Resume Classification

Anshuman Parida[1][1] Maliwal Raghav Kamleshkumar[2][2] Vasu Kodati[3][3]

K Nithin Rohit[4][§] Soham Nandkumar Kulkarni[5][4]

[1]Indian Institute of Technology Jodhpur

{b23es1008, b23ee1079, b23ee1030, b23ee1037, b23ee1098}@iitj.ac.in

Abstract

This project explores text classification of resumes using a class-balanced dataset and TF-IDF feature engineering. The resume texts are preprocessed and vectorized using TF-IDF with both unigrams and bigrams, capturing essential linguistic patterns for downstream modeling.

In addition to lexical features, we incorporate structural and semantic attributes and standardize them before combining with the TF-IDF features into a final sparse matrix. This enriched feature representation enables a comprehensive understanding of each resume.

We experiment with multiple machine learning models for classification, including Logistic Regression, Support Vector Machine (SVM), Random Forest, and XGBoost. Model performance is evaluated using accuracy, precision, recall, and F1-score. Among these, XGBoost consistently shows robust performance across most metrics, making it a strong candidate for deployment in a real-world resume screening system.

This report summarizes the preprocessing pipeline, feature engineering strategies, and classification performance of various models applied to the balanced dataset.

Contents

[1] Implemented data filtering and combined structural and semantic features with the TF-IDF matrix and converted the scaled structural and semantic features to sparse matrices and implemented tree based methods like random forest and XGBOOST.

[2] Contributed to TF-IDF modeling and clustering methods.

[3] Helped create dense embeddings using sentence transformers and did SVM modelling and analysis . [§]Helped validate preprocessing and other clustering methods.

[4] Ran Logistic Regression And tried ways to improve performance by bettering Logistic Regression and other linear methods.Use PCA and TSNE for visualization and understanding .

1 Introduction

The rapid digitization of recruitment processes has made automated resume screening not only a technical challenge but also an ethical imperative. Traditional manual resume reviews are labor-intensive and prone to subjective bias. To address this, we propose a machine learning framework that not only classifies resumes efficiently but also integrates fairness and interpretability into its core design.

This project aims to classify resumes into predefined categories such as software engineering, data science, electrical engineering, etc., based on text features extracted from the resumes.

Resume text is vectorized using the Term Frequency–Inverse Document Frequency (TF-IDF) technique, capturing both unigrams and bigrams to represent the document semantics effectively. Additionally, we engineer structural and semantic features such as average sentence length and domain-relevant embeddings to enhance the model's understanding.

Furthermore, we use Principal Component Analysis (PCA) to visualize high-dimensional feature distributions and audit model decisions across demographic and categorical boundaries, making the pipeline more interpretable and fairness-aware.

In the subsequent sections, we describe the data preprocessing techniques, feature engineering steps, model choices (Logistic Regression, SVM, Random Forest, XGBoost), performance comparisons, and visualization methods. This combination of classical modeling with cloud deployment and fairness auditing represents a practical yet ethically responsible solution for real-world resume classification.

2 Approaches Tried

Our pipeline consists of several stages including data preprocessing, feature engineering, and model evaluation. Each step is designed to ensure both the technical robustness and fairness of the classification task.

2.1 Data Preprocessing

- **Data Filtering:** To tackle class imbalance, we grouped the resumes by their respective categories. Categories with fewer than 30 samples were removed to ensure statistical relevance, and categories with more than 100 samples were capped at 100. This resulted in a class-balanced dataset.
- **Data Export:** The cleaned and balanced dataset was stored in raw dataset.csv for reproducibility and portability.
- **Text Cleaning:** Prior to vectorization, the text was preprocessed by removing stopwords, converting to lowercase, and eliminating non-alphanumeric characters to standardize the input text.

2.2 Feature Engineering

- **TF-IDF Vectorization:** We used TfidfVectorizer to convert resume texts into a high-dimensional feature matrix. Key hyperparameters were:
  - max features = 1200
  - ngram range = (1, 2) to capture both unigrams and bigrams
  - min df = 5, max df = 0.8 to discard rare and overly common terms

- **Structural Features:** Numerical features were extracted from the resumes such as average sentence length, total number of bullet points, and number of words. These features were standardized using StandardScaler.
- **Semantic Features:** We incorporated pre-computed semantic embeddings representing domainlevel resume information. These were also scaled using StandardScaler to ensure consistent feature magnitudes.
- **Feature Concatenation:** All three components — TF-IDF, structural features, and semantic vectors — were combined using horizontal stacking (scipy.sparse.hstack) to form a unified input matrix for classification.

2.3 Modeling and Evaluation

Model Evaluation

We evaluated multiple machine learning classifiers to benchmark performance in resume classification, focusing on accuracy and generalization:

- **Logistic Regression**: Served as our baseline linear model. It was trained using the liblinear solver with L2 regularization and achieved a modest accuracy of **61.7%.**
- **Random Forest**: A tree-based ensemble model with 100 estimators. It effectively captured nonlinear relationships in the textual data and achieved an accuracy of **74.0%.**
- **Gradient Boosting**: This model sequentially builds trees to minimize errors from previous ones. It showed competitive performance with an accuracy of **68.0%.**
- **HistGradientBoosting**: A histogram-based version of gradient boosting that is highly efficient on large datasets. It achieved an accuracy of **74.0%.**
- **XGBoost**: A scalable and regularized gradient-boosted tree algorithm. It was among the top performers with an accuracy of **75.0%.**
- **LightGBM**: Known for its speed and efficiency, LightGBM matched XGBoost with an accuracy of **75.0%**, making it a strong candidate for deployment.
- **Artificial Neural Network (ANN)**: A feedforward neural network trained with backpropagation. Despite being more complex, it achieved a moderate accuracy of **70.0%.**
- **AdaBoost**: A boosting ensemble that focuses on difficult samples. It underperformed in our case with an accuracy of only **17.0%.**

2.4 Evaluation Metrics

Each model was evaluated using the following metrics:

- **Accuracy:** Overall correctness of the model predictions.
- **F1-Score (Macro-Averaged):** Used to ensure balanced evaluation across all classes regardless of support.
- **Confusion Matrix:** To visualize the model's performance on each class.

