☁ DSS5208 Project 2 (Markdown Reference)

[0] Setup — Initialize Google Cloud Workbench and Environment

Before running any Dataproc job (ETL, Random Forest, GBT, etc.), please first prepare your Cloud Shell / Workbench session to connect with the correct project, region, and bucket. This section documents those steps.

```
# 0.1  Open Google Cloud Workbench or Cloud Shell
# You used the built-in Cloud Shell in your project console:
# https://console.cloud.google.com/
# Project ID : distributed-map-475111-h2
# Region     : asia-southeast1
# Bucket     : weather-ml-bucket-1760514177
```

Authenticate and Set Defaults

Make sure you are logged in and the right project is active:

```
gcloud auth login
gcloud config set project distributed-map-475111-h2
gcloud config set dataproc/region asia-southeast1
gcloud config list
```

Check Bucket and Paths

Confirm the GCS bucket already exists and that your folders follow the structure:

```
gsutil ls gs://weather-ml-bucket-1760514177/
# Expected sub-folders:
#   ├── data/                ← raw NOAA CSV files
#   ├── warehouse/noaa_parquet
#   ├── warehouse/noaa_clean_std
#   ├── scripts/             ← Python scripts (noaa_etl.py, noaa_train_rf.py,
noaa_train_gbt.py)
#   └── outputs/             ← model results and metrics
```

Set Convenient Environment Variables

These variables simplify later commands.

```
export PROJECT_ID="distributed-map-475111-h2"
export REGION="asia-southeast1"
export BUCKET="weather-ml-bucket-1760514177"
export PY_SCRIPTS="gs://${BUCKET}/scripts"
export DATA_IN="gs://${BUCKET}/data/csv"
export DATA_OUT="gs://${BUCKET}/warehouse/noaa_parquet"
```

Optional — Check Dataproc Permissions

Ensure your default Compute Engine service account has storage access:

```
PN=$(gcloud projects describe $PROJECT_ID --format="value(projectNumber)")
SA="${PN}-compute@developer.gserviceaccount.com"
gsutil iam ch serviceAccount:${SA}:roles/storage.objectAdmin gs://${BUCKET}
```

After this one-time initialization, please proceed with the first Dataproc Serverless ETL job below.

---

1 ETL Job — CSV ➜ Parquet

```
gcloud dataproc batches submit pyspark \
  gs://weather-ml-bucket-1760514177/scripts/noaa_etl.py \
  --region=asia-southeast1 \
  --deps-bucket=weather-ml-bucket-1760514177 \
  --subnet=default \
  -- \
  gs://weather-ml-bucket-1760514177/data/csv/*.csv \
  gs://weather-ml-bucket-1760514177/warehouse/noaa_parquet
```

Explanation:

- `batches submit pyspark` → runs the job **without needing a live cluster**.
- `--deps-bucket` → GCS bucket used for temporary job staging.
- Final two arguments are script parameters:
  1. Input CSV glob (the raw NOAA CSV files)
  2. Output path for partitioned Parquet files.

---

2 Cleanup Job — Parquet ➜ Cleaned Parquet (chunked mode)

```
gcloud dataproc batches submit pyspark \
  gs://weather-ml-bucket-1760514177/scripts/noaa_cleanup.py \
  --region=asia-southeast1 \
  --deps-bucket=weather-ml-bucket-1760514177 \
  --subnet=default \
  -- \
  gs://weather-ml-bucket-1760514177/warehouse/noaa_parquet \
  gs://weather-ml-bucket-1760514177/warehouse/noaa_clean_std \
  --chunked true
```

Explanation:

- The cleanup script reads the parquet data, fills missing values, applies filters, and writes out cleaned results.
- The flag `--chunked true` writes month-by-month partitions — safer for large datasets.

