

# Telco Churn: Lightweight Inference Pipeline

# 1.Executive Summary

I successfully deployed the churn model as an automated AWS pipeline connected to my original code. Using AWS S3 and Lambda, I built a serverless inference system that loads the lightweight model artifact from main code and returns predictions via API Gateway.

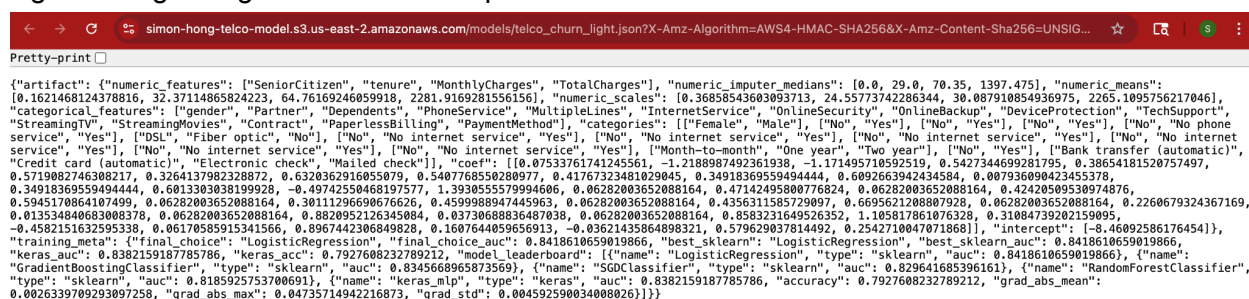
## 2.Approach & Methodology

I uploaded the lightweight artifact to AWS S3, used AWS Lambda with boto3 library to load the artifact from S3, parse the input payload, and return predictions via API Gateway. This architecture aims to extend this project beyond local analysis, but implement a serverless, automated inference pipeline. This system allows real-time churn predictions on new customer data without requiring persistent server or local dependencies. The aim was lightweight, where we used an extracted linear model.

## Architecture

The architecture follows the pipeline flow of S3, Lambda, and API Gateway. S3 serves as the model registry where the trained model parameters including weights, intercepts, and scaling factors are transformed to a JSON artifact and uploaded to S3 bucket. Then, the function in Lambda is triggered, which downloads the artifact and reconstructs the model to compute probability. Then, the API Gateway takes the Lambda function to the public and allows the client to request prediction.

