<u>PRODUCT DEVELOPMENT USING DEEP LEARNING</u>

TITLE

PREDICTING THE AMAZON PRODUCT QUALITY WITH IT'S CUSTOMER REVIEW

Name: Abdullah Firdowsi M F

Roll Number: 717821i102

Course Code: 21ID24

Department: AD

ABSTRACT

To develop a deep learning model to predict Amazon product quality based on customer reviews. The model will analyse sentiment and content of customer reviews, extracting relevant features like sentiment polarity, review length, and keyword frequency. A dataset of Amazon product reviews will be collected and pre-processed, and trained using various architectures. it can be deployed to make predictions on new, unseen products based on their customer reviews. The results of the model will provide valuable insights for both customers and sellers. Customers will benefit from more accurate predictions of product quality, enabling them to make informed purchasing decisions. Sellers can leverage these predictions to identify areas for improvement and enhance their product offerings. The findings will contribute to the field of sentiment analysis and have practical implications for e-commerce platforms like Amazon.

DATA COLLECTION

I used Scraper API to retrieve live streaming Amazon data, you can follow these steps to collect the data:

1. Set up your Scraper API account: Sign up for a Scraper API account and obtain your API key. This key will be used to authenticate your requests to the Scraper API.
2. Configure your scraper: Use the Scraper API documentation to configure your scraper to retrieve live streaming Amazon data. This may involve specifying the target URL, setting up any required headers or parameters, and defining the desired data to be extracted.
3. Make API requests: Use the Scraper API client library or make HTTP requests directly to the Scraper API endpoint, passing in your API key and the necessary parameters. This will trigger the scraper to retrieve the live streaming Amazon data.
4. Process the data: Once you receive the data from the Scraper API, you can process it as needed. This may involve parsing the HTML or JSON response, extracting the relevant information (such as product reviews), and storing it in a suitable format for further analysis.

5. Handle rate limits and errors: Scraper API may have rate limits or encounter errors during the scraping process. Make sure to handle these situations gracefully by implementing appropriate error handling and retry mechanisms.
6. Ensure compliance: When scraping live streaming Amazon data, it is important to comply with Amazon's terms of service and usage policies. Make sure to review and adhere to these guidelines to avoid any legal or ethical issues.

By using Scraper API to retrieve live streaming Amazon data, you can collect real-time information and perform analysis or build models based on the latest data available.

EXPLANATION OF CONCEPT WITH ALGORITHM USED

The algorithm used for text classification in this project is a combination of Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) models.

Convolutional Neural Network (CNN) model for text classification

CNN utilizes an activation function which helps it run in kernel (i.e) high dimensional space for neural processing. For Natural language processing, text classification is a topic in which one needs to set predefined classes to free-text documents. It used for image processing tasks, but they can also be applied to text classification. In the context of text classification, a CNN model utilizes convolutional layers to extract local features from the input text. These convolutional layers apply filters to the text, capturing important patterns and features. The CNN model for text classification consists of the following steps:

1. Input Layer: The input layer takes the text data as input.
2. Embedding Layer: The embedding layer converts the text data into a numerical representation, such as word embeddings or character embeddings.
3. Convolutional Layers: The convolutional layers apply filters of different sizes to the embedded text, capturing local features.
4. Pooling Layers: The pooling layers reduce the dimensionality of the extracted features, retaining the most important information.
5. Fully Connected Layers: The fully connected layers process the pooled features and make predictions for the predefined classes.
6. Output Layer: The output layer produces the final classification results.

LSTM (Long Short-Term Memory) model for text classification

LSTM (Long Short-Term Memory) models have proven to be a powerful tool for text classification in Python. With their ability to capture long-term dependencies and handle sequential data, LSTM models offer improved accuracy in classifying text. This models are a type of recurrent neural network (RNN) that are specifically designed to handle sequential data. In the context of text classification, LSTM models excel at capturing long-term dependencies and understanding the context of the text. The LSTM model for text classification consists of the following steps:

1. Input Layer: The input layer takes the text data as input.
2. Embedding Layer: The embedding layer converts the text data into a numerical representation, such as word embeddings or character embeddings.
3. LSTM Layers: The LSTM layers process the embedded text, capturing the sequential information and understanding the context.
4. Fully Connected Layers: The fully connected layers process the LSTM outputs and make predictions for the predefined classes.
5. Output Layer: The output layer produces the final classification results.