Monitor and Inspect Jobs

You can track your job's progress and logs here: Google Cloud Console → Dataproc → Batches

Or from Cloud Shell:

```
gcloud dataproc batches list --region=asia-southeast1
gcloud dataproc batches describe BATCH_ID --region=asia-southeast1
```

☑ Notes for `.md` documentation

Please include this table in the submission:

Step	Script	Input	Output	Mode	Command
1	noaa_etl.py	data/csv/*.csv	warehouse/noaa_parquet	Serverless	ETL Command
2	noaa_cleanup.py	warehouse/noaa_parquet	warehouse/noaa_clean_std	Serverless	Cleanup Command

3 Training Job — Random Forest (Cleaned Parquet → Model + Metrics)

💡 Training Method Used: Dataproc Serverless — "No-Cluster" (Serverless) Approach

◇ What it is

- The **Dataproc Serverless** mode lets you run **Spark jobs without creating or managing a cluster**.
- Google Cloud automatically provisions temporary compute resources to execute your PySpark script, then tears them down after completion.
- You only pay for the compute time used while the job is running — no cost for idle clusters.

◇ Why it's called "no-cluster"

- Unlike the traditional Dataproc model where you manually create a cluster (master + workers), serverless jobs don't require you to:
  - specify VM types,
  - manage nodes or scaling,
  - start/stop clusters manually.
- The infrastructure is fully managed by Google.

💻 CLI Command Used (Dataproc Serverless — RF Small Mode)

This is the structure of the command (ran it from Cloud Shell):

```
REGION="asia-southeast1"
PYFILE="gs://weather-ml-bucket-1760514177/scripts/noaa_train_rf.py"
IN="gs://weather-ml-bucket-1760514177/warehouse/noaa_clean_std"
OUT="gs://weather-ml-bucket-1760514177/outputs/rf_small"

gcloud dataproc batches submit pyspark $PYFILE \
  --region=$REGION \
  --deps-bucket=weather-ml-bucket-1760514177 \
  --properties spark.sql.shuffle.partitions=64,spark.default.parallelism=64 \
  -- $IN $OUT small
```

☑ **Three positional arguments to be provided:**

1. `$IN` → Input Parquet dataset (`noaa_clean_std`)
2. `$OUT` → Output folder for metrics and model (`rf_small`)

3. `small` → Mode flag for your script (`small` mode uses fewer trees & rows)

🗀 Output Location

All results were stored at:

```
gs://weather-ml-bucket-1760514177/outputs/rf_small
```

You can confirm contents with:

```
gsutil ls -r gs://weather-ml-bucket-1760514177/outputs/rf_small
gsutil cat gs://weather-ml-bucket-1760514177/outputs/rf_small/metrics/*.csv
```

Typical contents:

```
rf_small/
 ├── metrics/
 │   └── part-00000-*.csv         ← contains model & RMSE
 ├── sample_predictions_RandomForest/
 │   └── part-00000-*.csv         ← subset of predictions
 └── best_RandomForest_model/     ← saved Spark model
```

⚙ Key Characteristics of the `rf_small` Run

- Method: **Dataproc Serverless batch**
- Input: Clean Parquet (`noaa_clean_std`)
- Output: GCS folder `/outputs/rf_small`
- Scale: Small dataset (≈4.2 million rows)
- RMSE: ~36.7853
- No cluster logs or manual YARN management — all auto-handled by Serverless

4 Training — Random Forest (Dataproc Cluster)

> Here's the **rewritten, verified, and quota-accurate section** reflecting exactly how to creat and run the Dataproc cluster earlier (same parameters, same bucket, same mode structure).

> It covers **full** runs using `noaa_train_rf.py` script on a manually created Dataproc cluster. It mirrors the same inputs/outputs and keeps sizing within the current quotas.

4.1 Create the Dataproc Cluster

```
gcloud dataproc clusters create noaa-rf-cluster \
  --region=asia-southeast1 \
  --image-version=2.2-debian12 \
  --master-machine-type=n2-standard-4 \
  --master-boot-disk-size=100GB \
  --worker-machine-type=n2-standard-8 \
  --worker-boot-disk-size=150GB \
  --num-workers=2 \
  --optional-components=JUPYTER \
  --enable-component-gateway
```

> *Tip:* If you see "insufficient quota" errors, reduce machine sizes to `n2-standard-2` or use `--num-workers=1`.

