# Project Deliverable D1.3

# Sentiment Analysis for Financial News

## Team Information

**Team Name:** Finance bros
**Members:**

- **Makeev Roman** - r.makeev@innopolis.university
- **Martyshov Maxim** - m.martyshov@innopolis.university
- **Smirnov Elisey** - el.smirnov@innopolis.university

## Repository Link (Has been changed)

*https://github.com/ForYourEyesOnlyyy/Sentiment_Analysis_for_Financial_News*

## Progress Overview

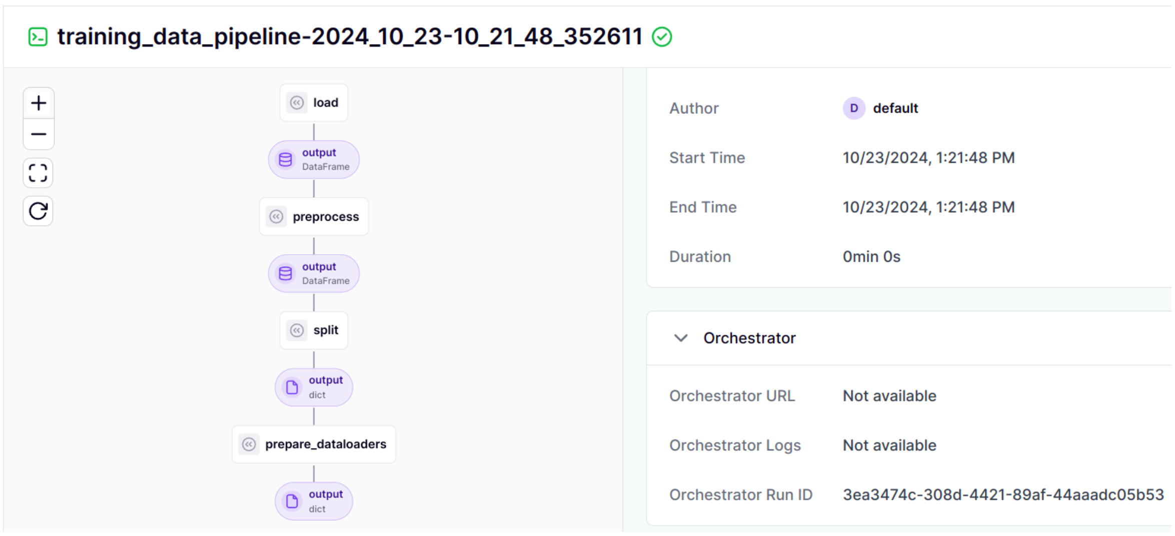
1. **ETL Pipeline Implementation with ZenML**

   We implemented an Extract-Transform-Load (ETL) pipeline using ZenML to streamline our data processing workflow. This ETL pipeline ensures our data is efficiently prepared for training and evaluation, enhancing both consistency and reusability. The pipeline is divided into the following stages, each responsible for a specific part of the data processing flow:

   a. **Load**: This stage extracts raw financial news data from our sources, ensuring the data structure aligns with subsequent processing requirements. ZenML's efficient data connectors facilitate seamless access to stored datasets.
   b. **Preprocess**: At this stage, the raw data undergoes essential transformations, including cleaning, tokenization, and standardization. This step is crucial in transforming text data into a format suitable for natural language processing tasks, handling aspects like punctuation removal, case normalization, and initial token preparation.
   c. **Split**: The data is split into training, validation, and test sets, ensuring our model has balanced and representative subsets for training and evaluation. ZenML's splitting mechanisms support various strategies, allowing us to create customized data splits based on specific project needs.

d. **Prepare Dataloaders**: In this final stage, we prepare the data for model input by creating PyTorch dataloaders, which handle batching, shuffling, and other data-loading optimizations. This step ensures that the model can efficiently process data in manageable batches during training and validation.

The entire pipeline is orchestrated in *training_data_pipeline.py*, enabling seamless re-running of each stage as new data becomes available or preprocessing parameters change. This modularity and reproducibility help ensure our data processing is consistent across all experiments and iterations.



training_data_pipeline-2024_10_23-10_21_48_352611

| | |
|---|---|
| Author | D default |
| Start Time | 10/23/2024, 1:21:48 PM |
| End Time | 10/23/2024, 1:21:48 PM |
| Duration | 0min 0s |