Figure 1: Lightweight JSON artifact uploaded to S3



```
{
  "artifact": {
    "numeric_features": {
      "SeniorCitizen": "tenure",
      "MonthlyCharges": "TotalCharges",
      "numeric_imputer_medians": [0.0, 29.0, 70.35, 1397.475],
      "numeric_means": [0.1621468124378816, 32.37114865824223, 64.76169246059918, 2281.9169281556156],
      "numeric_scales": [0.36858543603093713, 24.55773742286344, 30.087910854936975, 2265.1095756217046]
    },
    "categorical_features": {
      "gender": "Partner",
      "Dependents": "PhoneService",
      "MultipleLines": "InternetService",
      "OnlineSecurity": "OnlineBackup",
      "DeviceProtection": "TechSupport",
      "StreamingTV": "StreamingMovies",
      "Contract": "PaperlessBilling",
      "PaymentMethod": "categories": [
        [{"Female": "Male"}, {"No": "Yes"}, {"No": "Yes"}, {"No": "Yes"}, {"No": "No phone service"}, {"Yes"}, {"DSL", "Fiber optic"}, {"No"}, {"No": "No internet service"}, {"Yes"}, {"No": "No internet service"}, {"Yes"}, {"No": "No internet service"}, {"Yes"}, {"Month-to-month", "One year", "Two year"}, {"No": "Yes"}, {"Bank transfer (automatic)", "Credit card (automatic)", "Electronic check", "Mailed check"}],
        "coef": [
          [0.07533761741245561, -1.2188987492361938, -1.171495710592519, 0.5427344699281795, 0.38654181520757497, 0.5719082746308217, 0.3264137982328872, 0.6320362916055079, 0.5407768550280977, 0.41767323481829045, 0.34918369559494444, 0.6092663942434584, 0.06282003652088164, 0.42420509530974876, 0.34918369559494444, 0.6013303038199928, -0.49742550468197577, 1.3930555579994606, 0.06282003652088164, 0.47142495800776824, 0.06282003652088164, 0.42420509530974876, 0.5945170864107499, 0.06282003652088164, 0.30111296690676626, 0.45999889474445963, 0.06282003652088164, 0.4356311585729097, 0.6695621208807928, 0.06282003652088164, 0.2260679324367169, 0.013534840683008378, 0.06282003652088164, 0.8828952126345084, 0.83730688836487038, 0.06282003652088164, 0.8583231649526352, 1.105817861076328, 0.31084739202159095, -0.4582151632595338, 0.06170585915341566, 0.8967442306849828, 0.1007644059656913, -0.0362143586489321, 0.579629837814492, 0.2542710047071868],
          "intercept": [-8.46092586176454]
        ],
        "training_meta": {
          "final_choice": "LogisticRegression",
          "final_choice_auc": 0.8418610659019866,
          "best_sklearn": "LogisticRegression",
          "best_sklearn_auc": 0.8418610659019866,
          "keras_auc": 0.8382159187785786,
          "keras_acc": 0.7927608232789212,
          "model_leaderboard": [
            {
              "name": "LogisticRegression",
              "type": "sklearn",
              "auc": 0.8418610659019866
            },
            {
              "name": "GradientBoostingClassifier",
              "type": "sklearn",
              "auc": 0.8345668965873569
            },
            {
              "name": "SGDClassifier",
              "type": "sklearn",
              "auc": 0.829641685396161
            },
            {
              "name": "RandomForestClassifier",
              "type": "sklearn",
              "auc": 0.8185925753700691
            },
            {
              "name": "keras_mlp",
              "type": "keras",
              "auc": 0.8382159187785786,
              "accuracy": 0.7927608232789212,
              "grad_abs_mean": 0.002639709293097258,
              "grad_abs_max": 0.04735714942216873,
              "grad_std": 0.004592590034008026
            }
          ]
        }
      }
    }
  }
}
```

## Implementation

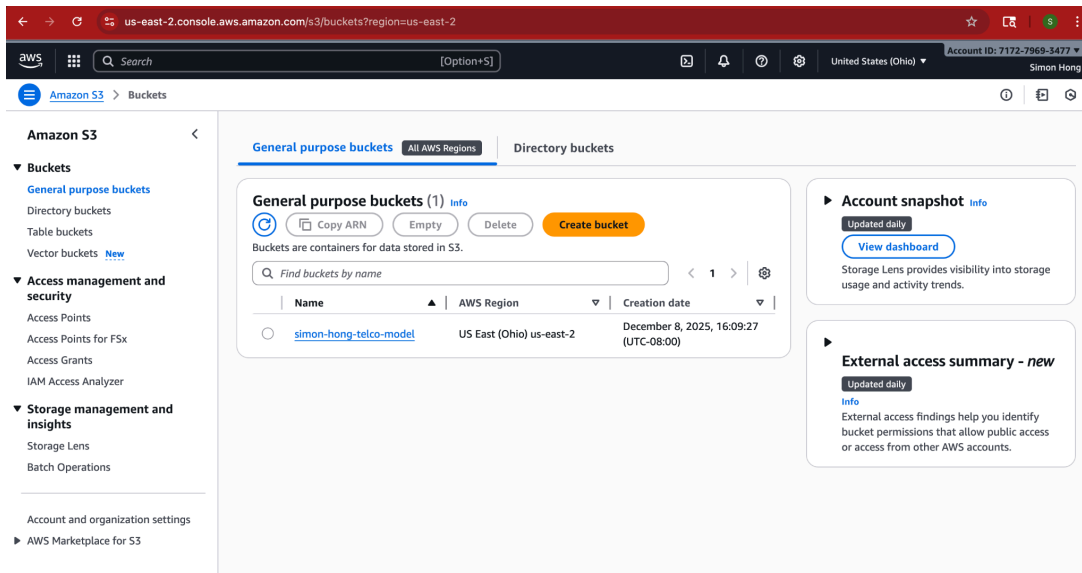
The key to this challenge was to keep everything lightweight for easy deployment. That means it is difficult to install libraries to AWS Lambda. So, I trained an SGD classifier locally using

Scikit-Learn, extracted the key coefficients and statistics through the artifact, and let Lambda do this directly.

