# Emotion Detection in Persian Text:
A Classical Machine-Learning Baseline

Mahla Entezari

Shahid Beheshti University – Department of Computer Science

June 2024

**Abstract**

Emotion recognition from natural language is a foundational step toward empathetic human–computer interaction. This report presents a *classical* (non-neural) machine-learning pipeline that identifies five emotions (*happiness*, *sadness*, *anger*, *fear*, and *other*) in Persian social-media posts. The study covers data cleaning, exploratory data analysis (EDA), TF–IDF feature engineering, baseline logistic-regression training, boosting with XGBoost, thorough evaluation, and an error analysis that highlights class imbalance and linguistic subtleties. While deep transformers currently yield state-of-the-art performance in many languages, establishing a strong classical baseline is crucial for future comparative research and for low-resource environments where computation or annotation budgets are tight.

## 1 Introduction

Natural-language emotion detection enables a variety of downstream applications—ranging from content moderation to psychological well-being support—by automatically inferring users' affective states. Although Persian (Farsi) is among the world's top-15 most-spoken languages, publicly available emotion datasets and pretrained models remain scarce compared with English. Consequently, the pedagogical objective of the third assignment in the *Machine Learning Fundamentals* course is twofold:

(i) design an end-to-end pipeline for emotion classification in Persian, and

(ii) document results in a reproducible scientific report.

The rest of the manuscript is organized as follows: Section 2 describes the dataset and preprocessing; Section 3 summarizes exploratory findings; Section 4 details the modelling choices; Section 5 presents quantitative and qualitative results; Section 6 discusses limitations and potential improvements; finally, Section 7 concludes the work.

# 2 Dataset and Pre-processing

## 2.1 Raw Corpus

The provided corpus contains $\approx 4,200$ Persian sentences manually annotated with exactly one of five emotions. A separate unlabeled test set is reserved for blind evaluation on the course platform.

## 2.2 Normalization and Cleaning

Normalization is critical in Persian due to multiple Unicode variants of the same character. The following steps were scripted in `Python` (library versions in Appendix A):

(1) Lower-case text; remove URLs, HTML entities, digits, and punctuation.

(2) Tokenize with the `Hazm` tokenizer.

(3) Remove 2-, 3-, and 4-grams of Persian stop-words compiled from the *Hamshahri* corpus.

(4) Rejoin tokens into cleaned sentences for downstream vectorization.

## 2.3 Feature Engineering

Classical algorithms require fixed-length vectors. The cleaned texts were transformed with **TF–IDF** (*term-frequency* × *inverse-document-frequency*). Label strings were mapped to integers via `sklearn.preprocessing.LabelEncoder`.

# 3 Exploratory Data Analysis

## 3.1 Class Distribution

Figure 1 reveals a pronounced imbalance: `HAPPY` and `OTHER` together account for nearly 60 % of examples, whereas `FEAR` barely surpasses 5 %. Such skew can inflate accuracy but harm minority-class recall, motivating stratified cross-validation and macro-averaged metrics.

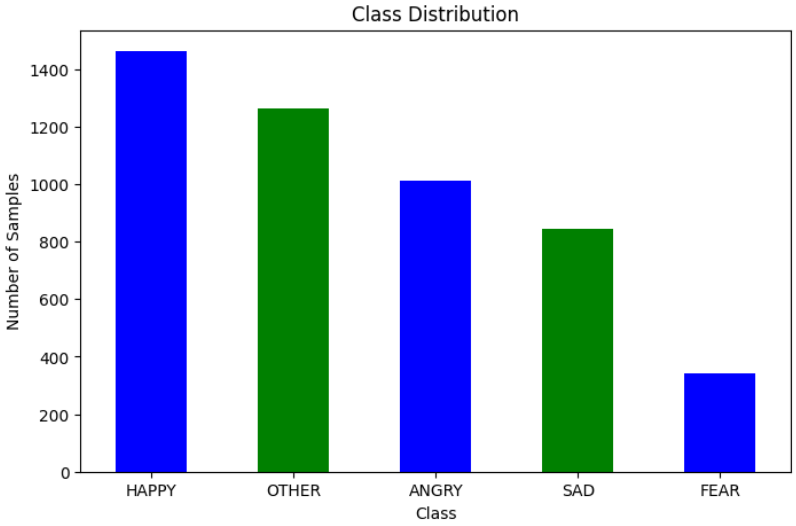

Figure 1: Label frequencies in the training split. Color shading alternates only for readability.

