

**CSE 330:
Cloud Computing**

PROJECT DETAILS

Submitted to

**SRI RAMACHANDRA INSTITUTE OF HIGHER EDUCATION AND RESEARCH SRI
RAMACHANDRA ENGINEERING AND TECHNOLOGY**

for the Award of the Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

(Cyber Security and Internet of Things) By

K.P. SHRIRAM - E0219040

NAVEEN PRATAP - E0219021

UMESH KUMAR - E0219019

Under the Supervision of

Prof. Ragavan Veerarajan



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRET, CHENNAI - 600116 AUGUST 2021

TOPIC : RECOMMENDATION SYSTEM USING AMAZON SAGEMAKER

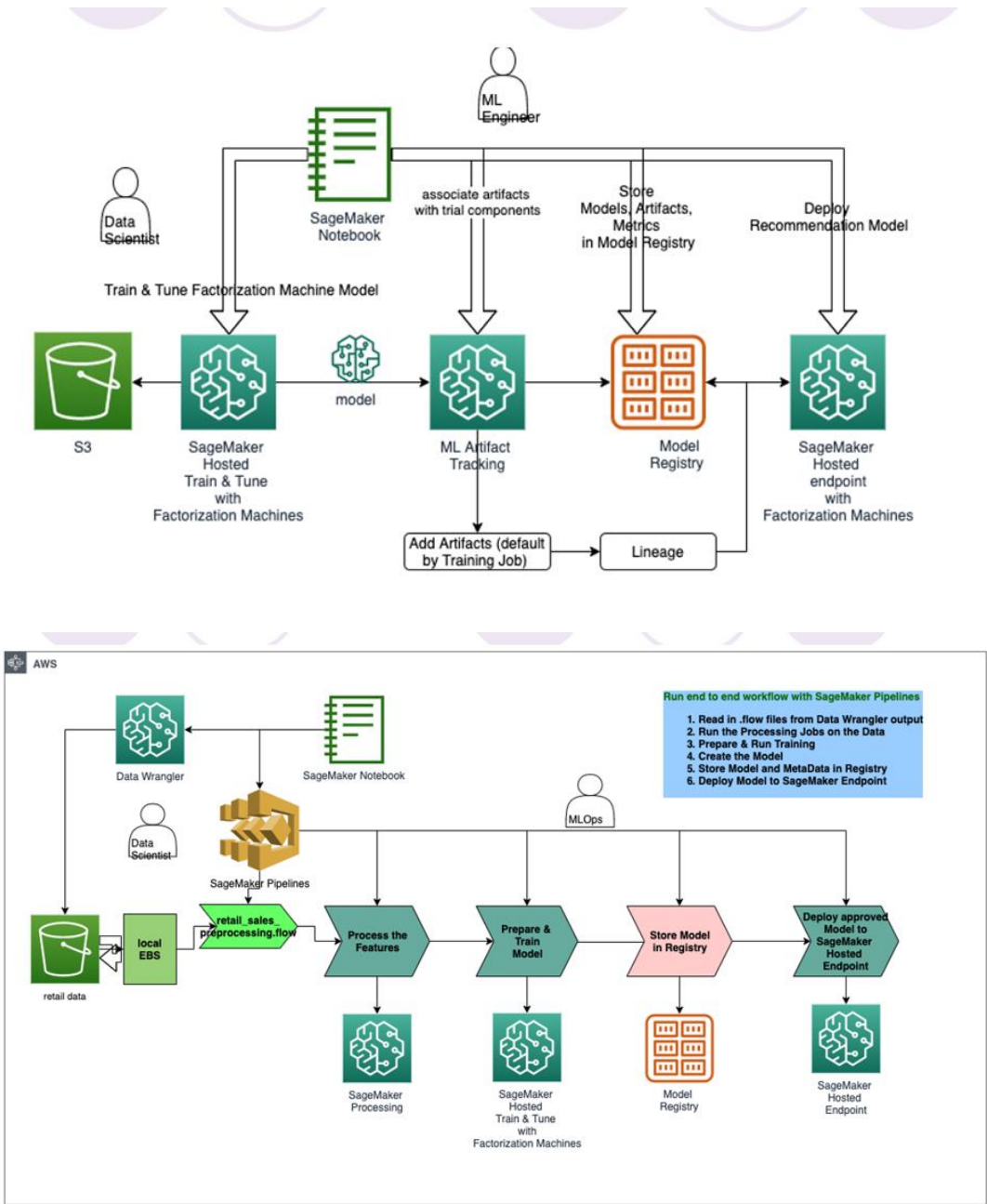
ABSTRACT:

Building a Recommendation system using Ecommerce data to predict the product recommendation to customers. Using Sagemaker for playing around with data and also using amazon pipelines to do automated task for prediction.

