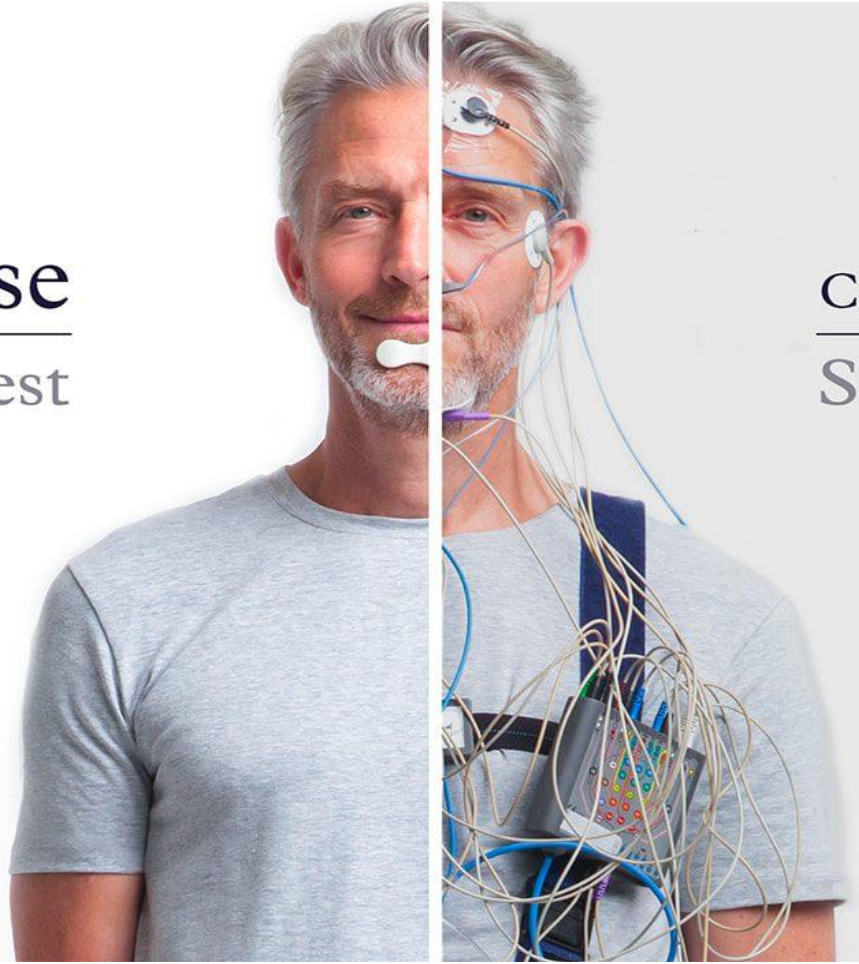
AXA interview: Use case presentation

Faouzi Braza, 23/11/2023

Sunrise

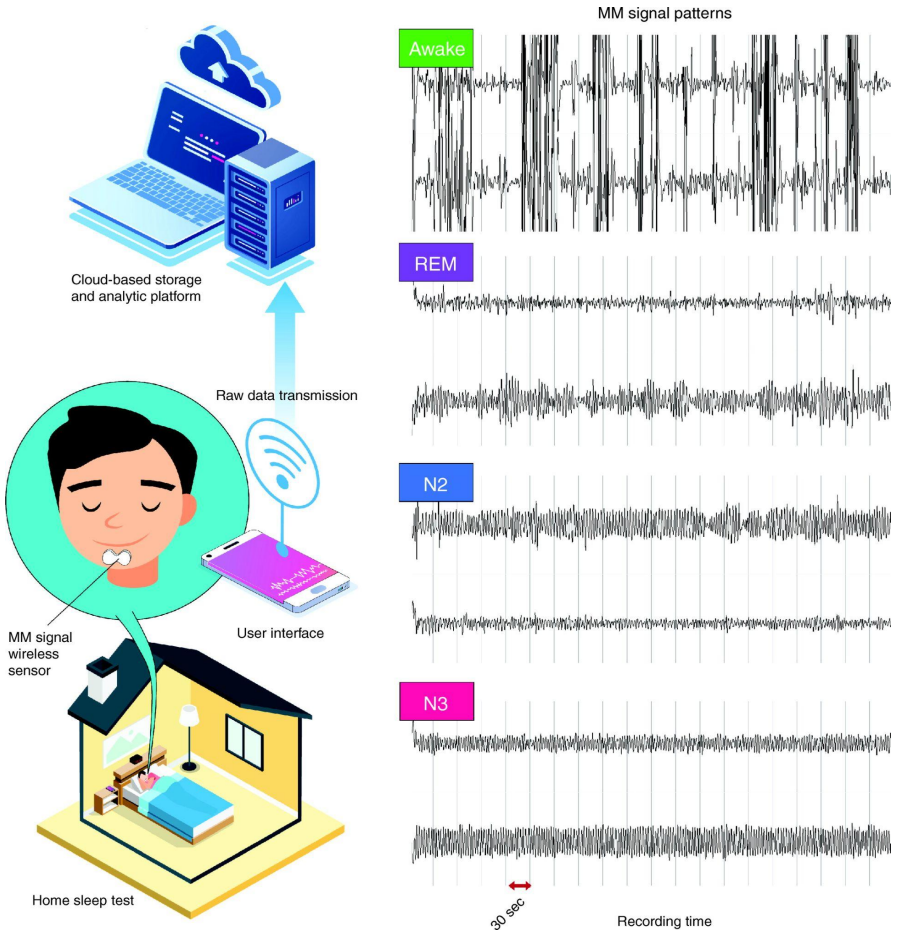
Sleep Test



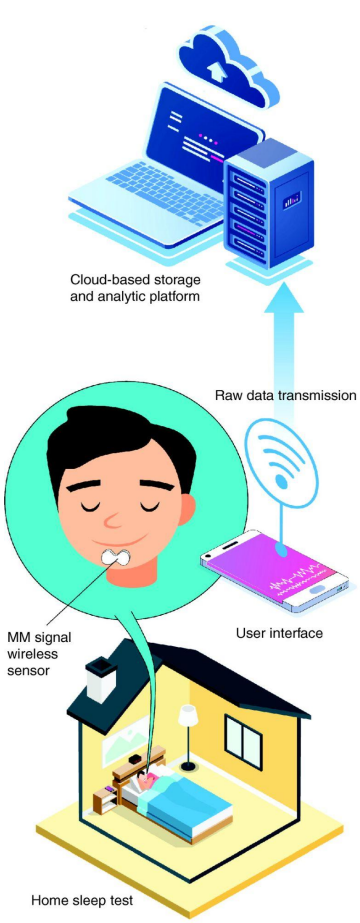
Conventional

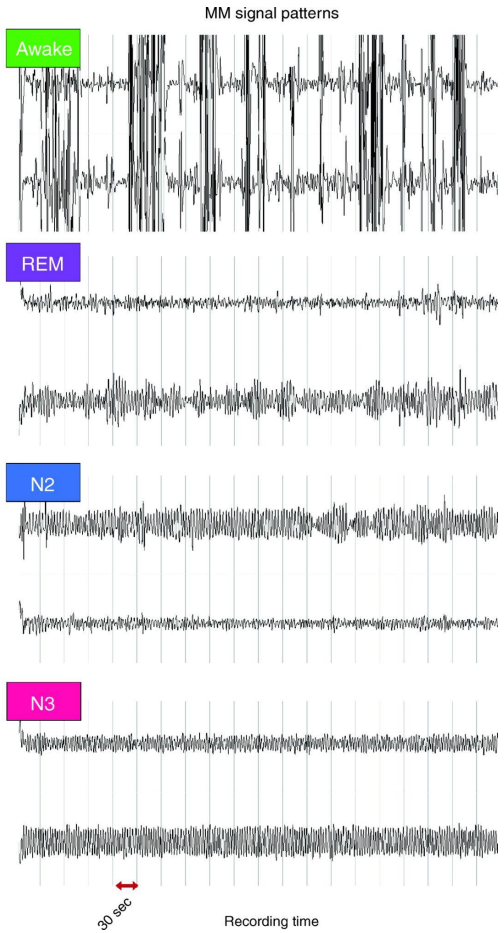
Sleep Test

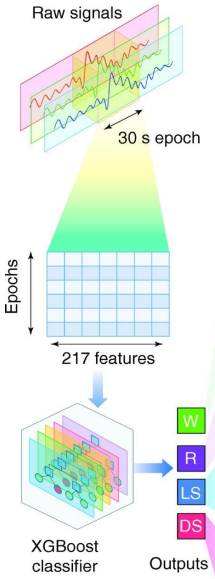
Predictions of sleep staging with machine learning

