AI-powered Resume

NAME: SAYAN ROY CHOWDHURY

2025

INDEX

Introduction

Recruiters often spend a significant amount of time skimming through resumes to find the best candidates for a job position. With potentially hundreds of applications per position, manual screening becomes time-consuming and inefficient. Traditional keyword-matching techniques are widely used but often fail to capture the true relevance of a candidate's skills and experience.

This project aims to develop an AI-powered Resume Parser that automates resume screening using Natural Language Processing (NLP) and Machine Learning (ML) techniques. The goal is to identify skilled candidates while filtering out irrelevant applications, reducing the workload of recruiters and increasing hiring efficiency.

The final outcome of this project is a scoring mechanism that assigns a weightage score (from 0 to 10) to each resume based on its relevance to a given job profile. This score will help recruiters instantly identify top candidates, eliminating those who do not meet the required qualifications while prioritizing the most promising applicants.

Project Objectives

The key objectives of this project are:

- ✓ Automate the process of parsing resumes using AI and NLP.
- ✓ Extract and analyze key skills, experience, and qualifications.
- ✓ Score each resume based on relevance to the job role.
- ✓ Improve hiring efficiency by reducing manual screening efforts.

Dataset Description

The Resume Dataset from Kaggle has been used for this project. It contains resumes along with their associated job categories. This dataset is essential for training the AI-powered Resume Parser, as it provides structured resume data that can be analyzed and processed using Natural Language Processing (NLP) and Machine Learning (ML) techniques.

The dataset consists of two primary columns:

- Job Category: The specific job role, industry, or field associated with the candidate's resume. This helps in classifying resumes into relevant professional domains such as Data Science, Web Development, Mechanical Engineering, Business Analysis, etc.
- Resume Text: The unstructured textual content of the candidate's resume, containing personal information, skills, work experience, education, certifications, and other relevant details.

Data Preprocessing

Given the raw and unstructured nature of the dataset, several preprocessing steps are performed before using it for classification and clustering:

1. Text Cleaning:
    - Removal of URLs, special characters, punctuations, and stopwords.
    - Standardization of text (lowercasing, stemming, and lemmatization).
2. Tokenization & Vectorization:
    - The resume text is converted into numerical features using TF-IDF (Term Frequency-Inverse Document Frequency) to capture important terms.
3. Label Encoding:

- The Job Category column is converted into numerical labels for machine learning models.

4. Feature Engineering:
   - Extraction of key skills, education details, and experience levels from resumes.
5. Data Splitting:
   - The dataset is divided into training (80%) and testing (20%) sets for model evaluation.

This dataset forms the backbone of the project, enabling automated resume screening by learning from real-world job applications and mapping candidate skills to job roles effectively.

Model Selection and Training

Several **machine learning models** were trained and evaluated to classify resumes and automate the screening process. The aim was to accurately categorize resumes into relevant job domains and identify skilled candidates based on their experience and qualifications.

1. K-Means Clustering

- **Purpose**: An unsupervised learning algorithm used to group resumes into clusters based on similarity.
- **Process**:
  - The **Elbow Method** was used to determine the optimal number of clusters.
  - Resumes were grouped into these clusters, aiding in efficient sorting.

2. K-Nearest Neighbors (KNN)

- **Purpose**: A supervised learning algorithm that classifies resumes by comparing them to similar ones in the dataset.
- **Process**:
  - The algorithm calculates the distance between resumes and assigns categories based on the **k-nearest** similar resumes.

3. Logistic Regression

- **Purpose**: A classification algorithm that predicts the probability of a resume belonging to a specific category.
- **Process**:
  - Uses **TF-IDF vectorized** data to classify resumes based on textual features.

Final Model Selection

After evaluating different models based on accuracy, precision, recall, and F1-score, **KNN and Logistic Regression** were chosen for classification, while **K-Means was used for clustering similar resumes** but its accuracy was very less.

These models effectively streamline the resume screening process, reducing manual effort and improving candidate selection.

