Data Glacier

Your Deep Learning Partner

Deployment on Flask

Sentiment Analysis for Movie Reviews

Bora Engin Deniz

01.08.2023

Agenda

Data Glacier

Your Deep Learning Partner

- The Problem of the Week
- Information about Dataset
- Preprocess steps and Clear Data
- Learning Part and Results
- Frontend Part
- Flask Deployment and Connecting the Pieces
- Results and Testing
- End

Problem

Data Glacier

Your Deep Learning Partner

Task:

1. Select any toy data (simple data).

2. Save the model

3. Deploy the model on flask (web app)

4. Create pdf document (Name, Batch code, Submission date, Submitted to) which should contain snapshot of each step of deployment)

5. Upload the document to Github

6. Submit the URL of the uploaded document.

Dataset

Data Glacier

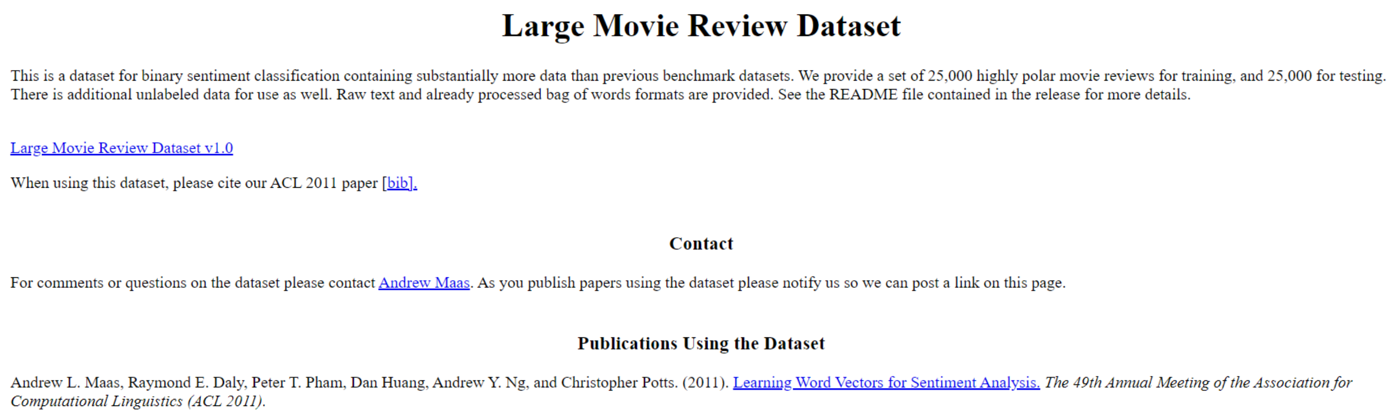
Your Deep Learning Partner

- I found a dataset that contains movie reviews from IMDB size of 50000.
- The great thing about that data is that in every 2-3 data , the comment of the film is changing so that there is no repeating or greater intensity of emotion in one direction.
- Although my dataset is very large, I preferred not to use most of it because I wanted the Flask interface to run faster. I can say that this situation still did not cause me any loss in terms of performance.
- I just used the entire dataset while working on Notebook, it allowed me to better evaluate the data when using my own test inputs.

Dataset

Data Glacier

Your Deep Learning Partner



Large Movie Review Dataset

This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. We provide a set of 25,000 highly polar movie reviews for training, and 25,000 for testing. There is additional unlabeled data for use as well. Raw text and already processed bag of words formats are provided. See the README file contained in the release for more details.

Large Movie Review Dataset v1.0

When using this dataset, please cite our ACL 2011 paper [bib].

Contact

For comments or questions on the dataset please contact Andrew Maas. As you publish papers using the dataset please notify us so we can post a link on this page.

Publications Using the Dataset

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).*

Pic 1: Website of the dataset that I used.

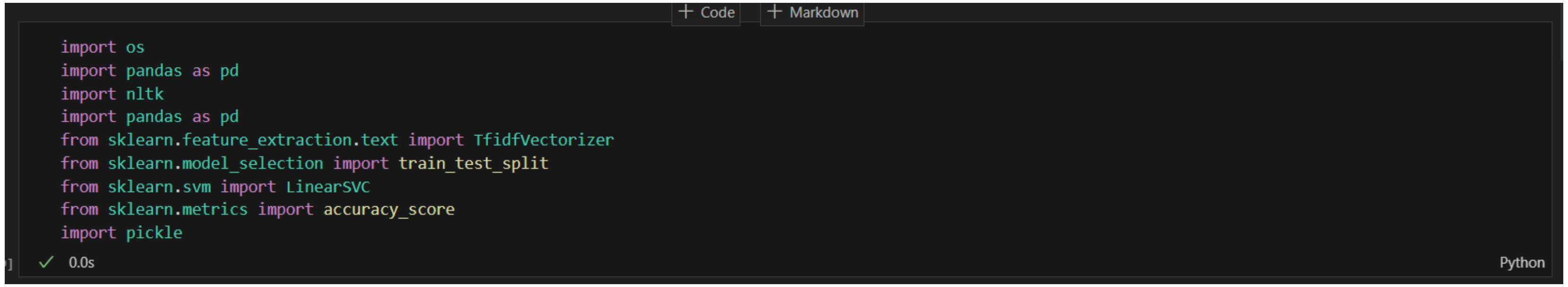
https://ai.stanford.edu/~amaas/data/sentiment/

I obtained dataset from here.

