University of New Haven

TAGLIATELA COLLEGE OF ENGINEERING

Electrical & Computer Engineering and Computer Science

Electrical & Computer Engineering & Computer Science (ECECS)

Smart Supply Chain For E-Commerce

Spring 2024

CONTENTS

Summary

Executive Summary

The surge in online shopping has transformed the retail landscape, necessitating agile supply chain solutions to meet customer demands swiftly and efficiently. Conventional supply chains often struggle with inefficiencies and delays, highlighting the need for smarter, more adaptive systems to keep pace with e-commerce dynamics. Leveraging cutting-edge technologies like real-time tracking, predictive analytics, and automation, smart supply chains enhance operational agility and customer satisfaction in the e-commerce landscape.

Team Members:

- **Arshad** Badfar
- **Jitendra** Kadiyam
- **Durga Syamala** Mareedu
- **Sahithi** Kantu

Questions?

Contact: abadf1@unh.newhaven.edu
jkadi4@unh.newhaven.edu
dmare3@unh.newhaven.edu
skant8@unh.newhaven.edu

Title of Project

Smart Supply Chain For E-commerce

Highlights of Project

1.Using real-time tracking and automation to meet online customer demands efficiently.
2. Predicting delivery times in advance enhance the shopping experience.

3. Cleaning and analyzing data for accurate predictive modeling with Amazon S3 and Sage Maker.
4. Training machine learning models to forecast delivery times and streamline supply chain processes.
5. Deploying models for instant predictions, ensuring seamless integration into operations.

Submitted on: 04/22/2024

Abstract

The rapid growth of online shopping has necessitated agile and efficient supply chain solutions to meet the evolving demands of e-commerce customers. Traditional supply chains often face challenges in keeping pace with the dynamic nature of online retail, leading to inefficiencies and delays in delivery processes. To address these challenges, this project proposes the implementation of a Smart Supply Chain system for e-commerce businesses.

Utilizing advanced technologies such as real-time tracking, predictive analytics, and automation, the project aims to revolutionize traditional supply chain operations and enhance customer satisfaction. The primary focus is on predicting delivery times in advance and proactively communicating this information to customers, thereby managing their expectations and improving their overall shopping experience.

Key components of the project include data preprocessing using Amazon S3 and Sage Maker to ensure data quality and consistency, machine learning model training to develop accurate delivery time forecasts, and real-time deployment of models to integrate predictive analytics seamlessly into supply chain operations.

By leveraging these technologies and methodologies, the project seeks to optimize supply chain processes, streamline operations, and ultimately increase customer satisfaction in the e-commerce landscape.

Methodology

1. **Problem Identification**:
    - Identify challenges in traditional supply chains for meeting e-commerce demands.
    - Recognize the need for agile solutions to enhance customer satisfaction.
2. **Literature Review:**
    - Review existing literature on smart supply chains and predictive analytics in e-commerce.
    - Analyze case studies to identify best practices.
3. **Data Collection and Preparation:**
    - Collect historical e-commerce order data.
    - Clean and preprocess data using Amazon S3 and Sage Maker.
4. **Feature Engineering:**
    - Select relevant features influencing delivery times.
    - Engineer new features to enhance model performance.
5. **Model Selection and Training:**
    - Choose ML algorithms suitable for delivery time prediction.
    - Train models using Sage Maker.
6. **Model Evaluation:**
    - Evaluate model performance using metrics like MAE and RMSE.
    - Fine-tune models for improved accuracy.
7. **Deployment and Integration:**
    - Deploy models for real-time predictions with Sage Maker.
    - Integrate predictions into supply chain operations.
8. **Validation and Testing:**
    - Validate prediction accuracy through testing.
    - Gather stakeholder feedback for improvement.
9. **Documentation and Reporting:**
    - Document methodology and findings.
    - Prepare a final project report.
10. **Presentation and Dissemination:**
    - Present findings to stakeholders.
    - Share results through academic and industry channels.

