Systems Analysis and Design
Workshop No. 2
System Design Document — Psychopathy Prediction on Twitter

Juan David Bejarano Cristancho 20232020056

Raúl Andrés Diaz Losada 20232020058

Juan David Avila 20232020154

David Sanchez Acero 20232020049

Systems Engineering Program

Francisco José de Caldas District University

October 18, 2025

1. Introduction

This document presents the System Design for the Kaggle competition *"Predicting Psychopathy Based on Twitter Usage"*. Building on the analytical findings from Workshop #1, this design aims to define a robust, modular, and scalable system architecture capable of addressing the high-dimensional, unbalanced, and chaotic nature of the problem.

The design adheres to systems engineering principles, including modularity, maintainability, and adaptability, while also considering the chaotic behaviors and sensitivity factors identified in the previous analysis.

2. 1. Summary of Workshop #1 Findings

The Workshop #1 analysis revealed a system characterized by high complexity, multidimensional data (337 features), temporal dependencies, and strong sensitivity to initial conditions. Key insights included:

- The dataset is unbalanced, with only 3 % of users classified as high-psychopathy individuals.
- Feature engineering introduces sensitivity due to linguistic ambiguities and correlation among variables.

- Model training and evaluation are highly influenced by small variations in parameters (chaotic behavior).
- The competition's feedback mechanisms (leaderboards, cross-validation) act as nonlinear feedback loops.

These findings directly inform the proposed design, emphasizing robustness, adaptability, and chaos-aware modeling strategies.

3. 2. System Requirements

3.1. 2.1 Functional Requirements

- Process and integrate data from 2,927 anonymized Twitter users.
- Handle 337 linguistic, behavioral, and social interaction features.
- Train and validate machine learning models to predict psychopathy levels.
- Implement cross-validation and ensemble mechanisms.
- Generate predictions and evaluation metrics compatible with Kaggle submissions.

3.2. 2.2 Non-Functional Requirements

- **Scalability:** Handle computationally intensive feature processing and model training.
- **Privacy:** Ensure anonymized data is preserved across all pipeline stages.
- **Reproducibility:** Guarantee consistent results under controlled configurations.
- **Maintainability:** Modular and documented code structure for future iterations.
- **Robustness:** Minimize sensitivity to data perturbations or random initialization.

4. 3. High-Level Architecture

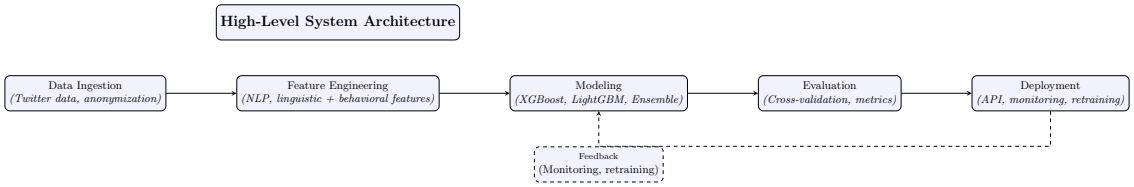

Figura 1: High-level system architecture for Psychopathy Prediction on Twitter

4.1. 3.1 Architectural Overview

The proposed system architecture consists of five main layers:

1. **Data Ingestion Layer:** Responsible for loading, validating, and anonymizing Twitter-derived data.
2. **Feature Engineering Layer:** Applies natural language processing (NLP), behavioral metrics, and temporal feature extraction.
3. **Modeling Layer:** Trains machine learning and ensemble models, handling class imbalance via resampling and cost-sensitive techniques.
4. **Evaluation Layer:** Conducts cross-validation, generates leaderboard-compatible predictions, and stores performance metrics.
5. **Deployment Layer:** Provides an interface for model reusability, monitoring, and continuous evaluation.

4.2. 3.2 Systems Engineering Principles Applied

- **Modularity:** Each layer performs a well-defined function, facilitating independent maintenance.
- **Scalability:** The system supports distributed training using frameworks like *Dask* or *Ray*.
- **Reliability:** Redundant validation mechanisms mitigate overfitting risks.
- **Maintainability:** Clear interfaces between components support iterative refinement.

