

# Deployment Of AI

## Group-C

### Final Report

#### Price Optimization for Retail Industry

Our Team:

Student Name	Student ID
Abhishek Pandey	500227770
Ram Krishna Dhakal	500228601
Bibek Khadka	500216129

Govind Ram Gupta Belde	500228074
Kamalpreet Kaur	500227884

## Detailed Report on AI Project for Price Optimization in the Retail Industry (Maven Toy Store)

### Project Overview

The objective of this project is to optimize pricing strategies in the retail toy industry using machine learning techniques. The project leverages the Maven Toy Store dataset to build a decision tree-based machine learning model for predicting Maximum increase in Profit a retailer can make and deployed the solution using modern DevOps tools such as Jenkins, Docker, and Git. Additionally, a user-friendly interface has been developed to visualize the model's predictions and performance.

### Dataset

- **Dataset Source:** Maven Toy Store Dataset
- **Description:**
  - Contains historical sales data of toys, including features such as toy categories, pricing, seasonal trends, customer demographics, and sales volumes.
  - Includes over 800,000 records with fields such as:
    - Toy ID
    - Product Category
    - Product name
    - Product price
    - Product cost

- Store name
- Store city
- No.of units

## Data Preprocessing

### 1. Cleaning:

- a. Removed duplicate records.
- b. Handled missing values using mean/median imputation for numerical columns and mode imputation for categorical columns.

### 2. Feature Engineering:

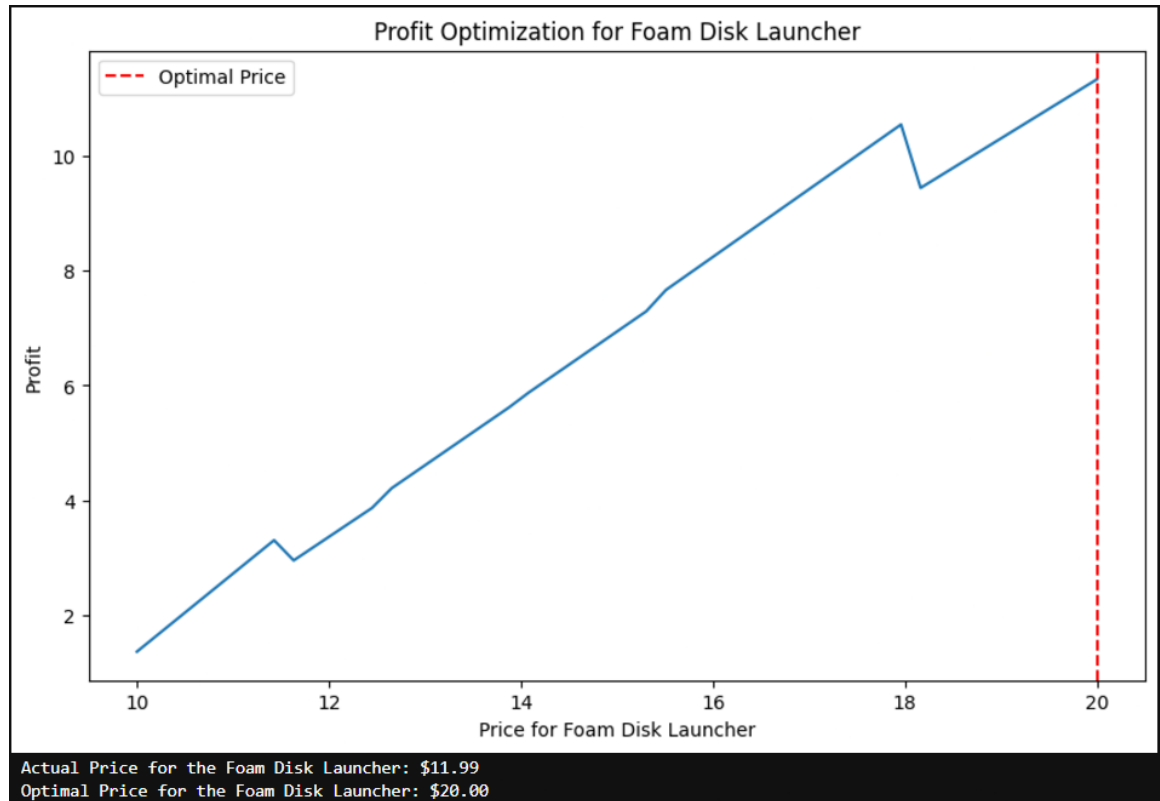
- a. Created new features such as "Product price" and "Profit"
- b. One-hot encoded categorical variables such as "Category" and "Season."
- c. Merge sales with products data on 'Product\_ID'
- d. Merge the result with stores data on 'Store\_ID'
- e. Remove dollar signs and convert 'Product\_Cost' and 'Product\_Price' to numeric
- f. Convert 'Date' to datetime and extract year, month

