Project:

Classification of Rakuten e-commerce products

Team members:

Karim Osman
&
Josipa Rupčić

Project overview:

- This project involves deploying three machine learning models for an e-commerce platform
- Goal of the project is to classify product descriptions and images into predefined categories

Model used:

- Text LSTM model
- Image VGG16 model
- Combined model integrating outputs of the first two models

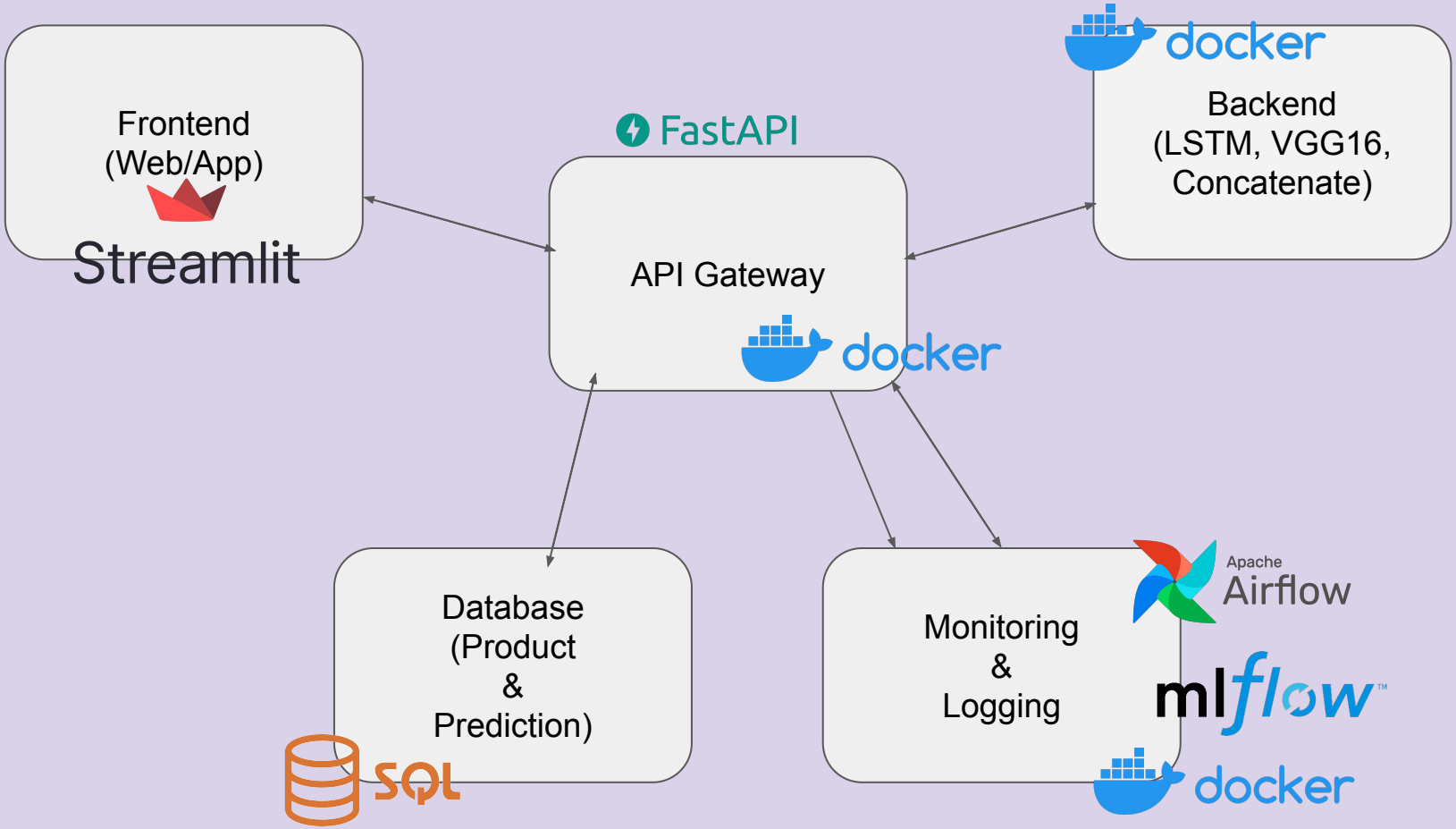
- *Purpose* of the project is to improve accuracy and efficiency for categorizing a large volume of products

