**Amazon ML Hackathon**

**Team Name – The Shifters**

**Members –**

**Nandini Pajgade**

**Prathamesh Sable**

**Aryan Parishwad**

**Mugdha Arvind**

**Feature Extraction from Images Using Machine Learning**

**Introduction:**

**In digital marketplaces, precise product details are often missing from product listings. These details, such as weight, volume, dimensions, voltage, and wattage, are crucial for buyers, especially in industries like healthcare and e-commerce. Our goal is to build a machine learning model that can predict these key entity values from images, addressing the challenge of incomplete product descriptions.**

**This document outlines the step-by-step approach used to create a machine learning model that extracts and standardizes key product details from images and text.**

**Problem Statement:**

**The problem is to develop a model that can extract and predict entity values from images and text in the dataset. Key entities like weight, width, depth, height, and volume need to be extracted and standardized into predefined units such as centimetres, kilograms, and watts.**

**Approach:**

**Step 1: Load Data**

**The training and test datasets are loaded using Python’s Pandas library. The training data contains key information such as group\_id, entity\_name, and entity\_value.**

import pandas as pd

train\_df = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/train.csv')

test\_df = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/test.csv')

**Step 2: Data Preprocessing**

**We concatenate the train and test datasets to apply transformations consistently. Then, the categorical columns (group\_id and entity\_name) are encoded using Scikit-learn’s LabelEncoder. This encoding converts categorical text features into numeric values that can be processed by machine learning algorithms.**

from sklearn.preprocessing import LabelEncoder

le\_group = LabelEncoder()

le\_entity = LabelEncoder()

# Concatenate train and test data

all\_data = pd.concat([train\_df, test\_df], axis=0)

all\_data['group\_id\_encoded'] = le\_group.fit\_transform(all\_data['group\_id'])

all\_data['entity\_name\_encoded'] = le\_entity.fit\_transform(all\_data['entity\_name'])

# Split data back

train\_df = all\_data.iloc[:len(train\_df)].copy()

test\_df = all\_data.iloc[len(train\_df):].copy()

**Step 3: Extract Numeric Values**

**The entity\_value column contains both text and numeric data. We use a regular expression to extract the numeric portion of the entity\_value, which is crucial for predicting values like weight and volume.**

train\_df = train\_df.assign(numeric\_value=train\_df['entity\_value'].str.extract(r'(\d+\.\d+|\d+)').astype(float))

**Step 4: Model Training and Validation**

**We split the training data into training and validation sets (80% for training and 20% for validation) to evaluate model performance. We use a RandomForestRegressor for numeric value prediction. This model is chosen for its ability to handle structured data effectively and perform regression tasks.**

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestRegressor

X = train\_df[['group\_id\_encoded', 'entity\_name\_encoded']]

y = train\_df['numeric\_value']

# Split into training and validation sets

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train the RandomForest model

model = RandomForestRegressor(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

**Step 5: Prediction on Test Data**

**We predict the numeric values for the test dataset using the trained RandomForest model. The predictions represent the entity values like weight, volume, and dimensions.**

X\_test = test\_df[['group\_id\_encoded', 'entity\_name\_encoded']]

pred\_numeric\_values = model.predict(X\_test)

**Step 6: Standardizing Units**

**For consistency, each entity type must be standardized to the correct unit. The allowed units are mapped using a dictionary, ensuring that all predictions are in the correct format. For instance, dimensions like width and height are measured in centimetres, and weight is measured in kilograms.**

entity\_unit\_map = {

'width': 'centimetre',

'depth': 'centimetre',

'height': 'centimetre',

'item\_weight': 'kilogram',

'maximum\_weight\_recommendation': 'kilogram',

'voltage': 'volt',

'wattage': 'watt',

'item\_volume': 'litre'

}

# Function to standardize unit

def standardize\_unit(entity\_name):

return entity\_unit\_map.get(entity\_name, 'unit')

# Formatting predictions with standardized units

def format\_prediction(value, entity\_name):

unit = standardize\_unit(entity\_name)

return f"{value:.2f} {unit}"

# Apply to test dataset

test\_df['prediction'] = [format\_prediction(val, name) for val, name in zip(pred\_numeric\_values, test\_df['entity\_name'])]

**Step 7: Submission File**

**Finally, we create the submission file that contains the index and prediction columns. The predictions are saved in a CSV file format for easy submission to the hackathon platform.**

submission\_df = test\_df[['index', 'prediction']]

submission\_df.to\_csv('submission.csv', index=False)

print("Submission file saved as submission.csv")

**Conclusion:**

**This approach leverages RandomForest regression and machine learning techniques to predict key product entity values from the provided data. The predictions are standardized using predefined units for consistency, and the results are saved in a submission file for evaluation.**

**By automating the extraction of key product information from images, this model can save time, improve accuracy, and ensure that marketplaces provide vital details to customers, enhancing the overall digital shopping experience.**