Resume Scoring Mechanism

To enhance the resume screening process, a custom scoring system was developed to assign a weightage score to each resume. This score reflects the relevance of the resume to the desired job category and helps recruiters quickly identify the most suitable candidates.

The scoring system assigns scores on a scale of 0 to 10, where:

- 0 represents the least favorable resume, meaning it does not match the job requirements.
- 10 represents the most favorable resume, indicating a strong alignment with the job role and necessary skills.

A custom scoring system assigns a weightage score (0 to 10) to each resume based on relevance, helping recruiters identify the best candidates efficiently.
Scoring Criteria

1. Job Category and Skill Matching
    - Resumes are assessed for alignment with the target job category and required skills.
    - Higher scores are assigned to resumes that closely match the role.
2. Relevant Keywords
    - Keywords related to technical skills, industry terms, and certifications are extracted.
    - Resumes with more relevant terms receive a higher score.
3. Experience Level
    - The system evaluates years of experience, projects, and certifications to determine expertise.
    - More experienced candidates receive higher scores.

The final score is calculated based on these factors, ranking resumes effectively for quick and accurate shortlisting.

Results and Performance Evaluation

The models were thoroughly evaluated using accuracy metrics and classification reports to determine their effectiveness in classifying resumes. Among the various models tested, Logistic Regression demonstrated the highest accuracy in predicting the correct job category, making it the most reliable option for resume classification.

On the other hand, the K-Means clustering algorithm yielded the lowest accuracy among all models. While it helped in grouping resumes based on textual similarity, it lacked the precision required for accurate job classification. This result highlights the limitation of unsupervised learning methods for this task, as they do not leverage labeled data for improved classification performance.

Code And Output

1.Importing Libraries And Dataset:

```
[1] import numpy as np
    import pandas as pd
    import  matplotlib.pyplot as plt
```

```
df=pd.read_csv('/content/UpdatedResumeDataSet.csv')
df.head(10)
```

	Category	Resume
0	Data Science	Skills * Programming Languages: Python (pandas...
1	Data Science	Education Details \r\nMay 2013 to May 2017 B.E...
2	Data Science	Areas of Interest Deep Learning, Control Syste...
3	Data Science	Skills â□¢ R â□¢ Python â□¢ SAP HANA â□¢ Table...
4	Data Science	Education Details \r\n MCA YMCAUST, Faridab...
5	Data Science	SKILLS C Basics, IOT, Python, MATLAB, Data Sci...
6	Data Science	Skills â□¢ Python â□¢ Tableau â□¢ Data Visuali...
7	Data Science	Education Details \r\n B.Tech Rayat and Bahr...
8	Data Science	Personal Skills â□¢ Ability to quickly grasp t...
9	Data Science	Expertise â□□ Data and Quantitative Analysis â...

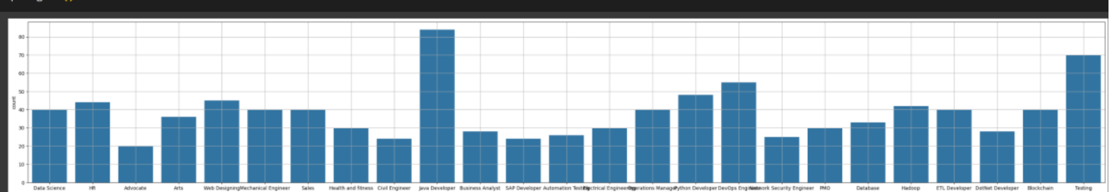
2.Getting Information From the Dataset

```
print(df['Category'].unique())
print(df['Category'].value_counts())
```

```
['Data Science' 'HR' 'Advocate' 'Arts' 'Web Designing'
 'Mechanical Engineer' 'Sales' 'Health and fitness' 'Civil Engineer'
 'Java Developer' 'Business Analyst' 'SAP Developer' 'Automation Testing'
 'Electrical Engineering' 'Operations Manager' 'Python Developer'
 'DevOps Engineer' 'Network Security Engineer' 'PMO' 'Database' 'Hadoop'
 'ETL Developer' 'DotNet Developer' 'Blockchain' 'Testing']
Category
Java Developer               84
Testing                      70
DevOps Engineer              55
Python Developer             48
Web Designing                45
HR                           44
Hadoop                       42
Sales                        40
Data Science                 40
Mechanical Engineer          40
ETL Developer                40
Blockchain                   40
Operations Manager           40
Arts                         36
Database                     33
Health and fitness           30
PMO                          30
Electrical Engineering       30
Business Analyst             28
DotNet Developer             28
Automation Testing           26
Network Security Engineer    25
Civil Engineer               24
SAP Developer                24
Advocate                     20
Name: count, dtype: int64
```