AIM OF THE PROJECT:

To get a best products for the customers to recommend from the ecommerce data through AWS SAGEMAKER

ARCHITECTURE:



SERVICES USED:

AWS SERVICES USED

- 1. AWS SAGEMAKER
- 2. AWS PIPELINES
- 3. AWS S3 BUCKET
- 4. AWS MODEL REGISTRY
- 5. AWS LINEAGE

AWS SAGEMAKER :

Amazon SageMaker

Prepare →

SageMaker Ground Truth

Label training data for machine learning

SageMaker Data Wrangler NEW

Aggregate and prepare data for machine learning

SageMaker Processing

Built-in Python, BYO R/Spark

SageMaker Feature Store NEW

Store, update, retrieve, and share features

SageMaker Clarify NEW

Detect bias and understand model predictions

Build →

SageMaker Studio Notebooks

Jupyter notebooks with elastic compute and sharing

Built-in and Bring-your-own Algorithms

Dozens of optimized algorithms or bring your own

Local Mode

Test and prototype on your local machine

SageMaker Autopilot

Automatically create machine learning models with full visibility

SageMaker JumpStart NEW

Pre-built solutions for common use cases

Train & tune →

One-click Training

Distributed infrastructure management

SageMaker Experiments

Capture, organize, and compare every step

Automatic Model Tuning

Hyperparameter optimization

Distributed Training Libraries NEW

Training for large datasets and models

SageMaker Debugger NEW

Debug and profile training runs

Managed Spot Training

Reduce training cost by 90%

Deploy & manage →

One-click Deployment

Fully managed, ultra low latency, high throughput

Kubernetes & Kubeflow Integration

Simplify Kubernetes-based machine learning

Multi-Model Endpoints

Reduce cost by hosting multiple models per instance

SageMaker Model Monitor

Maintain accuracy of deployed models

SageMaker Edge Manager NEW

Manage and monitor models on edge devices

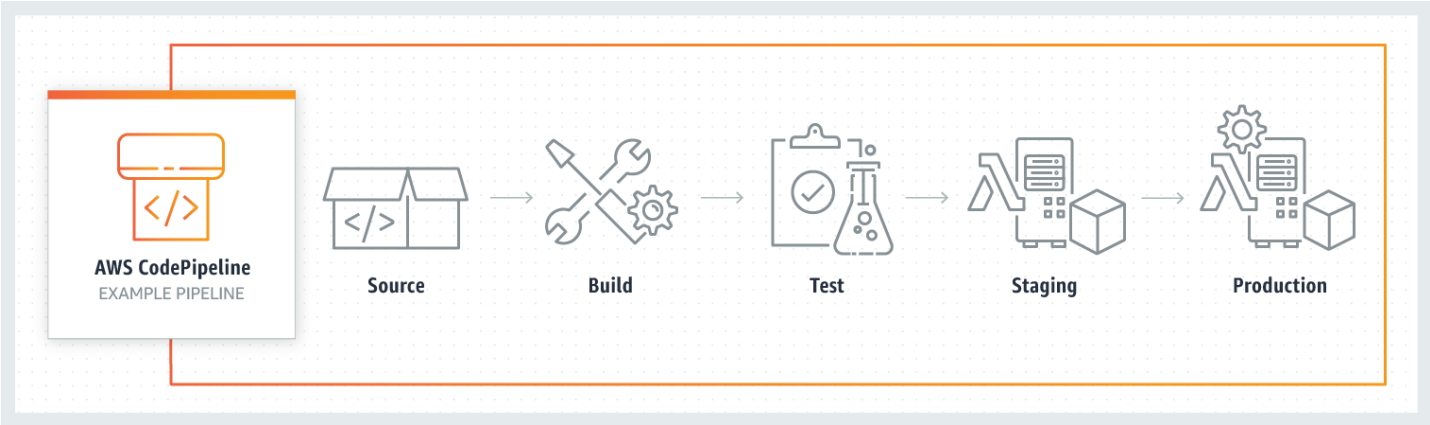
SageMaker Pipelines NEW

Workflow orchestration and automation

SageMaker Studio

Integrated development environment (IDE) for ML

AWS PIPELINES :



ESTIMATED COSTS:

pricing calculator

Feedback

English

Contact Sales

AWS Pricing Calculator > My Estimate

My Estimate [Edit](#)

