RESUME CLASSIFICATION USING MACHINE LEARNING ALGORITHMS

Major Project Report submitted in partial fulfillment of the requirements for the award of the Degree of B.E. in Computer Science and Engineering

By

160620733122 Anna Kavya
160620733129 Dri Suvarna
160620733144 Pattam Vedavathi

Under the Guidance of

Dr. C. Kishor kumar
Associate Professor,
Department of Computer Science & Engineering

Department of Computer Science and Engineering Stanley College of Engineering & Technology for Women

(Autonomous)

Chapel Road, Abids, Hyderabad - 500001

(Affiliated to Osmania University, Hyderabad, Approved by AICTE, Accredited by NBA & NAAC with A Grade)

2024

RESUME CLASSIFICATION USING MACHINE LEARNING ALGORITHMS

Major Project Report submitted in partial fulfillment of the requirements for the award of the

Degree of B.E in Computer Science and Engineering

By

160620733122 Anna Kavya
160620733129 Erri Suvarna
160620733144 Patnam Vedavathi

Under the Guidance of

Dr. C. Kishor kumar
Associate Professor,
Department of Computer Science & Engineering

Department of Computer Science and Engineering Stanley College of Engineering & Technology for Women (Autonomous)

Chapel Road, Abids, Hyderabad – 500001
(Affiliated to Osmania University, Hyderabad, Approved by AICTE, Accredited by NBA & NAAC with A Grade)

2024

tanley College of Engineering & Technology for Women

(Autonomous)

Chapel Road, Abids, Hyderabad – 500001

(Affiliated to Osmania University, Hyderabad, Approved by AICTE, Accredited by NBA & NAAC with A Grade)

CERTIFICATE

This is to certify that major project report entitled **Resume Classification Using Machine Learning Algorithms**

being submitted by

160620733122	Anna Kavya
160620733129	Erri Suvarna
160620733144	Patnam Vedavathi

in partial fulfillment for the award of the Degree of Bachelor of Engineering in Computer Science & Engineering to the Osmania University, Hyderabad is a record of bonafide work carried out under my guidance and supervision. The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree or Diploma.

Guide

Dr. C. Kishor kumar

Associate professor, Dept of CSE

Head of the Department

Dr Y V S S Pragathi

Prof & HoD, Dept of CSE

Project Coordinator

Srilatha

Assistant professor, Dept of CSE

External Examiner

DECLARATION

We hereby declare that major project work entitled **Resume Classification Using Machine Learning Algorithms** submitted to the Osmania University, Hyderabad, is a record of original work done by us. This project work is submitted in partial fulfilment of the requirements for the award of the degree of the B.E in Computer Science and Engineering.

160620733122 Anna Kavya
160620733129 Erri Suvarna
160620733144 Patnam Vedavathi

ACKNOWLEDGEMENT

Firstly, we are grateful to The Almighty God for establishing us to complete this Major Project. We pay our respects and love to our parents and all our family members and friends for their love and encouragement throughout our career.

We wish to express our sincere thanks to Sri. Kodali Krishna Rao, Correspondent and Secretary Stanley College of Engineering & Technology for Women, for providing us with all the necessary facilities.

We place on record, our sincere gratitude to Prof. Satya Prasad Lanka, Principal, for his constant encouragement. We deeply express our sincere thanks to our Head of the Department, Prof Y V S S Pragathi, for encouraging and allowing us to present the Major Project on the topic **Resume Classification Using Machine Learning Algorithms** at our department premises for the partial fulfillment of the requirements leading to the award of the B.E. degree.

It is our privilege to express sincere regards to our project guide Dr.C. Kishor kumar, for the valuable inputs, able guidance, encouragement, whole-hearted co-operation and constructive criticism throughout the duration of our project.

We take this opportunity to thank all our faculty, who have directly or indirectly helped our project. Last but not least, we express our thanks to our friends for their co- operation and support.

ABSTRACT

In today's competitive job market, the ability to efficiently and accurately classify resumes is crucial for both job seekers and recruiters. The emergence of machine learning (ML) algorithms has revolutionized the process of resume classification, enabling automated analysis and decision-making.

This project focuses on implementing and comparing various ML algorithms for the classification of resumes. The algorithms investigated include Logistic Regression, Decision Trees, k-Nearest Neighbors (kNN), Support Vector Machines (SVM), Naive Bayes, and AdaBoost. Each algorithm is evaluated based on its performance metrics such as accuracy, precision, recall, and F1-score.

The dataset used for training and testing consists of labeled resumes, where each resume is associated with a specific job category or class. The preprocessing phase involves text cleaning, feature extraction, and vectorization to transform the textual data into a format suitable for ML algorithms.

Experimental results demonstrate the effectiveness of each algorithm in accurately categorizing resumes into their respective job categories. The findings of this study provide valuable insights into the performance of different ML techniques for resume classification tasks, thereby aiding in the development of robust and efficient automated resume screening systems.

Keywords: Keywords:

Resume classification, Machine learning algorithms, Logistic Regression, Decision Trees, k-Nearest Neighbors (kNN), Support Vector Machines (SVM), Naive Bayes, AdaBoost, Text preprocessing,Feature extraction, Vectorization, Performance evaluation, Accuracy, Precisio, Recall, F1-score, Job categorization, Automated resume screening, Data analysis, Comparative study

ontents

List of Figures

CHAPTER 1

INTRODUCTION

1.1 About Project

The resume classification project aims to automate the categorization of resumes into predefined job categories using machine learning algorithms. This addresses the challenges faced by recruiters in manually screening large volumes of resumes, improving efficiency and reducing human bias. By leveraging various machine learning techniques, the project seeks to build a robust classification system capable of handling diverse resume formats and content, streamlining the recruitment process and enabling faster, more accurate candidate shortlisting.

1.2 Objectives

The main objectives of this project are:

1. To implement and compare the performance of different machine learning algorithms for resume classification.
2. To identify the most effective algorithms for categorizing resumes based on predefined job categories.
3. To evaluate the accuracy, precision, recall, and F1-score of each algorithm.
4. To develop a robust system that can handle a diverse set of resumes and accurately classify them.