3.Bar Plot Analysis For Each Category

```
import seaborn as sns
plt.figure(figsize=(40,6))
ax=sns.countplot(x="Category",data=df)
plt.grid()
```



4.Text Processing And Cleaning

[6] Generated code may be subject to a license | sunkusowmyasree/Disaster_Respone_Pipeline |

```python
import re
import string
import spacy
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem.porter import PorterStemmer
```

[7]

```python
nltk.download("stopwords")
nlp=spacy.load("en_core_web_sm")
stemmer=PorterStemmer()
def text_clearing(text):
  text=re.sub(r"http\S+"," ",text)
  text=re.sub(r"@\S+"," ",text)
  text=re.sub(r"#\S+"," ",text)
  text=re.sub(f"[{re.escape(string.punctuation)}]"," ",text)
  doc=nlp(text)
  tokens=[token.lemma_ for token in doc if token.text not in stopwords.words("english")]
  tokens=[stemmer.stem(token) for token in tokens]
  text=" ".join(tokens)
  return text
df['Resume']=df['Resume'].apply(text_clearing)
print(df['Resume'].iloc[0])
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
skill    program languag   python   panda   numpi   scipi   scikit learn   matplotlib    sql   java   javascript jqueri    machin learn   regress   svm   naã¯v bay   knn   random forest   decis tree   boost techniqu

 data scienc assur associ

 data scienc assur associ    ernst    young llp
 skill detail
 javascript   exprienc    24 month
 jqueri   exprienc    24 month
 python   exprienc    24 monthscompani detail
 compani    ernst    young llp
 descript    fraud investig disput servic    assur
 technolog assist review
 tar   technolog assist review   assist acceler review process run analyt gener report
   core member team help develop autom review platform tool scratch assist e discoveri domain   tool implement predict code topic model autom review   result reduc labor cost time spend lawyer review
   understand end end flow solut   research develop classif model   predict analysi mine inform present text datum   work analyz output precis monitor entir tool
   tar assist predict code   topic model evid follow ey standard   develop classifi model order identifi   red flag   fraud relat issu

 tool    technolog   python   scikit learn   tfidf   word2vec   doc2vec   cosin similar   naã¯v bay   lda   nmf topic model   vader text blob sentiment analysi   matplot lib   tableau dashboard report

 multipl data scienc and analyt project   usa client
 text analyt    motor vehicl custom review data    receiv custom feedback survey datum past one year   perform sentiment   posit   neg    neutral   time seri analysi custom comment across 4 categori
   creat heat map term survey categori base frequenc word    extract posit neg word across survey categori plot word cloud
   creat custom tableau dashboard effect report visual
 chatbot    develop user friendli chatbot one product handl simpl question hour oper   reserv option
   thi chat bot serv entir product relat question   give overview tool via qa platform also give recommend respons user question build chain relev answer
   thi intellig build pipelin question per user requir ask relev   recommend question

 tool    technolog   python   natur languag process   nltk   spaci   topic model   sentiment analysi   word emb   scikit learn   javascript jqueri   sqlserver

 inform govern
 organ make inform decis inform store   the integr inform govern portfolio synthes intellig across unstructur data sourc facilit action ensur organ well posit counter inform risk
   scan datum multipl sourc format pars differ file format   extract meta data inform   push result index elast search creat custom   interact dashboard use kibana
   preform rot analysi datum give inform datum help identifi content either redund   outdat   trivial
   preform full text search analysi elast search predefin method tag   pii   person identifi inform   social secur number   address   name   etc    frequent target cyber attack
 tool    technolog   python   flask   elast search   kibana

 fraud analyt platform
 fraud analyt investig platform review red flag case
 â¢ fap fraud analyt investig platform inbuilt case manag suit analyt variou erp system
   it use client interrog account system identifi anomali indic fraud run advanc analyt
 tool    technolog   html   javascript   sqlserver   jqueri   css   bootstrap   node js   d3 js   dc js
```

