



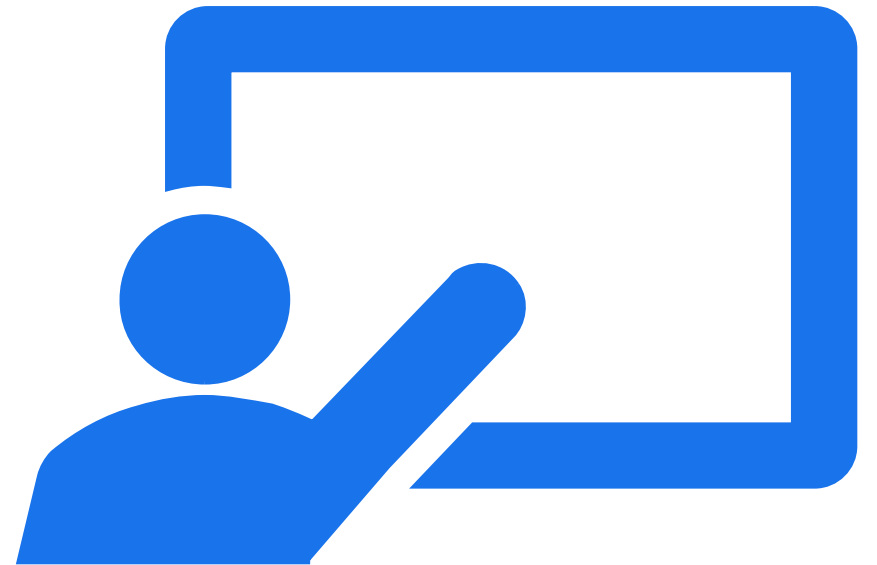
SentiVista: Know
What a Tweet
Really Feels

SentiVista

GUENFOUDI IHABE
FLOREA ROBERT

Summary

- Project presentation & Usage
- EDA
- Models
- API
- CI/CD
- Conclusion and improvements



Presentation



Retrieve tweets (or written text from social media)



Build models and train them on the tweets to analyze their sentiment



Build an API to analyze the sentiment from any English sentence



Measure how confident are our models

Usage

- Analyze the sentiment of large sets of tweets
- Statistics on the sentiments for a certain topic
- Anticipate trends