### 3. Splitting:

- a. Dataset split into training (80%) and testing (20%) sets.

### 4. Evaluation Metrics:

- a. Random Forest Mean Squared Error: 0.6435769340355606
- b. Random Forest R-squared: 0.05284726033696985



C.

From the above picture we can see that the model predicted an increase in 8 dollars for the above product is possible considering the market scenario.

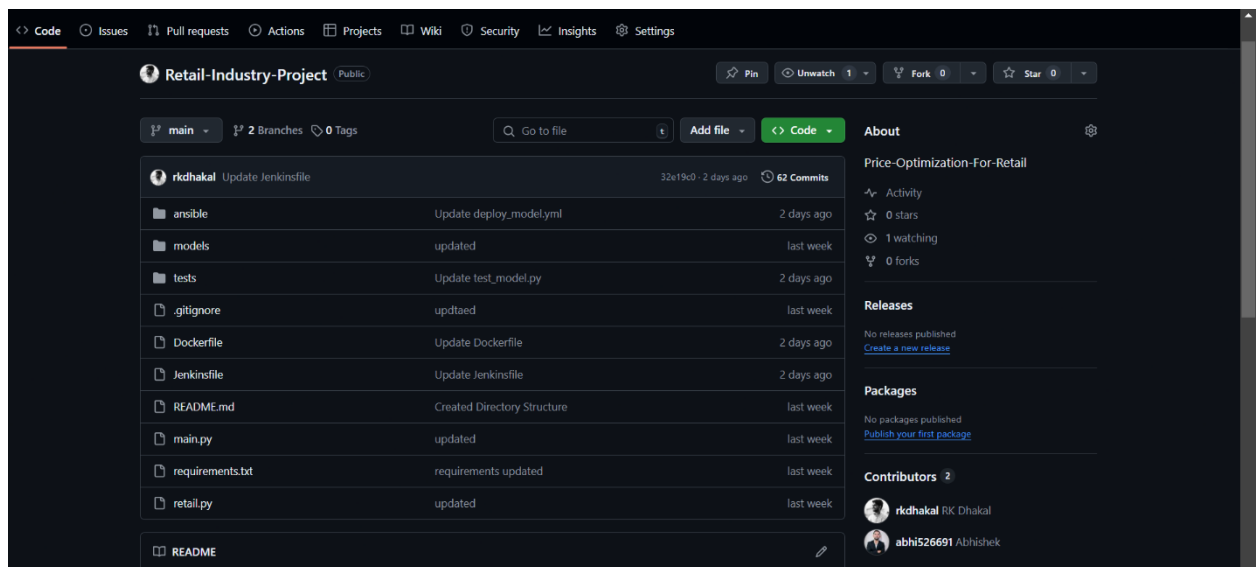
## Machine Learning Model

- **Model Type:** Decision Tree
- **Reason for Selection:**
  - Handles categorical and numerical data effectively.
  - Captures non-linear relationships between pricing and sales volume.
  - Provides interpretable results for business stakeholders.
- **Model Training:**
  - Hyperparameter tuning performed using grid search to optimize parameters
  - Created a function to calculate Profit for a given price
  - Created a Dataframe to maintain the feature names
  - Plotted the profits for the different product categories using random forest regressor.

