# Resume Classification System using Natural Language Processing & Machine Learning Techniques

*Irfan Ali†, Nimra†, Ghulam Mujtaba†, Zahid Hussain Khand†, Zafar Ali†, and Sajid Khan†*

*†Center of Excellence for Robotics, Artificial Intelligence, and Blockchain, Department of Computer Science, Sukkur IBA University, Sukkur-65200, Sindh Pakistan*

**Abstract**

The selection and recommendation of a suitable job applicant from the pool of thousands of applications are often daunting jobs for an employer. The recommendation and selection process significantly increases the workload of the concerned department of an employer. Thus, Resume Classification System using the Natural Language Processing (NLP) and Machine Learning (ML) techniques could automate this tedious process and ease the job of an employer. Moreover, the automation of this process can significantly expedite and transparent the applicants' selection process with mere human involvement. Nevertheless, various Machine Learning approaches have been proposed to develop Resume Classification Systems. However, this study presents an automated NLP and ML-based system that classifies the Resumes according to job categories with performance guarantees. This study employs various ML algorithms and NLP techniques to measure the accuracy of Resume Classification Systems and proposes a solution with better accuracy and reliability in different settings. To demonstrate the significance of NLP & ML techniques for processing & classification of Resumes, the extracted features were tested on nine machine learning models Support Vector Machine - SVM (Linear, SGD, SVC & NuSVC), Naïve Bayes (Bernoulli, Multinomial & Gaussian), K-Nearest Neighbor (KNN) and Logistic Regression (LR). The Term-Frequency Inverse Document (TF-IDF) feature representation scheme proven suitable for Resume Classification Task. The developed models were evaluated using $F\text{-}Score_M$, $Recall_M$, $Precision_M$, and overall Accuracy. The experimental results indicate that using the One-Vs-Rest-Classification strategy for this multi-class Resume Classification task, the SVM class of Machine Learning algorithms performed better on the study dataset with over 96% overall accuracy. The promising results suggest that NLP & ML techniques employed in this study could be used for the Resume Classification task.

***Keywords:***

*Resume Classification, Natural Language Processing, Machine Learning, Text Classification, Recommender System*

## 1. Introduction

Internet-based recruiting systems have been rapidly adopted by recruiters in recent years. The rapid growth of the internet caused an identical growth in quantity of obtainable online information [1]. As a result, information is widely available. Contrary to this, information became overloaded and resulted in the need for information management [2, 3]. Moreover, the ever-increasing unemployment rate in developing countries like Pakistan results in considerable amount of job applications for a vacant position[4]. Thus, the selection of suitable job applicants from the pool of thousands applications is often a daunting job for an employer. Recruiters need to screen through a large amount of data to select the most suitable application from the pool. Thus, it significantly increases the workload of the concerned department of Recruiter [5]. Moreover, this process involves the engagement of considerable Human Resources and requires rigorous efforts and resources to finalize the most suitable applicant for further recruiting process. If the recruiters can figure out the non-relevant profiles at the earlier stages of the hiring process, this can significantly save time and money [6].

The Resume is a portfolio document developed by job applicants to present the relevant details for the vacant job. In this document, the applicant provides personal details, Educational details, accomplishments, competencies, skills, and experiences. This resume helps recruiters to shortlist the applicant from the pool of applications as it provides the complete picture of the applicant's competencies and skills. The resume screening demands domain knowledge to understand the suitability and relevance of an applicant for the advertised job vacancy. However, the current global economic condition that companies face of getting less capital to speculate within their HR department, while desperate to ensure that they are choosing the highly competitive applicant fitted to the job description [1]. Thus, recruiters are facing three main challenges:

- **Making sense of Resume:** This is a fact that Resumes in the market have no defined standard. Every resume may have a different structure in the pool of applications. Thus, HR needs to manually go through each resume to find out the best resume.
- **Mapping resume to the job description:** This is based on mapping the applicant's Resume to the requirements criteria provided by the recruiter. This process involves detailed screening and requires domain experts to efficiently perform this task.

Manuscript received July 5, 2024
Manuscript revised July 20, 2024
https://doi.org/10.22937/IJCSNS.2024.24.7.13

- **Managing the cost:** For, Screening and selection, Recruiters need to adopt automated processes with mere human involvement to save time and money.

Hence, Machine Learning based automated Resume Classification Systems can be used to classify the Resumes according to the job category. This approach can automate the tedious process of Resume Selection and support recruiters to overcome the above-mentioned challenges. Moreover, the automation of this process can significantly expedite the applicants' shortlisting process and also transparent the selection process with mere human involvement.

Text classification (TC) is a technique to automatically classify the predefined classes relevant to a particular text document [7, 8]. TC is one of the most fundamental tasks of Natural Language Processing (NLP). TC is carried out with the involvement of Supervised Machine Learning techniques. These techniques require text representation as a fixed-length feature vector [7]. Thus, Preprocessing and Feature Engineering are the most important and fundamental steps for such text classification tasks where we apply various feature extraction and feature representation techniques [9].

Feature extraction typically finds the set of most informative features whereas feature representation figures out the most suitable way to represent the values of extracted features. The most widely used feature extraction techniques for text documents are *N-grams*, *Bag of Words (BoW), and Word-to-Vec*. Every extracted feature assigned the numeric value using different representation techniques such as *Binary* and *TF-IDF*. Every feature engineering task has some pros and cons. Hence, the job of a Machine Learning Engineer is to find the most useful technique for the problem under consideration. Nevertheless, various Machine Learning approaches have been proposed to develop Resume Classification Systems in literature. However, this study aims at developing an ML-based system that classifies the Resumes according to job categories. The study applies the Supervised Machine Learning approach for resume classification to correctly classify 25 different job categories resumes belong to. The dataset has 962 labeled resumes' categories to train the classifier. Thus, various multi-class classification algorithms and NLP techniques are employed to measure the accuracy of Resume Classification using performance metrics such as overall accuracy, $\text{F-Score}_M$, $\text{Precision}_M$, and, $\text{Recall}_M$. This study proposes an ML-based Resume Classifier with better accuracy and performance guarantees.

