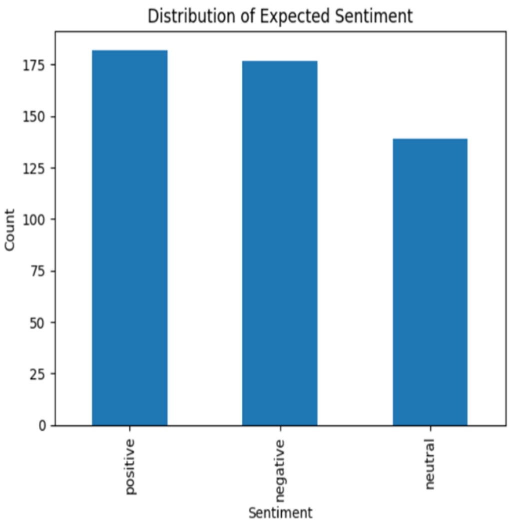
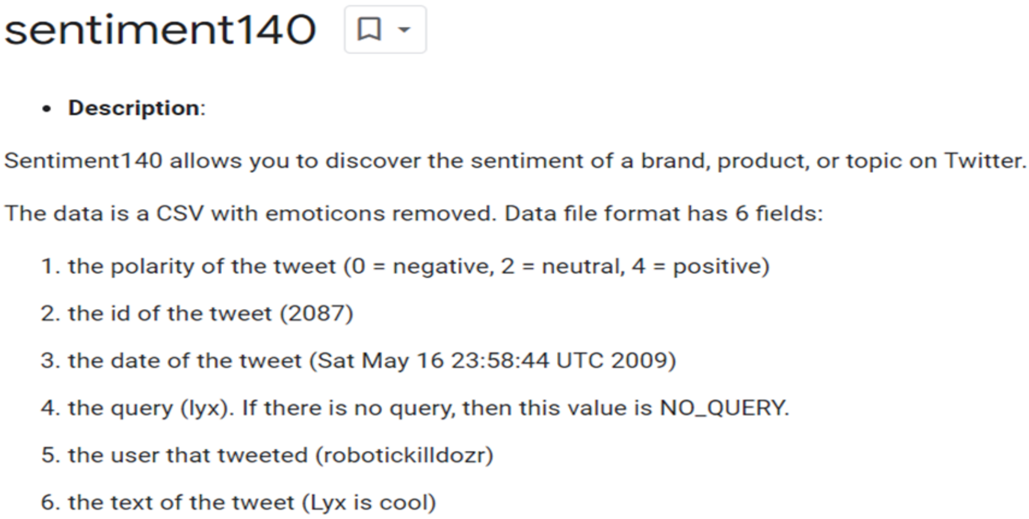
Sentiment Analysis Problem Background Overview

The goal is to create a model that can identify the sentiment of a given query. This can be useful for understanding customer opinions about a product or situation. The model should output the sentiment and confidence score.

In this presentation, I will walk you through my methodology, give a brief overview of the code, show a demo of the working model, present the test cases output with metrics such as accuracy, and suggest possible future improvements

Methodology

1. `Define the problem`: Our test cases come from the Sentiment140 dataset, which is a multiclass dataset. We will use a hybrid lexicon approach to analyze the data {positive, neutral, negative}.
    - `sentiment analysis` is an important KPI for many enterprises.
    - `sentiment_test_cases.csv`: a modified version of the Sentiment140 - Test dataset containing 489 test cases
        - lexical-based unsupervised learning problem: multi-class (positive, negative, and neutral)
        - balanced dataset: instances of each class are relatively close to each other
        - 

    - `sentiment140`: commonly used as a benchmark dataset for evaluating sentiment analysis methods.
        - 

sentiment140

- **Description**:

Sentiment140 allows you to discover the sentiment of a brand, product, or topic on Twitter.

The data is a CSV with emoticons removed. Data file format has 6 fields:

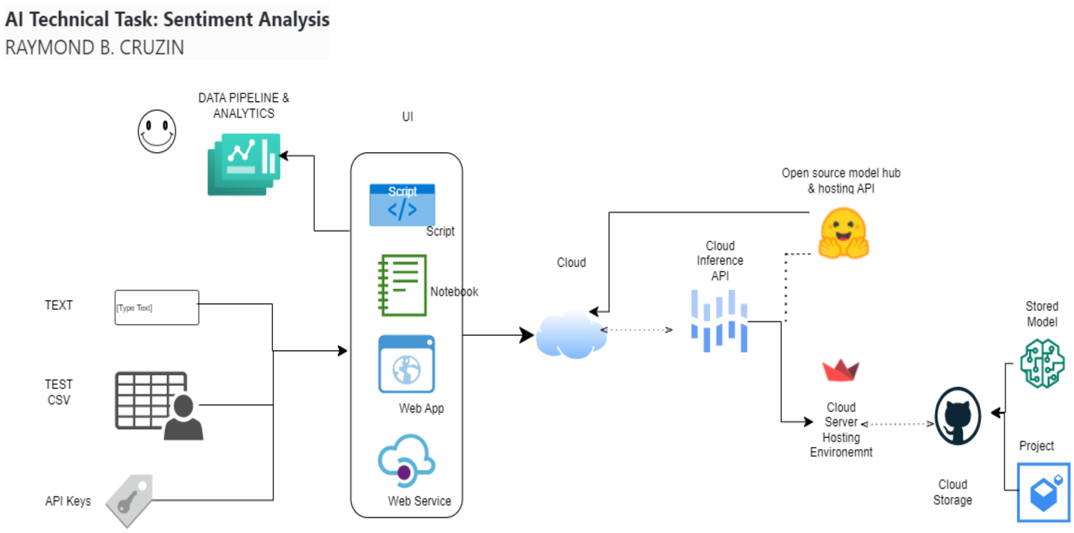
1. the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive)
2. the id of the tweet (2087)
3. the date of the tweet (Sat May 16 23:58:44 UTC 2009)
4. the query (lyx). If there is no query, then this value is NO_QUERY.
5. the user that tweeted (robotickilldozr)
6. the text of the tweet (Lyx is cool)

2. `Model selection`: We will benchmark related open-source models based on their size, parameters, training dataset, training date, popularity, etc.