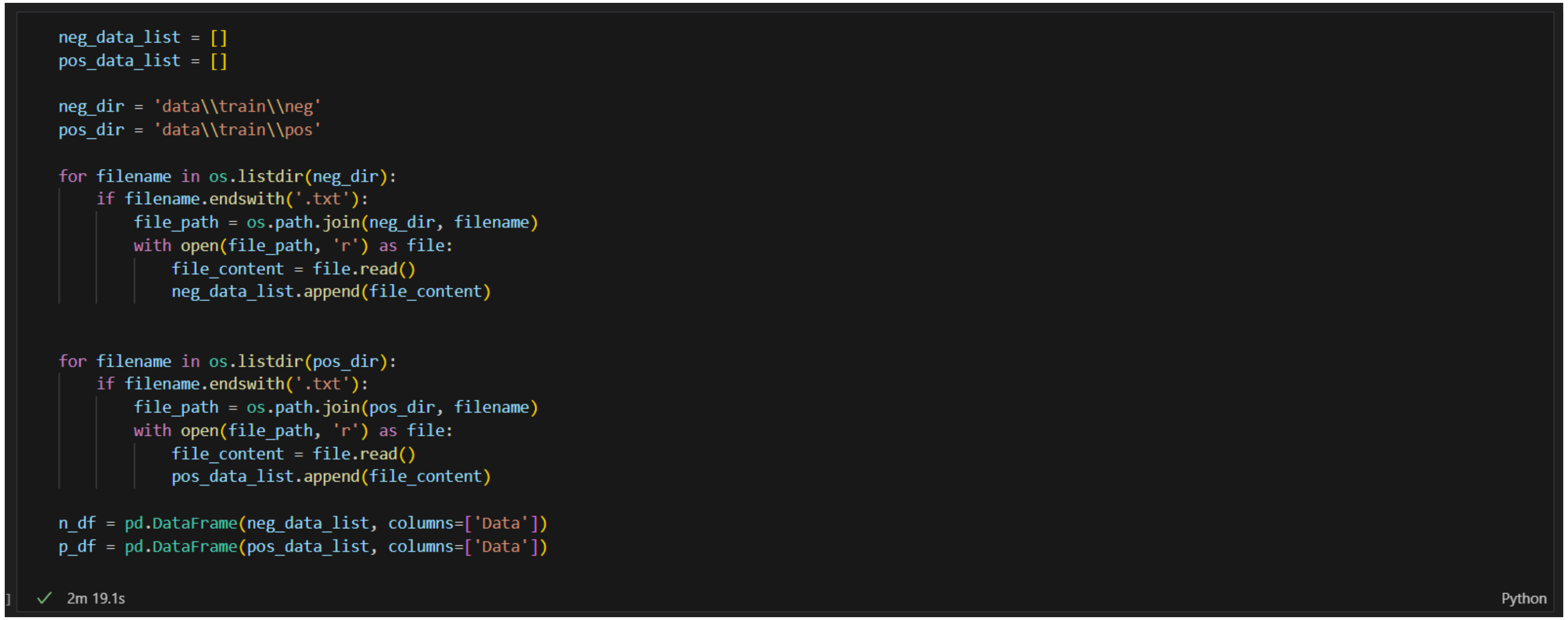
Read Data and Preprocess

Data Glacier

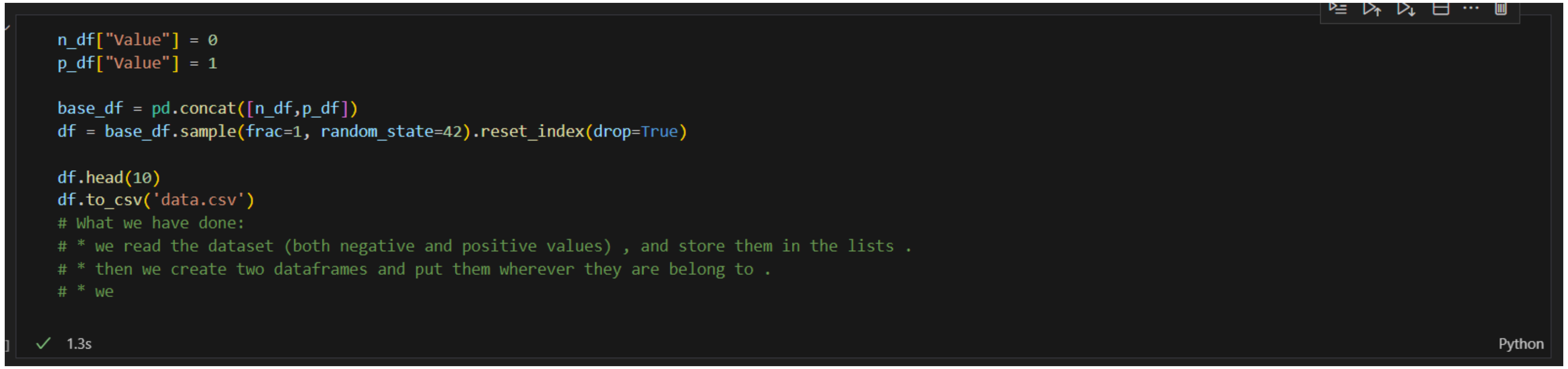
Your Deep Learning Partner



```python
import os
import pandas as pd
import nltk
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score
import pickle
```



```python
neg_data_list = []
pos_data_list = []

neg_dir = 'data\\train\\neg'
pos_dir = 'data\\train\\pos'

for filename in os.listdir(neg_dir):
    if filename.endswith('.txt'):
        file_path = os.path.join(neg_dir, filename)
        with open(file_path, 'r') as file:
            file_content = file.read()
            neg_data_list.append(file_content)


for filename in os.listdir(pos_dir):
    if filename.endswith('.txt'):
        file_path = os.path.join(pos_dir, filename)
        with open(file_path, 'r') as file:
            file_content = file.read()
            pos_data_list.append(file_content)

n_df = pd.DataFrame(neg_data_list, columns=['Data'])
p_df = pd.DataFrame(pos_data_list, columns=['Data'])
```



```python
n_df["Value"] = 0
p_df["Value"] = 1

base_df = pd.concat([n_df,p_df])
df = base_df.sample(frac=1, random_state=42).reset_index(drop=True)

df.head(10)
df.to_csv('data.csv')
# What we have done:
# * we read the dataset (both negative and positive values) , and store them in the lists .
# * then we create two dataframes and put them wherever they are belong to .
# * we
```

Read Data and Preprocess

- Since each of my data is in a separate .txt file, I read this data with a "for loop".
- Negative and Positive inputs were in different directories , so that firstly I put them in separate dataframes , and I created another column for that dataframes named "Value".
- I filled this column with 1 for Positives and 0 for negatives , then I concatenate these dataframes and shuffle them to make this dataframe more evenly distributed.

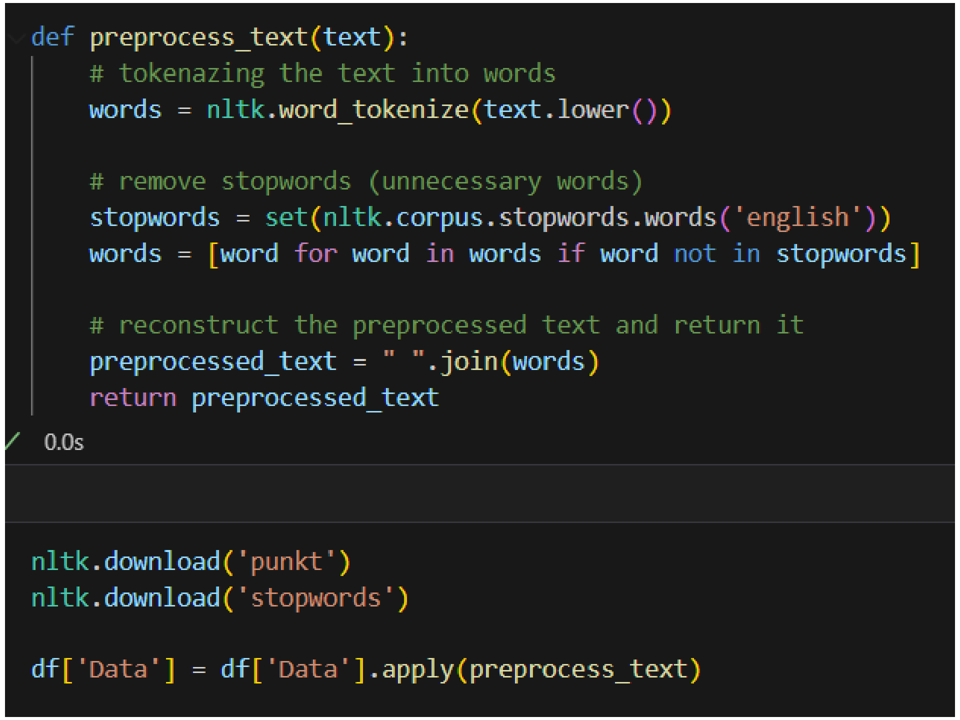
Data Glacier

Your Deep Learning Partner

Preprocess

Data Glacier

Your Deep Learning Partner



```python
def preprocess_text(text):
    # tokenazing the text into words
    words = nltk.word_tokenize(text.lower())

    # remove stopwords (unnecessary words)
    stopwords = set(nltk.corpus.stopwords.words('english'))
    words = [word for word in words if word not in stopwords]

    # reconstruct the preprocessed text and return it
    preprocessed_text = " ".join(words)
    return preprocessed_text
```

```
✓  0.0s
```

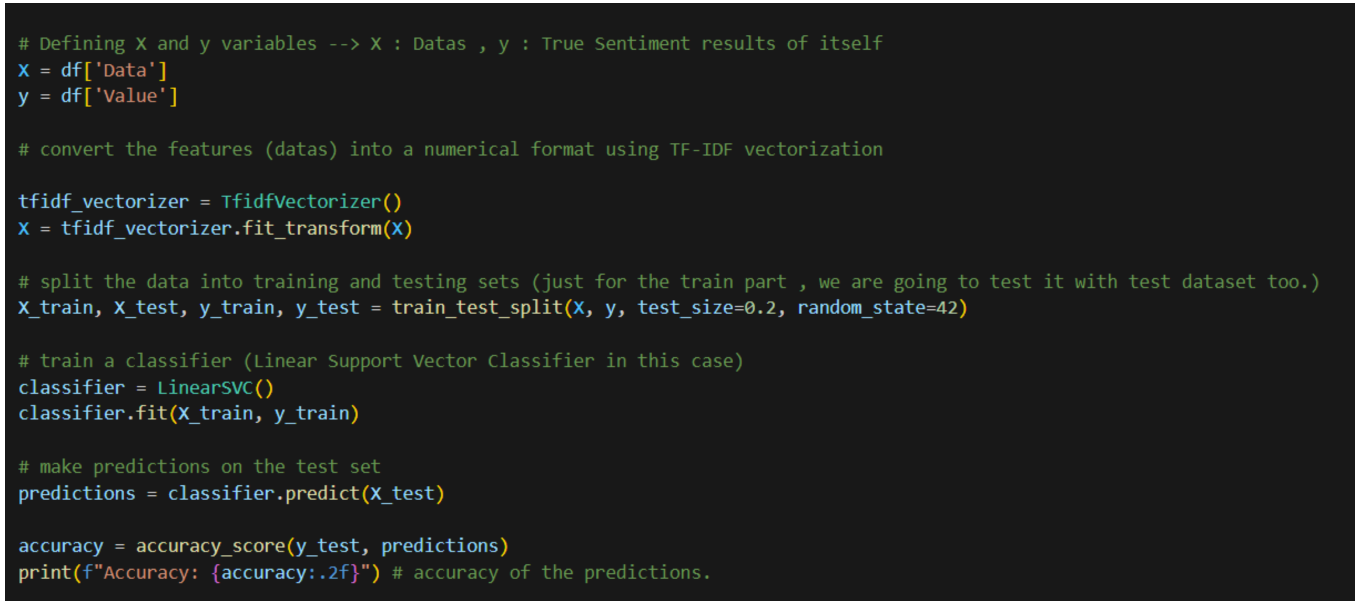
```python
nltk.download('punkt')
nltk.download('stopwords')

df['Data'] = df['Data'].apply(preprocess_text)
```

- With the function I used here, I made the letters in each data lowercase and removed the words that would not be used during the analysis from the sentence, so they became ready for use.