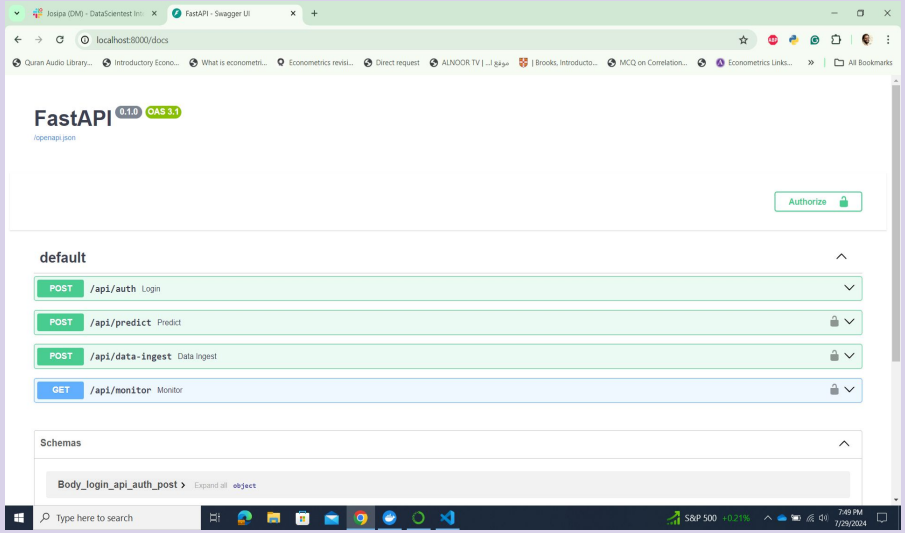
Stakeholders:

- **App Sponsor**: The e-commerce company aiming to enhance their product categorization system.
- **Users**: Internal staff, such as data analysts and product managers, who need accurate product categorization for inventory management and sales strategies.
- **Application Administrator**: IT and DevOps teams responsible for maintaining the application and ensuring its smooth operation.

Application Integration:

- **Context**: The application will be integrated into the company's existing e-commerce platform, hosted on a cloud server.
- **Medium**: The application will be accessed via an API, facilitating interaction between the models, the database, and the user interface.


Frontend
(Web/App)
Streamlit
FastAPI
API Gateway
docker
docker
Backend
(LSTM, VGG16,
Concatenate)
Database
(Product
&
Prediction)
SQL
Monitoring
&
Logging
Apache
Airflow
mlflow
docker



- **Database Interaction:**
  - SQLite for storing product and prediction data.
  - Functions to connect, query, and manage database operations.
- **Utilities:**
  - Text and image preprocessing functions to prepare data for model input.
  - Logging for monitoring and error tracking.

Purpose: This application demonstrates integrating FastAPI with machine learning models for secure, scalable, and efficient predictions and data handling.

- **Authentication:**
  - Utilizes OAuth2 for secure authentication.
  - Access tokens generated with JWT, secured by SECRET_KEY and HS256 algorithm.
- **Machine Learning Models:**
  - **Text Model:** Best LSTM model for text classification.
  - **Image Model:** Best VGG16 model for image processing.
  - Models loaded and configured from JSON files.
- **APIs:**
  - **/api/auth:** User login, returns JWT access token.
  - **/api/predict:** Predicts using text and image inputs. Saves predictions to the database.
  - **/api/data-ingest:** Admin-only endpoint for ingesting new product data.
  - **/api/monitor:** Returns model health metrics and status.

Predictions Table:

- **Purpose**: This table stores the results of product classification predictions made by the machine learning models.
- **Role**: It acts as a repository for keeping track of prediction outputs, which can be used for analysis and further processing.

Products Table:

- **Purpose**: This table holds the detailed information about the products, such as their unique identifiers and descriptions.
- **Role**: It serves as the main source of product data that is used for generating predictions and managing inventory.

SQLite Sequence Table:

- **Purpose**: This table is part of the SQLite database system and is used to keep track of the primary keys for other tables.
- **Role**: It ensures that unique identifiers are correctly increased, supporting the integrity and consistency of the database.

Create Table | Create Index | Modify Table | Delete Table | Print | Refresh

Name	Type	Schema
Tables (3)		
predictions		CREATE TABLE predictions (id INTEGER PRIMARY KEY AUTOINCREMENT, product_id INTEGER, description TEXT, im
products		CREATE TABLE products (id INTEGER PRIMARY KEY AUTOINCREMENT, product_id INTEGER, designation TEXT, des
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
Indices (0)		
Views (0)		
Triggers (0)		

Table: predictions

	id	product_id	description	image	prediction
	Fil...	Filter	Filter	Filter	Filter
1	1	NULL	This is a sample product description.	/9j/4AAQSkZJRgABAgAAAQABAAD/...	1302
2	2	NULL	This product is PSC 2020	/9j/4AAQSkZJRgABAgAAAQABAAD/...	1302
3	3	NULL	this movie or product for adult	/9j/4AAQSkZJRgABAgAAAQABAAD/...	1300
4	4	NULL	This game for Maria	/9j/4AAQSkZJRgABAgAAAQABAAD/...	1280
5	5	NULL	this test fastapi after edit code	/9j/4AAQSkZJRgABAgAAAQABAAD/...	10
6	6	NULL	This is book for cooking	/9j/4AAQSkZJRgABAgAAAQABAAD/...	2280

Predictions Table

- **Purpose**: Stores prediction results related to products, often used in machine learning contexts.
- **Columns**:
  - **Results Storage**: This could include various columns, such as a **Product ID** to associate predictions with specific products, predicted category or label, prediction scores, timestamps, etc.
- **Importance**: The **Predictions** table is crucial for applications that utilize machine learning to make predictions about products, such as recommending categories or predicting sales trends. Storing predictions allows for analysis, reporting, and improving predictive models.