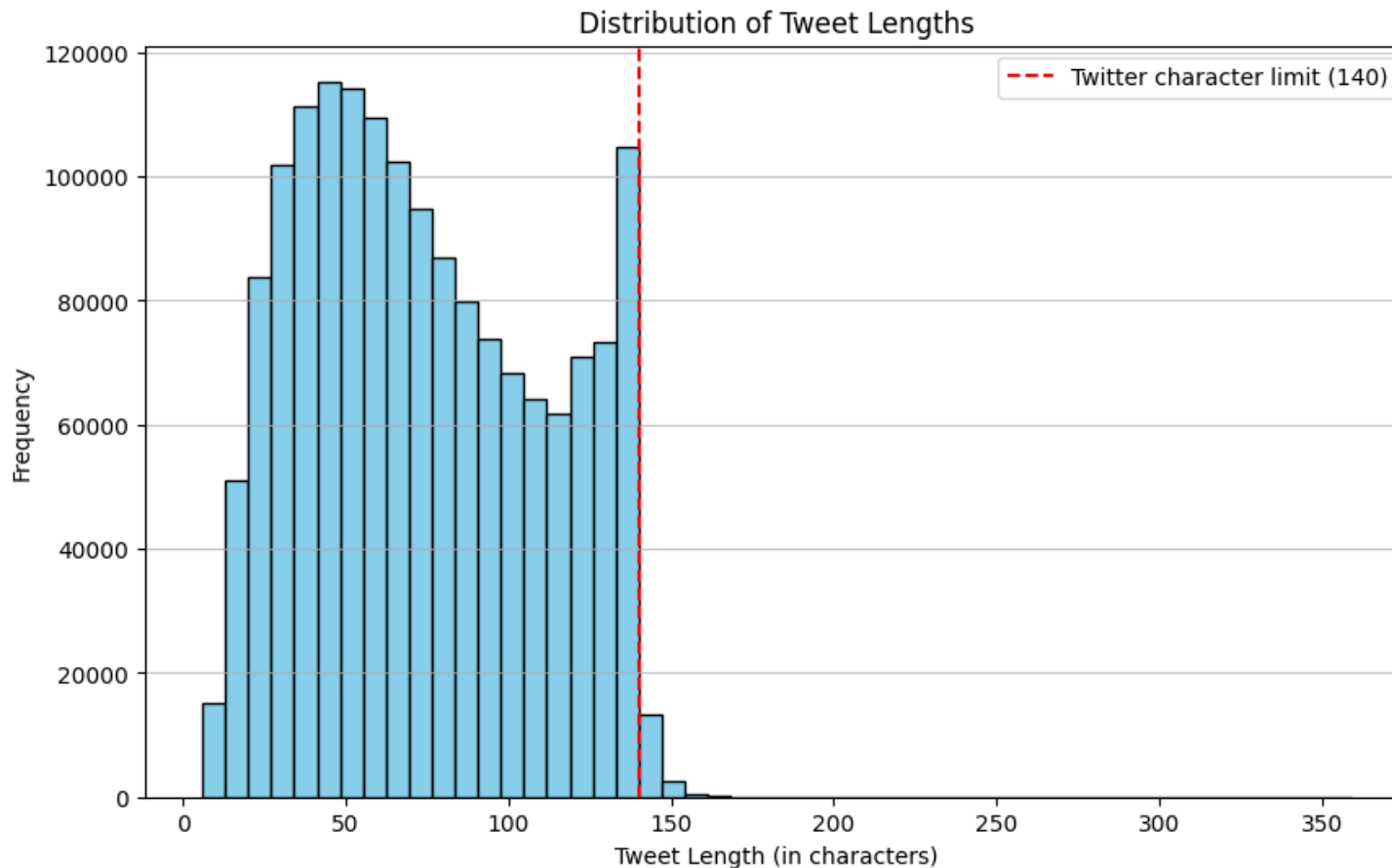


Dataset

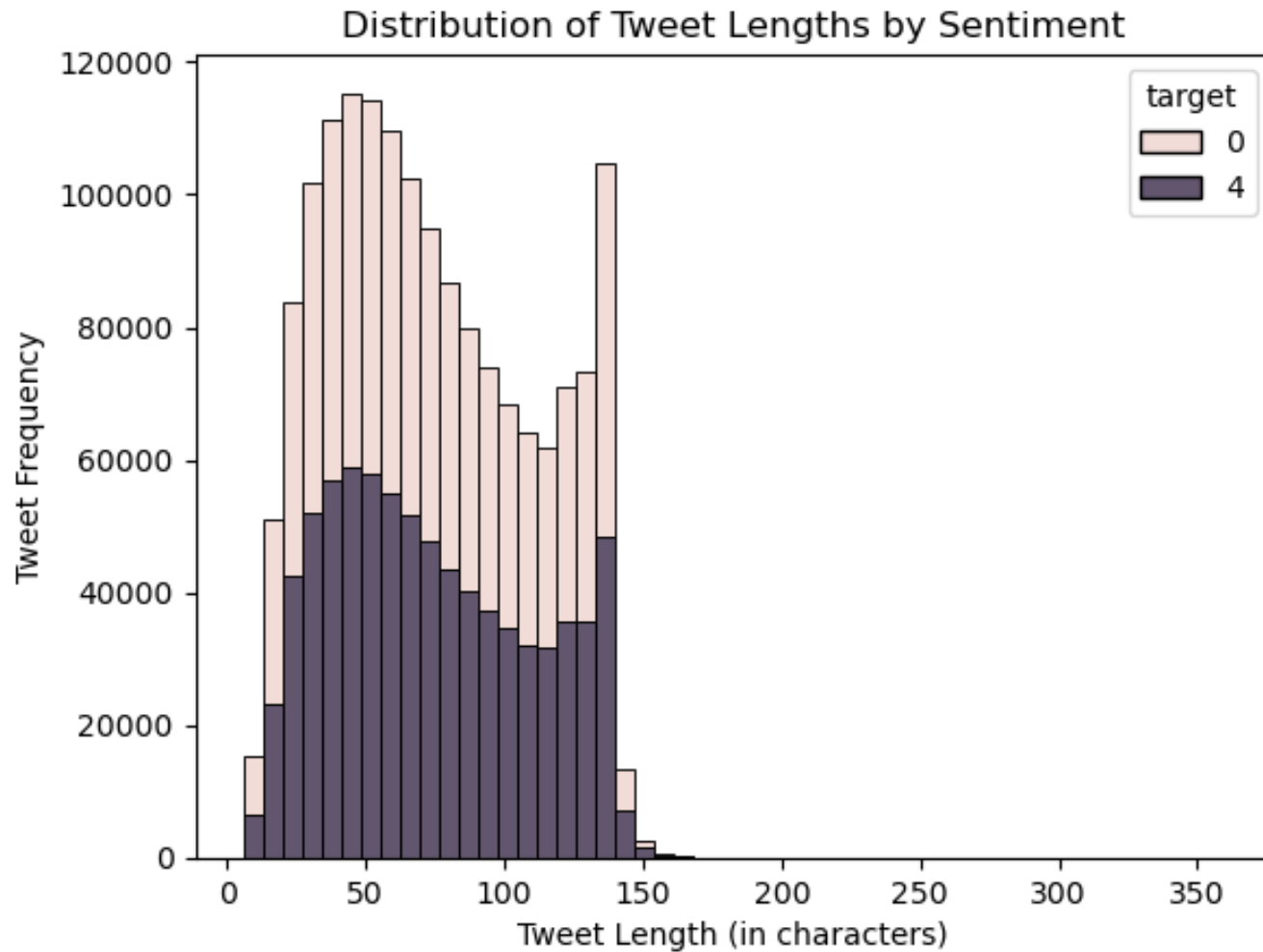
- 1.6 millions tweets from 2009
- Each tweet has
 - The Sentiment (between 0 = Positive and 4 Negative)
 - The tweet
 - The id of the tweet
 - The date of the tweet
 - The username of the author
- In practice only the first two features have been used
- In practice, each tweet have only 0 or 4 assigned



