

Sentiment Analysis Workshop

Customer Feedback Classification

Using GenAI

About Me

- **Current Role:** Learning Experience Manager at [AlignAI](#)



- **Experience:** 15+ years in analytics
- **Passions:** Artificial Intelligence (AI), Governance, and Instructional Design
- **Memberships & Volunteer Work:** WIA and MORPC
- **Education:** Doctoral student at Franklin University



Kristy Wedel

Learning Experience Manager |
Transforming Data Governance & An...



Workshop Goals

- Explore AI for sentiment analysis
- Hands-on approach using **Cursor**
- Build, train, and evaluate a sentiment analysis model
- Gain practical skills to adapt and refine models

Requirements

Follow Sentiment Analysis Workshop Quick Start.docx Instructions

Install Conda

[Miniconda \(lightweight\)](#)

OR

[Anaconda \(comprehensive\)](#)

Install Cursor

[Download Cursor](#)

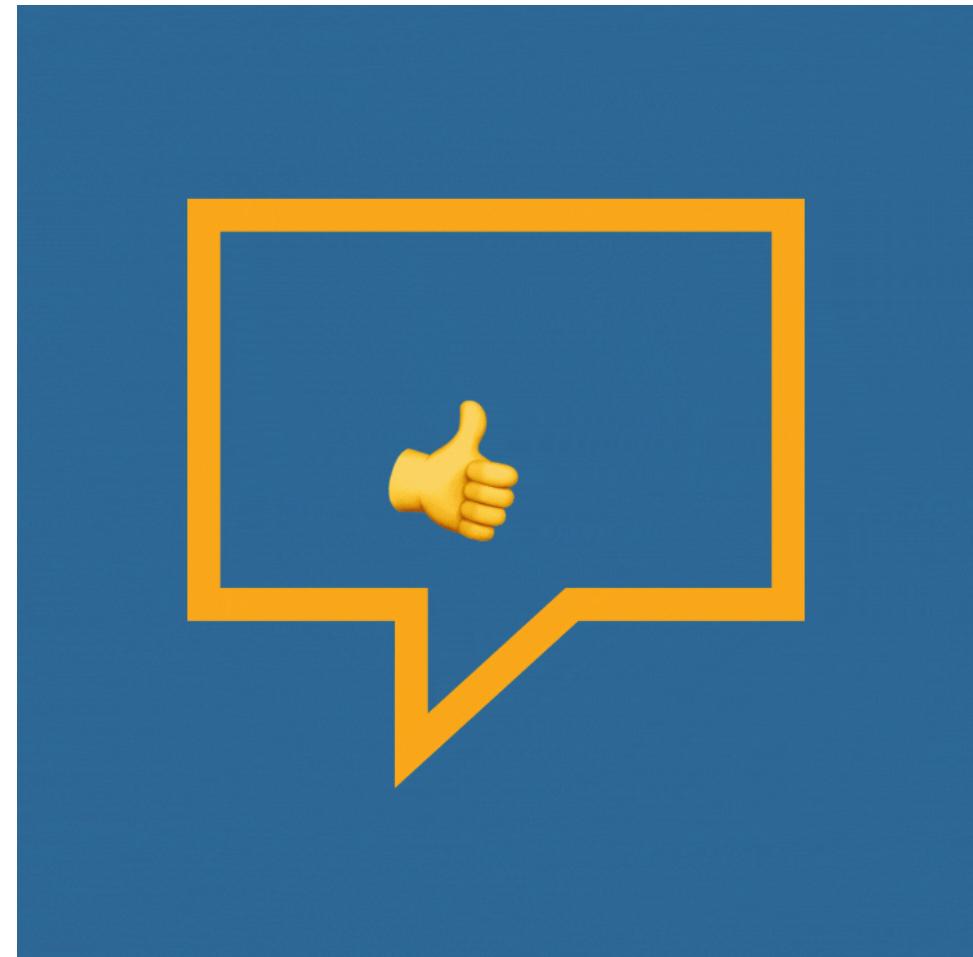
Files



<https://bit.ly/3PIK4RU>

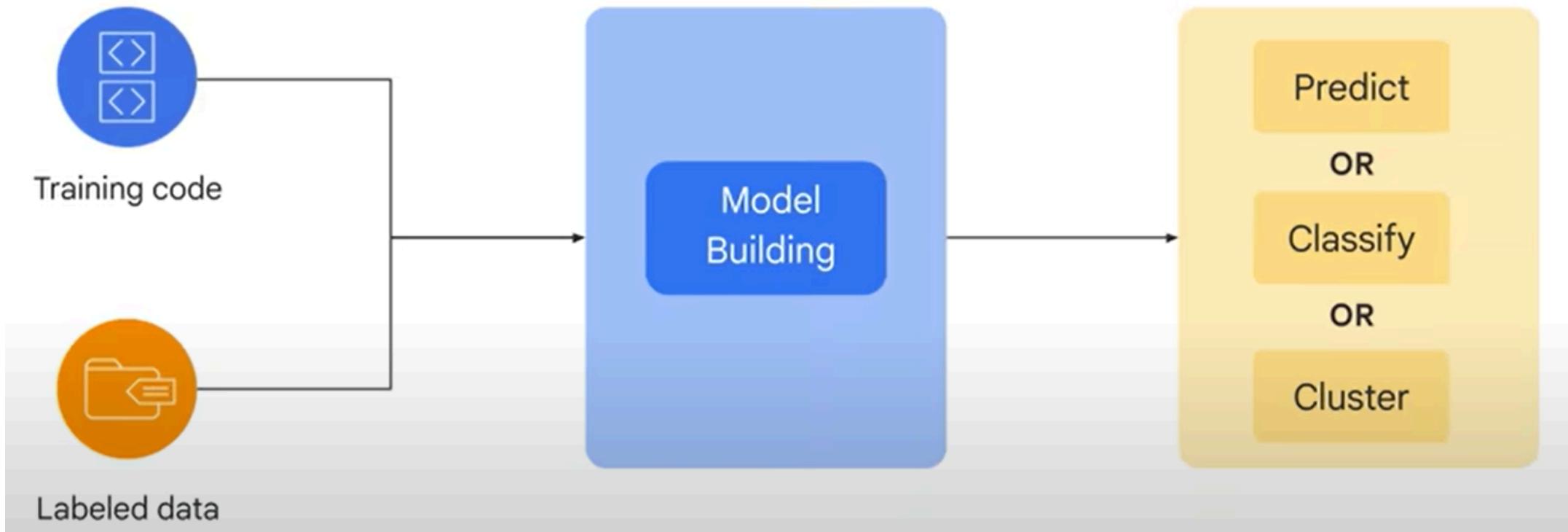
Introduction to Sentiment Analysis

- NLP technique to classify text into:
 - Positive, Negative, Neutral
- Applications:
 - Customer feedback analysis
 - Social media monitoring
 - Prioritizing support tickets