Table: products

	id	product_id	designation	description	image_id	prdtypecode
	Fil...	Filter	Filter	Filter	Filter	Filter
1	1	NULL	NULL	This a rabbit game	NULL	Children games
2	2	NULL	NULL	this is an image for kids games	NULL	children games
3	3	NULL	NULL	This product for Josipa	NULL	300
4	4	NULL	NULL	This product for test	/9j/4AAQSkZJRgABAgAAAQABAAD/...	5000
5	5	NULL	NULL	This movie for adventure people .	/9j/4AAQSkZJRgABAgAAAQABAAD/...	3150
6	6	NULL	NULL	This movie is for fun	/9j/4AAQSkZJRgABAgAAAQABAAD/...	9000

Products Table

- **Purpose**: Stores information about individual products.
- **Columns**:
  - **IDs**: A unique identifier for each product. This is typically the primary key of the table, ensuring that each product can be uniquely identified.
  - **Descriptions**: Textual information about the product, which could include details such as name, features, or specifications.
  - **Image Paths**: File paths or URLs that point to images of the products. This allows the application to display product images as needed.
- **Importance**: The **Products** table is essential for storing all the key information about each product. It allows quick access to product details and images, which is vital for applications such as e-commerce platforms, where users need to view product descriptions and images.

Model Comparison Table

Model	Type	Function
Text LSTM	LSTM Neural Network	Classifies product descriptions
Image VGG16	CNN (VGG16)	Classifies product images
Concatenate	Combined Output Model	Integrates text and image data

Text LSTM Model:

- **Type**: Long Short-Term Memory (LSTM) neural network.
- **Function**: Processes and classifies product descriptions.
- **Performance**: Evaluated based on accuracy, robustness, and prediction time.

```
class TextLSTMModel:
    def __init__(self, max_words=10000, max_sequence_length=10):
        self.max_words = max_words
        self.max_sequence_length = max_sequence_length
        self.tokenizer = Tokenizer(num_words=max_words, oov_token='')
        self.model = None

    def preprocess_and_fit(self, X_train, y_train, X_val, y_val):
        self.tokenizer.fit_on_texts(X_train['description'])

        tokenizer_config = self.tokenizer.to_json()
        with open('/content/tokenizer_config.json', 'w', encoding='utf-8') as json_file:
          json_file.write(tokenizer_config)

        train_sequences = self.tokenizer.texts_to_sequences(X_train['description'])
        train_padded_sequences = pad_sequences(train_sequences, maxlen=self.max_sequence_length, padding='post',

        val_sequences = self.tokenizer.texts_to_sequences(X_val['description'])
        val_padded_sequences = pad_sequences(val_sequences, maxlen=self.max_sequence_length, padding='post', trun
```

```python
class ImageVGG16Model:
    def __init__(self):
        self.model = None

    def preprocess_and_fit(self, X_train, y_train, X_val, y_val):
        # Paramètres
        batch_size = 32
        num_classes = 27

        df_train = pd.concat([X_train, y_train.astype(str)], axis=1)
        df_val = pd.concat([X_val, y_val.astype(str)], axis=1)

        # Créer un générateur d'images pour le set d'entraînement
        train_datagen = ImageDataGenerator()  # Normalisation des valeurs de pixel
        train_generator = train_datagen.flow_from_dataframe(
            dataframe=df_train,
            x_col='image_path',
            y_col='prdtypecode',
            target_size=(224, 224),  # Adapter à la taille d'entrée de VGG16
            batch_size=batch_size,
            class_mode='categorical',  # Utilisez 'categorical' pour les entiers encodés en one-hot
            shuffle=True
        )

        # Créer un générateur d'images pour le set de validation
        val_datagen = ImageDataGenerator()  # Normalisation des valeurs de pixel
        val_generator = val_datagen.flow_from_dataframe(
            dataframe=df_val,
            x_col='image_path',
            y_col='prdtypecode',
            target_size=(224, 224),
            batch_size=batch_size,
            class_mode='categorical',
            shuffle=False  # Pas de mélange pour le set de validation
        )
```

Image VGG16 Model:

- **Type**: Convolutional Neural Network (CNN) based on the VGG16 architecture.
- **Function**: Processes and classifies product images.
- **Performance**: Evaluated on accuracy, robustness, and prediction time.

Concatenate Model:

- **Type**: Combines the probabilities from the Text LSTM and Image VGG16 models.
- **Function**: Integrates text and image data for final classification.
- **Performance**: Combines the strengths of the individual models for improved accuracy.

