SCHOOL OF INTELLIGENT ENGINEERING (FORMER SCHOOL OF MECHANICAL AND ELECTRICAL ENGINEERING), SHAOXING UNIVERSITY

AI-Powered Sentiment Analysis and Recommendation System for E-Commerce Reviews

Project Training Course Final Project

GORASH PLATON

Student ID: 22605107

This project develops an AI-powered sentiment analysis and recommendation system for e-commerce reviews. Different machine learning models used are Naive Bayes, SVM, and LSTM for the classification of sentiments into positive, negative, and neutral categories. The system also has a recommendation engine to recommend products based on user sentiment. A web interface based on Django has been provided for end-to-end interaction by the user, and MySQL is used for storing user data along with reviews. The project evaluates model performance across English and Spanish reviews, revealing challenges with neutral sentiment classification in Spanish and suggesting ways for improvement toward better multilingual support.

Introduction

Customer reviews are the lifeblood of modern e-commerce, shaping buyer decisions, surfacing product problems, and powering personalization. However, most platforms still rely on rudimentary signals-star ratings or keyword counts-that miss nuance, misread sarcasm/irony/idioms, and fail to connect opinion to product features. This project report describes the development of an **AI-Powered Sentiment Analysis and Recommendation System for E-Commerce Reviews** that turns raw review text into actionable insights: accurate sentiment labels, such as positive, negative, and neutral, as well as personalized product suggestions powered by user sentiment and behavior.

Using the public *Amazon Reviews dataset* and a *Django* web interface backed by *MySQL*, we will implement a complete end-to-end application suitable for a course project. Baseline classifiers (*TF-IDF + Logistic Regression / LinearSVC* or *Multinomial Naive Bayes*) provide robust and interpretable sentiment detection; optional *LSTM* experiments investigate gains from sequence models. Aspect extraction relies on *targeted NLP*, namely noun-phrase extraction and keyword matching, so the system links sentiment to product features without heavyweight models. Recommendations combine simple collaborative-filtering or content-based approaches with sentiment signals to produce more relevant suggestions.

The project follows *standard project management phases*: **Initiation**, **Planning**, **Execution**, **Monitoring & Control**, and **Closing**. Scope control, mitigation of risk, and reproducibility are checked. We also provide multilingual support for both English and Spanish through light translation or dual pipelines, and explicit fairness checks that detect performance disparities across languages or categories. The pragmatic ML choices, disciplined development, and straightforward NLP techniques demonstrated in this project significantly enhance review understanding and recommendation quality. This report discusses the design, implementation plan, evaluation strategy, and ethical safeguards for guiding the project from prototype to tested, maintainable application.

Methodology

1. **Initiation:** Define the project's goal, success metrics, scope, and constraints. Confirm the dataset and resource needs. Create an initial risk log covering data quality, model accuracy, and schedule concerns. Deliverables: project charter, risk log, timeline.

2. Planning: Break the work into tasks and create a sprint plan. Select the technology stack (Python, scikit-learn/TensorFlow, Pandas, NumPy, Django, MySQL, SQLAlchemy). Design the data schema and prepare UML diagrams. Outline the ML approach, including baseline models, optional LSTM testing, and multilingual handling. Set evaluation criteria. Deliverables: task list, UML/schema, environment checklist.

3. Execution: Build the data pipeline to load and clean the dataset and detect languages. Implement text preprocessing (tokenization, stop-word removal, TF-IDF). Train baseline models and run LSTM tests if needed, handling class imbalance. Implement aspect extraction using noun phrases and keywords. Build a simple recommender using collaborative or content-based filtering. Develop the Django web app and connect it to MySQL. Add unit and integration tests. Deliverables: trained models, endpoints, Django prototype, unit tests.

4. Monitoring: Track progress with status reports and sprint reviews. Use continuous integration to ensure code quality and testing. Log model runs and metrics. Monitor model performance across languages and categories. Address risks as they arise. Deliverables: status updates, logs, performance metrics, an updated risk log.

5. Closing: Run final evaluations and summarize results. Package the codebase, setup instructions, and database scripts. Conduct a project review and outline future improvements. Deliverables: final code, documentation, trained models, project review, slides.

Data

This project relies on the *Amazon Review Full Score Dataset Version 3 (September 2015),* adapted from a large corpus of Amazon product reviews originally collected by McAuley & Leskovec (2013). The larger Amazon dataset covers an 18-year timespan and includes roughly **35 million reviews** submitted through March 2013, each with product information, user ID, star rating, and text review. Based on this massive repository, Xiang Zhang et al. built a standardized text-classification benchmark for machine-learning research (Zhang et al., 2015).

Reviews were sampled to ensure a balanced representation throughout the full 1-to-5 rating scale. For each of the five rating categories, specifically, **600,000 training examples** and **130,000 testing examples** were randomly selected, resulting in total of **3 million training samples** and **650,000 testing samples**. Each instance in the dataset includes three fields:

- *numerical rating label* (1–5, where 1 is negative and 5 is positive);
- *review title*;
- *review body*.

The dataset contains millions of customer reviews over various product types and different writing styles. Most are in English, but several others are in other languages, including Spanish. Text fields are quoted and follow standard CSV escaping conventions, such as internal quotation marks doubled and line breaks encoded as "\". It is designed in such a way that it is suited for simple sentiment classification but can also be used to study multilingual processing and model consistency across languages, forming a good middle ground between simple examples and large, complex datasets.

	A	B	C
1	1	mens ultrasheer	This model may be ok for sedentary types, but I'm active and get around alot in my job - co
2	4	Surprisingly delightful	This is a fast read filled with unexpected humour and profound insights into the art of poli
3	2	Works, but not as advertised	I bought one of these chargers..the instructions say the lights stay on while the battery cha
4	2	Oh dear	I was excited to find a book ostensibly about Muslim feminism, but this volume did not live
5	2	Incorrect disc!	I am a big JVC fan, but I do not like this model, I was suspiscious when I saw several units i
6	2	Incorrect Disc	I love the style of this, but after a couple years, the DVD is giving me problems. It doesn't e
7	2	DVD menu select problems	I cannot scroll through a DVD menu that is set up vertically. The triangle keys will only sele
8	3	My 2 y/o grandson loves it!!	This movie with all of its animals really keeps my grandson occupied when I'm babysitting.
9	5	A Cookbook Every Baker Should Own	I found a copy of this cookbook at a local used book store and mixed up a sourdough start
10	3	good basic	The book is a basic "how to" book for using sourdough. The author obviously has lots of ex
11	3	nice screen for a nice price but....	I compared a few different flat panels with review before I narrowed down my pick, which
12	3	Poor maps, no hostels	It's a good book, but the maps are not very good, covering only parts of central areas and t
13	2	Profound then. Truly horrible now.	The narrative style of this work by famous founders' biographer John Morse is arrogant, fl
14	1	A complete Bust	This game requires quicktime 5.0 to work...if you have a better version of quicktime (I have
15	5	Barbie as Rapunzel: A Creative Adventure	I purchased this software for my 5 year old granddaughter and she loves playing it so muc
16	5	Best Game for Young Girls	My daughter absolutely loves this game! It's easy to use and requires no adult supervision
17	2	Not so good - ok for a rental!	This game is pretty exciting and it is very nice and it is really creative and one of a kind. The
18	1	NOT OS X but MAC CLASSIC	NEGATIVE: This game does not work with OS X and requires the CD so you can't have it on
19	4	Good Program, Lots of Flexibility	My 4year old really likes this game. She's not addicted to it but loves the style of the progra
20	3	A Blast From My Past	I once purchased a 12" of Jesse Rae's song RUSHA. I really enjoyed it and thought it might l
21	1	Very disappointed!	This perfume is just AWFUL! Smells nothing like freesia.The gift recipient was not impresse
22	5	Filled from cover to cover with practical guidance	Written by Elizabeth Crary (a parent educator of 25 years' experience), Dealing With Disap
23	5	Hardest Pit in the South	Big Pokey is one of the best rappers out of the South. No one can do it like him. This album
24	5	Amazin	This is an amazing blend of hot beats and a good rapper. It's underground, so many of you