1.3 Scope

The scope of this resume classification project encompasses several key aspects essential for developing a functional and effective system. Initially, the project involves gathering a diverse dataset of resumes labeled with specific job categories. This dataset undergoes extensive preprocessing, including text normalization, tokenization, stop-word removal, and feature extraction using techniques like TF-IDF. The next phase focuses on implementing multiple machine learning algorithms, such as logistic regression, AdaBoost, SVM, Naive Bayes, random forest, decision tree, and KNN, and developing the necessary scripts and models to handle the classification tasks.

Following implementation, the models are trained on the preprocessed dataset and evaluated using metrics like accuracy, precision, recall, and F1-score. Cross-validation ensures the robustness and reliability of the results. To enhance performance, techniques such as grid search and randomized search are employed for hyperparameter tuning, ensuring the models are fine-tuned for optimal performance. A comparative analysis is then conducted to identify the most effective algorithms, considering accuracy, computational efficiency, and scalability.

The project also includes developing a prototype system that integrates the best-performing model(s) into a user-friendly application capable of handling real-world resume data in various formats and structures. Comprehensive documentation of the methodology, experiments, results, and findings is provided, along with a detailed report covering data preprocessing, model implementations, evaluations, and conclusions.

Additionally, the project identifies potential areas for future work, such as incorporating deep learning approaches, expanding the system to handle multilingual resumes, and suggesting improvements for managing larger datasets and real-time classification. This scope ensures that the project thoroughly addresses all necessary steps to develop a robust and efficient resume classification system using machine learning.

1.4 Advantages

1. Efficiency Improvement: The automated resume classification system significantly speeds up the resume screening process, allowing recruiters to process a large volume of resumes quickly and efficiently.
2. Reduction of Human Bias: By relying on machine learning algorithms, the system minimizes human bias in the recruitment process, leading to fairer and more objective candidate selection.
3. Consistency and Accuracy: Machine learning models, once trained, consistently apply the same criteria to all resumes, ensuring uniformity and accuracy in classification.
4. Scalability: The system can handle large datasets and can be scaled up to accommodate increasing numbers of resumes as needed.

1.5 Disadvantages

1. Data Dependency: The performance of the machine learning models heavily depends on the quality and quantity of the training data. Poorly labeled or insufficient data can lead to inaccurate classifications.
2. Maintenance and Updates: The system requires regular updates and maintenance to ensure continued accuracy and relevance, especially as job roles and market requirements evolve.
3. Handling Diverse Formats: Resumes come in various formats and structures, and the system must be capable of accurately processing this diversity. This can be challenging and might require extensive preprocessing and feature extraction techniques.

1.6 Applications

When a company receives thousands of resumes for the recruitment process, it is very difficult for the concerned authorities to go through each resumes and it is time consuming process. It is time saving

because machine itself classifies the resumes according to criteria. Classification will be done using Machine Learning on the basis of criteria, job profile, experience & specialization.

a) Attributes:

The Attributes used in the dataset are:

- Resume and Text.

b) Dataset Description:

- The dataset consists of resumes and related textual information for 12 individuals with backgrounds in software development. Each individual's profile provides details such as technical skills, work experience, education, projects undertaken, and personal information. The resumes are rich in information, covering various aspects of the candidates' professional journeys, skills, achievements, and interests.

1. Attributes in the dataset include: Resume, Text. Comprehensive information about the individual's professional experience, skills, and other relevant details.
2. Professional Summary: A summary highlighting key aspects of the individual's professional background and skills.
3. Technical Skills: Enumerating the technical skills possessed by each candidate.
4. Education Details: Providing information about the candidate's educational background.
5. Work Experience: Detailing the candidate's work history, roles, and responsibilities.
6. Project Details: Describing the projects the candidate has worked on, including project names, technologies used, roles, and responsibilities.
7. Personal Details: Presenting personal information such as nationality, gender, marital status, and languages known.
8. Declaration: A statement of truthfulness and accuracy of the information provided.

c) Dataset- Source:

- The Dataset has been taken from an open-source website called "Github".

Link to the dataset:

https://github.com/shanuhalli/Project-Resume-Classification/blob/main/Dataset/Raw_Resume.csv

1.7 Hardware and Software Requirements

Hardware Requirements

1. Processor: A multi-core processor (e.g., Intel Core i5/i7 or AMD Ryzen) is recommended to handle the computational load of training machine learning models efficiently. For large datasets or more complex models, a more powerful processor (e.g., Intel Xeon or AMD Threadripper) may be necessary.

2. Memory (RAM): A minimum of 16 GB of RAM is recommended to ensure smooth processing of data and model training. For handling very large datasets or training complex models, 32 GB or more might be required.

3. Storage: At least 500 GB of storage space is recommended. Solid State Drives (SSDs) are preferred over Hard Disk Drives (HDDs) for faster data access and better overall performance. If working with very large datasets, additional storage may be needed.

Software Requirements

1. Programming Language: Python is the primary programming language for this project due to its extensive libraries and community support for machine learning and data science.

2. Libraries and Frameworks:

- Scikit-learn: For implementing machine learning algorithms like logistic regression, SVM, random forest, etc.

Pandas: For data manipulation and analysis.

Numpy: For numerical computations and handling arrays.

NLTK (Natural Language Toolkit): For text preprocessing tasks like tokenization and stop-word removal.

TF-IDF Vectorizer: Available through Scikit-learn for feature extraction from text.

XGBoost: For implementing the AdaBoost algorithm, if required.

Matplotlib/Seaborn: For data visualization and plotting results.

3.Development Environment:

Jupyter Notebook: For interactive development and experimentation with code.