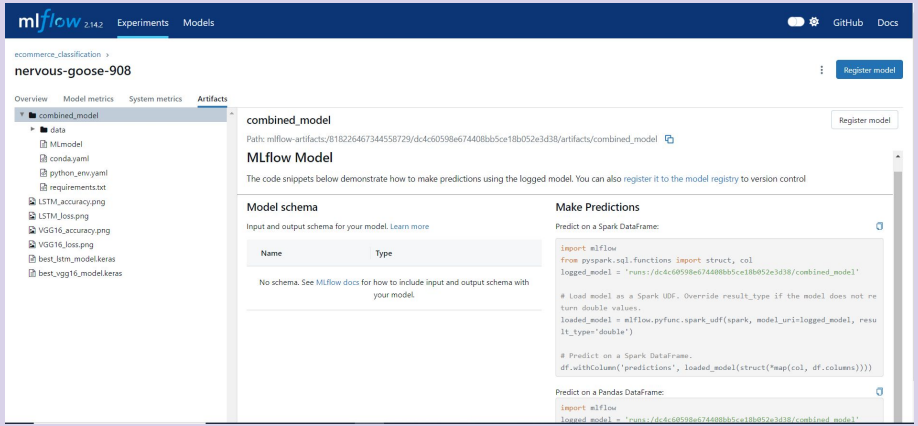
```python
def optimize(self, lstm_proba, vgg16_proba, y_train):
    # Recherche des poids optimaux en utilisant la validation croisée
    best_weights = None
    best_accuracy = 0.0

    for lstm_weight in np.linspace(0, 1, 101):  # Essayer différents poids pour LSTM
        vgg16_weight = 1.0 - lstm_weight  # Le poids total doit être égal à 1

        combined_predictions = (lstm_weight * lstm_proba) + (vgg16_weight * vgg16_proba)
        final_predictions = np.argmax(combined_predictions, axis=1)
        accuracy = accuracy_score(y_train, final_predictions)

        if accuracy > best_accuracy:
            best_accuracy = accuracy
            best_weights = (lstm_weight, vgg16_weight)

    return best_weights
```



Key Components:

1. MLflow Setup:

racking URI: Connects to the MLflow server (http://localhost:5000).

Experiment: Logs data under ecommerce_classification.

2. Data Preprocessing:

Text and Image Processing: Uses TextPreprocessor and ImagePreprocessor to clean and prepare data.

3 . Model Training:

Text LSTM Model: Trains on text data.

Image VGG16 Model: Trains on image data.

Combined Model: Integrates both LSTM and VGG16 outputs.

4. Logging and Visualization:

- **Metrics and Artifacts:** Logs accuracy, loss plots, and trained models.
- **Visualization:** Generates and saves accuracy and loss plots.

5. Model Saving:

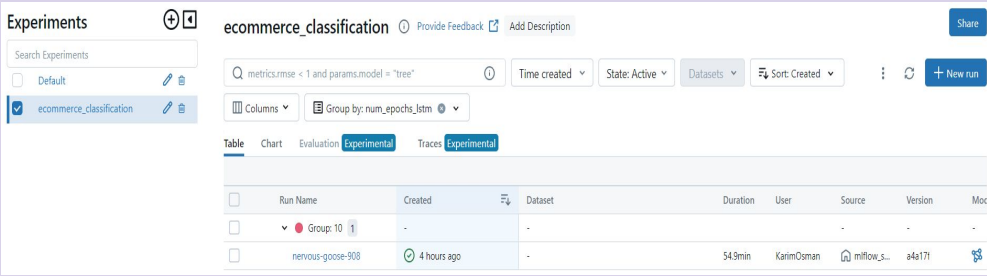
- **Paths:** Saves models to specified file paths for future use.

