Project Summary Report – Amazon ML Challenge 2024

# Team Name: AI Wizards

## Project Title: Automated Image Download and OCR for Entity Value Prediction

**Introduction**

This project aims to develop a model that downloads images from URLs provided in a dataset, preprocesses them, extracts text using Optical Character Recognition (OCR), and predicts entity values. The goal is to process data containing images and associated text to build a machine learning model for entity value prediction.

**Dataset Overview**

The following datasets were used:

1. **Train Dataset (train.csv)** – Contains image\_link, group\_id, entity\_name, and entity\_value. This data was used for training the model.
2. **Test Dataset (test.csv)** – Contains index, image\_link, group\_id, and entity\_name.
3. **Sample Test Dataset (sample\_test.csv)** – Used for validation checks.
4. **Sample Test Out Datasets (sample\_test\_out.csv, sample\_test\_out\_fail.csv)** – Used for final model validation.

The images contained information such as width, depth, item weight, voltage, and wattage. The task was to predict these entity values based on the provided image data.

**Methodology**

**1. Image Downloading**

A Python script was written to download images from the URLs provided in the dataset. The images were stored locally for processing.

python

Copy code

def download\_images(image\_url, image\_id, save\_dir):

response = requests.get(image\_url)

if response.status\_code == 200:

image = Image.open(BytesIO(response.content))

image.save(os.path.join(save\_dir, f'{image\_id}.jpg'))

**2. Image Preprocessing and Visualization**

After downloading the images, we visualized a few of them using matplotlib to verify that they were correctly saved and intact for further processing.

**3. Text Extraction with OCR**

Text from the images was extracted using pytesseract, with additional preprocessing to improve accuracy. This included converting the images to grayscale and applying thresholding using OpenCV for noise reduction.

python

Copy code

def extract\_text\_from\_image(image\_path):

img = preprocess\_image(image\_path)

return pytesseract.image\_to\_string(img)

**4. Entity Value Preprocessing**

The entity\_value field in the training dataset contained numerical values with units (e.g., "10 kilogram"). We wrote a function to handle different formats, including ranges and missing values, and standardized the numerical values and units.

python

Copy code

def preprocess\_entity\_value(value):

pattern = re.compile(r'(?:\[.\*?\]|\-?\d+(\.\d+)?)\s+[a-zA-Z\s]+(?:to\s+(?:\[.\*?\]|\-?\d+(\.\d+)?)\s+[a-zA-Z\s]+)?')

**5. Model Training**

Once the data was preprocessed, categorical features such as group\_id and entity\_name were one-hot encoded. The entity\_value column was treated as the target variable. We split the data into training and testing sets and trained a regression model to predict the values.

python

Copy code

X = pd.get\_dummies(train[['group\_id', 'entity\_name']])

y = train['value']

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

**6. Sanity Check and Validation**

We validated the model's predictions using the sample test datasets (sample\_test\_out.csv and sample\_test\_out\_fail.csv). This step ensured that the model was correctly predicting values and aligned with the dataset's expectations.

**Challenges**

* **OCR Accuracy:** Variations in image quality impacted the accuracy of text extraction. Preprocessing the images (e.g., grayscale and thresholding) improved OCR accuracy but could not eliminate all errors.
* **Handling Units:** The entity\_value field contained various units (e.g., kg, cm, mm), which required careful preprocessing to ensure consistency in the data before modeling.

**Conclusion**

The project successfully implemented an end-to-end pipeline for downloading images, extracting text using OCR, and predicting entity values. Preprocessing the images and applying machine learning techniques led to satisfactory predictions. The approach used here is scalable and can be extended to handle more complex tasks involving image and text data.

**Tools and Technologies Used**

* **Programming Language:** Python
* **Libraries:**
  + pandas, requests, PIL for data and image handling.
  + pytesseract, OpenCV for OCR and image preprocessing.
  + scikit-learn for data preprocessing and model training.
  + matplotlib for visualization.

**Next Steps:** Improving the OCR text extraction through deep learning-based approaches could further enhance the model’s performance. Additionally, incorporating more advanced machine learning models could yield better predictions.