# Deployment Pipeline

## 1. Version Control with Git:

- Repository hosted on GitHub.
- Contains structured directories for:
  - **Data:** Raw and processed data.
  - **Models:** Saved decision tree models.
  - **Scripts:** Python scripts for preprocessing, training, and prediction.
  - **UI:** Source code for the user interface.
  - Jenkins: for CI/CD pipelines
  - Docker: To automate the setup



Git link: <https://github.com/rkdhahal/Retail-Industry-Project>

## 2. Continuous Integration/Deployment with Jenkins:

- Automated pipeline setup using Jenkins to:
  - Fetch the latest code from Git.
  - Run unit tests to ensure code quality.
  - Build the Docker container containing the ML model and dependencies.
  - Deploy the container to a staging environment for validation.

← → ↻ 127.0.0.0:8080 80% ☆ 🔒 ⬇️ 🗨️ 📄 ☰

# Jenkins

Search (CTRL+K) 🔍 Group\_C 🗑️ log out

Dashboard >

+ New Item

📅 Build History

🔗 Project Relationship

🔍 Check File Fingerprint

⚙️ Manage Jenkins

📌 My Views

🌊 Open Blue Ocean

Build Queue

No builds in the queue.

Build Executor Status

0/2

Icon: S M L

All +

S	W	Name ↓	Last Success	Last Failure	Last Duration	F
✓	☀️	Data Analytics CI-CD	1 day 23 hr #59	1 day 23 hr #54	19 sec	▶️ ☆
✓	☁️	data-analytics-pipeline	2 days 5 hr #6	2 days 5 hr #4	0.88 sec	▶️ ☆
✗	☁️	priceoptimization	N/A	2 days 5 hr #2	0.11 sec	▶️ ☆
✗	☁️	Retail_Industry_Project	6 days 10 hr #2	6 days 10 hr #5	0.67 sec	▶️ ☆

Add description ✎

REST API Jenkins 2.479.2

📁 rkdhaka/Retail-Industry × 👤 Data Analytics CI-CD Config × +

← → ↻ 127.0.0.0:8080/Job/Data Analytics CI-CD/configure 80% ☆ 🔒 ⬇️ 🗨️ 📄 ☰

Dashboard > Data Analytics CI-CD > Configuration

## Configure

⚙️ General

🔑 Advanced Project Options

📄 Pipeline

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

https://github.com/rkdhaka/Retail-Industry-Project

Credentials ?