Model

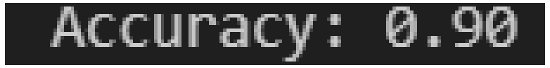
Data Glacier

Your Deep Learning Partner



```
# Defining X and y variables --> X : Datas , y : True Sentiment results of itself
X = df['Data']
y = df['Value']

# convert the features (datas) into a numerical format using TF-IDF vectorization

tfidf_vectorizer = TfidfVectorizer()
X = tfidf_vectorizer.fit_transform(X)

# split the data into training and testing sets (just for the train part , we are going to test it with test dataset too.)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# train a classifier (Linear Support Vector Classifier in this case)
classifier = LinearSVC()
classifier.fit(X_train, y_train)

# make predictions on the test set
predictions = classifier.predict(X_test)

accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy:.2f}") # accuracy of the predictions.
```

- Defining X and y -> TfidfVectorizer -> Train and test definiton -> Classifying and fitting train part -> prediction based on Test -> getting Acuuracy

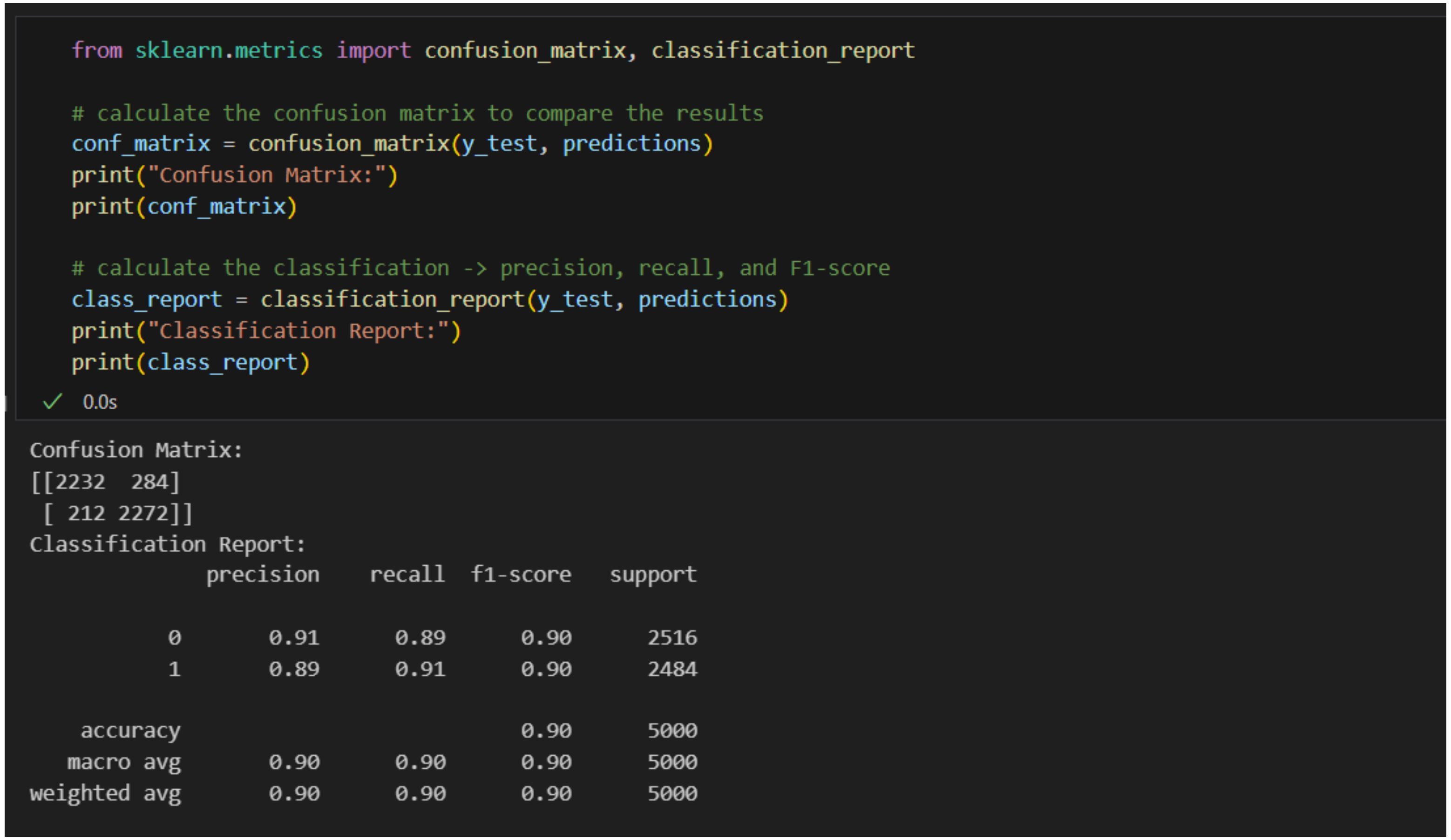


```
Accuracy: 0.90
```

Model

Data Glacier

Your Deep Learning Partner



```
from sklearn.metrics import confusion_matrix, classification_report

# calculate the confusion matrix to compare the results
conf_matrix = confusion_matrix(y_test, predictions)
print("Confusion Matrix:")
print(conf_matrix)

# calculate the classification -> precision, recall, and F1-score
class_report = classification_report(y_test, predictions)
print("Classification Report:")
print(class_report)
```

✓ 0.0s

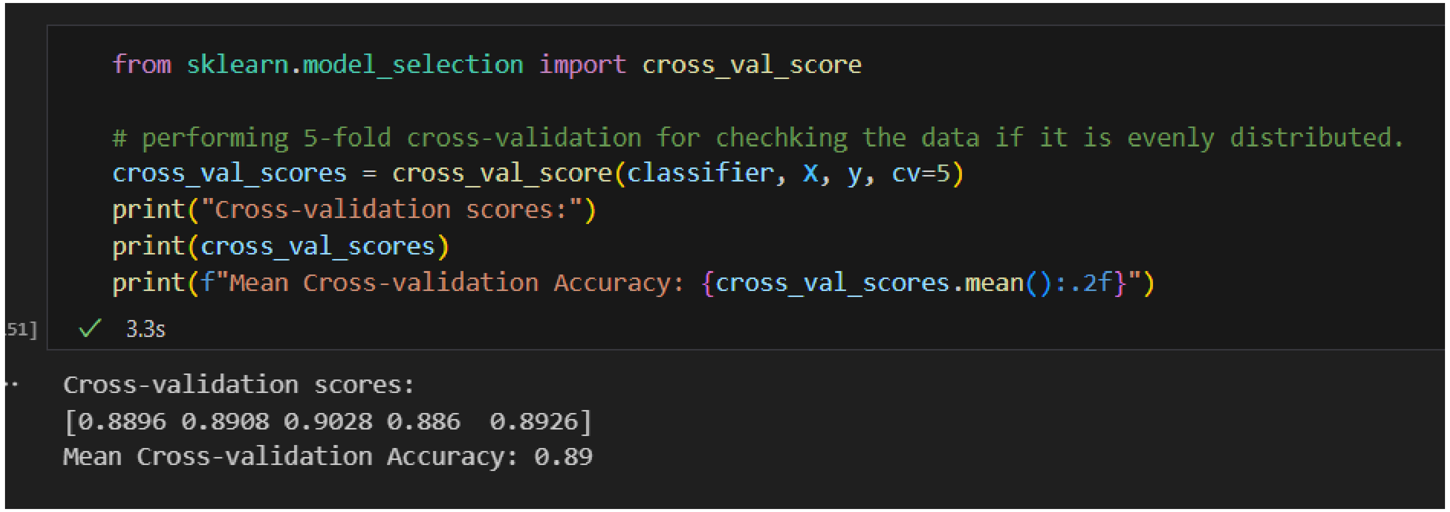
```
Confusion Matrix:
[[2232  284]
 [ 212 2272]]
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.89      0.90      2516
           1       0.89      0.91      0.90      2484

    accuracy                           0.90      5000
   macro avg       0.90      0.90      0.90      5000
weighted avg       0.90      0.90      0.90      5000
```

- In general, our prediction program produces 90% accurate predictions.