Model	Base Model	Description	Latest Update	Downloads last month	Language	Fine-tuned Dataset	Model Size	Parameters
cardiffnlp/twitter-roberta-base-sentiment-latest	RoBERTa-base	RoBERTa-base model trained on Twitter 2021 ~124M (RoBERTa-base) fine-tuned on TweetEval benchmark (~124M tweets)	2022	1,618,710	English	tweet_eval ~66k tweets (train=45.6k,test=12.3k,val=2k)	500MB	125M

Model	Base Model	Description	Latest Update	Downloads last month	Language	Fine-tuned Dataset	Model Size	Parameters
cardiffnlp/twitter-xlm-roberta-base-sentiment	XLM-RoBERTa-base	Multilingual XLM-RoBERTa-base model trained on ~198M tweets and fine-tuned for sentiment analysis	2021	1,179,935	Multilingual	8 languages (Ar, En, Fr, De, Hi, It, Sp, Pt) tweets	1.3GB	270M
cardiffnlp/twitter-roberta-base-sentiment	RoBERTa-base	RoBERTa-base model trained on Twitter ~58M (RoBERTa-base) fine-tuned on TweetEval benchmark (~58M tweets)	2021	771,567	English	tweet_eval ~66k tweets	500MB	125M
finiteautomata/bertweet-base-sentiment-analysis	BERTweet-base	VinAIResearch/BERTweet model trained on Twitter 2012-2019 845M English Tweets and 5M COVID-19 Tweets	2023	230,948	English	SemEval 2017 corpus (around ~40k tweets)	530MB	135M

```
* To have access on each of the model, you can clone from below repositories:
    * `git clone https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment-latest`
    * `git clone https://huggingface.co/cardiffnlp/twitter-xlm-roberta-base-sentiment`
    * `git clone https://huggingface.co/finiteautomata/bertweet-base-sentiment-analysis`
    * `git clone https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment`
```

3. ***`Development`**:
    - **App Work Flow**:
    - 

    - **Sampled Testing via Inference API**:
        - Objective: Easily load any model from HuggingFace INFERENCE and test the model output and confidence score of a given model from a given text.
        - `Input`: text
        - `Output`: model output {sentiment, confidence_score}
            - [GitHub Repository]()
            1. `RUN LOCAL SCRIPT`: i.e model_name = `cardiffnlp/twitter-xlm-roberta-base-sentiment`

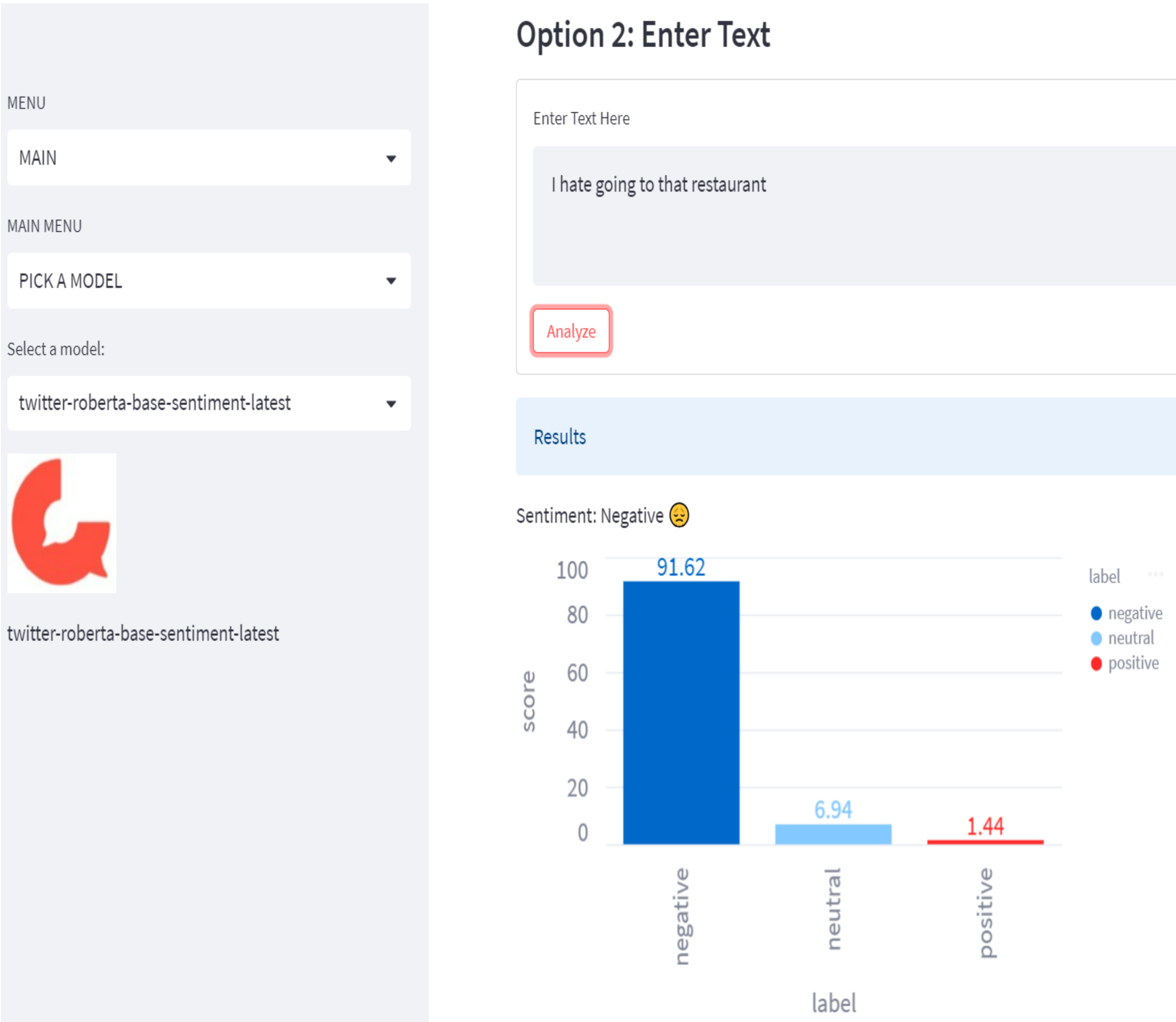
Task	The Python script should be able to take in a user input then output the sentiment and confidence score
Input	“I hate going to that restaurant”
Output	{“model_output”:“negative”,”confidence_score”: 98.42}

```
(my-venv) (base) PS C:\sprouts\script> python test.py
Input: "I hate going to that restaurant"
Output: {'model_output': 'negative', 'confidence_score': 95.5}
```

```
Input: "I love having coffee at work"
Output: {'model_output': 'positive', 'confidence_score': 90.58}
```

```
Input: "I am having a coffee"
Output: {'model_output': 'neutral', 'confidence_score': 79.07}
```