Figure 1: Preview of the dataset (CSV table format)

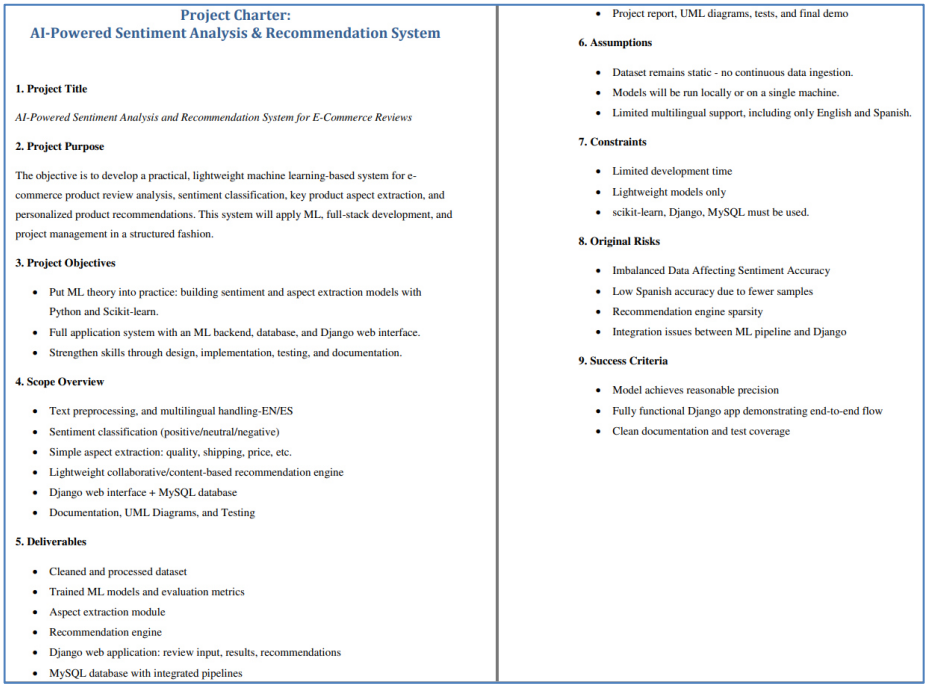
Initiation

Risk Description	Impact	Likelihood	Mitigation / Action
Noisy or incomplete review data	Medium	Low	Apply cleaning, deduplication, and validation checks
Imbalanced sentiment classes	High	High	Use class weighting, oversampling, and monitor per-class metrics
Lower accuracy on Spanish reviews	Medium	Medium	Use language detection; separate or translation-based pipeline
Model underperforms or overfits	High	Medium	Use cross-validation, simpler models first, track validation metrics
Weak recommendation quality	Medium	Medium	Tune similarity scoring; use popularity fallback
Django app slow due to model size or queries	Medium	Low	Cache model, optimize DB, add indexes
Time/resources insufficient for advanced features	High	High	Prioritize core tasks; freeze scope early
Fairness issues across languages or categories	High	Medium	Evaluate fairness; retrain if disparities exceed threshold
Library or version conflicts	Medium	Medium	Pin versions (`pip freeze`)
Database integration errors or migration	Medium	Medium	Test schema early; validate migrations

Figure 2: Risk Register Table

	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 5
INITIATION Project setup & charter Dataset acquisition Risk register					
PLANNING Requirements & scope ML/DB/Django design UML + architecture					
ML Data preprocessing Language detection (EN/ES) TF-IDF baseline models Model tuning & evaluation					
WEB + DB MySQL integration Web UI (review + results) Recommendation engine Integration of ML → Django					
TESTING Unit tests Performance checks					
MONITORING Experiment tracking Fairness checks (EN/ES) Issue triage & revisions					
CLOSING Final documentation Demo & presentation					

Figure 3: Gantt Chart Timeline



Project Charter:
AI-Powered Sentiment Analysis & Recommendation System

1. Project Title

AI-Powered Sentiment Analysis and Recommendation System for E-Commerce Reviews

2. Project Purpose

The objective is to develop a practical, lightweight machine learning-based system for e-commerce product review analysis, sentiment classification, key product aspect extraction, and personalized product recommendations. This system will apply ML, full-stack development, and project management in a structured fashion.

3. Project Objectives

- Put ML theory into practice: building sentiment and aspect extraction models with Python and Scikit-learn.
- Full application system with an ML backend, database, and Django web interface.
- Strengthen skills through design, implementation, testing, and documentation.

4. Scope Overview

- Text preprocessing, and multilingual handling-EN/ES
- Sentiment classification (positive/neutral/negative)
- Simple aspect extraction: quality, shipping, price, etc.
- Lightweight collaborative/content-based recommendation engine
- Django web interface + MySQL database
- Documentation, UML Diagrams, and Testing

5. Deliverables

- Cleaned and processed dataset
- Trained ML models and evaluation metrics
- Aspect extraction module
- Recommendation engine
- Django web application: review input, results, recommendations
- MySQL database with integrated pipelines
- Project report, UML diagrams, tests, and final demo

6. Assumptions

- Dataset remains static - no continuous data ingestion.
- Models will be run locally or on a single machine.
- Limited multilingual support, including only English and Spanish.

7. Constraints

- Limited development time
- Lightweight models only
- scikit-learn, Django, MySQL must be used.

8. Original Risks

- Imbalanced Data Affecting Sentiment Accuracy
- Low Spanish accuracy due to fewer samples
- Recommendation engine sparsity
- Integration issues between ML pipeline and Django

9. Success Criteria

- Model achieves reasonable precision
- Fully functional Django app demonstrating end-to-end flow
- Clean documentation and test coverage

Figure 4: Project Charter Overview

Planning

Planning	• Define the system requirements and features • Design ML pipeline and multilingual strategy • Design database schema and Django architecture • Create UML diagrams and sprint plan.
Data Processing	• Load and inspect dataset • Clean data: remove duplicates, fix missing values • Detect language, preprocess text - EN/ES • Optimization and downsampling
Machine Learning	• Implement TF-IDF + baseline models (NB, Logistic/LinearSVC) • LSTM experiment • Implement aspect extraction module • Save trained models and vectorizers
Recommendation System	• Build user–product interaction matrix • Implement content-based filtering • Precision evaluation and parameter tuning
Web & Database Development	• Django project setup and apps • Create MySQL database and integrate with Django ORM • Build review submission page and sentiment result page • Integrate ML pipeline into Django backend • Add recommendations view and basic UI templates
Testing	• Write unit tests for preprocessing, models and recommender • Perform simple performance testing
Monitoring & Control	• Track model experiments and metrics • Run fairness checks – English vs Spanish • Fix bugs and issues
Conclusion	• Finalize documentation • Prepare final demo and presentation • Conduct project retrospective and sign-off