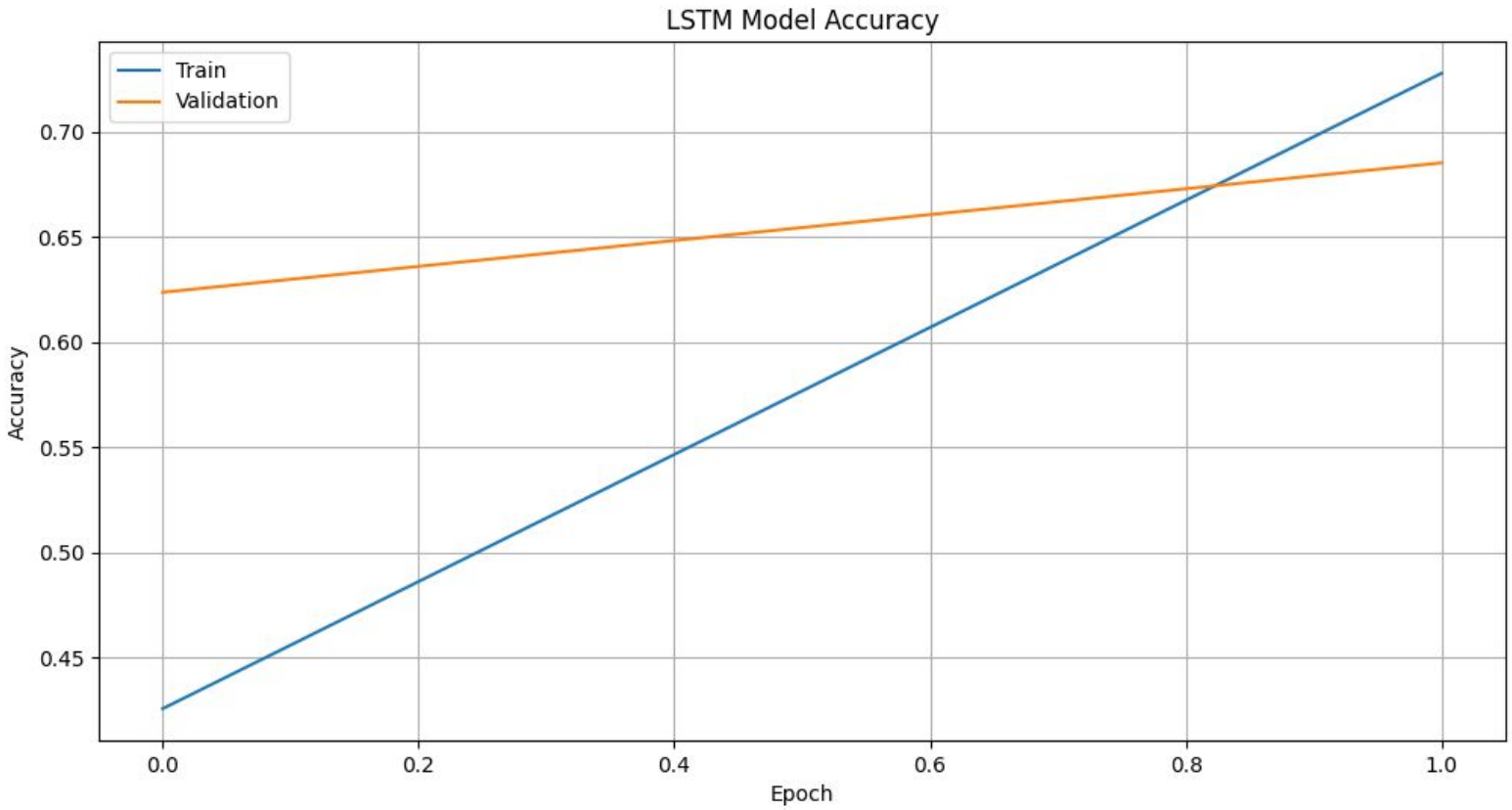
6. Metrics Logging:

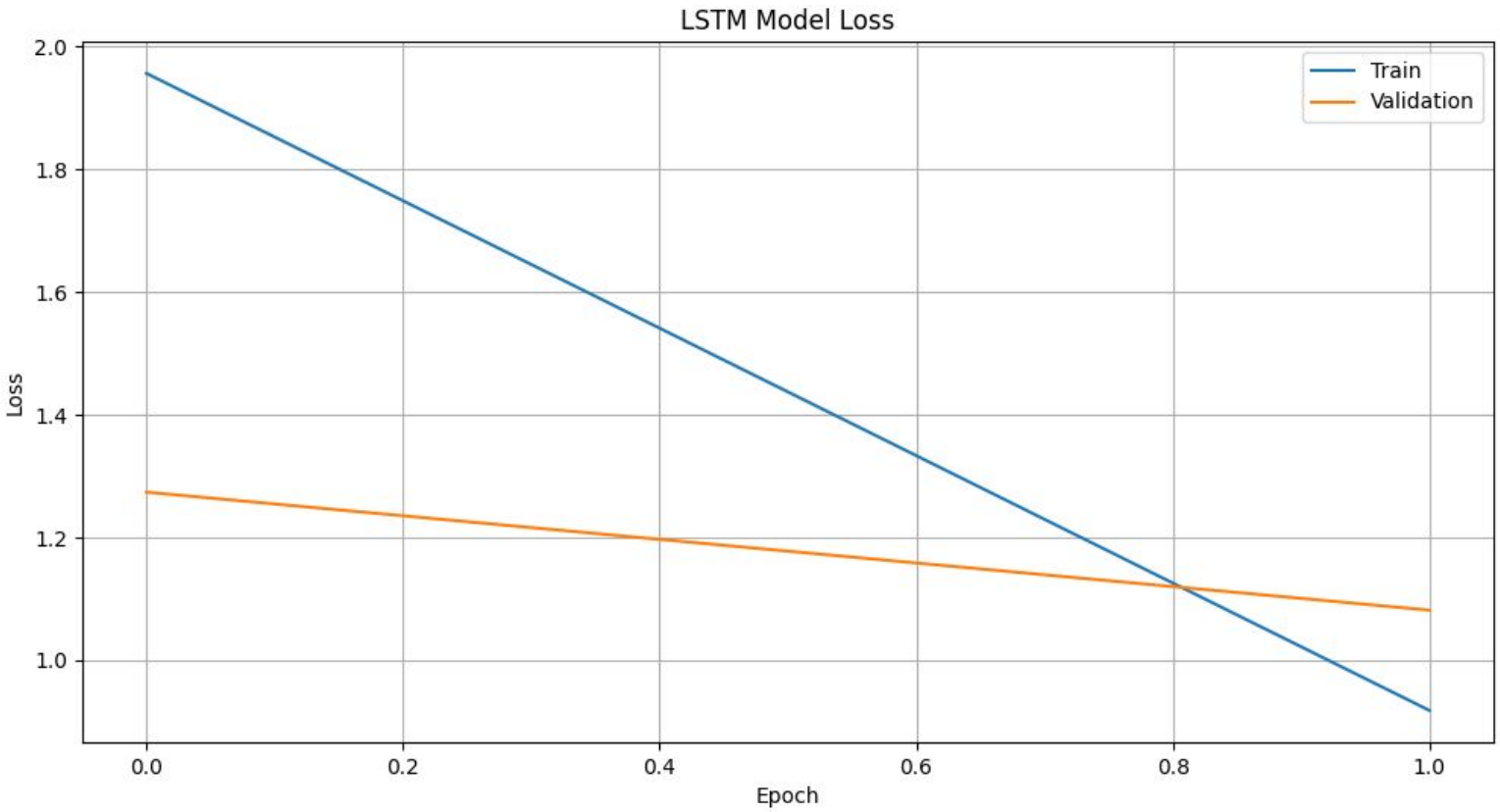
- **Performance Metrics:** Logs model accuracies (LSTM: 0.92, VGG16: 0.85).