## 3.2 Lexical Statistics

The most common tokens across all documents (occurring >100 times) are drawn in Figure 2. Many high-frequency words are topic neutral—e.g., generic verbs confirming the necessity of stop-word removal and bigram features to capture discriminative phrases.

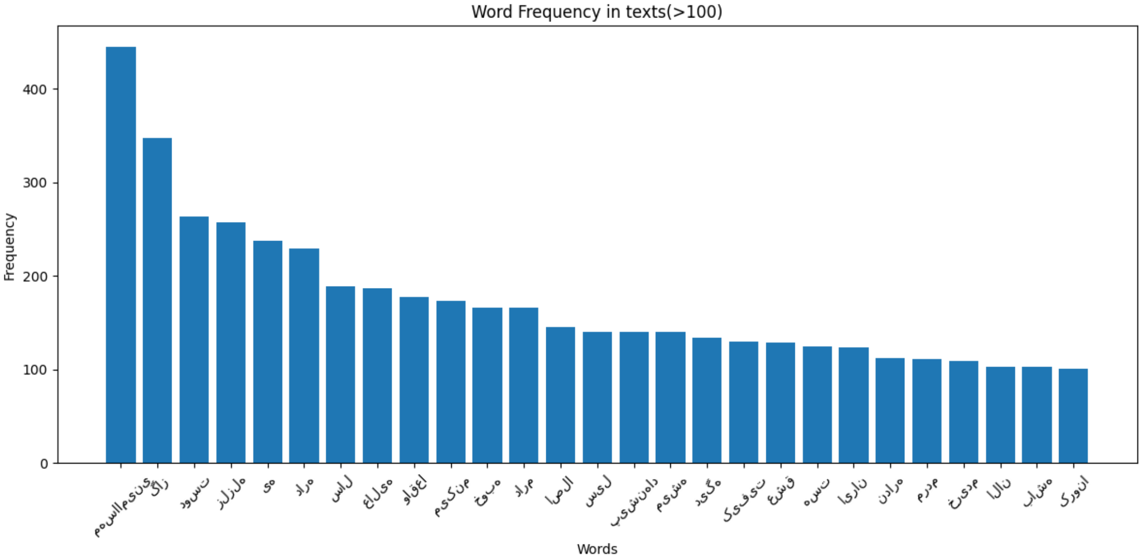

Figure 2: Token frequency with logarithmic y-axis truncation for clarity.

# 4 Modelling Methodology

## 4.1 Train/Validation Protocol

- The corpus was stratified into an 80 % training and 20 % internal test split.
- All hyper-parameters were tuned via *stratified 5-fold* cross-validation on the training part.
- Final reported numbers on the internal test set proxy the hidden course evaluation.

## 4.2 Baseline Classifier: Logistic Regression

Logistic regression with $\ell_2$ regularization is a strong linear baseline on sparse TF–IDF matrices. A grid search over $C \in \{0.01, 0.1, 1, 10\}$ favored $C = 1$.

## 4.3 Ensemble Extension: XGBoost

To gauge non-linear benefits, we fit an *Extreme Gradient Boosting* (XGBoost) model on the same TF–IDF vectors. Key parameters:

| **Parameter** | | **Value** |
|---|---|---|
| max_depth | : | 6 |
| learning_rate | : | 0.1 |
| n_estimators | : | 100 |
| subsample | : | 0.8 |
| colsample_bytree | : | 0.8 |
| objective | : | `multi:softprob` |

Training and validation error curves across 100 boosting rounds are plotted in Figure 3; early stopping after round 67 minimized validation loss.