Add service

Add support

Add group

Clear estimate

Export estimate

Share

Estimate summary [Info](#)

Upfront cost	Monthly cost	Total 12 months cost
0.00 USD	252.30 USD	3,027.56 USD

Getting Started with AWS

Contact Us

Sign in to the Console

Services (4)

Amazon API Gateway

Region: US East (Ohio)

Edit

Action

HTTP API requests units (millions), Average size of each request (34 KB), REST API request units (millions), Cache memory size (GB) (None), WebSocket message units (thousands), Average message size (32 KB), Requests (4 per month), Requests (12 per month)

Monthly: 46.00 USD

AWS Data Pipeline

Region: Asia Pacific (Tokyo)

Edit


Action

Number of high frequency activities in a month (3), Number of low frequency activities in a month (3), Number of inactive pipelines per month (1)	Monthly:	5.52 USD
--	----------	----------

<div> <div>Amazon Simple Storage Service (S3)</div> <div> <div>Region: US East (Ohio)</div> <div> <div>Edit</div> <div>Action ▼</div> </div> </div> </div>		
S3 Standard storage (1 GB per month)	Monthly:	0.03 USD
DT Inbound: Internet (1 TB per month), DT Outbound: US East (Verizon) - Miami (1 TB per month)	Monthly:	10.24 USD

<div> <div>Amazon SageMaker</div> <div> <div>Region: US East (Ohio)</div> <div> <div>Edit</div> <div>Action ▼</div> </div> </div> </div>		
Instance name (ml.c5.12xlarge), Number of Studio Notebook instances per data scientist (1), Studio Notebook hour(s) per day (4), Studio Notebook day(s) per month (10), Number of data scientist(s) (1)	Monthly:	97.92 USD
Storage (General Purpose SSD (gp2)), Instance name (ml.c4.2xlarge), Number of processing jobs per month (2), Number of instances per job (1), Hour(s) per instance per job (3)	Monthly:	2.97 USD
Storage (General Purpose SSD (gp2)), Instance name (ml.m5.12xlarge), Instance name (ml.m5.4xlarge), Number of data scientist(s) (1), Data Wrangler hour(s) per day (3), Data Wrangler day(s) per month (10),	Monthly:	44.38 USD
Instance name (ml.c5.12xlarge), Number of Studio Notebook instances per data scientist (1), Studio Notebook hour(s) per day (4), Studio Notebook day(s) per month (10), Number of data scientist(s) (1)	Monthly:	97.92 USD
Storage (General Purpose SSD (gp2)), Instance name (ml.c4.2xlarge), Number of processing jobs per month (2), Number of instances per job (1), Hour(s) per instance per job (3)	Monthly:	2.97 USD
Storage (General Purpose SSD (gp2)), Instance name (ml.m5.12xlarge), Instance name (ml.m5.4xlarge), Number of data scientist(s) (1), Data Wrangler hour(s) per day (3), Data Wrangler day(s) per month (10), Number of data wrangler jobs per month (1), Number of instances per job (2), Hour(s) per instance per job (3)	Monthly:	44.38 USD
Storage (General Purpose SSD (gp2)), Instance name (ml.c4.2xlarge), Number of training jobs per month (2), Number of instances per job (1), Hour(s) per instance per job (3)	Monthly:	3.01 USD
Storage (General Purpose SSD (gp2)), Instance name (ml.c4.2xlarge), Instance name (ml.c4.2xlarge), Number of models deployed (3), Number of models per endpoint (3), Number of instances per endpoint (2), Endpoint hour(s) per day (4), Endpoint day(s) per month (10), Number of Model Monitor jobs per month (1), Number of Model Monitor instances per job (2), Hour(s) per Model Monitor instance per job (4), Data Processed IN (1 GB), Data Processed OUT (1 GB)	Monthly:	42.23 USD

Acknowledgement

AWS Pricing Calculator provides only an estimate of your AWS fees and doesn't include any taxes that might apply. Your actual fees depend on a variety of factors, including your actual usage of AWS services. [Learn more](#) .

IMPLEMENTATION SCREEN SHOTS:

Amazon SageMaker

Amazon SageMaker Studio

Dashboard

Search

Images

▶ Ground Truth

▶ Notebook

▶ Processing

▶ Training

▶ Inference

▶ Augmented AI

▶ AWS Marketplace

Amazon SageMaker > SageMaker Studio

SageMaker Studio

Get started

Learn more about getting started with SageMaker Studio

Quick start

Let Amazon SageMaker handle configuring account and setting the permissions that you or a team in your organization need to use SageMaker Studio. Choosing this options uses standard encryption, which you can't change. If you need more control over configuration, choose Standard setup.

