



1. Data Flow (Lambda + RDS + S3)

A. Lambda Function 1 (e.g., get_new_eventlist_events)

- 1. **Pull New Data:** This Lambda is triggered on a schedule (e.g., weekly) via EventBridge. It queries an external API (DataGolf) for the latest event data.
- 2. **Update RDS:** It checks which events are new and inserts them into your MySQL RDS table (pga_with_sg_cat_eventslist).
- 3. **Pull Player-Round Data:** For any new events, it fetches detailed stats and inserts them into your player_round_stats_pga_with_sg_data table.
- 4. **Preprocess and Save to S3:**
 - As part of its workflow (or as a separate step), it can filter down to the columns you need for training (removing nulls, etc.).
 - Writes the cleaned CSV to S3. You can store each new dataset as its own CSV file in a dedicated folder (e.g., training_data/).

B. Lambda Function 2 (e.g., train_model)

- 1. **Triggered by S3 Event or Scheduled:**
 - When a new CSV is added to S3, you can have an S3 event notification trigger this function.
 - Alternatively, schedule the training (daily/weekly) via EventBridge if you don’t need real-time updates.
- 2. **Combine CSV Files:**
 - The Lambda lists all the CSVs in the S3 folder, downloads them, and combines them into a single DataFrame for training.
- 3. **Train Model:**
 - Uses a library like scikit-learn (you can include these dependencies in a Lambda Layer or in a container-based Lambda).
 - Trains the model on your chosen features (e.g., sg_total, driving_dist, driving_acc, etc.) to predict sg_t2g.
- 4. **Save Model to S3:**
 - Serialize the model (pickle, joblib, etc.) and store it in S3.
 - You can version the model or overwrite the existing file.

2. Model Serving (Docker + EC2)

A. Docker Container with Flask

- 1. **Pull the Latest Model from S3:**
 - When the container starts (or periodically), it downloads the latest model from your S3 bucket.
- 2. **Flask API Endpoint:**
 - You can create an endpoint, say /predict, that accepts JSON input with the features (e.g., sg_total, driving_dist, gir, etc.).
 - It loads the model from a local file (e.g., model.pkl) and returns the prediction in JSON format.
- 3. **Run on EC2 (or ECS/Fargate):**
 - You can run the container on a standard EC2 instance or deploy it via AWS ECS/Fargate for easier scaling.
- 4. **Public Endpoint:**
 - Expose port 5000 (or 80/443) so external requests (cURL, or a front-end UI) can hit the endpoint.

3. Putting It All Together

- 1. **User/Client** calls the Flask endpoint on EC2 with the features.
- 2. **Flask** loads the model from a local file system (which is kept updated by pulling from S3).
- 3. **Model** predicts the output (sg_t2g).
- 4. **Lambda Functions** keep the data fresh in RDS and S3, retraining the model whenever new data arrives.

This architecture separates concerns neatly:

- **Lambda 1** for data ingestion and CSV creation.
- **Lambda 2** for model training and S3 model artifact updates.
- **Docker/Flask** for inference.

Example Flow

- 1. **EventBridge** triggers get_new_eventlist_events (Lambda 1) on Tuesday.
- 2. Lambda 1 fetches new events from DataGolf, inserts them into pga_with_sg_cat_eventslist, and populates player_round_stats_pga_with_sg_data.
- 3. After cleaning and filtering the new data, Lambda 1 writes a new CSV (e.g., event_data_2025_02_25.csv) to S3.
- 4. An **S3 event** triggers train_model (Lambda 2).
- 5. Lambda 2:
 - Combines all CSVs in s3://my-bucket/training_data/ into one DataFrame.
 - Trains the model using scikit-learn.
 - Saves the updated model.pkl to s3://my-bucket/models/latest_model.pkl.
- 6. **EC2** with Docker + Flask:
 - On startup (or on a schedule), downloads latest_model.pkl from S3.
 - Exposes /predict to accept feature inputs and respond with a prediction.

4. Additional Tips

- 1. **Concurrency & Data Integrity:**
 - Using separate CSVs avoids concurrency issues of multiple writes to the same file.
 - If you expect a lot of parallel updates, consider using a queue or database triggers.
- 2. **Lambda Package Size:**
 - If your ML libraries are large, you can use container-based Lambdas or AWS Lambda Layers.
- 3. **Model Versioning:**
 - Keep multiple versions of your model if you want the option to roll back.
- 4. **Security:**
 - Ensure you use IAM roles to allow Lambda and EC2 to read/write to S3.
 - Use best practices for storing DB credentials (Secrets Manager, Parameter Store).
- 5. **Monitoring:**
 - Enable CloudWatch Logs for both Lambdas.
 - Monitor the health of your Flask container (e.g., use CloudWatch or a separate monitoring tool like Datadog).