The combination of CNN and LSTM models leverages the strengths of both architectures. The CNN model is effective at capturing local features and patterns in the text, while the LSTM model excels at capturing long-term dependencies and understanding the context. This combination results in improved accuracy and performance in text classification tasks. The concept being used in this project is text classification, where the goal is to assign predefined classes to free-text documents. Text classification is a common task in natural language processing (NLP) and has various applications, including sentiment analysis, spam detection, and topic classification.

Overall, the CNN and LSTM models used in this project provide a powerful approach for text classification, allowing for accurate and efficient classification of free-text documents.

Program:

Import packages

```
import requests
from bs4 import BeautifulSoup
import time
import pandas as pd
```

Loading the streaming datasets using api

```
# Scrape the latest reviews for product with given ASIN using
scraperapi
```

```
asin = 'B08N5NQ869'
soup_contents = []
for page in range(1,60):
  time.sleep(20) # avoids CAPTCHA
  url = f'https://www.amazon.com/product-
reviews/{asin}/ref=cm_cr_arp_d_viewopt_srt?ie=UTF8&reviewerType=all_rev
iews&sortBy=recent&pageNumber={page}'
  payload = {'api_key':'bd6c22692874b6bd87ff88737c5f3d6e',
          'url': url}
  response = requests.get('http://api.scraperapi.com', params =
payload)
  soup = BeautifulSoup(response.content, 'html.parser')
  soup_contents.append(soup)
```

```
# Check content of the html
soup = soup_contents[2]
soup.prettify
```

```
ratings = soup.find_all('div', attrs={'class' : "a-section celwidget"})
rating = ratings[0]

title = rating.find('a', attrs = {'data-hook':'review-
title'}).text.strip('\n')
body = rating.find('span', attrs = {'data-hook':'review-
body'}).text.strip('\n')
date = rating.find('span', attrs = {'data-hook':'review-date'}).text
try:
  verified = rating.find('span', attrs = {'class':'avp-badge'}).text
except:
  verified = 'Not verified'
star = float(rating.find('a', attrs = {'class':'a-link-
normal'}).text[:3])
try:
  votes = rating.find('span', attrs={'class':'review-votes'}).text
except:
  votes = 'no vote'

prod = rating.find('a', attrs = {'data-hook':'format-
strip'}).text.split(':')
color = prod[1].split('Config')[0].strip()
configuration = prod[-1].strip()

print(title)
print(body)
print(date)
print(verified)
print(star)
print(votes)
```

```
print(color)
print(configuration)
```

Output:

```
Love it.
Exactly what we were looking for.
Reviewed in the United States on April 26, 2023
Not verified
5.0
no vote
Venetian Bronze
Doorbell only
```

```
reviews_list = []
for page in soup_contents:
  # Extrack all user reviews for the page
  ratings = page.find_all('div', attrs={'class' : "a-section
celwidget"})

  # Iterate over each review and extract rewiew contents
  for rating in ratings:
    title = rating.find('a', attrs = {'data-hook':'review-
title'}).text.strip('\n')
    body = rating.find('span', attrs = {'data-hook':'review-
body'}).text.strip('\n')
    date = rating.find('span', attrs = {'data-hook':'review-
date'}).text
    try:
      verified = rating.find('span', attrs = {'data-hook':'avp-
badge'}).text
    except:
      verified = 'Not verified'
    star = rating.find('a', attrs = {'class':'a-link-normal'}).text
    try:
      votes = rating.find('span', attrs = {'data-hook':'helpful-vote-
statement'}).text
    except:
      votes = 'no vote'
    if rating.find('img').get('src'):
      img = 'Yes'
    else:
      img = 'No'

    prod = rating.find('a', attrs = {'data-hook':'format-
strip'}).text.split(':')
    color = prod[1].split('Config')[0].strip()
    configuration = prod[-1].strip()

    # Create a dictionary for the review
    rating_dict = {'title': title,
```

```python
                    'body': body,
                    'date': date,
                    'status': verified,
                    'votes': votes,
                    'contains_image': img,
                    'color':color,
                    'configuration':configuration,
                    'score': star,
                    }

    # Append the dictionary review object to the list
    reviews_list.append(rating_dict)

# Convert into dataframe
data = pd.DataFrame(reviews_list)
data['product'] = 'Ring Video Doorbell - 1080p HD video, improved motion detection, easy installation'
data['asin'] = 'B08N5NQ869'
# Save Data
data.to_csv('ring_video_doorbell.csv', index = False)
# Load and check data
data = pd.read_csv('ring_video_doorbell.csv')
data.head()
```