User name

default-1630563256235

The user name can have up to 63 characters. Valid characters: A-Z, a-z, 0-9, and - (hyphen)

Execution role

SageMaker Studio requires permissions to access other AWS services, such as Amazon SageMaker and Amazon S3. The execution role must have the [AmazonSageMakerFullAccess policy](#) attached. If you don't have a role with this policy attached, we can create one for you.

Choose an IAM role

SageMaker Projects and JumpStart

Enable access and provisioning of AWS Service Catalog Portfolio of products in Amazon SageMaker Studio for Amazon SageMaker Projects and JumpStart.

Enable Amazon SageMaker project templates and JumpStart for this account and Studio users

If enabled, the administrator can view the Amazon SageMaker provided project templates and JumpStart solutions published in AWS Service Catalog and users who are configured to use the domain execution are allowed to create projects using those templates and solutions with JumpStart. A launch constraint role and a project use role are automatically generated in IAM for your account.

Amazon SageMaker Studio

File Edit View Run Kernel Git Tabs Settings Help

Name Last Modified

Launcher

Get started

Explore one-click solutions, models, and tutorials

Build models automatically

Run open-source models

ML tasks and components

Notebooks and compute resources

30°C AQI 82

12:13

Jupyter Notebook :

```
In [1]: !pip install -Uq sagemaker boto3

In [2]: %store -r
%store

Stored variables and their in-db values:

In [3]: import sagemaker
import sagemaker.amazon.common as smac
import boto3

import io
import json
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from scipy.sparse import csr_matrix, hstack, save_npz
from sklearn.preprocessing import OneHotEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split

In [4]: assert sagemaker.__version__ >= "2.21.0"

In [5]: region = boto3.Session().region_name
boto3.setup_default_session(region_name=region)
boto_session = boto3.Session(region_name=region)

s3_client = boto3.client("s3", region_name=region)

sagemaker_boto_client = boto_session.client("sagemaker")
sagemaker_session = sagemaker.session.Session(
    boto_session=boto_session, sagemaker_client=sagemaker_boto_client
)
sagemaker_role = sagemaker.get_execution_role()

bucket = sagemaker_session.default_bucket()
print(f"using bucket{bucket} in region {region} \n")
```

Read the data

```
In [6]: df = pd.read_csv("data/Online Retail.csv")
print(df.shape)
df.head()
```

(541909, 8)

Out[6]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

Data Preprocessing :

Data Preprocessing

First, we check for any null (i.e. missing) values.

```
In [7]: df.isna().sum()
```

```
Out[7]: InvoiceNo      0
        StockCode     0
        Description  1454
        Quantity     0
        InvoiceDate    0
        UnitPrice     0
        CustomerID   135080
        Country       0
        dtype: int64
```

Drop any records with a missing CustomerID. If we do not know who the customer is, then it is not helpful to us when we make recommendations.

```
In [8]: df.dropna(subset=["CustomerID"], inplace=True)
        df["Description"] = df["Description"].apply(lambda x: x.strip())
        print(df.shape)

(406829, 8)
```

```
In [9]: plt.figure(figsize=(10, 5))
        sns.distplot(df["Quantity"], kde=True)
        plt.title("Distribution of Quantity")
        plt.xlabel("Quantity");
```

```
In [12]: df = df.groupby(["StockCode", "Description", "CustomerID", "Country", "UnitPrice"])[
        "Quantity"
        ].sum()
        df = df.loc[df > 0].reset_index()
        df.shape
```

```
Out[12]: (274399, 6)
```

```
In [13]: def loadDataset(dataframe):
        enc = OneHotEncoder(handle_unknown="ignore")
        onehot_cols = ["StockCode", "CustomerID", "Country"]
        ohe_output = enc.fit_transform(dataframe[onehot_cols])

        vectorizer = TfidfVectorizer(min_df=2)
        unique_descriptions = dataframe["Description"].unique()
        vectorizer.fit(unique_descriptions)
        tfidf_output = vectorizer.transform(dataframe["Description"])

        row = range(len(dataframe))
        col = [0] * len(dataframe)
        unit_price = csr_matrix((dataframe["UnitPrice"].values, (row, col)), dtype="float32")

        X = hstack([ohe_output, tfidf_output, unit_price], format="csr", dtype="float32")

        y = dataframe["Quantity"].values.astype("float32")

        return X, y
```

```
In [14]: X, y = loadDataset(df)
```

```
In [15]: # display sparsity
        total_cells = X.shape[0] * X.shape[1]
        (total_cells - X.nnz) / total_cells
```

```
Out[15]: 0.9991284988048746
```

Our data is over 99.9% sparse. Because of this high sparsity, the sparse matrix data type allows us to represent our data using only a small fraction of the memory that a dense matrix would require.