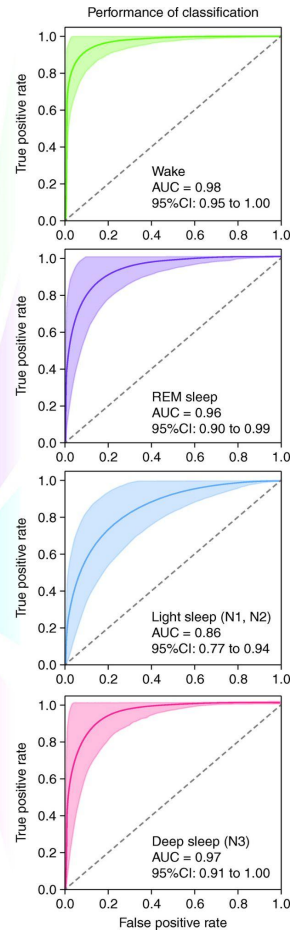


Predictions of sleep staging with machine learning

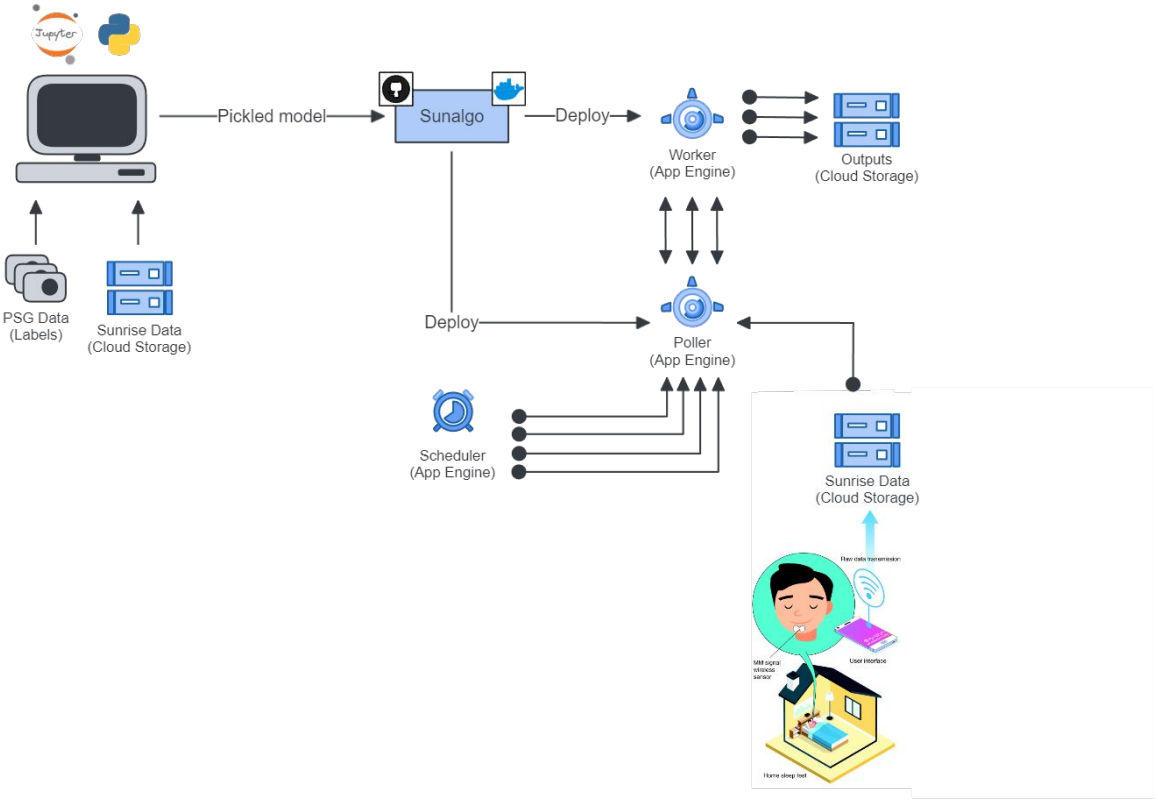




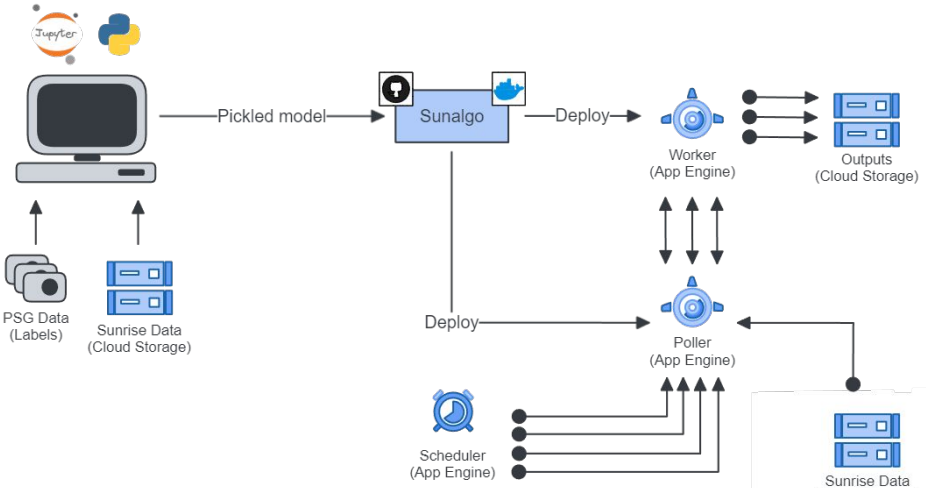


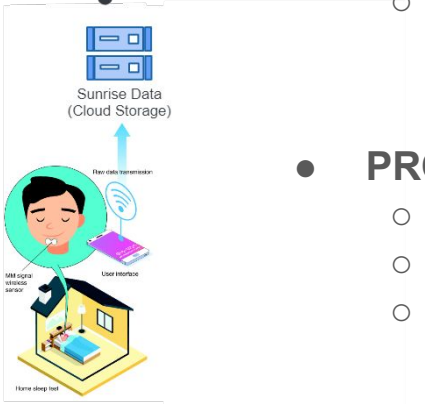


The old architecture



The old architecture





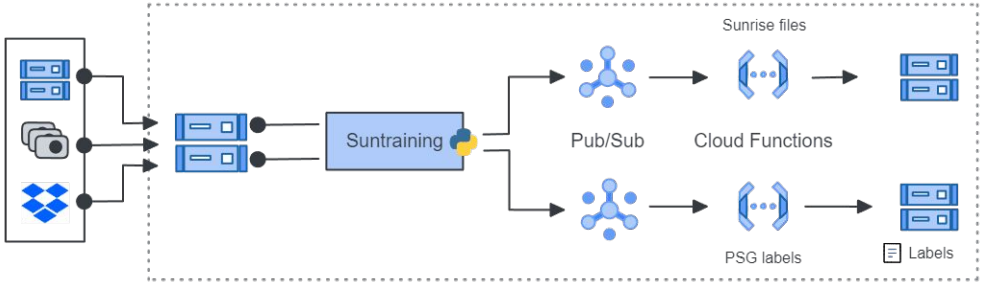
- **Research / DEV environment**
  - Scalability
  - Reproducibility
  - Flexibility
  - Alerting
  - Boyd's Law of Iteration

- **PROD environment**
  - Scalability
  - BI
  - Alerting

Boyd's Law of Iteration

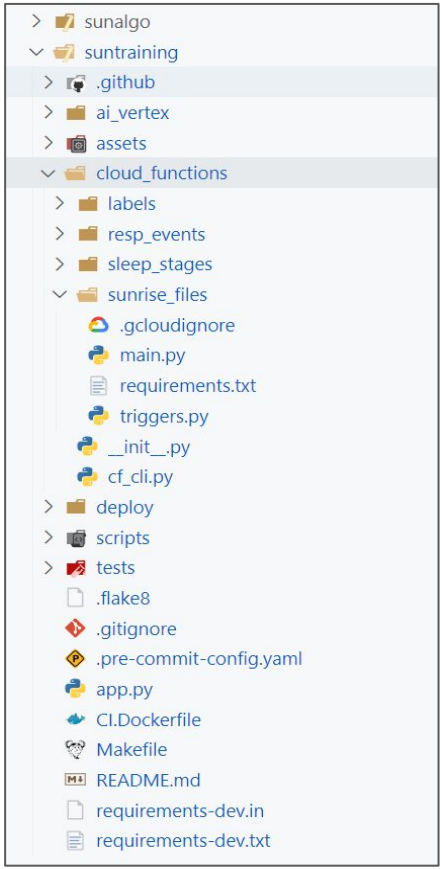
“Boyd decided that the primary determinant to winning dogfights was not observing, orienting, planning, or acting better. The primary determinant to winning dogfights was observing, orienting, planning, and acting *faster*. In other words, how quickly one could iterate. *Speed of iteration*, Boyd suggested, *beats quality of iteration*.”

The R&D platform



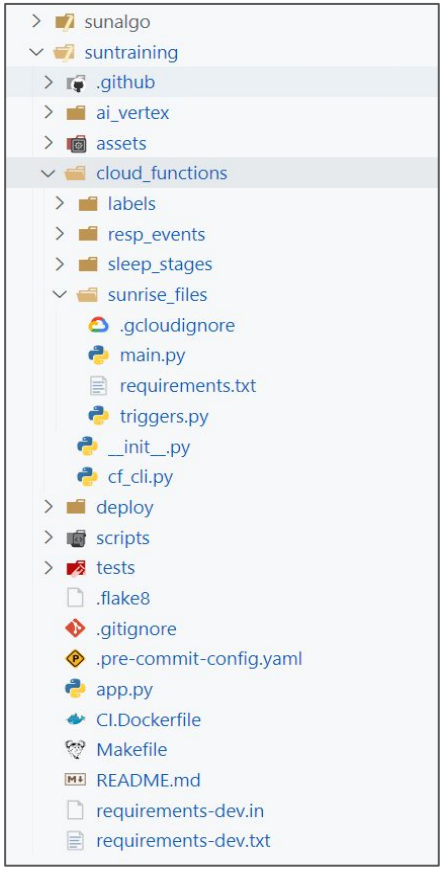
R&D Environment (DEV)

Anatomy of a cloud function



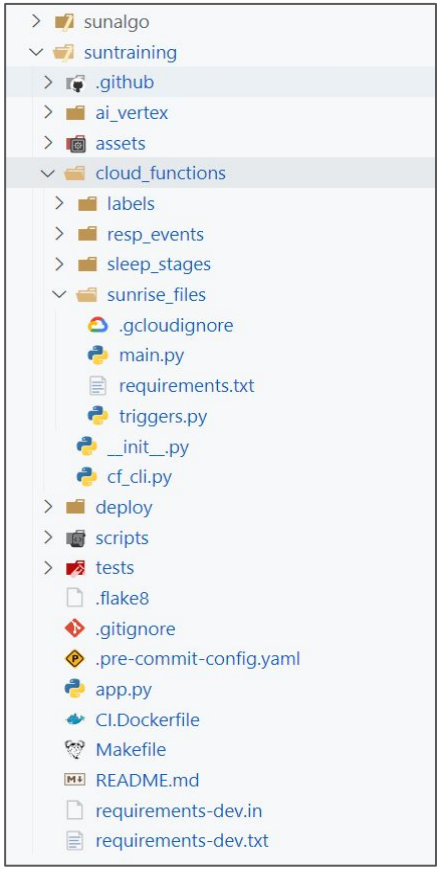

```python
def send_nights_to_pubsub(
    input_dir: str,
    output_dir: str,
    topic: str,
    bucket_name: str = training_bucket,
):
    all_files = get_all_files(bucket_name, input_dir)
    all_nights = get_all_nights(all_files)
    publisher = pubsub_v1.PublisherClient()
    for night in all_nights:
        missing = 0
        nights_files = [
            filename.replace(night + "/", "")
            for filename in all_files
            if filename.startswith(night) and is_needed(filename)
        ]
        if len(nights_files) != 5:
            missing += 1
            logger.warning(
                f"Number of selected file is incorrect : should be 5 : check night {night}"
            )
            logger.warning(f"{nights_files}")
            continue
        # collect the necessary files
        anonymised_file: str = os.path.join(
            input_dir, night, select_file(nights_files, ["anonymised.csv"])
        )
        hypno_file: str = os.path.join(
            input_dir,
            night,
            select_file(nights_files, ["Hypnogramme.txt", "Hypnogram.txt"]),
        )
```

Anatomy of a cloud function




```python
# generate expected payload
        sunrise_files_data = payloads.SunriseFilePayload(
            bucket=bucket_name,
            output_dir=output_dir,
            night=night,
            anomy=anonymised_file,
            hypno=hypno_file,
            accel=accel_file,
            gyros=gyros_file,
            patient=patient_file,
        )
        data = str(sunrise_files_data.model_dump()).encode("utf-8")

        try:
            topic_path = publisher.topic_path(project="sensav2", topic=topic)
            future = publisher.publish(topic_path, data)
            future.result()
        except Exception as err_msg:
            logger.error(f"Failed to send to topic {topic} : {err_msg}")
            return "Status: failed"
        logger.info(f"Triggers sent for night {night}")

    logger.info(f"{len(all_nights) - missing}/{len(all_nights)} nights sent")
    return "Status: ok"
```

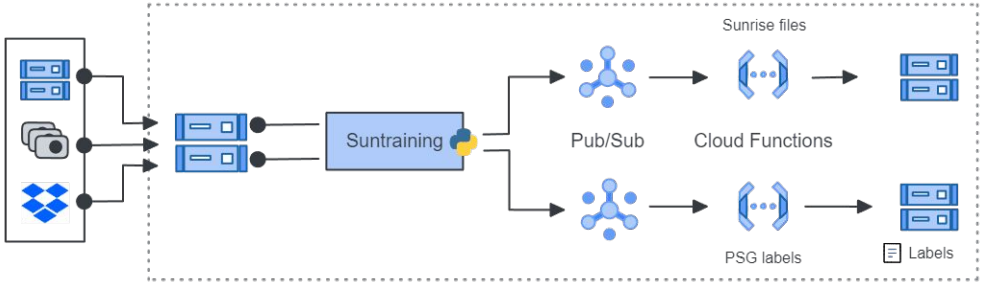
Anatomy of a cloud function




```
#! /bin/bash

SOURCE_DIR="cloud_functions/labels"

gcloud config set project sensav2;

gcloud functions deploy write-psg-labels-domino \
--region=europe-west1 \
--runtime=python310 \
--memory=2048 \
--source=${SOURCE_DIR} \
--entry-point=write_labels_for_domino \
--trigger-topic=write-labels-domino \
--timeout=540
```

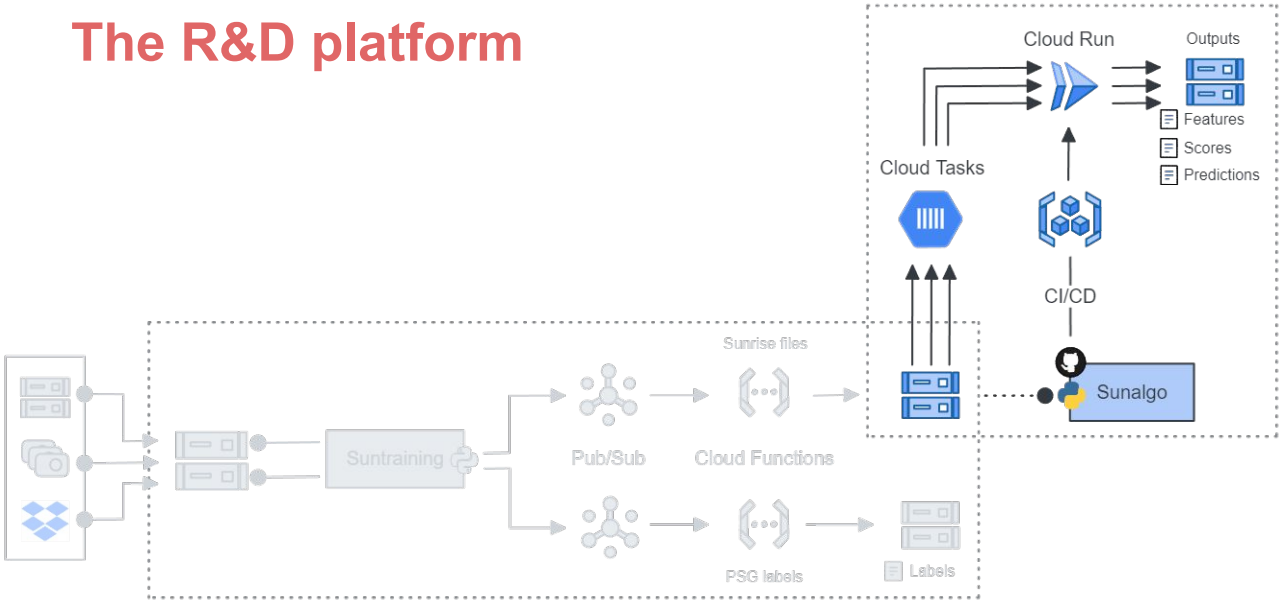
```
import subprocess
import typer

# cut for brievety

@sleep_stages_app.command(
    "deploy",
    help="Deploy cloud function to write sleep staging features for train and valid dataset",
)
def sleep_stages_deploy():
    subprocess.run(["bash", "deploy/cf_sleep_stages_features.sh"])


@sleep_stages_app.command(
    "trigger", help="Trigger the cloud function build-sleep-stages-datasets"
)
def sleep_stages_trigger():
    from sleep_stages import triggers

    triggers.send_nights_to_pubsub()
```