Some surprising EDA



- Some tweets have more than 140 characters !
- Some characters aren't well transformed in the data
- Some tweets can exceed 140 characters for commands and usernames
- Weird characters will be filtered during preprocessing



Tweet length
and
sentiment

Most used words per sentiment

Negative Tweets Word Cloud



Positive Tweets Word Cloud



Datapreprocessing

- Removes URLs
- Remove Mentions
- Clean special Characters & Digits
- Whitespace Normalization
- Tokenization & Filtering
- Stemming & Lemmatization



Original Tweet:
"@hater123 <http://x.com>
This is the worst experience
ever!"



Remove URLs and Mentions:
"This is the worst
experience ever!"



Remove Special Characters:
"This is the worst
experience ever"



Remove Stopwords (e.g.,
"this", "is", "the"):
"worst experience ever"



Apply
Stemming/Lemmatization:
"worst experi ever"

Baseline Models: Naive Bayes & Logistic Regression

Objective: Establish simple but effective baselines for tweet sentiment classification.

Selected Models:

- **Naive Bayes:** Assumes feature independence, fast and effective for high-dimensional sparse data like text.
- **Logistic Regression:** Learns a linear decision boundary; good balance between simplicity and performance.

Vectorization:

- **TF-IDF:** Converts text into numerical features by measuring term importance in each tweet.

Additional Features:

- Tweet length (character count)
- Hashtag count
- Mention count (@username)