2. `RUN ON CLOUD WEB APPLICATION`: model_name = `cardiffnlp/twitter-xlm-roberta-base-sentiment-latest`



- **Quick code walk-through**:
  - Objective: Quick demo of the basic test script application. The demo can be run on the console.
    - GitHub Repository

sentiment_analyzer.py	test.py

```python
# sentiment_analyzer.py

import requests
import pandas as pd

class SentimentAnalyzer:

    def __init__(self):
        self.sentiment_mapping = {
            "positive": ["positive", "POS", "LABEL_2"],
            "neutral": ["neutral", "NEU", "LABEL_1"],
            "negative": ["negative", "NEG", "LABEL_0"]
        }

    def standardize_sentiment_label(self, label):
        for key, value in self.sentiment_mapping.items():
            if label.lower() in [x.lower() for x in value]:
                return key
        return label

    def convert_to_df(self, result):
        sentiment_df = pd.DataFrame(result)
        sentiment_df.set_index("label", inplace=True)
        return sentiment_df

    def analyze_text(self, text, model_path):
        api_endpoint = f"https://api-inference.huggingface.co/models/{model_path}"
        headers = {"Content-Type": "application/json"}
        payload = {"inputs": text}

        try:
            response = requests.post(api_endpoint, headers=headers, json=payload, timeout=60)
            response.raise_for_status()
            result = response.json()
            standardized_result = [{"label": self.standardize_sentiment_label(x["label"]),
                                    "score": round(x["score"] * 100, 2)} for x in result[0]]
            return standardized_result
        except requests.exceptions.RequestException as e:
            print(f"An error occurred while sending the request: {e}")
            return None
```

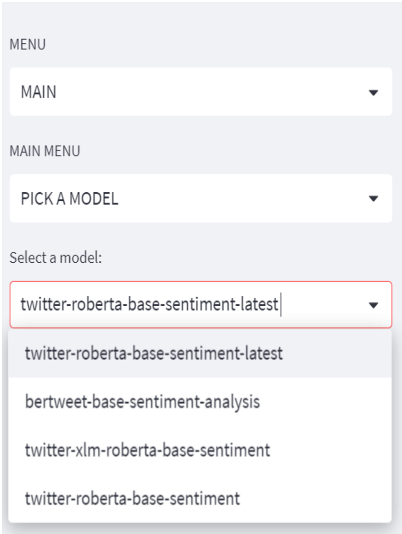
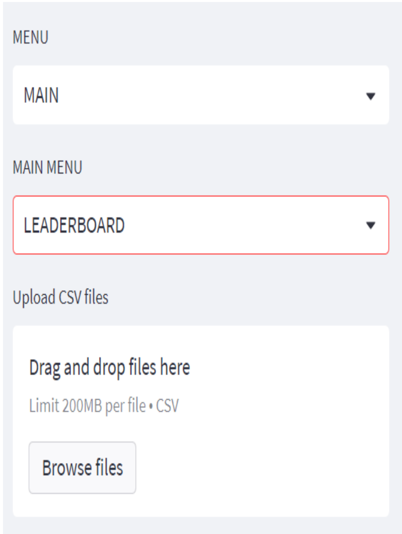
```python
# test.py

# Import the SentimentAnalyzer class from the sentiment_analyzer module
from sentiment_analyzer import SentimentAnalyzer

if __name__ == "__main__":

    # Create a SentimentAnalyzer object
    analyzer = SentimentAnalyzer()

    # Get user input
    user_input = input("Input: ")

    # Analyze sentiment and display result
    result = analyzer.analyze_text(user_input, "cardiffnlp/twitter-roberta-base-sentiment-latest")

    if result:
        # Sort the results by confidence score in descending order
        result.sort(key=lambda x: x["score"], reverse=True)

        # Select the sentiment with the highest confidence score
        sentiment = result[0]["label"]
        confidence = result[0]["score"]

        output = {"model_output": sentiment, "confidence_score": confidence}
        print(f"Output: {output}")
```