Anaconda: A distribution of Python and R for scientific computing, which simplifies package management and deployment

CHAPTER 2
LITERATURE SURVEY

Literature Survey Finding the need for resume classification:

As humans, everyone is bound to make mistakes. Storing and sharing physical copies safely is very inconvenient and inefficient. For this purpose of overcoming the drawbacks, need for resume classification significantly increases for which Artificial intelligence tools and Machine Learning algorithms [1] have been used widely. Categorization of similar data together: With the advancement of computer and information technologies, a large number of research papers have been published both online and offline, and as new study topics continue to emerge, users are having a difficult time discovering and classifying their relevant research articles. A classification method that can cluster research articles into

a meaningful class in which publications are extremely likely to have similar subjects is needed to overcome the restrictions. The suggested method uses K-means Clustering [2] and the Latent Dirichlet allocation (LDA) scheme [2] to extract representative keywords from the abstracts of each publication and subjects. Importance of Automation: the issue regarding lack of automated systems for medical institutions and hospitals have caused a splurge for development of automation for hospital system [3]. Natural Language Processing (NLP)[11] benefits society as a whole as text-based information handling is very difficult manually. Preprocessing of data using TF-IDF Vectorization: A well-developed categorization system that can group research papers into relevant classes based on their subjects [4] helps in finding out relevant data in the most time efficient manner. The suggested approach retrieves representative keywords from each paper's and topic's abstract. Then, using the Term frequency-inverse document frequency (TF-IDF) values [4] of each article, the K-means clustering method [4] is used to categorize the entire set of papers into research papers with comparable themes. Modelling of semi-structured documents to fetch job postings from resume: Matching semi-structured resumes with positions in a big size real-world collection is a tough challenge. Experiments reveal that the SRM technique [5] yielded encouraging results and outperformed traditional unstructured relevance models. Furthermore, we compared the suggested system's efficiency and efficacy to those of state-of-the-art online recruiting methods. A system that utilizes machine learning to match job postings and resumes for huge data sets; in our work, it is less difficult than previous papers; and using text mining, the extracted data from the resume is matched with the keyword recorded in the database and categorized for each job category.

2.1 Existing System

The existing systems for resume classification primarily rely on keyword-based and rule-based approaches, which are limited by their inability to understand context and require extensive manual setup and maintenance. Traditional machine learning methods, such as Naive Bayes, Support Vector Machines (SVM), and Random Forests, have been used to improve accuracy, but they still struggle with capturing the complexities of resume data. Machine learning ecurrent Neural Networks (RNNs), Long Short-Term Memory Networks (LS

shown significant improvements by understanding context and semantics more effectively. The proposed system aims to integrate these advancements by incorporating comprehensive data preprocessing techniques, such as text normalization and feature extraction using TF-IDF. It will leverage both traditional machine learning algorithms (e.g., Naive Bayes, SVM, Random Forests) and modern machine learning model to enhance classification accuracy. The evaluation will include metrics like accuracy, precision, recall, and F1-score, with model tuning and performance comparison to identify the most effective approach. This integration aims to overcome the limitations of existing systems and provide a robust solution for automated resume classification.

2.2 Proposed System

The proposed system for resume classification aims to leverage advancements in both traditional machine learning and modern deep learning techniques to enhance accuracy and efficiency. The system will begin with comprehensive data preprocessing, including text normalization (removing punctuation, lowercasing, stemming/lemmatization) and feature extraction using methods such as TF-IDF. For the classification task, the system will employ a variety of algorithms: Naive Bayes for baseline performance comparison, Support Vector Machines (SVM) utilizing TF-IDF features, and Random Forests to capture non-linear relationships. Additionally, machine learning models such as used to capture local features and hierarchical patterns, will capture sequential information and context. Transformer models, leveraging pre-trained contextual embeddings, will be utilized for their superior performance in understanding semantics and context. The system will be evaluated using metrics like accuracy, precision, recall, and F1-score, with model tuning and performance comparison to identify the most effective approach. By integrating there diverse techniques, the proposed system seeks to provide a robust, scalable solution for automated resume classification.

CHAPTER 3

PROPOSED ARCHITECTURE

1. Proposed Architecture

This section will describe the methodology and concepts that facilitate the building of classification model capable for resume classification and displaying the output with a suitable job profile for the candidate. The system works in the following phases as given below.

3.1 Data Gathering

Data Gathering includes collection of datasets from various websites like kaggle.com, glassdoor.com and indeed.com. The datasets are not classified and are unstructured datasets in which the data will be cleaned, classified, and stored in "25_cleaned_job_descriptions.csv" including some parts from Kaggle and some from glassdoor and indeed.com. 70% of the data is being used for training data and the remaining 30% will be used for test data.

3.2 Data Cleanup

The dataset containing a huge number of records is still very rough and unclassified. Data cleaning will be done by removing any blank spaces from the data, then changing all the text to lowercase to avoid confusion and removing stop words from the data. Stop words are those words which don't play an important role in sentence formation, such as "are", "we", "is", etc. Cleaned data is stored in a separate dataset containing 10,000 entries with two main classes of "query" and "description".

3.3 Tokenization

In this step, each entry in the corpus i.e., each entry in the document will be broken down into a set of words. To begin the tokenization process, we look for concepts or words that make up a character sequence. This is significant because we will be able to deduce meaning from the original text sequence using these terms. Tokenization is the process of separating large chunks of text into smaller pieces known as tokens. This is accomplished by deleting or isolating characters like as whitespace and punctuation. Tokens are phrases that are divided into individual words after being tokenized out of paragraphs. We may obtain information such as the number of words in a text, the frequency of a specific term in the text, and much more by doing Tokenization. Tokenization can be done in a variety of methods, such utilising the Natural Language Toolkit [NLTK], the spaCy library, and so on. Tokenization is a required step for subsequent text processing such as stop word removal, stemming, and lemmatization.

3.4 Stemming and Lemmatization

It is common to see a single English word employed in a variety of different ways in different phrases based on its grammatical rules. "Describe", "describing" and "described", for example, are all various

tenses of the same verb. This condition necessitates the reduction of all changed or derived versions of a word to its primary stem or base, so that these derivationally related terms with comparable meanings are not deemed distinct from one another. Both stemming and lemmatization have the same goal but take different approaches to achieve it. "**Stemming** is the mechanism of reducing inflected or derived words to their word root, or stem. It is a crude heuristic process that involves chopping off the ends of words to achieve this objective, and often includes the removal of **derivational affixes** [7]" These are rule-based algorithms that analyse a certain word under a variety of scenarios and then decide how to shorten it based on a list of recognized suffixes. It is worth noting that the root generated after stemming may not be the same as the word's morphological root. Stemming is prone to under and over-stemming. Porter-Stemmer, Snowball stemmer, and Lancaster stemmer are some common stemuning algorithms.

