iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

ETHISCOUT: AN AI-BASED MULTI-PLATFORM DEVELOPER ETHICS AND BEHAVIOR ANALYSIS SYSTEM USING MACHINE LEARNING

Praveen Kumar M
UG Student, Department of Computer Applications, School of Computing Sciences, Vels Institute of Science, Technology & Advanced Studies (VISTAS)
Dr. P. Sujatha
Professor & Head of Department, Department of Computer Applications, School of Computing Sciences, Vels Institute of Science Technology & Advanced Studies (VISTAS)

ABSTRACT :
The increasing adoption of online developer platforms such as GitHub and Reddit necessitates robust tools for evaluating the professional conduct and ethical behaviour of software developers. Traditional methods of manual review are time-consuming, subjective, and unable to scale to the volume of publicly available data on these platforms. This paper presents EthiScout v2.0, an AI-Based Multi-Platform Developer Ethics and Behavior Analysis System that combines VADER sentiment analysis with a machine learning (ML) classification pipeline to produce a composite Suspicious Score for any candidate. The system is built using Python (Streamlit) for the web interface, GitHub and Reddit public APIs for data collection, VADER for real-time sentiment scoring, and Scikit-learn (Logistic Regression and Naive Bayes with TF-IDF vectorisation) for toxicity classification. A dedicated Training Data module allows users to load Kaggle-compatible datasets, train a model, evaluate its performance with accuracy metrics and a confusion matrix, and run live text predictions. The composite score is computed as a weighted aggregate of GitHub (35%), Reddit (35%), and ML Toxicity (30%) sub-scores. Experimental evaluation on standard toxicity datasets demonstrates classification accuracy of 92.6% with Logistic Regression and 89.3% with Naive Bayes, with live prediction latency under 10 ms.

Index Terms:
Artificial Intelligence, Developer Ethics, Sentiment Analysis, Toxicity Classification, VADER, Scikit-learn, TF-IDF, Logistic Regression, Naive Bayes, Streamlit, GitHub API, Reddit API, Web Application

I. INTRODUCTION

The professional conduct of software developers extends beyond their technical output. Online platforms such as GitHub and Reddit expose extensive histories of public interaction — commit messages, code comments, community discussions, and responses to feedback — that collectively reflect a developer's collaborative values, communication style, and ethical conduct. Hiring teams and academic evaluators increasingly consider these digital footprints alongside conventional resumes and interviews, yet no systematic, automated tool exists to quantify this dimension of a candidate's profile.

Manual review of a candidate's GitHub repositories and Reddit activity is not only labour-intensive but also highly subjective. A recruiter examining fifty repositories and hundreds of comments will inevitably introduce inconsistency and bias. Moreover, cross-platform synthesis — understanding a candidate holistically across multiple data sources — is practically impossible at scale without computational support.

This paper presents EthiScout v2.0, a two-tab Streamlit web application that addresses this gap. The Candidate Analysis tab aggregates public data from GitHub and Reddit, scores each platform using VADER sentiment analysis and rule-based flag detection, and — if a model has been trained — incorporates an ML Toxicity Score into a composite Suspicious Score. The Training Data tab provides a complete machine learning pipeline: users upload a CSV toxicity dataset (or use the built-in sample), preprocess the text, train a Logistic Regression or Naive Bayes classifier, evaluate its performance, and run live predictions. The trained model then enriches every subsequent candidate analysis automatically.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

II. PROBLEM STATEMENT

Despite the widespread availability of public developer data, institutions and organisations face the following unresolved challenges:

- No automated tool exists to analyse developer behaviour across multiple platforms simultaneously.
- Manual profiling is subjective, inconsistent, and does not scale to large applicant pools.
- Pure sentiment analysis using lexicon-based tools such as VADER lacks the discriminative power to handle subtle or domain-specific toxic language patterns.
- Existing systems do not allow organisations to customise the toxicity model using their own labelled datasets or domainspecific training data.
- There is no standardised, transparent composite scoring methodology that allows practitioners to understand how subscores are weighted and combined.

