Case Study: H&M Seasonal Campaign Sentiment Analysis

BUS250x – AI for Business *Professor: Dr Veliota Drakopoulou*

https://github.com/vdrakopoulou

1. Introduction

H&M is a global fashion retailer with a strong digital presence across Instagram, X (Twitter), Facebook, and other platforms. For a new seasonal collection campaign, the marketing team launched a series of:

- Short-form videos
- Influencer posts
- Sponsored ads

Engagement surged: likes, comments, tags, and mentions all increased. However, managers were unsure whether this attention was **good** (excitement about the collection) or **bad** (complaints about sizing, quality, or delivery).

Reading thousands of comments manually is not scalable. The Global E-Commerce & Marketing Analytics team therefore decided to build an **automated sentiment analysis** solution to answer three key questions:

1. What is the **overall sentiment** towards the H&M seasonal campaign?
2. How does sentiment differ **by platform** (Instagram vs X vs Facebook)?
3. What are the **main themes** in positive vs negative feedback?

This case study describes how sentiment analysis was implemented using:

- Multiple approaches (**lexicon**, **machine learning**, **deep learning** conceptual view)
- **Classic NLP** for text preprocessing
- **Python** and common NLP libraries
- **LLMs (e.g., ChatGPT)** as a co-tutor to design and debug code
- **KNIME Analytics Platform** to implement a no-code / low-code workflow
- **Google Colab** to run and share Python notebooks
- Selected **Kaggle datasets** as external references

2. Data Description

The H&M social media team provided a labeled development dataset:

File name: `H&M_sentiment_reviews.csv`

Each row represents a single social media comment. Core columns:

- `comment_text` – the full text of the comment
- `platform` – source (Instagram / X / Facebook)
- `sentiment` – manually assigned:
    - **Positive** – clearly favorable feedback
    - **Neutral** – factual, mixed, or unclear
    - **Negative** – complaints, frustration, dissatisfaction

This labeled set was used to **train** and **evaluate** sentiment models in Python and KNIME.

---

3. Approaches to Sentiment Analysis (Conceptual Overview)

We want the computer to decide if a comment is **Positive**, **Neutral**, or **Negative**.
There are many ways to do this. In this project we look at:

1. **Lexicon-based methods** – use a ready-made "sentiment dictionary"
2. **Machine learning methods** – train a model on labeled H&M comments
3. **Deep learning methods** – use big neural models like BERT (concept only)
4. **Classic NLP** – cleaning text and turning it into numbers (supports 1–3)
5. **LLMs (e.g. ChatGPT)** – used as a helper/tutor and sometimes a classifier
6. **KNIME** – a visual, drag-and-drop tool to build the workflow without coding

#	Approach	Simple idea	What's good?	What's not so good?	How we use it here
1	**Lexicon-based**	Use a **sentiment dictionary** (e.g. VADER). Each word has a score (positive/negative). Add scores for a comment.	Fast, simple, no training data needed. Works OK on short social posts.	Doesn't handle sarcasm, slang, or brand-specific words very well.	Used as a **simple baseline** to compare with better models.
2	**Machine learning**	Clean the text → turn it into numbers (TF–IDF) → train a model (e.g. Logistic Regression) using H&M's labeled comments.	Learns from **H&M's own data**, usually more accurate than lexicon methods. We can see which words matter.	Needs labeled data. Quality depends on how good and balanced the data is.	This is our **main method**: TF–IDF + Logistic Regression in Python and KNIME.
3	**Deep learning (Transformers)**	Use big neural models like **BERT** that understand context. Can be used "as is" or fine-tuned on H&M data.	Often best performance on complex, tricky language; handles context like "not impressed".	More complex, needs more computing power, harder to explain to managers.	Only **explained conceptually** as a possible future improvement.
4	**Classic NLP + features**	"Cleaning steps": split into words, lowercase, remove punctuation & stop words, maybe stem, then convert words to numbers (Bag-of-Words, TF–IDF).	Makes the text clean and consistent so models can use it.	Needs some tuning (which steps help/hurt); by itself doesn't "understand" meaning.	Used to preprocess `comment_text` and build **TF–IDF features** for our ML models.
5	**LLMs (ChatGPT) as co-tutor**	Ask ChatGPT to design steps, explain ideas, write or debug code; can also classify comments via prompts.	You don't need to code everything from scratch; quick help with errors and ideas.	Needs clear prompts; it's a "black box" and you still need to check results.	Used mainly as a **tutor and coding assistant** (not the main model).
6	**KNIME visual workflow**	Drag-and-drop tool: connect blocks like CSV Reader → Text Cleaning → TF–IDF → Classifier → Scorer.	Very visual, little/no coding, easy to show to non-technical people.	Less flexible than writing code; you must learn how nodes work.	Used to **rebuild the Python pipeline visually** and confirm our results.