5.Label Encoding, Train_Test Split And Vectorization

[8]

```python
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
df['Category'] = label_encoder.fit_transform(df['Category'])
```

[9]

```python
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(df['Resume'], df['Category'], test_size=0.2, random_state=42)
```

```python
print("x_train size -- >> " , x_train.shape)
print("y_train size -- >> " , y_train.shape)
print("x_test size -- >> " , x_test.shape)
print("y_test size -- >> " , y_test.shape)
```

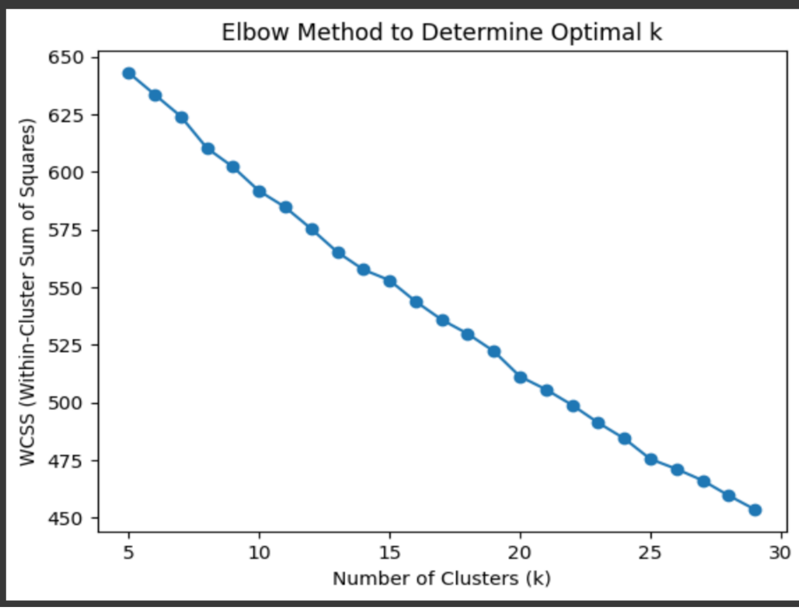
```
x_train size -- >>  (769,)
y_train size -- >>  (769,)
x_test size -- >>  (193,)
y_test size -- >>  (193,)
```

[11] Generated code may be subject to a license | AskMe-Phone/Askme-Backend

```python
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
x_train = vectorizer.fit_transform(x_train)
x_test = vectorizer.transform(x_test)
```

6.Finding Number Of Clusters By Elbow Method

```
from sklearn.cluster import KMeans
wcss = []
K_range = range(5, 30)
for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(x_train)
    wcss.append(kmeans.inertia_)
plt.plot(K_range, wcss, marker="o")
plt.xlabel("Number of Clusters (k)")
plt.ylabel("WCSS (Within-Cluster Sum of Squares)")
plt.title("Elbow Method to Determine Optimal k")
plt.show()
```