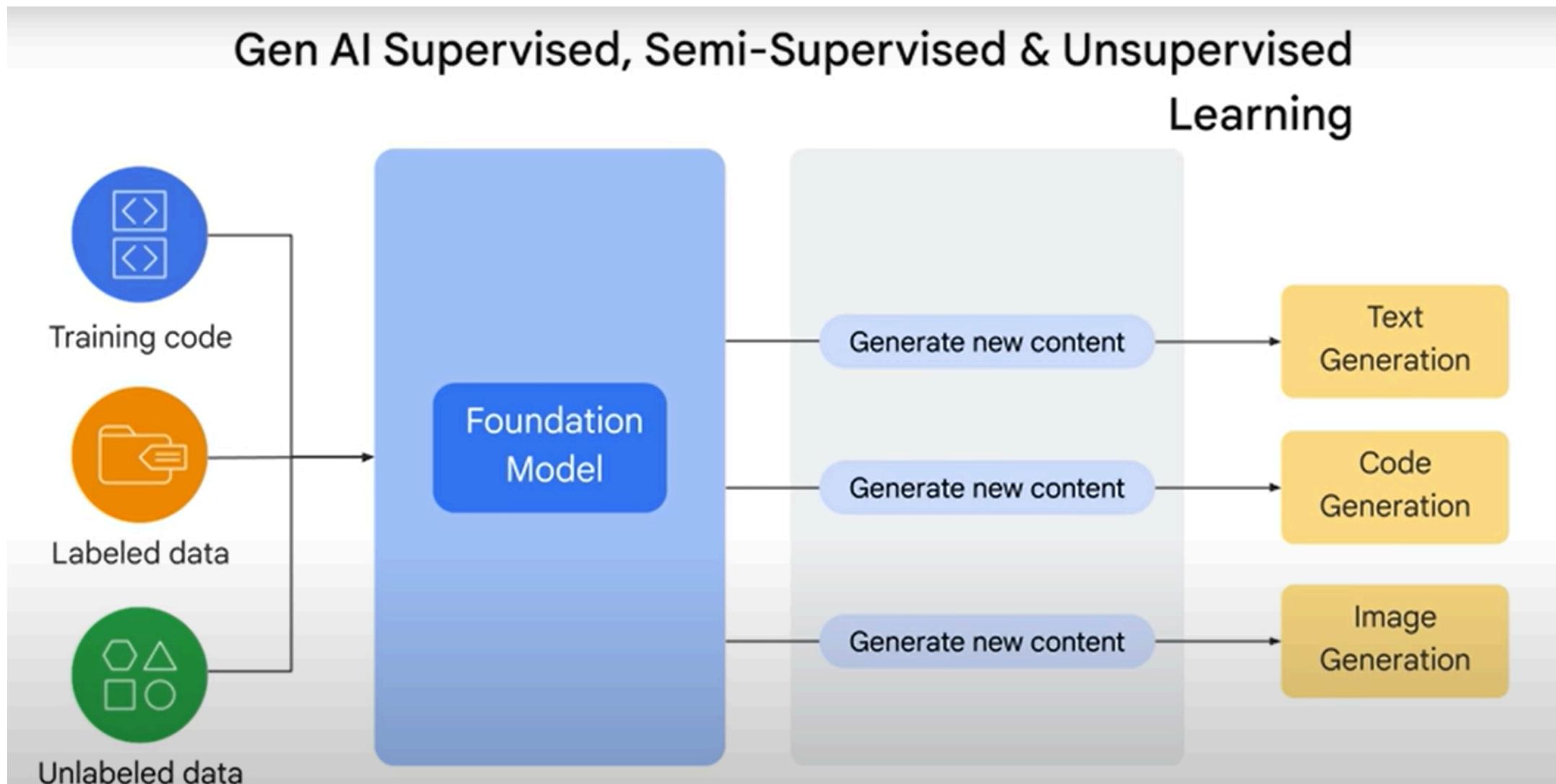


Traditional ML

Classical Supervised & Unsupervised Learning



Gen AI

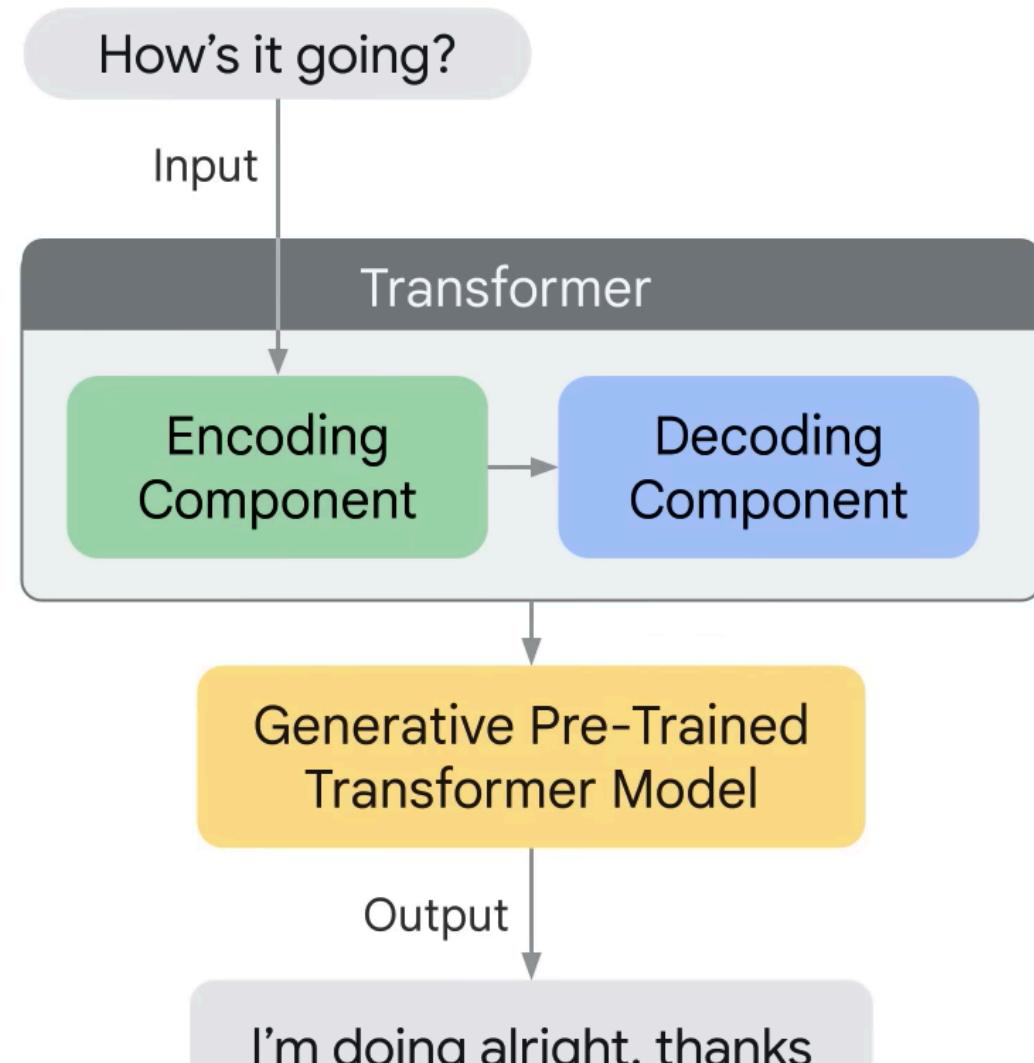


How Gen AI Works

How it Works

Pre-Training:

- Large amount of Data
- Billions of parameters
- Unsupervised learning

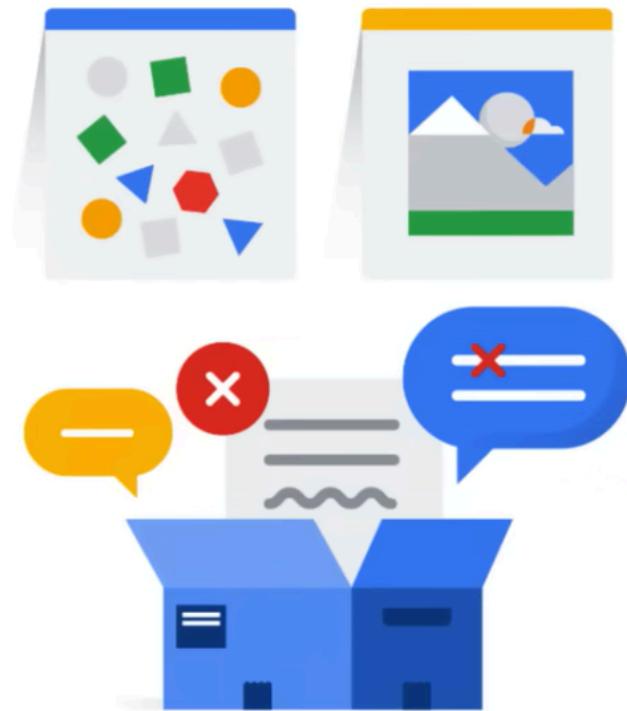


Hallucinations

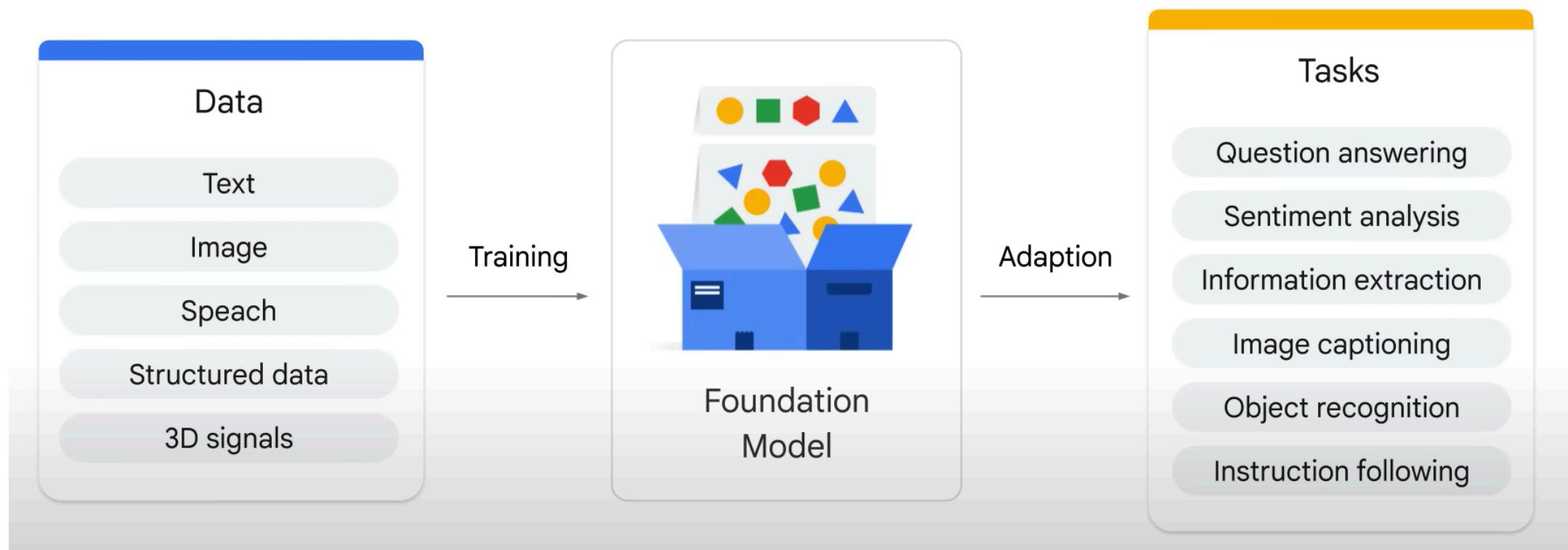
Hallucinations

Challenges

- ❗ The model is not trained on enough data
- ❗ The model is trained on noisy or dirty data
- ❗ The model is not given enough context
- ❗ The model is not given enough constraints



Customization

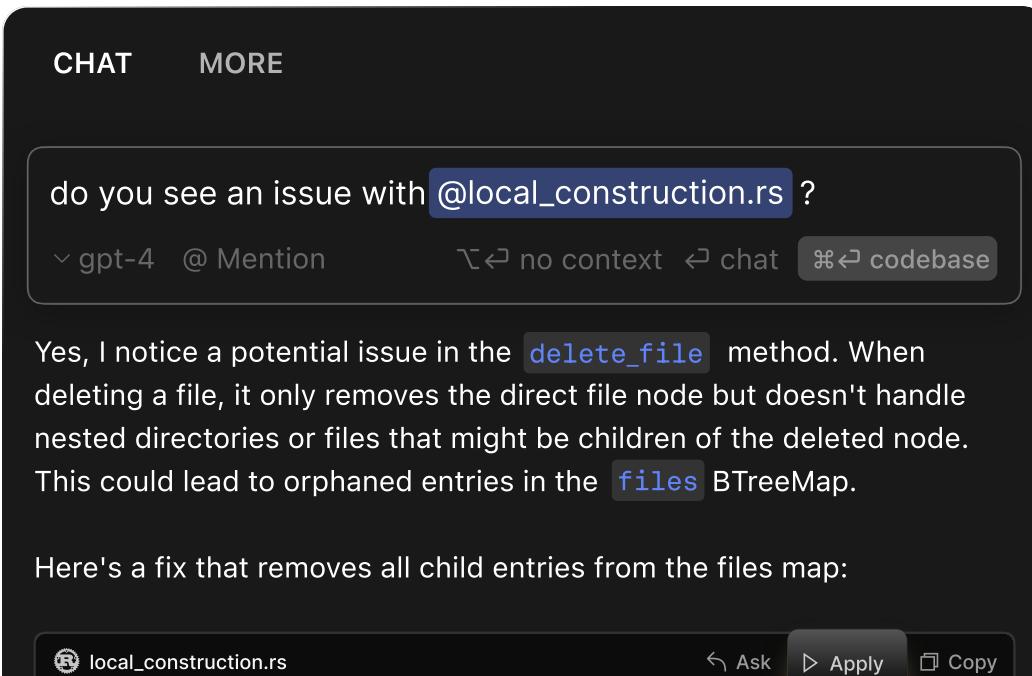


What is Cursor?