The R&D platform



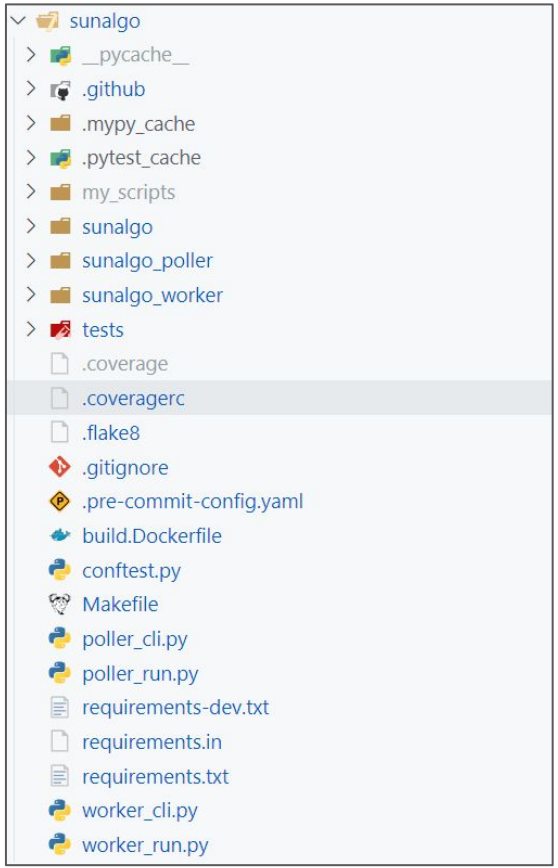
R&D Environment (DEV)

The R&D platform



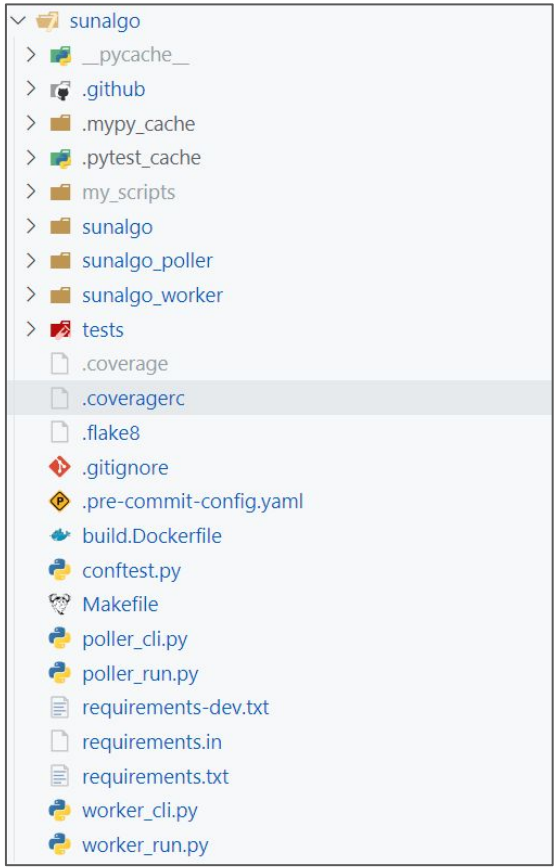
R&D Environment (DEV)

Sending tasks



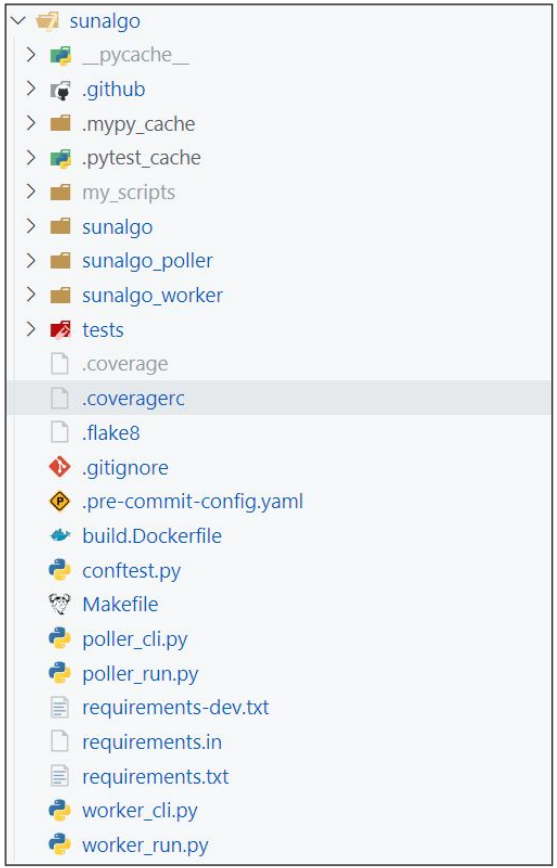

```python
@click.option("--file_prefix", required=False)
@click.option(
    "--append_file_name_to_output_dir",
    default=True,
    help="Add the filename to output_dir name",
)
def send_tasks_to_cloud_queue(
    worker_url,
    input_bucket,
    input_dir,
    output_bucket,
    output_dir,
    max_files,
    file_prefix,
    append_file_name_to_output_dir,
):
    # Instantiate client
    client = tasks_v2.CloudTasksClient()
    parent = client.queue_path(
        project="sensav2", location="europe-west1", queue="poller-queue"
    )
    # task dict
    task = {
        "http_request": {
            "http_method": tasks_v2.HttpMethod.POST,
            "url": f"{worker_url}/score",
        }
    }
    # change the timeout limit for cloud task, maximum is 30 minutes
    # https://github.com/googleapis/python-tasks/issues/93#issuecomment-827752337
    timeout = duration_pb2.Duration()
    timeout.FromSeconds(60 * 30)
    task["dispatch_deadline"] = timeout
    # base payload
    payload_dict: dict[str, str] = {
        "input_bucket": input_bucket,
        "output_bucket": output_bucket,
    }
```

Sending tasks



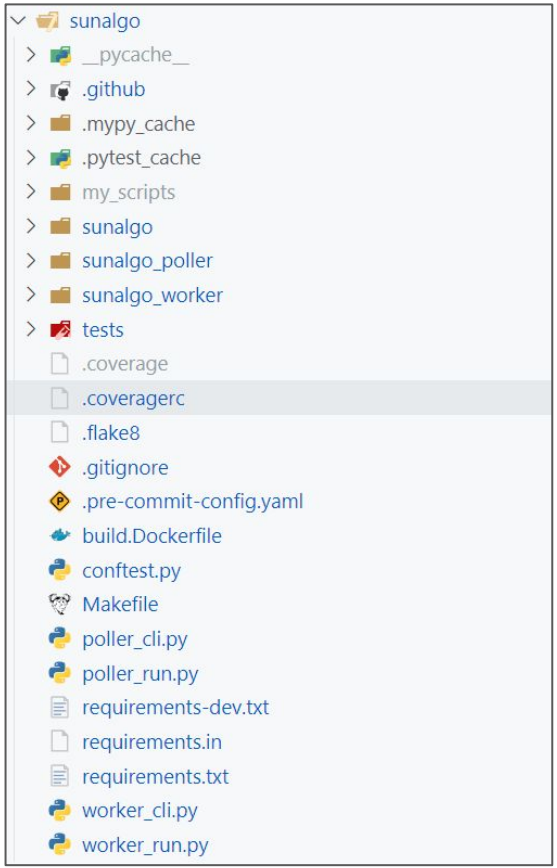

```python
# iterate over files and send the task
    target_files = files_names[:max_files] if max_files else files_names
    for file_name in target_files:
        logger.info(f"Processing file : {file_name}")
        payload_dict["file_name"] = file_name
        if append_file_name_to_output_dir:
            payload_dict["output_dir"] = (
                f"{output_dir}/{file_name}" if output_dir else file_name
            )
        bytes_payload = json.dumps(payload_dict).encode()
        date_str = datetime.now().strftime("%m-%d-%Y-%H-%M-%S-%f")
        task["name"] = client.task_path(
            project="sensav2",
            location="europe-west1",
            queue="poller-queue",
            task=f"{date_str}____{file_name.split('.')[0].replace(':', '_')}",
        )
        task["http_request"]["body"] = bytes_payload
        response = client.create_task(request={"parent": parent, "task": task})
        logger.info(f"Create task {response.name}")
    logger.info(f"Cloud Task : {len(target_files)} tasks sent to the queue")


if __name__ == "__main__":
    cli_()
```

Deploying to Cloud run



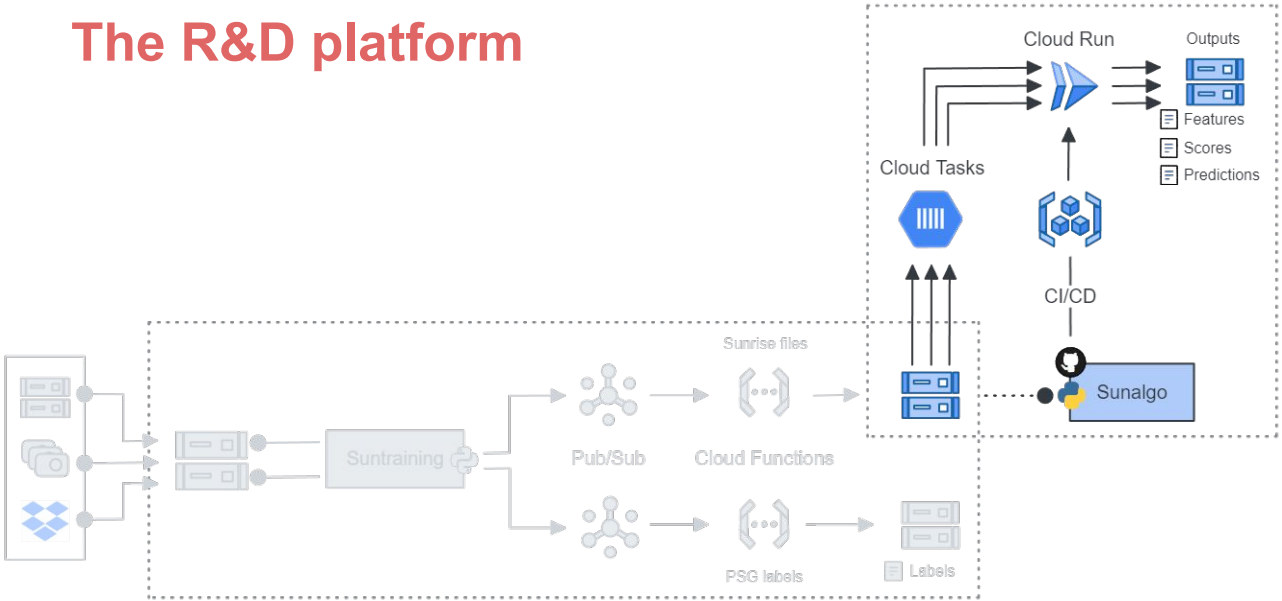

```yaml
name: sunalgo-cicd-dev

on:
  push:
    branches: ["dev"]
env:
  GCP_PROJECT_ID: "sensav2"
  ARTIFACT_REGISTRY: europe-west1-docker.pkg.dev/sensav2/sunalgo
  TAG: ${{ github.ref_name }}-${{github.run_id}}
jobs:
  build-and-test:
    runs-on: ubuntu-latest
    permissions:
      contents: 'read'
      id-token: 'write'
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Execute tests
        id: tests
        uses: docker/build-push-action@v4
        with:
          tags: sunalgo_dev_tests
          context: .
          target: test-env
          secrets: |
            "GCP_KEY=${{ secrets.SENSAV2_KEY }}"
          file: build.Dockerfile

      - name: GCP authentication
        id: auth
        uses: google-github-actions/auth@v1
        with:
          token_format: 'access_token'
          workload_identity_provider:
projects/350215846911/locations/global/workloadIdentityPools/sunrise-github/providers/github-actions
          service_account: github@sensav2.iam.gserviceaccount.com
```