The resume is an official and formal document used mainly for demonstrating the brief profile of a job applicant. The resume contains information related education, skills, experience, achievements, and portfolio of a job applicant. The resume often used as an effective tool to assess the overall suitability of an aspirant for the desired job. Moreover, in response to job postings applicants submit Resume as a formal document for job application consideration. The employer receives hundreds of Resumes for mere vacancies and finds it difficult to categorize and classify to a suitable job vacancy. Thus, this study attempts at developing an efficient and accurate Resume Classification System to ease the job of employers.

The rest of the paper is organized as follows. Section 2 presents the Review of related studies, Section3 describes the proposed Methodology to accomplish the objectives, Section 4 presents and discusses the findings of the study, and Section 5 presents the major limitations of the study and proposed future work and finally, section 6 concludes the study.

## 2. Related Work

In recent years, Machine Learning (ML) based text classification(TC) techniques have been widely employed in various domains [10] such as Sentiment analysis [11, 12], E-Commerce portals [13, 14], Email classification [15], Human Resource Management [2] and bioinformatics [16, 17]. In this study, ML-based text classification techniques are employed in the Human Resource Management domain. Various NLP and ML classification techniques have been employed to predict the category of Resume.

Several studies have proposed the Machine Learning based system for Human Resource Management and recruiting processes. For instance, the study [18] designed the approach for Resume ranking that uses that layered information retrieval framework to parse the resumes. The goal of this study was to help recruiters to find out the relevant job applicant for a job opening. Another study [19] designed the personalized approach for Resume-job matching that offers the statistical similarity for resume ranking according to the available jobs. This study could have been more generalized to recruiters as well as for job seekers. Employers can make use of this system to find the relevant resumes whereas job seekers can use to search the most relevant job matches their resumes. The fuzzy-based model used in [20] to evaluate the relevancy of a resume as compared to the job description. All the above-mentioned studies are working for document similarity by comparing the resume to the job description. However, few studies employed Supervised Text Classification Techniques to predict the category of Resume.

Perhaps, the most related work to the proposed approach is of [21]. In this wok, NLP and ML techniques were employed to predict the domain of resumes. This study aimed to allocate the relevant project to recruits. The study proposed the Named Entity Recognition (NER) approach coupled with various classification models such as Logistic Regression, K-Nearest Neighbors for the classification. Besides this, the study proposed an ensemble learning-based voting classifier that was retrained after a fixed

interval. Hence, the number of votes for each classifier was modified. The experimental results revealed that a voting based classifier produced 91.2% accuracy in predicting the categories while the accuracy was 84.2% without retraining. Another related study is of [22], in which the Convolutional Neural Network (CNN) was used to classify Resumes into 27 different job categories. In this study, CNN classifier was trained on word2Vec pre-trained representations to determine the category of Resume. This approach achieved 40.15% accuracy on resume classification and 74.88% accuracy on the job classification task. However, the study only used job summary text for classification and considered only one base method of fast Text for comparison of the performance.

Hence, both the aforementioned studies had some major limitations. The aforementioned studies had employed various classification techniques whereas failed to evaluate various preprocessing techniques for the proposed classifiers which may lead to low accuracy of the classifier. Further, only overall accuracy as a measure used for evaluation and failed to use various evaluation metrics such as F-Score, Precision, and Recall to evaluate the learning efficiency of classifiers.

It is evident from the above-mentioned studies that approaches used mainly suffered with two problems lower accuracy and performance comparison. Besides this, very few ML models were employed for the Resume Classification task and accuracy as the only measure used for performance. Moreover, the features extraction and representation techniques were not explored to overcome the less accuracy problem. To overcome the limitations of previously proposed studies, this study will use different NLP and Machine Learning techniques to improve the efficiency of classifiers and various performance matrices will be used for model evaluation. Also, various feature extraction and representation techniques would be employed for discriminative features contributing to better classification. Further, this study will provide discriminative features to several machine learning models, and various performance matrices such as $Precision_M$, $Recall_M$, and $F\text{-}Score_M$ will be used for performance measuring.

## 3. Methodology

This section discusses the proposed Methodology for building an efficient and accurate Resume Classification System in detail. To achieve the objective of Resume Classification, Natural Language Processing (NLP) and Machine Learning (ML) techniques employed using the best practices. The overall methodology designed approach devised in five stages as illustrated in figure 1: i) Data Collection and visualization ii) Preprocessing iii) Feature Engineering iv) Model Construction and v) Model Evaluation and testing in a real-time environment using Graphical User Interface (GUI).