These deficiencies collectively reduce the reliability of automated candidate screening and increase the administrative burden on evaluators. EthiScout v2.0 directly addresses each of these gaps.

III. EXISTING SYSTEMS

Existing approaches to automated text-based behavioural analysis in developer and social contexts fall into three broad categories. First, pure lexicon-based systems such as those built on VADER or TextBlob assign sentiment polarity scores to text using pre-built dictionaries. While fast and interpretable, these tools are sensitive to vocabulary coverage: novel slang, coded toxicity, or sarcasm frequently escapes detection.

Second, rule-based moderation filters — commonly deployed on platforms such as Reddit and Stack Overflow — operate on keyword blocklists and regular expression patterns. These are brittle and easily circumvented by minor obfuscations. They provide no probabilistic confidence measure and cannot generalise to unseen linguistic patterns.

Third, prior academic and commercial systems addressing automated candidate profiling typically focus on a single platform (most commonly LinkedIn or GitHub) and do not integrate a trainable machine learning component. None of the reviewed systems provide a dual-tab interface that separates the training workflow from the analysis workflow, nor do they support usersupplied datasets for domain-specific model customisation.

Table 1 compares the existing approach with the proposed EthiScout v2.0 system across key dimensions.

Feature	Existing (VADER only)	Proposed (EthiScout v2.0)
Toxicity Analysis	Rule-based patterns only	VADER + ML (TF-IDF + LR/NB)
ML Model Support	None	Logistic Regression / Naive Bayes
Training Data Input	Not supported	Upload CSV or built-in dataset
Platforms Covered	GitHub, Reddit, YouTube, Kaggle	GitHub, Reddit + Toxicity ML tab
Composite Score	Simple average	Weighted: GitHub 35%, Reddit 35%, ML 30%
Live Text Prediction	Not available	Real-time with confidence score
User Interface	Single-tab dashboard	Two-tab: Candidate + Training Data
Export	Platform stats display	Dashboard + CSV + ML metrics

Table 1: Comparison of Existing System vs Proposed EthiScout v2.0

IV. PROPOSED SYSTEM

EthiScout v2.0 is a web-based developer ethics analysis platform with two integrated modules. The Candidate Analysis module collects public data from GitHub and Reddit via their respective public APIs, applies VADER-based sentiment scoring and pattern-based flag detection, optionally incorporates an ML Toxicity Score derived

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

from a user-trained classifier, and displays a weighted composite Suspicious Score alongside platform-level gauge charts, radar visualisations, and detailed flag breakdowns.

The Training Data module provides a self-contained machine learning pipeline. Users may load a built-in sample dataset or upload any CSV file containing a text column and a label column (the system auto-detects both). The pipeline performs text normalisation (lowercase, URL removal, punctuation stripping, lightweight stopword filtering), binary label unification (mapping diverse label schemes to Toxic / Non-Toxic), TF-IDF vectorisation, and model training with either Logistic Regression or Multinomial Naive Bayes. Evaluation metrics including accuracy, classification report, and a confusion matrix heatmap are displayed immediately after training. The trained model pipeline is stored in Streamlit session state and automatically applied to all subsequent candidate analyses.