Model

Data Glacier

Your Deep Learning Partner

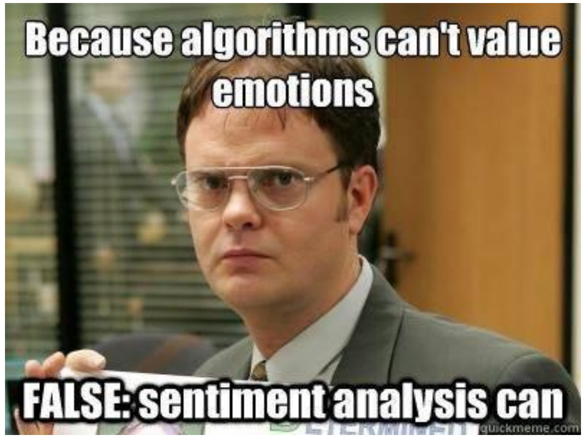


```
from sklearn.model_selection import cross_val_score

# performing 5-fold cross-validation for chechking the data if it is evenly distributed.
cross_val_scores = cross_val_score(classifier, X, y, cv=5)
print("Cross-validation scores:")
print(cross_val_scores)
print(f"Mean Cross-validation Accuracy: {cross_val_scores.mean():.2f}")
```

51] ✓ 3.3s

```
Cross-validation scores:
[0.8896 0.8908 0.9028 0.886  0.8926]
Mean Cross-validation Accuracy: 0.89
```

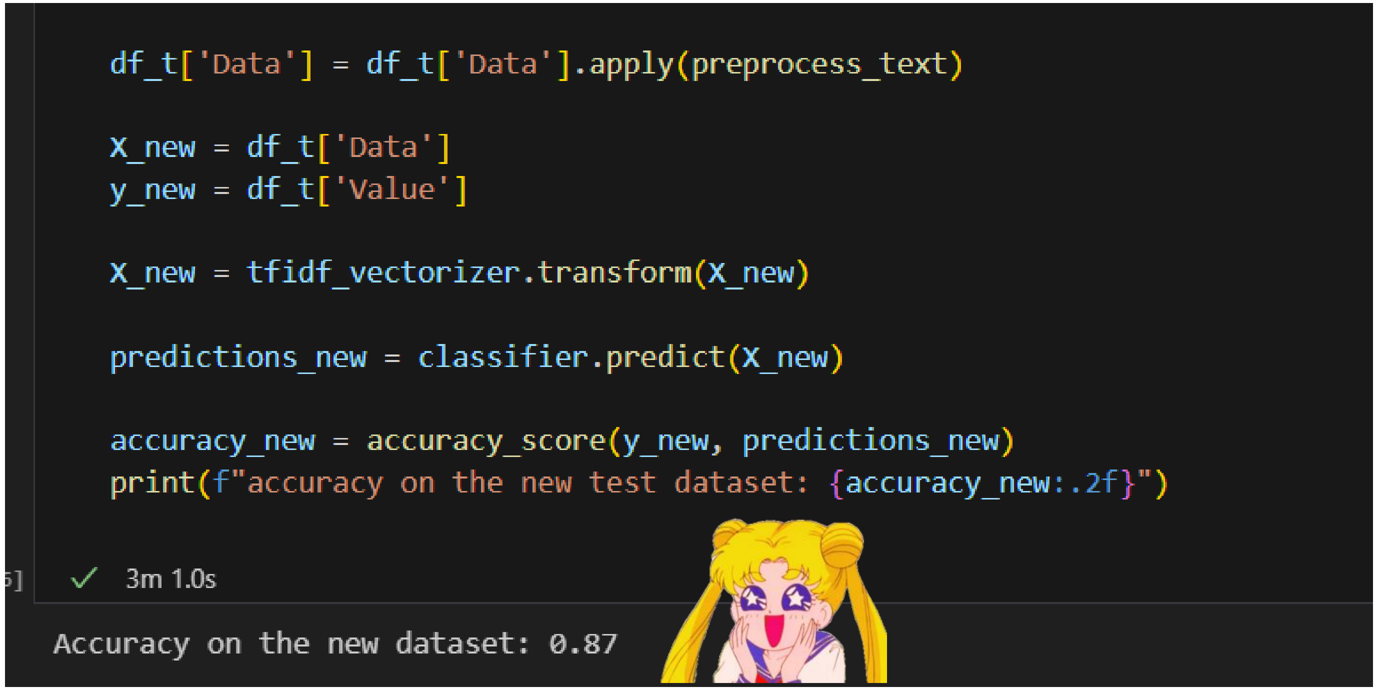
- We can say that data is evenly distributed , our data did not reveal any abnormal results in any of the ranges.


Because algorithms can't value emotions
FALSE: sentiment analysis can
quickmeme.com

Programmers looking at programming memes
Ah, humor based on my pain.

Test the Model with Test Dataset

- I also obtained test data from the dataset that I used so I wanted to use it to see result for the different inputs.
- I applied same steps of train dataset to test dataset like importing , preprocessing and testing .



```
df_t['Data'] = df_t['Data'].apply(preprocess_text)

X_new = df_t['Data']
y_new = df_t['Value']

X_new = tfidf_vectorizer.transform(X_new)

predictions_new = classifier.predict(X_new)

accuracy_new = accuracy_score(y_new, predictions_new)
print(f"accuracy on the new test dataset: {accuracy_new:.2f}")
```

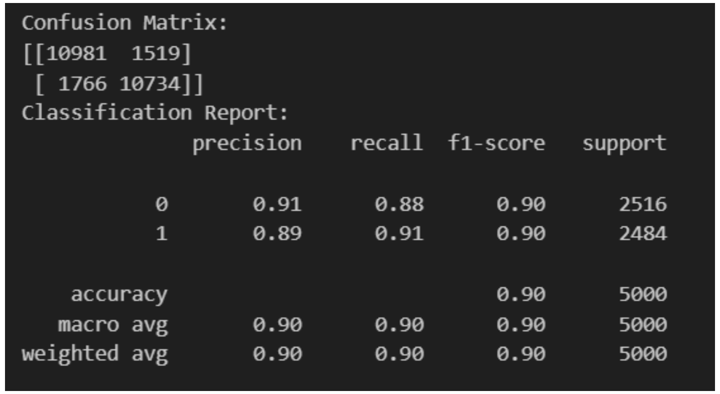
```
✓ 3m 1.0s

Accuracy on the new dataset: 0.87
```

Data Glacier

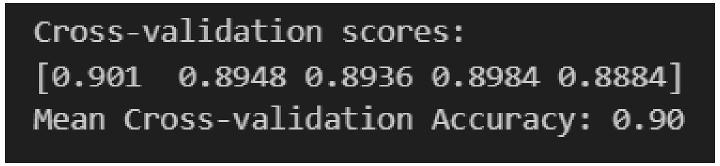
Your Deep Learning Partner

Test the Model with Test Dataset

I also applied the analysis part to test set and here is the results :



```
Confusion Matrix:
[[10981  1519]
 [ 1766 10734]]
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.88      0.90      2516
           1       0.89      0.91      0.90      2484

    accuracy                           0.90      5000
   macro avg       0.90      0.90      0.90      5000
weighted avg       0.90      0.90      0.90      5000
```