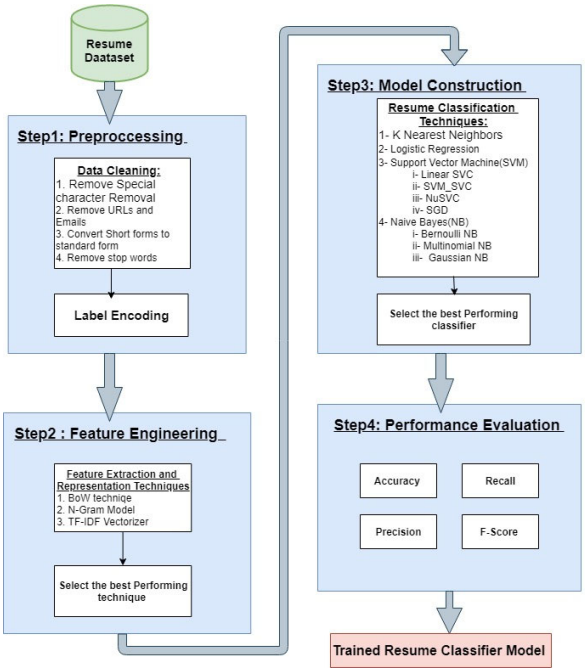


Fig. 1 The proposed methodology for resume classification

### 3.1 Data Collection and Visualization

The Resumes with Job Categories dataset was collected from an online data repository. The dataset is in Comma Separated Values (CSV) file format and has three columns namely ID, Category, and Resume's Text. The ID, Category and Resume Columns represent Index, Job Category/Field, and content of Resume respectively. The dataset contains 962 parsed and labeled resumes in 25 different job categories. The number of Resume instances for each class job category illustrated in figure 2 and category-wise distribution (percentage) of resume instances plotted in figure 3 using Python Matplotlib library. The visual evidence in Figure 2 shows that each job category has a different number of resume instances and this can lead to an imbalanced data problem. Moreover, the data for two categories namely *Java Developer* and *Testing* has the highest resumes instances and can be considered as biased class categories. Whereas, the resume instances for some categories such as *Advocate*, *Civil Engineer*, and *SAP Developer* are relatively less than some other categories for instance *Java Developer*. However, the category-wise total distribution in Figure 3 illustrates the overall representation of resume instances within the percentage range of (2.1 to 8.7%) in the dataset.

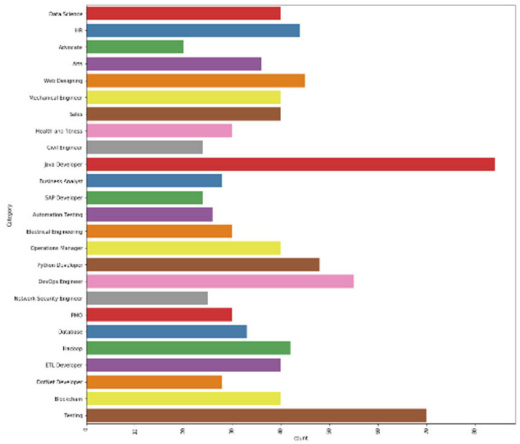

Fig. 2 Resume instances for each job category

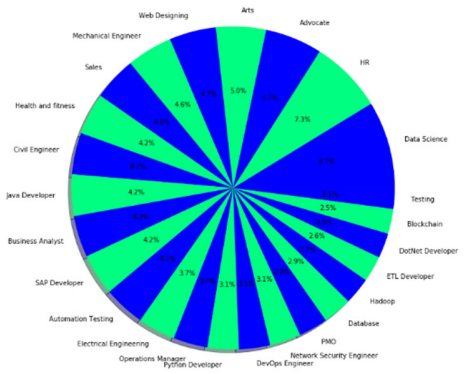


Fig. 3 Category-wise total distribution of Resume instances

## 3.2 Data Preprocessing

The Data preprocessing involves steps to transform raw data into meaningful information for the Machine Learning task. In the case of textual data for text classification, these steps involve cleaning raw text data, removing the unnecessary or meaning-less data, removing the repetitive (redundant) data, removing the missing (null) values, and transforming data to a common scale. To preprocess the resume's textual data for the Resume Classification task following key steps were performed.

### 3.2.1 Data Cleansing

The dataset contains the parsed resumes from different formats such as PDF, DOC, DOCX in a CSV format has a lot of unnecessary and unprocessed data in the resume column. Thus, the major efforts were required to preprocess the data and make it ready for Text Classification. In the data preprocessing step, the less informative text was cleaned using the Natural Language Processing Took Kit - NLTK [21] for stop words removal and Python 3.7.3 Regular Expressions. The following key tasks were performed for data preprocessing using the customized written program function in python as illustrated in Table 1.

I. The textual content of resumes was converted to lowercase
II. The special characters, punctuations, brackets, URLs, Email addresses, mentions, hash tags, apostrophes, leading & trailing characters, extra white spaces, and Non-ASCII characters were removed from the Resume's text
III. The masking was applied special escape sequences such as \n, \t, \a, \b, and so on
IV. The numbers were masked
V. The string fragmentations were masked
VI. The word phrases in short form such as I'll to I will were converted to their full forms
VII. Similar attributions were performed on unclean/unprocessed on raw resume's text data

Table 1 Method or Dataset cleaning

**CleanResume (resumeText):**

```
1.  resumeText = re.sub('http\S+\s*', ' ', resumeText)  # To remove URLs (Http, Https)
2.  resumeText = re.sub('RT|cc', ' ', resumeText)  # To remove RT and cc
3.  resumeText = re.sub('#\S+', '', resumeText)  # To remove possible hashtags
4.  resumeText = re.sub('@\S+', '  ', resumeText)  # To remove possible mentions and email addresses
5.  resumeText = re.sub('[%s]' % re.escape("""!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"""), ' ', resumeText)  # To  remove punctuations, brackets and special characters
6.  resumeText = re.sub(r'[^\x00-\x7f]',r' ', resumeText) # To replace Non-ASCII Characters with single space
7.  resumeText = re.sub('\s+', ' ', resumeText)  # To remove extra whitespaces
8.  resumeText = resumeText.lower() #To convert text to lowercase
9.  resumeText = re.sub(r"what's", "what is ", resumeText) # To chanage what's to what is
10. resumeText = re.sub(r"\'s", " ", resumeText) # To remove apposophy s
11. resumeText = re.sub(r"\'ve", " have ", resumeText) # To change 've to have
12. resumeText = re.sub(r"can't", "can not ", resumeText) # To change can't to can not
13. resumeText = re.sub(r"n't", " not ", resumeText) # To change n't to not
14. resumeText = re.sub(r"i'm", "i am ", resumeText) # To change i'm to i am
15. resumeText = re.sub(r"\'re", " are ", resumeText) # To change 're to are
16. resumeText = re.sub(r"\'d", " would ", resumeText) # To change 'd to would
17. resumeText = re.sub(r"\'ll", " will ", resumeText) # To change 'll to will
18. resumeText = re.sub(r"\'scuse", " excuse ", resumeText) # To change 'scuse to excuse
19. resumeText = re.sub('\W', ' ', resumeText) # To replace whitespaces
20. resumeText = resumeText.strip(' ') # To strip text (removing leading and trailing characters)
21. return resumeText
```

