intro, - 20s problem/data, - 30s approach, - 20s results, - 10s impact.

We will streamline talking points accordingly.

Each team member (if multiple presenters) can take a part to show multi-disciplinary nature (one as "NASA data expert", one as "marine biologist", etc., to demonstrate team breadth – but if solo, just cover all succinctly).

We will also be ready for Q&A if judges ask (with extra slides in backup if needed, like detailed methodology, or the tag diagram in detail).

Slide Design: Keep text minimal, use visuals from our actual analysis: - likely use dark blue ocean-themed background or high-contrast to make colored data pop (red/yellow hotspots). - Use the shark icon or NASA logo to decorate but focus on data viz.

In summary, the demo will walk judges through *why* the problem matters, *how* we solved it using NASA data in a clever way, and *what* the outcome and future implications are – all within 1.5 minutes, leaving them impressed with the thoroughness and potential.

What to do next: Build the slide deck in parallel while finalizing results (so we can drop in final maps/metrics last minute). Create the interactive map (or at least the data for it) and test it in the presentation environment. Write speaker notes aligning with timing. Have colleagues watch a practice run to ensure clarity. Finalize the checklist of criteria and ensure every judge concern (data usage, innovation, feasibility, etc.) is addressed either in presentation or backup answers.

This comprehensive plan and research-backed solution put us on a strong footing to win the "Sharks from Space" challenge. We have compelling science, excellent use of NASA's open data, a clear engineering path, and even a hardware concept bridging the gap – all delivered by a robust, largely autonomous pipeline.

---

1 2 68 English-NASA Space Apps Challenge 2025 - Summaries | PDF | Nasa | Agriculture
https://www.scribd.com/document/906422966/English-NASA-Space-Apps-Challenge-2025-Summaries

3 4 7 8 Judging & Awards Guideline | PDF
https://www.scribd.com/document/876777314/Judging-Awards-Guideline

5 10 25 42 50 51 Data from: Basking sharks and oceanographic fronts: quantifying associations in the north-east Atlantic - ePrints Soton
https://eprints.soton.ac.uk/449017/

6 14 Mesoscale eddies influence the movements of mature female white sharks in the Gulf Stream and Sargasso Sea - PubMed
https://pubmed.ncbi.nlm.nih.gov/29743492/

9 59 60 61 62 63 65 67 In the belly of the beast: resolving stomach tag data to link temperature, acceleration and feeding in white sharks (Carcharodon carcharias) | Animal Biotelemetry | Full Text
https://animalbiotelemetry.biomedcentral.com/articles/10.1186/s40317-015-0071-6

11 The relative importance of biological and environmental factors on ...
https://www.researchgate.net/publication/365324738_The_relative_importance_of_biological_and_environmental_factors_on_the_trophodynamics_of_a_pelagic_marine_predator_the_blue_shark_

12 Vertical patterns of chlorophyll a in the euphotic layer are related to ...
https://www.frontiersin.org/journals/marine-science/articles/10.3389/fmars.2022.948665/full

13 A novel method to estimate the 3D chlorophyll a distribution in the ...
https://www.nature.com/articles/s41598-024-76748-5

15 Brief Overview of the Great White Shark
http://www.elasmo-research.org/education/white_shark/overview.htm

16 Great white shark - Wikipedia
https://en.wikipedia.org/wiki/Great_white_shark

17 [PDF] Regional Essential Fish Habitat Geospatial Assessment ... - GovInfo
https://www.govinfo.gov/content/pkg/GOVPUB-I-06e4621a0c5d2fad6ef2c87cf1e69f35/pdf/GOVPUB-I-06e4621a0c5d2fad6ef2c87cf1e69f35.pdf

18 Three-Dimensional Movements and Habitat Selection of Young ...
https://www.frontiersin.org/journals/marine-science/articles/10.3389/fmars.2021.643831/full

19 Variability of chlorophyll-a concentration in the Gulf of Guinea and its ...
https://pmc.ncbi.nlm.nih.gov/articles/PMC5339419/

20 Shark conservation hindered by lack of habitat protection
https://www.sciencedirect.com/science/article/pii/S2351989419303786

21 The relative importance of biological and environmental factors on ...
https://www.sciencedirect.com/science/article/pii/S0141113622002537