Lemmatization, on the other hand, is the process of accurately reducing words to root words using a language dictionary. Lemmatization, as opposed to Stemming, which merely chops out tokens. by basic pattern matching, is a more sophisticated technique that employs language vocabulary and morphological study of words to provide linguistically proper lemmas. This This implies that lemmatization makes use of context information and may thus distinguish between betw words with various meanings based on parts of speech. For the English language, our system uses the NLTK python package's WordNet Lemmatizer (based on the WordNew Database).

The concept of stemming and lemmatization can be understood by a simple example. The word "loving" will turn into "lov" after stemming which has no meaning while if lemmatized, "loving" will turn into "love" which now has a proper meaning and use case

3.5 Parts of Speech (POS) Tagging

It is the process of associating grammatical information with a word depending on its context and relationship to other words in the sentence [8]. According to its usage in the phrase, the part-of-speech tag identifies whether the word is a noun, pronoun, verb, adjective, or other These tags must be assigned in order to grasp the right meaning of a phrase and to create knowledge bases for character recognition. This procedure is not as straightforward as mapping a word to its appropriate part of speech tags. This is became a word may have a distinct part of speech depending on the context in which it is spoken. For example, "writing" is a Verb in the statement "I am writing an essay," yet "building" is a Noun in the line "I stay in the tallest building in the entire town." It is a supervised learning approach that analyses information such as the preceding word, following word, initial letter capitalised or not, and so on to label the words after tokenization. It is also known as grammatical tagging.

3.6 TF-IDF Vectorization

TF-IDF stands for Term Frequency Inverse Document Frequency and it tells us how important a word is from the set of words in the dataset and assigns a tfidf value indicating the importance of the word as

per frequency The formula for term frequency is the division of number of occurrences of a word in a sentence by the total number of words in the sentence (1). Inverse document frequency is calculated by the log of total sentences in the document divided by the sentences that actually contain the word (2). The multiplication of TF formula and IDF formula will give us a value in the form of a vector with a gruph of importance on one hand and set of words on the other hand (3). Fig. 1 1 depicts a graph that shows TF-IDF vectorization which explains that the word with higher importance is placed according to ascending order of its importance and hence an inverse growth curve.

$$TF = \frac{No.of\ occurence\ in\ sentence}{Total\ no.of\ words\ in\ sentence}$$

$$IDF = \log\left[\frac{total\ sentences\ in\ document}{sentences\ which\ actually\ contain\ the\ word}\right]$$

$$TF\text{-}IDF\ Vectorization = TF \times IDF$$

3.7 Applying Classification Algorithm To Dataset

The following classification algorithms have been used

for classification and model training. Naive bayes is a classification algorithm that works on probabilistic output us in whether the event is going to occur of not t provide vided with a set of conditions. On the NB Classifier, the training data is fitted Then in the validation dataset labels are being predicted. To get accuracy, use accuracy score function. This classification model gave an accuracy of around 45% and did d not get the expected results

Support Vector Machine (SVM) algorithm classifies data by drawing a hyperplane between two or more items. The Hyperplane which best classifies the items is considered as ideal output. The working flow of SVM is similar to that of Naive Bayes. This classification model gave an accuracy of 617% which proved to be better than mtive bayes model.

Random Forest is a classification algorithm that works on the principle of decision trees. It takes in input. of many decision trees and gives the best majority output from all the inputted decision trees. On the RF Classifier the training data is fitted. Then, in the validation dataset labels are being predicted To get accuracy, use accuracy score function. Random forest model gave an accuracy of 70% and gave the most correct predictions as per the comparison

The comparison between the naïve bayes model, the SVM model and the random forest model with their accuracy, precision, recall and FI score is given in TABLE 1

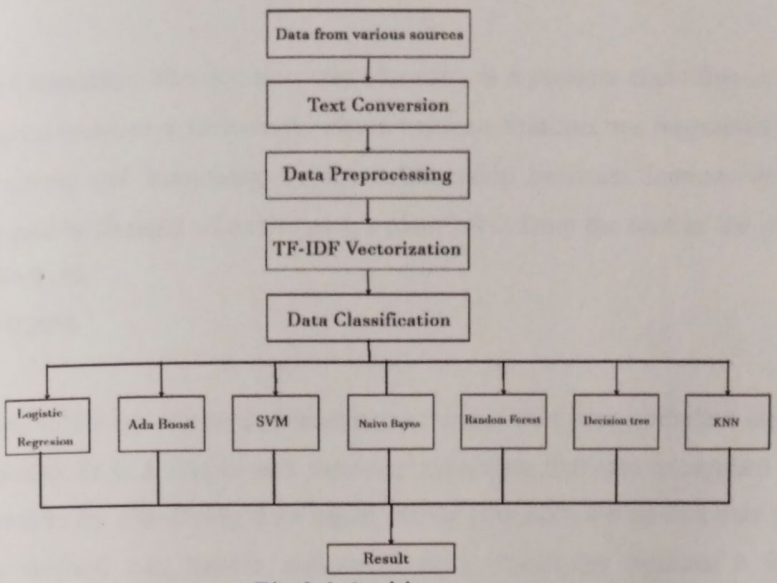

Fig 3.1 Architecture

2. Methodology Flowchart

The methodology is important to understand, and hence it becomes necessary to make a flowchart for easy understanding of the flow of system. Fig. 2. Shows the flowchart of the system

Finst the datuset needs to be scraped which can var websites like indeed.com, be done wing glassdoor.com and kaggle.com. Once the dataset has freen scrapped, preprocessing of data is to be done by doing proper stemuning and lemmatitation, removing stop words and filler words store tha televant wonda separate colums for further process. After the preprocessing o of data is done, mi data hu muk king sure that have the words which have occurred the highest the relevant amount of times need to be put in a term-frequency document using TF- IDF Vectorization. Once the previous steps are done, the cleaned data is now ready to test and form classification models using machine learning algorithens, Analysing the model by their accuracy and the forming confusion matrix for the same is required for better understanding of the results.

1) Random Forest Classifier: Random Forest is a supervised learning technique used for both classification and regression tasks. It is composed of multiple decision trees, forming a forest. Due to the involvement of several decision trees, Random Forest is considered to be more accurate and resilient. This technique does not suffer from overfitting issues and can handle missing values in the data. However, due to the presence of multiple trees, Random Forest generates predictions at a slower pace compared to a single decision tree. Additionally, interpreting the results of Random Forest is more complex than a decision tree due to the involvement of multiple decision trees.