### 3.2.2 Removal the stop words

Stop words removal is one of the most essential steps in data preprocessing. Stop words such as 'is', 'each', 'and' and so on appear most often in any textual data. However,

these most frequently occurring words in a text document are not the informative features (tokens) for any classifier. Thus, these stop words should be removed from the corpus for the classification model. The stop words from the resume's text column were removed by performing the following steps using the python programming:

I. The word tokenization was performed on the resume's text using NLTK library and token were stored in an array
II. The standard English language stop words were imported using NLTK corpus and compared with each element in the tokenized array
III. If any element of the tokenized array was found in the list of NLTK stop words, that particular element (tokenized word) was removed
IV. Repeated this process for all the tokens
V. The final tokenized elements array did not contain any stop word

To visualize the stop words removal process, the word cloud of most frequently occurring words in the corpus of resumes was generated using the python word cloud feature as illustrated in figure 4. It can be observed that the word cloud now contains more informative words other than frequently occurring stop words and these words would be more meaningful for classifiers to learn.

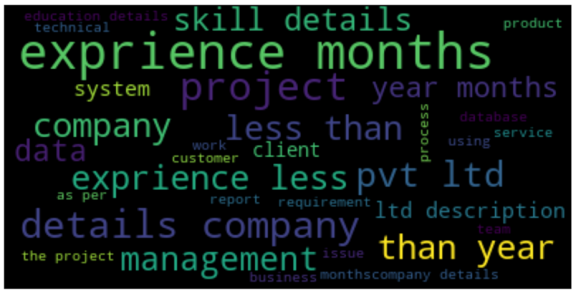


Fig. 4 Word Cloud of most frequent words in the cleaned dataset

### 3.2.3 Stemming & Lemmatization

Stemming & Lemmatization are known as Text Normalization or sometimes Word Normalization techniques in Natural Language Processing (NLP). The purpose of these techniques is to decrease word inflection in the corpus of classification text by mapping the group of words to the same *root stem*. Specifically, stemming and lemmatization remove the prefixes and suffixes (affixes) such as (-es, -s, -ed, in-, un-, -ing, etc) from words which result in inflectional (changing/deriving meaning of words). For instance, the *stem (root)* word for *Plays, Playing, and Played* is ***Play*** so the *stemming and lemmatization* techniques would map these words in the corpus of classification text to *root (stem)* word.

$$\begin{pmatrix} Works \\ Working \\ Worked \end{pmatrix} \rightarrow \begin{pmatrix} Work \\ (root\ stem\ word) \end{pmatrix}$$

$$\begin{pmatrix} am \\ are \\ is \end{pmatrix} \rightarrow \begin{pmatrix} be \\ (simple\ present\ forms\ of\ the\ verb\ -to\ be) \end{pmatrix}$$

$$\begin{pmatrix} Cars \\ Car's \\ Cars' \end{pmatrix} \rightarrow \begin{pmatrix} be \\ (singular\ and\ simple\ form\ of\ noun) \end{pmatrix}$$

*Using the above mappings, a sentence could be normalized using the stemming and lemmatization techniques as follow;*

$$(The\ boy's\ cars\ are\ different\ colors\ ) \rightarrow (The\ boy\ car\ be\ differ\ color)$$

The Natural Language Tool Kit (NLTK) library in python offers the implementation of stemming and lemmatization techniques in Python with different settings. However, unlike stemming offered by the NLTK library in Python, the lemmatization reduces the inflected words properly by ensuring the root word belongs to the language. Thus, we implemented lemmatization on our Resume's text corpus as the Resumes are more formal documents. The code implementation of lemmatization implementation presented in below code snippet;

```
#Pre-Processing Step Three : Stemming & Lemmitization

#Importing NLTK Library
import nltk
#Importing Word Net Lemmatizer from NLTK Stem Library
from nltk.stem import WordNetLemmatizer
#Importing String Library
import string
#Importing WordCloud for Plotting most frequest words
wordnet_lemmatizer = WordNetLemmatizer()

#Stop words from Enlish Language Corpus
oneSetOfStopWords = set(stopwords.words('english')+['``',"''"])
#array for storing total words
totalWords =[]
#Extracing Sentences (Data) from Resume Column
Sentences = DataSet['Resume'].values
cleanedSentences = ""
for i in range(0,160):
    #Cleaning Resume Text using CleanResume Function
    cleanedText = cleanResume(Sentences[i])
    cleanedSentences += cleanedText
    #Tokenizing using NLTK Word Tokenization
    requiredWords = nltk.word_tokenize(cleanedText)
    #Appending Tokenized words to totalwords array
    for word in requiredWords:
        if word not in oneSetOfStopWords and word not in string.punctuation:
            totalWords.append(wordnet_lemmatizer.lemmatize(word))
```

Fig. 5 Lemmatization - Code Snippet

### 3.2.4 Label Encoding

The label encoding technique handles the categorical values of variables in the Machine Learning Model. The label encoding technique assigns a unique integer value to a categorical variable. To make raw text data ready for the machine learning model the label encoding was done to assign a numerical label to all categories shown in figure 2. The Scikit-learn Label Encoder used for the mentioned purpose. Hence, the label encoder on the Category field of the data was applied. Figure 6 shows the code snippet for Label encoding.

```
from sklearn.preprocessing import LabelEncoder
var_mod = ['Category']
le = LabelEncoder()
for i in var_mod:
    DataSet[i] = le.fit_transform(DataSet[i])
```

Fig. 6 Label Encoding - Code Snippet

## 3.3 Feature Engineering

The feature engineering helps to extract, formulate, and represent the set of most discriminative (informative)

features from the corpus of text for the classification task. After data cleaning and preprocessing, the resume's text data has an informative set of words. However, figure 4 shows the list of most frequent words in our dataset. It shows dataset does not contain stop words and other less informative words now. Therefore, we use different methods for feature extraction however, The Term Frequency – Inverse Document Frequency (TF-IDF) features representation works best in our dataset. Thus, the **TF-IDF** technique used for representing the value of each extracted feature.

