Resume Selection with ML

A MINI PROJECT REPORT

Submitted by

PRAGYA AGARWAL [RA2011003010603]
DEBANJAN BASAK [RA2011003010606]
NISHANT JAIN [RA2011003010610]

Under the guidance of

Dr. ABIRAMI G.
(Designation, Dept name)

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING
With specialization in Computing Technologies

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

APRIL 2023

INDEX

ABSTRACT

Finding suitable candidates for an open role could be a daunting task, especially when there are many applicants. It can impede team progress for getting the right person on the right time. An automated way of “Resume Classification and Matching” could really ease the tedious process of fair screening and shortlisting, it would certainly expedite the candidate selection and decision-making process. This system could work with a large number of resumes for first classifying the right categories using different classifier, once classification has been done then as per the job description, top candidates could be ranked using Content-based Recommendation, using cosine similarity and by using k-NN to identify the CVs that are nearest to the provided job description.

<u>INTRODUCTION</u>

Talent acquisition is an important, complex, and time-consuming function within Human Resources (HR). The sheer scale of India’s market is overwhelming. Not only is there a staggering one million people coming into the job market every month, but there is also huge turnover. As per LinkedIn, India has the highest percentage of the workforce that is “actively seeking a new job”. Clearly, this is an extremely liquid, massive market but one that also has many frustrating inefficiencies.

The most challenging part is the lack of a standard structure and format for resume which makes short listing of desired profiles for required roles very tedious and time-consuming.
Effective screening of resumes requires domain knowledge, to be able to understand the relevance and applicability of a profile for the job role. With a huge number of different job roles existing today along with the typically large number of applications received, short-listing poses a challenge for the human resource department. Which is only further worsened by the lack of diverse skill and domain knowledge within the HR department, required for effective screening.

Today the industry face three major challenges:

- Separating right candidates from the pack
- Making sense of candidate CVs
- Knowing that candidates can do the job before you hire them.

To overcome the mentioned issues in the resume short-listing process, in this paper we present an automated Machine Learning based model.

The model takes the features extracted from the candidate’s resume as input and finds their categories, further based on the required job description the categorized resume mapped and recommends the most suitable candidate’s profile to HR.

METHODOLOGY/ TECHNIQUES

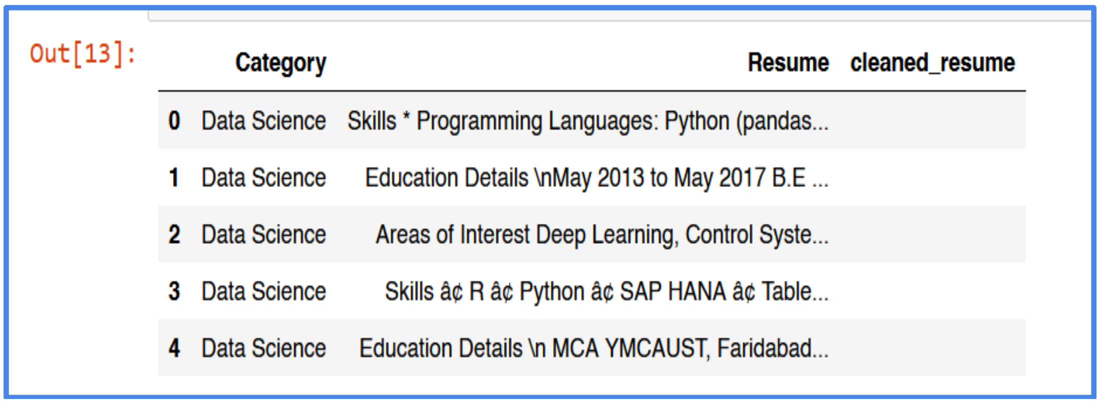
Our main contributions are listed below:

1. We developed an automated resume recommendation system.
2. Machine learning based classification techniques with similarity functions are used to find most relevant resume.
3. Linear SVM classifier performed best for our case compared to another ML classifiers.

The proposed system will use a supervised learning approach, where a dataset of resumes and job descriptions will be used to train a classification model. The dataset will be preprocessed to extract relevant features, such as skills, experience, and education, from each resume. The model will then be trained on this dataset using various machine learning algorithms, such as logistic regression, decision trees, and random forests. The performance of each model will be evaluated using metrics such as accuracy and precision.