- none -

+ Add

Advanced ▾

Add Repository

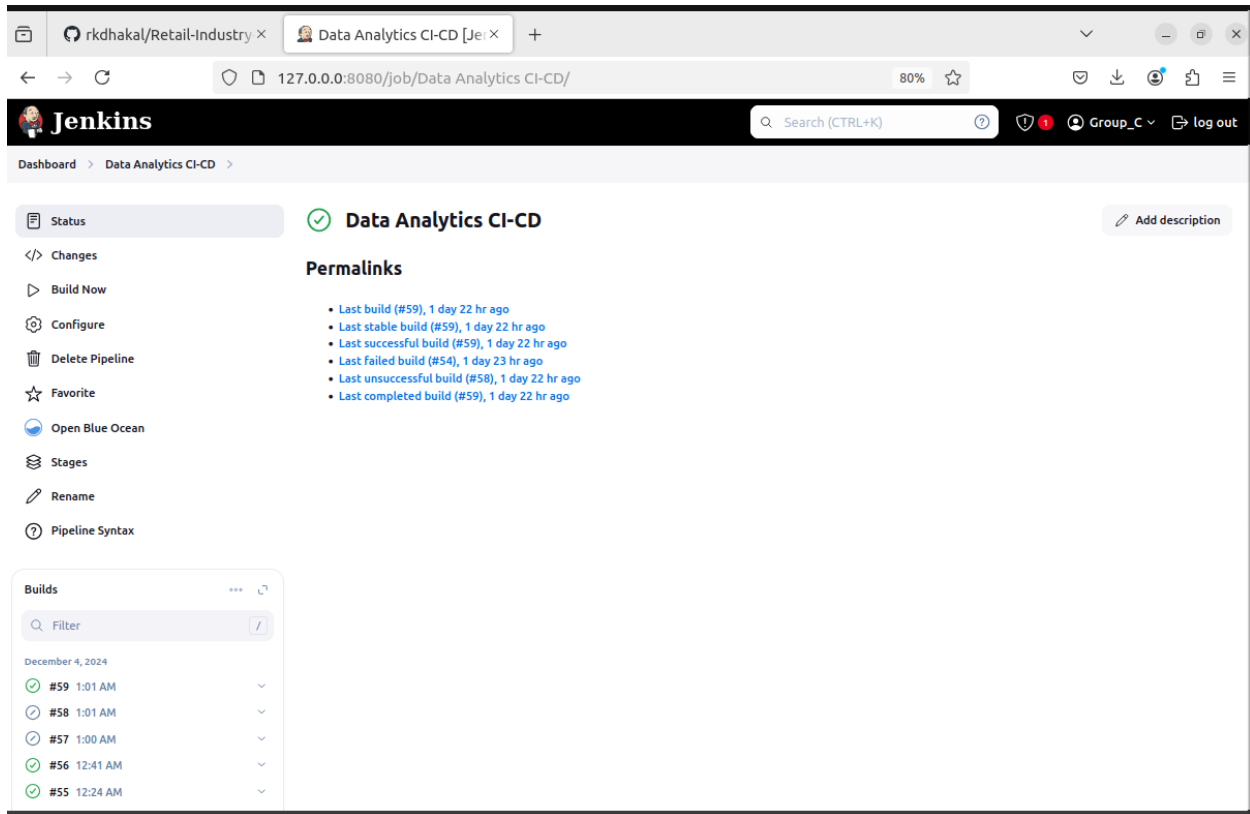
Branches to build ?

Branch Specifier (blank for 'any') ?

main

Save Apply

From the above pictures we can see the Jenkins is set up and running.



After running this is how it looks like.

### 3. Containerization with Docker:

- Created a Dockerfile to package the application.
- Image contains:
  - Preprocessed data pipeline.
  - Decision tree model.
  - Flask application for the user interface.
- Pushed the Docker image to DockerHub for easy access.

We have setup the docker.

Dockerfile > ...

```

2 WORKDIR /app
3 COPY requirements.txt .
4 RUN pip install --no-cache-dir -r requirements.txt
5 COPY . .
6 EXPOSE 8501
7 CMD ["streamlit", "run", "main.py", "--server.port=8501"]
8

```

OUTPUT

DEBUG CONSOLE

PORTS

JUPYTER

JUPYTER 8

...

^

X

TERMINAL

You can now view your Streamlit app in your browser.  
  
URL: http://0.0.0.0:8501  
  
2024-11-29 04:06:50.964  
'st.cache' is deprecated and will be removed soon. Please use one of Streamlit's new

0 / 8
0
Live Share
Share Code Link
Blackbox
Search Terminal Output

Containers / amazing\_spence

amazing\_spe

4c8cc1fb3fef

STATUS

Running (19 minutes ago)

8501:8501

Logs

Inspect

Bind mounts

Exec

Files

Stats

2024-11-28 23:06:18  
2024-11-28 23:06:18 Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.  
2024-11-28 23:06:18  
2024-11-28 23:06:18 You can now view your Streamlit app in your browser.  
2024-11-28 23:06:18  
2024-11-28 23:06:18 URL: http://0.0.0.0:8501  
2024-11-28 23:06:18  
2024-11-28 23:06:50 2024-11-29 04:06:50.964  
2024-11-28 23:06:50 'st.cache' is deprecated and will be removed soon. Please use one of Streamlit's new  
2024-11-28 23:06:50 caching commands, 'st.cache\_data' or 'st.cache\_resource'. More information  
2024-11-28 23:06:50 [in our docs](https://docs.streamlit.io/develop/concepts/architecture/caching).  
2024-11-28 23:06:50 \*\*Note\*\*: The behavior of 'st.cache' was updated in Streamlit 1.36 to the new caching  
2024-11-28 23:06:50 logic used by 'st.cache\_data' and 'st.cache\_resource'. This might lead to some problems  
2024-11-28 23:06:50 or unexpected behavior in certain edge cases.  
2024-11-28 23:06:50  
2024-11-28 23:07:14 2024-11-29 04:07:14.787