#### 3.3.1 Feature Extraction & Master Feature Creation

After applying the preprocessing step on the data, the dataset contains the words that are important features for the classification. To demonstrate the significance, different ways for feature extraction namely, *BoW*, *Word Vectorizer*, *N-Gram* were used and tested. However, our model showed the best results on *Word vectorizer* implementation using the TF-IDF feature representation. Steps for feature extraction are as follows:

I. ***Design Vocabulary***: Here, we made the list of all words presented in the Resume field of all records.
II. ***Document Vector***: In this step, we represent each word as a feature and make a separate field for each word of vocabulary. The objective of this step is to map each Resume free text into the vector.

#### 3.3.2 Feature Representation

This step aims to allocate an arithmetic value to each of the extracted features in the vector. The different methods for representing the features like *BOW*, *CountVectorizer,* and *N-gram* were employed however our model yielded the best accuracy on the TF-IDF vectorizer. Therefore, TF-IDF [23] used for representing the value of each extracted feature. **TF-IDF** is a numerical statistic that is intended to find the importance of a word to a document in the collection. This technique is concerned with two things. **TF** is concerned with the occurrences of each word/feature and determines how frequently the word appears in each document. Whereas, **IDF** is used to determine the weight of each word in the document. The objective of **TF-IDF** feature representation is to weigh down the more frequent words while scaling up the rare words in the document.

Hence, Tf-idfVectorizer implemented using Python Scikit-Learn library. It is used to perform both feature extraction and feature Representation for the task. There is a parameter that allows us to use the topmost features concerning the TF-IDF score. To compare the performance of most discriminative features, the different values for the max-feature sub-set were tested. However, the accuracy of classifiers was decreasing as the max-feature value was increased. For instance, the max-feature value 2000 and 1500 resulted in an accuracy of 95% and 97% respectively on SVM-SVC. Thus, can be concluded that the larger value of the max-feature sub-set was not significantly contributing to better accuracy so the max-feature value set to 1500. The following coding snippet represents the implementation of the feature representation using TF-IDF.

```
word_vectorizer = TfidfVectorizer(
    sublinear_tf=True,
    stop_words='english',
    max_features = 1500)
word_vectorizer.fit(requiredText)
WordFeatures = word_vectorizer.transform(requiredText)
```

Fig. 7 Feature Extraction and representation - Code Snippet

### 3.4 Resume Classifier Construction

The discriminative features extracted using the techniques described in the previous section were used to build the classifier to accurately classify the Resumes. Several Machine Learning classifiers were opted to select the best performing model for deployment and Graphical User Interface (GUI). The details of Classifier construction presented in sections below.

#### 3.4.1 Implementation details and experimental setup

After extracting features from the dataset, the data divided into training and testing. The dataset was divided into 70% and 30% for train and test set respectively. Nine different text classifiers were employed as each has its own philosophy to classify the instances. The "One-Vs-Rest-Classification" strategy for multiclass classification used [24]. The brief description of the implemented nine machine learning models is as follows:

1. **K Nearest Neighbors(KNN):** KNN is based on finding k-nearest data points to the new instance and assign the label according to the highest neighboring data points. KNN is also known as a lazy learner classifier because of its simplest method of Euclidian distance Eq 1 for classification tasks [24].

$$\sqrt{\sum_{i=1}^{n}(xi - yi)^2} \qquad (1)$$

2. **Multinomial Naïve Bayes (MNB):** Naïve Bayes classifier is based on the conditional probability. NB classifier finds the probability of a vector belongs to the class. It finds out the probability for all the given instances and classifies the with conditional probability. It is based on strong independence between the features. MNB is one variant of Naïve Bayes that multinomial distribution of all pairs [25].
3. **Bernoulli Naïve Bayes (BNB):** it is also a variant of Naïve Bayes that accepts the binary features only. BNB is also effective for classification tasks [26].
4. **Gaussian Naïve Bayes (GNB):** It is also a variant of NB that supports continuous-valued features that are assumed to be distributed according to Gaussian distribution. GNB only supports vectorized features representation to implement GNB vectorized features representation used [27].

5. **Logistic Regression (LR):** Logistic Regression applies the logistic function on the classification task with a threshold value. LR considered one of the easiest implementations for classification problems [28].
6. **Linear Support Vector Classifier (SVC):** It is based on finding the best separating line between two classes. It is the simplest form of Support vector machine that finds the linear hyperplane between two classes. Although, it will not give good results if the data is not linearly separable. Linear SVM is also known as the least square support vector machine classifier [29].
7. **Support Vector Classifier (SVC):** SVC overcomes the above-mentioned issue of Linear SVM by using the Kernel concept [30] that works well on data that is not linearly separable.
8. **Nu-Support Vector Classifier (NuSVC):** It is similar to the SVC but it also uses a parameter to control the number of support vectors.
9. **Stochastic Gradient Descent (SGD):** It uses SGD for training (that is, looking for the minima of the loss using SGD).

The extracted features and learned ML models were stored in Python external pkl file format for future evaluation and testing. The scikit-learn externals joblib library used to store extracted features representation and learned models on disk and later used in GUI for real-time testing.