CODE AND OUTPUT:

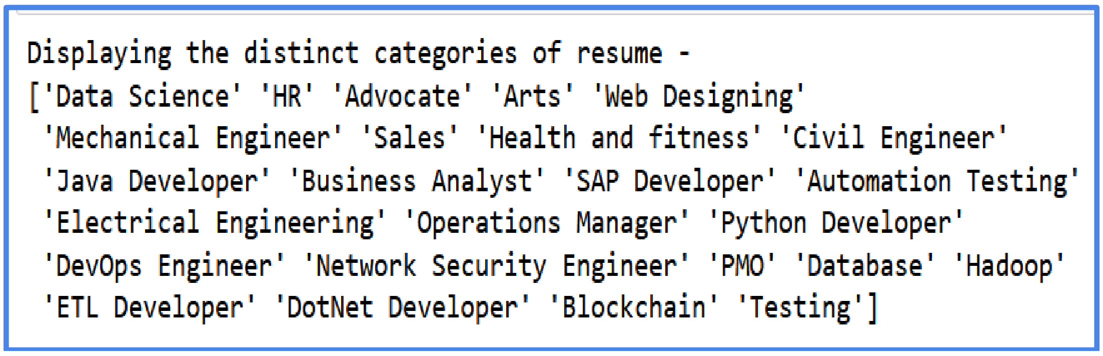
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from sklearn.naive_bayes import MultinomialNB
from sklearn.multiclass import OneVsRestClassifier
from sklearn import metrics
from sklearn.metrics import accuracy_score
from pandas.plotting import scatter_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
resumeDataSet = pd.read_csv('resume_dataset.csv' ,encoding='utf-8')
resumeDataSet['cleaned_resume'] = ''
resumeDataSet.head()
```



Out[13]:

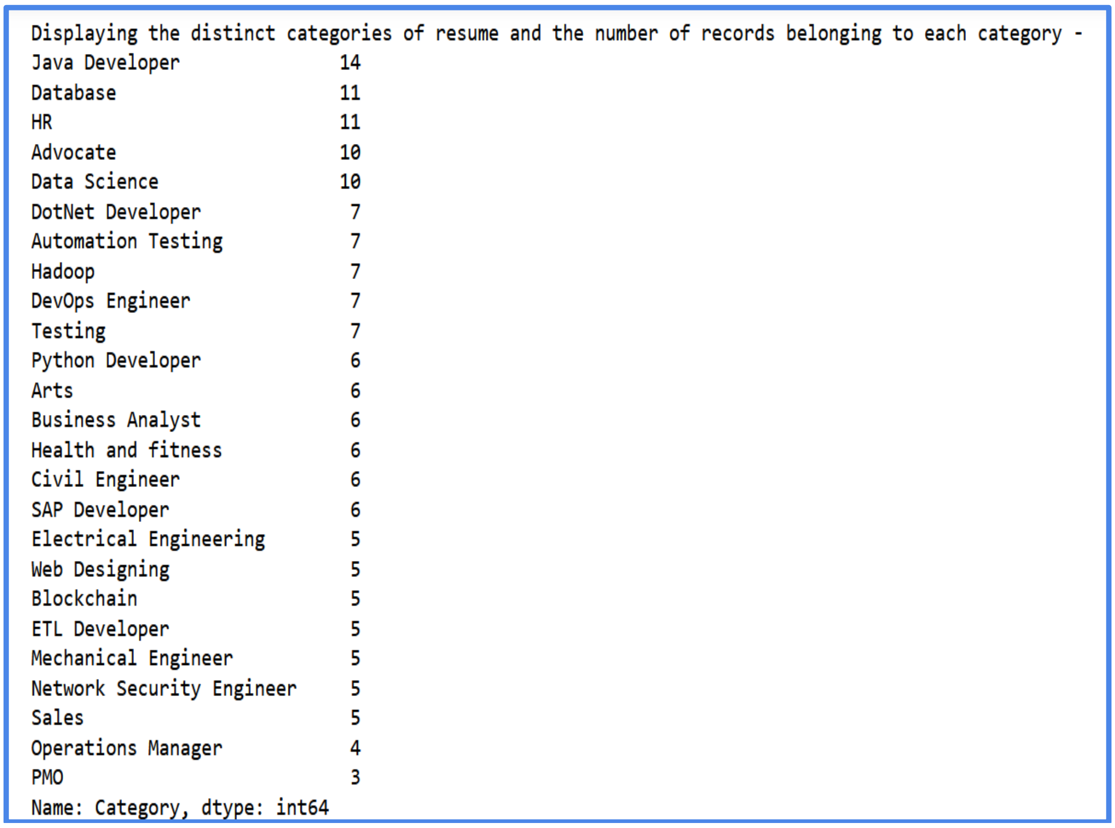
	Category	Resume	cleaned_resume
0	Data Science	Skills * Programming Languages: Python (pandas...	
1	Data Science	Education Details \nMay 2013 to May 2017 B.E ...	
2	Data Science	Areas of Interest Deep Learning, Control Syste...	
3	Data Science	Skills â¢ R â¢ Python â¢ SAP HANA â¢ Table...	
4	Data Science	Education Details \n MCA YMCAUST, Faridabad...	