3.4 Classic NLP (Text Preprocessing & Feature Engineering)

Classic NLP is the "plumbing" – the basic cleaning work we do on text before giving it to a model.

We apply these steps to each `comment_text`:

1. **Tokenization** – split the sentence into words.
   - Example: "Love the new H&M jeans" → ["Love", "the", "new", "H&M", "jeans"]
2. **Lowercasing** – make everything lowercase.
   - "AMAZING" → "amazing"
3. **Punctuation removal** – remove things like "!!!", "?", commas.
4. **Stop word removal** – remove very common words like "the", "is", "are" that usually don't carry sentiment.
5. **Stemming / lemmatization** – reduce different forms of a word to one base form.
   - "delayed", "delaying", "delays" → "delay"
6. **Vectorization** – turn words into numbers so the computer can use them:
   - **Bag-of-Words** – count how many times each word appears.
   - **TF–IDF** – similar to Bag-of-Words but gives more weight to important words and less weight to common words.

In this project, our main feature representation is **TF–IDF**, and we feed those TF–IDF numbers into **Logistic Regression** to predict sentiment.

3.5 LLMs (ChatGPT) as a Co-Tutor

Instead of building everything alone, we can "borrow" the intelligence of large language models like ChatGPT.

We use ChatGPT in two main ways:

1. **As a teacher / helper (co-tutor):**
   - Ask it to **explain concepts** in simple language (e.g., "What is TF–IDF?").
   - Ask it to **design a pipeline** (e.g., "How do I build TF–IDF + Logistic Regression in Python?").
   - Ask it to **write or debug code** ("Why do I get this error?").
2. **As a classifier (conceptual idea):**
   - Give ChatGPT a comment and ask:
     "Is this Positive, Neutral, or Negative? Explain briefly."
   - This can be useful if you don't want to build your own model, but we mainly use it as a **helper** in this case study.

In this project, LLMs are mainly used as a **tutor and coding assistant**:
they help generate and explain Python code and suggest improvements to the pipeline.

3.6 KNIME: A Visual Workflow Tool

KNIME Analytics Platform is a tool where you build data workflows by **dragging and connecting blocks** (nodes), instead of writing code.

For this sentiment analysis, KNIME lets us:

- Build a visual sentiment pipeline like:
  CSV Reader → Text Preprocessing → TF–IDF → Classifier → Scorer
- See each step clearly, which is helpful for people who don't like coding.
- Reproduce the **same logic** as the Python model, but in a **no-code / low-code** way.

So, Python is our **coding version**, and KNIME is our **visual version** of the same idea.

4. Explaining the Jargon in Simple Language

This part is for people in H&M's marketing team who are not technical.

- **NLP (Natural Language Processing)**
  Getting computers to read, process, and understand human language (like comments and reviews).
- **Tokenization**
  Cutting a sentence into separate words or pieces.
- **Stop words**
  Very common words like "the", "is", "and" that usually don't help with sentiment, so we often remove them.
- **Stemming / Lemmatization**
  Reducing words to a base form.
  Example: "delayed", "delaying", "delays" → "delay".
- **Vectorization / TF–IDF**
  Turning words into numbers. TF–IDF gives higher numbers to important words and lower numbers to very common ones.
- **Machine learning model**
  A computer program that learns from examples (comments with known sentiment) and then predicts the sentiment of new comments.
- **Confusion matrix**
  A table showing how many comments the model got right or wrong for each class (Positive, Neutral, Negative).
- **Accuracy**
  The percentage of comments that the model classified correctly.

4.1 What Are We Trying to Do?

We want the computer to read each comment and decide:

- **Positive** – the customer is happy
- **Neutral** – the comment is factual or mixed
- **Negative** – the customer is unhappy

Examples:

- "Love the new collection 😍" → **Positive**
- "Delivery is slow and sizes are wrong" → **Negative**
- "Order arrived yesterday." → **Neutral**

Instead of a human reading thousands of comments, we want the computer to **do this automatically** and then give us statistics.