- *Accuracy*: 98.70%
- *F1-Score*: 98.56%

2) Decision Tree Classifier: The decision tree classifier is a popular algorithm in supervised machine learning. Its design resembles a flowchart, where tests on features are represented by internal nodes, class labels by leaves, and branching by the relationship between features and class labels. The classification rule can be thought of as the path a plant takes from the root to the leaf in the tree.

- *Accuracy*: 94.81%
- *F1-Score*: 94.29%

3) KNN Classifier: KNN is a supervised learning technique that uses a labelled input dataset to predict the output data points. It is a simple and versatile algorithm that can be applied to a wide range of problems. KNN works by classifying data based on the similarity between a data point and its nearest neighbours. This method can handle real-world data effectively because it does not make any assumptions about the input data set, making it a more attractive option. KNN is a "lazy" algorithm since it keeps the training data set rather than learning from it. It can be used to solve both classification and regression problems.

- *Accuracy*: 94.81%
- *F1-Score*: 94.89%

4) SVM Classifier: The Support Vector Machine (SVM) technique seeks to locate an effective hyperplane in an N-dimensional space for separating the data points. The size of the hyperplane is determined by the number of features in the data set. If there are only two input features, the hyperplane takes the form of a line. However, if there are three input features, the hyperplane becomes a 2-D plane.

- *Accuracy*: 97.40%
- *F1-Score*: 97.08%

5) KNN Classifier: KNN is a supervised learning technique that uses a labelled input dataset to predict the output data points. It is a simple and versatile algorithm that can be applied to a wide range of problems. KNN works by classifying data based on the similarity between a data point and its nearest neighbours. This method can handle real-world data effectively because it does not make any assumptions about the input data set, making it a more attractive option. KNN is a "lazy" algorithm since it keeps the training data set rather than learning from it. It can be used to solve both classification and regression problems.

- *Accuracy*: 94.81%
- *F1-Score*: 94.89%

6) **Adaboost Classifier:** Adaboost, short for Adaptive Boosting, is a machine learning meta-algorithm that is used to boost the performance of weak learners. It works by combining multiple weak learners (often decision trees) to create a strong classifier. Adaboost assigns weights to each training example and focuses more on the difficult-to-classify instances in subsequent iterations. However, it is sensitive to noisy data and outliers.

- *Accuracy*: 84.42%
- *F1-Score*: 77.78%

CHAPTER 4

IMPLEMENTATION

4.1 Algorithm:

Random Forest Algorithm

1. Data Preprocessing:

- Normalize the text data by removing punctuation, lowercasing, and performing stemming or lemmatization to standardize the text format.

- Extract relevant features from the resumes using techniques like TF-IDF to represent the text data numerically.

2. Splitting the Dataset:

- Divide the preprocessed dataset into training and testing sets to evaluate the model's performance accurately.

3. Random Forest Model Initialization:

- Initialize a Random Forest classifier, specifying parameters such as the number of trees in the forest and maximum depth of each tree.

4. Training the Model:

- Fit the Random Forest classifier to the training data, allowing the model to learn patterns and relationships between the features and target labels (resume categories).

5. Model Evaluation:

- Evaluate the trained Random Forest model's performance on the testing dataset using evaluation metrics such as accuracy, precision, recall, and F1-score to assess its effectiveness in classifying resumes.

6. Hyperparameter Tuning:

- Fine-tune the hyperparameters of the Random Forest algorithm, such as the number of trees and maximum depth, using techniques like grid search or random search to optimize model performance.

7. Feature Importance Analysis:

- Analyze the importance of features (words or phrases) in the Random Forest model to understand which aspects of the resumes contribute most to the classification decisions.

8. Deployment:

- Deploy the trained Random Forest model into the production environment for real-time resume classification, ensuring seamless integration with existing systems or applications.

9. Monitoring and Maintenance:

Continuously monitor the performance of the deployed model and update it periodically with new data to adapt to changes in resume patterns and requirements.

By following these implementation steps, the Random Forest algorithm can be effectively utilized as part of the proposed resume classification system to accurately categorize resumes into predefined categories.

4.2 Code Implementation

1. Logistic regression:

```
kfold=KFold(n_splits=10)
logistic_model=LogisticRegression(penalty="l2",max_iter=1000)
logistic_model_result=cross_val_score(logistic_model,x,y,cv=kfold)
logistic_accuacy_mean=logistic_model_result.mean()
print(f"The mean of logistic regession is, {logistic_accuacy_mean}")
logistic_prediction=cross_val_predict(logistic_model,x,y,cv=kfold)
print("The confusion matrix for logistic regression:\n",confusion_matrix(y,logistic_prediction))
logistic_accuracy=accuracy_score(y,logistic_prediction)
print("The accuracy of the logistic regression",logistic_accuracy)
print("The Classification report of logistic regression\n",classification_report(y,logistic_prediction))
logistic_f1=f1_score(y,logistic_prediction,average="macro")
print("The F1 score for logitic regression: ",logistic_f1)
```

2. Naive Bayes:

```
nb_model=MultinomialNB()
nb_model_result=cross_val_score(nb_model, x, y, cv=kfold)
nb_accuracy_mean=nb_model_result.mean()
print("The mean of Multinomial NB is ",nb_accuracy_mean)
nb_prediction=cross_val_predict(nb_model,x,y,cv=kfold)
print("The confusion matrix of Multinomial NB is \n",confusion_matrix(y,nb_prediction))
NB_accuracy=accuracy_score(y,nb_prediction)
print("The accuracy of the Multinomial NB",NB_accuracy)
print("The Classification report of Multinomial NB\n",classification_report(y,nb_prediction))
NB_f1=f1_score(y,nb_prediction,average="macro")
print("The F1 score for Multinomial NB: ",NB_f1)
```