- **Resource Notebook for Benchmarking**:
  - Objective: Benchmark against test dataset and compare computational efficiency.
  - Notes: Since the model sizes are around 1GB, I downloaded the model, ran it locally to compare computational efficiency on my machine, then saved the model output in a csv for all test cases.
  - `Input`: sentiment_test_cases.csv
  - `Output`:
    - output_{`model_name`}_sentiment_test.csv per model
    - computational_efficiency.csv
    - GitHub Repository
  - Result `Computational Efficiency` (Running on CPU):

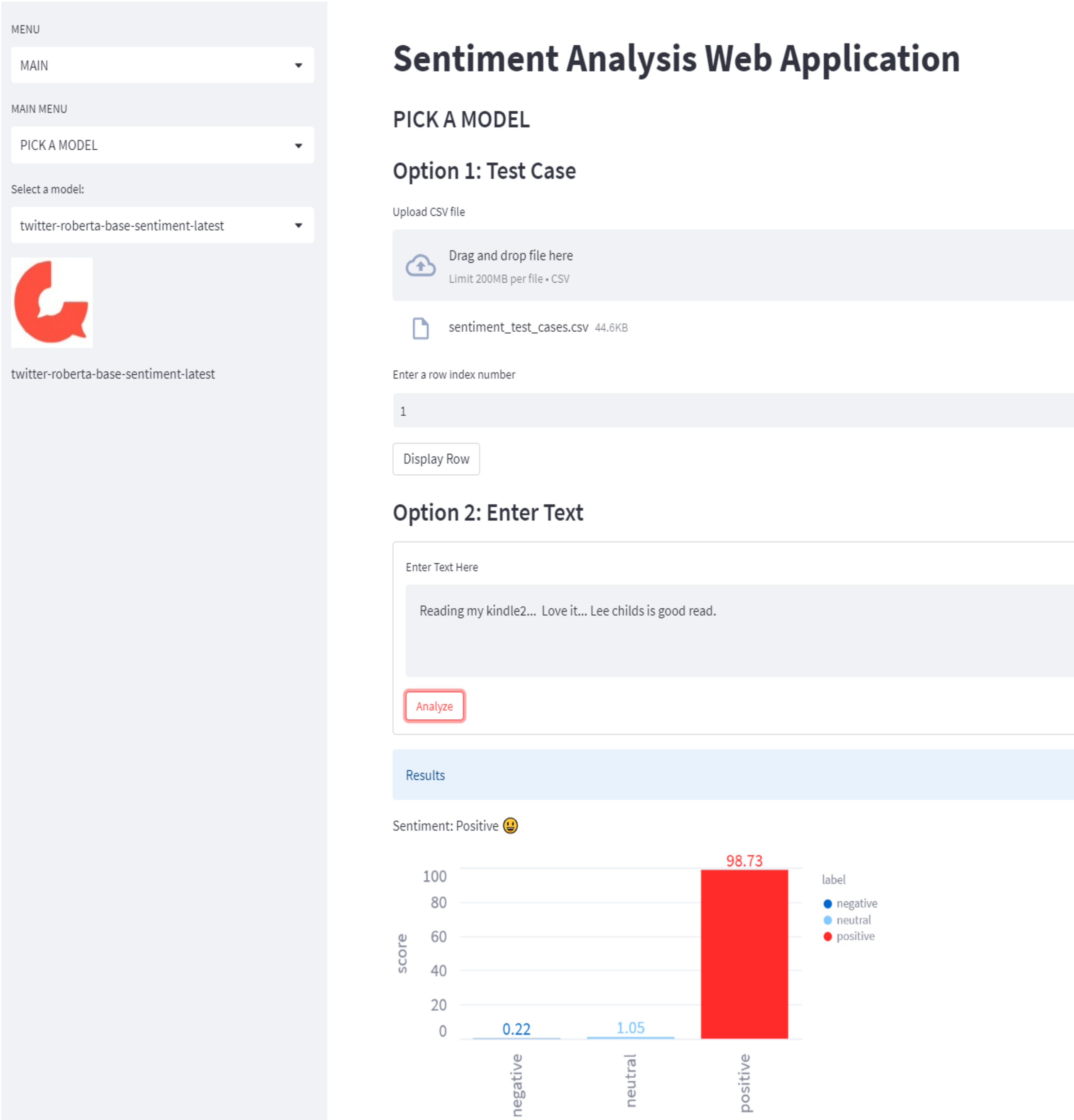
Model	Time (seconds)
twitter-roberta-base-sentiment-latest	27.27
bertweet-base-sentiment-analysis	27.52
twitter-xlm-roberta-base-sentiment	30.28
twitter-roberta-base-sentiment	29.30