- AI-powered code editor designed to boost developer productivity.
- Combines a clean interface with advanced AI capabilities.

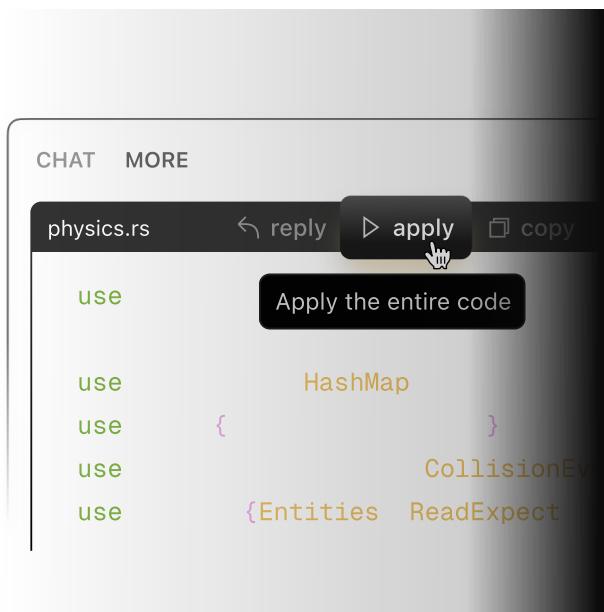
Cursor: Chat

- Chat - Chat lets you talk with an AI that sees your codebase. The chat can always see your current file and cursor, so you can ask it things like: "Is there a bug here?". You can add particular blocks of code to the context with Ctrl+Shift+L or "@." You can chat with your entire codebase with Ctrl+Enter.



Cursor: Instant Apply

- Instant Apply - Apply the code suggestions from chat back into your codebase by clicking the play button on top of any chat codeblock.



Cursor Benefits

- Saves time by reducing repetitive tasks.
- Enhances code quality with AI recommendations.
- Simplifies onboarding for new developers.
- Improves collaboration with integrated tools.

Workflow Steps

Step 1: Data Cleaning and Preprocessing

Step 2: Model Training and Evaluation

Step 3: Deployment and Prediction

Step 4: Real-Time Sentiment Predictions

Data Overview

- **Dataset Details:**
 - 1,000 customer feedback entries.
 - Columns: Review_ID , Review_Text , Sentiment .
 - Labels: Positive, Neutral, Negative.

Google Colab

<https://tinyurl.com/3jccy5yc>

Workflow Steps

Step 1: Data Cleaning and Preprocessing

Step 2: Model Training and Evaluation

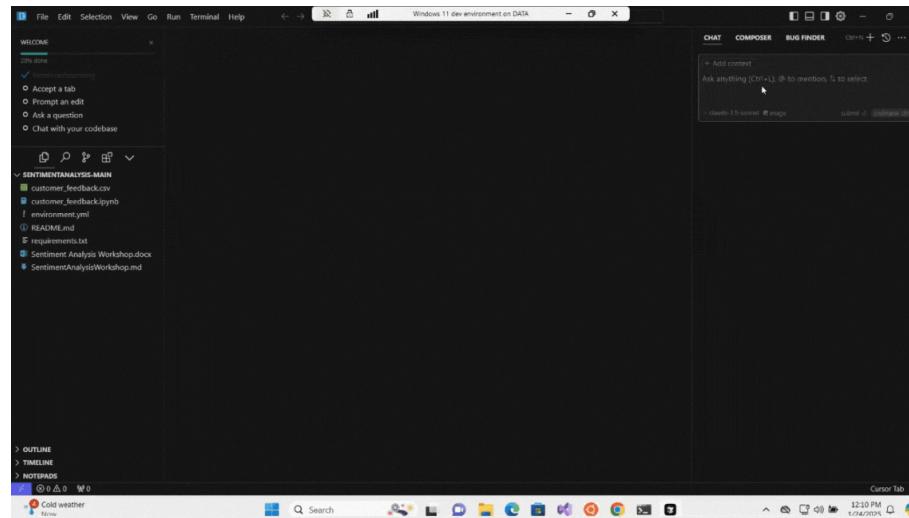
Step 3: Deployment and Prediction

Load Data

Prompt Example:

Write Python code to:

- Create a python jupyter notebook to load customer_feedback.csv



Data Preprocessing

- Data preprocessing is a critical step in data analysis and machine learning.
- It ensures the raw data is clean, structured, and usable for analysis.