## Amazon S3

I created a bucket with the name 'simon-hong-telco-model'. This works as the centralized model registry, which automatically uploads the artifact when the main script from my local device finishes.

Figure 2: Amazon S3 bucket



## AWS Lambda

I set the runtime to Python 3.14. Custom lambda functions including `lambda_handler` is used to fetch the JSON artifact from S3, which then applies the linear formula for SGD Classifier, and returns the prediction. I granted IAM permission to allow reading access to S3, and connect API Gateway as a method to stream the result out.

Figure 3: AWS Lambda

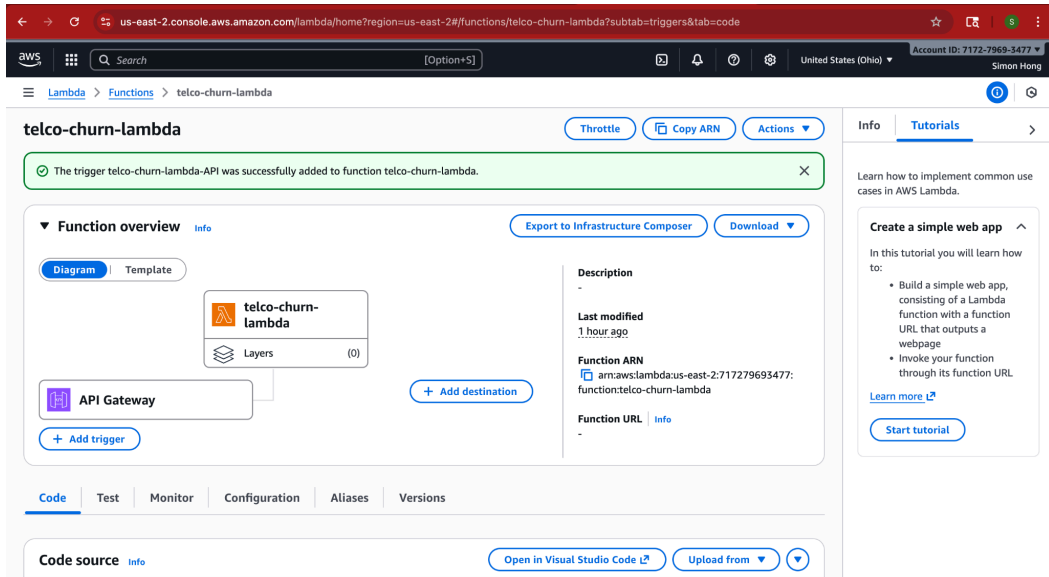


Figure 4: AWS Lambda IAM Permissions

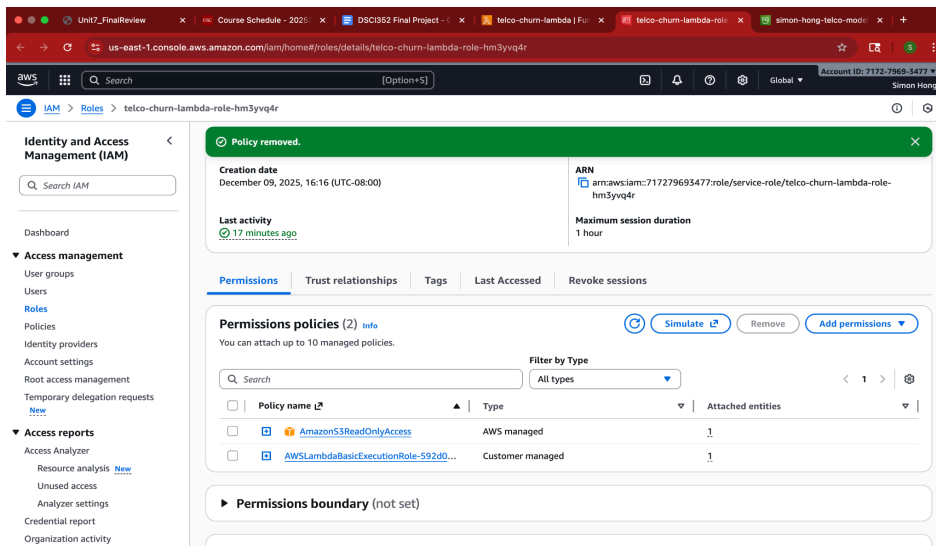
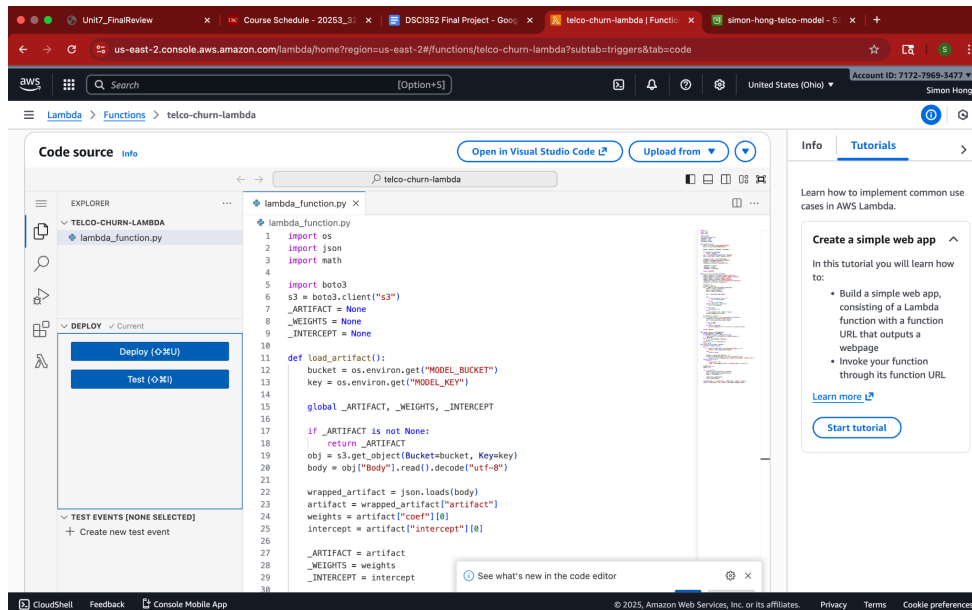


Figure 5: Custom lambda function uploaded to AWS Lambda



## API Gateway

I used HTTP API, which provides a public URL to trigger the Lambda function securely over AWS.

### Workflow

The pipeline is fully automated within the lambda functions in the main script `telco_churn_final.py`. The workflow starts at training and building the SGD Classifier from the main script. Then, the `upload_json_to_s3()` function builds and uploads it to S3 using the `boto3` library. The script then automatically selects a random sample customer for testing. The script sends POST requests to API Gateway URL with the customer data, where the Lambda processes the request using uploaded artifact and returns the Churn probability and label.

Figure 6: Automated AWS Pipeline execution result

```
✓ TERMINAL zsh + ▾ 🗑️

FINAL CHOICE (for your report): GradientBoostingClassifier AUC=0.8434
Saved model_leaderboard_telco.json

ENABLE_AWS_EXPORT=True → building lightweight artifact
and uploading to S3.
Uploaded to s3://simon-hong-telco-model/models/telco_churn_light.json
Uploaded lightweight artifact with metadata to S3.
SUCCESS. The API returned:
{'predictions': [0.005088353071451572], 'labels': [0]}
✨ (base) seunghyunhong@Seunghyunui-MacBookPro Final_Proje
ct %
```

This figure serves as the verification of success for the end-to-end pipeline. The local training script automatically triggered the API with a sample customer, and returned a prediction with the correct label, confirming infrastructure is fully operational.