4.2 Two Big Ideas: NLP + Machine Learning

To do this, we combine two big ideas:

- **NLP** – cleaning and preparing text so the computer can handle it (split into words, remove noise, turn into numbers).
- **Machine Learning** – showing the computer many examples with labels (Positive/Neutral/Negative) so it can learn patterns and then predict labels for new comments.

You can think of it like this:

1. **NLP = cleaning and organizing sentences**.
2. **Machine learning = teaching a model from labeled examples**, like teaching a child what "good" and "bad" mean by showing examples.

4.3 NLP Steps on a Sample Comment

Take one comment:

"OMG!!! These new H&M jeans are AMAZING 😍 😍 But delivery was 2 days late…"

a) Tokenization – cut into pieces

```
["OMG", "These", "new", "H&M", "jeans", "are", "AMAZING", "But", "delivery",
"was", "2", "days", "late"]
```

b) Lowercasing

```
["omg", "these", "new", "h&m", "jeans", "are", "amazing", "but", "delivery",
"was", "2", "days", "late"]
```

c) Remove punctuation / numbers (optional)

```
["omg", "these", "new", "h&m", "jeans", "are", "amazing", "but", "delivery",
"was", "days", "late"]
```

d) Stop word removal

Remove common words like "these", "are", "but", "was":

```
["omg", "new", "h&m", "jeans", "amazing", "delivery", "days", "late"]
```

e) Stemming / lemmatization

Group similar forms: "delayed", "delaying" → "delay".

f) Vectorization – words → numbers

We then convert the cleaned words into numeric features (TF–IDF), so the computer can process them.

4.4 Machine Learning in Simple Words

After NLP, each comment becomes:

- A **row of numbers** (features)
- A **label**: Positive / Neutral / Negative

We show many examples to the model:

"This is Positive."
"This is Negative."
"This is Neutral."

The model learns patterns (e.g., "love, amazing, great" → Positive; "late, terrible, refund" → Negative).
Then, for new comments, it can **predict** the sentiment.

---

5. Python NLP Libraries for Sentiment Analysis

Python offers several libraries (toolboxes). Key ones used or discussed in this case:

Library / Tool	Role	Typical Use in H&M Project
TextBlob	Simple NLP + sentiment	Quick polarity check for a few comments
VADER (in NLTK)	Social media sentiment lexicon	Lexicon baseline for Instagram/X comments
NLTK	Classic NLP toolkit	Tokenization, stopwords, helping preprocessing
scikit-learn	Machine learning	TF–IDF + Logistic Regression / SVM sentiment classifier
spaCy	Industrial NLP	Advanced tokenization/lemmatization (optional)
Transformers (HF)	Deep learning / BERT	Conceptual mention, possible future deep model
OpenAI SDK / LLM API	LLM access	Use ChatGPT as tutor and on-demand classifier (conceptual)

In the implementation, the main stack is:

- **pandas** for data handling
- **scikit-learn** for TF–IDF and classification

---

6. Real Data Sources from Kaggle (Benchmark & Context)

To situate our H&M analysis in a broader context, we looked at real sentiment datasets from **Kaggle**:

1. **Sentiment140** – ~1.6M tweets labeled as positive/negative.
    - Similar short, informal text.
    - Shows how sentiment analysis scales to very large volumes.
2. **Brand Sentiment Analysis Dataset (Twitter)** – tweets about brands with sentiment labels.
    - Conceptually similar to monitoring H&M's brand reputation.
    - Confirms that using social media sentiment for brand insights is a standard practice.

These datasets serve as examples and benchmarks, though our main modeling is done on the **H&M-specific** dataset to remain brand-relevant.

---

7. Implementation in Google Colab (Step-by-Step)

Google Colab is used as the main environment for building and testing the Python sentiment model.

7.1 Setup

1. Open **Google Colab** and create a **New Notebook**.
2. Upload `H&M_sentiment_reviews.csv` from your local machine to Colab's file system.