Key Steps

- **Data Cleaning:** Handle missing values, remove duplicates, and correct inconsistencies.
- **Data Transformation:** Normalize, encode variables, and create new features.
- **Data Integration:** Merge data from multiple sources, resolve schema mismatches.
- **Data Reduction:** Reduce dataset size while preserving important information.

Cleaning for Sentiment Analysis

- **Text Cleaning:**
 - Remove special characters and stop words and convert to lowercase.
- **Tokenization:**
 - Split `Review_Text` into words for analysis.
- **Lemmatization:**
 - Reduce words to their base forms (e.g., "running" → "run").
- **Handling Missing Data:**
 - Drop or impute missing entries.
- **Deduplication:**
 - Remove duplicate reviews to avoid bias.

Data Cleaning

Prompt Example:

Write Python code to replace the existing data:

- Remove duplicate rows.
- Drop rows where "Review_Text" or "Sentiment" columns have missing values.

Then, save the dataset as a new csv and visualize the distribution by sentiment with a simple chart.

The screenshot shows a Jupyter Notebook environment on a Windows desktop. The notebook has a single cell containing Python code. The code imports pandas, numpy, and matplotlib, reads a CSV file, displays its first 5 rows, and then performs several operations: it checks for missing values in the dataset, prints the count of missing values, and then drops rows with missing values from the DataFrame. A tooltip on the right provides instructions for using the notebook, such as loading the CSV file and running each cell sequentially.

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Read the CSV file
df = pd.read_csv('customer_feedback.csv')