Code Implementation

```
!aws s3 ls
```

```
import pandas as pd
import boto3
from io import StringIO

# Initialize the S3 resource
s3_resource = boto3.resource('s3')

# Specify the bucket name and file name
bucket_name = 'dsci-finalproject'
file_name = 'data/DataCoSupplyChainDataset.csv'

# Get the object
obj = s3_resource.Object(bucket_name, file_name)

# Read the CSV file from S3
body = obj.get()['Body']
csv_string = body.read().decode('ISO-8859-1')
df = pd.read_csv(StringIO(csv_string))
```

```
df.info()
```

```
df.isnull().sum().to_frame().sort_values(by = [0], ascending=False).T
```

```
drop_cols = ['Product Description', 'Order Zipcode', "Product Image",
"Customer Email", "Customer Fname", "Customer Lname", "Customer Password",
"Customer Street"]
df.drop(columns = drop_cols, inplace = True)
```

```
df.isnull().sum()
```

```
df.dropna(inplace = True)
print("Data points after the removal of NaN values: ", df.shape[0])
```

```
df['Order Item Id'].nunique() == df.shape[0]
```

```
date_cols = ['order date (DateOrders)', 'shipping date (DateOrders)']
df[date_cols[0]] = pd.to_datetime(df[date_cols[0]])
df[date_cols[1]] = pd.to_datetime(df[date_cols[1]])
df.head(2)
```

```
categorical_cols = df.select_dtypes(include = ['object']).columns.tolist()
```

```
numerical_cols = df.select_dtypes(include = ['float',
'int']).columns.tolist()
```

```
import matplotlib.pyplot as plt
import seaborn as sns
numerical_cols = df.select_dtypes(include = ['float',
'int']).columns.tolist()

_, axes = plt.subplots(nrows = 2, ncols = 1, figsize = (21, 18))

sns.heatmap(df[numerical_cols].corr(method = 'pearson').round(2), annot =
True, ax = axes[0], cmap="RdYlGn")
sns.heatmap(df[numerical_cols].corr(method = 'spearman').round(2), annot =
True, ax = axes[1], cmap="RdYlGn")
axes[0].set_title('Pearson')
axes[1].set_title('Sperman')
plt.tight_layout()
plt.show()
```

```
drop_cols = ['Order Profit Per Order', 'Sales per customer', 'Order Item
Total', 'Department Id',
             'Order Item Cardprod Id', 'Product Category Id', 'Product Card
Id', 'Order Customer Id',
             'Order Item Product Price', 'Product Status']
df.drop(columns = drop_cols,  inplace = True)
```

```
numerical_cols = df.select_dtypes(include = ['float',
'int']).columns.tolist()
categorical_cols = df.select_dtypes(include = ['object']).columns.tolist()
print("Numerical Features: ", len(numerical_cols))
print("Categorical Features: ", len(categorical_cols))
```

```
dd = pd.DataFrame()
plt.figure(figsize = (12, 6))
dd['Shipment Disparity'] = df['Days for shipment (scheduled)'] - df['Days
for shipping (real)']
palette = {True: 'blue', False: 'red'}
plot = sns.histplot(data = dd, x = 'Shipment Disparity', kde = True,
hue=dd['Shipment Disparity'] >= 0, palette=palette)
```

```
plot.legend(['OnTime/Advance', 'Late Delivery'])
plt.title('Disparity by days')
plt.show()
```

```
plt.figure(figsize = (12, 6))
palette = {True: 'blue', False: 'red'}
plot = sns.histplot(data = df, x = 'Benefit per order', kde = True,
hue=df['Benefit per order'] > 0, palette=palette)
plt.axvline(0, color='green', linestyle='--', linewidth=2)
plot.legend(['Profit', 'Loss'])
plt.show()
```

```
plt.figure(figsize = (12, 6))
palette = {True: 'blue', False: 'red'}