Preparing for Modeling :

Prepare Data For Modeling

- Split the data into training and testing sets
- Write the data to protobuf recordIO format for Pipe mode. [Read more](#) about protobuf recordIO format.

```
In [16]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
        X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
Out[16]: ((219519, 9284), (54880, 9284), (219519,), (54880,))
```

Save numpy arrays to local storage in /data folder

```
In [17]: df.to_csv("data/online_retail_preprocessed.csv", index=False)
        save_npz("data/X_train.npz", X_train)
        save_npz("data/X_test.npz", X_test)
        np.savez("data/y_train.npz", y_train)
        np.savez("data/y_test.npz", y_test)
```

```
In [18]: prefix = "personalization"

        train_key = "train.protobuf"
        train_prefix = f"{prefix}/train"

        test_key = "test.protobuf"
        test_prefix = f"{prefix}/test"

        output_prefix = f"s3://{bucket}/{prefix}/output"
```

```
In [ ]: def writeDatasetToProtobuf(X, y, bucket, prefix, key):
        buf = io.BytesIO()
        smac.write_spmatrix_to_sparse_tensor(buf, X, y)
        buf.seek(0)
        obj = "{}/{}/".format(prefix, key)
        boto3.resource("s3").Bucket(bucket).Object(obj).upload_fileobj(buf)
        return "s3://{}/{}".format(bucket, obj)

        train_data_location = writeDatasetToProtobuf(X_train, y_train, bucket, train_prefix, train_key)
        test_data_location = writeDatasetToProtobuf(X_test, y_test, bucket, test_prefix, test_key)
```

```
In [9]: container = sagemaker.image_uris.retrieve("factorization-machines", region=boto_session.region_name)

        fm = sagemaker.estimator.Estimator(
            container,
            sagemaker_role,
            instance_count=1,
            instance_type="ml.c5.xlarge",
            output_path=output_prefix,
            sagemaker_session=sagemaker_session,
        )

        fm.set_hyperparameters(
            feature_dim=input_dims,
            predictor_type="regressor",
            mini_batch_size=1000,
            num_factors=64,
            epochs=20,
        )
```

```
In [ ]: if 'training_job_name' not in locals():

        fm.fit({'train': train_data_location, 'test': test_data_location})
        training_job_name = fm.latest_training_job.job_name
        %store training_job_name

        else:
            print(f'Using previous training job: {training_job_name}')
```

```
In [11]: training_job_info = sagemaker_boto_client.describe_training_job(TrainingJobName=training_job_name)
```

Training data artifact

```
In [12]: training_data_s3_uri = training_job_info["InputDataConfig"][0]["DataSource"]["S3DataSource"][
        "S3Uri"]
        ]

        matching_artifacts = list(
            artifact.Artifact.list(source_uri=training_data_s3_uri, sagemaker_session=sagemaker_session)
        )

        if matching_artifacts:
            training_data_artifact = matching_artifacts[0]
            print(f"Using existing artifact: {training_data_artifact.artifact_arn}")
        else:
            training_data_artifact = artifact.Artifact.create(
                artifact_name="TrainingData",
                source_uri=training_data_s3_uri,
                artifact_type="Dataset",
                sagemaker_session=sagemaker_session,
            )
            print(f"Create artifact {training_data_artifact.artifact_arn}: SUCCESSFUL")
```

Using existing artifact: arn:aws:sagemaker:us-east-2:645431112437:artifact/cdd7fbecb4eefa22c43b2ad48140acc2

Final Output of predicted products :

```
In [32]: print("Top 5 recommended products:")
        get_recommendations(df, top_customer, n_recommendations=5, n_ranks=100)
```

Top 5 recommended products:

```
Out[32]: [['22423', 'REGENCY CAKESTAND 3 TIER'],
          ['22776', 'SWEETHEART CAKESTAND 3 TIER'],
          ['22624', 'IVORY KITCHEN SCALES'],
          ['85123A', 'WHITE HANGING HEART T-LIGHT HOLDER'],
          ['85099B', 'JUMBO BAG RED RETROSPOT']]
```

```
In [ ]:
```

Thank You