**CODE SCRIPT**

**1. Preprocessing code:**

# Install module, library related

%%capture

!pip install backoff openai -U

!pip install afinn

!pip install backoff emoji gdown openai -U

!gdown 1WsKe3RkGoRpCW8flaMmdYJK2AsP-SWT2

!pip install emot

import nltk

nltk.download('stopwords')

import re

**- Translate review data to english**

**- Using GPT3**

**+ For Product:**

openai.api\_key = ''

@backoff.on\_exception(backoff.expo, openai.error.RateLimitError)

def GPT3(prompt):

  response = openai.Completion.create(

      model = 'text-davinci-003',

      prompt = f'"{prompt}". Only extract the part related to product in this sentence if not exists return None',

      temperature = 0.7,

      max\_tokens = 512

  )

  return response

# PRODUCT

  # Separate context

product = []

for review in df['enreview'].tolist()[:]:

  product.append(GPT3(review))

  # Fill into product column in df

df['product']=''

df.iloc[:,-1] = product[:]

# DELIVERY

  # Separate context

delivery = []

for review in df['enreview'].tolist()[:]:

  delivery.append(GPT3(review))

**- Preprocessing Label**

# Import library

nltk.download('punkt')

import re

import spacy

def text\_cleaner(text):

    text = text.lower()

    text = text.strip()

    text = text.replace('\n', ' ')

    text = re.sub(r'[^a-zA-Z\s\']', '', text)

    text = nlp(text)

    text = " ".join([token.lemma\_ for token in doc])

    return text

 df['delivery']=df['delivery'].apply(text\_cleaner)

 df['product']= df['product'].apply(text\_cleaner)

 df['service']= df['service'].apply(text\_cleaner)

**- Using Afinn library to calculate sentiment**

from afinn import Afinn

afinn = Afinn()

def get\_sentiment(text):

    if text == 'none':

      return 0

    else:

      sentiment\_score = afinn.score(text)

      if sentiment\_score <0:

        return -1

      return 1

df['delivery\_sent']= df['delivery'].apply(get\_sentiment)

df['product\_sent']= df['product'].apply(get\_sentiment)

df['service\_sent']= df['service'].apply(get\_sentiment)

df.head(10)

**- Encoding Data**

#Encoding Data

def transfer(review):

    if review == 'none':

      return 0

    return 1

dfmodel['delivery'] = dfmodel['delivery'].apply(transfer)

dfmodel['service']= dfmodel['service'].apply(transfer)

dfmodel['product']= dfmodel['product'].apply(transfer)

**- Cleaning review column**

# Preprocessing Review column

import emoji

import pandas as pd

from emot.emo\_unicode import UNICODE\_EMOJI # For emojis

from emot.emo\_unicode import EMOTICONS\_EMO # For EMOTICONS

from nltk.corpus import stopwords

# from textblob import TextBlob

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

import spacy

nlp = spacy.load("en\_core\_web\_sm")

stop\_words = set(stopwords.words('english'))

def clean\_review(text):

    text = text.lower()

    text = re.sub(r'[^a-zA-Z\s]', '', text)

    text = text.strip()

    text = text.replace('\n', ' ')

    text = re.sub(r'[^\w\s]+|(\d+)', '', text)

    #tokenizing

    # lemmatization

    doc = nlp(text)

    text = " ".join([token.lemma\_ for token in doc])

    return text

dfmodel['enreview'] = dfmodel['enreview'].apply(clean\_review)

**2 The files for render to main file (app.py) which were built: (Building models and WebApp)**

Delivery.py:

import pandas as pd

from sklearn.svm import SVC

from sklearn.feature\_extraction.text import CountVectorizer

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

def delivery():

    dfmodel = pd.read\_csv('E:/STUDENTS-UNIVERSITY/FRESHER\_TERM\_2/CAPSTONE PJ 2/data/df\_sample1.csv')

    X = dfmodel['enreview']

    y = dfmodel['delivery']

    # y2 = dfmodel['delivery\_sent']

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.3,random\_state=188)

    cv\_delivery = CountVectorizer()

    X\_train = cv\_delivery.fit\_transform(X\_train)