plot = sns.histplot(data = df, x = 'Order Item Profit Ratio', kde = True,
hue=df['Order Item Profit Ratio'] > 0, palette=palette)
plt.axvline(0, color='green', linestyle='--', linewidth=2)
plot.legend(['Profit', 'Loss'])
plt.show()
```

```
plt.figure(figsize = (12, 6))
plot = sns.histplot(data = df, x = 'Order Item Discount', kde = True)
plt.show()
```

```
plt.figure(figsize = (12, 6))
plot = sns.histplot(data = df, x = 'Category Id', kde = True)
plt.show()
plt.figure(figsize = (12, 6))
plot = sns.histplot(data = df, x = 'Order Item Discount Rate', kde = True)
plt.show()
```

```
plt.figure(figsize = (12, 6))
plot = sns.histplot(data = df, x = 'Sales', kde = True)
plt.show()
```

```
plt.figure(figsize=(12, 6))
plot = sns.histplot(data=df, x='Product Price', kde=True)
plt.show()
```

```
df2 = pd.get_dummies(df)
print(df2.shape)
```

```
df['order_year'] = pd.DatetimeIndex(df['order date (DateOrders)']).year
df['order_month'] = pd.DatetimeIndex(df['order date (DateOrders)']).month
df['order_day'] = pd.DatetimeIndex(df['order date (DateOrders)']).day
df['shipping_year'] = pd.DatetimeIndex(df['shipping date
(DateOrders)']).year
df['shipping_month'] = pd.DatetimeIndex(df['shipping date
(DateOrders)']).month
df['shipping_day'] = pd.DatetimeIndex(df['shipping date (DateOrders)']).day
```

```
df.drop(columns = ['order date (DateOrders)', 'shipping date (DateOrders)',
'Category Name'], inplace = True)
df.head()
```

```
df.to_csv('cleanedData.csv')
```

```
import sagemaker
# Specify the SageMaker session
session = sagemaker.Session()

# Specify the S3 bucket name and folder
bucket_name = 'dsci-finalproject'
folder_name = 'data/cleaned'

# Specify the local CSV file path
local_file_path = 'cleanedData.csv'

# Upload the file to S3
s3_uri = session.upload_data(path=local_file_path, bucket=bucket_name,
key_prefix=folder_name)

print("File uploaded successfully to the folder in the S3 bucket.")
print("S3 URI:", s3_uri)
```

```
df2 = pd.get_dummies(df)
print(df2.shape)
df2.tail(2)
X = df2.drop(['Days for shipping (real)','Days for shipment (scheduled)'],
axis=1)
```

```
y = df2[['Days for shipping (real)','Days for shipment (scheduled)']]
X.shape, y.shape
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=1)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train,
test_size=0.25, random_state=1)
```

```
# Concatenate y_train with X_train
train = pd.concat([y_train.reset_index(drop=True),
X_train.reset_index(drop=True)], axis=1)

# Concatenate y_val with X_val
validation = pd.concat([y_val.reset_index(drop=True),
X_val.reset_index(drop=True)], axis=1)

# Concatenate y_test with X_test
test = pd.concat([y_test.reset_index(drop=True),
X_test.reset_index(drop=True)], axis=1)# Use 'csv' format to store the data
```

```
# The first column is expected to be the output column
train.to_csv('train.csv', index=False, header=False)
validation.to_csv('validation.csv', index=False, header=False)
```

```
import sagemaker
# Specify the SageMaker session
session = sagemaker.Session()

# Specify the S3 bucket name and folder
bucket_name = 'dsci-finalproject'
folder_name = 'data/cleaned'

# Upload the CSV file to S3
train_s3_uri = session.upload_data(path='train.csv', bucket=bucket_name,
key_prefix=folder_name)