```
print ("Displaying the distinct categories of resume -")
print (resumeDataSet['Category'].unique())
```



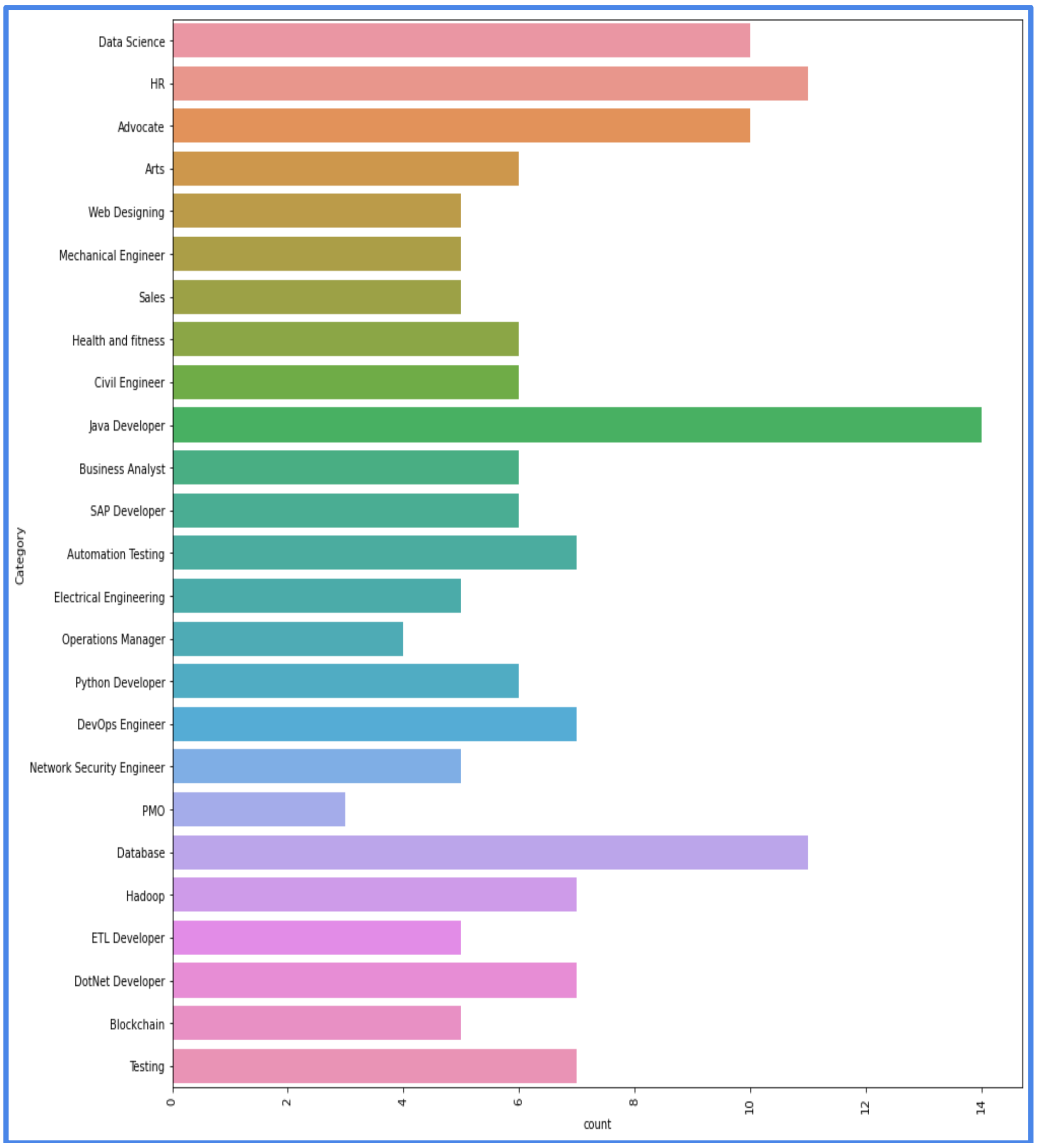
```
Displaying the distinct categories of resume -
['Data Science' 'HR' 'Advocate' 'Arts' 'Web Designing'
 'Mechanical Engineer' 'Sales' 'Health and fitness' 'Civil Engineer'
 'Java Developer' 'Business Analyst' 'SAP Developer' 'Automation Testing'
 'Electrical Engineering' 'Operations Manager' 'Python Developer'
 'DevOps Engineer' 'Network Security Engineer' 'PMO' 'Database' 'Hadoop'
 'ETL Developer' 'DotNet Developer' 'Blockchain' 'Testing']
```