Deploying to Cloud run



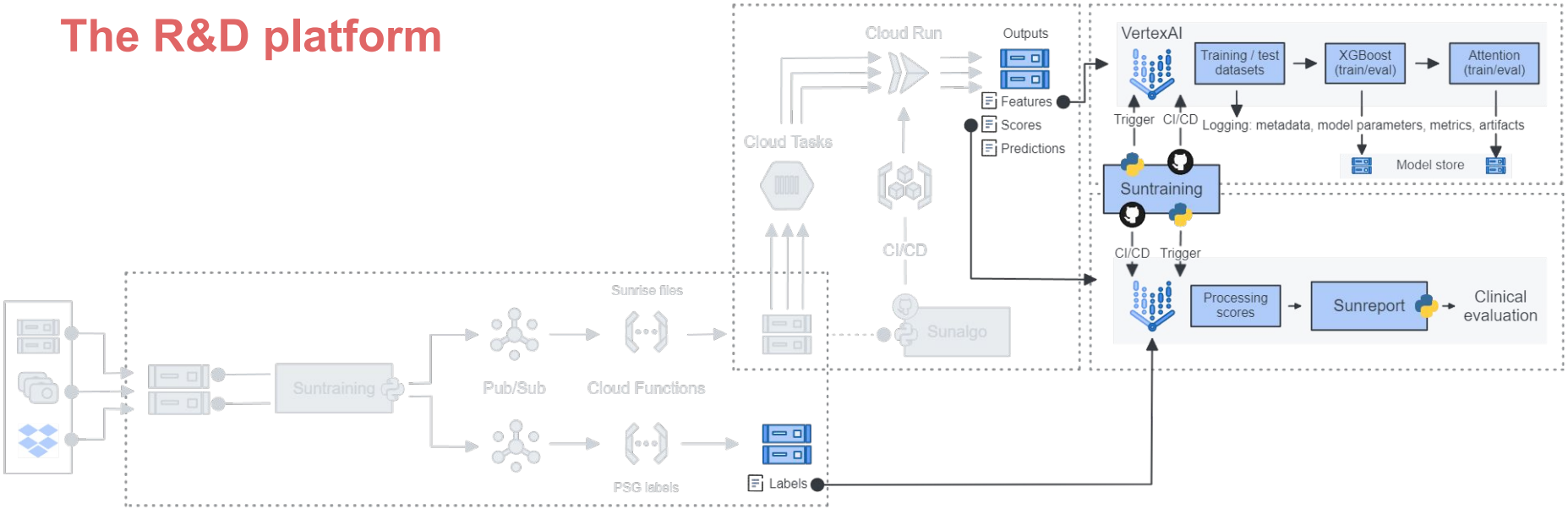

```yaml
- name: Login to Artifact Registry
        uses: docker/login-action@v2
        with:
          registry: europe-west1-docker.pkg.dev
          username: oauth2accesstoken
          password: ${{ steps.auth.outputs.access_token }}

      - name: Push worker
        uses: docker/build-push-action@v4
        with:
          push: true
          tags: |
            ${{ env.ARTIFACT_REGISTRY }}/sunalgo:${{ env.TAG }}_worker
          context: .
          target: runtime-worker
          secrets: |
            "GCP_KEY=${{ secrets.SENSAV2_KEY }}"
          file: build.Dockerfile

- name: Deploy worker
        uses: "google-github-actions/deploy-cloudrun@v1"
        with:
          service: "worker-dev"
          image: "${{ env.ARTIFACT_REGISTRY }}/sunalgo:${{ env.TAG }}_worker"
          region: europe-west1
          flags: '--concurrency=1 --max-instances=1000 --cpu=8000m --memory=32Gi --port=8080
--service-account=sunrise-worker-dev@sensav2.iam.gserviceaccount.com --timeout=3600'
```

The R&D platform



R&D Environment (DEV)

The R&D platform

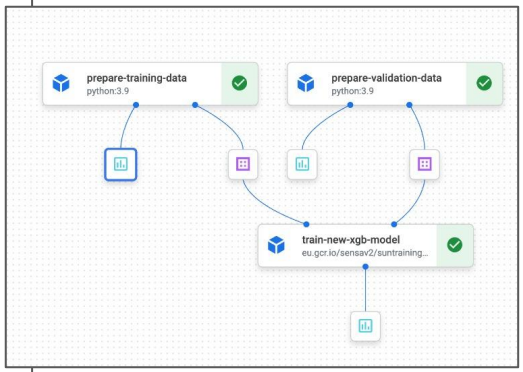


R&D Environment (DEV)

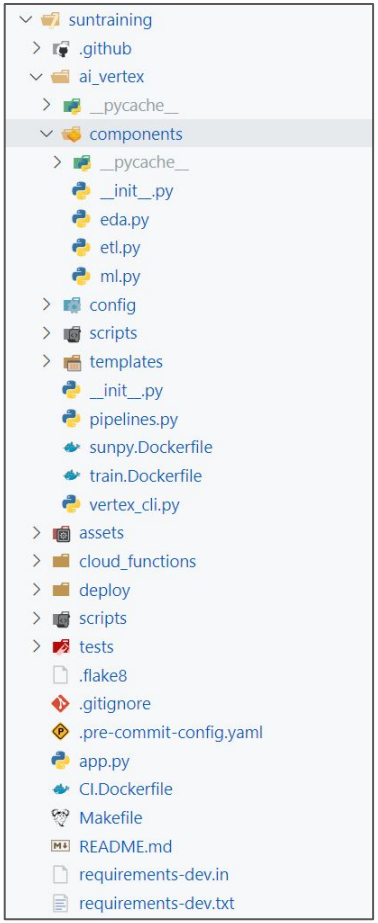
The pipeline

- suntraining
  - .github
  - ai_vertex
    - __pycache__
    - components
      - __pycache__
      - __init__.py
      - eda.py
      - etl.py
      - ml.py
    - config
    - scripts
    - templates
    - __init__.py
    - pipelines.py
    - sunpy.Dockerfile
    - train.Dockerfile
    - vertex_cli.py
  - assets
  - cloud_functions
  - deploy
  - scripts
  - tests
  - .flake8
  - .gitignore
  - .pre-commit-config.yaml
  - app.py
  - CI.Dockerfile
  - Makefile
  - README.md
  - requirements-dev.in
  - requirements-dev.txt

```python
@pipeline(
    pipeline_root=PIPELINE_ROOT,
    name="train-xgboost-challenger-sleep-stage-pipeline",
)
def train_xgb_challenger_sleep_stage_pipeline(
    experiment_name: str, experiment_run: str, staging_bucket: str, test_size: float
):
    prepare_training_data_task = (
        etl.prepare_training_data().set_cpu_limit("32").set_memory_limit("128G")
    )
    prepare_validation_data_task = (
        etl.prepare_validation_data().set_cpu_limit("32").set_memory_limit("128G")
    )

    with open(
        os.path.join(AI_VERTEX_PATH, "config", "sleep_stages_xgboost_params.json")
    ) as xgboost_conf:
        xgboost_params = json.load(xgboost_conf)
        (
            ml.train_new_xgb_model_sleep_stages(
                experiment_name=experiment_name,
                experiment_run=experiment_run,
                staging_bucket=staging_bucket,
                test_size=test_size,
                model_params=xgboost_params,
                train_src=prepare_training_data_task.outputs["training_dest"],
                valid_src=prepare_validation_data_task.outputs["validation_dest"],
            )
            .set_cpu_limit("32")
            .set_memory_limit("128G")
        )  # noqa
```



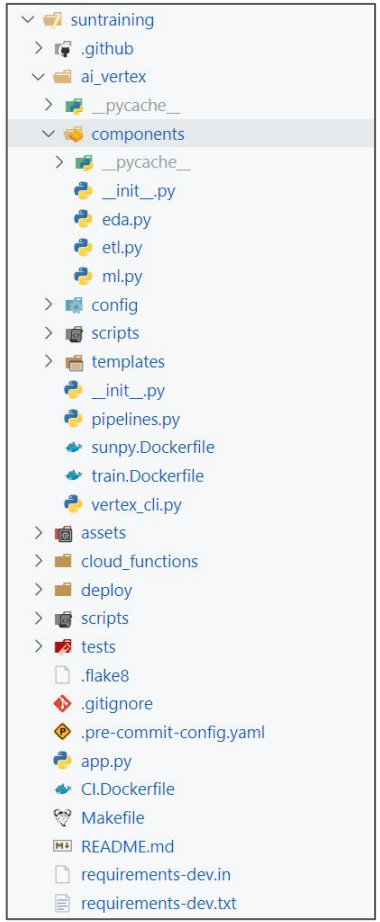
Anatomy of a Kubeflow component




```python
@component(
    packages_to_install=[
        "pandas",
        "scikit-learn",
        "google-cloud-aiplatform",
        "matplotlib",
    ],
    base_image="eu.gcr.io/sensav2/suntraining/sleep_stages:latest",
)
def train_new_xgb_model_sleep_stages(
    experiment_name: str,
    experiment_run: str,
    staging_bucket: str,
    test_size: float,
    model_params: dict,
    train_src: Input[Dataset],
    valid_src: Input[Dataset],
    confusion_mat: Output[ClassificationMetrics],
    loss_function_path: OutputPath(str),  # type: ignore
    loss_function_report: Output[Markdown],
):
    # ---- imports
    import pandas as pd
    from google.cloud import aiplatform
    from matplotlib import pyplot
    from sklearn import metrics
    from sklearn.model_selection import train_test_split
    from xgboost import XGBClassifier
    from sunpy import ml

    # ---- variable
    path_to_dir_model = "/gcs/training-patients-sun/PSG_Sensa/sleep_stages/models"

    # ---- Initialize Run
    print("INFO: Initialize experimental run")
    aiplatform.init(
        project="sensav2",
        staging_bucket=staging_bucket,
        experiment=experiment_name,
        location="europe-west1",
    )
```

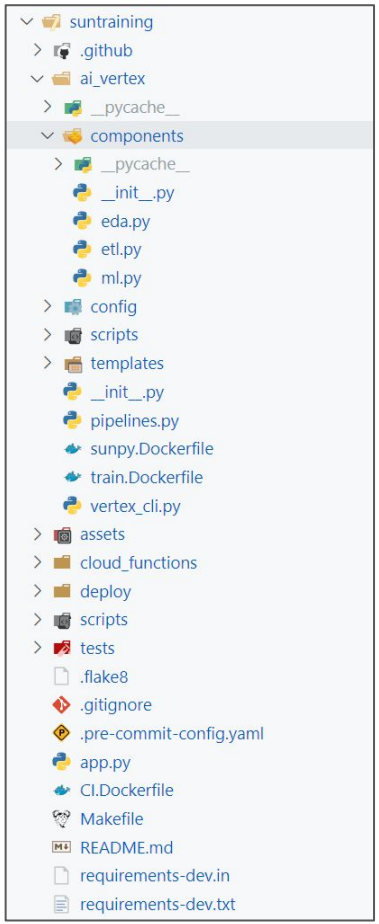
```dockerfile
FROM python:3.10.7-slim-buster

# Remove any third-party apt sources to avoid issues with expiring keys.
RUN rm -f /etc/apt/sources.list.d/*.list

# Install dependencies for python build
RUN    apt-get update \
    && apt-get install --no-install-recommends -qqy \
        gcc \
        gfortran \
        curl \
        ca-certificates \
        sudo \
        openssh-client \
        git \
        bzip2 \
        libx11-6 \
    && apt-get clean

ARG GCP_AUTH
RUN echo "${GCP_AUTH}" | base64 -d > /tmp/key-file.json
ARG export GOOGLE_APPLICATION_CREDENTIALS="/tmp/key-file.json"

ARG PIP_CONF
RUN echo "${PIP_CONF}" | base64 -d > pip.conf
ENV PIP_CONFIG_FILE=pip.conf

ARG PYPIRC
RUN echo "${PYPIRC}" | base64 -d > ~/.pypirc

RUN python3 -m pip install --upgrade pip \
    && pip install xgboost==1.3.3 \
    && pip install keyring \
    && pip install pip install keyrings.google-artifactregistry-auth \
    && pip install --index-url
https://europe-west1-python.pkg.dev/sensav2/sunrisepypi/simple/ sunpy
```