    X\_test = cv\_delivery.transform(X\_test)

    classifier\_delivery = SVC(kernel = 'linear', random\_state = 0)

    classifier\_delivery.fit(X\_train, y\_train)

    return classifier\_delivery,cv\_delivery

Delivery\_sentiment.py:

import pandas as pd

from sklearn.svm import SVC

from sklearn.feature\_extraction.text import CountVectorizer

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

def delivery\_sentiment():

    dfmodel = pd.read\_csv('E:/STUDENTS-UNIVERSITY/FRESHER\_TERM\_2/CAPSTONE PJ 2/data/df\_sample1.csv')

    X = dfmodel['enreview']

    # y = dfmodel['delivery']

    y2 = dfmodel['delivery\_sent']

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y2, test\_size = 0.3,random\_state=188)

    cv\_delivery\_sent = CountVectorizer()

    X\_train = cv\_delivery\_sent.fit\_transform(X\_train)

    X\_test = cv\_delivery\_sent.transform(X\_test)

    classifier\_delivery\_sent = SVC(kernel = 'linear', random\_state = 0)

    classifier\_delivery\_sent.fit(X\_train, y\_train)

    return classifier\_delivery\_sent,cv\_delivery\_sent

product.py:

import pandas as pd

from sklearn.svm import SVC

from sklearn.feature\_extraction.text import CountVectorizer

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

def product():

    dfmodel = pd.read\_csv('E:/STUDENTS-UNIVERSITY/FRESHER\_TERM\_2/CAPSTONE PJ 2/data/df\_sample1.csv')

    X = dfmodel['enreview']

    y = dfmodel['product']

    # y2 = dfmodel['delivery\_sent']

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.3,random\_state=188)

    cv\_product = CountVectorizer()

    X\_train = cv\_product.fit\_transform(X\_train)

    X\_test = cv\_product.transform(X\_test)

    classifier\_product= SVC(kernel = 'linear', random\_state = 0)

    classifier\_product.fit(X\_train, y\_train)

    return classifier\_product,cv\_product

product\_sentiment.py:

import pandas as pd

from sklearn.svm import SVC

from sklearn.feature\_extraction.text import CountVectorizer

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

def product\_sentiment():

    dfmodel = pd.read\_csv('E:/STUDENTS-UNIVERSITY/FRESHER\_TERM\_2/CAPSTONE PJ 2/data/df\_sample1.csv')

    X = dfmodel['enreview']

    y2 = dfmodel['product\_sent']

    # y2 = dfmodel['delivery\_sent']

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y2, test\_size = 0.3,random\_state=188)

    cv\_product\_sent = CountVectorizer()

    X\_train = cv\_product\_sent.fit\_transform(X\_train)

    X\_test = cv\_product\_sent.transform(X\_test)

    classifier\_product\_sent = SVC(kernel = 'linear', random\_state = 0)

    classifier\_product\_sent.fit(X\_train, y\_train)

    return classifier\_product\_sent,cv\_product\_sent

service.py:

import pandas as pd

from sklearn.svm import SVC

from sklearn.feature\_extraction.text import CountVectorizer

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

def service():

    dfmodel = pd.read\_csv('E:/STUDENTS-UNIVERSITY/FRESHER\_TERM\_2/CAPSTONE PJ 2/data/df\_sample1.csv')

    X = dfmodel['enreview']

    y = dfmodel['service']

    # y2 = dfmodel['delivery\_sent']

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.3,random\_state=188)

    cv\_service = CountVectorizer()

    X\_train = cv\_service.fit\_transform(X\_train)

    X\_test = cv\_service.transform(X\_test)

    classifier\_service = SVC(kernel = 'linear', random\_state = 0)

    classifier\_service.fit(X\_train, y\_train)

    return classifier\_service,cv\_service

service\_sentiment.py:

import pandas as pd

from sklearn.svm import SVC

from sklearn.feature\_extraction.text import CountVectorizer

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

def service\_sentiment():

    dfmodel = pd.read\_csv('E:/STUDENTS-UNIVERSITY/FRESHER\_TERM\_2/CAPSTONE PJ 2/data/df\_sample1.csv')

    X = dfmodel['enreview']

    # y = dfmodel['delivery']

    y2 = dfmodel['service\_sent']

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y2, test\_size = 0.3,random\_state=188)

    cv\_service\_sent = CountVectorizer()

    X\_train = cv\_service\_sent.fit\_transform(X\_train)

