A Machine Learning Model for

Resume Screening

By
Jyothi Yanapu
194287, ECE 2nd year
&
Mahaveer Kamuju
191128,Civil 2nd year

What is Resume Screening?

Resume screening is the process of determining whether a candidate is qualified for a role based on his/ her education,
experience and other information captured on their resume .

Need of Resume Screening

It becomes very difficult for the hiring teams to read the resume and select the resume according to the requirement, there is no problem if there are one or two resumes but it is very difficult to go through many resumes and select the best one.

To solve this problem, we need our machines to work for us and screen the resume.

Problem statement

Develop a machine learning model for screening resumes of different categories

This model consists of 3 parts

Part 1: Data visualization
Part 2: Data Preprocessing
Part 3: Training the Machine learning Model

Part 1: Data visualization

Let us understand how the raw data is…

Step 0: Exploring the Dataset

https://www.kaggle.com/gauravduttakiit/resume-dataset

The data set has 25 categories of resumes and each of them has their keywords

Step 1: Import the necessary libraries

```
import numpy as np # It contains multi dimensional arrays and matrices
import pandas as pd # for data storing
import matplotlib.pyplot as plt # for plotting we use this
import warnings
warnings.filterwarnings('ignore')# we are ignoring warning messages
from sklearn.naive_bayes import MultinomialNB
# helps in classifind discrete features
from sklearn.multiclass import OneVsRestClassifier
 # to classify the positive and negative samples regarding the category
from sklearn import metrics
# metrics is used for comparison and  tracking performing and production
# sklearn supports classification,regression  , naive bayes and many other features
from sklearn.metrics import accuracy_score
#set of labels predicted for a sample must exactly match the corresponding set of ... contd
#...labels in y and how accurate it is we can get.
from pandas.plotting import scatter_matrix
#to draw scatter plots
from sklearn.neighbors import KNeighborsClassifier
# his classifier implements learning based on the k nearest neighbors.
```

Step 2: Import the dataset and printing the data

```
resumeDataSet = pd.read_csv('UpdatedResumeDataSet.csv' ,encoding='utf-8')
#UTF-8 is one of the most commonly used encodings, and Python often defaults to using it.
```

	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...

Step 3: Printing the Categories in the Dataset

```
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']
```

Step 4: Printing the Categories and no. of records in them

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

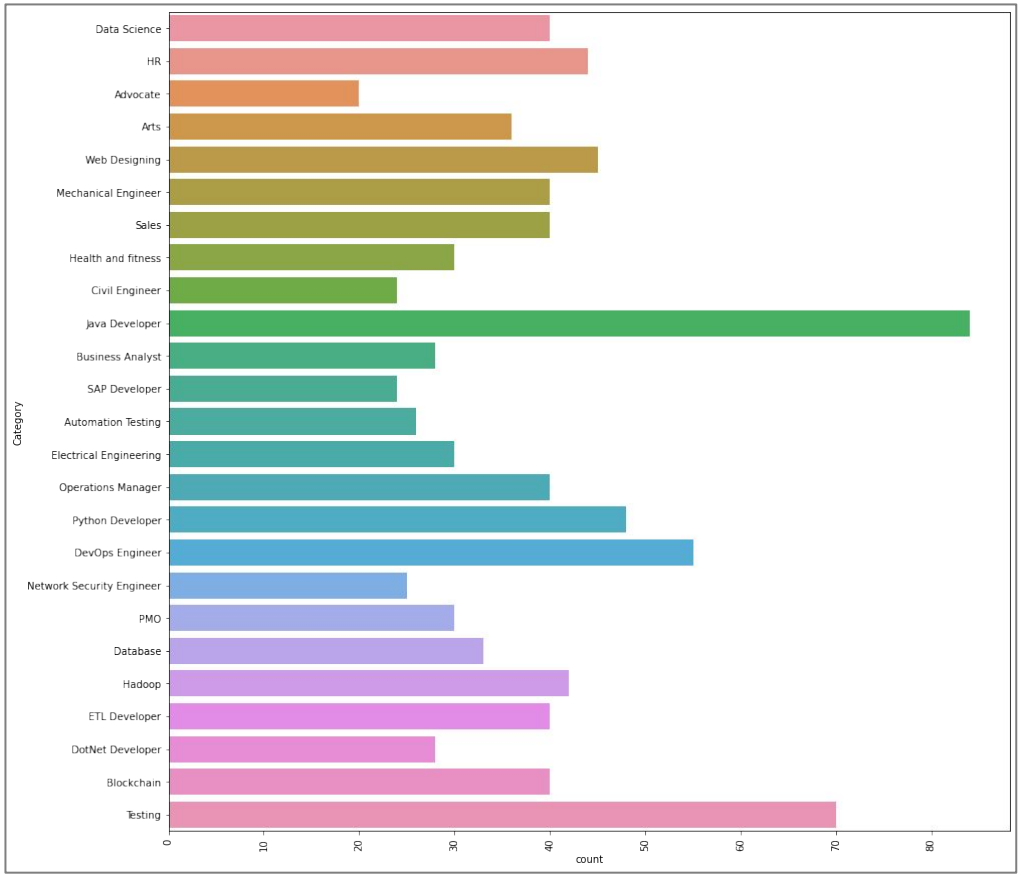
Step 5:Visualising the Categories in a Count graph

```
import seaborn as sns
#Seaborn is a Python data visualization library based on matplotlib.
plt.figure(figsize=(15,15))
#figsize is a tuple of the width and height of the figure in inches
plt.xticks(rotation=90)

# here we used xtick function such that x label had rotated 90 degree.


sns.countplot(y="Category", data=resumeDataSet)
```