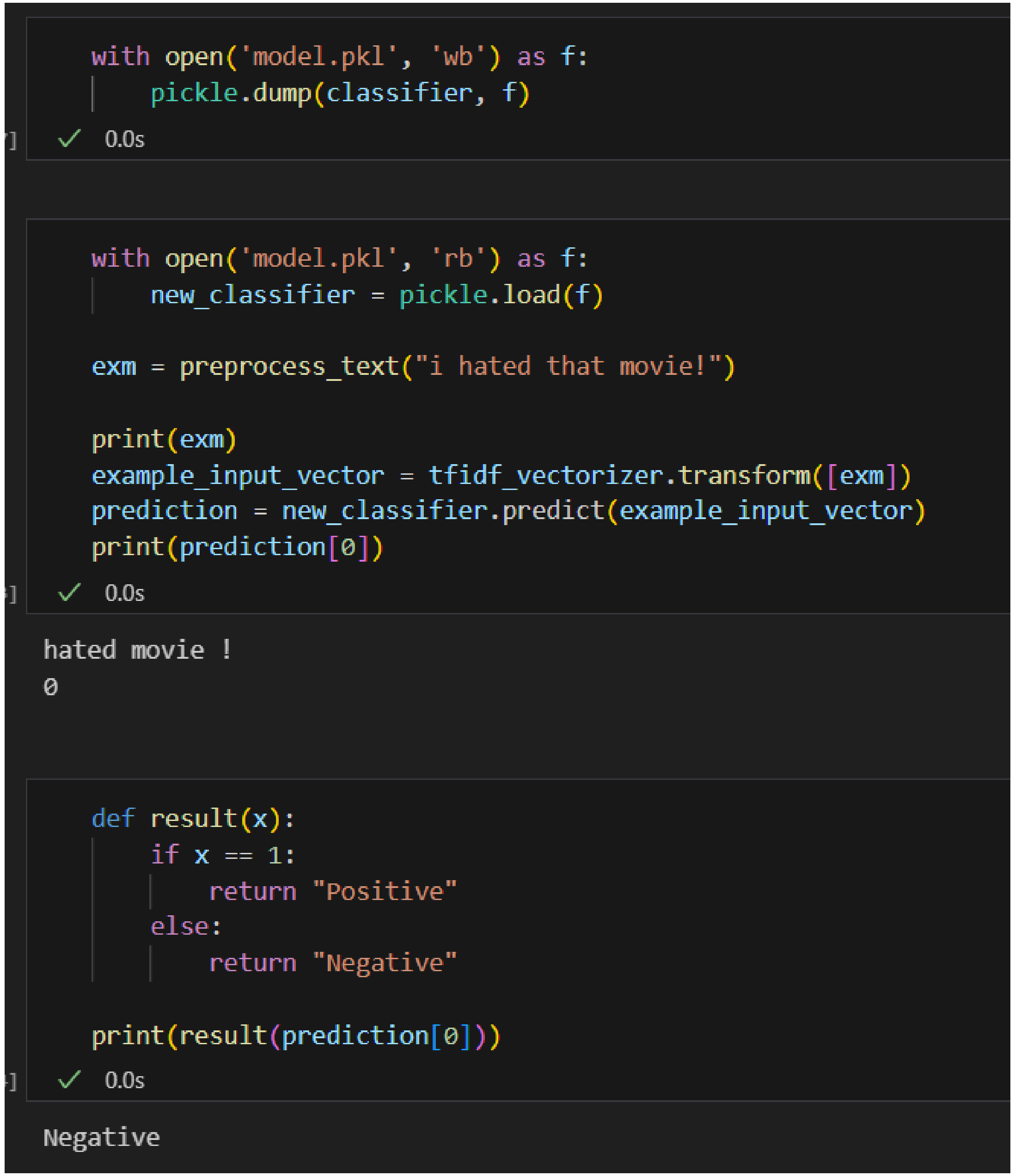
```
Cross-validation scores:
[0.901  0.8948 0.8936 0.8984 0.8884]
Mean Cross-validation Accuracy: 0.90
```

Test dataset has 12500 positive and 12500 negative reviews , 25000 in total , same as train dataset.

Data Glacier

Your Deep Learning Partner

Pickle Dump



```
with open('model.pkl', 'wb') as f:
    pickle.dump(classifier, f)
```

```
with open('model.pkl', 'rb') as f:
    new_classifier = pickle.load(f)

exm = preprocess_text("i hated that movie!")

print(exm)
example_input_vector = tfidf_vectorizer.transform([exm])
prediction = new_classifier.predict(example_input_vector)
print(prediction[0])
```

```
hated movie !
0
```

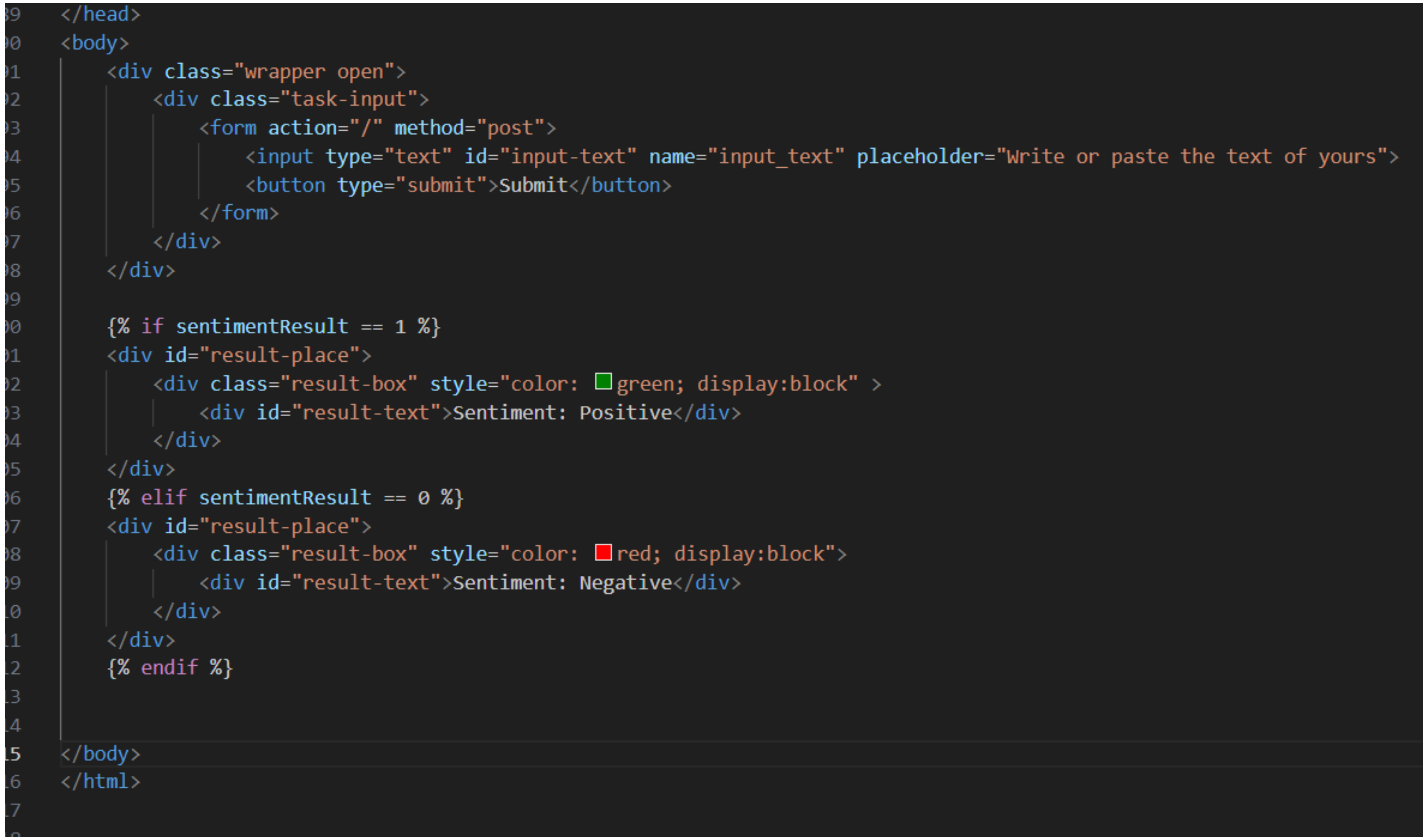
```
def result(x):
    if x == 1:
        return "Positive"
    else:
        return "Negative"

print(result(prediction[0]))
```

```
Negative
```