Output:

	title	body	date	status	votes	contains_image	color	configuration	score
0	Love it.	Exactly what we were looking for.	Reviewed in the United States on April 26, 2023	Verified Purchase	no vote	Yes	Venetian Bronze	Doorbell only	5.0 out of 5 stars
1	Son feels safer	Easy to install!	Reviewed in the United States on April 26, 2023	Verified Purchase	no vote	Yes	Venetian Bronze	Doorbell only	5.0 out of 5 stars
2	Great Device - Beware of Pre-Setup	I have for Amazon Echo Dot's in my home and pu...	Reviewed in the United States on April 25, 2023	Verified Purchase	no vote	Yes	Venetian Bronze	Doorbell only	4.0 out of 5 stars
3	Not bad	Should have gotten the battery one.	Reviewed in the United States on April 25, 2023	Verified Purchase	no vote	Yes	Venetian Bronze	Doorbell only	3.0 out of 5 stars
4	Door ring	Must keep charging batteries	Reviewed in the United States on April 25, 2023	Verified Purchase	no vote	Yes	Satin Nickel	Doorbell only	2.0 out of 5 stars

```python
data.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 520 entries, 0 to 519
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   title           520 non-null    object
 1   body            520 non-null    object
 2   date            520 non-null    object
 3   status          520 non-null    object
 4   votes           520 non-null    object
 5   contains_image  520 non-null    object
 6   color           520 non-null    object
 7   configuration   520 non-null    object
 8   score           520 non-null    object
 9   product         520 non-null    object
 10  asin            520 non-null    object
dtypes: object(11)
memory usage: 44.8+ KB
```

```
data['rating'] = data['score'].str[:3]
data['rating'] = pd.to_numeric(data['rating'], errors='coerce')
data = data.reindex(columns=['title', 'rating', 'body'])
data['title'] = data['title'].str.split('\n').str[1]
data['text'] = data['title'] + ' ' + data['body']
data.head()
```

Output:

	title	rating	body	text
0	Ring camera	1.0	I like the camera but it skips time and I hate...	Ring camera I like the camera but it skips tim...
1	Missing pieces	3.0	Note: I will come back and modify my review i...	Missing pieces Note: I will come back and mod...
2	Just what I was looking for	5.0	Love my ring camera. Easy to install and use. ...	Just what I was looking for Love my ring camer...
3	Love this thing	5.0	Wish I would have bought one of these years ag...	Love this thing Wish I would have bought one o...
4	Clear views!	5.0	Video is clear and accessible! Battery lasts a...	Clear views! Video is clear and accessible! Ba...

Model Creation - Deep Learning Approach

Convolutional Neural Network (CNN) model for text classification

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Conv1D,
GlobalMaxPooling1D, Dense
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split

# Sample data
reviews = data['text'].tolist()

labels = ['good product', 'worst product']

# Tokenize the text data
tokenizer = Tokenizer()
tokenizer.fit_on_texts(reviews)
sequences = tokenizer.texts_to_sequences(reviews)

# Pad sequences to have the same length
max_length = max([len(seq) for seq in sequences])
padded_sequences = pad_sequences(sequences, maxlen=max_length)

# Convert labels to numerical values
label_mapping = {'good product': 1, 'worst product': 0}
numeric_labels = np.array([label_mapping[label] for label in labels])

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(padded_sequences,
numeric_labels, test_size=0.2, random_state=42)
```

```python
# Build the CNN model
vocab_size = len(tokenizer.word_index) + 1
embedding_dim = 100

model = Sequential()
model.add(Embedding(vocab_size, embedding_dim,
input_length=max_length))
model.add(Conv1D(128, 5, activation='relu'))
model.add(GlobalMaxPooling1D())
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])


# Train the model
model.fit(np.array(X_train), np.array(y_train), epochs=10,
batch_size=16, validation_data=(np.array(X_test), np.array(y_test)))

# Make predictions
new_reviews = ['Excellent product!', 'Awful experience.']
new_sequences = tokenizer.texts_to_sequences(new_reviews)
new_padded_sequences = pad_sequences(new_sequences, maxlen=max_length)
predictions = model.predict(np.array(new_padded_sequences))

for i, review in enumerate(new_reviews):
    if predictions[i] > 0.5:
        print(f'Review: {review} - Category: good product')
    else:
        print(f'Review: {review} - Category: worst product')
```