    X\_test = cv\_service\_sent.transform(X\_test)

    classifier\_service\_sent = SVC(kernel = 'linear', random\_state = 0)

    classifier\_service\_sent.fit(X\_train, y\_train)

    return classifier\_service\_sent,cv\_service\_sent

File app.py which is gathering models built:

import pandas as pd

import re

from flask import Flask, request, render\_template

from flask import make\_response

import spacy

from nltk.corpus import stopwords

from delivery import delivery

from delivery\_sentiment import delivery\_sentiment

from product import product

from product\_sentiment import product\_sentiment

from service import service

from service\_sentiment import service\_sentiment

nlp = spacy.load("en\_core\_web\_sm")

stop\_words = set(stopwords.words('english'))

def clean\_text(text):

    text = text.lower()

    text = re.sub(r'[^a-zA-Z\s]', '', text)

    text = text.strip()

    text = text.replace('\n', ' ')

    text = re.sub(r'[^\w\s]+|(\d+)', '', text)

    doc = nlp(text)

    text = " ".join([token.lemma\_ for token in doc])

    return text

app = Flask(\_\_name\_\_, template\_folder='templates')

classifier\_delivery, cv\_delivery = delivery()

classifier\_delivery\_sent, cv\_delivery\_sent = delivery\_sentiment()

classifier\_product, cv\_product = product()

classifier\_product\_sent, cv\_product\_sent = product\_sentiment()

classifier\_service, cv\_service = service()

classifier\_service\_sent, cv\_service\_sent = service\_sentiment()

@app.route('/')

def home():

    return render\_template('index.html')

@app.route('/predict', methods=['POST'])

def predict():

    file = request.files['file']

    df = pd.read\_csv(file)

    # df['enreview'] = df['enreview'].apply(clean\_text)

    delivery\_predictions = classifier\_delivery.predict(cv\_delivery.transform(df['enreview']))

    delivery\_sentiment\_predictions = classifier\_delivery\_sent.predict(cv\_delivery\_sent.transform(df['enreview']))

    product\_predictions = classifier\_product.predict(cv\_product.transform(df['enreview']))

    product\_sentiment\_predictions = classifier\_product\_sent.predict(cv\_product\_sent.transform(df['enreview']))

    service\_predictions = classifier\_service.predict(cv\_service.transform(df['enreview']))

    service\_sentiment\_predictions = classifier\_service\_sent.predict(cv\_service\_sent.transform(df['enreview']))

    result\_df = pd.DataFrame({

        'enreview': df['enreview'],

        'delivery': delivery\_predictions,

        'delivery\_sentiment': delivery\_sentiment\_predictions,

        'product': product\_predictions,

        'product\_sentiment': product\_sentiment\_predictions,

        'service': service\_predictions,

        'service\_sentiment': service\_sentiment\_predictions

    })

    return render\_template('result.html', result=result\_df)

@app.route('/download', methods=['POST'])

def download():

    csv\_data = request.form['data']

    response = make\_response(csv\_data)

    response.headers['Content-Disposition'] = 'attachment; filename=result.csv'

    response.headers['Content-type'] = 'text/csv'

    return response

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(port=3000, debug=True)

Templates which use for design rest API:

Index.html:

<!DOCTYPE html>

<html>

  <head>

    <title>home</title>

    <style>

      body {

        font-family: Arial, sans-serif;

        margin: 0;

        padding: 0;

        background-image: url("https://images.unsplash.com/photo-1578916171728-46686eac8d58?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=1374&q=80"); /\* Replace "your-image-url.jpg" with the URL or path to your desired background image \*/

        background-size: cover;

        background-position: center;

        margin-top: 100px;

      }

      .jumbotron {

        background-color: rgba(248, 249, 250, 0.8);

        padding: 20px;

        border-radius: 5px;

        margin-bottom: 20px;

        width: 100%;

        margin: 0 auto;

      }

      .container {

        display: flex;

        flex-direction: column;

        justify-content: center;

        align-items: center;

      }

      h1 {

        text-align: center;

        color: #45a049;

        margin-bottom: 30px;

        font-size: 36px;

      }

      .form-row {

        display: flex;

        justify-content: center;

        align-items: center;

        margin-bottom: 20px;

      }

      .file-upload {

        position: relative;

        overflow: hidden;

        display: inline-block;

        vertical-align: middle;

        margin-right: 10px;

      }

      .file-upload input[type="file"] {

        position: absolute;

        left: 0;

        top: 0;

        opacity: 0;

      }

      .file-upload label {

        display: inline-block;

        padding: 10px 20px;

        background-color: #4caf50;

        color: #fff;

        border-radius: 5px;

        cursor: pointer;

      }

      .file-upload label:hover {

        background-color: #45a049;