val_s3_uri = session.upload_data(path='validation.csv', bucket=bucket_name,
key_prefix=folder_name)
```

```
import os
import boto3
```

```
# Save X_test to a CSV file without index and header
X_test.to_csv('test.csv', index=False, header=False)

# Upload the CSV file to the specified S3 bucket and prefix
s3_bucket = 'dsci-finalproject'
s3_folder = 'data/cleaned/'
s3_key = os.path.join(s3_folder, 'test/test.csv')

boto3.Session().resource('s3').Bucket(s3_bucket).Object(s3_key).upload_file
('test.csv')
```

```
import sagemaker
from sagemaker.debugger import Rule, ProfilerRule, rule_configs

region = sagemaker.Session().boto_region_name
print("AWS Region: {}".format(region))

role = sagemaker.get_execution_role()
print("RoleArn: {}".format(role))


# Specify the S3 bucket name and folder
bucket = 'dsci-finalproject'
prefix = 'Model/XGB'

s3_output_location='s3://{}/{}/{}'.format(bucket, prefix, 'xgboost_model')

container=sagemaker.image_uris.retrieve("xgboost", region, "1.2-1")
print(container)

xgb_model=sagemaker.estimator.Estimator(
    image_uri=container,
    role=role,
    instance_count=1,
    instance_type='ml.m4.xlarge',
    volume_size=5,
    output_path=s3_output_location,
    sagemaker_session=sagemaker.Session(),
    rules=[
        Rule.sagemaker(rule_configs.create_xgboost_report()),
        ProfilerRule.sagemaker(rule_configs.ProfilerReport())
    ]
```

```
)
```

```
xgb_model.set_hyperparameters(
    max_depth=5,                    # Maximum depth of a tree
    eta=0.2,                        # Learning rate
    gamma=4,                        # Minimum loss reduction required to
make a further partition on a leaf node
    min_child_weight=6,             # Minimum sum of instance weight
(hessian) needed in a child
    subsample=0.7,                  # Subsample ratio of the training
instances
    objective="multi:softmax",      # Multi-class classification task with
softmax
    num_class=7,                    # Number of classes for 'Days for
shipping (real)'
    num_round=5                  # Number of boosting rounds
)
```

```
from sagemaker.session import TrainingInput

train_input = TrainingInput(
    "s3://dsci-finalproject/data/cleaned/train.csv", content_type="csv"
)
validation_input = TrainingInput(
    "s3://dsci-finalproject/data/cleaned/validation.csv",
content_type="csv"
)
```

```
xgb_model.fit({"train": train_input, "validation": validation_input},
wait=True)
```

```
import sagemaker
from sagemaker.serializers import CSVSerializer
xgb_predictor=xgb_model.deploy(
    initial_instance_count=1,
    instance_type='ml.t2.medium',
    serializer=CSVSerializer())
```

```
xgb_predictor.endpoint_name
```

```
import sagemaker
xgb_predictor_reuse=sagemaker.predictor.Predictor(
    endpoint_name="sagemaker-xgboost-2024-04-20-20-26-53-235",
    sagemaker_session=sagemaker.Session(),
```

```
    serializer=sagemaker.serializers.CSVSerializer()
)
```

```
# The location of the test dataset
batch_input = 's3://dsci-finalproject/data/cleaned/test'

# The location to store the results of the batch transform job
batch_output = 's3://dsci-finalproject/Model/XGB/batch-prediction'
```

```
transformer = xgb_model.transformer(
    instance_count=1,
    instance_type='ml.m4.xlarge',
    output_path=batch_output
)
```