```
Epoch 1/10
1/1 [==============================] - 3s 3s/step - loss: 0.6913 - accuracy: 0.6667 - val_loss: 0.6687 - val_accuracy: 1.0000
Epoch 2/10
1/1 [==============================] - 0s 45ms/step - loss: 0.6567 - accuracy: 0.6667 - val_loss: 0.6721 - val_accuracy: 1.0000
Epoch 3/10
1/1 [==============================] - 0s 59ms/step - loss: 0.6268 - accuracy: 1.0000 - val_loss: 0.6762 - val_accuracy: 1.0000
Epoch 4/10
1/1 [==============================] - 0s 64ms/step - loss: 0.6006 - accuracy: 1.0000 - val_loss: 0.6812 - val_accuracy: 1.0000
Epoch 5/10
1/1 [==============================] - 0s 45ms/step - loss: 0.5780 - accuracy: 1.0000 - val_loss: 0.6854 - val_accuracy: 1.0000
Epoch 6/10
1/1 [==============================] - 0s 64ms/step - loss: 0.5574 - accuracy: 1.0000 - val_loss: 0.6887 - val_accuracy: 1.0000
Epoch 7/10
1/1 [==============================] - 0s 62ms/step - loss: 0.5380 - accuracy: 1.0000 - val_loss: 0.6916 - val_accuracy: 1.0000
Epoch 8/10
1/1 [==============================] - 0s 42ms/step - loss: 0.5197 - accuracy: 1.0000 - val_loss: 0.6946 - val_accuracy: 0.0000e+00
Epoch 9/10
1/1 [==============================] - 0s 60ms/step - loss: 0.5020 - accuracy: 1.0000 - val_loss: 0.6967 - val_accuracy: 0.0000e+00
Epoch 10/10
1/1 [==============================] - 0s 47ms/step - loss: 0.4852 - accuracy: 1.0000 - val_loss: 0.6978 - val_accuracy: 0.0000e+00
1/1 [==============================] - 0s 106ms/step
Review: Excellent product! - Category: good product
Review: Awful experience. - Category: worst product
```

```
model.save('model.h5')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079: UserWarning: You are saving your model as an HDF5 file via `model.save()`.
  saving_api.save_model(
```

Evaluate the model

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split

# Sample data
reviews = data['text'].tolist()
labels = ['good product', 'worst product']
# Tokenize the text data
tokenizer = Tokenizer()
tokenizer.fit_on_texts(reviews)
sequences = tokenizer.texts_to_sequences(reviews)

# Pad the sequences to have the same length
max_length = max([len(seq) for seq in sequences])
padded_sequences = pad_sequences(sequences, maxlen=max_length)

# Convert labels to numerical values
label_mapping = {'good product': 1, 'worst product': 0}
numeric_labels = np.array([label_mapping[label] for label in labels])

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(padded_sequences,
numeric_labels, test_size=0.2, random_state=42)

# Build the LSTM model
vocab_size = len(tokenizer.word_index) + 1
embedding_dim = 100

model = Sequential()
model.add(Embedding(vocab_size, embedding_dim,
input_length=max_length))
model.add(LSTM(128))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
```