7.2 Install and import libraries

In the first cell:

```
!pip install pandas scikit-learn
```

Then:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
```

7.3 Load the H&M dataset

```
df = pd.read_csv("H&M_sentiment_reviews.csv")
print(df.head())
print(df.columns)
```

Verify that `comment_text` and `sentiment` columns exist.

7.4 Split into train and test sets

```
X = df["comment_text"].astype(str)
y = df["sentiment"].astype(str)

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)
```

7.5 Build TF–IDF + Logistic Regression pipeline

```
model = Pipeline(steps=[
    ("tfidf", TfidfVectorizer(
        max_features=10000,
        ngram_range=(1, 2),
        stop_words="english"
    )),
    ("clf", LogisticRegression(max_iter=1000))
])
```

7.6 Train and evaluate

```
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification report:\n", classification_report(y_test, y_pred))
print("\nConfusion matrix:\n", confusion_matrix(y_test, y_pred))
```

This produces overall accuracy and a breakdown by class (Positive/Neutral/Negative).

7.7 Predict sentiment for new comments

```
new_comments = [
    "Love the new Spring collection, super stylish!",
    "Delivery was late and the quality is not good.",
    "Order arrived yesterday."
]
print(model.predict(new_comments))
```

These predictions can be used to test the model qualitatively.

---

8. Using ChatGPT to Generate and Improve Python Code

LLMs like ChatGPT were used as a **co-tutor** during development.

8.1 Initial pipeline prompt

A typical prompt:

"I have a CSV file `H&M_sentiment_reviews.csv` with columns `comment_text`, `sentiment`, and `platform`.
I want to build a TF–IDF + Logistic Regression sentiment classifier in Python (for Google Colab).
Please give step-by-step code and explain:
– loading data
– splitting train/test
– building the pipeline
– training
– evaluation (accuracy, classification report, confusion matrix)
– predicting sentiment for new H&M comments."

ChatGPT generated a pipeline similar to the one in Section 7, which was then **reviewed**, **modified**, and **commented** by the student.

8.2 Debugging with ChatGPT

When errors occurred (e.g., missing columns, mismatched shapes), relevant code snippets and error messages were pasted into ChatGPT with a prompt such as:

"Here is my code and the error. What is wrong and how can I fix it?"

This accelerated learning and helped fix syntax and logic issues.

8.3 Improving the model

After obtaining a first result, ChatGPT was asked:

"My model accuracy is about X%. Suggest three ways to improve performance using scikit-learn and show example code (e.g. try LinearSVC, tune TF–IDF parameters, etc.)."

These suggestions guided experiments with:

- Different `ngram_range`
- Larger `max_features`
- Alternative classifiers like `LinearSVC`

The final choices were made based on validation performance and interpretability for the business audience.

---

9. Step-by-Step Sentiment Analysis in KNIME

The same logic was implemented as a visual workflow in **KNIME Analytics Platform**.

9.1 Data input

1. **File Reader / CSV Reader** node:
   - Read `H&M_sentiment_reviews.csv`
   - Ensure `comment_text`, `sentiment`, `platform` are String.
2. Optional **Data Explorer** node to inspect distributions.

9.2 NLP preprocessing

Nodes in sequence:

1. **Column Filter**:
   - Keep `comment_text`, `sentiment`, `platform`.
2. **Strings to Document**:
   - Convert `comment_text` into a `Document` column.
3. **Case Converter**:
   - Convert to lower case.
4. **Punctuation Erasure**:
   - Remove punctuation.
5. **Number Filter** (optional):
   - Remove numbers.
6. **Stop Word Filter**:
   - Remove common English stop words.
7. **(Optional) Snowball Stemmer / Lemmatizer**:
   - Reduce words to base forms.

9.3 Feature extraction (BoW + TF–IDF)

1. **BoW Creator**:
   - Create bag-of-words representation.
2. **TF–IDF**:
   - Compute TF–IDF weights.

3. **Document Vector**:
    - Convert TF–IDF into a numeric table (rows = comments, columns = features).

Ensure that the `sentiment` label stays attached to each row; if needed, join back to the original data by RowID.

9.4 Train/test split

- **Partitioning** node:
    - Training = 80%, Test = 20%
    - Use **stratified sampling** by `sentiment` if available.

9.5 Model training

Option A: Logistic Regression

1. **Logistic Regression Learner**:
    - Input: training partition.
    - Target: `sentiment`.
    - Features: all TF–IDF columns.