Data Glacier

Your Deep Learning Partner

HTML Code

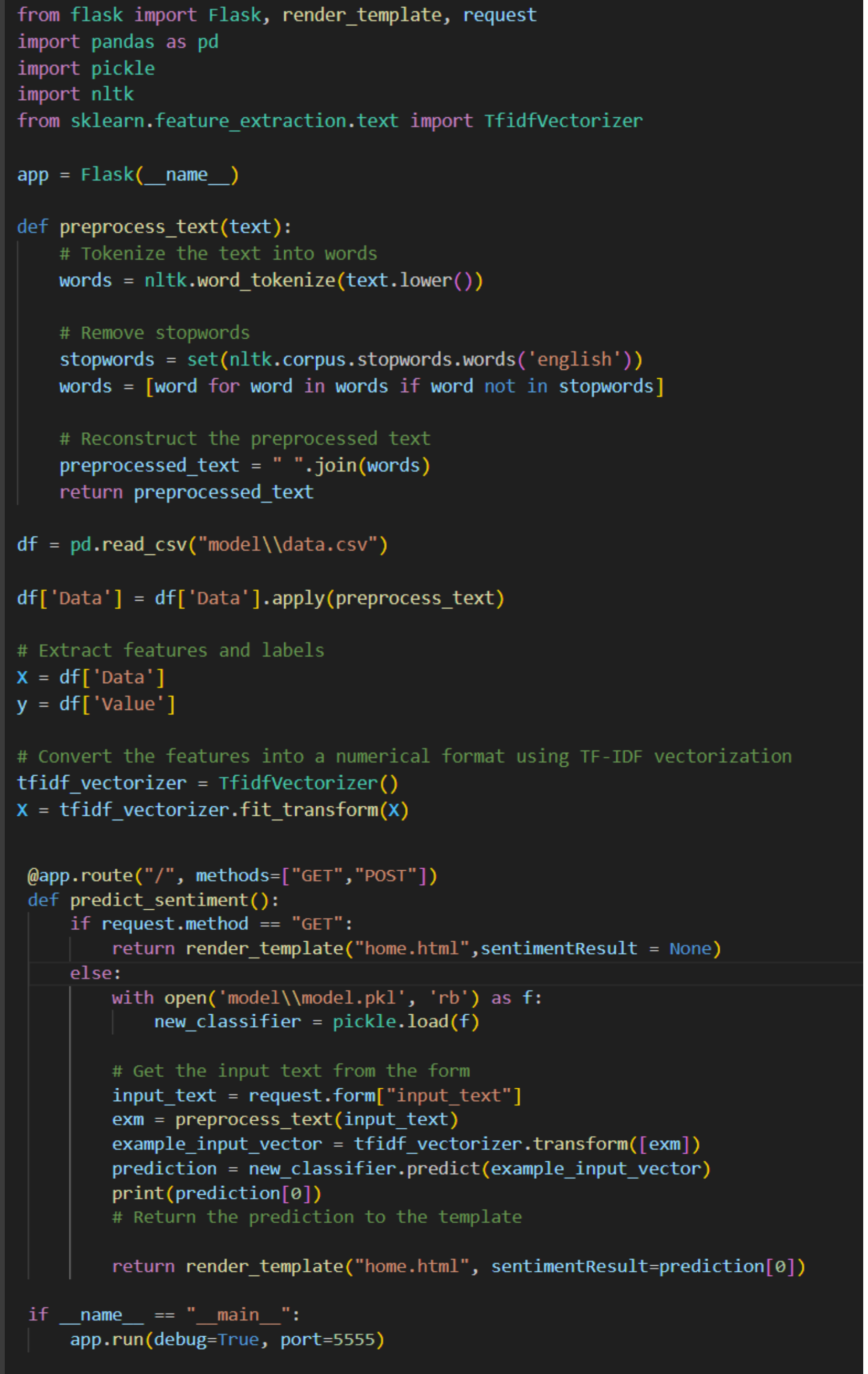


```
</head>
<body>
    <div class="wrapper open">
        <div class="task-input">
            <form action="/" method="post">
                <input type="text" id="input-text" name="input_text" placeholder="Write or paste the text of yours">
                <button type="submit">Submit</button>
            </form>
        </div>
    </div>

    {% if sentimentResult == 1 %}
    <div id="result-place">
        <div class="result-box" style="color: 🟩green; display:block" >
            <div id="result-text">Sentiment: Positive</div>
        </div>
    </div>
    {% elif sentimentResult == 0 %}
    <div id="result-place">
        <div class="result-box" style="color: 🟥red; display:block">
            <div id="result-text">Sentiment: Negative</div>
        </div>
    </div>
    {% endif %}


</body>
</html>
```

Data Glacier

Your Deep Learning Partner

Flask Code

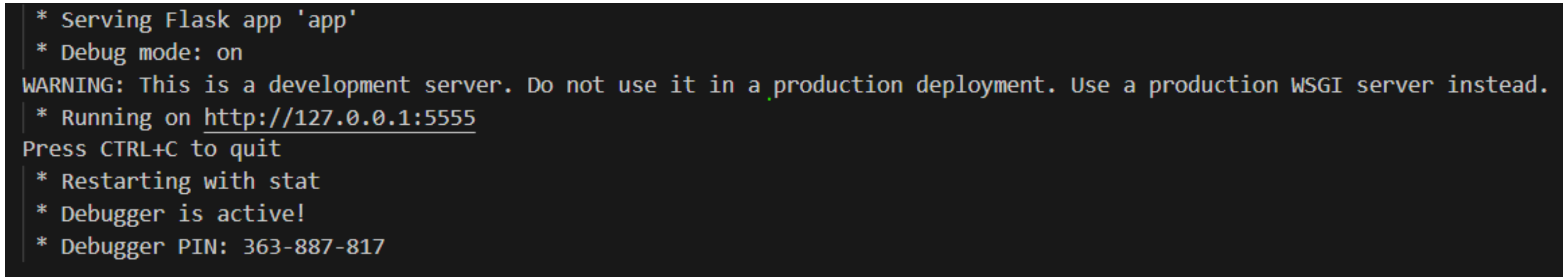


```python
from flask import Flask, render_template, request
import pandas as pd
import pickle
import nltk
from sklearn.feature_extraction.text import TfidfVectorizer

app = Flask(__name__)

def preprocess_text(text):
    # Tokenize the text into words
    words = nltk.word_tokenize(text.lower())

    # Remove stopwords
    stopwords = set(nltk.corpus.stopwords.words('english'))
    words = [word for word in words if word not in stopwords]

    # Reconstruct the preprocessed text
    preprocessed_text = " ".join(words)
    return preprocessed_text

df = pd.read_csv("model\\data.csv")

df['Data'] = df['Data'].apply(preprocess_text)

# Extract features and labels
X = df['Data']
y = df['Value']

# Convert the features into a numerical format using TF-IDF vectorization
tfidf_vectorizer = TfidfVectorizer()
X = tfidf_vectorizer.fit_transform(X)


@app.route("/", methods=["GET","POST"])
def predict_sentiment():
    if request.method == "GET":
        return render_template("home.html",sentimentResult = None)
    else:
        with open('model\\model.pkl', 'rb') as f:
            new_classifier = pickle.load(f)

        # Get the input text from the form
        input_text = request.form["input_text"]
        exm = preprocess_text(input_text)
        example_input_vector = tfidf_vectorizer.transform([exm])
        prediction = new_classifier.predict(example_input_vector)
        print(prediction[0])
        # Return the prediction to the template

        return render_template("home.html", sentimentResult=prediction[0])

if __name__ == "__main__":
    app.run(debug=True, port=5555)
```