#### 3.4.2 Graphical User Interface & system Evaluation in a real-time environment

To evaluate the trained and learned ML models in real-time settings on unseen data the Graphical User Interface (GUI) designed using the Python Tkinter. The extracted features and learned ML models imported to use in GUI. The designed and developed GUI allows users to provide a resume in text format or select a resume text from an unseen test dataset. The GUI also leverages users to select from nine ML learned models for classification of resume. This implementation ensures the transparency and real-time analysis of Resume Classification on nine learned models. The designed GUI would be also helpful for implementing Machine Learning models in a real-time environment and helpful for recruiters to tackle the tedious task of Resume Classification in different job categories.

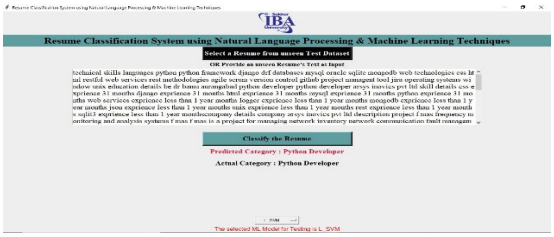


Fig. 8 Graphical User Interface of the proposed system

### 3.5 Evaluation Matrices

To measure the performance of the mentioned Classification models, we use different performance evaluation matrices. As the dataset was imbalanced (shown in Figures 2 and 3) so the overall accuracy was not only a significant matrix for model evaluation. Therefore, for performance evaluation, Overall accuracy, $\text{Precision}_M$, $\text{Recall}_M$, $\text{F-Score}_M$ matrices were used. The brief description of performance matrices is as follows.

I. **Overall Accuracy:** Accuracy is a fraction of predictions that are correctly identified by the algorithms. However, Accuracy itself does not tell the full story when we are working the imbalanced data.

II. **Macro Precision ($\text{Precision}_M$):** $\text{Precision}_M$ attempts to answer that from all the positive predictions, what fraction of actually positive? The value of precision is between 0 and 1. Any model that does not produce false-positive results has a precision of 1. It gives us the idea that how precisely the model is identifying the True positive values of classes. In multiclass classification problem precision for all classes is computed and then the average of all results is computed. Macro-average computes the metric independently for each class and then take the average. The mathematics definition of $\text{Precision}_M$ is as follows;

$$Precision_M = \frac{\sum_{i=1}^{c} \frac{TP_i}{TP_i+FP_i}}{C}$$

III. **Macro Recall ($\text{Recall}_M$):** $\text{Recall}_M$ attempts to answer that from all the actual positive records, what fraction is correctly identified? The value of precision is between 0 and 1. Any model that does not produce false-negative results has a precision of 1. In multiclass classification problem precision for all classes computed then the average of all results computed. Macro-average computes the metric independently for each class and then take the average. The mathematical definition of $\text{Recall}_M$ is as follows.

$$Recall_M = \frac{\sum_{i=1}^{c} \frac{TP_i}{TP_i+FN_i}}{C}$$

IV. **Macro F-Score ($\text{F-Score}_M$):** $\text{F-Score}_M$ or $\text{F-Measure}_M$ is defined by the weighted harmonic mean of test's precision and recall. The values are between 0 and 1 where highest value '1' shows that algorithm reaches to best precision and recall values.

## 4. Results and Discussion

Table 2 presents the $\text{Precision}_M$, $\text{Recall}_M$, $\text{F-Score}_M$, and Overall accuracy of all the trained model on test data. The variation in the performance of trained models can be significantly observed. The Support Vector Machine class of learning algorithms perform better than other classifiers. In all 318 analyses on test data instances, the Linear Support

Vector Classifier out-perform the other eight classifiers with nearly 98% overall accuracy and 1.0 **Precision$_M$** which can be generalized as for the Resume Text Classification task SVM class classifiers perform best.

Table 2: Performance Evaluation of learned ML Models

| Classifier | Precision$_M$ | Recall$_M$ | F-Score$_M$ | Overall Accuracy % | Misclassification % |
|---|---|---|---|---|---|
| LSVC | 1.00 | 1.00 | 1.00 | 99.6 | 0.4 |
| SGD | 1.00 | 1.00 | 1.00 | 99.6 | 0.4 |
| LR | 1.00 | 0.99 | 0.99 | 99.3 | 0.7 |
| SVC | 1.00 | 0.99 | 0.99 | 99.3 | 0.7 |
| NuSVC | 0.99 | 0.99 | 0.99 | 99.3 | 0.7 |
| KNN | 0.99 | 0.98 | 0.99 | 97.2 | 2.8 |
| GNB | 0.98 | 0.96 | 0.96 | 96.5 | 3.5 |
| MNB | 0.98 | 0.95 | 0.96 | 94.8 | 5.2 |
| BNB | 0.89 | 0.76 | 0.79 | 79.2 | 20.8 |

Table 1 summarizes the Precision$_M$, Recall$_M$, F-Score$_M$, and overall Accuracy of classifiers on testing data. The results show that most of the algorithms produced excellent results on study data this can be comprehended as the dataset size was optimal and best NLP & ML techniques were employed to achieve significantly better results. It is also shown that LSVC, SGD, LR, and SVC produced exceptionally well results. Thus, the LSVC classifier is the best performing classifier.

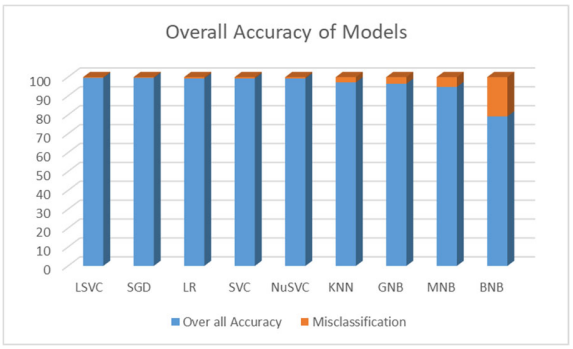


Fig. 9: Overall Accuracy vs Misclassification Report

Figure 9 illustrates the overall accuracy and misclassification report of the classifiers. It can also be seen that BNB (Bernoulli Naïve Bayes) didn't produce better results as compared to all other classifiers while MNB (Multinomial Naïve Bayes) performed well on the dataset. The misclassification of BNB is high as compare to all other classifiers. One of the reasons for that misclassification is Bernoulli's classifier mainly used for Binary classification and treating all values as the negative class whereas, the Resume Classification is a multi-class problem. Most of the models produced better approximately similar results except the BNB. The overall misclassification report is relatively low; thus this can be inferred that the extracted features using TF-IDF were the most discriminative for the Resume Classification Task. Moreover, the GNB & BNB model requires a vectorized representation of features and this could be a reason for slightly poor performance.