# Display the first 5 rows of the dataset
print("First 5 rows of the dataset:")
display(df.head())

# Get basic information about the dataset
print("Basic information about the dataset")
print(df.info())

# Display basic statistics of numerical columns
print("Basic statistics of numerical columns")
display(df.describe())

# Check for missing values
print("Missing values:")
print(df.isnull().sum())

... Ahs ...
```

Review_ID	Review_Text	Sentiment	
0	522	Excellent service and fast delivery. Highly recomended.	Positive
1	728	Terrible experience. The product broke after a few days.	Negative
2	741	Neutral feelings about this product. It's functional.	Neutral
3	661	This is the best purchase I've made this year!	Positive
4	412	The product is okay, but nothing special.	Neutral

Workflow Steps

Step 1: Data Cleaning and Preprocessing

Step 2: Model Training and Evaluation

Step 3: Deployment and Prediction

Generative AI Key Terms

- **Tokens:** The smallest units of text, such as words or subwords, used in AI processing.
- **Encoding:** The process of converting text into numerical representations for machine learning models.
- **Transformer:** A neural network architecture that uses self-attention to process sequences of data.
- **Loss:** A metric that measures the difference between predicted and actual outputs, guiding model training.
- **Epochs:** Complete passes through the entire training dataset during model training.

Overview of Pretrained Models

distilbert-base-uncased-finetuned-sst-2-english

- **Base Model:** DistilBERT, a lighter, faster version of BERT.
- **Fine-Tuned Task:** Sentiment analysis using the SST-2 dataset.
- **Performance:** Delivers accurate binary sentiment predictions (positive/negative).
- **Use Cases:** Ideal for analyzing customer feedback, social media, and reviews.

Why Use Pretrained Models?

- Minimize training time and computational resources.
- Achieve state-of-the-art results with minimal fine-tuning.
- Versatile across various NLP tasks.

Key Preprocessing Steps for DistilBERT

Step	Example	Explanation
Convert to Lowercase	"Great Day" → "great day"	Ensures uniformity by treating uppercase and lowercase words as the same.
Remove Punctuation	"Hello, world!" → "Hello world"	Eliminates unnecessary symbols to focus on meaningful content.
Remove Stop Words	"This is an example" → "example"	Removes common words that do not contribute to understanding the meaning.
Apply Lemmatization	"running", "runs" → "run"	Reduces words to their base forms, grouping similar words together.
Tokenization	"I love AI" → ["I", "love", "AI"]	Splits text into smaller units (tokens) for easier processing.
Padding/Truncation	["I", "love"] → ["I", "love", "[PAD]"]	Ensures all inputs are of uniform length for batch processing.
Add Special Tokens	["I", "love"] → "[CLS]", "I", "love", "[SEP]"	Special tokens [CLS] and [SEP] mark the start and end of the input.
Create Attention Masks	["I", "love", "[PAD]"] → [1, 1, 0]	Binary masks differentiate real input tokens from padding.
Numerical Encoding	["I", "love"] → [101, 2027]	Converts tokens into numerical IDs for processing by the model.

Other Pretrained Models

1. BERT:

- Bidirectional context understanding for tasks like Q&A and classification.
- Popular variants: BERT-Base, BERT-Large.

2. GPT:

- Autoregressive model for text generation and conversational AI.

3. DistilBERT:

- Lightweight, fast, and efficient for edge devices.

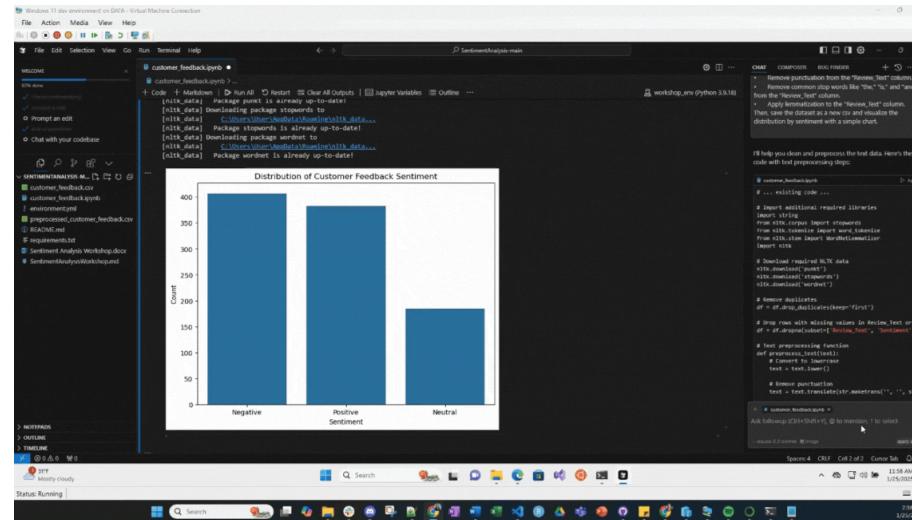
Key Terms in LLM Model Training

- 1. Epoch** One full pass through the dataset; more epochs improve learning but can cause overfitting.