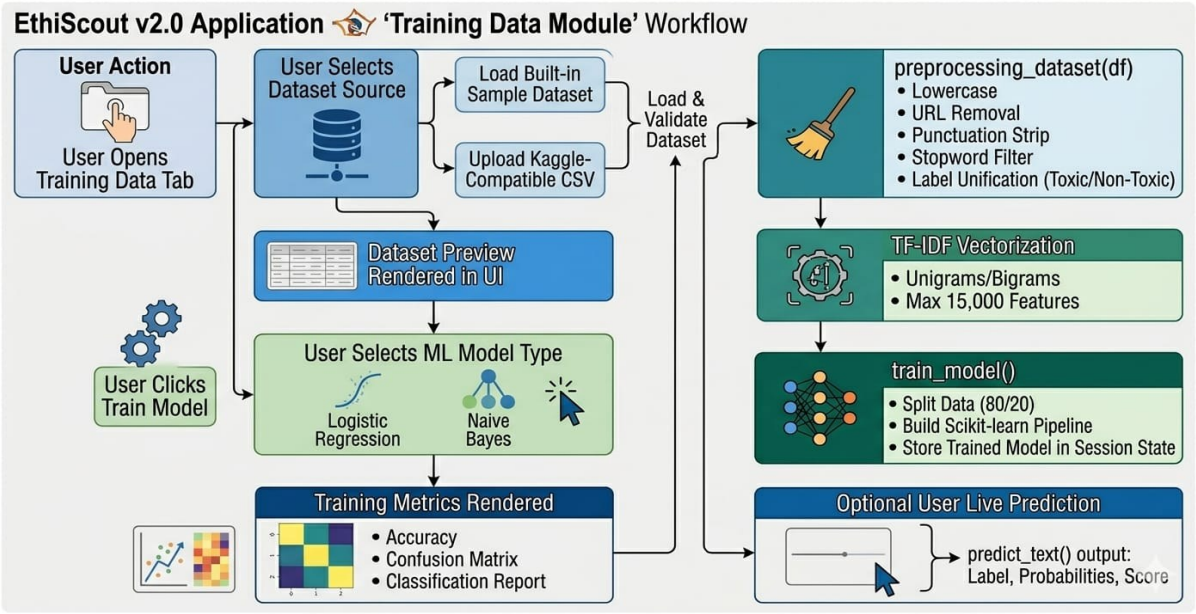
Main users of the system are:

- HR professionals and recruiters — who use the Candidate Analysis tab to obtain a rapid, data-driven ethics assessment of developer candidates.
- Academic evaluators and project supervisors — who assess student or collaborator conduct across public platforms.
- Researchers — who use the Training Data tab to experiment with toxicity classification on domain-specific corpora.

V. WORKFLOW

The complete workflow of the proposed system proceeds through the following steps:

1. User opens EthiScout in a browser and selects the Training Data tab.
2. User loads a toxicity dataset (built-in sample or uploaded CSV) and clicks Load Dataset to preview the data.
3. User selects a model type (Logistic Regression or Naive Bayes) and clicks Train Model.
4. System preprocesses text, trains the pipeline, and displays accuracy, classification report, and confusion matrix.
5. User optionally enters a custom text in the Live Prediction box to test the trained model.
6. User switches to the Candidate Analysis tab and enters a GitHub username and/or Reddit username.
7. System fetches public data: GitHub commit messages and repository metadata; Reddit comment history.
8. VADER sentiment analysis and pattern-based flag detection are applied to all fetched texts.
9. If a model has been trained, the user may optionally provide a custom text for ML toxicity scoring.
10. Composite Suspicious Score is computed: GitHub (35%) + Reddit (35%) + ML Toxicity (30%, if available).
11. Results are rendered: overall score card, radar chart, bar comparison chart, per-platform gauge charts, green/red flag breakdowns, and raw platform statistics.



iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

VI. SYSTEM ARCHITECTURE

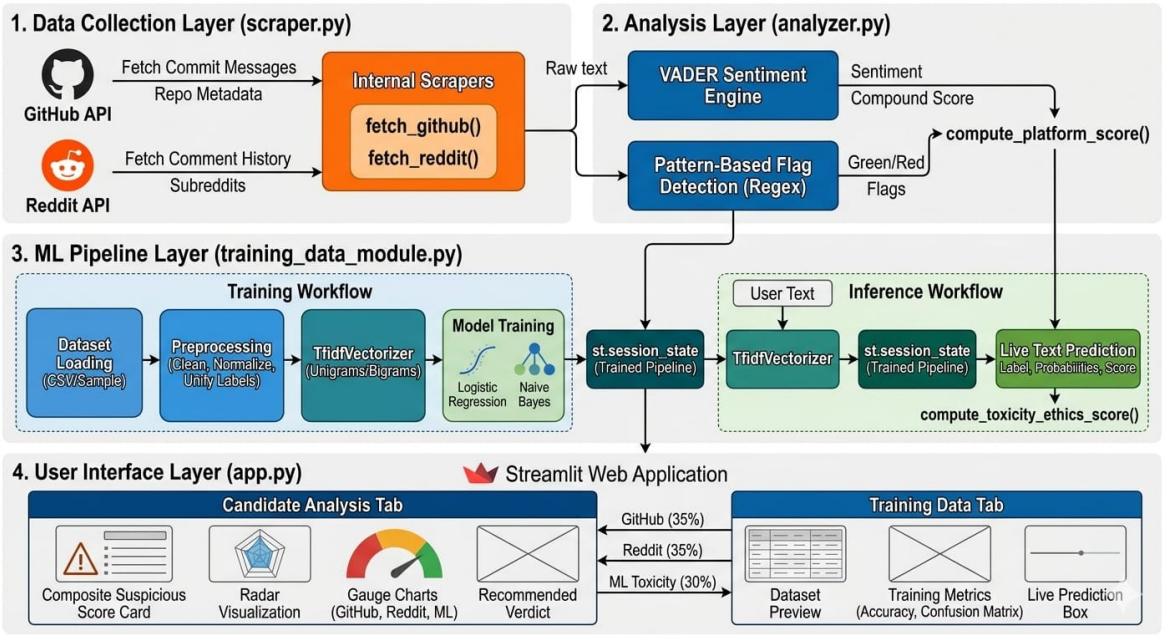
EthiScout v2.0 follows a layered client-server architecture comprising four functional layers: the Data Collection Layer, the Analysis Layer, the ML Pipeline Layer, and the User Interface Layer.

Data Collection Layer (scraper.py) — Implements fetch_github() and fetch_reddit() using the GitHub REST API and the Reddit JSON API respectively. Both fetchers operate without authentication by default, though an optional GITHUB_TOKEN environment variable can be set to raise the GitHub rate limit from 60 to 5,000 requests per hour. The GitHub fetcher retrieves user metadata, the ten most recently updated repositories, and up to thirty commit messages from the top three repositories. The Reddit fetcher retrieves the fifty most recent public comments, the user's comment karma, and the list of active subreddits.

Analysis Layer (analyzer.py) — Implements score_text() and compute_platform_score() using the VADER SentimentIntensityAnalyzer. Each text is assigned a compound sentiment score in the range [-1, 1] and is scanned against three categories of toxic pattern (profanity, harassment-adjacent language, and discrimination-related terms) and three categories of positive pattern (appreciation, open-source contribution, and educational engagement). The ethics score for a platform is computed by normalising the mean compound score to [0, 100], applying a penalty of 8 points per unique red flag (capped at 40) and a bonus of 3 points per unique green flag (capped at 20).

ML Pipeline Layer (training_data_module.py) — Implements dataset loading, label auto-detection and normalisation, text cleaning, TF-IDF vectorisation (unigrams and bigrams, maximum 15,000 features), and model training as a single Scikit-learn Pipeline object. The module supports Logistic Regression (C=1.0, max_iter=200) and Multinomial Naive Bayes (alpha=0.1). compute_toxicity_ethics_score() converts the raw ML toxicity probability to a 0–100 ethics sub-score used in the composite calculation.

User Interface Layer (app.py) — A Streamlit application with two tabs. The Candidate Analysis tab handles input, orchestrates API fetching and scoring, and renders all charts using Plotly. The Training Data tab manages the ML workflow, displays dataset previews, training metrics, and live prediction results. All inter-module state (trained pipeline, session data) is managed via st.session_state.