4.2 Grant the Cluster Storage Access (One-Time)

```
PN=$(gcloud projects describe distributed-map-475111-h2 --format="value(projectNumber)")
SA="${PN}-compute@developer.gserviceaccount.com"

# Allow the Dataproc cluster service account to read/write to your bucket
gsutil iam ch serviceAccount:${SA}:roles/storage.objectAdmin gs://weather-ml-bucket-1760514177
```

4.3 Run the **Small (Sanity)** Random Forest Job (optional and I didn't run small test again in Cluster)

```
IN=gs://weather-ml-bucket-1760514177/warehouse/noaa_clean_std
OUT=gs://weather-ml-bucket-1760514177/outputs/rf_cluster_small

gcloud dataproc jobs submit pyspark \
  gs://weather-ml-bucket-1760514177/scripts/noaa_train_rf.py \
  --cluster=noaa-rf-cluster \
  --region=asia-southeast1 \
  --properties=spark.sql.adaptive.enabled=true,\
spark.sql.shuffle.partitions=64,spark.default.parallelism=64,\
spark.executor.instances=4,spark.executor.cores=1,\
spark.executor.memory=3g,spark.driver.memory=4g \
  -- \
  "$IN" "$OUT" false   # false = small mode
```

Expected behavior: This run processes roughly 1% of the data (a few million rows) and should complete within 30–60 minutes depending on load.

4.4 Run the **Full** Random Forest Job

```
IN=gs://weather-ml-bucket-1760514177/warehouse/noaa_clean_std
OUT=gs://weather-ml-bucket-1760514177/outputs/rf_cluster

gcloud dataproc jobs submit pyspark \
  gs://weather-ml-bucket-1760514177/scripts/noaa_train_rf.py \
  --cluster=noaa-rf-cluster \
  --region=asia-southeast1 \
  --properties=spark.sql.adaptive.enabled=true,\
spark.sql.shuffle.partitions=64,spark.default.parallelism=64,\
spark.executor.instances=4,spark.executor.cores=1,\
spark.executor.memory=3g,spark.driver.memory=4g \
  -- \
  "$IN" "$OUT" true    # true = full mode
```

Expected behavior: This run uses the entire cleaned NOAA dataset and produces the final model artifacts and metrics.

4.5 Monitor Job Progress and Logs

```
# List current jobs
gcloud dataproc jobs list --region=asia-southeast1

# Get details of a job (replace JOB_ID)
gcloud dataproc jobs describe JOB_ID --region=asia-southeast1 \
  --format="value(driverOutputResourceUri)"

# Stream recent driver logs
gsutil cat gs://dataproc-staging-asia-southeast1-*/google-cloud-dataproc-
metainfo/*/jobs/JOB_ID/driveroutput.000000001 | tail -n 200
```

You can also check logs visually in the Cloud Console under **Dataproc → Jobs → (Job ID)** or by visiting the Spark History Server link printed in the logs.

4.6 Review the Outputs

```
# View metrics for small and full runs
gsutil cat gs://weather-ml-bucket-1760514177/outputs/rf_small/metrics/*.csv
gsutil cat gs://weather-ml-bucket-1760514177/outputs/rf_cluster/metrics/*.csv

# Artifacts should include:
# - metrics/
# - best_RandomForest_model/
# - sample_predictions_RandomForest/
```

4.7 Delete the Cluster (Stop Charges)

```
gcloud dataproc clusters delete noaa-rf-cluster --region=asia-southeast1
```

5 Training — Gradient Boosted Trees (GBT) with Dataproc Serverless

5.1 Overview

The GBT training was migrated to **Dataproc Serverless for Spark** to bypass compute quota limits and enable flexible scaling. It uses the same cleaned dataset and maintains the same structure as Random Forest runs.

5.2 Input / Output Structure

Path	Description
`gs://weather-ml-bucket-1760514177/warehouse/noaa_clean_std`	Cleaned NOAA dataset
`gs://weather-ml-bucket-1760514177/scripts/noaa_train_gbt.py`	Updated GBT training script
`gs://weather-ml-bucket-1760514177/outputs/gbt_serverless_*`	Output directory for metrics, model, and predictions