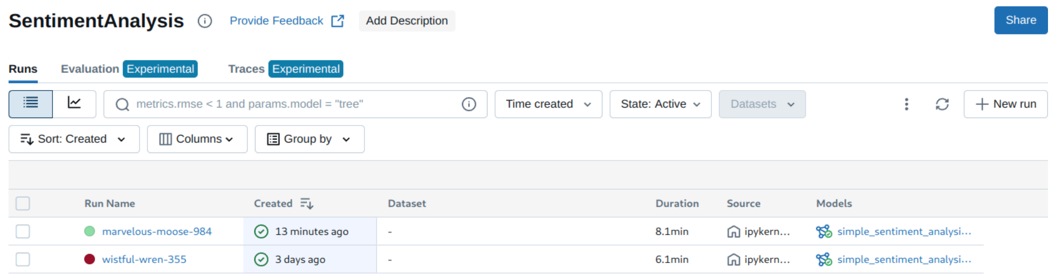
Orchestrator

| | |
|---|---|
| Orchestrator URL | Not available |
| Orchestrator Logs | Not available |
| Orchestrator Run ID | 3ea3474c-308d-4421-89af-44aaadc05b53 |

2. **Model Experimentation and Tracking with MLflow**

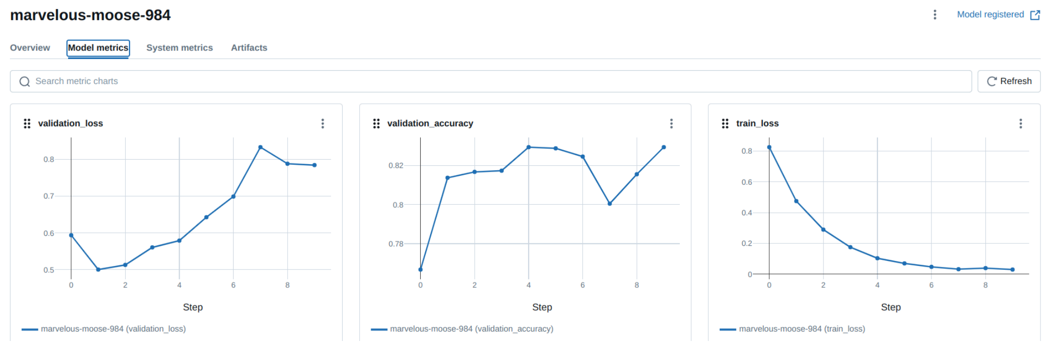
To manage and monitor our model experiments systematically, we integrated MLflow for comprehensive model tracking and version control. This setup allows us to experiment with various model architectures and hyperparameter configurations, all while maintaining a clear record of each experiment’s performance.

a. **Experiment Tracking and Metrics Logging** Using MLflow’s tracking capabilities, we have systematically logged a variety of models, including each model's parameters, configurations, and output metrics across training and validation stages.

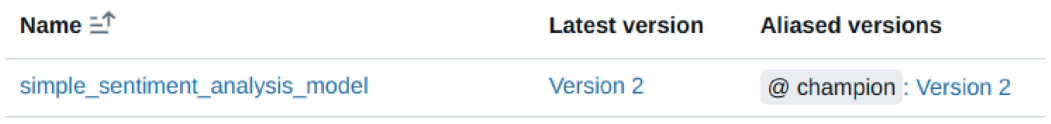


| | Run Name | Created | Dataset | Duration | Source | Models |
|---|---|---|---|---|---|---|
| | ● marvelous-moose-984 | ✓ 13 minutes ago | - | 8.1min | ipykern... | simple_sentiment_analysi... |
| | ● wistful-wren-355 | ✓ 3 days ago | - | 6.1min | ipykern... | simple_sentiment_analysi... |

b. **Model Metrics Visualization MLflow** also provides dynamic graphs of the metrics for each experiment, allowing us to analyze trends over time directly within the MLflow interface. By visualizing these metrics, we can better understand how different models perform in real-time, making it easier to identify areas for improvement and track the impact of various adjustments. This visualization feature has been invaluable in comparing experiments and honing in on the best-performing models.



c. **Automatic Model Selection with Champion Tagging** Once all models have been evaluated, MLflow automatically tags the best-performing model with the *@champion* alias, designating it as the current best candidate for deployment. This automation saves time by clearly highlighting the model with the most promising performance and keeps our workflow organized by providing a clear 'champion' model for future deployment and testing



| Name | Latest version | Aliased versions |
|---|---|---|
| simple_sentiment_analysis_model | Version 2 | @ champion : Version 2 |

By leveraging MLflow, our team has streamlined the experimentation process, making it easier to track, evaluate, and select models in a structured and efficient manner. All details are in *model_experiments.ipynb*

## Work Distribution:

**Roman Makeev** - Integrated MLflow for tracking experiments, logging metrics, and automating the tagging of the top model as *@champion*.

**Elisey Smirnov** - Tested the ZenML and MLflow setups, ran model experiments, visualized metrics, and adjusted hyperparameters based on results.
**Maxim Martyshov** - Developed the ETL pipeline using ZenML, managing the data loading, preprocessing, splitting, and dataloader preparation stages in *training_data_pipeline.py*.

## Plan for the Next Weeks

1. Finalize model deployment setup.
2. Write documentation and usage guidelines.
3. Develop automation scripts for streamlined deployment and data processing.