Data Glacier

Your Deep Learning Partner

Stand the Website and Testing

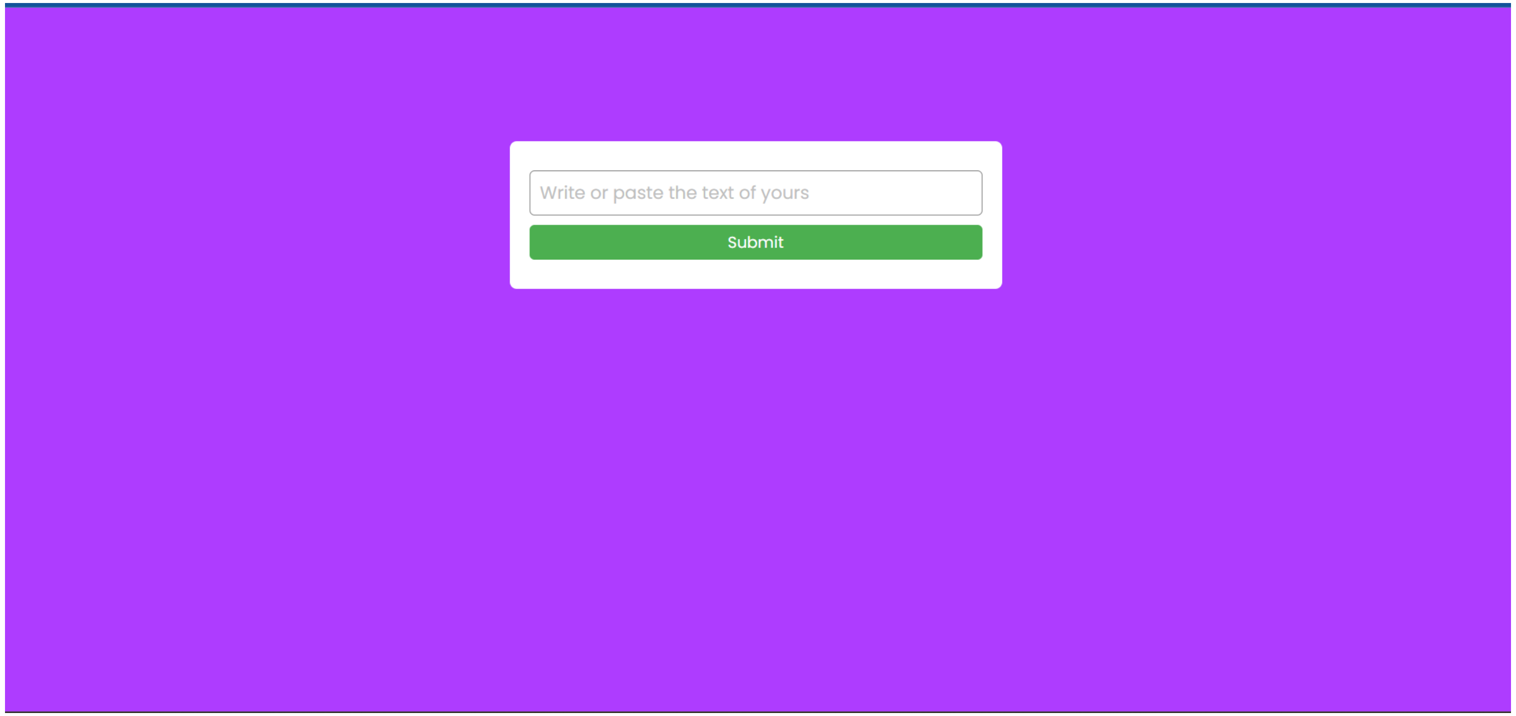


```
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5555
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 363-887-817
```

Data Glacier

Your Deep Learning Partner

Stand the Website and Testing



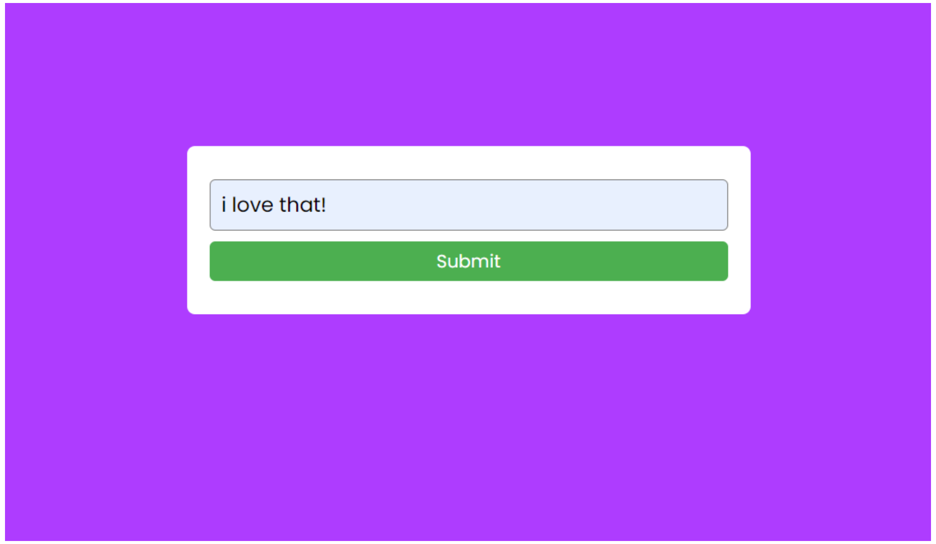
Opening Page of the site.

Your Deep Learning Partner

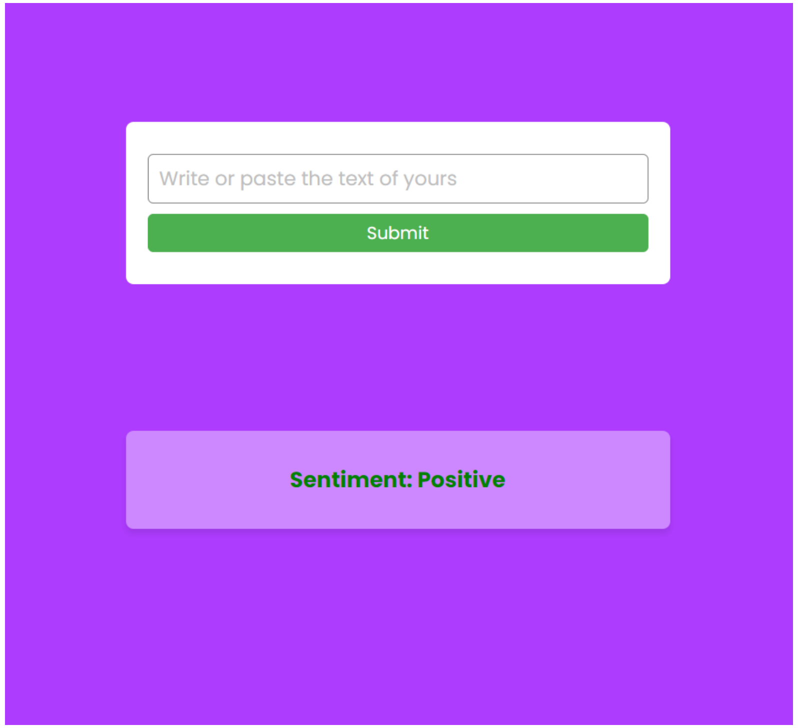
Stand the Website and Testing

Data Glacier

Your Deep Learning Partner



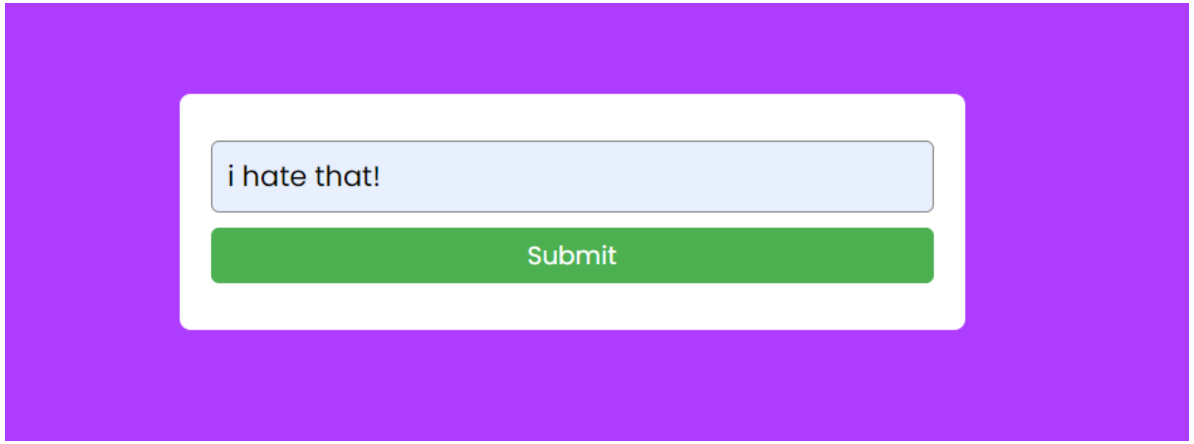
i love that!