Figure 5: Task List

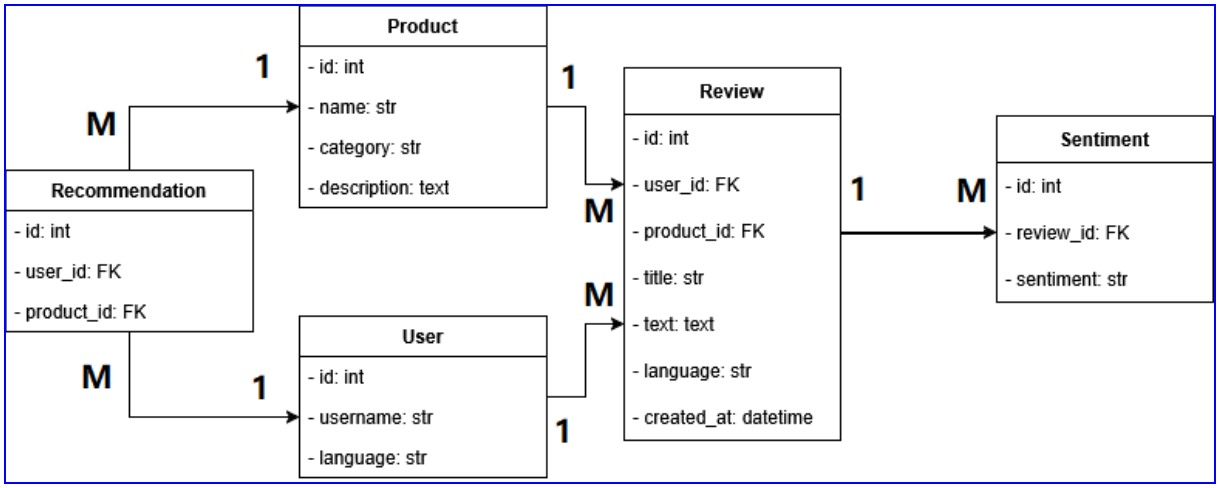
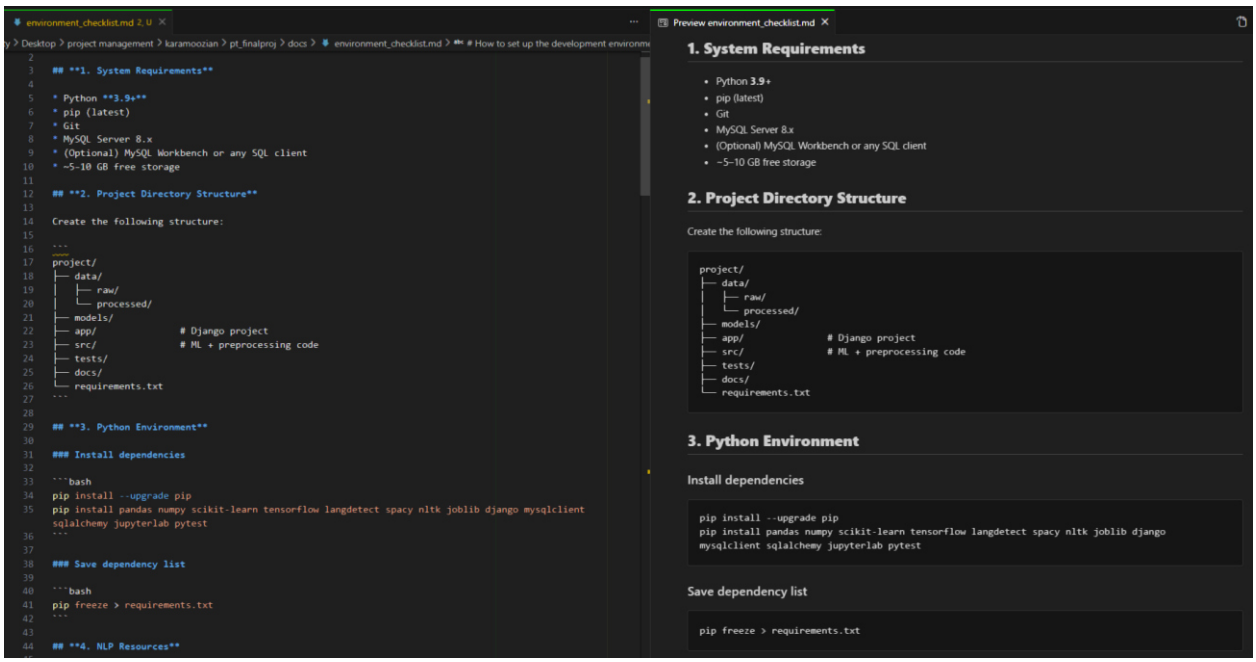


Figure 6: Database UML Scheme



1. System Requirements

- Python 3.9+
- pip (latest)
- Git
- MySQL Server 8.x
- (Optional) MySQL Workbench or any SQL client
- ~5–10 GB free storage

2. Project Directory Structure

Create the following structure:

```
project/
├── data/
│   ├── raw/
│   └── processed/
├── models/
├── app/              # Django project
├── src/              # ML + preprocessing code
├── tests/
├── docs/
└── requirements.txt
```

3. Python Environment

Install dependencies

```
pip install --upgrade pip
pip install pandas numpy scikit-learn tensorflow langdetect spacy nltk joblib django
mysqlclient sqlalchemy jupyterlab pytest
```

Save dependency list

```
pip freeze > requirements.txt
```

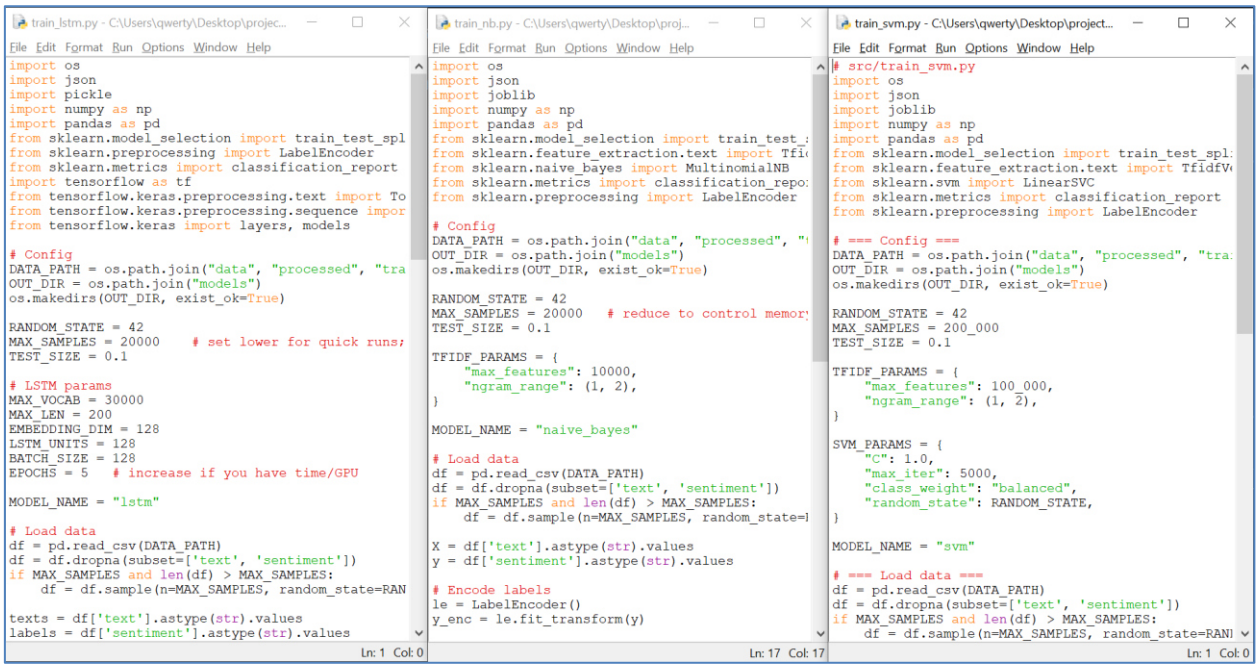
Figure 7: Environment Checklist in Markdown Format