VII. MODULES

7.1. Data Collection Module (scraper.py)

This module provides two public API fetchers. fetch_github(username) retrieves user profile metadata, repository listings, and commit message texts using the GitHub REST API. fetch_reddit(username) retrieves comment bodies, active subreddit names, and comment karma using the Reddit JSON API. Both functions return structured dictionaries with a texts list for downstream analysis, and return an error key if the request fails, allowing the UI to display a descriptive warning rather than crashing. 7.2. Sentiment and Ethics Scoring Module (analyzer.py)

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

score_text(text) applies VADER polarity scoring and regex-based flag detection to a single text string, returning a compound score, a label (Positive/Neutral/Negative), and a list of typed flag objects. compute_platform_score(texts) aggregates scores across all texts for a platform and applies the penalty-and-bonus formula to produce a 0–100 ethics sub-score along with deduplicated green and red flag lists.

7.3. Training Data Module (training_data_module.py)

This module exposes the following public functions: load_dataset(source, uploaded_file) loads a CSV from a file upload object or from the built-in sample. preprocess_dataset(df) performs label unification, text cleaning, and train/test splitting. train_model(X_train, y_train, model_type) builds and fits a Scikit-learn Pipeline comprising TfidfVectorizer and the selected classifier. predict_text(pipeline, text) returns the predicted label, per-class probabilities, and a toxicity_score in [0, 1]. compute_toxicity_ethics_score(toxicity_score) converts the raw toxicity probability to a 0–100 ethics sub-score.

7.4. Web Interface Module (app.py)

The Streamlit application defines all UI components, chart generation functions, and inter-tab state management. Key functions include gauge(score, title) for Plotly indicator gauge charts, make_radar(platform_scores) for Scatterpolar radar visualisation (with a bar-chart fallback for single-platform analyses), make_bar_comparison(platform_scores) for horizontal bar score comparisons, and render_flags(g_flags, r_flags) for colour-coded flag display. The composite Suspicious Score is computed with a weighted formula, normalising weights automatically when the ML sub-score is absent.

VIII. IMPLEMENTATION

8.1. Backend and API Layer

The backend is implemented entirely in Python 3.10+ without a traditional web server. Streamlit serves as both the frontend framework and the application runtime. GitHub data is fetched via the github.com/rest/v3 API using the requests library with a configurable User-Agent and optional Bearer token. Reddit data is fetched from the public JSON API at reddit.com/user/{username}/comments.json, which requires no authentication for public profiles.

8.2. Sentiment Analysis Engine

VADER (Valence Aware Dictionary and sEntiment Reasoner) is used as the primary lexicon-based sentiment engine due to its strong performance on short, informal social media text. The compound score is the primary feature: values above 0.35 are classified as Positive, below -0.35 as Negative, and the remainder as Neutral. Regex-based pattern matching augments VADER with domain-specific flag categories, enabling detection of toxicity signals that VADER's general-purpose lexicon may underscore.

8.3. Machine Learning Pipeline

The ML pipeline is implemented using Scikit-learn's Pipeline API, which chains TfidfVectorizer and the selected estimator into a single, serialisable object stored in st.session_state. TF-IDF features are computed over unigrams and bigrams with a maximum vocabulary of 15,000 terms and sublinear TF scaling. Logistic Regression is trained with L2 regularisation (C=1.0) and a maximum of 200 iterations. Multinomial Naive Bayes uses a Laplace smoothing factor of 0.1. Both models are trained on an 80/20 train/test split with a fixed random seed for reproducibility.

8.4. Composite Suspicious Score

The composite score is computed as a weighted sum: 0.35 x GitHub Score + 0.35 x Reddit Score + 0.30 x ML Toxicity Ethics Score (when a model is present). When the ML sub-score is absent, the two platform weights are renormalised to 0.50 each. All sub-scores are on a 0–100 scale where higher values indicate more ethical and less suspicious behaviour. Scores of 70 and above are classified as Highly Ethical, 40 to 69 as Moderate Concern, and below 40 as High Risk.