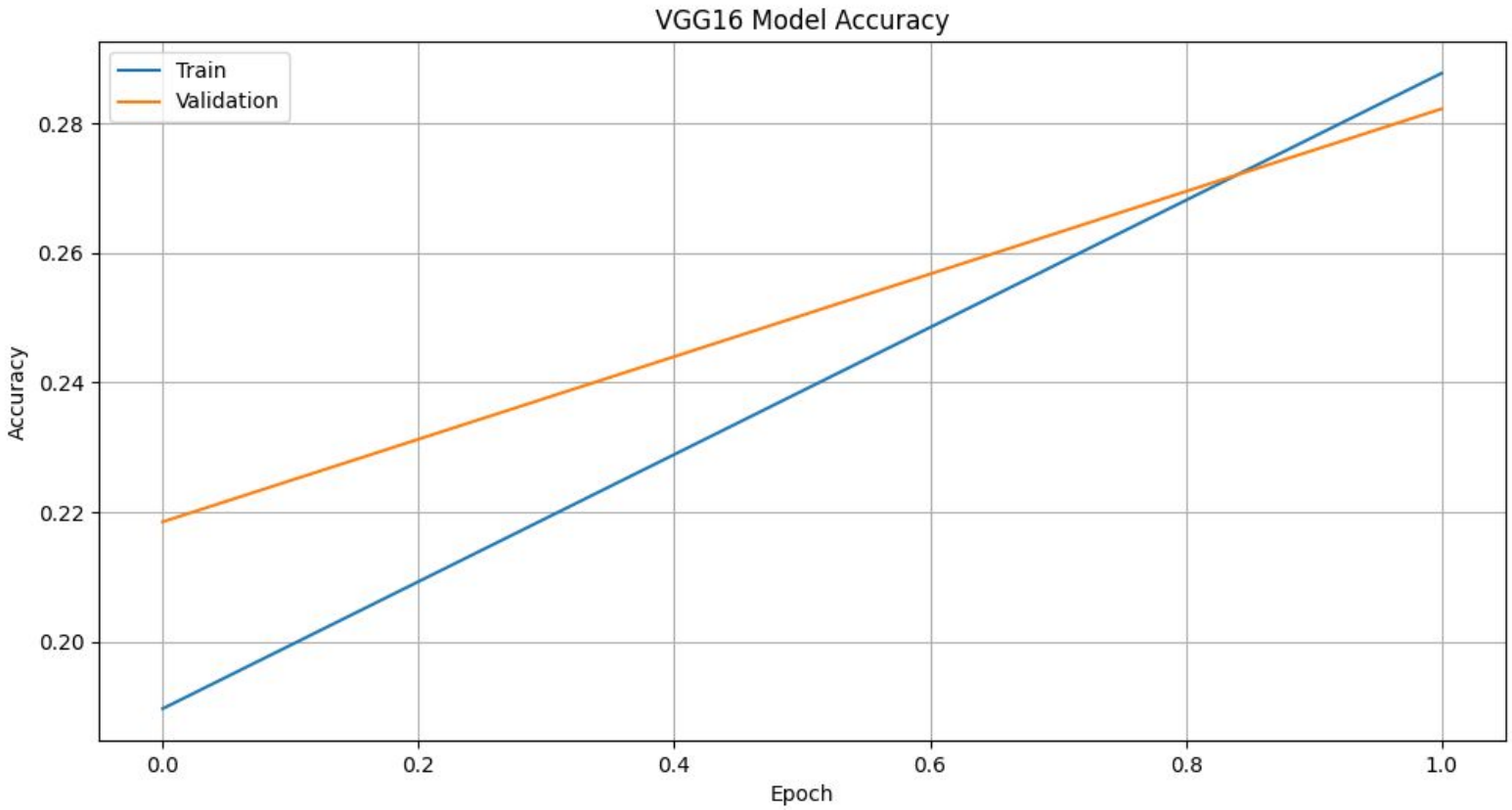
Benefits:

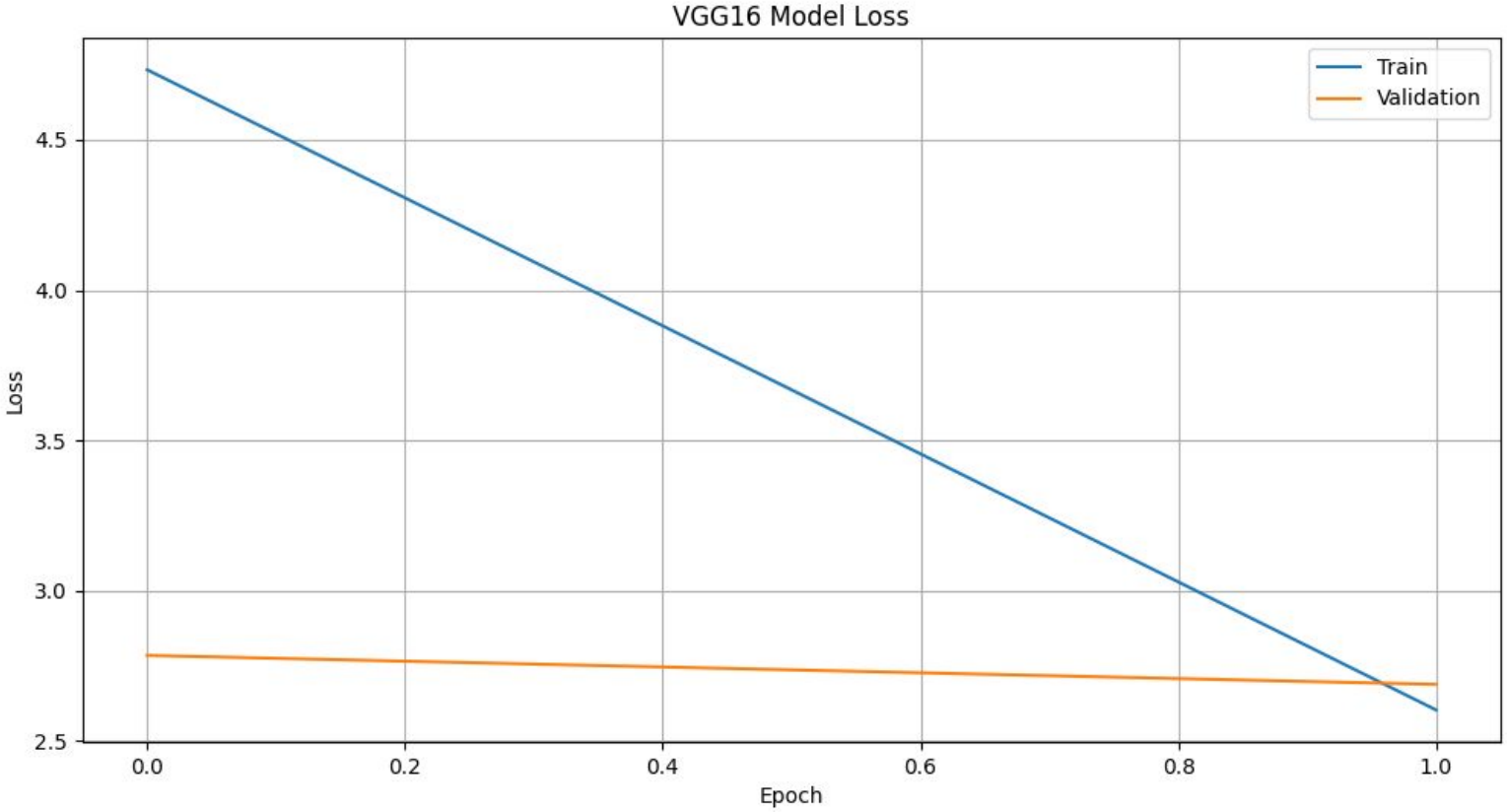
- **Reproducibility:** Ensures experiments can be repeated with consistent results.
- **Tracking and Management:** Centralizes experiment tracking and artifact storage.
- **Automation:** Streamlines the entire ML pipeline from data ingestion to model deployment.