      }

      input[type="submit"] {

        display: block;

        padding: 10px 20px;

        font-size: 16px;

        background-color: #4caf50;

        color: #fff;

        border: none;

        border-radius: 5px;

        cursor: pointer;

        margin: 0 auto;

      }

      input[type="submit"]:hover {

        background-color: #45a049;

      }

    </style>

  </head>

  <body>

    <div class="jumbotron jumbotron-fluid">

      <div class="container">

        <h1 class="display-4">Getting response for your data</h1>

        <form action="/predict" method="post" enctype="multipart/form-data">

          <div class="form-row">

            <div class="file-upload">

              <label for="file">Choose File</label>

              <input type="file" name="file" id="file" accept=".csv" required>

            </div>

            <input type="submit" value="Process">

          </div>

        </form>

      </div>

    </div>

  </body>

</html>

Result.html:

<!DOCTYPE html>

<html>

  <head>

    <title>result</title>

    <style>

      body {

        font-family: Arial, sans-serif;

        margin: 0;

        padding: 0;

        background-color: #f2f2f2;

      }

      h1 {

        text-align: center;

        color: #333;

        margin-top: 20px;

      }

      table {

        border-collapse: collapse;

        width: 100%;

        margin-top: 20px;

      }

      th,

      td {

        text-align: left;

        padding: 10px;

      }

      tr:nth-child(even) {

        background-color: #ffffff;

      }

      th {

        background-color: #4caf50;

        color: white;

      }

      form {

        text-align: center;

        margin-top: 20px;

      }

      button {

        padding: 10px 20px;

        font-size: 16px;

        background-color: #4caf50;

        color: #fff;

        border: none;

        border-radius: 5px;

        cursor: pointer;

      }

      button:hover {

        background-color: #45a049;

      }

    </style>

  </head>

  <body>

    <h1>SECTIONS WITH SENTIMENTS</h1>

    <form>

      <label for="section">Select a Section:</label>

      <select id="section" onchange="filterResults()">

        <option value="all">All</option>

        <option value="delivery">Delivery</option>

        <option value="product">Product</option>

        <option value="service">Service</option>

      </select>

    </form>

    <table id="resultTable">

      <thead>

        <tr>

          <th>Review</th>

          <th>Delivery</th>

          <th>Delivery Sentiment</th>

          <th>Product</th>

          <th>Product Sentiment</th>

          <th>Service</th>

          <th>Service Sentiment</th>

        </tr>

      </thead>

      <tbody>

        {% for index, row in result.iterrows() %}

        <tr>

          <td>{{ row['enreview'] }}</td>

          <td>{% if row['delivery\_sentiment'] != 0 %}delivery{% else %}none{% endif %}</td>

          <td>{{ row['delivery\_sentiment'] }}</td>

          <td>{% if row['product\_sentiment'] != 0 %}product{% else %}none{% endif %}</td>

          <td>{{ row['product\_sentiment'] }}</td>

          <td>{% if row['service\_sentiment'] != 0 %}service{% else %}none{% endif %}</td>

          <td>{{ row['service\_sentiment'] }}</td>

        </tr>

        {% endfor %}

      </tbody>

    </table>

    <form action="/download" method="POST">

      <input type="hidden" name="data" value="{{ result.to\_csv(index=False) }}">

      <button type="submit">Download CSV</button>

    </form>

    <script>

      function filterResults() {

        var section = document.getElementById("section").value;

        var table = document.getElementById("resultTable");

        var rows = table.getElementsByTagName("tr");

        for (var i = 1; i < rows.length; i++) {

          var row = rows[i];

          var delivery = row.cells[1].innerHTML;

          var product = row.cells[3].innerHTML;

          var service = row.cells[5].innerHTML;

          if (

            section === "all" || // Add this condition for the "All" option

            (section === "delivery" && delivery !== "none") ||

            (section === "product" && product !== "none") ||

            (section === "service" && service !== "none")

          ) {

            row.style.display = "";

          } else {

            row.style.display = "none";

          }

          if (section === "delivery") {

            if (delivery !== "none") {

              row.style.display = "";

            } else {

              row.style.display = "none";

            }

          } else if (section === "product") {

            if (product !== "none") {

              row.style.display = "";

            } else {

              row.style.display = "none";

            }

          } else if (section === "service") {

            if (service !== "none") {

              row.style.display = "";

            } else {

              row.style.display = "none";

            }

          }

        }

      }

    </script>