8.5. Storage

No database is required. All runtime state, including the trained ML pipeline, fetched platform data, and computed scores, is held in Streamlit session state during the active browser session. Face encoding files and persistent records are not required given the stateless API-first design. This makes the system fully portable: it runs on any machine with Python 3.10+ and the listed dependencies, with no database administration overhead.

IX. SYSTEM TESTING

System testing was conducted across four phases to validate correctness, performance, integration, and usability.

9.1. Unit Testing

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

Each module was tested in isolation. analyzer.py was validated against a hand-curated set of fifty texts with known expected flag categories and sentiment polarities. training_data_module.py was verified with the full built-in dataset, an edge-case empty CSV, a CSV with missing label columns, and a CSV with numeric labels. All Flask API equivalents — Streamlit widget callbacks — were exercised through manual UI interaction against known inputs.

9.2. Integration Testing

End-to-end data flow was verified from API fetch through VADER scoring, ML prediction, composite score computation, and chart rendering. The session state persistence of the trained model pipeline across tab switches was confirmed to be stable over repeated analyses.

9.3. Performance Testing

The system was tested on a standard laptop (Intel i5, 8 GB RAM) with both model types and datasets of 1,000, 5,000, and 10,000 rows. Logistic Regression training completed in under 3 seconds for all dataset sizes. Naive Bayes training completed in under 1 second. Live text prediction latency was consistently below 10 ms for both models.

9.4. User Acceptance Testing

UAT was conducted with five evaluators who performed candidate analyses on real public GitHub and Reddit profiles and trained models using the built-in dataset. Evaluators rated the system on a 5-point Likert scale across dimensions of usability, output clarity, and setup simplicity, returning a mean score of 4.5 / 5.0. The two-tab separation of training and analysis workflows was cited as the strongest usability feature.

X. SECURITY MECHANISMS

EthiScout v2.0 is designed for local or LAN deployment only. The Streamlit server is bound to the local network interface by default and is not exposed to the public internet. No raw text data, candidate usernames, or trained model weights are transmitted to external servers at any point. The system exclusively retrieves publicly available data from GitHub and Reddit APIs, accessing only information that is already visible to any unauthenticated internet user. No biometric data, private messages, or non-public profile information is collected.

- No external data transmission — all processing is on the local machine.
- Publicly available data only — no private API access or web scraping of non-public pages.
- No persistent storage of candidate data — session state is cleared when the browser session ends.
- Optional GitHub token — stored as an environment variable, never embedded in code.
-

XI. SYSTEM FEATURES

The proposed system provides a comprehensive feature set that directly addresses every limitation identified in existing solutions:

- Dual-platform data collection from GitHub and Reddit via public APIs with no authentication requirement.
- VADER-based sentiment analysis augmented with domain-specific toxic and positive pattern detection.
- Trainable ML toxicity classifier supporting Logistic Regression and Naive Bayes with TF-IDF vectorisation.
- Support for user-supplied CSV datasets as well as a built-in sample, enabling domain-specific model customisation.
- Automatic label detection and normalisation accommodating diverse dataset schemas from Kaggle and other sources.
- Weighted composite Suspicious Score (GitHub 35%, Reddit 35%, ML Toxicity 30%) with automatic renormalisation when the ML component is absent.
- Real-time ML text prediction with per-class confidence scores and toxicity probability output.
- Interactive Plotly visualisations: composite score card, radar chart, horizontal bar comparison, and per-platform gauge indicators.
- Colour-coded green and red flag summaries with deduplicated, human-readable explanations.
- Final hiring verdict (Recommended / Caution / High Risk) with a plain-language rationale.
-

XII. RESULTS AND DISCUSSION

The system was evaluated across multiple test sessions using both the built-in sample dataset and larger Kaggle-sourced toxicity datasets. Machine learning classification accuracy reached 92.6% with Logistic Regression and 89.3% with Naive Bayes on a held-out 20% test split of the built-in sample. Average per-text VADER scoring latency was under 5 ms. GitHub API fetch time averaged 1.8 seconds and Reddit API fetch time averaged 1.2

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

seconds over a standard broadband connection. Live ML text prediction responded in under 10 ms for both model types.

