

**G H Patel College of Engineering & Technology**

**(The Charutur Vidyamandal CVM University) V. V. Nagar**

**COMPUTER SCIENCE AND DESIGN**

**Project Report on**

***Resume Scanner***

**Submitted By**

**Niyati Lad**

**12202130501046**

**Prachi Surti**

**12202130501050**

**ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING (202100806)**

**A.Y. 2024-25 EVEN TERM**

## **TABLE OF CONTENTS**

1. **Objectives 3**
2. **Dataset Used 3**
3. **Model Chosen 4**
4. **Performance Metrics 5**
5. **Challenges & Learnings 6**
6. **Future Enhancement 7**
7. **References 7** 
   1. **OBJECTIVE**

The primary objective of this project is to build a machine learning-based application capable of scanning and analyzing resumes to determine the specific domain of Computer Science a candidate is most likely associated with. This system is designed to automate the classification of resumes by leveraging natural language processing to extract meaningful content and applying trained models to identify patterns that align with various technical domains. Ultimately, the goal is to enhance the efficiency of resume screening by providing accurate, data-driven predictions that assist in talent identification and domain categorization.

* 1. **DATASET USED**

The project utilizes a curated dataset named **"Updated Resume Dataset"**, which serves as the primary source for training and evaluating the resume classification model. This dataset is structured to reflect real-world resumes and is specifically designed for machine learning applications in the recruitment and HR domain.

The dataset comprises two main columns:

* Resume: Contains raw textual content extracted from resumes. This includes information such as education, experience, skills, projects, certifications, and personal summaries.
* Category: Represents the labeled domain or functional area of the resume within the field of Computer Science. Some of the domains included are:
  + Data Science
  + Web Development
  + Machine Learning
  + DevOps
  + Testing
  + HR
  + Operations
  + Networking
  + Mobile Development
  + Cloud Computing
  + Business Analyst
  + Python Developer
  + SAP Developer
  + Java Developer
  + Automation Testing
  + Digital Marketing
  + Others

With thousands of entries, this dataset captures a broad and realistic distribution of skills and job roles. Each entry is a unique resume and is already labeled with the appropriate category, making it suitable for supervised learning.

Before feeding into the machine learning pipeline, the dataset undergoes several preprocessing steps, including:

* Text cleaning: Removal of URLs, special characters, non-ASCII characters, and extra white spaces.
* Normalization: Converting text to lowercase for uniformity.
* Vectorization: Application of TF-IDF (Term Frequency-Inverse Document Frequency) to convert textual data into numerical feature vectors.

This dataset plays a crucial role in training the classification models by helping them learn domain-specific vocabulary, structure, and terminology, ultimately enabling accurate prediction of a resume's domain based on its content.

* 1. **MODEL CHOSEN**

### To accurately classify resumes into their respective domains within the field of Computer Science, a comparative analysis of multiple machine learning algorithms was conducted. The goal was to identify the most effective model in terms of **accuracy, processing efficiency**, and **generalization** for real-world deployment.

### Models Explored:

* **Support Vector Classifier (SVC)**
  + A powerful supervised learning model that constructs a hyperplane in a high-dimensional space to separate data into different classes. It is particularly effective for text classification tasks due to its ability to handle sparse and high-dimensional feature vectors.
* **K-Nearest Neighbors (KNN)**
  + A distance-based algorithm that classifies data points based on their proximity to neighbors.
* **Random Forest Classifier**
  + An ensemble learning method that constructs multiple decision trees and merges them for more accurate and stable predictions.

All models were trained using TF-IDF (Term Frequency–Inverse Document Frequency) transformed resume text, following extensive text cleaning and preprocessing. Evaluation metrics such as accuracy, confusion matrix, and classification report were used to assess their performance.

After comprehensive testing, the Support Vector Classifier (SVC) outperformed the other models in terms of precision and overall classification effectiveness. Its ability to handle high-dimensional feature spaces made it particularly suitable for text-based classification tasks like resume analysis.

To ensure model compatibility, Label Encoding was applied to convert job domain categories into numerical labels. The trained SVC model, along with the TF-IDF vectorizer and label encoder, was serialized using Python’s pickle module and seamlessly integrated into a Streamlit-based web application.

The final deployment allows users to upload resumes in PDF, DOCX, or TXT format and receive an instant prediction of the relevant job domain classification.

* 1. **PERFORMANCE METRICS**

To determine the most effective machine learning model for classifying resumes into various Computer Science domains, we evaluated three classification algorithms: RandomForestClassifier, Support Vector Classifier (SVC), and KNeighborsClassifier. Each model was trained on feature-engineered resume data and evaluated using accuracy, confusion matrix, and classification report metrics.