7.K-Means Clustering Algorithm And Che

```
[13] optimal_k = 25
     kmeans = KMeans(n_clusters=optimal_k, random_state=42)
     kmeans.fit(x_train)
     x_predict= kmeans.fit_predict(x_test)
     print(x_predict)
     all_data_tfidf = vectorizer.transform(df['Resume'])
     df['cluster'] = kmeans.predict(all_data_tfidf)
```

```
[15 15 15 17 23  3  7  6 24 23 17  9 16  5 10  4 13 13 13  0  7 23 12  6
  5 20  1  7  6  9  7 23  3 15 19  2 14  9  0  3 22 13 15 13  0  6 20  7
  7 13 15  7 19  4 15 17  6  7  4  9  1 15 12 15 20  7  7 12 12 12 12 15
  0 20 14 20 24  3 16 23  8 19 19 18 10 11 13  3 24 13  0 12  6  2 10 10
 21 15  7  9 14 16  3  7  7 11 23  7 23 23  7  1 18 13 20  1  2 13  2  4
 19 15  9 11  4 11 14  9  7 18 10  7 11 15  0 13 16 21 20  4 10 15 16 14
 14 18 15  1 23 15 17 10 14 13  7 19  4 15  4  9  0 15  7 20  3  2 13  9
 15 16  8  7  3  7 11  7 21  5  9 14 16 22  4 11 22 15  6  6 10 20 15 14
 19]
```

8.Checking Accuracy Score

```
y_pred = kmeans.predict(x_test)
from sklearn.metrics import accuracy_score, classification_report
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.08808290155440414
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.00      0.00      0.00         6
           2       0.00      0.00      0.00         5
           3       0.00      0.00      0.00         7
           4       0.00      0.00      0.00         4
           5       0.00      0.00      0.00         9
           6       0.00      0.00      0.00         5
           7       0.00      0.00      0.00         8
           8       0.00      0.00      0.00        14
           9       0.00      0.00      0.00         5
          10       0.00      0.00      0.00         7
          11       0.00      0.00      0.00         6
          12       0.00      0.00      0.00        12
          13       0.00      0.00      0.00         4
          14       0.00      0.00      0.00         7
          15       0.52      0.73      0.61        15
          16       0.86      0.75      0.80         8
          17       0.00      0.00      0.00         3
          18       0.00      0.00      0.00        12
          19       0.00      0.00      0.00         7
          20       0.00      0.00      0.00        10
          21       0.00      0.00      0.00         7
          22       0.00      0.00      0.00         8
          23       0.00      0.00      0.00        16
          24       0.00      0.00      0.00         5

    accuracy                           0.09       193
   macro avg       0.06      0.06      0.06       193
weighted avg       0.08      0.09      0.08       193
```

8.KNN Algorithm And Checking Accuracy Score

Generated code may be subject to a license | sivareddy101/SmartCityParkingMgmt |

```
from sklearn.neighbors import KNeighborsClassifier
knn =KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train, y_train)
y_pred1 = knn.predict(x_test)
from sklearn.metrics import accuracy_score, classification_report
accuracy = accuracy_score(y_test, y_pred1)
print("Accuracy:", accuracy)
print(classification_report(y_test, y_pred1))
```

```
Accuracy: 0.9844559585492227
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         3
           1       1.00      1.00      1.00         6
           2       1.00      1.00      1.00         5
           3       1.00      1.00      1.00         7
           4       1.00      1.00      1.00         4
           5       1.00      1.00      1.00         9
           6       1.00      0.60      0.75         5
           7       1.00      1.00      1.00         8
           8       1.00      0.93      0.96        14
           9       1.00      1.00      1.00         5
          10       1.00      1.00      1.00         7
          11       1.00      1.00      1.00         6
          12       1.00      1.00      1.00        12
          13       1.00      1.00      1.00         4
          14       1.00      1.00      1.00         7
          15       1.00      1.00      1.00        15
          16       1.00      1.00      1.00         8
          17       1.00      1.00      1.00         3
          18       1.00      1.00      1.00        12
          19       0.88      1.00      0.93         7
          20       1.00      1.00      1.00        10
          21       0.78      1.00      0.88         7
          22       1.00      1.00      1.00         8
          23       1.00      1.00      1.00        16
          24       1.00      1.00      1.00         5

    accuracy                           0.98       193
   macro avg       0.99      0.98      0.98       193
weighted avg       0.99      0.98      0.98       193
```