5.3 Command — Small Mode (Sanity Test)

```
```

```
REGION="asia-southeast1"
IN="gs://weather-ml-bucket-1760514177/warehouse/noaa_clean_std"
OUT="gs://weather-ml-bucket-1760514177/outputs/gbt_serverless_small_$(date -u +%Y%m%d-%H%M%S)"

gcloud dataproc batches submit pyspark \
  gs://weather-ml-bucket-1760514177/scripts/noaa_train_gbt.py \
  --region="$REGION" \
  --deps-bucket=weather-ml-bucket-1760514177 \
  --
properties=spark.sql.adaptive.enabled=true,spark.sql.shuffle.partitions=64,spark.default.paralle
lism=64 \
  -- "$IN" "$OUT" false 0
```

5.4 Command — Full Mode (Production)

The following command launches the **full Gradient Boosted Trees (GBT)** training using **Dataproc Serverless** with the correct runtime flag `--version`.

```
REGION="asia-southeast1"
IN="gs://weather-ml-bucket-1760514177/warehouse/noaa_clean_std"
OUT="gs://weather-ml-bucket-1760514177/outputs/gbt_serverless_full_$(date -u +%Y%m%d-%H%M%S)"

gcloud dataproc batches submit pyspark \
  gs://weather-ml-bucket-1760514177/scripts/noaa_train_gbt.py \
  --region="$REGION" \
  --deps-bucket=weather-ml-bucket-1760514177 \
  --version="2.2" \
  --properties=spark.sql.adaptive.enabled=true,\
spark.sql.shuffle.partitions=192,\
spark.default.parallelism=192 \
  -- "$IN" "$OUT" true 128
```

Notes:

- `--version` replaces the deprecated `--runtime-version` flag.
- The run performs the full grid search and 3-fold cross-validation.
- Outputs will appear under the automatically timestamped folder in the bucket path above.
- The job can be monitored from the Dataproc → Batches Console.

5.5 Monitoring and Logs

```
gcloud dataproc batches list --region=asia-southeast1
gcloud dataproc batches describe <BATCH_ID> --region=asia-southeast1
```

Or in the console: 👉 Dataproc → Batches

5.6 Output Artifacts

Folder	Description
`metrics/`	RMSE and runtime summary
`cv_results/`	Cross-validation grid scores
`feature_importances/`	Top 50 features

Folder	Description
`sample_predictions/`	Predictions for sample rows
`best_GBT_model/`	Saved Spark GBT pipeline model

5.7 Advantages of Dataproc Serverless

- No cluster management or quota constraints
- Auto-scaling workers and ephemeral resources
- Pay-per-use billing
- Compatible with the same PySpark pipeline
- Full monitoring via Cloud Console

6 Comparison and Summary

6.1 Example Result Comparison

Model	Environment	RMSE	Runtime (min)
Random Forest	Dataproc Cluster	0.XXX	15
GBT	Dataproc Serverless	0.XXX	20

6.2 Quick Reference Table

Step	Script	Input	Output	Mode	Runner
1	`noaa_etl.py`	`data/csv/*.csv`	`warehouse/noaa_parquet`	ETL	**Dataproc Serverless (Data Processing)**
2	`noaa_cleanup.py`	`warehouse/noaa_parquet`	`warehouse/noaa_clean_std`	Cleanup	**Dataproc Serverless (Data Processing)**
3	`noaa_train_rf.py`	`warehouse/noaa_clean_std`	`outputs/rf_small`	RF-small	**Dataproc Serverless (Spark ML)**
4	`noaa_train_rf.py`	`warehouse/noaa_clean_std`	`outputs/rf_cluster`	RF-full	**Dataproc Cluster (Batch Spark)**
5	`noaa_train_gbt.py`	`warehouse/noaa_clean_std`	`outputs/gbt_serverless_small`	GBT-small	**Dataproc Serverless (Spark ML)**
6	`noaa_train_gbt.py`	`warehouse/noaa_clean_std`	`outputs/gbt_serverless_full`	GBT-full	**Dataproc Serverless (Spark ML)**

⚙ Clarification

- **Dataproc Serverless (Data Processing):** Used in ETL and Cleanup stages to perform large-scale data conversion and preprocessing without managing clusters.

- **Dataproc Serverless (Spark ML):** Used in RF-small and GBT training for machine learning model development, scaling automatically with Spark tasks.
- **Dataproc Cluster (Batch Spark):** Used in RF-full for full Random Forest training under managed cluster configuration.