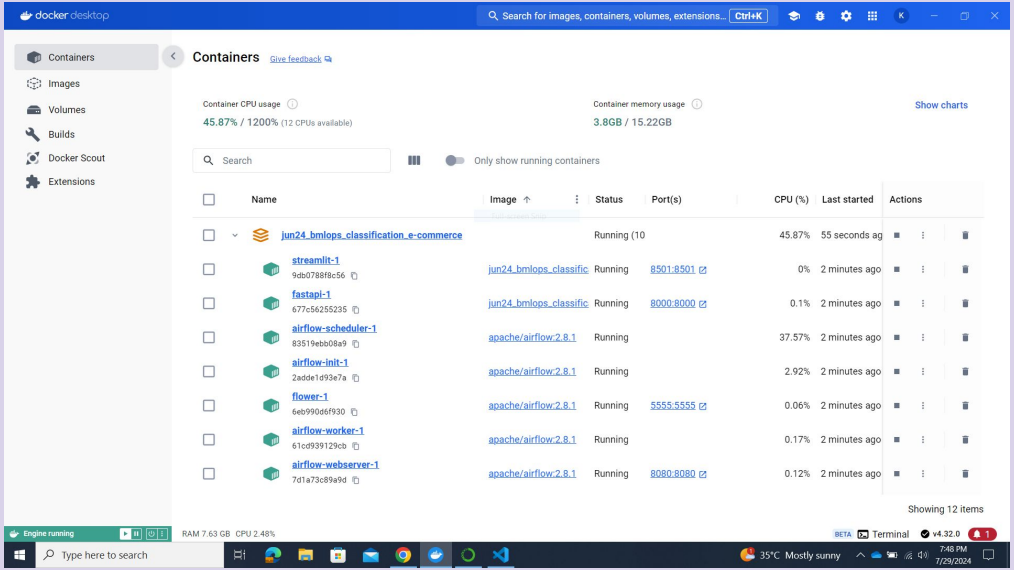



LSTM Model Accuracy
Train
Validation
0.70
0.65
0.60
0.55
0.50
0.45
Accuracy
0.0
0.2
0.4
0.6
0.8
1.0
Epoch


LSTM Model Loss
Train
Validation
Loss
Epoch
2.0
1.8
1.6
1.4
1.2
1.0
0.0
0.2
0.4
0.6
0.8
1.0


VGG16 Model Accuracy
Train
Validation
Accuracy
0.28
0.26
0.24
0.22
0.20
0.0
0.2
0.4
0.6
0.8
1.0
Epoch


VGG16 Model Loss
Train
Validation
Loss
4.5
4.0
3.5
3.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0
Epoch



4. Portability:

- **Description:** Docker containers can run on any system that supports Docker, from local machines to cloud servers.
- **Benefit:** This portability simplifies the transition between development, testing, and production environments.

5. Simplified Deployment:

- **Description:** Dockerfiles provide a blueprint for building containers, automating the setup of environments for each component.
- **Benefit:** This automation reduces manual configuration efforts and speeds up the deployment process.

Consistent Environment:

- **Description:** Docker ensures that all components (frontend, API gateway, backend models) run in identical environments across different stages of development, testing, and production.
- **Benefit:** This consistency eliminates issues caused by differences in environments, making the deployment process more reliable.

2. Isolation and Security:

- **Description:** Each Docker container operates independently, providing isolation between different components.
- **Benefit:** This isolation enhances security and ensures that changes in one component do not affect others.

3. Scalability:

- **Description:** Docker makes it easy to scale individual components by running multiple instances of a container as needed.
- **Benefit:** This flexibility supports the demands of varying workloads, especially important for handling large volumes of data in real-time.

Pipelines:

- **Automation of Testing, Building, and Deploying:**
  - **Description:** CI/CD pipelines automate the process of testing code, building Docker images, and deploying them to production environments. This ensures that code changes are integrated and deployed smoothly.
  - **Benefit:** Reduces manual intervention, accelerates deployment cycles, and improves the reliability of deployments.
- **Workflows Based on ci.yml and deployment.yml:**
  - **ci.yml:** Defines the steps for continuous integration, including code checkout, dependency installation, and testing.
  - **deployment.yml:** Specifies the deployment process, including building Docker images and deploying them to the target environment.

```yaml
name: Airflow Workflow

on:
  push:
    branches:
      - main

jobs:
  airflow-setup:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Set up Python
        uses: actions/setup-python@v4
        with:
          python-version: 3.9

      - name: Install dependencies
        run: |
          pip install -r airflow/requirements.txt

      - name: Build Docker image
        run: docker build -t airflow-image -f airflow/Dockerfile .

      - name: Run Airflow Docker container
        run: |
          docker run -d -p 8080:8080 airflow-image

      - name: Check Airflow webserver status
        run: curl -sSf http://localhost:8080
```

Deploy

Login

Username

Password

Login

Model Prediction

Upload an Image

Drag and drop file here

Limit 200MB per file • JPG, JPEG, PNG

Browse files

Product Description