3. Decision Tree

```
DT_model=DecisionTreeClassifier(criterion="gini",max_depth=3)
DT_model_result=cross_val_score(DT_model,  x, y, cv=kfold)
DT_accuracy_mean=DT_model_result.mean()
print("The mean of Decision Tree is ",DT_accuracy_mean)
DT_prediction=cross_val_predict(DT_model,x,y,cv=kfold)
print("The confusion matrix of Decision Tree is \n",confusion_matrix(y,DT_prediction))
DT_accuracy=accuracy_score(y,DT_prediction)
print("The accuracy of the Decision Tree",DT_accuracy)
print("The Classification report of Decision Tree\n",classification_report(y,DT_prediction))
DT_f1=f1_score(y,DT_prediction,average="macro")
print("The F1 score for Decision Tree: ",DT_f1)
```

4.Random Forest

```
RT_model=RandomForestClassifier(n_estimators=200,criterion="gini",max_depth=5)
RT_model_result=cross_val_score(RT_model,  x, y, cv=kfold)
RT_accuracy_mean=DT_model_result.mean()
print("The mean of Random Forest is ",RT_accuracy_mean)
RT_prediction=cross_val_predict(RT_model,x,y,cv=kfold)
print("The confusion matrix of Random Forest is \n",confusion_matrix(y,RT_prediction))
RT_accuracy=accuracy_score(y,RT_prediction)
print("The accuracy of the Random Forest",RT_accuracy)
print("The Classification report of Random Forest\n",classification_report(y,RT_prediction))
RT_f1=f1_score(y,RT_prediction,average="macro")
print("The F1 score for DRandom Forest: ",RT_f1)
```

5. SVM

```
clf = SVC()
parameters = {'kernel':['rbf','poly', 'linear', 'sigmoid'],
              'gamma':[100, 50, 5, 0.5, 0.1, 0.01, 0.0001, 'auto'],
              'C':[50, 15, 10, 6, 5, 0.1, 0.001, 0.0001]}
gsv = GridSearchCV(clf, param_grid = parameters, cv = 10)
gsv.fit(x, y)
```

```
print(gsv.best_params_)
print(gsv.best_score_)
svm_model=SVC(C= 50, gamma= 100, kernel= 'poly')
svm_model_result=cross_val_score(svm_model, x, y, cv=kfold)
svm_accuracy_mean=svm_model_result.mean()
print("The mean of support vector machine is ",svm_accuracy_mean)
svm_prediction=cross_val_predict(svm_model,x,y,cv=kfold)
print("The confusion matrix of support vector machine is \n",confusion_matrix(y,svm_prediction))
svm_accuracy=accuracy_score(y,svm_prediction)
print("The accuracy of the support vector machine",svm_accuracy)
print("The Classification report of support vector machine\n",classification_report(y,svm_prediction))
svm_f1=f1_score(y,svm_prediction,average="macro")
print("The F1 score for support vector machine: ",svm_f1)
```

6. KNN

```
knn_model=KNeighborsClassifier(n_neighbors=3)
knn_model_result=cross_val_score(knn_model, x, y, cv=kfold)
knn_accuracy_mean=knn_model_result.mean()
print("The mean of KNN is ",knn_accuracy_mean)
knn_prediction=cross_val_predict(knn_model,x,y,cv=kfold)
print("The confusion matrix of KNN is \n",confusion_matrix(y,knn_prediction))
knn_accuracy=accuracy_score(y,knn_prediction)
print("The accuracy of the KNN",knn_accuracy)
print("The Classification report of KNN\n",classification_report(y,knn_prediction))
knn_f1=f1_score(y,knn_prediction,average="macro")
print("The F1 score for KNN: ",knn_f1)
```

7. Adaboost

```
AB_model=AdaBoostClassifier(n_estimators=100,learning_rate=0.9,random_state=7)
AB_model_result=cross_val_score(AB_model, x, y, cv=kfold)
AB_accuracy_mean=GB_model_result.mean()
print("The mean of AdaBoost is ",AB_accuracy_mean)
AB_prediction=cross_val_predict(AB_model,x,y,cv=kfold)
```

```
AB_accuracy=accuracy_score(y,AB_prediction)
print("The confusion matrix of MLP is \n",confusion_matrix(y,AB_prediction))
print("The accuracy of the AdaBoost",AB_accuracy)
print("The Classification report of AdaBoost\n",classification_report(y,AB_prediction))
AB_f1=f1_score(y,AB_prediction,average="macro")
print("The F1 score for AdaBoost: ",AB_f1)
X, y = x, y
y_binary = label_binarize(y, classes=np.unique(y))
# Create figure for ROC curves
plt.figure(figsize=(10, 5))
# Loop through classifiers and plot ROC curves
classifiers = [(RT_model, 'Random Forest'),
               (logistic_model, 'Logistic Regression'),
               (DT_model, 'Decision Tree'),
               (knn_model, 'K-Nearest Neighbors'),
               (clf, 'Multilayer Perceptron'),
               (nb_model,"Multinomial"),
               (svm_model,"SVM"),
               (AB_model,"AdaBoost"),

for classifier, label in classifiers:
    if hasattr(classifier, 'predict_proba'):
        classifier.fit(X, y)
        y_probs = classifier.predict_proba(X)
    else:
        ovr_classifier = OneVsRestClassifier(classifier)
        ovr_classifier.fit(X, y)
        y_probs = ovr_classifier.decision_function(X)

    if y_probs.ndim > 1 and y_probs.shape[1] > 2:
        # For multiclass classification, y_probs should have shape (n_samples, n_classes)
```

```
AB_accuracy=accuracy_score(y,AB_prediction)
print("The confusion matrix of MLP is \n",confusion_matrix(y,AB_prediction))
print("The accuracy of the AdaBoost",AB_accuracy)
print("The Classification report of AdaBoost\n",classification_report(y,AB_prediction))
AB_f1=f1_score(y,AB_prediction,average="macro")
print("The F1 score for AdaBoost: ",AB_f1)
X, y = x, y
y_binary = label_binarize(y, classes=np.unique(y))
# Create figure for ROC curves
plt.figure(figsize=(10, 5))
# Loop through classifiers and plot ROC curves
classifiers = [(RT_model, 'Random Forest'),
               (logistic_model, 'Logistic Regression'),
               (DT_model, 'Decision Tree'),
               (knn_model, 'K-Nearest Neighbors'),
               (clf, 'Multilayer Perceptron'),
               (nb_model,"Multinomial"),
               (svm_model,"SVM"),
               (AB_model,"AdaBoost"),

    for classifier, label in classifiers:
        if hasattr(classifier, 'predict_proba'):
            classifier.fit(X, y)
            y_probs = classifier.predict_proba(X)
        else:
            ovr_classifier = OneVsRestClassifier(classifier)
            ovr_classifier.fit(X, y)
            y_probs = ovr_classifier.decision_function(X)

        if y_probs.ndim > 1 and y_probs.shape[1] > 2:
            # For multiclass classification, y_probs should have shape (n_samples, n_classes)
```

```
    # Use label_binarize to convert predicted probabilities into binary matrix representation
    y_binary_probs = label_binarize(np.argmax(y_probs, axis=1),
classes=np.arange(y_probs.shape[1]))
    y_probs = y_binary_probs
  fpr = dict()
  tpr = dict()
  roc_auc = dict()
  for i in range(y_binary.shape[1]):
    fpr[i], tpr[i], _ = roc_curve(y_binary[:, i], y_probs[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
  all_fpr=np.unique(np.concatenate([fpr[i] for i in range(y_binary.shape[1])]))
  mean_tpr=np.zeros_like(all_fpr)
  for i in range(y_binary.shape[1]):
   mean_tpr += np.interp(all_fpr,fpr[i],tpr[i])
  mean_tpr/=y_binary.shape[1]
  roc_auc=auc(all_fpr,mean_tpr)
  plt.plot(all_fpr,mean_tpr,label=f'{label} (AUC={roc_auc:.2f})')
plt.plot([0,1],[0,1],'k--',label="Random")
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.0])
plt.legend(loc='lower right')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.show()
```