```
print ("Displaying the distinct categories of resume and the number of records belonging to each category -")
print (resumeDataSet['Category'].value_counts())
```



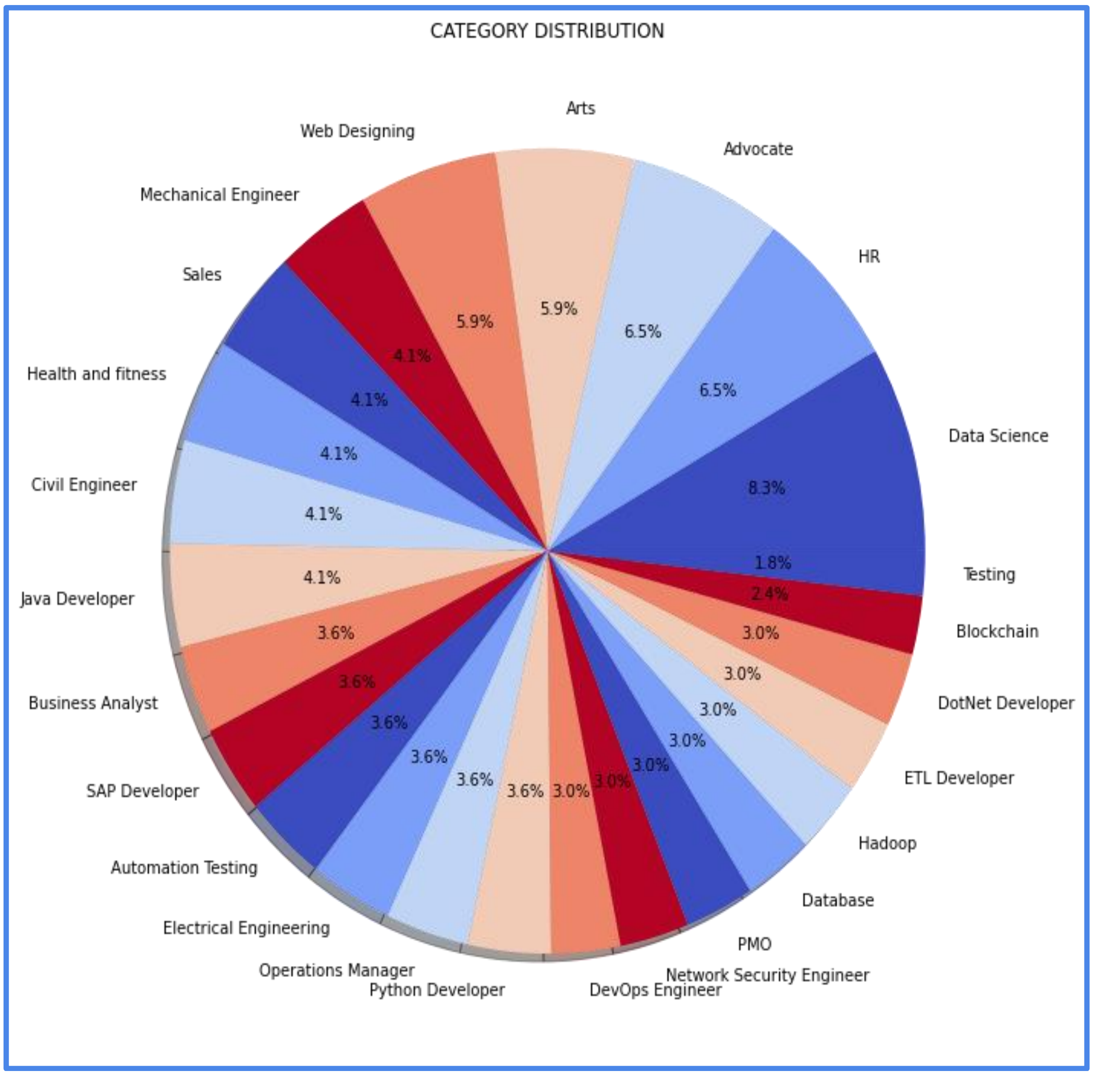
```
Displaying the distinct categories of resume and the number of records belonging to each category -
Java Developer               14
Database                     11
HR                           11
Advocate                     10
Data Science                 10
DotNet Developer              7
Automation Testing            7
Hadoop                        7
DevOps Engineer               7
Testing                       7
Python Developer              6
Arts                          6
Business Analyst              6
Health and fitness            6
Civil Engineer                6
SAP Developer                 6
Electrical Engineering        5
Web Designing                 5
Blockchain                    5
ETL Developer                 5
Mechanical Engineer           5
Network Security Engineer     5
Sales                         5
Operations Manager            4
PMO                           3
Name: Category, dtype: int64
```