Machine Learning Implementation

File Edit Format Run Options Window Help

```
import pandas as pd
import os
import re
from langdetect import detect, DetectorFactory
DetectorFactory.seed = 0

CHUNKSIZE = 300

def clean_text(s):
    if not isinstance(s, str):
        return ""
    s = s.replace("\\n", " ").replace("\\"+"n", " ")
    s = re.sub(r'\s+', ' ', s).strip()
    return s

def map_score_to_sentiment(score):
    score = int(score)
    if score <= 2:
        return "negative"
    elif score == 3:
        return "neutral"
    else:
        return "positive"

def preprocess_file(input_path, output_path, iterations=None):
    if os.path.exists(output_path):
        raise FileExistsError()
    print("Processing", output_path + ":")
    os.makedirs(os.path.dirname(output_path), exist_ok=True)
    first = True
    for i, chunk in enumerate(pd.read_csv(input_path, header=None, names=['score','title','text'],
                                          chunksize=CHUNKSIZE, quoting=1, engine='python')):
        # combine title+text
        chunk['text'] = (chunk['title'].fillna('') + ". " + chunk['text'].fillna('')).apply(clean_text)
        chunk = chunk[['score','text']]
        # deduplicate
        chunk = chunk.drop_duplicates(subset=['text'])
        # detect language
        langs = []
        for txt in chunk['text'].tolist():
```

Figure 8: Data Preprocessing



train_lstm.py - C:\Users\qwerty\Desktop\projec...

```
import os
import json
import pickle
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_spl
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report
import tensorflow as tf
from tensorflow.keras.preprocessing.text import To
from tensorflow.keras.preprocessing.sequence impor
from tensorflow.keras import layers, models

# Config
DATA_PATH = os.path.join("data", "processed", "tra
OUT_DIR = os.path.join("models")
os.makedirs(OUT_DIR, exist_ok=True)

RANDOM_STATE = 42
MAX_SAMPLES = 20000    # set lower for quick runs;
TEST_SIZE = 0.1

# LSTM params
MAX_VOCAB = 30000
MAX_LEN = 200
EMBEDDING_DIM = 128
LSTM_UNITS = 128
BATCH_SIZE = 128
EPOCHS = 5   # increase if you have time/GPU

MODEL_NAME = "lstm"

# Load data
df = pd.read_csv(DATA_PATH)
df = df.dropna(subset=['text', 'sentiment'])
if MAX_SAMPLES and len(df) > MAX_SAMPLES:
    df = df.sample(n=MAX_SAMPLES, random_state=RAN

texts = df['text'].astype(str).values
labels = df['sentiment'].astype(str).values
```

Ln: 1 Col: 0

train_nb.py - C:\Users\qwerty\Desktop\proj...

```
import os
import json
import joblib
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_
from sklearn.feature_extraction.text import Tfi
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_repo
from sklearn.preprocessing import LabelEncoder

# Config
DATA_PATH = os.path.join("data", "processed", "
OUT_DIR = os.path.join("models")
os.makedirs(OUT_DIR, exist_ok=True)

RANDOM_STATE = 42
MAX_SAMPLES = 20000   # reduce to control memory
TEST_SIZE = 0.1

TFIDF_PARAMS = {
    "max_features": 10000,
    "ngram_range": (1, 2),
}

MODEL_NAME = "naive_bayes"

# Load data
df = pd.read_csv(DATA_PATH)
df = df.dropna(subset=['text', 'sentiment'])
if MAX_SAMPLES and len(df) > MAX_SAMPLES:
    df = df.sample(n=MAX_SAMPLES, random_state=

X = df['text'].astype(str).values
y = df['sentiment'].astype(str).values

# Encode labels
le = LabelEncoder()
y_enc = le.fit_transform(y)
```

Ln: 17 Col: 17

train_svm.py - C:\Users\qwerty\Desktop\project...

```
# src/train_svm.py
import os
import json
import joblib
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_spl
from sklearn.feature_extraction.text import TfidfV
from sklearn.svm import LinearSVC
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder

# === Config ===
DATA_PATH = os.path.join("data", "processed", "tra
OUT_DIR = os.path.join("models")
os.makedirs(OUT_DIR, exist_ok=True)

RANDOM_STATE = 42
MAX_SAMPLES = 200_000
TEST_SIZE = 0.1

TFIDF_PARAMS = {
    "max_features": 100_000,
    "ngram_range": (1, 2),
}

SVM_PARAMS = {
    "C": 1.0,
    "max_iter": 5000,
    "class_weight": "balanced",
    "random_state": RANDOM_STATE,
}

MODEL_NAME = "svm"

# === Load data ===
df = pd.read_csv(DATA_PATH)
df = df.dropna(subset=['text', 'sentiment'])
if MAX_SAMPLES and len(df) > MAX_SAMPLES:
    df = df.sample(n=MAX_SAMPLES, random_state=RAN
```

Ln: 1 Col: 0

Figure 9: LSTM, Naive Bayes and SVM Model Creation

```
import spacy
from collections import Counter
nlp = spacy.load("en_core_web_sm")

def extract_aspects(text, topn=5):
    doc = nlp(text)
    # noun chunks + nouns
    aspects = [chunk.text.lower() for chunk in doc.noun_chunks]
    aspects += [token.text.lower() for token in doc if token.pos_ == 'NOUN']
    c = Counter(aspects)
    return [a for a,_ in c.most_common(topn)]

# Example:
# print(extract_aspects("The battery life is great but the shipping was slow. The screen quality is fantastic."))
```

Figure 10: Aspect Extractor Experiment

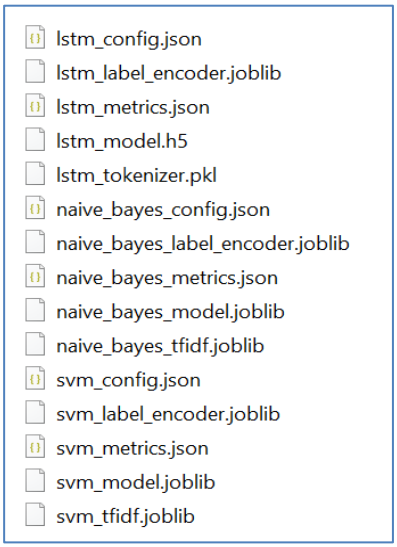
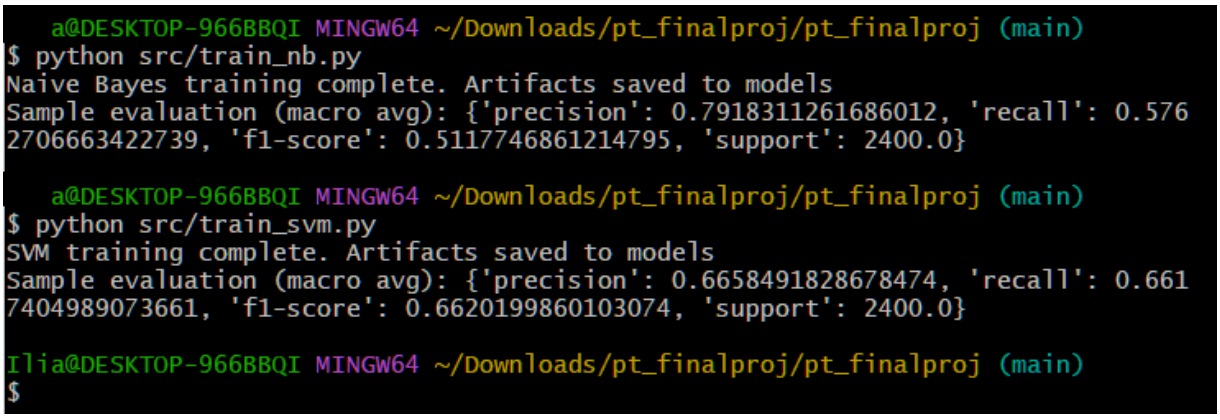