CHAPTER 5

RESULTS

Result snippet:

1. Bar Plot: Top 15 common Words present in particular the resume.

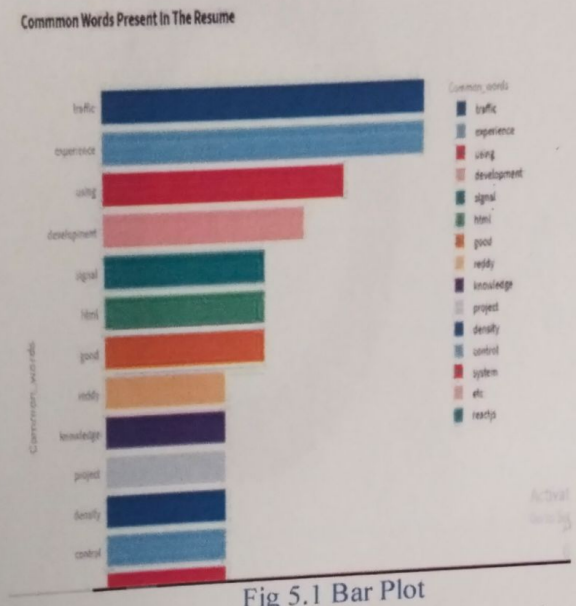

Fig 5.1 Bar Plot

2. Word Cloud: Display most frequent words in the resume.

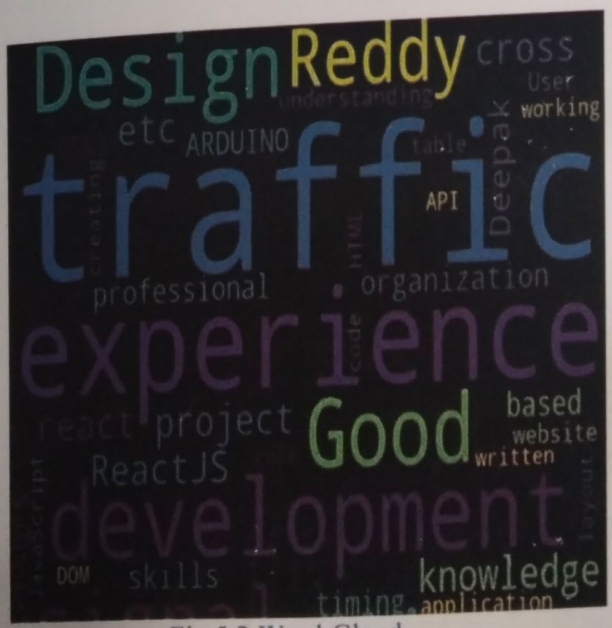

Fig.5.2 Word Cloud

3. Pie Chart: Display percentage of unique and duplicate words present in the resume.

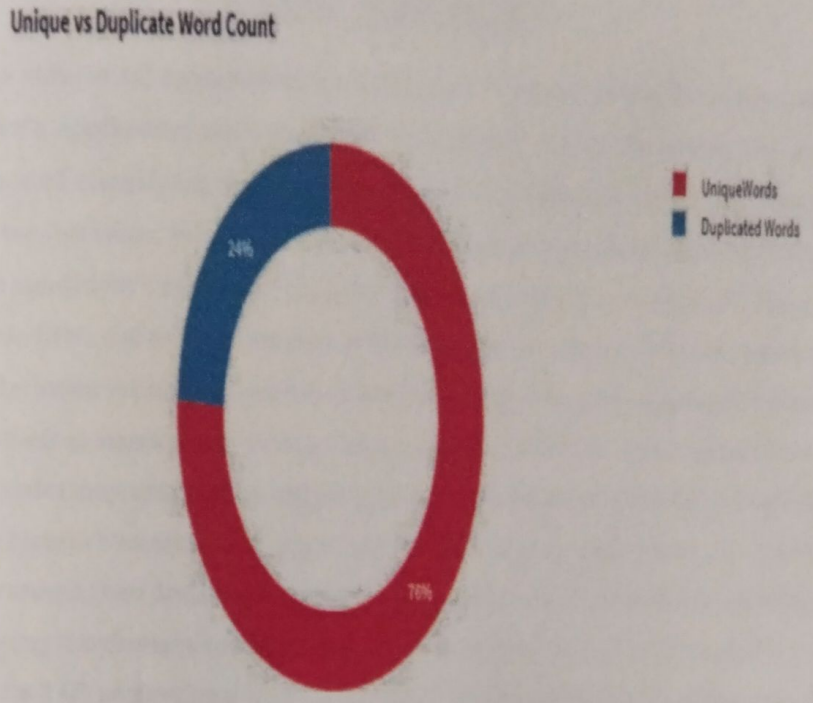

Fig 5.3 Pie Chart

CHAPTER 6
CONCLUSION

Huge number of applications received by the organization for every job post. Finding the relevant candidate's application from the pool of resumes is a tedious task for any organization nowadays. The process of classifying the candidate's resume is manual, time consuming, and waste of resources. To overcome this issue, we have proposed an automated machine learning based model which recommends suitable candidate's resume to the HR based on given job description. The proposed model worked in two phases: first, classify the resume into different categories. Second, recommends resume based on the similarity index with the given job description. The proposed approach effectively captures the resume insights, their semantics and yielded an accuracy of 78.53% with LinearSVM classifier. The performance of the model may enhance by utilizing the deep learning models like: Convolutional Neural Network, Recurrent Neural Network, or Long-Short Term Memory and others. If an Industry provides a large number of resume,then Industry specific model can be developed by utilizing the proposed approach. By involving the domain experts like HR professional would help to build a more accurate model, feedback of the HR professional helps to improve the model iteratively.Resume classification is a time-consuming, costly, and tedious job for an organization. In this regard, this study proposes an automated approach that uses various machine learning and NLP techniques for the classification of Resumes. The proposed methodology used several NLP and ML techniques for preprocessing data, feature extraction and representation, model construction, and evaluation for the Resume Classification task. The study results suggested that the TF-IDF vectorizer performed best in feature extraction and representation as the extracted features yielded excellent results on almost all classifiers. However, the Support Vector Machine (SVM) class algorithms such as (Linear, SVC) performed exceptionally good with over 98% and 96% accuracy respectively on the train and unseen test data. The study results are quite encouraging to automate the job application categorization and recommendation based on the content of the Resumes. The developed system can be deployed in real-time settings for an employer to automate the recruiting process.

In conclusion, the research paper demonstrases the effectiveness of using Natural Language Processing techniques and Machine Learning algorithms to classify resumes into their matching job roles. The proposed solution offers an automated and efficient way to screen resmes, reducing the time and cost associated with manual screening while ensuring fairness. The results of the study highlight the potential for this approach to improve the hiring process and provide a more efficient way to match job seekers with suitable job openings. As the demand for automated screening of resumes continues to grow, this research offers a promising solution for streamlining the recruitment process and providing better outcomes for both employers and job seekers.

CHAPTER 7

FUTURE SCOPE