RAM 1.44 GB
CPU 0.00%

(no s)
Retail
Chat
Dock
Dock
Dal X
2024
main
+
-
X

0:8080/job/Data Analytics CI-CD/56/console
90%

```

#8 [4/8] RUN groupadd -f docker && useradd -r -m -s /bin/bash -G docker dockeruser || echo "User already exists"
#8 CACHED

#9 [5/8] RUN echo "dockeruser ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers
#9 CACHED

#10 [6/8] COPY requirements.txt .
#10 CACHED

#11 [7/8] RUN pip install --no-cache-dir -r requirements.txt
#11 CACHED

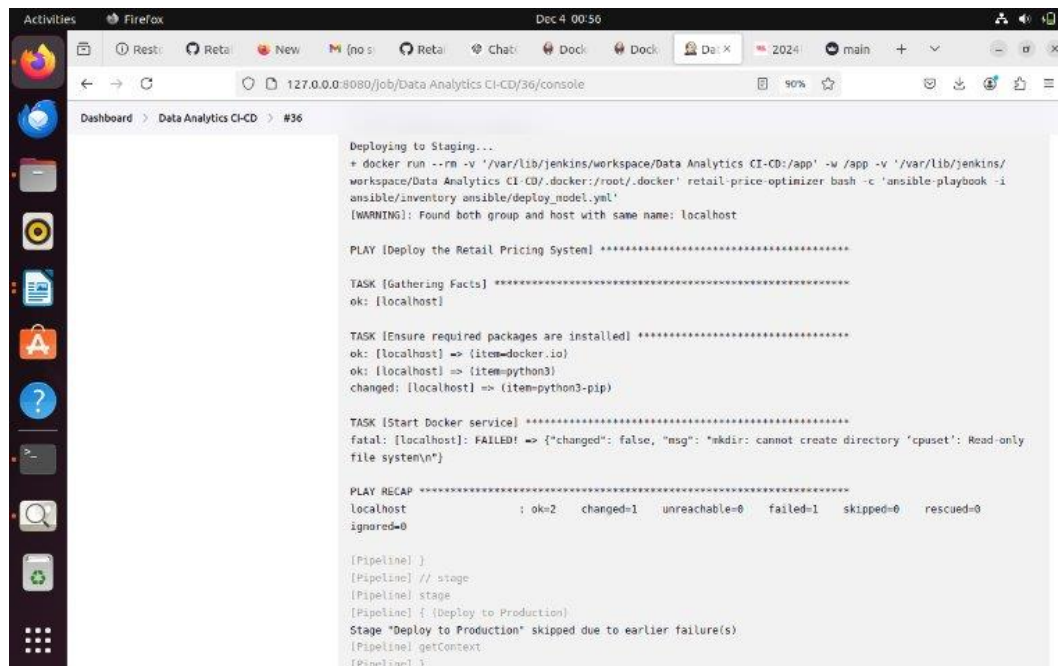
#12 [8/8] COPY . .
#12 DONE 0.5s

#13 exporting to image
#13 exporting layers 0.1s done
#13 writing image sha256:709e5cef09b6f5d81ccaf40f65b62b23b0197dd04543e66a7f818b4f1480033a done
#13 naming to docker.io/library/retail-price-optimizer done
#13 DONE 0.1s
WARNING: current commit information was not captured by the build: git was not found in the system: exec: "git": executable file not found in $PATH
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy to Staging)
[Pipeline] sh