Figure 11: Trained models, metrics and tokenizers



```
a@DESKTOP-966BBQI MINGW64 ~/Downloads/pt_finalproj/pt_finalproj (main)
$ python src/train_nb.py
Naive Bayes training complete. Artifacts saved to models
Sample evaluation (macro avg): {'precision': 0.7918311261686012, 'recall': 0.576
2706663422739, 'f1-score': 0.5117746861214795, 'support': 2400.0}

a@DESKTOP-966BBQI MINGW64 ~/Downloads/pt_finalproj/pt_finalproj (main)
$ python src/train_svm.py
SVM training complete. Artifacts saved to models
Sample evaluation (macro avg): {'precision': 0.6658491828678474, 'recall': 0.661
7404989073661, 'f1-score': 0.6620199860103074, 'support': 2400.0}

Ilia@DESKTOP-966BBQI MINGW64 ~/Downloads/pt_finalproj/pt_finalproj (main)
$
```

Figure 12: Naive Bayes and SVM Model Metrics

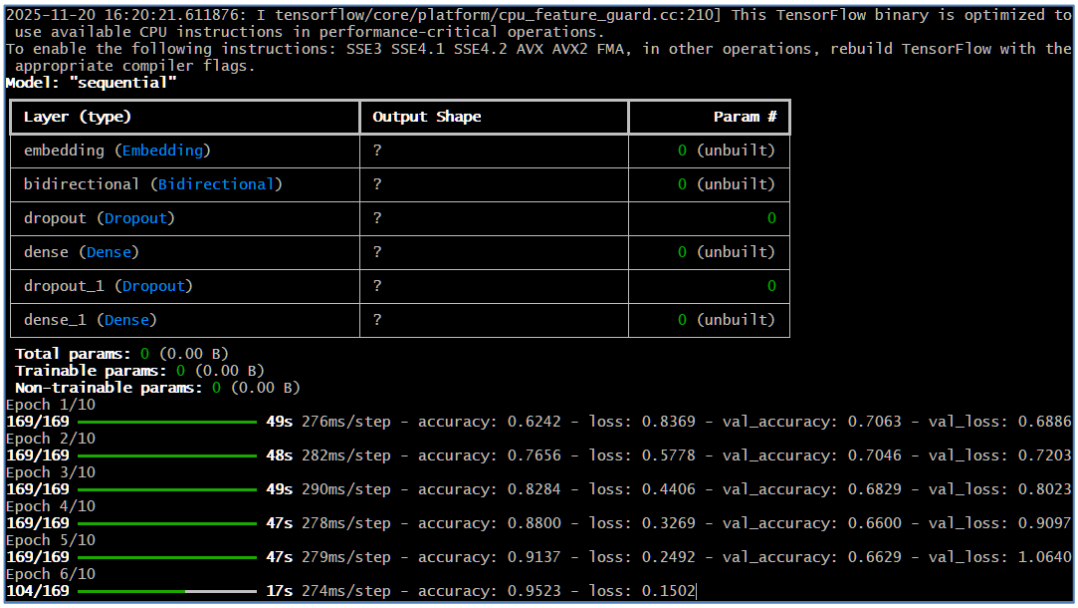
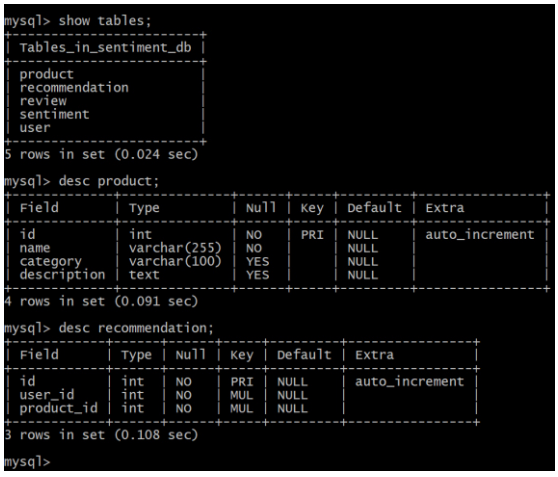


Figure 13: Tensorflow ML Training Process

Database development



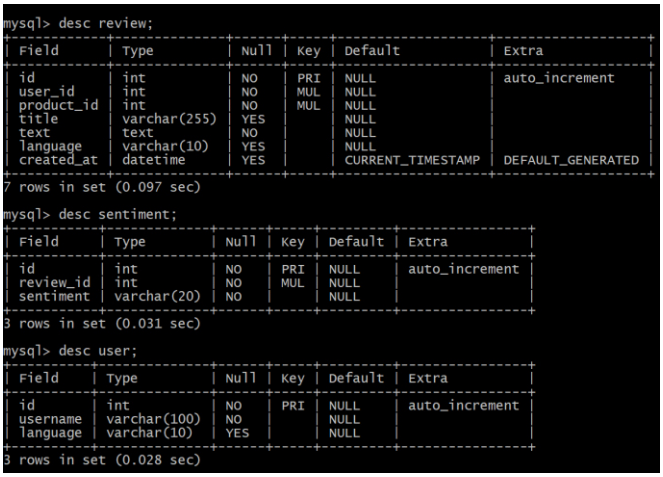


Figure 14, 15: MySQL Database Structure

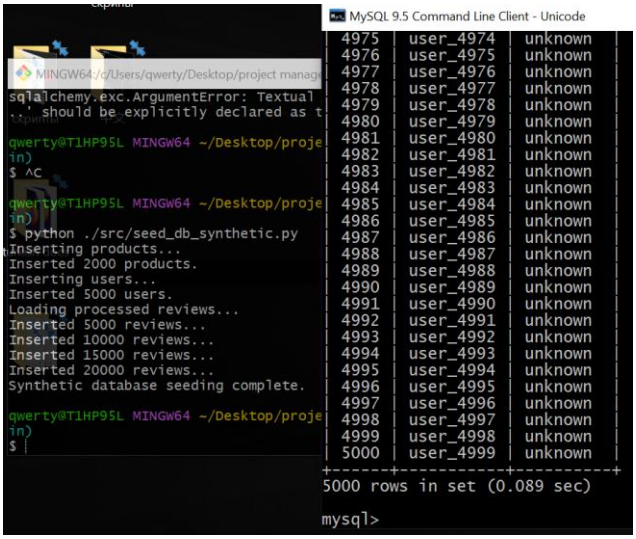


Figure 16: Connecting the MySQL Database to Django and Synthetically Populating the Database

Web App Implementation

```python
from django.db import models

class Category(models.Model):
    name = models.CharField(max_length=100, unique=True)
    slug = models.SlugField(max_length=120, unique=True, blank=True)
    description = models.TextField(blank=True)
    class Meta:
        db_table = "Category"
        ordering = ["name"]
    def __str__(self):
        return self.name


class Product(models.Model):
    name = models.CharField(max_length=255)
    slug = models.SlugField(max_length=255, unique=True, blank=True)
    category = models.ForeignKey(Category, on_delete=models.SET_NULL, null=True,
    description = models.TextField(blank=True)
    price = models.DecimalField(max_digits=10, decimal_places=2, null=True, blan
    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        db_table = "Product"
        ordering = ["-created_at"]

    def __str__(self):
        return self.name

    @property
    def short_description(self):
        if not self.description:
            return ""
        return (self.description[:120] + "...") if len(self.description) > 120 e

class Recommendation(models.Model)
    rec_id = models.IntegerField()
```

Figure 17: Implementing Django's Models using OOP Principles to Connect with the MySQL Database