- **Cloud Web Application for Testing and Benchmarking**:
- `Use cases`
-

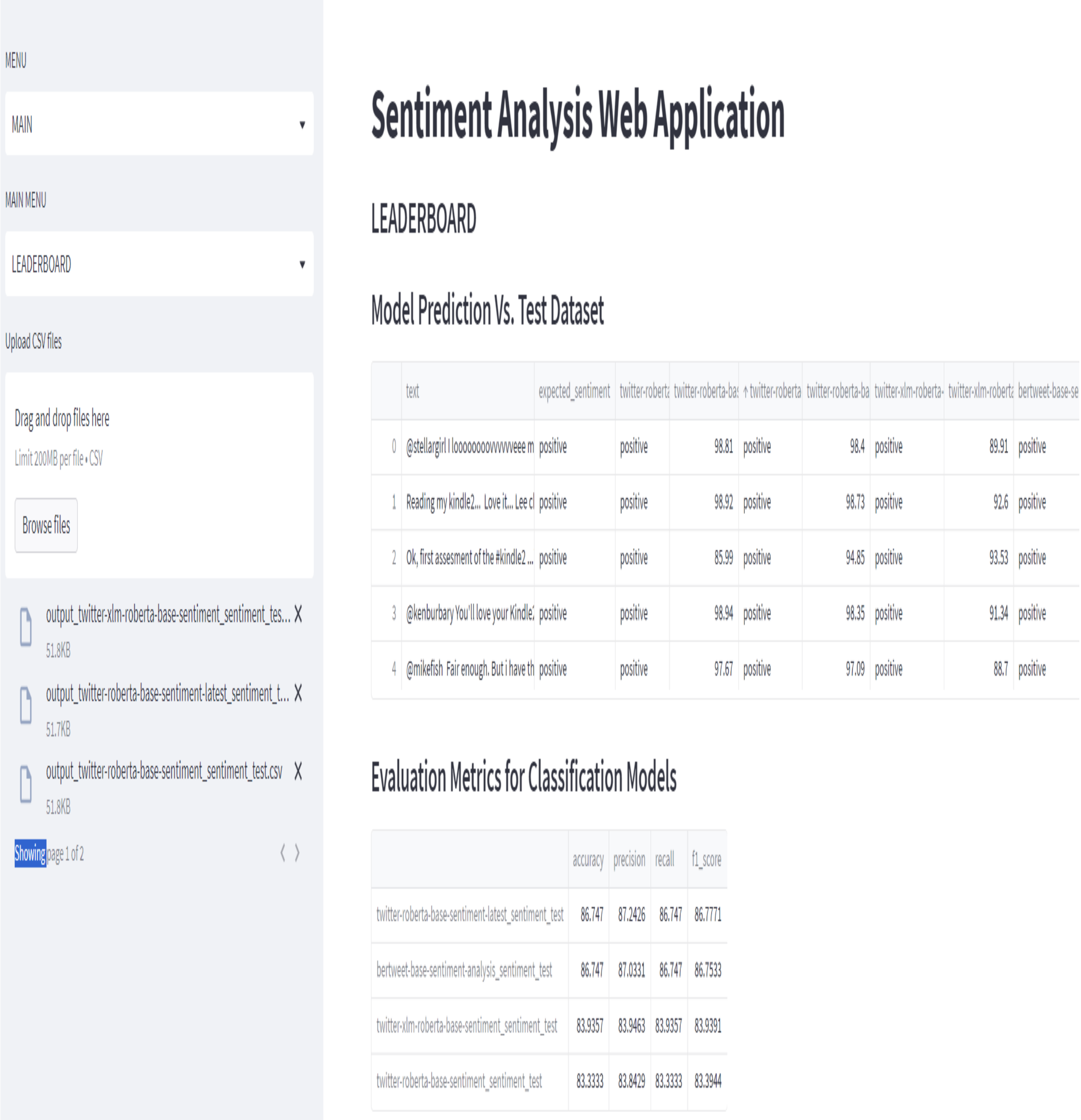
Pick A Model:	Model Leaderboard

Pick A Model:	Model Leaderboard
	

- `Model Testing`



-
- `Benchmarking`

- 

4. `Results and analysis`: Leaderboard and evaluation of each model from `sentiment_test_cases.csv dataset` accuracy, weighted average f1_score, etc.
   - **Web application**: Walk through of how to choose a model and do evaluation of the results of model leaderboard against the test_cases.
     - Input: output_{`model_name`}_sentiment_test.csv `[upload multiple csv files]`
     - Output: summary of model performance metrics in dataframe and visuals
   - **Demo**: Host selected model from Streamlit Cloud Server (no depedency on HuggingFace Inference) from GitHub Repo
     - `WEB DASHBOARD` [BENCHMARK]
       - Input: upload `sentiment_test_cases.csv`
       - Output: display output dataframe in requirement for the submission of `output_sentiment_test.csv`
       - Repo
       - Web Dashboard
         - `URL: https://cardiffnlp-twitter-roberta-base-sentiment-latest-rcruzin-ai.streamlit.app`
         - `Tested on input text:`

cardiffnlp/twitter-roberta-base-sentiment-latest

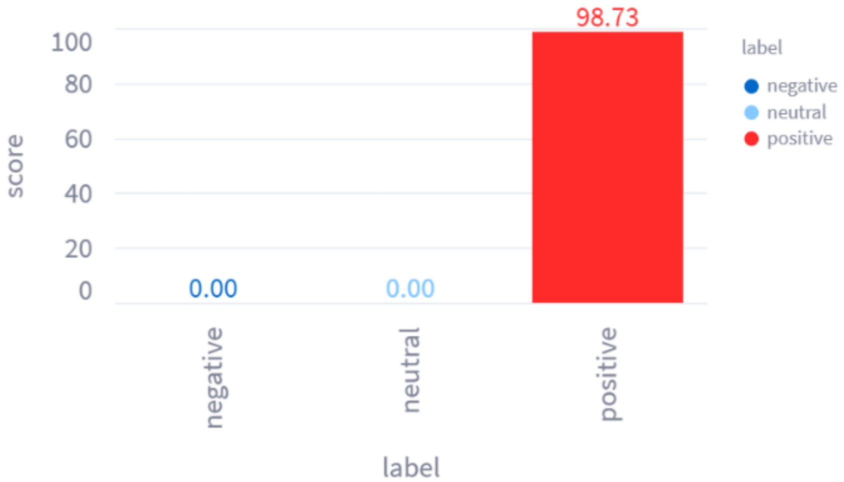
Option 1: Enter Text

Enter Text Here

Reading my kindle2... Love it... Lee childs is good read.