2.5 Visualization and Interpretability

To interpret the feature space and gain insights into class separability:

- **PCA Visualization:** Principal Component Analysis was applied on the final feature matrix to reduce dimensionality and plot a 2D representation of the resume clusters.
- **Model Fairness Audits:** Visual inspections and category-wise breakdowns were used to ensure there were no major discrepancies in model performance across categories.

3 Results

In this section, we present the results of our resume classification models. We evaluated four different machine learning algorithms: Logistic Regression, Support Vector Machine (SVM), Random Forest, and XGBoost. For each model, we measured the accuracy, precision, recall, and F1-score to thoroughly assess their performance.

3.1 Model Performance Comparison

Table 1 summarizes the performance metrics of all four classification models.

Table 1: Performance Comparison of Classification Models

Model	Accuracy
Logistic Regression	0.617
Artificial Neural Network	0.702
Random Forest	0.740
Gradient Boosting	0.680
HistGradient Boosting	0.740
XGBoost	0.754
LightGBM	0.750
AdaBoost	0.170

3.2 Result Analysis

3.2 Result Analysis

- **XGBoost** achieved the highest performance with an accuracy of **75.4%**, making it the most effective model for this classification task. Its regularized gradient boosting approach allowed it to capture complex patterns without overfitting.
- **LightGBM** and **Random Forest** followed closely, both achieving around **74–75% accuracy**, demonstrating that ensemble methods are particularly well-suited for resume classification.
- **HistGradientBoosting** also matched Random Forest with **74% accuracy**, offering a faster alternative with comparable performance.
- **Gradient Boosting** delivered decent results with **68% accuracy**, but was slightly less effective than its histogram-based and optimized counterparts.
- **Artificial Neural Network (ANN)** achieved **70.2% accuracy**, showing potential for further improvement with deeper architectures or hyperparameter tuning.
- **Logistic Regression**, while being a simple linear model, reached **61.7% accuracy**. It served as a solid baseline but was outperformed by all ensemble and deep models.
- **AdaBoost** was the least effective with just **17% accuracy**, possibly due to its sensitivity to noisy features and limited adaptability in high-dimensional spaces.

Overall, the superior performance of **ensemble models** like XGBoost, LightGBM, and Random Forest can be attributed to their ability to model complex, nonlinear relationships in the resume text features. These methods are better at generalizing patterns necessary for accurate classification without overfitting.

3.3 Feature Importance Analysis

Based on our feature extraction process that combined TF-IDF vectorization (with 1200 features), structural features (resume length, word counts), and semantic embeddings from the 'all-MiniLM-L6-v2' Sentence Transformer model, we were able to create a robust representation of each resume. The high performance across models suggests that our feature engineering approach successfully captured the discriminative characteristics of different resume categories.

4 Summary

In this phase of the project, we focused on fair data representation through class balancing and feature extraction using TF-IDF. These steps lay the foundation for training robust classification models in future work. We ensured that the features capture both frequent terms and multi-word phrases using bigrams.

References

[1] Jain, A., & Sharma, R. (2020). *Resume Classification using Natural Language Processing Techniques*. Journal of Computer Science Applications.

[2] Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.

[3] Chen, T., & Guestrin, C. (2016). *XGBoost: A Scalable Tree Boosting System*. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[4] Reimers, N., & Gurevych, I. (2019). *Sentence-BERT: Sentence Embeddings using Siamese BERTNetworks*. arXiv preprint arXiv:1908.10084.

[5] Google Cloud. *Google Cloud AutoML*. Accessed 2025.

[6] Van der Maaten, L., & Hinton, G. (2008). *Visualizing Data using t-SNE*. Journal of Machine Learning Research, 9(Nov), 2579–2605.

[7] Wold, S., Esbensen, K., & Geladi, P. (1987). *Principal Component Analysis*. Chemometrics and Intelligent Laboratory Systems, 2(1-3), 37–52.

5 Contribution of each member

1. Anshuman Parida[1]: Implemented data filtering, combined structural and semantic features with the TF-IDF matrix, converted the scaled structural and semantic features to sparse matrices, and implemented tree-based methods like Random Forest and XGBoost.

2. Maliwal Raghav Kamleshkumar[2]: Contributed to TF-IDF modeling and clustering methods.Deploying the model for web demo.

3. Vasu Kodati[3]: Helped create dense embeddings using sentence transformers, created initial text processing pipeline and performed SVM and ANN modeling and analysis.

4. K Nithin Rohit[4]: Helped validate preprocessing and other clustering methods.

5. Soham Nandkumar Kulkarni[5]: Ran Logistic Regression And tried ways to improve performance by bettering Logistic Regression and other linear methods.Use PCA and TSNE for visualization and understanding .