22 Seasonal changes in the habitat suitability of immature white sharks ...
https://academic.oup.com/icesjms/article/82/5/fsaf062/8140451

23 24 Frontiers | Global-Scale Environmental Niche and Habitat of Blue Shark (Prionace glauca) by Size and Sex: A Pivotal Step to Improving Stock Management
https://www.frontiersin.org/journals/marine-science/articles/10.3389/fmars.2022.828412/full

26 (PDF) Habitat assessment of Whale Sharks (Rhincodon typus) in Saleh bay, Indonesia: Linking chlorophyll-a and sea surface temperature using AQUA MODIS data
https://www.researchgate.net/publication/391316093_Habitat_assessment_of_Whale_Sharks_Rhincodon_typus_in_Saleh_bay_Indonesia_Linking_chlorophyll-a_and_sea_surface_temperature_using_AQUA_MODIS_data

27 28 29 41 NASA PACE - Data Access
https://pace.oceansciences.org/access_pace_data.htm

30 The Multi-scale Ultra-high Resolution (MUR) | PO.DAAC / JPL / NASA
https://podaac.jpl.nasa.gov/MEaSUREs-MUR

31 NASA PODAAC Multiscale Ultrahigh Resolution SST
https://eastcoast.coastwatch.noaa.gov/cw_podaac-mur_sst.php

32 43 44 45 46 47 58 EarthData Cloud Cookbook - How do I subset data granules?
https://nasa-openscapes.github.io/earthdata-cloud-cookbook/how-tos/subset.html

33 MEaSUREs Gridded Sea Surface Height Anomalies Version 2205
https://podaac.jpl.nasa.gov/dataset/SEA_SURFACE_HEIGHT_ALT_GRIDS_L4_2SATS_5DAY_6THDEG_V_JPL2205

34 MEaSUREs Gridded Sea Surface Height Anomalies Version 2205 ...
https://podaac.jpl.nasa.gov/node/1745

35 Ocean surface currents from satellite data - Dohan - 2017
https://agupubs.onlinelibrary.wiley.com/doi/10.1002/2017JC012961

36 OSCAR - Ocean Currents - LuckGrib
https://luckgrib.com/models/oscar/

37 RSS SMAP Level 3 Sea Surface Salinity Standard Mapped Image 8 ...
https://podaac.jpl.nasa.gov/dataset/SMAP_RSS_L3_SSS_SMI_8DAY-RUNNINGMEAN_V6

38 Sea Surface Salinity - Near Real Time - SMAP - NOAA CoastWatch
https://coastwatch.noaa.gov/cwn/products/sea-surface-salinity-near-real-time-smap.html

39 GPM IMERG Late Precipitation L3 1 day 0.1 ... - Dataset - Catalog
https://catalog.data.gov/dataset/gpm-imerg-late-precipitation-l3-1-day-0-1-degree-x-0-1-degree-v07-gpm-3imergdl-at-ges-disc-72ac7

40 Evaluating Precipitation Events Using GPM IMERG 30-Minute Near ...
https://journals.ametsoc.org/view/journals/hydr/25/7/JHM-D-23-0141.1.xml

48 IMERG Final Run | NASA Global Precipitation Measurement Mission
https://gpm.nasa.gov/taxonomy/term/1417

49 [PDF] Evaluation of Front Detection Methods for Satellite-Derived SST ...
https://digitalcommons.uri.edu/cgi/viewcontent.cgi?article=1274&context=gsofacpubs

52 53 54 56 npg.copernicus.org
https://npg.copernicus.org/articles/23/159/2016/npg-23-159-2016.pdf

55 An Eulerian Scheme for Identifying Fronts and Vortices in Quasi ...
https://journals.ametsoc.org/view/journals/phoc/51/10/JPO-D-21-0037.1.xml

57 Regional variations in the influence of mesoscale eddies on near ...
https://agupubs.onlinelibrary.wiley.com/doi/10.1002/2014JC010111

64 69 Feeding event shown in white shark stomach tag data. The stomach...
https://www.researchgate.net/figure/Feeding-event-shown-in-white-shark-stomach-tag-data-The-stomach-temperature-of-a-wild_fig5_283616743

66 A new acoustic pH transmitter for studying the feeding habits of free ...
https://www.alr-journal.org/articles/alr/pdf/2007/04/alr006-07.pdf