Tracking ML artifacts & parameters

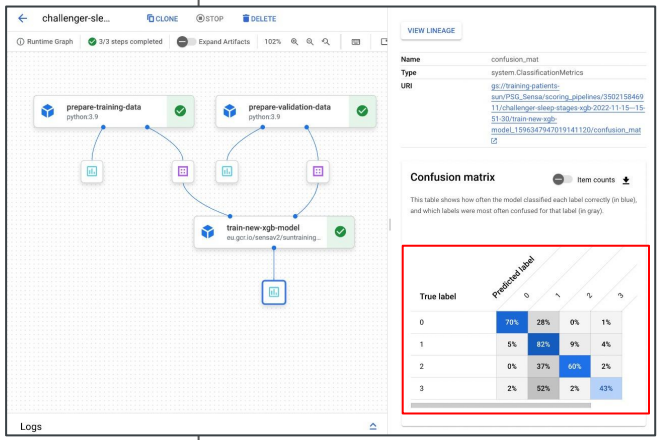



```python
with aiplatform.start_run(run=f"challenger-xgboost-{experiment_run}") as run:
        # --- Load datasets
        print("INFO: Load datasets")
        train_df = pd.read_csv(train_src.path)
        train_df.drop(columns=["night"], inplace=True)
        train_df.columns = [int(col) for col in train_df.columns]
        valid_df = pd.read_csv(valid_src.path)
        valid_df.drop(columns=["night"], inplace=True)
        valid_df.columns = [int(col) for col in valid_df.columns]
        X_train, X_test, y_train, y_test = train_test_split(
            train_df.values[:, 1:],
            train_df.values[:, :1],
            test_size=test_size,
            random_state=1,
        )
        X_valid, y_valid = (valid_df.values[:, 1:], valid_df.values[:, :1])

        # ---- Instantiate model and log parameters
        model = XGBClassifier(**model_params)

        # --- Train model
        print("INFO: Start training")
        model.fit(
            X=X_train,
            y=y_train,
            eval_set=[(X_train, y_train), (X_test, y_test), (X_valid, y_valid)],
            early_stopping_rounds=10,
            eval_metric="mlogloss",
            verbose=True,
        )

        # --- Save model
        print(
            f"INFO: Save model to our registry at {path_to_dir_model}/{experiment_run}"
        )
        model.save_model(f"{path_to_dir_model}/{experiment_run}")
```

Tracking artifacts




```python
# ---- Make predictions & Compute classification metrics
print("INFO: Compute predictions & metrics")
y_pred_from_train = model.predict(X_train)
y_pred_from_test = model.predict(X_test)
y_pred_from_valid = model.predict(X_valid)
acc_train, bacc_train, kapa_train = ml.compute_metrics(y_train, y_pred_from_train)
acc_test, bacc_test, kapa_test = ml.compute_metrics(y_test, y_pred_from_test)
acc_valid, bacc_valid, kapa_valid = ml.compute_metrics(y_valid, y_pred_from_valid)

confusion_mat.log_confusion_matrix(
            ["0", "1", "2", "3"],
            metrics.confusion_matrix(y_valid, y_pred_from_valid).tolist(),
        )

# ---- Collect metrics
print("INFO: collecting metrics")
metrics_to_log_for_run = {
    "acc_train": acc_train,
    "acc_test": acc_test,
    "acc_valid": acc_valid,
    "bacc_train": bacc_train,
    "bacc_test": bacc_test,
    "bacc_valid": bacc_valid,
    "kapa_train": kapa_train,
    "kapa_test": kapa_test,
    "kapa_valid": kapa_valid,
}
run.log_metrics(metrics_to_log_for_run)

try:
    run.log_params(model.get_params())
except Exception as exc:
    print(f"Cannot log params from model: {exc} : Load from json")
    run.log_params(model_params)

# --- Loss Function
results = model.evals_result()
pyplot.plot(results["validation_0"]["mlogloss"], label="train")
pyplot.plot(results["validation_1"]["mlogloss"], label="test")
pyplot.plot(results["validation_2"]["mlogloss"], label="valid")
pyplot.legend()
pyplot.savefig(f"{loss_function_path}.png")
```



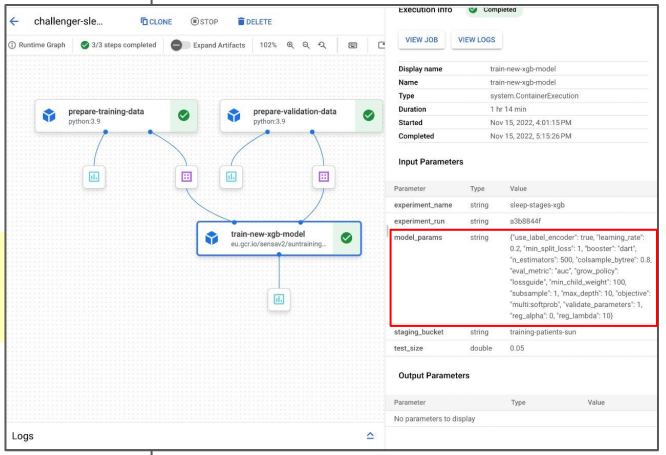
Tracking parameters

- suntraining
  - .github
  - ai_vertex
    - __pycache__
    - components
      - __pycache__
      - __init__.py
      - eda.py
      - etl.py
      - ml.py
    - config
    - scripts
    - templates
    - __init__.py
    - pipelines.py
    - sunpy.Dockerfile
    - train.Dockerfile
    - vertex_cli.py
  - assets
  - cloud_functions
  - deploy
  - scripts
  - tests
  - .flake8
  - .gitignore
  - .pre-commit-config.yaml
  - app.py
  - CI.Dockerfile
  - Makefile
  - README.md
  - requirements-dev.in
  - requirements-dev.txt

```python
# ---- Make predictions & Compute classification metrics
print("INFO: Compute predictions & metrics")
y_pred_from_train = model.predict(X_train)
y_pred_from_test = model.predict(X_test)
y_pred_from_valid = model.predict(X_valid)
acc_train, bacc_train, kapa_train = ml.compute_metrics(y_train, y_pred_from_train)
acc_test, bacc_test, kapa_test = ml.compute_metrics(y_test, y_pred_from_test)
acc_valid, bacc_valid, kapa_valid = ml.compute_metrics(y_valid, y_pred_from_valid)

confusion_mat.log_confusion_matrix(
            ["0", "1", "2", "3"],
            metrics.confusion_matrix(y_valid, y_pred_from_valid).tolist(),
        )

# ---- Collect metrics
print("INFO: collecting metrics")
metrics_to_log_for_run = {
    "acc_train": acc_train,
    "acc_test": acc_test,
    "acc_valid": acc_valid,
    "bacc_train": bacc_train,
    "bacc_test": bacc_test,
    "bacc_valid": bacc_valid,
    "kapa_train": kapa_train,
    "kapa_test": kapa_test,
    "kapa_valid": kapa_valid,
}
run.log_metrics(metrics_to_log_for_run)
```

```python
try:
    run.log_params(model.get_params())
except Exception as exc:
    print(f"Cannot log params from model: {exc} : Load from json")
    run.log_params(model_params)
```

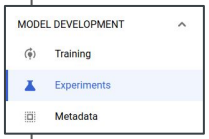
```python
# --- Loss Function
results = model.evals_result()
pyplot.plot(results["validation_0"]["mlogloss"], label="train")
pyplot.plot(results["validation_1"]["mlogloss"], label="test")
pyplot.plot(results["validation_2"]["mlogloss"], label="valid")
pyplot.legend()
pyplot.savefig(f"{loss_function_path}.png")
```

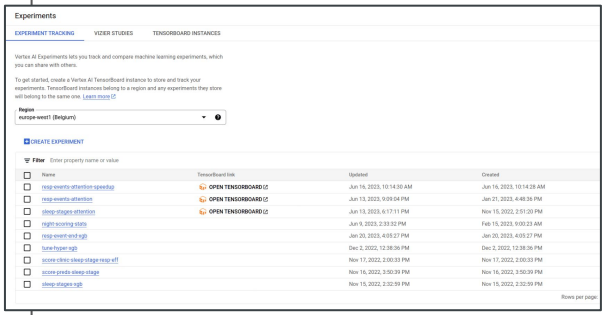


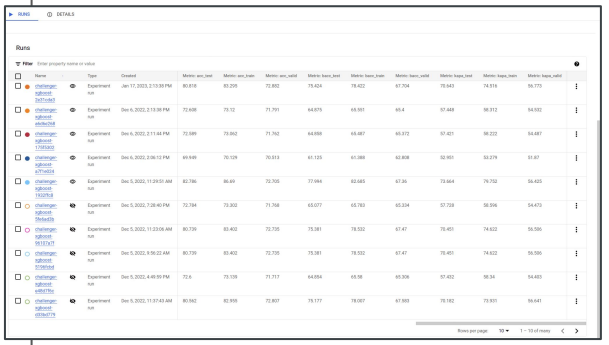
Tracking experimental run metrics

- suntraining
  - .github
  - ai_vertex
    - __pycache__
    - components
      - __pycache__
      - __init__.py
      - eda.py
      - etl.py
      - ml.py
    - config
    - scripts
    - templates
    - __init__.py
    - pipelines.py
    - sunpy.Dockerfile
    - train.Dockerfile
    - vertex_cli.py
  - assets
  - cloud_functions
  - deploy
  - scripts
  - tests
  - .flake8
  - .gitignore
  - .pre-commit-config.yaml
  - app.py
  - CI.Dockerfile
  - Makefile
  - README.md
  - requirements-dev.in
  - requirements-dev.txt

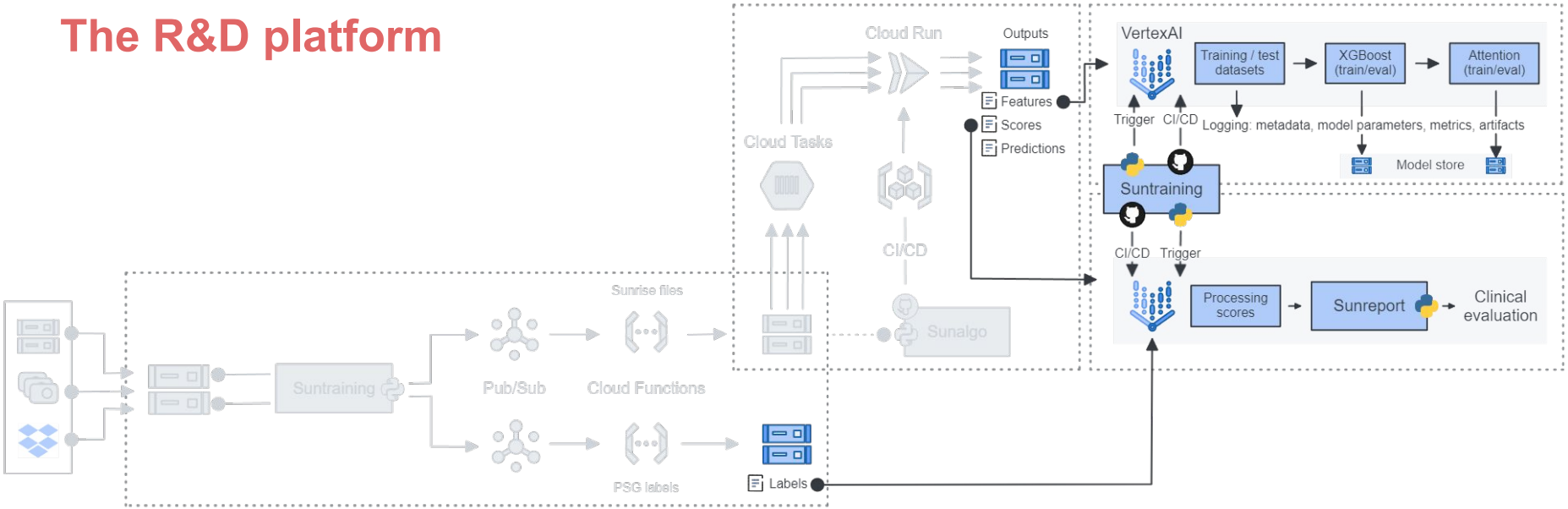
```python
# ---- Make predictions & Compute classification metrics
print("INFO: Compute predictions & metrics")
y_pred_from_train = model.predict(X_train)
y_pred_from_test = model.predict(X_test)
y_pred_from_valid = model.predict(X_valid)
acc_train, bacc_train, kapa_train = ml.compute_metrics(y_train, y_pred_from_train)
acc_test, bacc_test, kapa_test = ml.compute_metrics(y_test, y_pred_from_test)
acc_valid, bacc_valid, kapa_valid = ml.compute_metrics(y_valid, y_pred_from_valid)

confusion_mat.log_confusion_matrix(
            ["0", "1", "2", "3"],
            metrics.confusion_matrix(y_valid, y_pred_from_valid).tolist(),
        )

# ---- Collect metrics
print("INFO: collecting metrics")
metrics_to_log_for_run = {
    "acc_train": acc_train,
    "acc_test": acc_test,
    "acc_valid": acc_valid,
    "bacc_train": bacc_train,
    "bacc_test": bacc_test,
    "bacc_valid": bacc_valid,
    "kapa_train": kapa_train,
    "kapa_test": kapa_test,
    "kapa_valid": kapa_valid,
}
run.log_metrics(metrics_to_log_for_run)

try:
    run.log_params(model.get_params())
except Exception as exc:
    print(f"Cannot log params from model: {exc} : Load from json")
    run.log_params(model_params)

# --- Loss Function
results = model.evals_result()
pyplot.plot(results["validation_0"]["mlogloss"], label="train")
pyplot.plot(results["validation_1"]["mlogloss"], label="test")
pyplot.plot(results["validation_2"]["mlogloss"], label="valid")
pyplot.legend()
pyplot.savefig(f"{loss_function_path}.png")
```