- 2. Loss Function** Measures how far predictions are from actual values; lower loss means better performance.
- 3. Encodings** Converts text into numerical representations for model processing using methods like one-hot encoding or embeddings.
- 4. Tokens** Smallest text units (words or subwords) processed by the model, helping manage variable-length inputs.
- 5. Accuracy** Measures how often predictions are correct but should be balanced with loss for a full performance picture.

Prompt Example:

Use Hugging Face Transformers and tensorflow to train a sentiment classification model:

- Model: distilbert-base-uncased-finetuned-sst-2-english.
- Group Positive and Neutral sentiments together.
- Provide metrics to evaluate: Accuracy, Precision, Recall, F1-score.



Overfitting vs. Underfitting in Machine Learning Models

What is Overfitting?

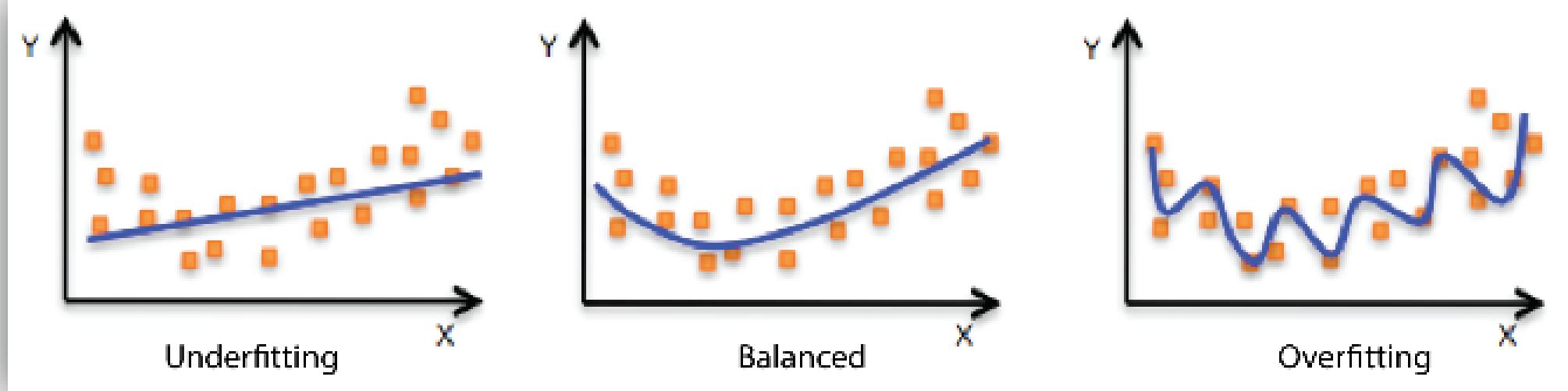
- **Definition:** When a model performs well on training data but poorly on unseen data.
- **Indicators:**
 - High training accuracy, low validation/test accuracy.
 - Large gap between training and validation loss.
- **Causes:**
 - Model memorizes training data instead of generalizing patterns.
 - Excessive model complexity (e.g., too many layers or parameters).

What is Underfitting?

- **Definition:** When a model performs poorly on both training and unseen data.
- **Indicators:**
 - Low accuracy on training data.
 - Training and validation loss remain high.
- **Causes:**
 - Model fails to capture underlying patterns in the data.
 - Model is too simple (e.g., insufficient layers or parameters).

Key Comparison

Overfitting	Underfitting
Memorizes training data	Fails to learn patterns
High training accuracy, low test accuracy	Low accuracy overall
Caused by excessive complexity	Caused by insufficient complexity



Strategies to Address

1. For Overfitting:

- Data augmentation, dropout, reduce model complexity, K-fold cross-validation.

2. For Underfitting:

- Use a more complex model, increase training time, improve feature engineering, or collect more data.

Workflow Steps

Step 1: Data Cleaning and Preprocessing

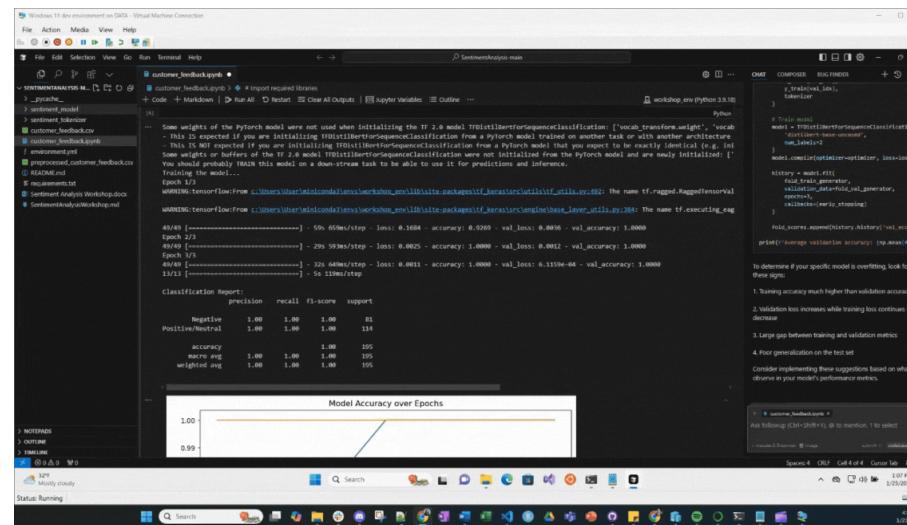
Step 2: Model Training and Evaluation

Step 3: Deployment and Prediction

Prompt Example:

Deploy a simple FastAPI application for sentiment analysis with:

1. TensorFlow integration.
2. Environment variables to suppress TensorFlow warnings.
3. Endpoint: "/predict" to classify sentiment.
4. An application with an app.py and app_test.py. The app_test.py file should have a visual output.
5. There should also be an interactive front end with streamlit.



The screenshot shows a Jupyter Notebook interface running on a Windows 11 desktop. The notebook is titled "SentimentAnalysis-main". The code cell contains Python code for training a TensorFlow model using sequence classification. The output pane shows training logs with metrics like accuracy and loss, a classification report table, and a line chart titled "Model Accuracy over Epochs" showing accuracy increasing from approximately 0.85 to 0.95. The desktop taskbar at the bottom shows various open applications like a browser and file explorer.

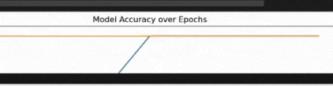
```
File Action Media View Go Run Terminal Help SentimentAnalysis-main + customer_feedback.ipynb - _practical + Code + Markdown | Run A Cell | Restart | Clear All Outputs | Spyder Variables | Outline | workshop-env (Python 3.9.10) + customer_feedback.ipynb - sentiment_tokenizer + customer_feedback.csv + customer_feedback.jsonb + preprocessed_customer_feedback.csv REAMER.mdf requirements.txt SentimentAnalysisWorkshop.ipynb SentimentAnalysisWorkshop.ipynb
```