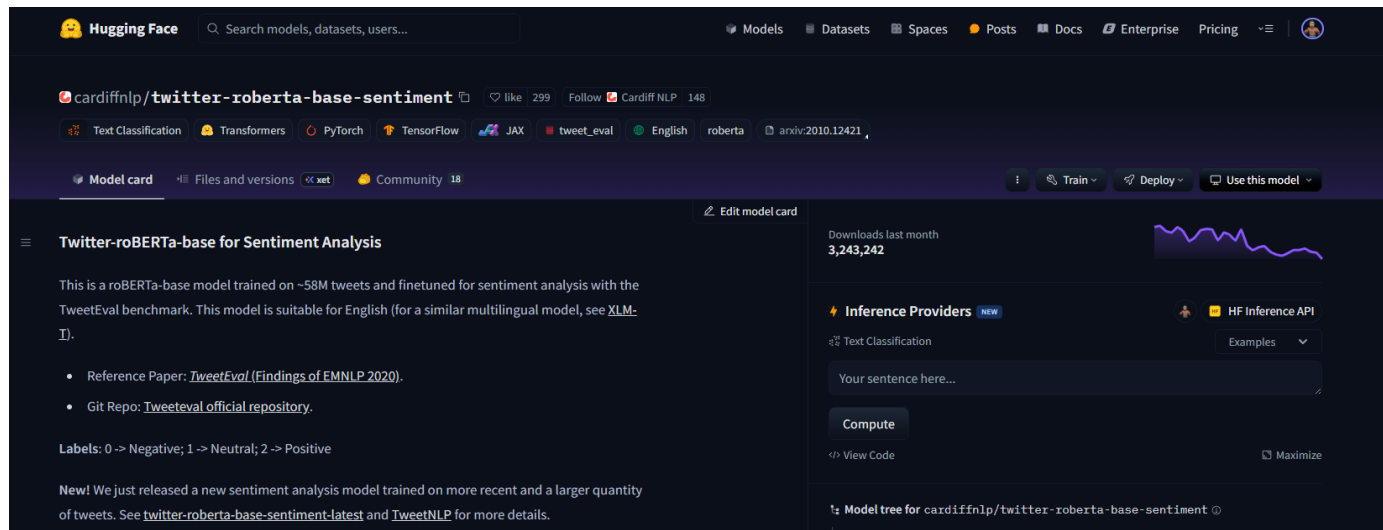
Train/Test Split:

- 80% training set
- 20% test set

Framework:

- Models implemented and evaluated using `scikit-learn`.

Roberta-finetuned



- ✓ Model: cardiffnlp/twitter-roberta-base-sentiment
- Based on RoBERTa-base architecture
- Pre-trained on 160M tweets for social media understanding
- Fine-tuned on the TweetEval sentiment dataset:
 - Labels: 0 = Negative, 1 = Neutral, 2 = Positive

Challenge in using Roberta



Custom Fine-Tuning

Original model
(cardiffnlp/twitter-
roberta-base-sentiment)
trained on 3 classes:






Negative (0),
Neutral (1),
Positive (2)

Our dataset contains
only binary labels:

0 = Negative,
1 = Positive

Evaluation of the model

Metric Breakdown

-  **Precision:** Out of the predicted labels, how many were actually correct?
Measures false positives (how many wrong positive predictions?)
-  **Recall:** Out of the actual ground truth, how many did we catch?
Measures false negatives (how many did we miss?)
-  **F1-Score:** Balance between Precision and Recall
Useful when classes are balanced, like in our case
-  **Support:** Number of real samples per class used in the test set
-  **Accuracy:** Overall correctness of predictions across all tw

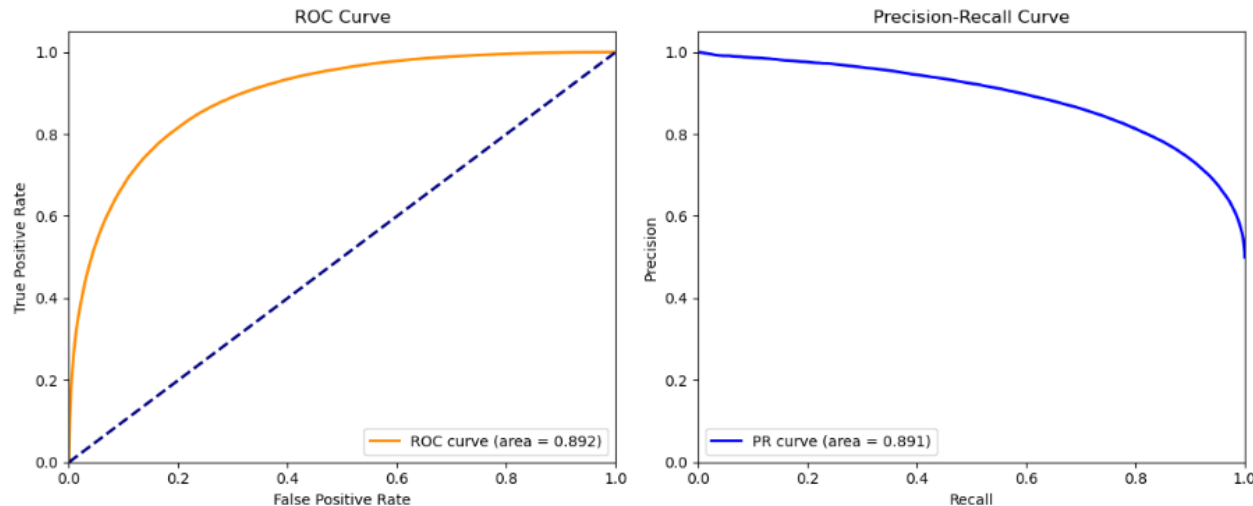
Sentiment Classifier Comparison (No Support)