The R&D platform

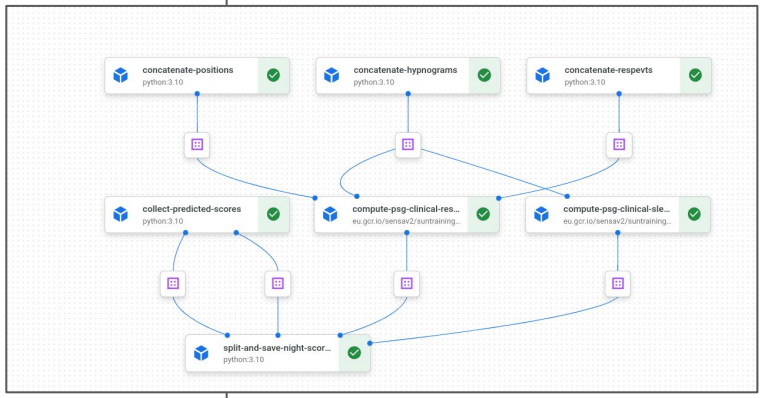


R&D Environment (DEV)

Evaluation

- suntraining
  - .github
  - ai_vertex
    - __pycache__
    - components
      - __pycache__
      - __init__.py
      - eda.py
      - etl.py
      - ml.py
    - config
    - scripts
    - templates
    - __init__.py
    - pipelines.py
    - sunpy.Dockerfile
    - train.Dockerfile
    - vertex_cli.py
  - assets
  - cloud_functions
  - deploy
  - scripts
  - tests
  - .flake8
  - .gitignore
  - .pre-commit-config.yaml
  - app.py
  - CI.Dockerfile
  - Makefile
  - README.md
  - requirements-dev.in
  - requirements-dev.txt

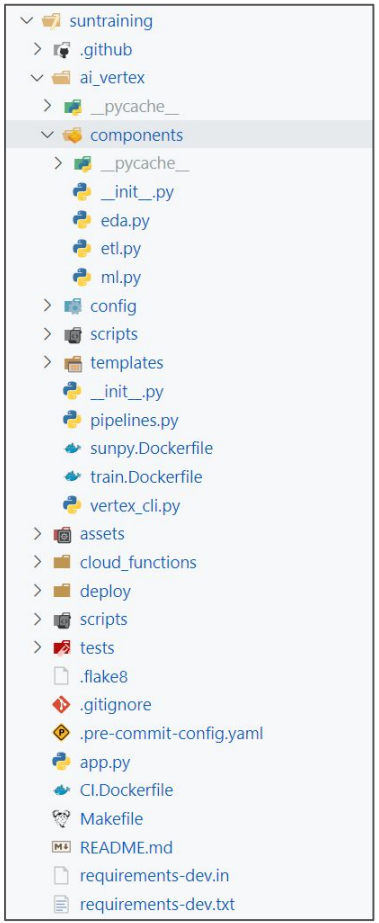
```python
@pipeline(
    pipeline_root=PIPELINE_ROOT,
    name="evaluate-clinical-scores",
)
def evaluate_clinical_scores(
    sunalgo_labels_path: str,
    sunalgo_outputs_path: str,
    scores_saving_path: str,
    device: str,
    experiment_name: str,
    experiment_run: str,
    staging_bucket: str,
):
    concatenate_hypnograms_task = etl.concatenate_hypnograms(
        sunalgo_labels_path=sunalgo_labels_path
    )
    concatenate_position_task = etl.concatenate_positions(
        sunalgo_labels_path=sunalgo_labels_path
    )
    concatenate_respevts_task = etl.concatenate_respevts(
        sunalgo_labels_path=sunalgo_labels_path
    )
    concatenate_marousals_task = etl.concatenate_marousal(
        sunalgo_labels_path=sunalgo_labels_path
    )
    clinical_score_task = eda.compute_psg_clinical_sleep_scores(  # noqa
        hypnogram_labels=concatenate_hypnograms_task.output,
        marousal_labels=concatenate_marousals_task.output,
        device=device,
    )
    clinical_resp_score_task = eda.compute_psg_clinical_respiratory_scores(  # noqa
        hypnogram_labels=concatenate_hypnograms_task.output,
        positions_labels=concatenate_position_task.output,
        respevts_labels=concatenate_respevts_task.output,
        device=device,
    )
    collect_predicted_scores_task = etl.collect_predicted_scores(  # noqa
        sunalgo_outputs_path=sunalgo_outputs_path
    )
    write_scores_task = etl.split_and_save_night_scores(  # noqa
        scores_saving_path=scores_saving_path,
        predicted_scores=collect_predicted_scores_task.outputs["predicted_scores"],
        psg_sleep_scores=clinical_score_task.output,
        psg_resp_scores=clinical_resp_score_task.output,
    )
```



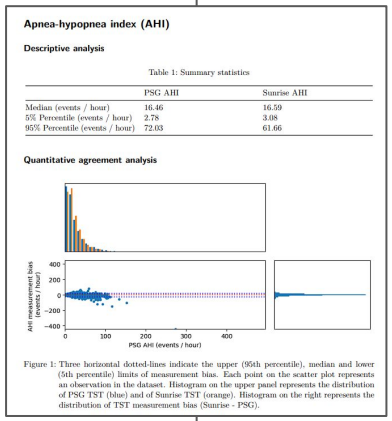
papermill

quarto

Evaluation




```python
@pipeline(
    pipeline_root=PIPELINE_ROOT,
    name="evaluate-clinical-scores",
)
def evaluate_clinical_scores(
    sunalgo_labels_path: str,
    sunalgo_outputs_path: str,
    scores_saving_path: str,
    device: str,
    experiment_name: str,
    experiment_run: str,
    staging_bucket: str,
):
    concatenate_hypnograms_task = etl.concatenate_hypnograms(
        sunalgo_labels_path=sunalgo_labels_path
    )
    concatenate_position_task = etl.concatenate_positions(
        sunalgo_labels_path=sunalgo_labels_path
    )
    concatenate_respevts_task = etl.concatenate_respevts(
        sunalgo_labels_path=sunalgo_labels_path
    )
    concatenate_marousals_task = etl.concatenate_marousal(
        sunalgo_labels_path=sunalgo_labels_path
    )
    clinical_score_task = eda.compute_psg_clinical_sleep_scores(  # noqa
        hypnogram_labels=concatenate_hypnograms_task.output,
        marousal_labels=concatenate_marousals_task.output,
        device=device,
    )
    clinical_resp_score_task = eda.compute_psg_clinical_respiratory_scores(  # noqa
        hypnogram_labels=concatenate_hypnograms_task.output,
        positions_labels=concatenate_position_task.output,
        respevts_labels=concatenate_respevts_task.output,
        device=device,
    )
    collect_predicted_scores_task = etl.collect_predicted_scores(  # noqa
        sunalgo_outputs_path=sunalgo_outputs_path
    )
    write_scores_task = etl.split_and_save_night_scores(  # noqa
        scores_saving_path=scores_saving_path,
        predicted_scores=collect_predicted_scores_task.outputs["predicted_scores"],
        psg_sleep_scores=clinical_score_task.output,
        psg_resp_scores=clinical_resp_score_task.output,
    )
```



Apnea-hypopnea index (AHI)

Descriptive analysis

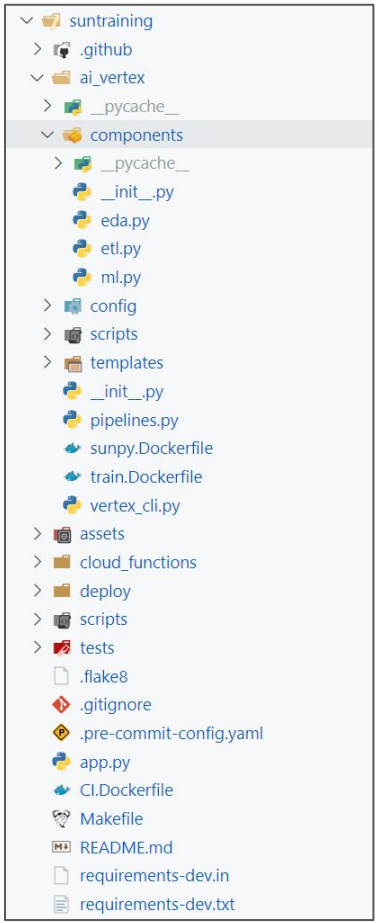
Table 1: Summary statistics

	PSG AHI	Sunrise AHI
Median (events / hour)	16.46	16.59
5% Percentile (events / hour)	2.78	3.08
95% Percentile (events / hour)	72.03	61.66

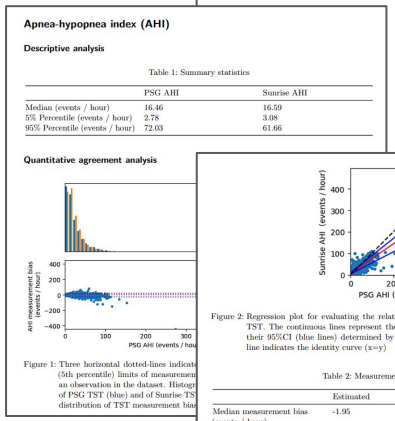
Quantitative agreement analysis

Figure 1: Three horizontal dotted-lines indicate the upper (95th percentile), median and lower (5th percentile) limits of measurement bias. Each point on the scatter plot represents an observation in the dataset. Histogram on the upper panel represents the distribution of PSG TST (blue) and of Sunrise TST (orange). Histogram on the right represents the distribution of TST measurement bias (Sunrise - PSG).