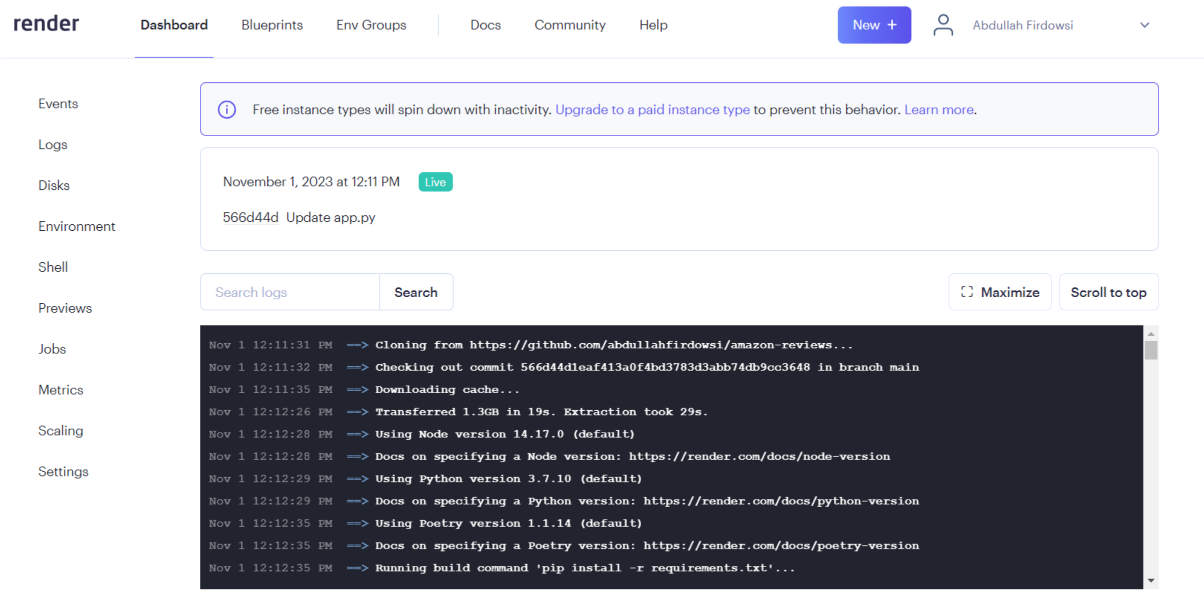
```python
# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=16,
validation_data=(X_test, y_test))

# Make predictions
new_reviews = ['Excellent product!', 'Awful experience.']
new_sequences = tokenizer.texts_to_sequences(new_reviews)
new_padded_sequences = pad_sequences(new_sequences, maxlen=max_length)
predictions = model.predict(new_padded_sequences)

for i, review in enumerate(new_reviews):
    if predictions[i] > 0.5:
        print(f'Review: {review} - Category: good product')
    else:
        print(f'Review: {review} - Category: worst product')
```

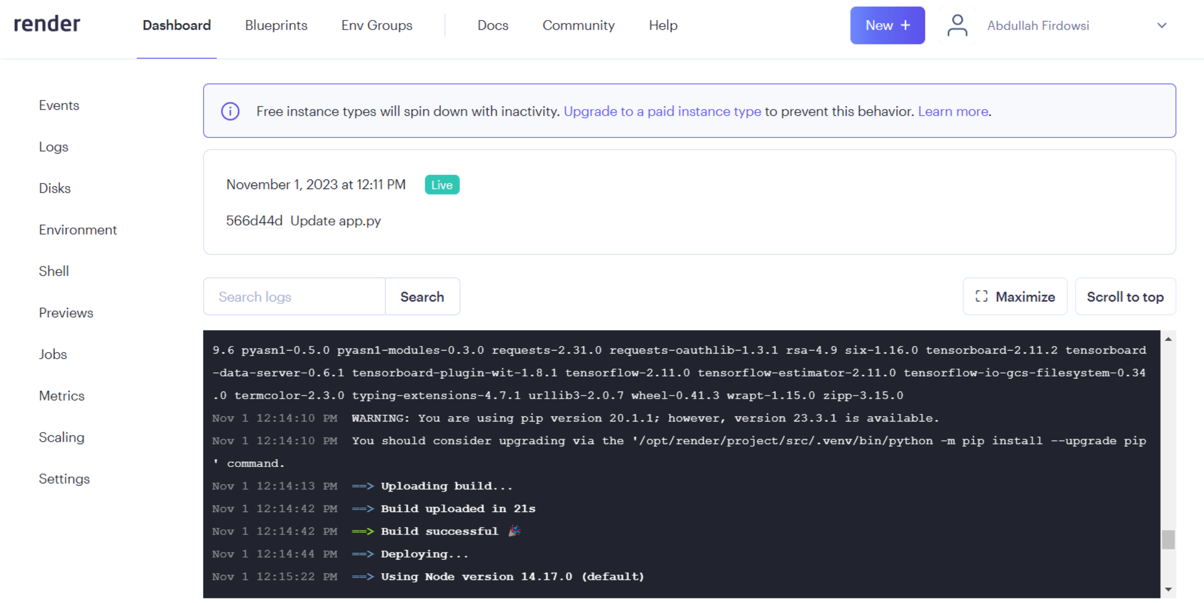
```
Review: Excellent product! - Category: good product
Review: Awful experience. - Category: worst product
```