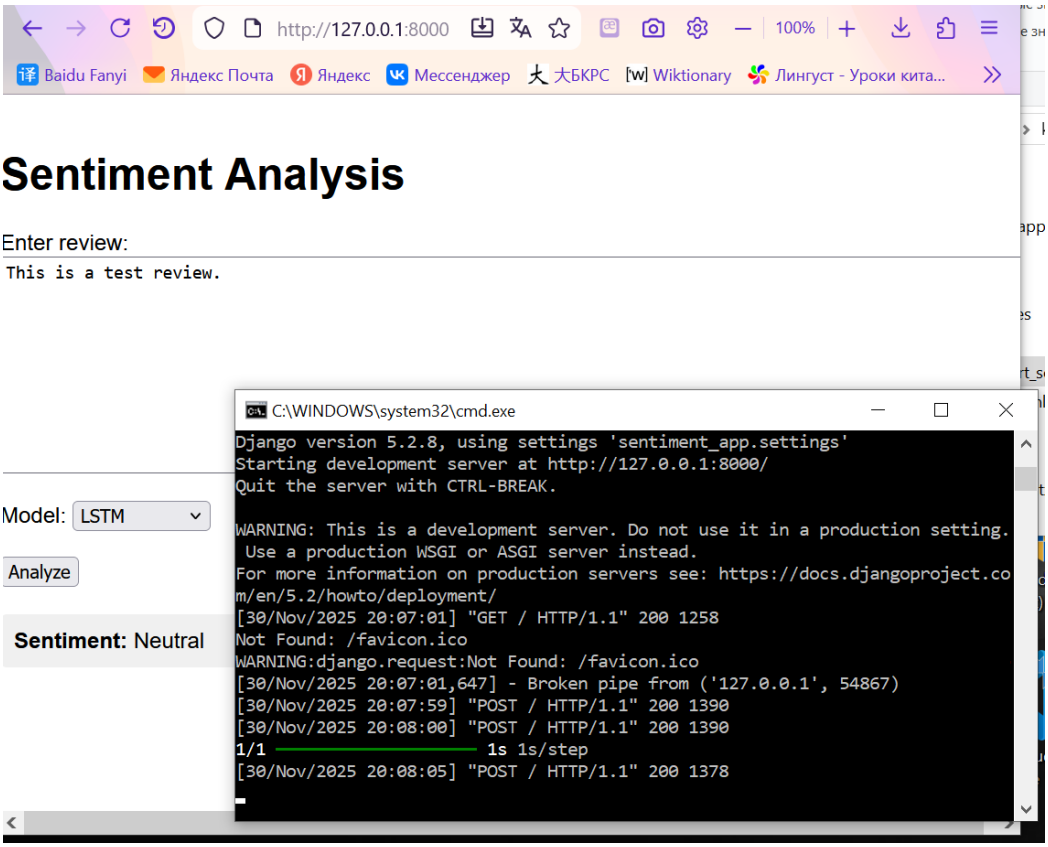


Figure 18: Locally Running Django Server and Homepage Visualization

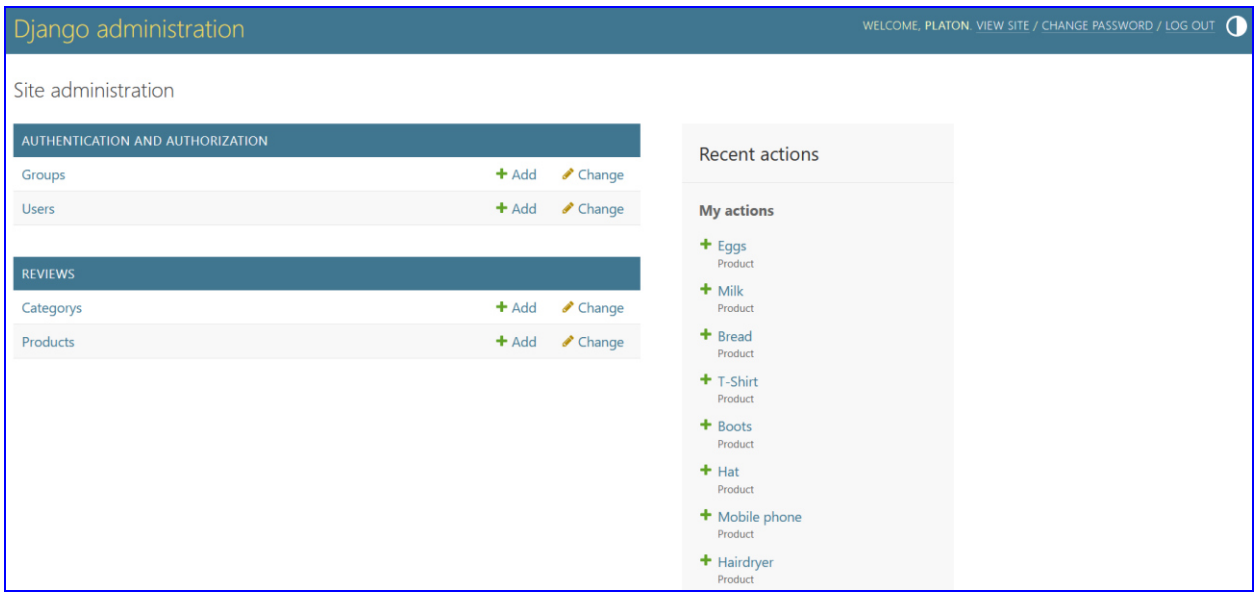


Figure 19: Django Administration Panel (added product data visible to the right)