1. **RandomForestClassifier**

* Accuracy: 100%
* Confusion Matrix: Perfect diagonal alignment, indicating zero misclassifications.
* Classification Report:
  + Precision, Recall, F1-Score (per class): 1.00
  + Macro Average & Weighted Average: 1.00

Interpretation:  
The RandomForestClassifier demonstrates flawless performance on the test dataset, achieving perfect scores across all evaluation metrics. The completely diagonal confusion matrix confirms zero misclassifications, showcasing the model’s exceptional accuracy. Its consistent precision and recall across all 21 distinct domains indicate strong generalization capabilities, making it a highly robust and reliable candidate for production deployment.

1. **Support Vector Classifier (SVC)**

* Accuracy: 100%
* Confusion Matrix: Identical to RandomForest — all predictions were accurate with zero misclassifications.
* Classification Report:
  + Precision, Recall, F1-Score: 1.00 across all classes
  + Macro Avg & Weighted Avg: 1.00

Interpretation:  
The Support Vector Classifier achieved perfect performance, matching the RandomForest model with 100% accuracy and flawless classification metrics. This indicates that the SVC model effectively captured the decision boundaries in the dataset, demonstrating excellent generalization capability even in complex and potentially high-dimensional feature spaces.

1. **KNeighborsClassifier**

* Model Accuracy: 100%
* Confusion Matrix: The matrix is perfectly diagonal, indicating zero misclassifications across all 21 classes.
* Classification Report:
* Precision: 1.00 for all classes
* Recall: 1.00 for all classes
* F1-Score: 1.00 for all classes
* Macro Average: 1.00
* Weighted Average: 1.00

Interpretation:  
The RandomForestClassifier demonstrated flawless performance on the test dataset, accurately classifying all instances across the 21 distinct computer science domains. The perfect precision, recall, and F1-scores indicate exceptional model generalization, making it highly reliable and well-suited for production deployment.

* 1. **CHALLENGES & LEARNINGS**

**Challenges:**

1. **Dataset Imbalance & Diversity:**  
   Curating a dataset that equally represented all 21 domains was a major hurdle. Initial data was skewed toward popular fields like Web Development and Machine Learning, which led to biased model predictions. We addressed this through data augmentation and stratified sampling.
2. **Text Preprocessing Complexity:**  
   Resumes vary widely in format, structure, and terminology. Extracting relevant keywords and standardizing the text while retaining domain-specific context required iterative experimentation with NLP techniques such as TF-IDF vectorization and custom stopword filtering.
3. **Multi-Class Classification with High Precision:**  
   Achieving high accuracy across **21 distinct classes**—many with overlapping terminologies—was particularly challenging. It required fine-tuning of the RandomForestClassifier and evaluating multiple models before arriving at the ideal one.
4. **Evaluation at Scale:**  
   With a large test set and high dimensional feature space, ensuring the reliability of metrics like precision and recall for each class demanded careful performance validation using cross-validation and manual inspection of misclassifications (though in the final model, there were none).

**Learnings:**

1. **Model Selection Matters:**  
   Experimenting with several algorithms (Logistic Regression, SVM, Random Forest) deepened our understanding of how different models behave with sparse text data. The RandomForestClassifier stood out for its robustness and interpretability.
2. **Importance of Clean, Labeled Data:**  
   Building a high-quality labeled dataset from resumes emphasized how crucial clean, domain-specific input is for model success—garbage in truly results in garbage out.
3. **Effective Feature Engineering:**  
   We learned that well-engineered features (like domain keyword frequency and resume length normalization) significantly improve classifier performance compared to relying on raw text alone.
4. **Model Evaluation Beyond Accuracy:**  
   We understood the importance of **precision, recall, F1-score**, and the **confusion matrix**—especially in multi-class scenarios. A single accuracy metric isn't enough when dealing with diverse outputs.
5. **Real-world Readiness:**  
   This project provided practical insights into building a deployable ML pipeline—from preprocessing and training to evaluation and interpretation—bridging the gap between academic knowledge and production-level implementation.

**6. Future Enhancement**

* After analysis, suggest improvements and allow users to generate a stronger resume based on missing skills/sections.
* Save history of scanned resumes per user.
* Extract structured data: name, email, phone, skills, education, experience, etc.
* Rank resumes based on a score using cosine similarity or machine learning.

**7. References**

* Resume Dataset - Kaggle

<https://www.kaggle.com/datasets/gauravduttakiit/resume-dataset>

* Automated Resume Classification Using Machine Learning

<https://ieeexplore.ieee.org/document/9215524>

* Deep Learning for Resume Classification

<https://arxiv.org/abs/1907.05585>