```

Deployed to staging





```
Deploying to Staging...
+ docker run --rm -v '/var/lib/jenkins/workspace/Data Analytics CI-CD:/app' -w /app -v '/var/lib/jenkins/workspace/Data Analytics CI-CD/.docker:/root/.docker' retail-price-optimizer bash -c 'ansible-playbook -i ansible/inventory ansible/deploy_model.yml'
[WARNING]: Found both group and host with same name: localhost

PLAY [Deploy the Retail Pricing System] *****

TASK [Gathering Facts] *****
ok: [localhost]

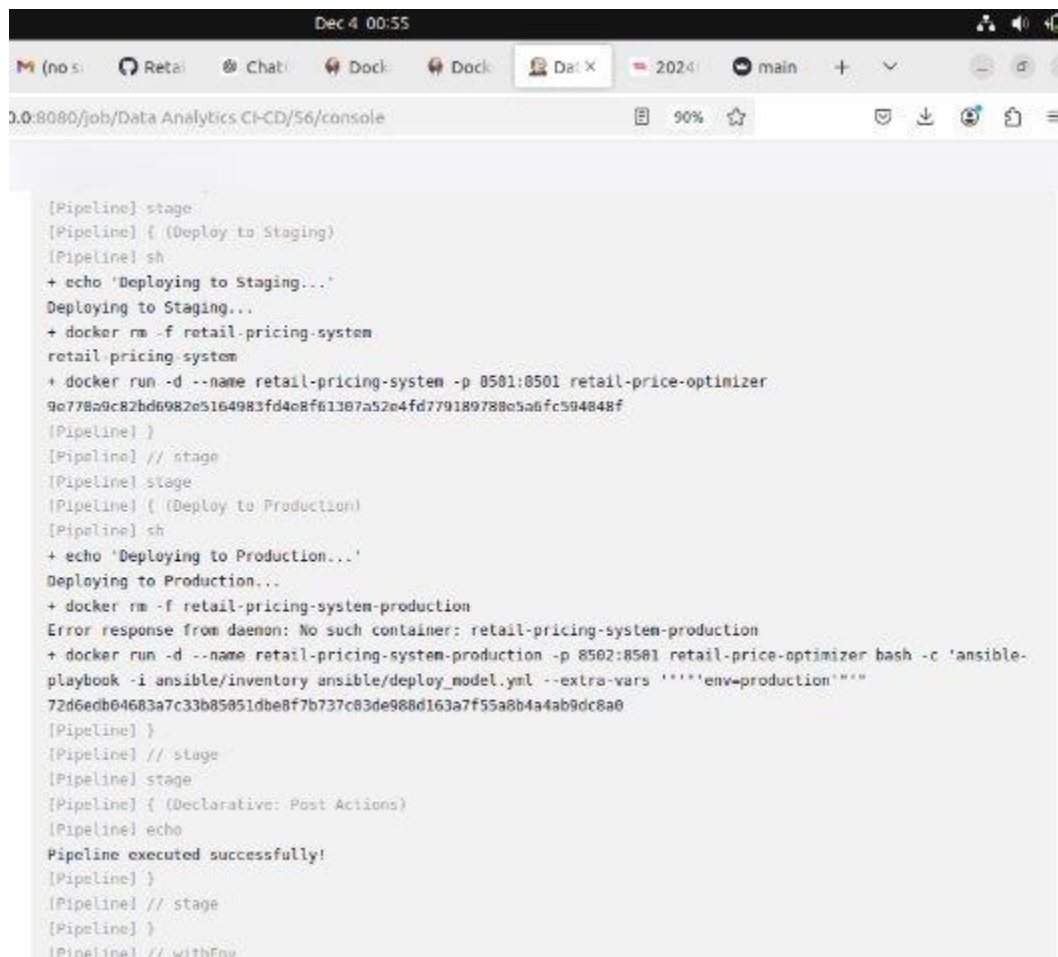
TASK [Ensure required packages are installed] *****
ok: [localhost] => (item=docker.io)
ok: [localhost] => (item=python3)
changed: [localhost] => (item=python3-pip)

TASK [Start Docker service] *****
fatal: [localhost]: FAILED! => {"changed": false, "msg": "mkdir: cannot create directory 'cpuset': Read-only file system\n"}

PLAY RECAP *****
localhost                : ok=2    changed=1    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0

[Pipeline]
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy to Production)
Stage "Deploy to Production" skipped due to earlier failure(s)
[Pipeline] getContext
[Pipeline] }
```