2. **Logistic Regression Predictor**:
    - Apply model to test partition.

Option B: Naive Bayes
Replace the learner and predictor nodes with the Naive Bayes equivalents.

9.6 Evaluation with Scorer

- **Scorer** node:
    - Actual column: `sentiment`.
    - Predicted column: `Prediction (sentiment)`.

Outputs:

- Confusion matrix.
- Overall accuracy.
- Per-class metrics (depending on options).

These results can be compared with Python outcomes for consistency.

9.7 Per-platform analysis in KNIME

1. After prediction, use a **Rule Engine** or **Math Formula** to create:
    - `correct = 1` if `sentiment` = `Prediction (sentiment)`
    - `correct = 0` otherwise
2. Use a **GroupBy** node:

- Group by `platform`.
- Aggregate mean of `correct` → accuracy per platform.

This allows H&M to see whether sentiment is easier/harder to predict on different social networks and which platforms have more negative comments.

---

10. Results, Insights, and Recommendations (Template)

(You should fill this section with your actual numbers and charts.)

10.1 Model performance (example structure)

- Lexicon (VADER) baseline accuracy: **X%**
- TF–IDF + Logistic Regression (Python): **Y%**
- TF–IDF + Logistic Regression (KNIME): **Y%** (similar)

Confusion matrices show that:

- Positive and Negative are generally well detected.
- Neutral comments are the hardest to classify (often confused with Positive).

10.2 Business insights

From model predictions on the dataset:

1. **Overall sentiment distribution**:
    - ~A% Positive, B% Neutral, C% Negative.
2. **Platform comparison**:
    - Instagram: more Positive comments about style and visuals.
    - X (Twitter): higher proportion of Negative comments, especially about delivery and customer service.
    - Facebook: mixed, more detailed feedback about sizes and returns.
3. **Themes in positive feedback**:
    - Keywords: "love", "style", "fit", "affordable", "colours".
    - Customers praise design and price point.
4. **Themes in negative feedback**:
    - Keywords: "late", "delivery", "wrong size", "cheap quality", "refund".
    - Pain points: logistics, sizing consistency, perceived quality.

10.3 Recommendations to H&M

1. **Act on delivery & logistics**:
    - Investigate carriers and warehouses causing delays for the campaign period.

    - Communicate expected delivery times clearly in ads and checkout.
2. **Improve sizing information**:
    - Add clearer size guides and user reviews mentioning fit.
    - Highlight “true to size” items and flag those that run small/large.
3. **Leverage positive style feedback**:
    - Use positive comments in marketing (user testimonials).
    - Double down on popular pieces in inventory and future designs.
4. **Ongoing monitoring**:
    - Integrate the sentiment analysis pipeline (Python or KNIME) into a **weekly dashboard**.
    - Track sentiment over time by platform and by new campaigns.

---

11. Limitations and Future Work

- **Data representativeness:**
  Data is limited to a sample from certain platforms and time periods. It may not capture all markets or languages.
- **Language and sarcasm:**
  Current models may miss sarcasm (“Yeah, great job H&M…”) or mixed emotions.
- **Multilingual comments:**
  Non-English comments are not yet fully supported and may need language detection and translation.
- **Deep learning models:**
  Transformer-based models (e.g., BERT) could further improve accuracy, especially on ambiguous comments, but are not yet implemented in production.

Future improvements:

- Fine-tune a transformer model on H&M data.
- Incorporate emoji and hashtag features more explicitly.
- Extend analysis to topics (topic modeling) in addition to sentiment.

---

12. Conclusion

This case study shows how H&M can move from manual reading of social media comments to an **automated sentiment analysis pipeline** that:

- Provides **quantitative insight** into how customers feel about seasonal campaigns.
- Highlights differences between **Instagram**, **X**, and **Facebook**.
- Identifies **themes** in positive and negative feedback that can guide marketing and operations.

By combining:

- **Classic NLP** and **machine learning** in Python (Google Colab),
- **Visual workflows** in KNIME,
- And **LLMs** like ChatGPT as a co-tutor,

BUS250X – AI for Business

H&M Seasonal Campaign – Sentiment Analysis

Student Handout / Answer Sheet

Student Name: ____________________________