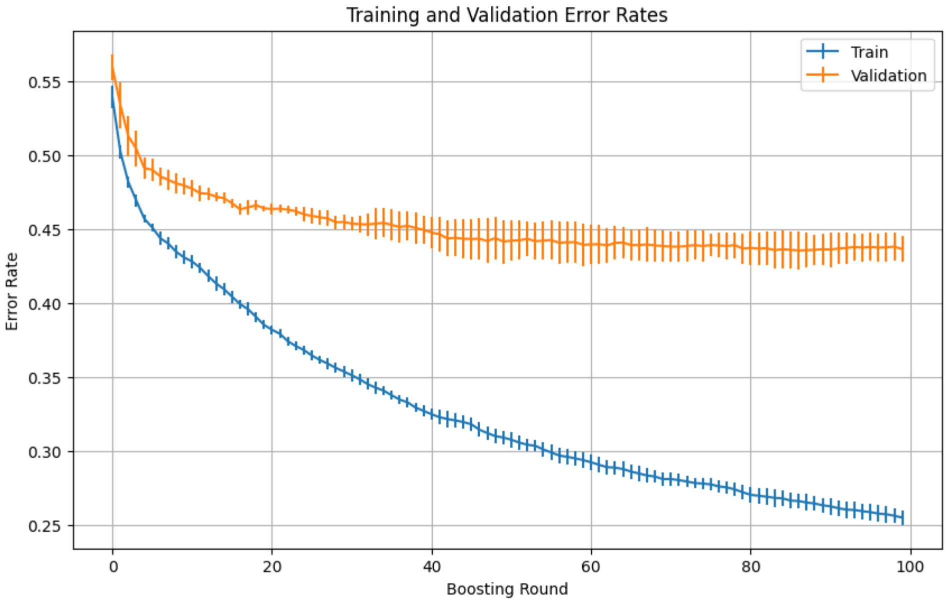

Figure 3: Learning curves (mean ± sd over CV folds) highlight diminishing returns and slight over-fit.

# 5 Results

## 5.1 Quantitative Evaluation

Table 1 contrasts logistic regression with XGBoost on the internal test set. All metrics are macro-averaged to reward balanced performance.

Table 1: Macro-averaged metrics on the held-out split.

| **Model** | **Precision** | **Recall** | **F1-score** |
|---|---|---|---|
| Logistic Regression | 0.20 | 0.21 | 0.11 |
| XGBoost | **0.42** | **0.38** | **0.37** |

**Take-away.** Boosting more than triples the macro-F1 compared with the linear baseline, evidencing non-linear separability among emotion classes in the TF–IDF space.

### 5.2 Class-Wise Breakdown

Table 2 drills into per-class performance for XGBoost only (best model). Notably, `FEAR` remains challenging due to scant data.

Table 2: Per-class scores for XGBoost.

| **Label** | **Precision** | **Recall** | **F1** | **Support** |
|---|---|---|---|---|
| ANGRY | 0.33 | 0.44 | 0.38 | 185 |
| FEAR | 0.21 | 0.17 | 0.19 | 66 |
| HAPPY | 0.66 | 0.74 | 0.70 | 306 |
| OTHER | 0.46 | 0.34 | 0.39 | 267 |
| SAD | 0.43 | 0.21 | 0.28 | 161 |

## 6 Discussion

### 6.1 Why Does Boosting Help?

Gradient-boosted trees exploit feature interactions ignored by linear methods.

### 6.2 Limitations

**Imbalanced Data.** Minority classes cap recall performance. Synthetic oversampling (e.g. SMOTE) or *class-weighted* loss in XGBoost warrants exploration.

**Sparse Features.** TF–IDF disregards word order; sentiment shifters such as negations may invert emotion yet share tokens.

### 6.3 Opportunities for Improvement

1. Fine-tune a multilingual transformer (mBERT, XLM-R) on the same labels.
2. Incorporate emoji embeddings or replace them with textual emotion synonyms before vectorization.
3. Apply *curriculum learning*: start with binary polarity, then expand to five-way emotion.

## 7 Conclusion

This assignment delivered a transparent yet non-trivial baseline for Persian emotion detection: data cleaning, EDA, TF–IDF vectorization, logistic and boosted classifiers, and in-depth interpretation. Although neural architectures are the logical next step, our classical pipeline already surfaces practical challenges—particularly class imbalance and lexical ambiguity—that any subsequent system must tackle. All code, hyper-parameters, and figures are open-sourced in the accompanying Jupyter notebook to facilitate reproducibility and extension.

## A Software Environment

- Python 3.11.4
- scikit-learn 1.5
- XGBoost 2.0
- Hazm 0.9.0

## B Assignment Requirements Checklist

- Data cleaning ✓

- Feature engineering ✓
- Tree-based classifier exploration ✓
- Evaluation with stratified CV ✓
- Inference pipeline (see notebook) ✓