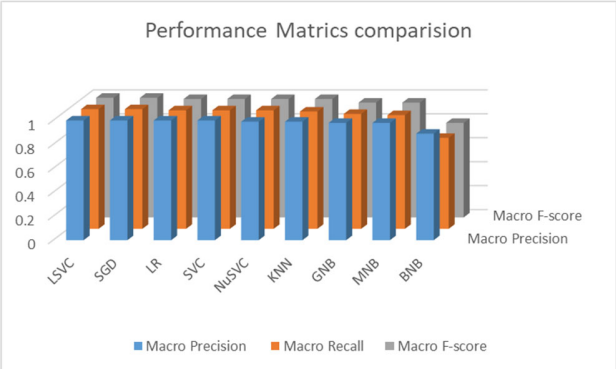


Fig. 10 PrecissionM, RecallM, F-ScoreM - Performance Matrices

Figure 10 illustrates the Precission$_M$, Recall$_M$, F-Score$_M$ of all the models. There is a minor difference in all three Precission$_M$, Recall$_M$, F-Score$_M$. Well, this was not the case with un-processed data was used. The same performance matrices were measured on raw data and results were not encouraging. Hence, our designed Methodology extracted the most discriminative features from the dataset. That is the reason why most of the classifiers yielded the best performance.

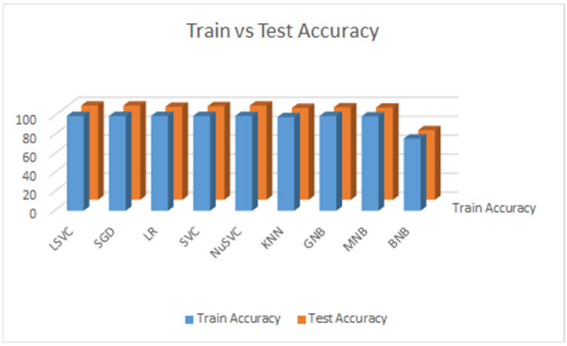


Fig. 11 Train vs Test Accuracy

Figure 11 illustrates the Train versus Test Accuracy of used nine classifiers. The overall dataset was divided into 70% and 30% for training and testing

respectively. Machine Learning models often suffer with ***overfitting*** and ***underfitting*** problems.

The ***overfitting*** problem occurs when the learned ML model performs best on *training data* and yields better accuracy however, fails to perform well on the test or *unseen data* [31]. The ***overfitting*** problem yields *higher train accuracy* and *lower test accuracy*. Whereas, the ***underfitting*** problem occurs when the model fails to perform well either on test or train data. The ***underfitting*** yields slighter lower accuracies for train and test data.

It is evident from figure 11 that the proposed models in this study are neither ***overfitting*** nor ***underfitting*** the train or test data. The trained models equally performing better on training and test data. It can be inferred that the overall process of Natural Language Processing (NLP) and Machine Learning (ML) techniques employed efficiently to yield balanced and better performance on test and train data.

## 5. Limitations and Future work

The major limitation and challenge for the Resume Classification & 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.

## 6. Conclusion

Resume classification is a time-consuming, costly, and tedious job for an organization. In this regard, this study proposed an automated approach that uses various machine learning and NLP techniques for the classification of Resumes. The proposed methodology used several NLP & 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, NuSVC, and SGD) 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 Resumes. The developed system can be deployed in real-time settings for an employer to automate the recruiting process.

**References**


1. Koyande, B.A., et al., *Predictive Human Resource Candidate Ranking System.*
2. Al-Otaibi, S.T. and M. Ykhlef, *A survey of job recommender systems.* International Journal of Physical Sciences, 2012. **7**(29): p. 5127-5142.
3. Färber, F., T. Weitzel, and T. Keim, *An automated recommendation approach to selection in personnel recruitment.* AMCIS 2003 proceedings, 2003: p. 302.
4. Breaugh, J.A., *The use of biodata for employee selection: Past research and future directions.* Human Resource Management Review, 2009. **19**(3): p. 219-231.
5. Lin, Y., et al., *Machine learned resume-job matching solution.* arXiv preprint arXiv:1607.07657, 2016.
6. Yi, X., J. Allan, and W.B. Croft. *Matching resumes and jobs based on relevance models*. in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. 2007.
7. Sebastiani, F., *Machine learning in automated text categorization.* ACM computing surveys (CSUR), 2002. **34**(1): p. 1-47.
8. Nigam, K., et al., *Text classification from labeled and unlabeled documents using EM.* Machine learning, 2000. **39**(2-3): p. 103-134.
9. Uysal, A.K. and S. Gunal, *The impact of preprocessing on text classification.* Information Processing & Management, 2014. **50**(1): p. 104-112.
10. Otter, D.W., J.R. Medina, and J.K. Kalita, *A Survey of the Usages of Deep Learning for Natural Language Processing.* IEEE Transactions on Neural Networks and Learning Systems, 2020: p. 1-21.
11. Parkhe, V. and B. Biswas, *Sentiment analysis of movie reviews: finding most important movie aspects using driving factors.* Soft Computing, 2016. **20**(9): p. 3373-3379.
12. Bakshi, R.K., et al. *Opinion mining and sentiment analysis*. in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. 2016. IEEE.
13. Sivapalan, S., et al. *Recommender systems in e-commerce*. in *2014 World Automation Congress (WAC)*. 2014. IEEE.
14. Srifi, M., et al., *Recommender Systems Based on Collaborative Filtering Using Review Texts—A Survey.* Information, 2020. **11**(6): p. 317.
15. Mujtaba, G., et al., *Email classification research trends: review and open issues.* IEEE Access, 2017. **5**: p. 9044-9064.
16. Al-garadi, M.A., et al., *Using online social networks to track a pandemic: A systematic review.* Journal of biomedical informatics, 2016. **62**: p. 1-11.
17. Mujtaba, G., et al. *Automatic text classification of ICD-10 related CoD from complex and free text forensic autopsy reports*. in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 2016. IEEE.