The following is the output in Step 5



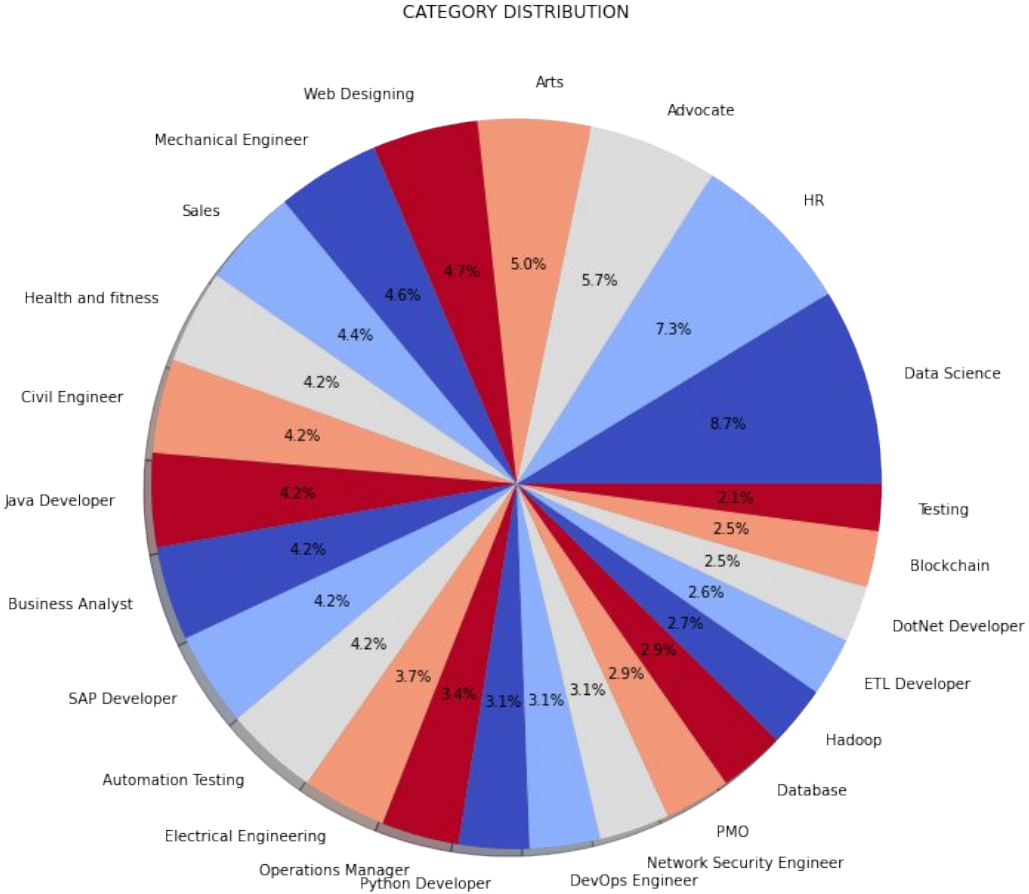
Step 6:Visualising the percentages of each Category in a Pie chart

```
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, 5)]
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()
```

The following is the output in Step 6



From step 0 – step 6
The Data visualization is done

Part 2: Data Preprocessing

In order to let computer understand the data ,
We have to first refine the data ,perform required operations

Step 7: Adding a new column “Cleaned resume” in the dataset and printing it

```
resumeDataSet['cleaned_resume'] = ''
# we are defining cleaned_Resume as blank at present.
resumeDataSet.head()
```

	Category	Resume	cleaned_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...	