```
Some weights of the PyTorch model were not used when initializing the TF 2.0 model. In particular: 'vocab_size': This is expected if you are initializing a TF2servingForSequenceClassification from a PyTorch model trained on another task or with another architecture. You may need to manually initialize these weights. Some weights or buffers of the TF 2.0 model TF2servingForSequenceClassification were not initialized from the PyTorch model and are newly initialized: ['cls.predictions.decoder.weight', 'cls.predictions.decoder.bias'] Training the model... Epoch 1/1 40/40 [=====] - 56s 60ms/step - loss: 0.1600 - val_loss: 0.0000 - val_accuracy: 1.0000 40/40 [=====] - 29s 59ms/step - loss: 0.0000 - accuracy: 1.0000 - val_loss: 0.0000 - val_accuracy: 1.0000 Epoch 3/3 40/40 [=====] - 33s 60ms/step - loss: 0.0000 - accuracy: 1.0000 - val_loss: 0.1159e-06 - val_accuracy: 1.0000 13/13 [=====] - 5s 119ms/step
```

```
Classification Report: precision recall f1-score support
```

	precision	recall	f1-score	support
Negative	1.00	1.00	1.00	81
Positive/Neutral	1.00	1.00	1.00	114
accuracy	1.00	1.00	1.00	195
macro avg	1.00	1.00	1.00	195
weighted avg	1.00	1.00	1.00	195

```
Model Accuracy over Epochs
```



```
Model Accuracy over Epochs
```

The chart shows accuracy increasing from approximately 0.85 to 0.95 across 13 epochs.

Conclusion

- **Key Steps:**

- Data preparation.
- Model training and evaluation.
- FastAPI deployment.

- **Takeaways:**

- High-quality data is critical.
- Deployment enables real-time applications.
- Simplifying sentiment categories impacts granularity.

Thank You!

- Slides are located in **SentimentAnalysisWorkshop.pdf**
- A completed project is located in **CompletedProject.zip**



Kristy Wedel
Learning Experience Manager |
Transforming Data Governance & An...