18. Gonzalez, T., et al. *Adaptive Employee Profile Classification for Resource Planning Tool*. in *2012 Annual SRII Global Conference*. 2012.
19. Guo, S., F. Alamudun, and T. Hammond, *RésuMatcher: A personalized résumé-job matching system.* Expert Systems with Applications, 2016. **60**: p. 169-182.
20. Golec, A. and E. Kahya, *A fuzzy model for competency-based employee evaluation and selection.* Computers & Industrial Engineering, 2007. **52**(1): p. 143-161.
21. Gopalakrishna, S.T. and V. Vijayaraghavan, *Automated Tool for Resume Classification Using Sementic Analysis.* International Journal of Artificial Intelligence and Applications (IJAIA), 2019. **10**(1).
22. Sayfullina, L., et al. *Domain adaptation for resume classification using convolutional neural networks*. in *International Conference on Analysis of Images, Social Networks and Texts*. 2017. Springer.
23. Ramos, J. *Using tf-idf to determine word relevance in document queries*. in *Proceedings of the first instructional conference on machine learning*. 2003. New Jersey, USA.
24. Xu, J., *An extended one-versus-rest support vector machine for multi-label classification.* Neurocomputing, 2011. **74**(17): p. 3114-3124.
25. Loper, E. and S. Bird, *NLTK: the natural language toolkit.* arXiv preprint cs/0205028, 2002.
26. Kibriya, A.M., et al. *Multinomial naive bayes for text categorization revisited*. in *Australasian Joint Conference on Artificial Intelligence*. 2004. Springer.
27. McCallum, A. and K. Nigam. *A comparison of event models for naive bayes text classification*. in *AAAI-98 workshop on learning for text categorization*. 1998. Citeseer.
28. Raschka, S., *Naive bayes and text classification i-introduction and theory.* arXiv preprint arXiv:1410.5329, 2014.
29. Xu, S., *Bayesian Naïve Bayes classifiers to text classification.* Journal of Information Science, 2018. **44**(1): p. 48-59.
30. Schölkopf, B., A.J. Smola, and F. Bach, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. 2002: MIT press.
31. Suykens, J.A. and J. Vandewalle, *Least squares support vector machine classifiers.* Neural processing letters, 1999. **9**(3): p. 293-300.


**Irfan Ali** is currently pursuing Master's Degree in Computer Science at the Department of Computer Science, Sukkur IBA University, Pakistan. He is also working as the Assistant Director Research at Office of Research, Innovation & Commercialization – ORIC at the Benazir Bhutto Shaheed University of Technology and Skill Development – BBUSTSD, Khairpur Mir's, Pakistan. Prior to this, He remained associated with Sukkur IBA University for more than five years and worked as Research Associate in various departments during this period he worked on several research projects, organized international research conferences and seminars. Ali received his Bachelor's Degree in Computer Science from Sukkur IBA University. His research areas of interest are Artificial Intelligence, Machine & Deep Learning, Data Science, Big Data Analytics, Text Mining, Text Classification, Image Classification, and IoT.

**Nimra** is MS Research Scholar at the Department of Computer Science, Sukkur IBA University, Pakistan. She is also working as a Subject Specialist, Computer Science at IBA Public School, Sukkur, Pakistan. She received her Bachelor's degree in Computer Science from Sukkur IBA University, Pakistan in 2018. Her research interests include Machine Learning, Natural Language Processing, Text mining, text classification, and Cloud Computing.

**Ghulam Mujtaba** is currently working as an Associate Professor at the Department of Computer Science, Sukkur IBA University. He is also working as the Director of the Center of Excellence for Robotics, Artificial Intelligence, and Blockchain (CRAIB). He has been associated with Sukkur IBA University since 2006. He received his Doctorate in the field of Computer Science from the University of Malaya, Kuala Lumpur, Malaysia in 2018. His field of research includes artificial intelligence, machine learning, online social networking, text mining, text classification, image classification, and deep learning. Dr. Mujtaba teaches various courses such as Computer Programming, Object-Oriented Programming, Data Science, Machine Learning, Natural Language Processing, Deep Learning, and Advanced Research Methods. He has authored or co-authored several articles in academic journals indexed in well-reputed databases.

**Zahid Hussain Khand** is currently working as Registrar, Sukkur IBA University. He has been associated with Sukkur IBA University since 2003. His field of research includes Information and Communication Technology, Agri-tech, and Smart-tech. Mr. Khand teaches various courses such as Network Security, Computer Networks, Data Communication, Internet of Things, and Research Methods. He has authored or co-authored several articles in academic journals indexed in well-reputed databases.

**Zafar Ali** is currently working as an ERP Manager at Sukkur IBA University. He has been associated with Sukkur IBA University since 2008. His field of research includes Information and Communication Technology, ERP, Visual Programming, Databases, Information Retrieval, and Distributed Databases.

**Sajid Khan** is currently working as an Assistant Professor at the Department of Computer Science, Sukkur IBA University He received the B.S. degree in telecom engineering from FAST-NUCES University, Pakistan, in 2011, and the M.S. leading to Ph.D. degrees in electronics and communication engineering from Hanyang University, Ansan, South Korea.