```
import seaborn as sns
plt.figure(figsize=(15,15))
plt.xticks(rotation=90)
sns.countplot(y="Category", data=resumeDataSet)
```



```
from matplotlib.gridspec import GridSpec
targetCounts = resumeDataSet['Category'].value_counts()
targetLabels  = resumeDataSet['Category'].unique()
# Make square figures and axes
plt.figure(1, figsize=(25,25))
the_grid = GridSpec(2, 2)


cmap = plt.get_cmap('coolwarm')
colors = [cmap(i) for i in np.linspace(0, 1, 6)]
plt.subplot(the_grid[0, 1], aspect=1, title='CATEGORY
DISTRIBUTION')
```

```
source_pie = plt.pie(targetCounts, labels=targetLabels,
autopct='%1.1f%%', shadow=True, colors=colors)
plt.show()
```

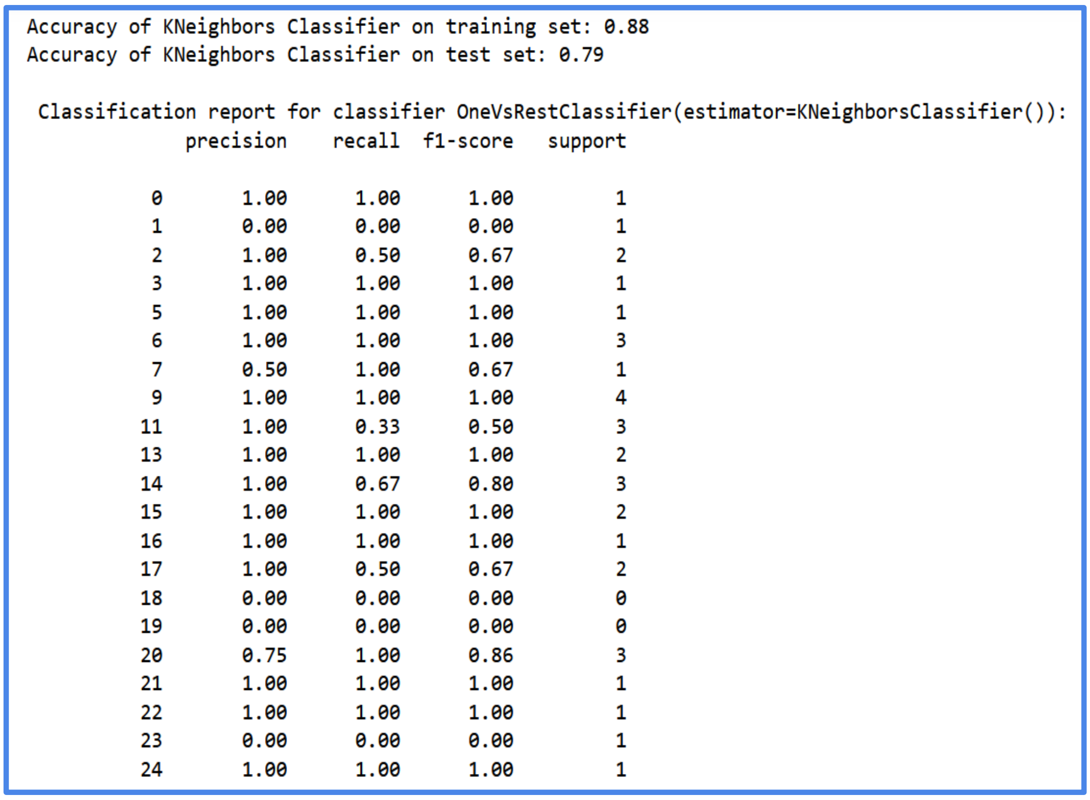