Analyze

Sentiment: Positive 😃



-
    - `Tested on sentiment_test_cases.csv:`

sentiment_test_cases.csv 44.6KB

Enter a row index number

1

Display Row

Benchmark

Accuracy: 86.75 %

	text	expected_sentiment	model_output	confidence_score
0	@stellargirl I loooooooovvvvvveee my Kindle2	positive	positive	98.4
1	Reading my kindle2... Love it... Lee childs is gc	positive	positive	98.73
2	Ok, first assesment of the #kindle2 ...it fucking	positive	positive	94.85
3	@kenburbary You'll love your Kindle2. I've hac	positive	positive	98.35
4	@mikefish Fair enough. But i have the Kindle2	positive	positive	97.09
5	@richardebaker no. it is too big. I'm quite hap|	positive	positive	82.42
6	Fuck this economy. I hate aig and their non loa	negative	negative	95.57
7	Jquery is my new best friend.	positive	positive	94.89
8	Loves twitter	positive	positive	94.11
9	how can you not love Obama? he makes jokes	positive	positive	53.31

- `WEB SERVICE`
  - Input: "I hate going to that restaurant"
  - Output: {"model_output":"negative","confidence_score": 98.42}
    - Github: https://github.com/rcruzin-ai/cardiffnlp-twitter-roberta-base-sentiment-latest-webservice.git
    - Demo Web Service: https://cardiffnlp-twitter-roberta-rcruzin-ai-webservice.streamlit.app/?text="I hate going to that restaurant"
    - Tested on browser:


```
{
  "model_output" : "negative"
  "confidence_score" : 90.99
}
```

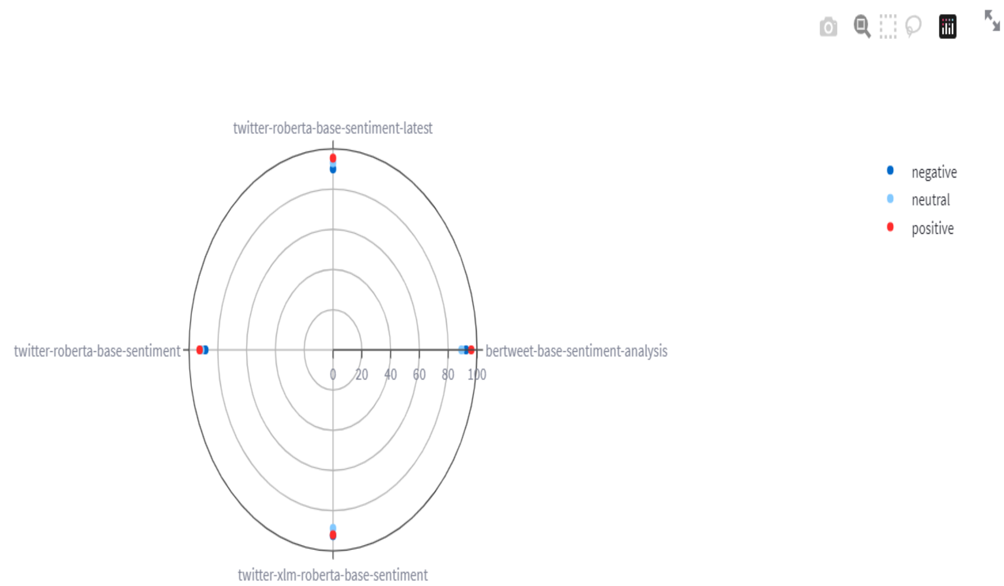

- **Evaluation Metrics (Overall Performance)**

Evaluation Metrics for Classification Models

	accuracy	precision	recall	f1_score
bertweet-base-sentiment-analysis	86.747	87.0331	86.747	86.7533
twitter-roberta-base-sentiment-latest	86.747	87.2426	86.747	86.7771
twitter-xlm-roberta-base-sentiment	83.9357	83.9463	83.9357	83.9391
twitter-roberta-base-sentiment	83.3333	83.8429	83.3333	83.3944