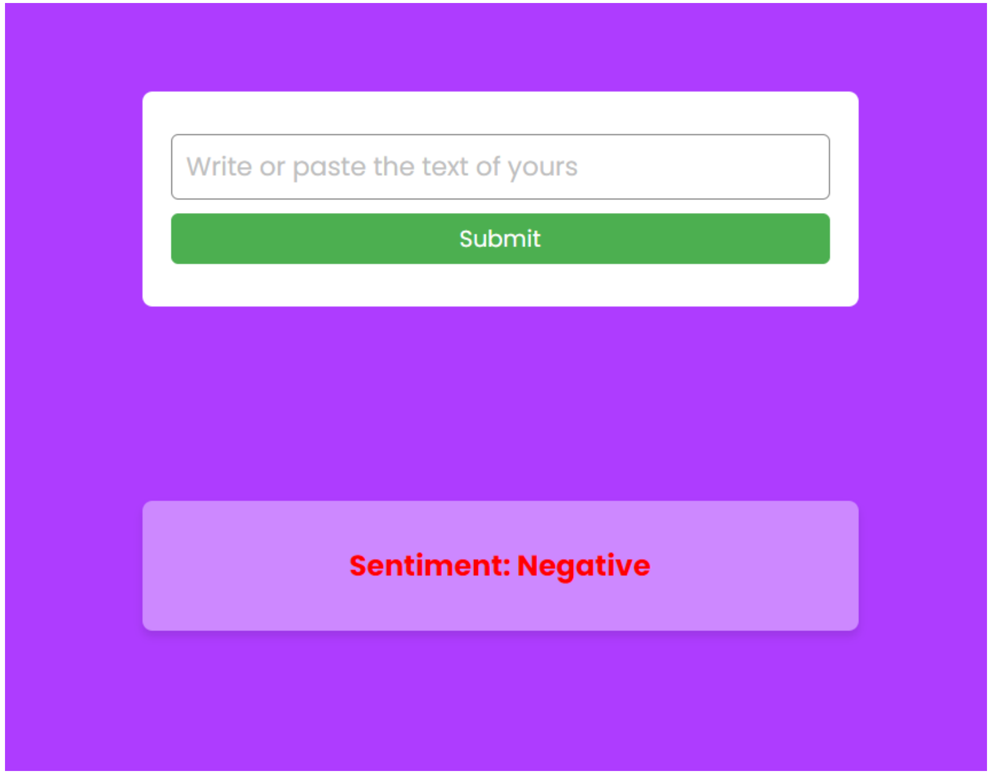
Sentiment: Positive

Writing input then press submit button , the result is would be under the input box.

Stand the Website and Testing



i hate that!



Write or paste the text of yours

Sentiment: Negative

Data Glacier

Your Deep Learning Partner

Stand the Website and Testing

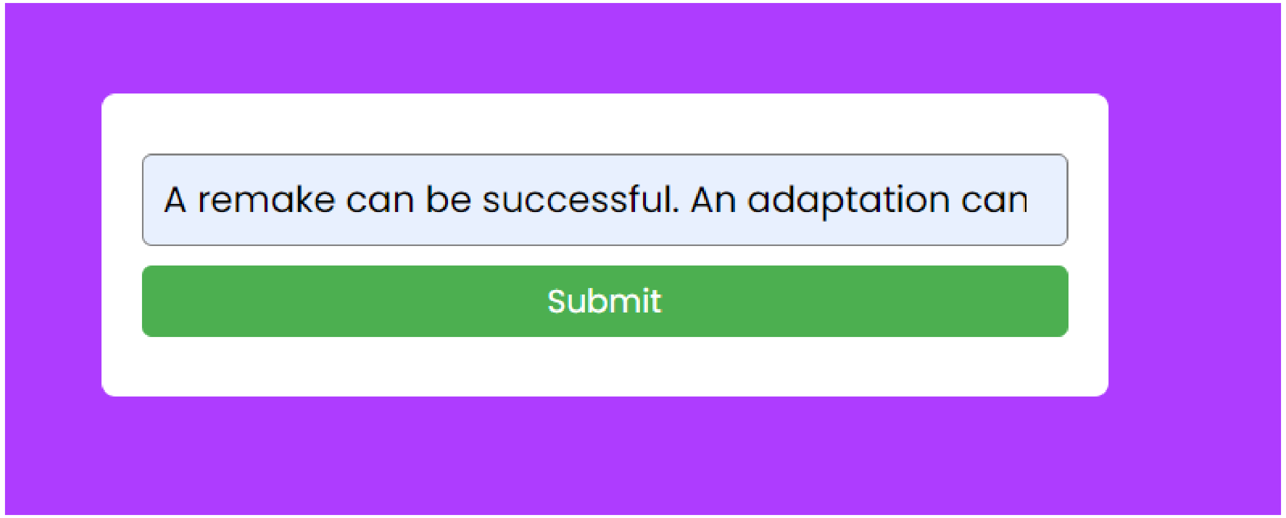
Data Glacier

Your Deep Learning Partner

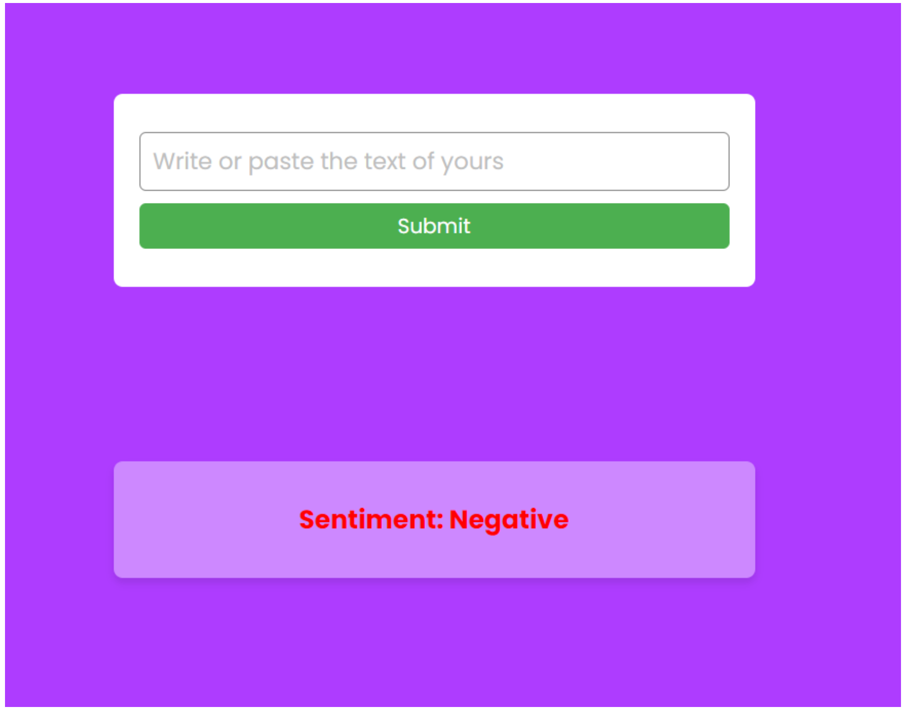
NEGATIVE INPUT EXAMPLE

A remake can be successful. An adaptation can be successful. It isn't relevant whether its a remake or an adaptation. A good movie is a good movie and a poor movie is a poor movie, regardless. Sarkar, I am afraid, was a very poor movie. First of all, just by making characters look dangerous, or macho, they don't bring in an aura about them. What was so brilliant about Nagre(Amitabh Bacchan's character) that we should have been in aura of his 'power' and what showed the 'benevolence' of the character? Nothing. This fact was said by a commentator and Amitabh kept giving facial expressions. Now Amitabh can give brilliant facial expressions but why should it mean any thing if there is no history or story to go with it.There wasn't proper charecterisation of the characters who worked under 'sarkar' too. Just because a man had spectacles, why should we assume he is wise. The flow of the movie was generally dullbecause scenes from the Godfather were created

Stand the Website and Testing



A remake can be successful. An adaptation can



Write or paste the text of yours

Sentiment: Negative

Data Glacier

Your Deep Learning Partner

End

Data Glacier

Your Deep Learning Partner

THANK YOU !!!!!