```
import re
def cleanResume(resumeText):
    resumeText = re.sub('http\S+\s*', ' ', resumeText)  # remove URLs
    resumeText = re.sub('RT|cc', ' ', resumeText)  # remove RT and cc
    resumeText = re.sub('#\S+', '', resumeText)  # remove hashtags
```

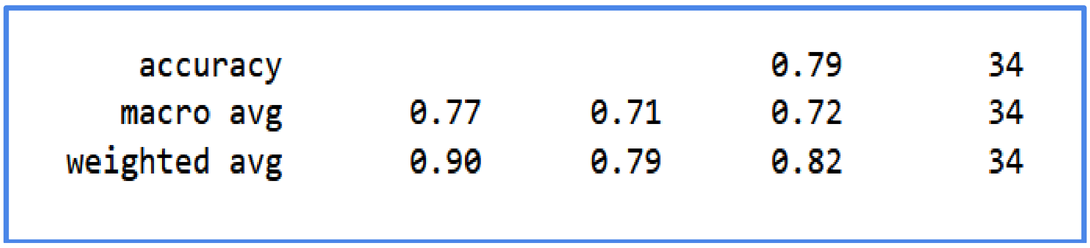
```
 resumeText = re.sub('@\S+', '  ', resumeText)  # remove mentions
    resumeText = re.sub('[%s]' % re.escape("""!"#$%&'()*+,-
./:;<=>?@[\]^_`{|}~"""), ' ', resumeText)  #



remove punctuations
    resumeText = re.sub(r'[^\x00-\x7f]',r' ', resumeText)
    resumeText = re.sub('\s+', ' ', resumeText)  # remove extra
whitespace
    return resumeText

resumeDataSet['cleaned_resume'] =
resumeDataSet.Resume.apply(lambda x: cleanResume(x))
import nltk
nltk.download('stopwords')
nltk.download('punkt')
from nltk.corpus import stopwords
import string
from wordcloud import WordCloud

oneSetOfStopWords = set(stopwords.words('english')+['``',"''"])
totalWords =[]
Sentences = resumeDataSet['Resume'].values
cleanedSentences = ""
for i in range(0,160):
    cleanedText = cleanResume(Sentences[i])
    cleanedSentences += cleanedText
    requiredWords = nltk.word_tokenize(cleanedText)
    for word in requiredWords:
        if word not in oneSetOfStopWords and word not in
string.punctuation:
            totalWords.append(word)

wordfreqdist = nltk.FreqDist(totalWords)
mostcommon = wordfreqdist.most_common(50)
print(mostcommon)
```

```
wc = WordCloud().generate(cleanedSentences)
plt.figure(figsize=(15,15))
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.show()
from sklearn.preprocessing import LabelEncoder


var_mod = ['Category']
le = LabelEncoder()
for i in var_mod:
    resumeDataSet[i] = le.fit_transform(resumeDataSet[i])
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from scipy.sparse import hstack

requiredText = resumeDataSet['cleaned_resume'].values
requiredTarget = resumeDataSet['Category'].values

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

print ("Feature completed .....")