Figure 20: Positive Analysis Result

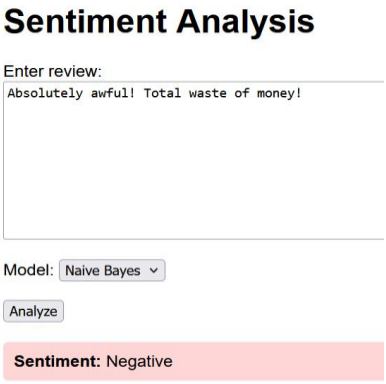


Figure 21: Negative Analysis Result

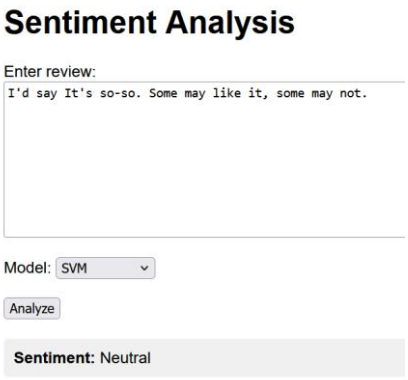


Figure 22: Neutral Analysis Result

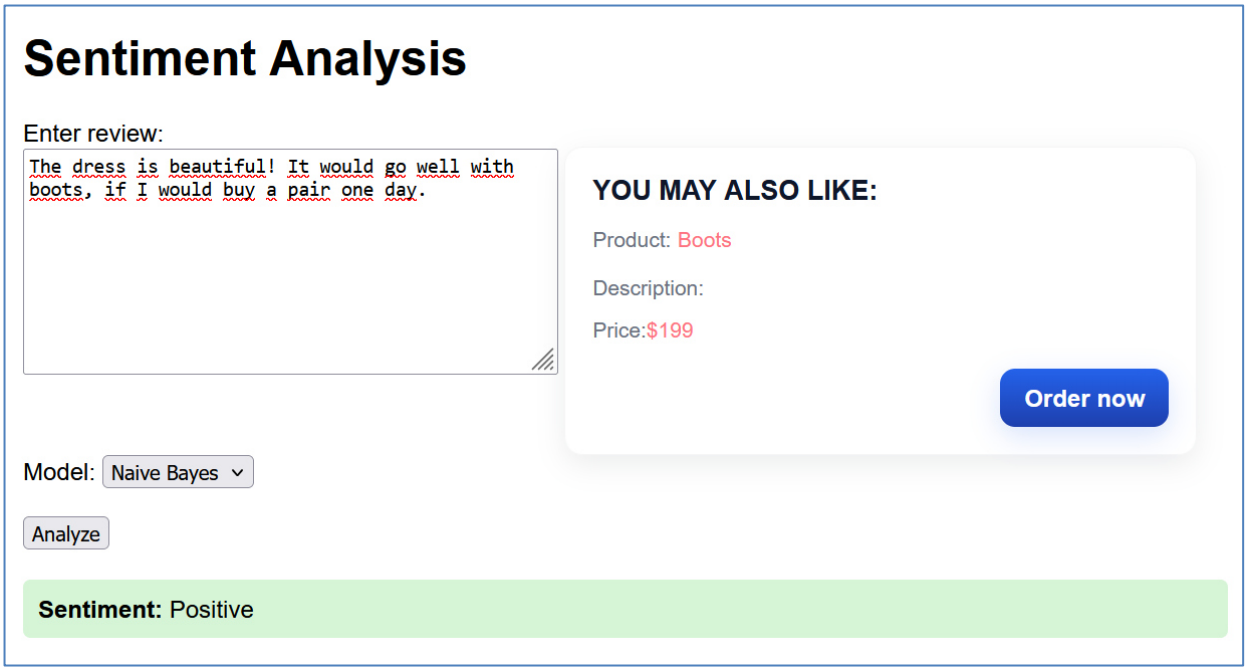


Figure 23: Product Recommendation System Visualization

Testing and Analysis

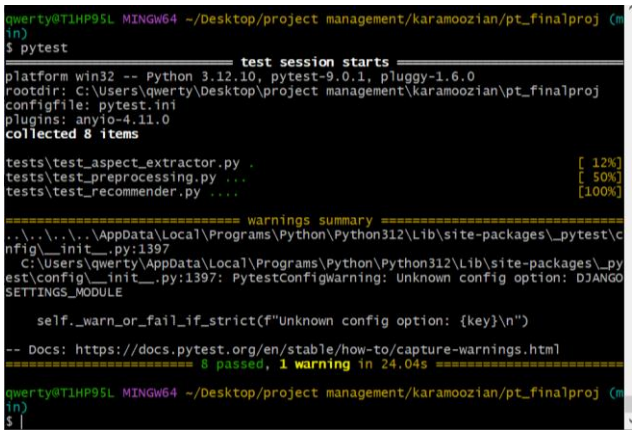


Figure 24: Running Unit Tests

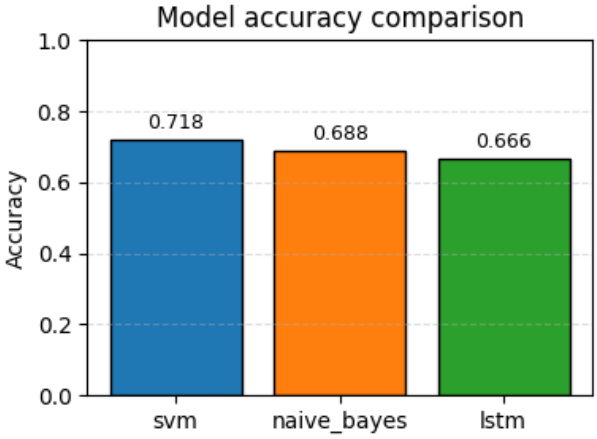
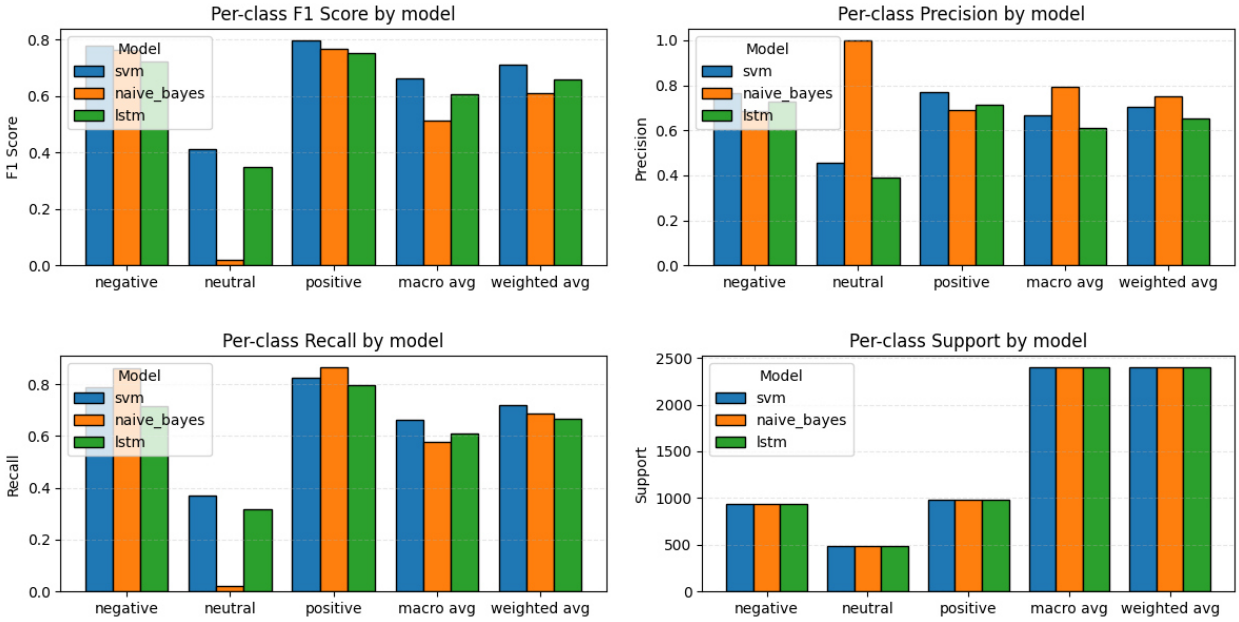


Figure 25: Models Accuracy Comparison



Figures 26–29: Model Metrics Comparison (note anomalies in *naive_bayes/neutral*)

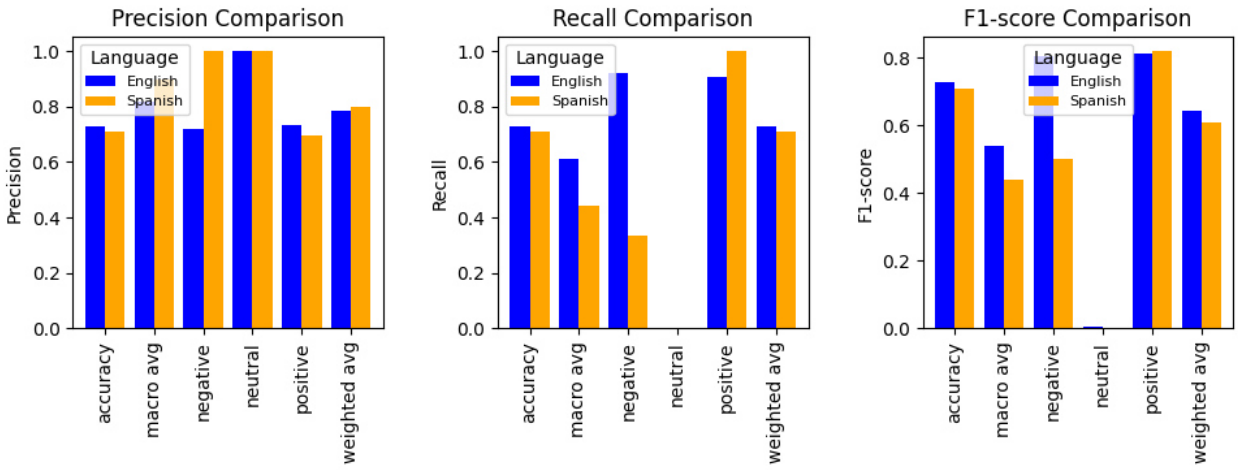


Figure 30: English versus Spanish Fairness Comparison (also note anomalies for the *neutral* class)

A score comparison of F1, precision, recall, and support between English and Spanish reviews shows differences, suggesting possible issues with model output, data spread, or how features are shown. Overall, the models did poorer on the Spanish reviews than the English ones, mostly for the neutral class.