- **Weighted F1 Score Per Class**

Weighted Ave F1_Score for Multi Classification Models



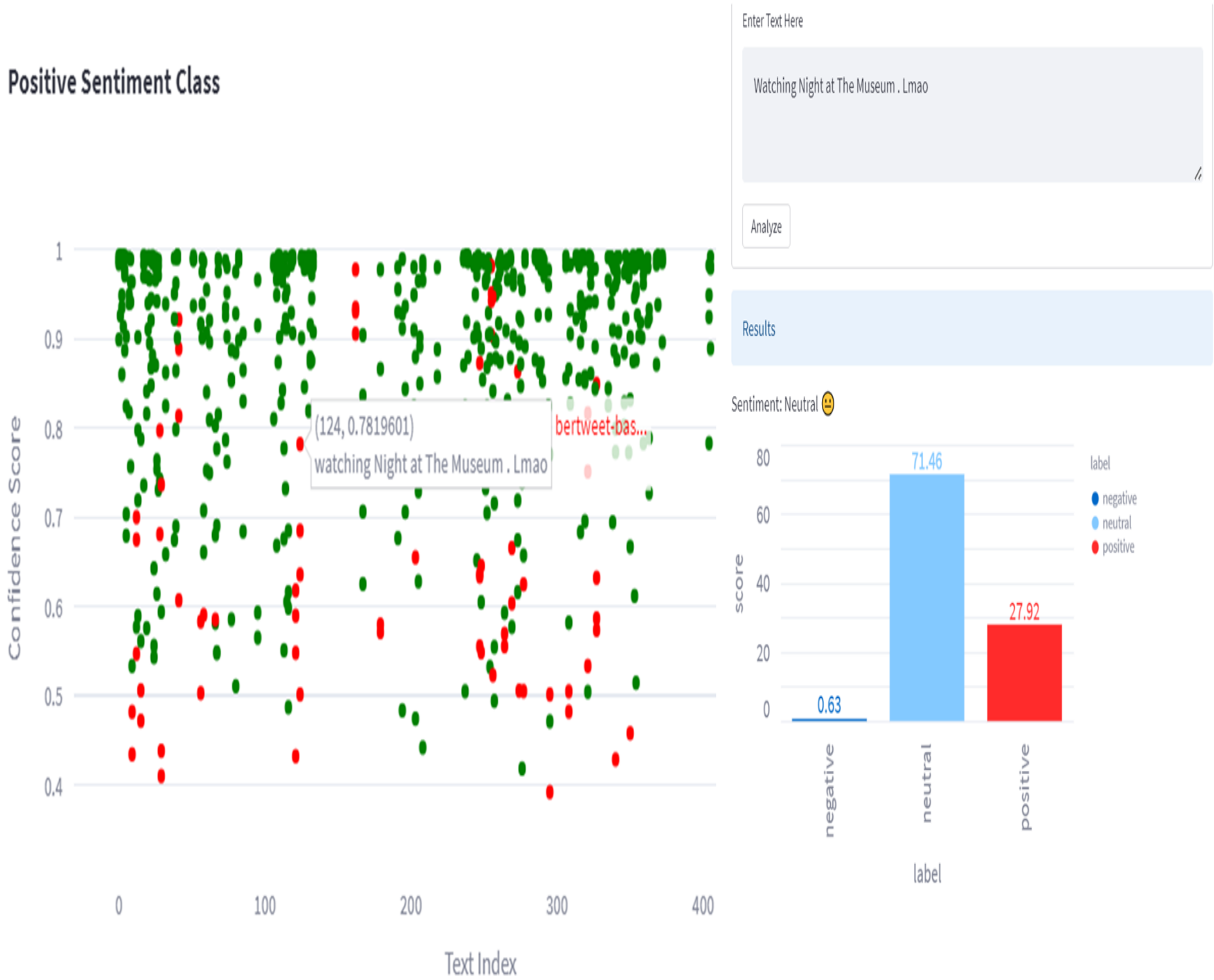
	model	negative	neutral	↓ positive
0	bertweet-base-sentiment-analysis	92.4	89.24	96
1	twitter-roberta-base-sentiment-latest	90.06	93.08	95.4
2	twitter-roberta-base-sentiment	88.68	91.41	92.63
3	twitter-xlm-roberta-base-sentiment	92.4	88.8	91.99

- Streamlit Using Huffing Face Inference API
- `URL: https://ai-task-rcruzin-ai.streamlit.app` * GitHub Repository

- **Summary of Results**
  - Winner `**twitter-roberta-base-sentiment-latest**` 😉 from
    - The best model for the `sentiment_test_cases.csv` dataset.
    - This model has the highest overall performance in terms of accuracy, precision, recall, and F1 score.
    - It also has a good balance between computational efficiency and performance, with a relatively fast processing time
    - high F1 scores for all sentiment classes.

5. `Possible future improvements`:

   1. We really cannot achieve a near perfect accuracy of sentiment analysis since this problem has subjective and biases to it.


Positive Sentiment Class
Confidence Score
Text Index
(124, 0.7819601)
watching Night at The Museum . Lmao
bertweet-bas...
Enter Text Here
Watching Night at The Museum . Lmao
Analyze
Results
Sentiment: Neutral
score
label
0.63
71.46
27.92
negative
neutral
positive


Negative Sentiment Class
Confidence Score
(259, 0.908822)
Enter Text Here
Kobe is the best in the world not lebron .
Analyze
Results
Sentiment: Positive
score
label
0.42
8.70
90.88
negative
neutral
positive

2. Additional insights from the model benchmarking results suggest that `BERT` or `RoBERTa` is a good choice for this problem.
    - The results suggest that the `RoBERTa-base model` is well suited for sentiment analysis tasks, particularly when fine-tuned on a large dataset of English tweets.
    - The `twitter-xlm-roberta-base-sentiment` model supports 8 languages, but since the `test_cases.csv` dataset only contains English tweets, this model may not have performed as well as other models specifically trained on English data.
    - The `bertweet-base-sentiment-analysis` model has a high F1 score for the negative class, indicating that it may be particularly good at identifying negative sentiment in tweets.
    - The `twitter-roberta-base-sentiment-latest` and `bertweet-base-sentiment-analysis` models are both popular, with a large number of downloads last month. This may indicate that they are widely used and well-regarded by the community.
    - Achieving an accuracy of `80%-85%` is a good benchmark. All models were able to deliver this given their base model and the amount of data they were trained and fine-tuned on.
    -

3. Even though higher accuracy is desirable, this last suggestion might achieved higher accuracy but will be more biased towards the Sentiment140 dataset. Thus, benchmarking base transformer models is always the first step before training your own.
    - To achieve higher accuracy on a specific dataset, we can fine-tune the model (using base models such as RoBERTa or BERTweet) on the Sentiment140 dataset for 3-class labels.