Student ID: ______________________________
Group (if any): __________________________
Date: ____________________________________

---

1. Business Understanding (Short Summary)

1.1 What is the business problem?
(In 2–4 sentences, explain why H&M cares about sentiment.)

1.2 Key business questions

Tick or briefly describe which questions you focused on:

- Overall sentiment towards the campaign
- Differences in sentiment by platform (Instagram / X / Facebook)
- Main themes in positive comments
- Main themes in negative comments
- Other: ____

---

2. Data Description

2.1 Dataset used

- File name: ____
- Number of comments (rows): _______________________________
- Time period (if known): __________________________________

2.2 Columns (tick what you had)

- `comment_text`
- `sentiment` (Positive / Neutral / Negative)
- `platform` (Instagram / X / Facebook)
- Other: ____

2.3 Sentiment distribution

Fill in from your data:

Sentiment	Count	Percentage
Positive	______	______ %
Neutral	______	______ %
Negative	______	______ %
Total	______	100 %

---

3. Approaches to Sentiment Analysis (Conceptual)

In 1–2 sentences each, explain in your own words:

3.1 Lexicon-based sentiment (e.g. VADER)
How does it work? Pros/cons for H&M comments?

3.2 Machine learning-based sentiment (TF–IDF + classifier)
What is the idea? Why is it useful here?

3.3 Deep learning (BERT etc.) – conceptual only
What's different about this compared to classic ML?

---

4. Explaining the Jargon (Plain English)

Choose **3–4 terms** and explain them simply.

Examples: NLP, tokenization, stop words, TF–IDF, confusion matrix, accuracy.

- **Term 1:** ________________________
  Explanation: ____

- **Term 2:** ________________________
  Explanation: ____

- **Term 3:** ________________________
  Explanation: ____

- **Term 4 (optional):** ______________
  Explanation: ____

---

5. Python / Google Colab – Model & Results

5.1 Libraries used (tick all that apply)

- pandas
- scikit-learn
- NLTK / VADER
- TextBlob
- Other: ___________________________

5.2 Model pipeline (brief)

Describe your main Python model in one short paragraph (e.g., TF–IDF + Logistic Regression).

5.3 Python model performance

Copy your main test results:

- Overall accuracy (Python): __________

Fill in from your `classification_report`:

Class	Precision	Recall	F1-score
Positive	________	______	________
Neutral	________	______	________
Negative	________	______	________

5.4 Any quick observations about these results?

6. Kaggle / External Datasets (Short Reference)

List **1–2 Kaggle datasets** related to sentiment that you looked up (even briefly).

1. Dataset name: ____
   Why it's relevant: ____

2. Dataset name: ____
   Why it's relevant: ____

7. Using ChatGPT / LLMs as a Co-Tutor

7.1 How did you use ChatGPT in this project?
(Tick all that apply)

- To generate initial Python code
- To debug error messages
- To explain evaluation metrics (accuracy, precision, recall)
- To suggest model improvements
- To explain NLP concepts

7.2 One concrete example

Describe one helpful interaction (e.g., a bug it helped you fix or code it helped you improve):

8. KNIME Workflow – Steps & Results

8.1 Main KNIME steps (summarize)
(Write bullet points: CSV Reader → Text preprocessing → TF–IDF → Learner → Predictor → Scorer)

8.2 KNIME model performance

- Overall accuracy (KNIME): __________

8.3 Compare Python vs KNIME

Which was higher, and were they similar?

9. Insights for H&M

Use your model and data to answer the business questions.

9.1 Overall sentiment

"Overall, about ______ % of comments are Positive, ______ % Neutral, and ______ % Negative. This suggests that the campaign sentiment is mostly ___________________________."

9.2 By platform (if you computed it)

Fill in any numbers or just relative trends:

- Instagram: ____
- X (Twitter): ____
- Facebook: ____

9.3 Main themes in Positive comments

List a few recurring ideas or keywords:

9.4 Main themes in Negative comments

10. Recommendations & Reflection

10.1 Recommendations for H&M

Give 2–3 short recommendations based on your analysis. Examples: improve delivery; clarify sizing; reuse positive themes in marketing.

1. ______________________________
2. ______________________________
3. ______________________________

10.2 Reflection: What did you learn?

In 2–4 sentences, reflect on what you learned about:

- Sentiment analysis
- Using Python / KNIME
- Working with ChatGPT

____
____
____
____

(End of Handout)