**AI-Powered Content Analysis & Recommendation - Round 2 Report**[**GitHub Repository Link**](https://github.com/92Suhail/Codrelate--2025.git)

**1. Problem Definition**

The task involves building a robust and scalable AI-powered system for content analysis and recommendation based on a large corpus of Medium articles. The core goals include:

* **Tag Modeling:** Predict appropriate tags for an article using its title and content.
* **Engagement Prediction:** Forecast the popularity (high or low) of an article based on features such as title, tags, and reading time.
* **Keyword Extraction:** Automatically extract meaningful keywords to summarize articles.
* **Frontend Visualization:** Deploy a user-friendly interface to showcase model results.

This is a **multi-task problem**, involving multi-label classification (tag modeling), binary classification (engagement prediction), and unsupervised keyword extraction using NLP.

**2. Methodology**

**2.1 Data Preparation**

* **Data Cleaning** (performed in Round 1):
  + Removed null/missing values.
  + Converted list-type columns (tags, authors) to Python lists.
  + Standardized timestamps.
* **Feature Engineering**:
  + reading\_time: Estimated from word count assuming 200 WPM.
  + title\_len, num\_tags, weekday: Engineered features from existing columns.

**2.2 Tag Modeling (Multi-Label Classification)**

* Used TF-IDF vectorization on combined title + text.
* MultiLabelBinarizer used for encoding tag labels.
* Model: MultiOutputClassifier with Logistic Regression.
* Evaluation: F1-score (micro).

**2.3 Engagement Prediction (Binary Classification)**

* Target: Binary label (is\_popular) based on median claps.
* Features: reading\_time, title\_len, num\_tags, weekday.
* Model: XGBoostClassifier with hyperparameter tuning via GridSearchCV.
* Evaluation: Accuracy, F1-score.

**2.4 Keyword Extraction**

* TF-IDF scoring used to extract top N keywords per article.
* Used TfidfVectorizer on individual articles.

**2.5 Explainability**

* Used SHAP (SHapley Additive exPlanations) to explain predictions of the engagement model.
* Generated SHAP summary plots.

**2.6 Frontend Deployment**

* Created a Streamlit-based interactive UI:
  + Input: Article content.
  + Output: Predicted popularity.
* Models and vectorizers saved using joblib.

**3. Results & Metrics**

**3.1 Tag Modeling**

* **Model:** Logistic Regression
* **F1-score (micro):** ~0.69

**3.2 Engagement Prediction**

* **Model:** XGBoost
* **Accuracy:** ~0.76
* **F1-score:** ~0.78

**3.3 Keyword Extraction**

* Works effectively with TF-IDF for summarizing content.

**3.4 Explainability**

* SHAP values reveal key contributors:
  + Higher reading\_time and num\_tags => Higher popularity
  + Short title\_len often indicates lower engagement

**4. Key Insights**

* **Title and Text** are strong predictors for both tags and engagement.
* **Tags** influence content discovery significantly—quality tagging improves reach.
* **Reading Time** plays a critical role—longer content, if valuable, drives higher engagement.
* SHAP enhances model trustworthiness and helps authors optimize for impact.

**5. Next Steps / Recommendations**

* Use LLMs like BERT for improved tag prediction and semantic keyword extraction.
* Implement author influence analysis using article metrics.
* Integrate collaborative filtering for personalized recommendations.
* Expand frontend to support:
  + Article similarity search
  + Real-time tag suggestion
  + Optimization tips for authors

**6. Tools & Technologies**

* Python, Pandas, NumPy
* Scikit-learn, XGBoost, SHAP
* TF-IDF (sklearn)
* Streamlit (Frontend Deployment)
* Joblib (Model Persistence)

**7. Code Structure**

* notebooks/round2\_model.ipynb — Model building & evaluation
* models/ — Saved models (tag classifier, engagement predictor)
* streamlit\_app.py — Frontend app
* README.md — Project documentation

All code is modular, well-commented, and documented for future development.

**8. Conclusion**

This solution demonstrates a scalable, explainable, and innovative system for analyzing and recommending Medium articles. By combining NLP, ML, explainability, and frontend design, we’ve created a well-rounded prototype ready for further productionization.