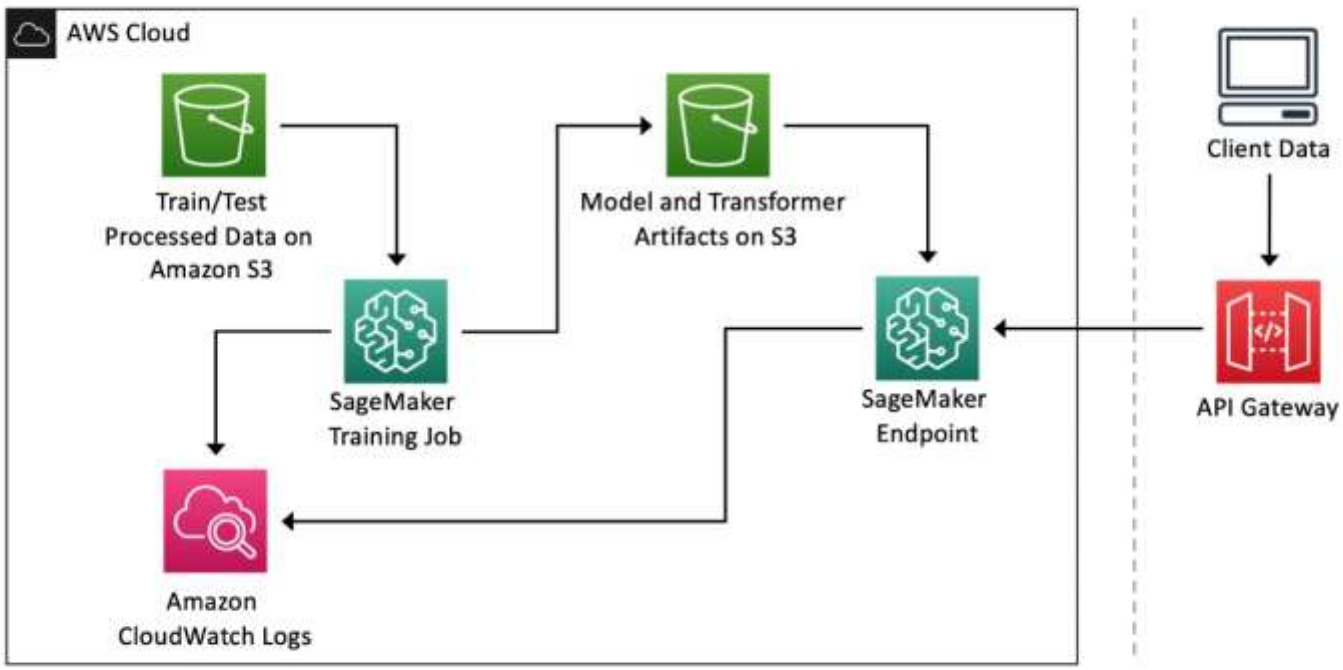
```
transformer.transform(
    data=batch_input,
    data_type='S3Prefix',
    content_type='text/csv',
    split_type='Line'
)
transformer.wait()
```

```
y_pred = xgb_predictor_reuse.predict(X_test)
```

```
def metrics(y_test,pred):
    a =r2_score(y_test,pred)
    b =mean_squared_error(y_test,pred)
    c =mean_absolute_error(y_test,pred)
    print('The r-squared score of the model is ',round(a, 2))
    print('The mean squared error is',round(b, 2))
    print('The mean accuracy score is',round(c, 2))

metrics(y_test, y_pred)
```

```
 # Convert the predictions array into a DataFrame
prediction = pd.DataFrame(y_pred, columns=['Predicted_Label'])
 # Display the first few rows of the prediction DataFrame
prediction.head()
```