Evaluation




```python
@pipeline(
    pipeline_root=PIPELINE_ROOT,
    name="evaluate-clinical-scores",
)
def evaluate_clinical_scores(
    sunalgo_labels_path: str,
    sunalgo_outputs_path: str,
    scores_saving_path: str,
    device: str,
    experiment_name: str,
    experiment_run: str,
    staging_bucket: str,
):
    concatenate_hypnograms_task = etl.concatenate_hypnograms(
        sunalgo_labels_path=sunalgo_labels_path
    )
    concatenate_position_task = etl.concatenate_positions(
        sunalgo_labels_path=sunalgo_labels_path
    )
    concatenate_respevts_task = etl.concatenate_respevts(
        sunalgo_labels_path=sunalgo_labels_path
    )
    concatenate_marousals_task = etl.concatenate_marousal(
        sunalgo_labels_path=sunalgo_labels_path
    )
    clinical_score_task = eda.compute_psg_clinical_sleep_scores(  # noqa
        hypnogram_labels=concatenate_hypnograms_task.output,
        marousal_labels=concatenate_marousals_task.output,
        device=device,
    )
    clinical_resp_score_task = eda.compute_psg_clinical_respiratory_scores(  # noqa
        hypnogram_labels=concatenate_hypnograms_task.output,
        positions_labels=concatenate_position_task.output,
        respevts_labels=concatenate_respevts_task.output,
        device=device,
    )
    collect_predicted_scores_task = etl.collect_predicted_scores(  # noqa
        sunalgo_outputs_path=sunalgo_outputs_path
    )
    write_scores_task = etl.split_and_save_night_scores(  # noqa
        scores_saving_path=scores_saving_path,
        predicted_scores=collect_predicted_scores_task.outputs["predicted_scores"],
        psg_sleep_scores=clinical_score_task.output,
        psg_resp_scores=clinical_resp_score_task.output,
    )
```



Apnea-hypopnea index (AHI)

Descriptive analysis

Table 1: Summary statistics

	PSG AHI	Sunrise AHI
Median (events / hour)	16.46	16.59
5% Percentile (events / hour)	2.78	3.08
95% Percentile (events / hour)	72.03	61.66

Quantitative agreement analysis

Figure 1: Three horizontal dotted-lines indicate (5th percentile) limits of measurement an observation in the dataset. Histogr of PSG TST (blue) and of Sunrise TST distribution of TST measurement bia

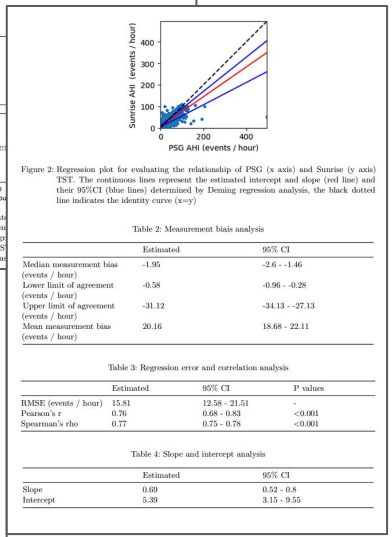


Figure 2: Regression plot for evaluating the relationship of PSG (x axis) and Sunrise (y axis) TST. The continuous lines represent the estimated intercept and slope (red line) and their 95%CI (blue lines) determined by Deming regression analysis, the black dotted line indicates the identity curve (x=y)

Table 2: Measurement biais analysis

	Estimated	95% CI
Median measurement bias (events / hour)	-1.95	-2.6 - -1.46
Lower limit of agreement (events / hour)	-0.58	-0.96 - -0.28
Upper limit of agreement (events / hour)	-31.12	-34.13 - -27.13
Mean measurement bias (events / hour)	20.16	18.68 - 22.11

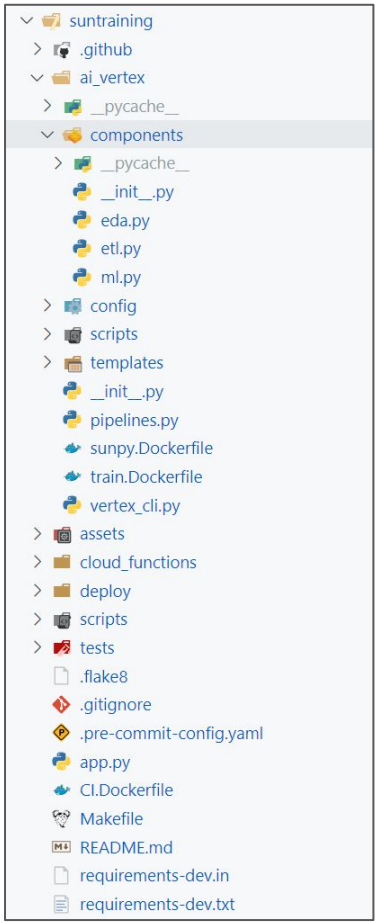
Table 3: Regression error and correlation analysis

	Estimated	95% CI	P values
RMSE (events / hour)	15.81	12.58 - 21.51	-
Pearson's r	0.76	0.68 - 0.83	<0.001
Spearman's rho	0.77	0.75 - 0.78	<0.001

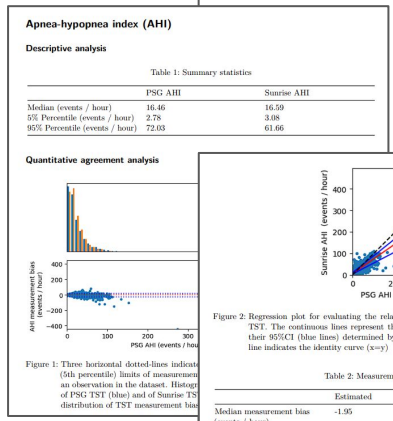
Table 4: Slope and intercept analysis

	Estimated	95% CI
Slope	0.69	0.52 - 0.8
Intercept	5.39	3.15 - 9.55

Evaluation




```python
@pipeline(
    pipeline_root=PIPELINE_ROOT,
    name="evaluate-clinical-scores",
)
def evaluate_clinical_scores(
    sunalgo_labels_path: str,
    sunalgo_outputs_path: str,
    scores_saving_path: str,
    device: str,
    experiment_name: str,
    experiment_run: str,
    staging_bucket: str,
):
    concatenate_hypnograms_task = etl.concatenate_hypnograms(
        sunalgo_labels_path=sunalgo_labels_path
    )
    concatenate_position_task = etl.concatenate_positions(
        sunalgo_labels_path=sunalgo_labels_path
    )
    concatenate_respevts_task = etl.concatenate_respevts(
        sunalgo_labels_path=sunalgo_labels_path
    )
    concatenate_marousals_task = etl.concatenate_marousal(
        sunalgo_labels_path=sunalgo_labels_path
    )
    clinical_score_task = eda.compute_psg_clinical_sleep_scores(  # noqa
        hypnogram_labels=concatenate_hypnograms_task.output,
        marousal_labels=concatenate_marousals_task.output,
        device=device,
    )
    clinical_resp_score_task = eda.compute_psg_clinical_respiratory_scores(  # noqa
        hypnogram_labels=concatenate_hypnograms_task.output,
        positions_labels=concatenate_position_task.output,
        respevts_labels=concatenate_respevts_task.output,
        device=device,
    )
    collect_predicted_scores_task = etl.collect_predicted_scores(  # noqa
        sunalgo_outputs_path=sunalgo_outputs_path
    )
    write_scores_task = etl.split_and_save_night_scores(  # noqa
        scores_saving_path=scores_saving_path,
        predicted_scores=collect_predicted_scores_task.outputs["predicted_scores"],
        psg_sleep_scores=clinical_score_task.output,
        psg_resp_scores=clinical_resp_score_task.output,
    )
```


Apnea-hypopnea index (AHI)

Descriptive analysis

Table 1: Summary statistics

	PSG AHI	Sunrise AHI
Median (events / hour)	16.46	16.59
5% Percentile (events / hour)	2.78	3.08
95% Percentile (events / hour)	72.03	61.66

Quantitative agreement analysis

Figure 1: Three horizontal dotted-lines indicate (5th percentile) limits of measurement an observation in the dataset. Histogr of PSG TST (blue) and of Sunrise TST distribution of TST measurement bia

Figure 2: Regression plot for evaluating the relationship of PSG (x axis) and Sunrise (y axis) TST. The continuous lines represent the estimated intercept and slope (red line) and their 95%CI (blue lines) determined by Deming regression analysis, the black dotted line indicates the identity curve (x=y)

Table 2: Measurement biais analysis

	Estimated	95% CI
Median measurement bias (events / hour)	-1.95	-2.6 - -1.46
Lower limit of agreement (events / hour)	-0.58	-0.96 - -0.28
Upper limit of agreement (events / hour)	-31.12	-34.13 - -27.13
Mean measurement bias (events / hour)	20.16	

Table 3: Regression error an

	Estimated	95
RMSE (events / hour)	15.81	12
Pearson's r	0.76	0.6
Spearman's rho	0.77	0.7

Table 4: Slope and int

	Estimated
Slope	0.69
Intercept	5.39

Table 1: Classification metrics for diagnostic

	Detecting AHI >= 5		Detecting AHI >= 15		Detecting AHI >= 30	
	Estimated	95% CI	Estimated	95% CI	Estimated	95% CI
Sensitivity	0.94	0.93 - 0.95	0.81	0.78 - 0.83	0.71	0.67 - 0.73
Specificity	0.43	0.39 - 0.49	0.75	0.73 - 0.77	0.93	0.91 - 0.94
PPV	0.92	0.92 - 0.93	0.79	0.77 - 0.81	0.78	0.75 - 0.81
NPV	0.50	0.46 - 0.56	0.77	0.74 - 0.79	0.90	0.88 - 0.91
FPR	0.57	0.51 - 0.61	0.25	0.23 - 0.27	0.07	0.06 - 0.09
FNR	0.06	0.05 - 0.07	0.19	0.17 - 0.22	0.29	0.27 - 0.33
LR+	1.66	1.54 - 1.85	3.26	2.98 - 3.59	9.60	8.18 - 11.01
LR-	0.14	0.11 - 0.17	0.26	0.23 - 0.29	0.31	0.29 - 0.36
F1 score	0.93	0.92 - 0.94	0.80	0.78 - 0.82	0.74	0.72 - 0.76
AUC	0.86	0.85 - 0.88	0.86	0.85 - 0.87	0.92	0.91 - 0.93
Accuracy	0.88	0.87 - 0.89	0.78	0.76 - 0.79	0.87	0.85 - 0.88
BAC	0.69	0.66 - 0.72	0.78	0.76 - 0.79	0.82	0.8 - 0.83

PPV/NPV: positive/negative predictive value, FPR/FPR: false positive/negative rate, LR+/LR-: positive/negative likelihood ratio, AUC: area under the curve, BAC: balanced accuracy.

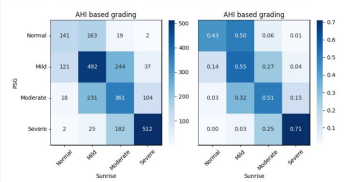
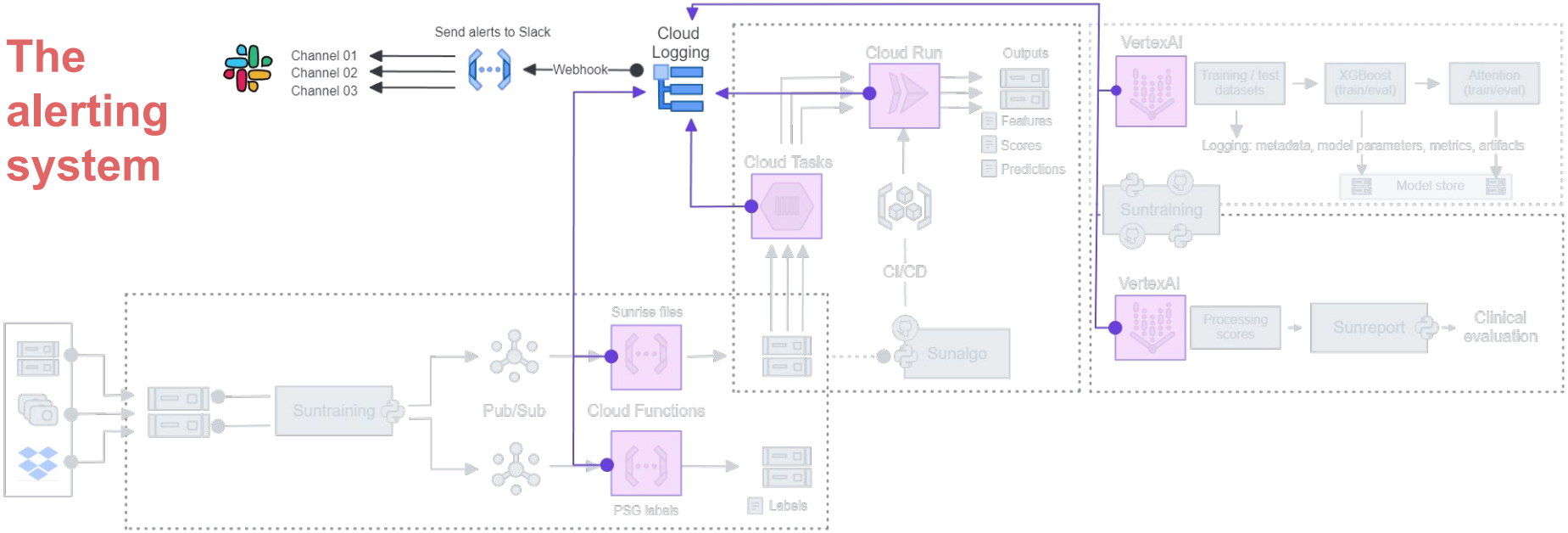