9. Logistic Regression And Checking Accuracy Score

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(x_train, y_train)
y_pred2 = lr.predict(x_test)
from sklearn.metrics import accuracy_score, classification_report
accuracy = accuracy_score(y_test, y_pred2)
print("Accuracy:", accuracy)
print(classification_report(y_test, y_pred2))
```

```
Accuracy: 0.9948186528497409
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         3
           1       1.00      1.00      1.00         6
           2       1.00      1.00      1.00         5
           3       1.00      1.00      1.00         7
           4       1.00      1.00      1.00         4
           5       1.00      1.00      1.00         9
           6       1.00      1.00      1.00         5
           7       1.00      1.00      1.00         8
           8       1.00      0.93      0.96        14
           9       1.00      1.00      1.00         5
          10       1.00      1.00      1.00         7
          11       1.00      1.00      1.00         6
          12       1.00      1.00      1.00        12
          13       1.00      1.00      1.00         4
          14       1.00      1.00      1.00         7
          15       1.00      1.00      1.00        15
          16       1.00      1.00      1.00         8
          17       1.00      1.00      1.00         3
          18       1.00      1.00      1.00        12
          19       0.88      1.00      0.93         7
          20       1.00      1.00      1.00        10
          21       1.00      1.00      1.00         7
          22       1.00      1.00      1.00         8
          23       1.00      1.00      1.00        16
          24       1.00      1.00      1.00         5

    accuracy                           0.99       193
   macro avg       0.99      1.00      1.00       193
weighted avg       1.00      0.99      0.99       193
```

9. Assigning final weightage score to each resume from 0 to 10

```python
from sklearn.preprocessing import MinMaxScaler
import numpy as np
def process_user_resume(text):
    text = text_clearing(text)
    text_tfidf = vectorizer.transform([text])
    return text_tfidf


user_resume = input("Enter the resume text: ")

processed_resume = process_user_resume(user_resume)
predicted_category = lr.predict(processed_resume)[0]

confidence_score = np.max(lr.predict_proba(processed_resume))

final_score = confidence_score * 10
category_name = label_encoder.inverse_transform([predicted_category])[0]


print(f"\nPredicted Category: {category_name}")
print(f"Weightage Score (0-10): {final_score:.2f}")
```

```
Enter the resume text: Areas of Interest Deep Learning, Control System Design, Programming in-Python, Electric Machinery, Web Development, Analyt

Predicted Category: Data Science
Weightage Score (0-10): 5.61
```

Future Scope

1. **Improving Accuracy** – Incorporate **deep learning models** like BERT for better resume classification.
2. **Enhanced Skill Matching** – Use **semantic similarity** and **knowledge graphs** to match skills more effectively.
3. **Real-Time Screening** – Develop a **web-based API** for instant resume parsing and integration with **ATS systems**.
4. **Personalized Job Matching** – Implement **AI-driven recommendations** based on past hiring trends.
5. **Multi-Language Support** – Extend to **multilingual NLP models** for parsing resumes in various languages.
6. **Bias Reduction** – Use **fair AI techniques** to ensure unbiased screening.
7. **Social Media Integration** – Analyze **LinkedIn, GitHub, and Kaggle** profiles for a more complete candidate assessment.

Conclusion

The **Resume Parser AI** project successfully streamlines the recruitment process by leveraging **Natural Language Processing (NLP)** and **Machine Learning** for automated resume screening. By categorizing resumes and assigning relevance scores, the model significantly reduces manual effort, making candidate selection more efficient.

While **Logistic Regression** proved to be the most effective classification model, **K-Means Clustering** showed limitations in accuracy. The system effectively ranks resumes based on **skills, experience, and job relevance**, providing recruiters with a data-driven approach to hiring.

In the future, enhancements such as **deep learning integration, multi-language support, and real-time API deployment** can further improve the system's accuracy and usability. With continuous advancements, the Resume Parser AI has the potential to become an essential tool in **modern recruitment workflows**.