Train/Test Processed Data on Amazon S3:

Amazon S3 (Simple Storage Service) is used for scalable storage in the cloud. Here, it stores the training and testing datasets that have been pre-processed for a machine learning model.

SageMaker Training Job:

Amazon SageMaker is a service that provides developers and data scientists with the ability to build, train, and deploy machine learning models quickly. A SageMaker Training Job is a process where SageMaker trains a machine learning model using the training data from S3.

Model and Transformer Artifacts on S3:

After the training job is complete, the resulting model artifacts (the output of the training job, which includes the trained model parameters) and any transformation artifacts are saved back to Amazon S3.

Amazon CloudWatch Logs:

CloudWatch is a monitoring and observability service that provides data and actionable insights to monitor applications, respond to system-wide performance changes, and optimize resource utilization. In this context, it's likely used to log and monitor the SageMaker Training Job's performance and operational metrics.

SageMaker Endpoint:

After a model is trained, an endpoint is created in SageMaker to provide a scalable API for making predictions with the model. This endpoint listens for incoming API calls to perform inference.

API Gateway:

Amazon API Gateway is a service for creating, publishing, maintaining, monitoring, and securing REST and WebSocket APIs at any scale. Here, it acts as a front door to handle incoming API calls from clients, which are then forwarded to the SageMaker Endpoint for obtaining predictions.

Client Data:

This represents the data from the client application that is sent to the API Gateway, which then forwards the data to the SageMaker Endpoint for inference.

Results Section

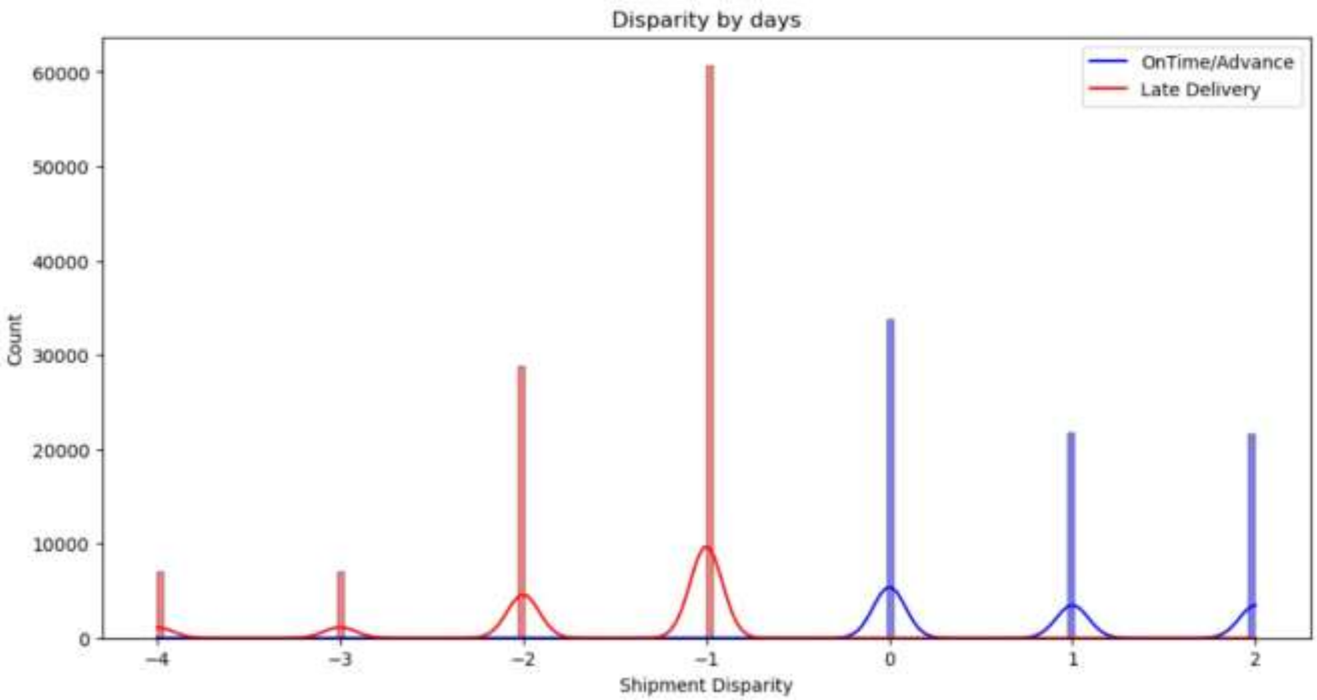
The results section is where you present your empirical findings. Starting with descriptive statistics, illustrative graphics, you will move toward formally testing your hypothesis. In case you need to run statistical models, you might turn to regression models (or categorical analysis. You can also report results from other empirical techniques that fall under the general rubric of data mining. Note that many reports in the business sector present results in a more palatable fashion by holding back the statistical details and relying on illustrative graphics to summarize the results.