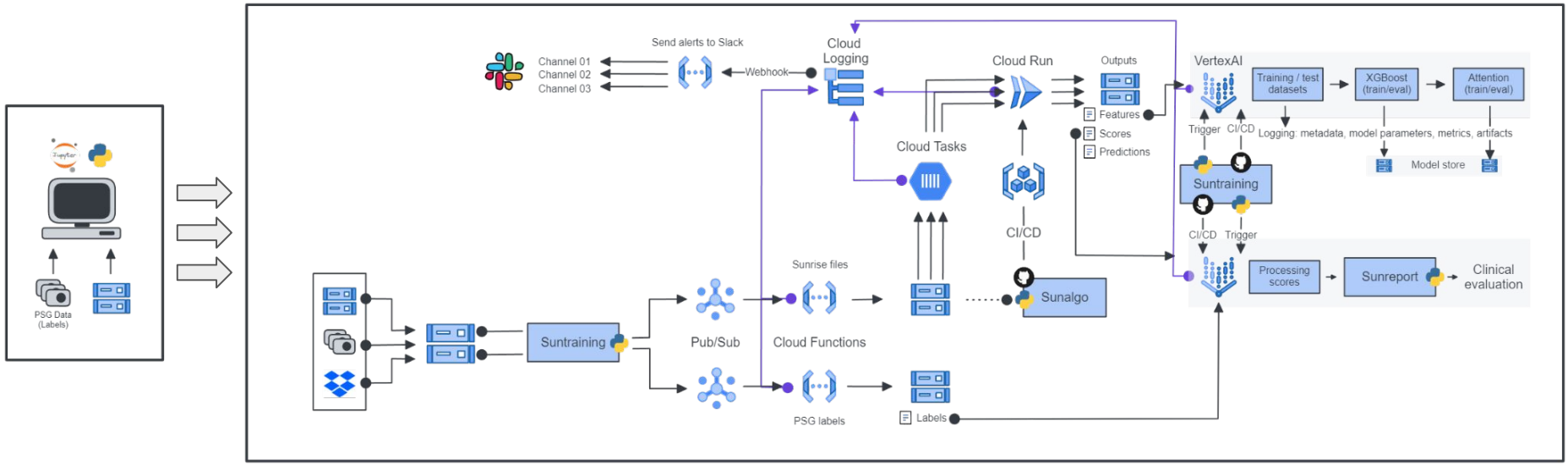
Figure 2: Confusion matrice for evaluating the OSA severity grading based on the Sunrise score and conventional thresholds (5, 15 and 30 e/h). The columns of the matrix represent the instances in the classified class and each row represents the instances in the true labels.

The alerting system

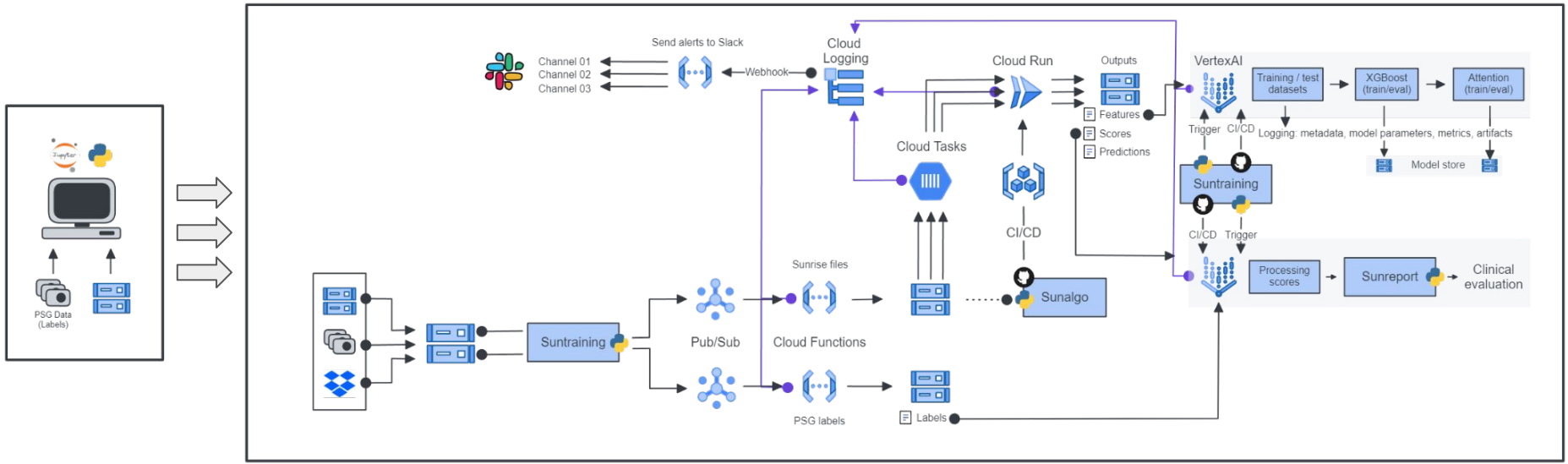


R&D Environment (DEV)

The R&D training platform

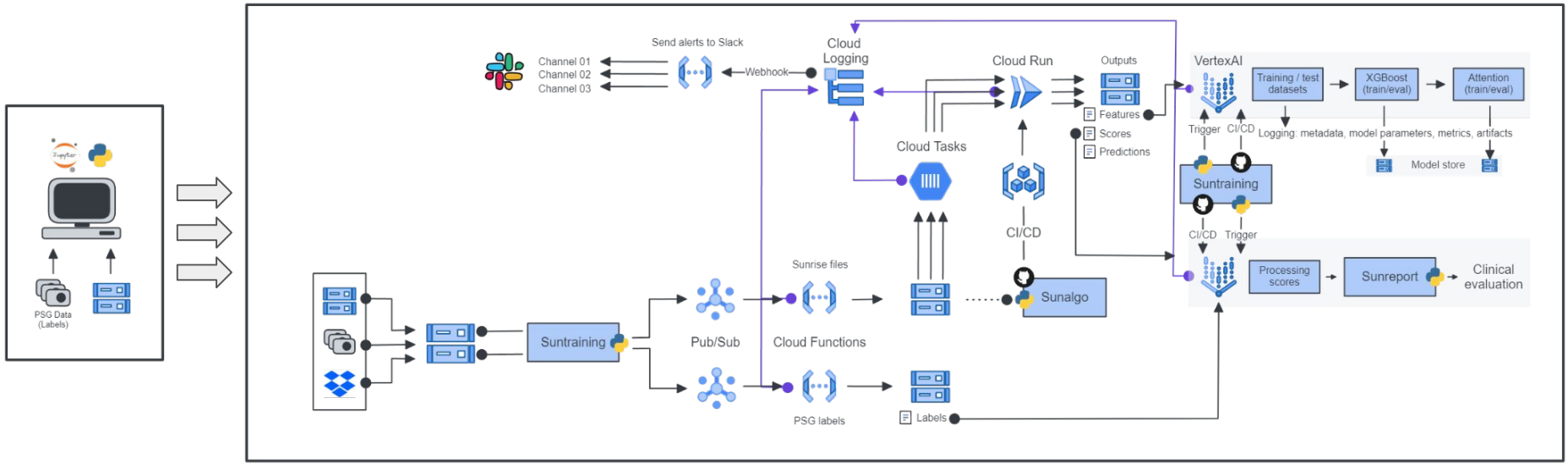


The R&D training platform



- Scalable (Cloud functions, Cloud Run)
- Reproducible (Vertex.ai, Github, Docker)
- Flexible (A lot of tooling, fast for *ad hoc*)
- Optimal cost

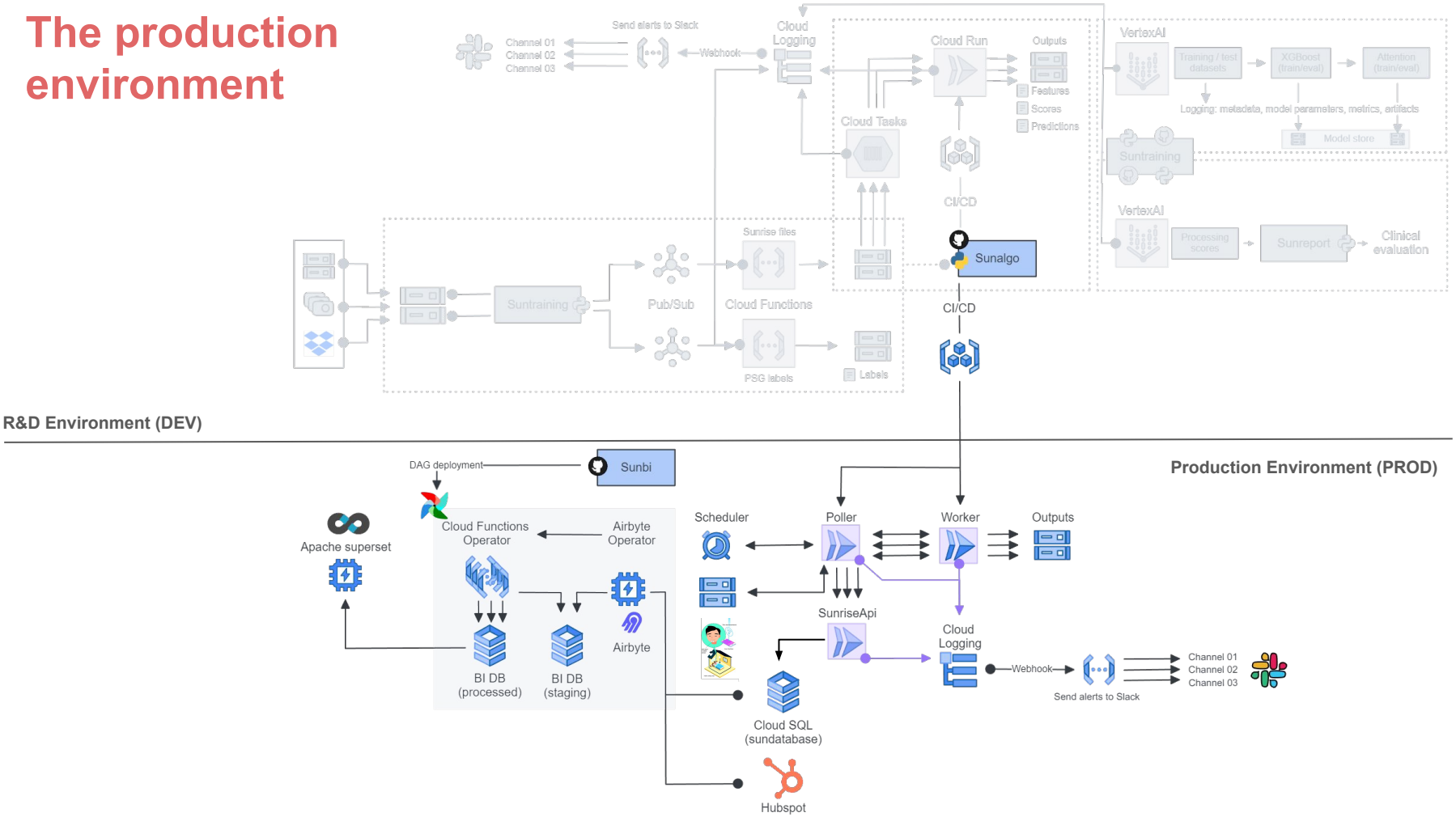
The R&D training platform



- Scalable (Cloud functions, Cloud Run)
- Reproducible (Vertex.ai, Github, Docker)
- Flexible (A lot of tooling, fast for *ad hoc*)
- Optimal cost

- Not fully automated (human loop present)
- Data lake lacks some organisation (data depth)
- Vertex.ai not as flexible as MLFlow
- Sunalgo is a monolithic
- Alerting and logging could be improved

The production environment



Rôle & contributions

- Design, implemented and plug together all the described components
- Code review, a lot of *ad hoc* meetings

- I was working at the interface with the research data scientists
- I was supporting data scientists for DevOps tasks, services monitoring, bug fixing, documentation

- Code contribution
  - Suntraining (90%)
  - Sunreport (100%)
  - Sunbi (50%)
  - Sunalgo (20%)

Many thanks for your attention - Q&A

Faouzi Braza, 23/11/2023