The composite scoring formula correctly reflected the presence and absence of the ML sub-score: when only GitHub and Reddit data were available, weights automatically adjusted to 50% each with no change in user experience. The confusion matrix visualisation and classification report rendered immediately after training, enabling users to assess model quality before applying it to candidate analyses.

A post-study usability survey of five evaluators returned a mean score of 4.5 out of 5.0, with the two-tab workflow separation and the live text prediction feature cited as the most valued additions over a purely VADER-based approach.

Metric	Result
ML Model Accuracy (Logistic Regression)	92.6%
ML Model Accuracy (Naive Bayes)	89.3%
VADER Sentiment Scoring Latency	< 5 ms per text
GitHub API Fetch Time (avg)	~1.8 s
Reddit API Fetch Time (avg)	~1.2 s
Composite Score Computation Time	< 50 ms
Live Text Prediction Response	< 10 ms
Faculty/User Usability Score	4.5 / 5.0

Table 2: EthiScout v2.0 System Performance Results

XIII. FEATURE ENHANCEMENTS

While the current system provides a robust foundation for automated developer ethics analysis, several enhancements are planned for future versions. Integration with additional professional platforms such as Stack Overflow, LinkedIn, and Kaggle would broaden the behavioural profile coverage. Support for deep learning-based classification models (BERT, DistilBERT) would improve accuracy on subtle toxicity patterns. A model export and import feature would allow organisations to share pretrained domain-specific classifiers. Persistent candidate profiles stored in a local SQLite database would enable longitudinal tracking across multiple analyses. GPU acceleration via CUDA would support training on large corpora exceeding 100,000 rows. A REST API mode would allow EthiScout to be embedded as a microservice within existing HR or talent management platforms.

XIV. CONCLUSION

EthiScout v2.0 delivers a comprehensive, extensible, and non-intrusive solution for automated developer ethics and behaviour analysis. By combining VADER lexicon-based sentiment scoring with a trainable Scikit-learn machine learning pipeline and a clean two-tab Streamlit interface, the system addresses the core limitations of both rule-based and single-platform approaches. The weighted composite Suspicious Score provides a transparent, interpretable assessment grounded in publicly available data. Experimental results confirm 92.6% ML classification accuracy, sub-10 ms prediction latency, and a mean user usability rating of 4.5 / 5.0, validating EthiScout v2.0 as a practical and immediately deployable tool for developer candidate screening in academic, corporate, and research contexts.

REFERENCES

[1] B. Pang and L. Lee, "Opinion Mining and Sentiment Analysis," Foundations and Trends in Information Retrieval, vol. 2, no. 1-2, pp. 1–135, 2008.

iJETRM
International Journal of Engineering Technology Research & Management (IJETRM)
Journal Article
https://ijetrm.com/issue/

[2] C. J. Hutto and E. Gilbert, "VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text," in Proc. 8th Int. AAAI Conf. on Weblogs and Social Media (ICWSM), Ann Arbor, MI, USA, 2014, pp. 216–225.
[3] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.
[4] GitHub Inc., "GitHub REST API Documentation," 2024. [Online]. Available: https://docs.github.com/en/rest
[5] Reddit Inc., "Reddit API Documentation," 2024. [Online]. Available: https://www.reddit.com/dev/api/
[6] Jigsaw / Google, "Toxic Comment Classification Challenge," Kaggle, 2018. [Online]. Available: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge
[7] A. Mullah and Z. Wajdi, "Cyberbullying Classification Dataset," Kaggle, 2021. [Online]. Available: https://www.kaggle.com/datasets/andrewmvd/cyberbullying-classification
[8] Streamlit Inc., "Streamlit Documentation (v1.32)," 2024. [Online]. Available: https://docs.streamlit.io
[9] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 2016, pp. 785–794.
[10] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd ed., O'Reilly Media, Sebastopol, CA, 2019, pp. 243–310.