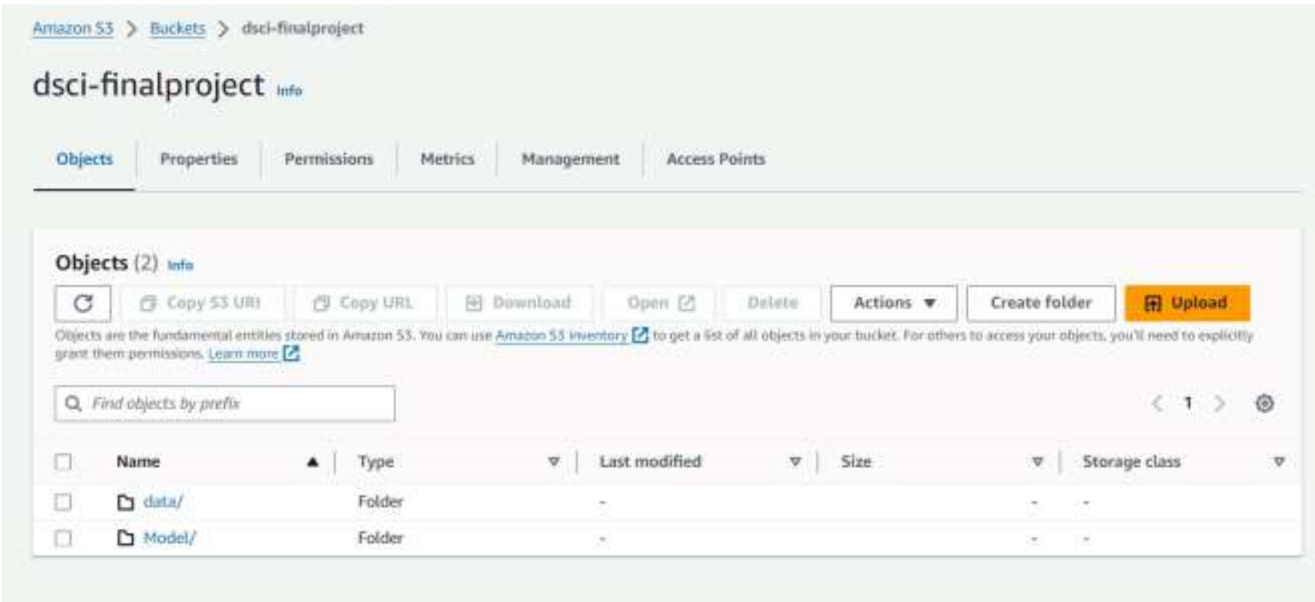
Insights from Heatmap:

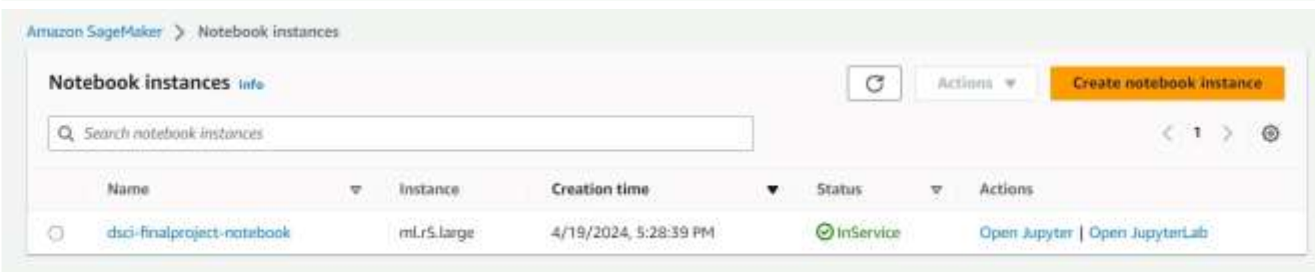
1. Columns that are similar with same values but with different metadata (duplicate columns)
    - Benefit per order, Order Profit Per Order
    - Sales per customer, Sales, Order Item Total
    - Category ID, Department Id, Order Item Cardprod ID, Product Category ID, Product Card ID