Model	Sentiment	Precision	Recall	F1-Score	Accuracy
Naive Bayes	Negative	0.72	0.74	0.73	0.72
	Positive	0.73	0.70	0.72	
Logistic Regression	Negative	0.74	0.74	0.74	0.74
	Positive	0.74	0.74	0.74	
RoBERTa (fine-tuned)	Negative	0.80	0.82	0.81	0.81
	Positive	0.81	0.80	0.81	

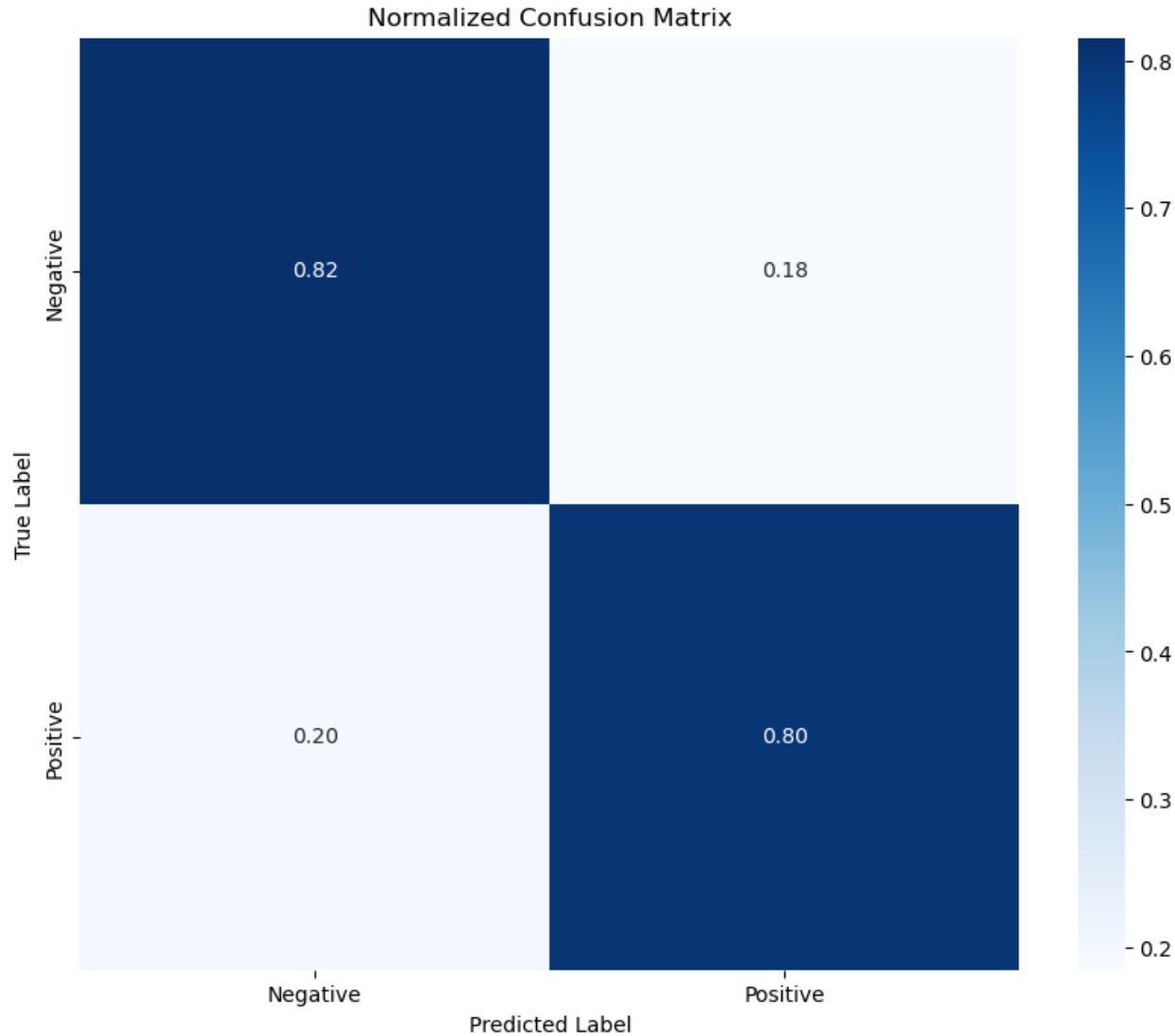
Insights:

- RoBERTa clearly outperforms traditional models across all metrics.
- Logistic Regression performs better than Naive Bayes but remains behind RoBERTa.
- Fine-tuning on a large dataset gives RoBERTa the edge in nuanced sentiment understanding.

ROC Curve / Precision-Recall Curve:



- **ROC Curve:**
 - Shows the trade-off between True Positive Rate and False Positive Rate.
 - AUC = **0.892** → indicates strong overall classification performance.
 - Curve significantly above the diagonal → model is much better than random guessing.
- **Precision-Recall Curve:**
 - Highlights the balance between Precision and Recall, useful in imbalanced datasets.
 - AUC = **0.891** → suggests good ability to detect positive cases while minimizing false positives.
 - Precision stays relatively high even as recall increases.



Confusion matrix

- Confusion Matrix Analysis

- True Negatives (Top-left, 0.82): 82% of actual negative tweets were correctly classified.
- False Positives (Top-right, 0.18): 18% of negative tweets were incorrectly classified as positive.
- False Negatives (Bottom-left, 0.20): 20% of positive tweets were misclassified as negative.
- True Positives (Bottom-right, 0.80): 80% of actual positive tweets were correctly predicted.

False Positives (actually positive, but labeled “negative”)

- “@billpalmer
welcome to the
new facebook!”

- Predicted: Positive
- True label: Negative
- Comment: Celebratory language (“awesome,” “beautiful”), with only a light-hearted reference to “no pot of gold.”
Hardly negative.

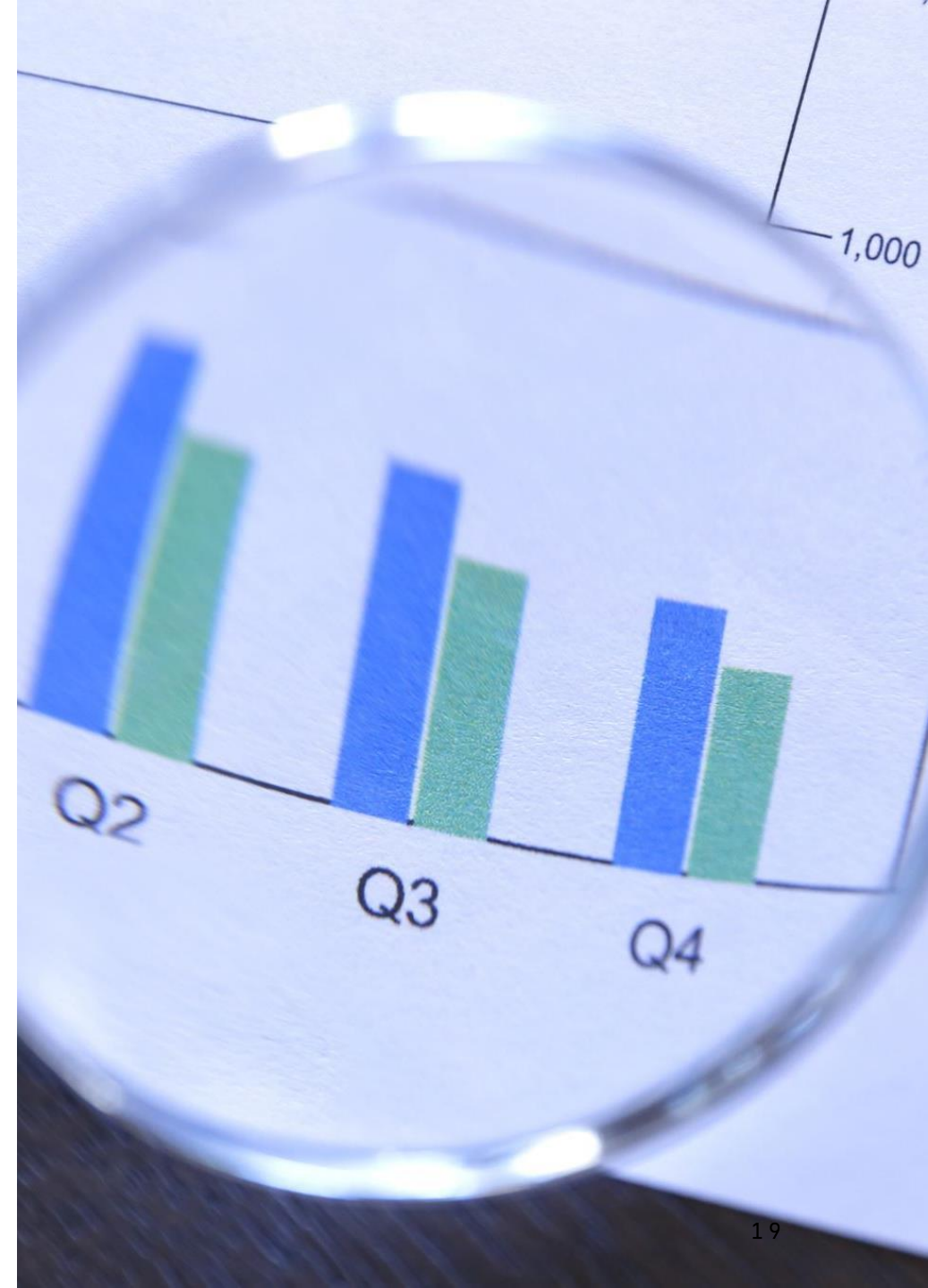
False Negatives (actually negative, but labeled “positive”)