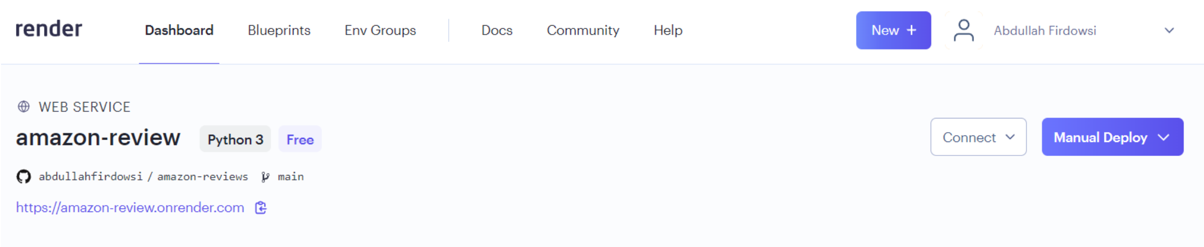
Deployment - app.py

```python
from flask import Flask, render_template, request
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import tokenizer_from_json

app = Flask(__name__)

# Load the trained model
model = load_model('model.h5')

# Load the tokenizer
with open('tokenizer.json', 'r') as f:
    tokenizer = tokenizer_from_json(f.read())

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    review = request.form['review']
    sequence = tokenizer.texts_to_sequences([review])
    padded_sequence = pad_sequences(sequence, maxlen=100)
    prediction = model.predict(padded_sequence)
    category = 'good product' if prediction > 0.5 else 'worst
product'
    return render_template('result.html', review=review,
category=category)

if __name__ == '__main__':
    app.run(debug=True)
```

Global Deployment using render:



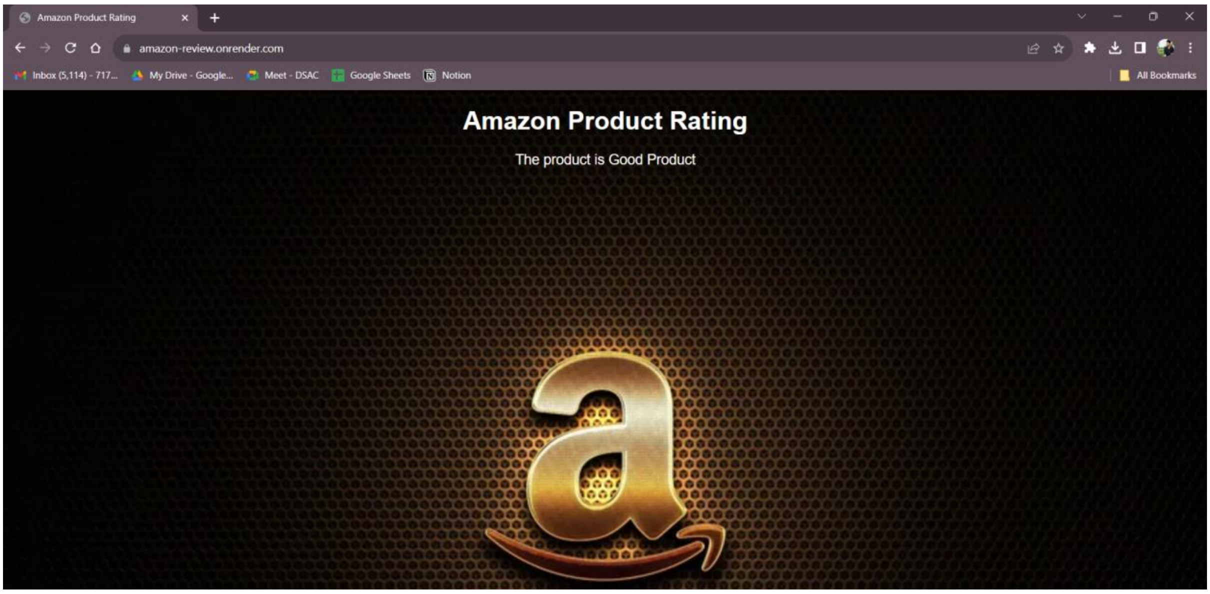




Webpage link: https://amazon-review.onrender.com

Github Repo: https://github.com/abdullahfirdowsi/amazon-review

SCREENSHOTS OF THE APPLICATION:



In results, The Convolutional Neural Network (CNN) model is a reliable and efficient tool for text classification, analysing customer reviews' sentiment and content. It extracts relevant features like sentiment polarity, review length, and keyword frequency, allowing real-time prediction and classification of unseen reviews. The model's high-dimensional processing and activation functions make it practical for e-commerce platforms like Amazon, providing valuable insights into product quality.