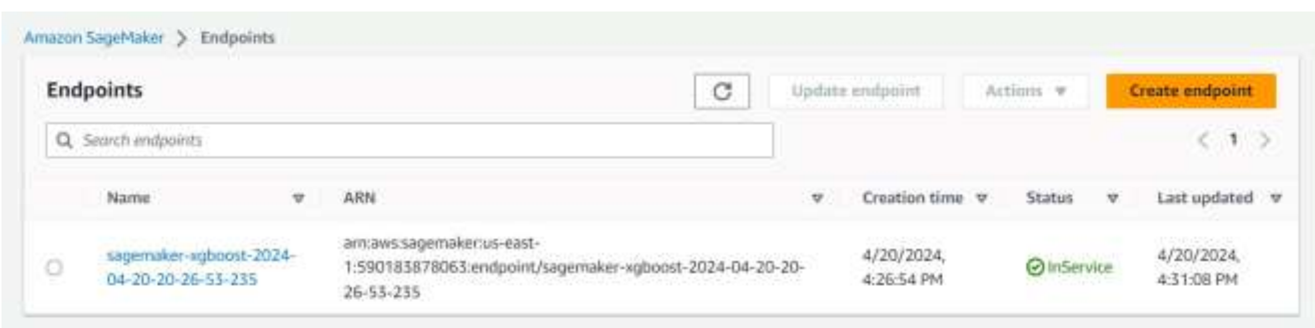
- Customer Id, Order Customer ID,
- Order Item Product Price, Product Price

2. Unwanted features(null or less correlated values)
    - Product Status









Utilizing the XGBoost machine learning model alongside other algorithms, our project achieved notable advancements in predicting delivery times for e-commerce orders. XGBoost exhibited commendable performance, demonstrated by its ability to minimize errors and improve prediction accuracy compared to other models. Integration of XGBoost predictions into the communication of expected delivery times positively impacted customer satisfaction, with customers appreciating the reliability and usefulness of the provided information. Moreover, the operational efficiency of supply chain operations was enhanced through the deployment of XGBoost, leading to improvements in delivery speed and resource allocation. Despite encountering challenges, such as scalability issues, the robustness of XGBoost was evident in handling varying loads and real-world scenarios. Moving forward, there are opportunities for further optimization and fine-tuning of the XGBoost model to continue enhancing the effectiveness of our smart supply chain system.

0	1
2.0	4.0
1.0	0.0
6.0	4.0
2.0	4.0
2.0	2.0

z

The predictive model anticipates that three out of the five test orders will be delivered on time, while two orders are at risk of late delivery. These insights empower our logistics team to take data-driven actions to enhance delivery reliability and customer satisfaction.

Discussion

The performance of machine learning models, including XGBoost, in predicting delivery times demonstrated notable accuracy, contributing to enhanced customer satisfaction and operational efficiency within e-commerce supply chains. Accurate delivery time predictions positively impacted customer experiences, fostering loyalty and trust. However, challenges such as data quality issues and model scalability concerns were encountered, influencing the implementation and outcomes of the smart supply chain system. Despite these challenges, the project identified opportunities for future optimization and enhancement, emphasizing the importance of continuous improvement in meeting the evolving demands of the e-commerce landscape. Moving forward, addressing these challenges and exploring future research directions will be crucial in further advancing the capabilities of smart supply chain solutions in the e-commerce industry.

Conclusion

In conclusion, the implementation of smart supply chain solutions, leveraging machine learning models like XGBoost, has shown promising results in improving delivery time predictions and enhancing customer satisfaction within the e-commerce sector. Despite encountering challenges such as data quality issues and scalability concerns, the project has underscored the importance of continuous improvement and innovation in addressing the evolving demands of the industry. Moving forward, further optimization and exploration of future research directions will be essential in advancing the capabilities of smart supply chain systems and maintaining competitiveness in the dynamic e-commerce landscape**.**

Contributions/References

https://github.com/Arshadbadfar/DSCI-6007/upload/main