The concept of classification is grasped by Resume Classification, and classification models have bom built using numerous techniques. This resume categorization platform will make the e-recruitment process more efficient and user-friendly. This approach will assist businesses and time throughout the recruitment Funre work includes combining these classification models with frontend resume portal where the user will be able to upload their resume and resume classification can work in the backend. From the results and discusions, we can conclude that various algorithms have given different accuracy score percentages with Random Forest having the highest accuracy among all three of them. Also, the confusion matrix of random forest is well suited for all the observed classesThereby, resume classification is achieved using proprocessing of data, eleaning of data and by applying various clansification algorithms.

This research paper can also be extended with the combination of other video interviewing features such as facial recognition, voice text generation and voice analysis. Building a plugin for people to include resum,classification models into their project is also one of thefuture goals of this research paper and project work. The major limitation and challenge for the Resume Classification and Recommendation task is finding an appropriate and standard dataset to process using the NLP techniques and train the ML models. Since the resume is not a standard document and there is no specific industry standard, thus major efforts were put on processing the documents in the dataset which were parsed from different formats and layouts. Moreover, the dataset size was a bit low to train the ML model for generalized classification. However, efforts were put to find a more suitable dataset for the classification task.The study achieved significant accuracy and performance gain on Resume Classification in different job categories. Therefore, in future work, the model will be extended to match the content of the resume with the provided job description. The extension in future work will enable the proposed system suitable for the complete recruiting process. The proposed system will perform the most tedious tasks of recruiting process; categorization and recommendation of suitable resumes for a given job description.

CHAPTER 8

REFERENCES

[1]. B. Balci, D. Saadati, D. Shiferaw, Handwritten Text Recognition using Deep Learning, STAN. U. (2017).

[2] Breaugh, J.A., 2009. The use of biodata for employee selection: Past research and future directions. Human Resource Management Review19, 219–231.

[3] Breiman, L., 2001. Random forests. Machine learning 45, 5–32.

[4] Pradeep Kumar Roya,* , Sarabjeet Singh Chowdhary, Rocky Bhatia, A Machine Learning approach for automation of Resume Recommendation system. International Conference on Computational Intelligence and Data Science (ICCIDS 2019)

[5] Sukanya V N Justin Seby, A Machine learning approach for Industry classification Using resume data .(IJCRT 2021)

[6] Lin, Y., Lei, H., Addo, P.C., Li, X., 2016. Machine learned resume-job matching solution. arXiv preprint arXiv:1607.07657 , 1–8.

[7]J. Ramos, Using TF-IDF to Determine Word Relevance in Document Queries, Rutgers University.

[8]Malinowski, J., Keim, T., Wendt, O., Weitzel, T., 2006. Matching people and jobs: A bilateral recommendation approach, in: Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06), IEEE. pp. 137c–137c.

[9]Farber, F., Weitzel, T., Keim, T., 2003. An automated recommendation approach to selection in personnel recruitment. AMCIS 2003 proceed- ¨ ings , 302

[10] Homa B. Hashemi, Amir Asiaee, Reiner Kraft, Query Intent Deduction using CNN, ACM, 2016

[11] Kevin, Neha, Detecting and classifying toxic comments, Stanford University, 2016

[12] Ladislav Lenc, Pavel Kr, Deep Neural Networks for Czech Multi-label Document Classification, CoRR, 2017

[13] Jonathan Woodbridge, Hyrum S. Anderson, Anjum Ahuja, and Daniel Grant, Predicting Domain Generation Algorithms with Long Short-Term Memory Networks, CoRR, 2016

[14] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei, Largescale Video Classification using CNN, Stanford University, 2014

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, Large-scale Video Classification using CNN, Stanford University,2014