Deployed to production is successful



```
Dec 4 00:55
127.0.0.1:8080/job/Data Analytics CI-CD/56/console
90%

[Pipeline] stage
[Pipeline] { (Deploy to Staging)
[Pipeline] sh
+ echo 'Deploying to Staging...'
Deploying to Staging...
+ docker rm -f retail-pricing-system
retail-pricing-system
+ docker run -d --name retail-pricing-system -p 8501:8501 retail-price-optimizer
9e778a9c82bd6982e5164983fd4e8f613b7a52e4fd779189788e5a6fc594848f
[Pipeline]
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy to Production)
[Pipeline] sh
+ echo 'Deploying to Production...'
Deploying to Production...
+ docker rm -f retail-pricing-system-production
Error response from daemon: No such container: retail-pricing-system-production
+ docker run -d --name retail-pricing-system-production -p 8502:8501 retail-price-optimizer bash -c 'ansible-playbook -i ansible/inventory ansible/deploy_model.yml --extra-vars ''env=production''
72d6edb04683a7c33b85051dbe8f7b737c83de988d163a7f55a8b4a4ab9dc6a0
[Pipeline]
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] echo
Pipeline executed successfully!
[Pipeline]
[Pipeline] // stage
[Pipeline]
[Pipeline] // withEnv
```

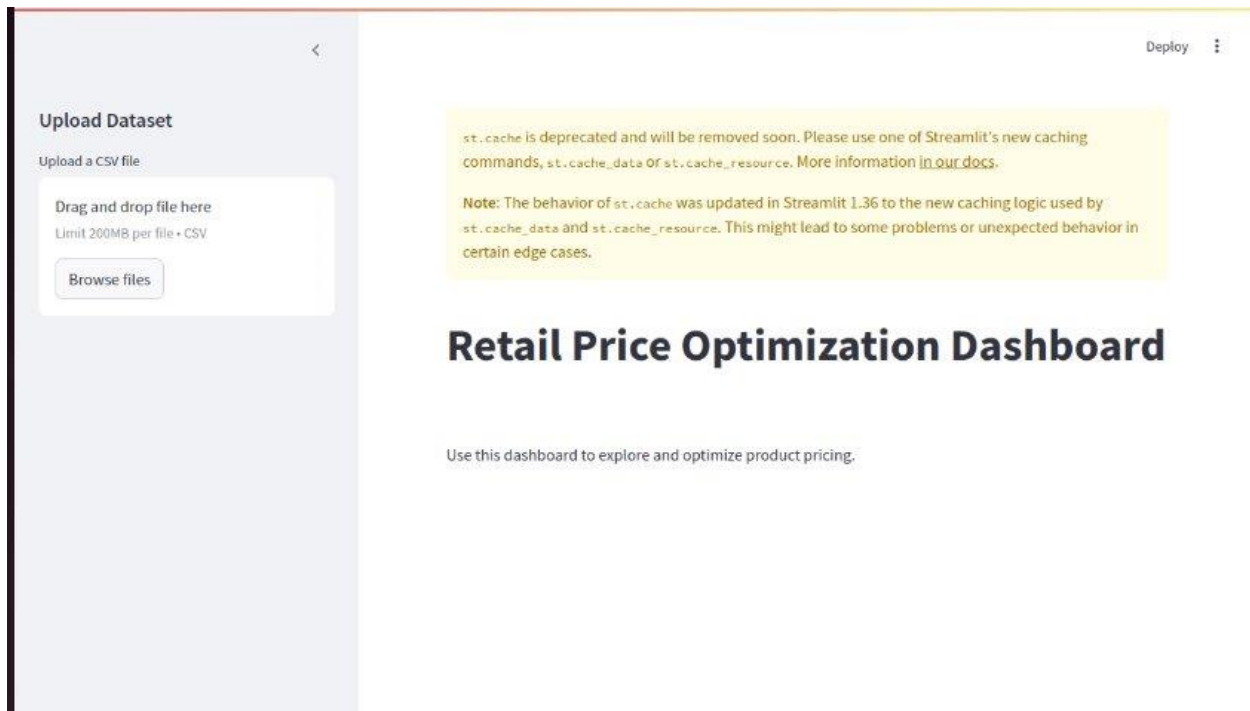
```

vboxuser@ram: $ ^C
vboxuser@ram: $ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS        NAMES
vboxuser@ram: $ sudo systemctl restart jenkins
[sudo] password for vboxuser:
vboxuser@ram: $ sudo systemctl restart docker
vboxuser@ram: $ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS        NAMES
097c4d50720c   retail-price-optimizer   "streamlit run main..." 20 seconds ago Up 18 seconds 0.0.0.0:8501->8501/tcp, :::8501->8501/tcp   retail-pricing-system
vboxuser@ram: $ sudo systemctl restart jenkins
[sudo] password for vboxuser:
Sorry, try again.
[sudo] password for vboxuser:
vboxuser@ram: $ sudo systemctl restart docker
vboxuser@ram: $ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS        NAMES
vboxuser@ram: $ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS        NAMES
72d6edb04683   retail-price-optimizer   "bash -c 'ansible-pl..." 11 seconds ago Up 10 seconds 0.0.0.0:8502->8501/tcp, :::8502->8501/tcp   retail-pricing-system-production
9e770a9c82bd   retail-price-optimizer   "streamlit run main..." 14 seconds ago Up 12 seconds 0.0.0.0:8501->8501/tcp, :::8501->8501/tcp   retail-pricing-system
vboxuser@ram: $