5. 4. Addressing Sensitivity and Chaos

5.1. 4.1 Sensitivity Control Strategies

To address sensitivity in model behavior and feature variability:

- **Feature Normalization:** Apply standard scaling and variance thresholding to minimize sensitivity to outliers.
- **Regularization:** Use techniques such as L1/L2 regularization to prevent instability from correlated features.
- **Cross-Validation Seeds:** Fix random seeds for controlled reproducibility across experiments.

5.2. 4.2 Chaos Mitigation and Monitoring

Chaos theory implications identified in Workshop #1 are managed through feedback and monitoring strategies:

- **Feedback Control:** Monitor training variance and adjust learning rates dynamically.
- **Ensemble Stability:** Combine models with different random seeds to reduce chaotic divergence.

- **Anomaly Detection:** Integrate monitoring scripts that flag sudden performance bifurcations.
- **Sensitivity Analysis:** Use variance-based global sensitivity measures (Sobol, Morris) to identify influential features.

5.3. 4.3 Managing Nonlinear Feedback Loops

The system's leaderboard feedback mechanism can create emergent overfitting behavior. The design proposes:

- Internal hold-out datasets separate from Kaggle leaderboard data.
- Model averaging across time to counteract feedback oscillations.
- Controlled submission frequency to reduce competitive chaos.

6. 5. Technical Stack and Implementation Plan

6.1. 5.1 Proposed Tools and Frameworks

Component	Technology / Framework
Data Processing	Python (Pandas, NumPy)
Feature Engineering	NLTK, SpaCy, Scikit-learn, LIWC dictionaries
Model Training	Scikit-learn, XGBoost, LightGBM
Evaluation	Cross-validation, SHAP interpretability tools
Deployment	Flask or FastAPI (for REST API)
Version Control	Git + GitHub (Workshop_2_Design repository)
Visualization	Matplotlib, Seaborn, Plotly

Cuadro 1: Technical stack for system implementation

6.2. 5.2 Implementation Plan

1. **Phase 1 – Data Integration:** Collect and validate dataset; perform anonymization checks.
2. **Phase 2 – Feature Engineering:** Apply NLP and behavioral metrics extraction.
3. **Phase 3 – Model Development:** Train multiple models, tune hyperparameters, and evaluate ensemble strategies.
4. **Phase 4 – System Integration:** Connect modules into a cohesive pipeline.
5. **Phase 5 – Deployment and Monitoring:** Expose results through a REST API and integrate error monitoring.

6.3. 5.3 Design Patterns

- **Pipeline Pattern:** Sequentially processes data through independent modules.
- **Observer Pattern:** Enables monitoring components to detect performance anomalies.
- **Factory Pattern:** Supports flexible creation of models with different configurations.

7. 6. Discussion and Conclusions

The proposed system design builds upon the complex dynamics of the psychopathy prediction competition, translating analytical insights into a concrete engineering framework.

Key achievements:

- Designed a modular, chaos-resilient architecture capable of handling high-dimensional linguistic data.
- Integrated sensitivity analysis and feedback control mechanisms.
- Ensured ethical and privacy-preserving data handling.

Future improvements:

- Extend to real-time social media streams.
- Implement federated learning to further enhance privacy.
- Expand interpretability using deep learning explainability tools.

8. References

1. Online Privacy Foundation. (2012). Psychopathy Prediction Based on Twitter Usage. Kaggle Competition.
2. Kossiakoff, A., et al. (2011). *Systems Engineering Principles and Practice*. Wiley.
3. Borgonovo, E. & Plischke, E. (2016). Sensitivity analysis: A review of recent advances. *European Journal of Operational Research*, 248(3), 869–887.
4. Razavi, S., et al. (2021). The Future of Sensitivity Analysis: An essential discipline for systems modeling and policy support. *Environmental Modelling & Software*, 137, 104954.
5. Paulhus, D.L. & Jones, D.N. (2011). Introducing a short measure of the Dark Triad. Poster, Society for Personality and Social Psychology.