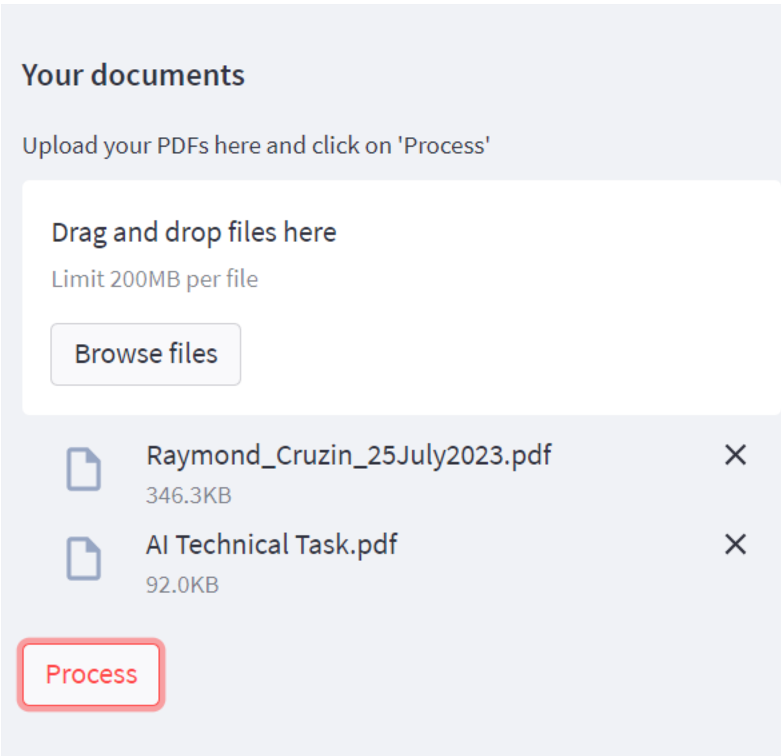
    - Adopting semantic similarity vectors to handle emoji or emoticons in relation to the entire context.

Table 3: List of Emoticons

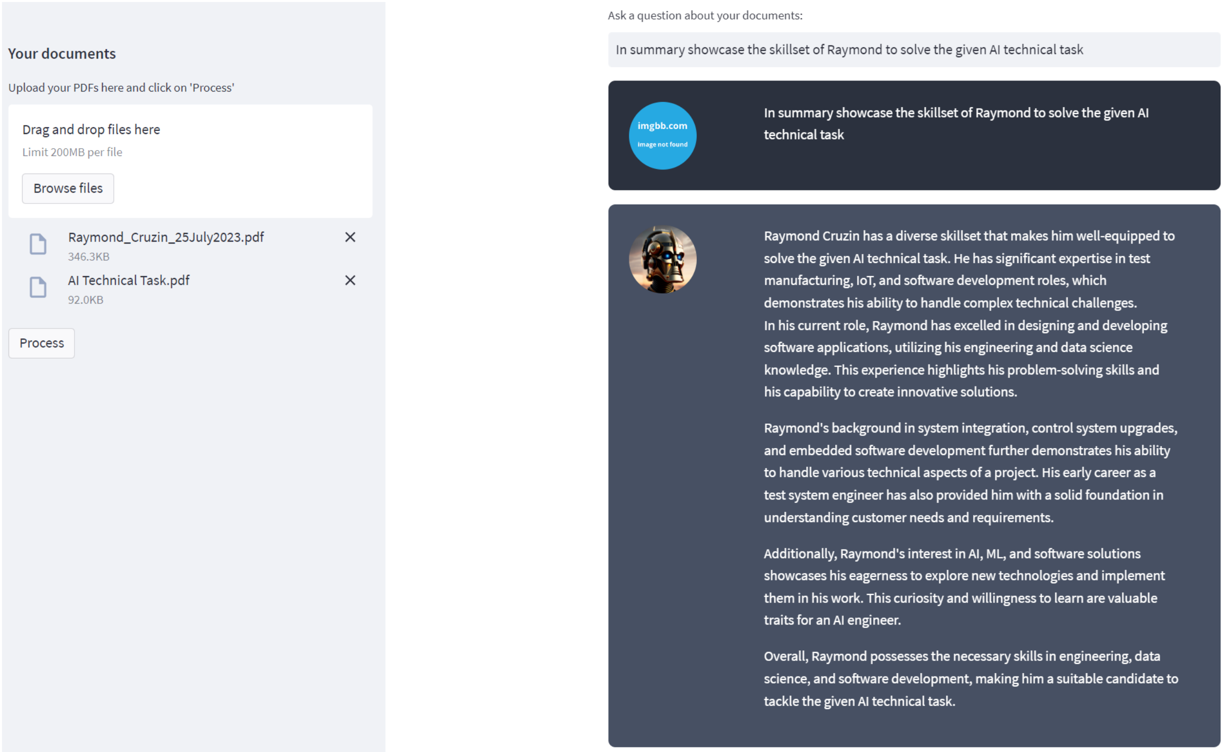
Emoticons mapped to :)	Emoticons mapped to :(
:)	:(
:-)	:-(
:)	: (
:D	
=)	

- Reinforcement learning - Fine-tuning on Sentiment140 dataset (~1.6M tweets)
  - Sample Notebook: `https://www.kaggle.com/code/nguyncaoduy/twitter-sentiment-analysis-roberta-96-accuracy/notebook`

6. **Additional(optional) demo to showcase personal projects:**
   - `Chat AI application` [semantic similarity , open ai or open source embeddings and llm, vector databases, langchain framework, streamlit, etc.]
     - Chat With Your Document:
     - Lets have a chat, use openai service to check if `Raymond_Cruzin.pdf` can do the `Ai_Technical_Task.pf` given
     - [`Process Documents:`]



     - [`Ask Questions:`]

- 

- A fine-tuned model (on Sentiment140 dataset) using `roBERTa` model.