```

## User Interface

- **Framework Used:** Flask (Python-based web framework).
- **Key Features:**
  - **Input Panel:** Allows users to input toy attributes such as category, base price, and seasonal index to predict optimal price.
  - **Output Panel:**
    - Displays the predicted price.
    - Showcases the impact of discounts and seasonal factors.
  - **Visualization:**
    - Line charts showing actual vs. predicted sales.
    - Heatmaps depicting the relationship between price and sales volume.



## Key Challenges and Solutions

1. **Data Quality:**
  - a. Challenge: Missing values and outliers in sales data.
  - b. Solution: Implemented robust cleaning and outlier detection mechanisms.
2. **Model Interpretability:**
  - a. Challenge: Business stakeholders required easy-to-understand results.
  - b. Solution: Selected decision tree models for their inherent interpretability and provided feature importance visualizations.
3. **Deployment Complexity:**
  - a. Challenge: Coordinating CI/CD and containerization workflows.
  - b. Solution: Streamlined the pipeline using Jenkins' declarative pipeline scripts and Docker.

Errors we faced while working in this project:

Java Error





## Jenkins Issue:

### Solution to jenkins issue:

```
pipeline {
  agent {
    docker {
      image 'docker:24.0.7'
      args '--privileged -v /var/run/docker.sock:/var/run/docker.sock -v $WORKSPACE/.docker:/root/.docker'
    }
  }

  environment {
    DOCKER_IMAGE = "retail-price-optimizer"
    REPO_URL = "https://github.com/rkdhakal/Retail-Industry-Project"
    DOCKER_CONFIG = "$WORKSPACE/.docker"
  }
}
```

## Conclusion and Future Work

The project successfully demonstrates the application of AI to optimize pricing strategies in a retail toy store. The integration of machine learning with modern DevOps tools ensures a robust and scalable solution.

### ***Future Enhancements:***

1. Integrate additional features such as competitor pricing and inventory levels.
2. Enhance the user interface with predictive analytics dashboards.
3. Explore advanced ML models (e.g., ensemble methods) for improved accuracy.
4. Implement real-time data pipelines for dynamic price optimization.