- “I cut my hand while doing pull-ups, I did a few more sets after the minor injury.”

- Predicted: Negative
- True label: Positive
- **Why it’s likely mislabeled:** This text describes pain/injury. The overall tone is far from “positive” in the sentiment sense.

Dataset Limitations

- Dataset is outdated (collected several years ago)
- May contain labeling errors and inconsistencies
- Regular review and relabeling recommended



Model confidence

- **Traditional ML Models (LR & NB):**
 - **scikit-learn** → `model.predict_proba(X)` returns $[P(\text{neg}), P(\text{pos})]$
 - **joblib** for loading & saving models/vectorizers
 - **NumPy** & **scipy.sparse** for feature stacking
 - **Confidence** = $\max(P(\text{neg}), P(\text{pos}))$
- **RoBERTa Model:**
 - **PyTorch** → raw logits from forward pass
 - **Hugging Face Transformers** for tokenizer & model
 - Apply `torch.softmax(logits, dim=-1)` → $[P(\text{neg}), P(\text{pos})]$
 - **Confidence** = $\max(P(\text{neg}), P(\text{pos}))$

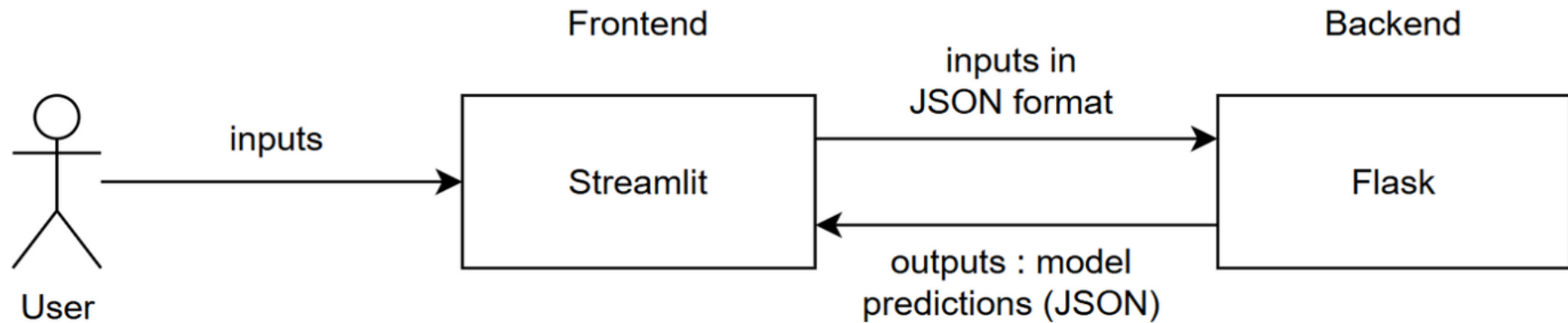
Roberta-fine tuned limitations

- ⚠ Label Reduction During Fine-Tuning
- To adapt the model to our binary classification use case (positive/negative), we **removed the "neutral" label** from the original 3-class setup.
- As a result, the model **tends to overconfidently classify texts as either positive or negative**, even when they may be neutral in tone.

Text	Original Model Prediction (3 labels)	Fine-Tuned Model Prediction (binary)
"Good night 😊"	1) Positive (0.8466) 2) Neutral (0.1458) 3) Negative (0.0076)	1) Positive (0.9270) 2) Negative (0.0730)
"This is the worst experience ever."	1) Negative (0.9766) 2) Neutral (0.0199) 3) Positive (0.0036)	1) Negative (0.9074) 2) Positive (0.0926)

API Implementation

- Usage of Flask and Streamlit
- Split backend and frontend
- Makes /predict endpoint available for other applications if needed



API Docker

- One docker per module
- One docker-compose.yml file to build both at the same time
- .yml file also makes the connection between those modules

API Deployment

```
docker build -t gcr.io/sentivista-453008/senti-api:latest ./flask
docker push      gcr.io/sentivista-453008/senti-api:latest

docker build -t gcr.io/sentivista-453008/senti-ui:latest ./streamlit
docker push      gcr.io/sentivista-453008/senti-ui:latest

gcloud run deploy senti-api --image gcr.io/sentivista-453008/senti-api:latest --platform
managed --allow-unauthenticated --port 5000

gcloud run deploy senti-ui --image gcr.io/sentivista-453008/senti-ui:latest --platform
managed --allow-unauthenticated --set-env-vars API_URL=https://senti-api-
24294949938.europe-west1.run.app --port 8501
```

Vertex AI

The goal is to :

- Load the data used for training
- Process the data
- Train the model
- Compute the different metrics



GitHub Actions Workflow – ci-cd.yml

- **Triggers:**

- On push to `main` or `ci-cd-setup`
- On pull requests to `main`
- Only for changes in README, API code, or tests

- **Jobs:**

- **Pre-commit:** code formatting with pre-commit hooks (Python 3.11)
- **Pytest:** runs tests after pre-commit, installs dependencies, downloads NLTK, runs tests/
- **Deploy (mock):** triggered on `main`, simulates deployment (no actual production push)

Purpose:

- Validates the `clean_tweet()` function
- Ensures correct preprocessing of:
 - URLs, @mentions, special characters, numbers
 - Space normalization
- Prevents broken inputs to sentiment models
- Acts as a safety net for text processing

Impact:

- Broken preprocessing = inaccurate predictions
- Reliable tests = higher model performance

Preprocessing Example – `clean_tweet()`

Original Tweet:

"@john123 I hated the service!! Worst ever <https://badreview.com>"

After `clean_tweet()`:

"hated the service worst ever"

Transformation Steps:

- Removed mention: @john123
- Removed punctuation: !!
- Removed emojis and link
- Lowercased and normalized spaces

If you want
to try it out





Conclusion

- Our deployment works
- User can choose between the three models
- User can rely on confidence percentage



Next steps

- Use a better dataset that can infer nuanced sentiments
- Do sentiments analysis on topics from batches of tweets
- Try other models

CI/CD

```
1  import os
2  import sys
3
4  sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), '..')))
5
6  from api.flask.app import clean_tweet
7
8
9  def test_clean_tweet():
10     # Test URL removal
11     assert "hello world" in clean_tweet("hello world https://example.com")
12
13     # Test mention removal
14     assert "@user" not in clean_tweet("hello @user")
15
16     # Test special character removal
17     assert "!" not in clean_tweet("hello world!")
18
19     # Test number removal
20     assert "123" not in clean_tweet("hello123")
21
22     # Test multiple spaces
23     assert "hello    world" not in clean_tweet("hello    world")
24     assert "hello world" in clean_tweet("hello    world")
25
```