The F1 scores for neutral reviews are very low, especially in Spanish (0.004 for Naive Bayes and 0.349 for LSTM), and precision and recall metrics also differ a lot for both languages. These issues likely arise from unequal class sizes or poor feature representation, with the mass of reviews either rated negatively or positively, not right in the middle. To counter this, class resampling could be attempted by either oversampling neutral reviews or changing class weights, as well as improving feature extraction through TF-IDF with n-grams or other methods. An alternative for such datasets is simply working only with two classes – negative and positive – throughout the whole pipeline.

The Spanish data is very uneven, from 3 reviews for the negative class to 5 for neutral and 16 for positive. This makes the model do poorer on this subset. Also, the model's support values are low for Spanish, with poorer evaluation metrics, especially for the neutral reviews. This also appears in worse macro and weighted averages for precision, recall, and F1-score for Spanish. These differences tell us more work is needed to balance the data or change model settings to deal with languages that have fewer samples. Adjusting models in multilingual tasks using cross-lingual embeds or specific training data might increase output across languages.

Workflow Overview

The first step was to gather and clean e-commerce review data. This involved preprocessing, data manipulation and tokenizing the text. Then, we trained three ML models – Naive Bayes, SVM, and LSTM – for sentiment analysis. We checked how well they worked on both English and Spanish data. After that, we built a recommendation system that suggests products based on what users think. Next, we created a web interface using Django, which lets users enter reviews, see sentiment results, and get product suggestions. The system used a MySQL database to keep track of user data, reviews, and recommendations. Several unit tests were written and evaluated, both for the machine learning pipeline and the Django app. We managed the project in stages and tried to make the models work better and solve language-related sentiment analysis problems. In the end, we showed the results visually and looked for bias to make sure the system worked fairly for different languages.

Results

In this work, we have developed an AI-driven sentiment analysis and recommendation system for e-commerce reviews by meeting the primary objectives of processing user reviews, classifying sentiment, and giving out personalized product recommendations. The approach was based on machine learning techniques that classify sentiments into positive, negative, or neutral using models such as Naive Bayes, SVM, and LSTM. Besides, we incorporated a content-based recommendation engine that suggested products based on user sentiment. The web interface was designed in Django; users can input their reviews, view sentiment analysis results, and get personalized recommendations for products. Another implemented part is the MySQL database for the storage of reviews, data about users, and recommendations.

The project was developed following a structured project management methodology, taking it through the distinct phases of initiation, planning, execution, monitoring, and closing. In the data collection and preprocessing, we handle missing values, remove duplicates, and tokenize the text data. For sentiment analysis, we train multiple machine learning models, evaluate their performance, and integrate the models into the web interface. Testing was extensive, and it included unit and integration tests to ensure that the system was robust and stable. In fact, our efforts were dedicated to obtaining high accuracy in classifying sentiments and making the user experience seamless.

The recommendation engine was able to deliver the desired result, making personalized suggestions using sentiment analysis. However, test results of models did not reflect the same performance on all languages. The test results were good, expect for the performance on neutral sentiments, which clearly indicates an issue of class imbalance, feature extraction, or lack of training data, especially regarding reviews in Spanish. Checks for fairness were conducted, and indeed the performance came out biased toward the English language, indicating areas of improvement in the multilingual application.

During development, we have investigated multilingual sentiment analysis, including bias detection, but some challenges remained because of the imbalanced dataset. We also created visualizations to compare model performance across languages and metrics, giving us valuable insights into the strengths and weaknesses of the system. The core system worked well on the local environment. Eventually, we developed a prototype that worked and demonstrated how real-time sentiment analysis and personalized recommendations could be possible, with the potential for further improvements, especially in the area of handling multilingual data.

Contribution

This project delves into e-commerce through its creation of an AI sentiment analysis and product recommendation system. It works by processing customer reviews to classify feelings and giving shoppers focused suggestions. The system uses popular machine learning options like the Naive Bayes, SVM, and LSTM models, aloowin understand user feelings and make better, individualized suggestions. The work looks into the problems of analyzing feelings in multiple languages, especially differences in results between English and Spanish. It suggests ways to make the system fairer and more correct for different languages. The system shows how machine learning can be used in practical ways for e-commerce internationally.

Conclusion

The AI-powered sentiment analysis and recommendation system developed in this project offers a solid solution to a significant challenge in e-commerce: understanding and responding to customer sentiment. The project aimed to use machine learning techniques to automate sentiment classification (positive, negative, neutral) and provide personalized product recommendations based on customer reviews. The system combines three distinct machine learning models, Naive Bayes, SVM, and LSTM, to evaluate the sentiment of reviews. While these models performed well on English data, they faced challenges with Spanish reviews, especially in classifying neutral sentiments. This was linked to issues with class imbalance and insufficient training data.

An important part of the project was creating a recommendation engine that analyzed user sentiment and generated personalized product suggestions. This improved the user experience. The system was built using the Django framework, which allowed users to easily submit reviews, view sentiment results, and receive recommendations. All data was stored and managed through a MySQL database. This setup ensured a smooth user experience and provided an effective way to store large volumes of review data for analysis.

Development followed a clear project management approach from start to finish: laying out the plan, executing it step by step, and making sure each phase stayed on track with what was set out to do. Each phase focused on specific objectives to ensure the project stayed on track with its goals.

The project has shown the potential of AI to improve customer engagement in e-commerce, but there is still room to refine the models to better handle diverse user groups and language-specific nuances. This work contributes to the growing field of AI-driven applications in e-

commerce, providing valuable insights and potential solutions to enhance user experience and decision-making.

References

1. Gomaa, H. (2011). *Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures*. Cambridge University Press.
2. Pressman, R. S., & Maxim, B. R. (2015). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill Education.
3. *scikit-learn: Machine Learning in Python*. Retrieved from https://scikit-learn.org/stable/
4. Kaggle. *Datasets*. Retrieved from https://www.kaggle.com/datasets
5. McAuley, J., & Leskovec, J. (2013). Hidden factors and hidden topics: Understanding rating dimensions with review text. Proceedings of the 7th ACM Conference on Recommender Systems (RecSys 2013). ACM. Computer Science
6. Xiang Zhang, Junbo Zhao & LeCun, Y. (2015). Character-level convolutional networks for text classification. Advances in Neural Information Processing Systems 28 (NIPS 2015)
7. Amazon Reviews for Sentiment Analysis: Xiang Zhang's Text Classification Datasets. Retrieved from drive.google.com/drive/folders/0Bz8a_Dbh9Qhbfll6bVpmNUtUcFdjYmF2SEpmZUZUcVNiMUw1TWN6RDV3a0JHT3kxLVhVR2M.
8. Daza, A., et al. (2024). Sentiment analysis on e-commerce product reviews: A comprehensive view of machine learning and deep learning algorithms. Journal of Big Data.
9. Kontonatsios, G., et al. (2023). FABSA: An aspect-based sentiment analysis dataset of user reviews. Neurocomputing.
10. Hua, Y.-C., et al. (2024). A systematic review of aspect-based sentiment analysis: Trends, dataset diversity, and research gaps. Artificial Intelligence Review.
11. Mozilla Developer Network. (2025). Django Web Framework (Python). MDN Web Docs. Retrieved from https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/Django
12. Bednarz, B., & Miłosz, M. (2025). Benchmarking the performance of Python web frameworks. Journal of Computer Sciences Institute, 36, 336–341.