X_train,X_test,y_train,y_test =
train_test_split(WordFeatures,requiredTarget,random_state=0,
test_size=0.2)
print(X_train.shape)
print(X_test.shape)

clf = OneVsRestClassifier(KNeighborsClassifier())
clf.fit(X_train, y_train)
```

```
prediction = clf.predict(X_test)
print('Accuracy of KNeighbors Classifier on training set:
{:.2f}'.format(clf.score(X_train, y_train)))
print('Accuracy of KNeighbors Classifier on test set:
{:.2f}'.format(clf.score(X_test, y_test)))

print("\n Classification report for classifier %s:\n%s\n" % (clf,
metrics.classification_report(y_test, prediction)))
```



```
Accuracy of KNeighbors Classifier on training set: 0.88
Accuracy of KNeighbors Classifier on test set: 0.79

 Classification report for classifier OneVsRestClassifier(estimator=KNeighborsClassifier()):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       0.00      0.00      0.00         1
           2       1.00      0.50      0.67         2
           3       1.00      1.00      1.00         1
           5       1.00      1.00      1.00         1
           6       1.00      1.00      1.00         3
           7       0.50      1.00      0.67         1
           9       1.00      1.00      1.00         4
          11       1.00      0.33      0.50         3
          13       1.00      1.00      1.00         2
          14       1.00      0.67      0.80         3
          15       1.00      1.00      1.00         2
          16       1.00      1.00      1.00         1
          17       1.00      0.50      0.67         2
          18       0.00      0.00      0.00         0
          19       0.00      0.00      0.00         0
          20       0.75      1.00      0.86         3
          21       1.00      1.00      1.00         1
          22       1.00      1.00      1.00         1
          23       0.00      0.00      0.00         1
          24       1.00      1.00      1.00         1
```



```
    accuracy                           0.79        34
   macro avg       0.77      0.71      0.72        34
weighted avg       0.90      0.79      0.82        34
```

IMPLEMENTATION & RESULT

1. **Data Collection:** To build the resume selection system, we need a large dataset of resumes and job descriptions. We will collect this data from various sources, such as online job boards, career websites, and social media platforms. The data will be preprocessed to remove any irrelevant or duplicate entries and ensure that it is clean and ready for analysis.

2. **Feature Extraction:** To train the machine learning model, we need to extract relevant features from each resume and job description. These features could include skills, experience, education, certifications, and any other information that is relevant to the job. We will use various NLP techniques, such as named entity recognition, part-of-speech tagging, and sentiment analysis, to extract these features from the text.

3. **Model Training:** Once we have extracted the relevant features, we will use supervised learning algorithms to train a classification model. We will use a variety of algorithms, such as logistic regression, decision trees, and random forests, and compare their performance on evaluation metrics such as accuracy & precision. We will also use techniques such as cross-validation and hyperparameter tuning to ensure that the model is robust and generalizes well to new data.

4. **Model Integration:** Once we have selected the best-performing model, we will integrate it into a web-based application that allows recruiters to upload job descriptions and resumes and get an automated recommendation on which candidates to consider for the position. The system will be designed to be user-friendly and intuitive, with clear instructions and feedback to guide the recruiter through the selection process.

5. **Evaluation:** We will evaluate the performance of the system using a real-world dataset of resumes and job openings. We will compare its performance to the traditional resume selection process to assess its effectiveness in reducing bias and improving the quality of the candidate selection process. We will also conduct user surveys and interviews to gather feedback on the usability and effectiveness of the system and identify any areas for improvement.

CONCLUSION

The final model will be selected based on its performance on the evaluation metrics. The selected model will be integrated into a web-based application that allows recruiters to upload job descriptions and resumes and get an automated recommendation on which candidates to consider for the position. The system will be tested using a real-world dataset of resumes and job openings, and its performance will be compared to the traditional resume selection process to assess its effectiveness in reducing bias and improving the quality of the candidate selection process.

REFERENCES

- https://www.kaggle.com/code/gauravduttakiit/resume-screening-using-machine-learning
- https://www.analyticsvidhya.com/blog/2021/06/resume-screening-with-natural-language-processing-in-python/
- https://github.com/milaan9/93_Python_Data_Analytics_Projects/tree/main/004_Resume_Selection_with_ML