Step 8: Removing the unnecessary words and characters in the resume column and storing the data in new cleaned resume column

```
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))
```

Step 9: Importing the NLTK libraries and downloading the required functions for cleaning the data

```
import nltk
#natural language tool kit,it is one of the nlp library.
#which contains packages to make machines understand human language and reply to it with an appropriate response.
nltk.download('stopwords')
# stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignor
nltk.download('punkt')
#punkt is a tokenizer

from nltk.corpus import stopwords
#corpus is collection of texts,
#corpus package automatically creates a set of corpus reader instances that can be used to access the corpora in the NLTK data package.
import string
from wordcloud import WordCloud
#. Significant textual data points can be highlighted using a word cloud.
```

Step 10:Cleaning the data and visualizing the word frequency in the data

```
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=(25,25))
plt.imshow(wc, interpolation='bilinear')
#Interpolation is the process of estimating unknown values that fall between known values.
plt.axis("off")
plt.show()
```

The following is the output in Step 9

Step 11: Encoding the Category Column in the dataset

```
from sklearn.preprocessing import LabelEncoder
#Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form.

var_mod = ['Category']
#encode the 'Category' column using LabelEncoding
le = LabelEncoder()
#it sorts categories in alphabetical order and selects distinct element
for i in var_mod:
    resumeDataSet[i] = le.fit_transform(resumeDataSet[i])
    # fit_transform - Fit label encoder and return encoded labels.
```

Step 12: Converting the data into vectors

```
from sklearn.model_selection import train_test_split
#Split arrays or matrices into random train and test subsets
from sklearn.feature_extraction.text import TfidfVectorizer
#Convert a collection of raw documents to a matrix of TF-IDF(vectors for execution) features.
from scipy.sparse import hstack
#scipy.sparse data structures that enable us store large matrices with very few non-zero elements aka sparse matrices
#scipy.sparse allows us to perform complex matrix computations.
#hstack is used to Stack sparse matrices horizontally (column wise)
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)
```

From step 7 – step 11
The Data Preprocessing is done

Part 3: Training the Machine learning Model

Step 13: Splitting The Data For Training And Testing

```
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)
```

```
Feature completed .....
(769, 1500)
(193, 1500)
```

Step 14: Classifying the Data using KNN classifier, screening the test samples in the data set and printing the classification report

```
clf = OneVsRestClassifier(KNeighborsClassifier())
#KNeighborsClassifier implements classification based on voting by nearest k-neighbors of target point
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)))
```

The following is the output in Step 14

```
Accuracy of KNeighbors Classifier on training set: 0.99
Accuracy of KNeighbors Classifier on test set: 0.99

 Classification report for classifier OneVsRestClassifier(estimator=KNeighborsClassifier(algorithm='auto',
                                                          leaf_size=30,
                                                          metric='minkowski',
                                                          metric_params=None,
                                                          n_jobs=None, n_neighbors=5,
                                                          p=2, weights='uniform'),
                    n_jobs=None):
```

The following is the output in Step 14

….continued

	precision	recall	f1-score	support
0	1.00	1.00	1.00	3
1	1.00	1.00	1.00	3
2	1.00	0.80	0.89	5
3	1.00	1.00	1.00	9
4	1.00	1.00	1.00	6
5	0.83	1.00	0.91	5
6	1.00	1.00	1.00	9
7	1.00	1.00	1.00	7
8	1.00	0.91	0.95	11
9	1.00	1.00	1.00	9
10	1.00	1.00	1.00	8
11	0.90	1.00	0.95	9
12	1.00	1.00	1.00	5
13	1.00	1.00	1.00	9
14	1.00	1.00	1.00	7
15	1.00	1.00	1.00	19
16	1.00	1.00	1.00	3
17	1.00	1.00	1.00	4
18	1.00	1.00	1.00	5
19	1.00	1.00	1.00	6
20	1.00	1.00	1.00	11
21	1.00	1.00	1.00	4
22	1.00	1.00	1.00	13
23	1.00	1.00	1.00	15
24	1.00	1.00	1.00	8
accuracy			0.99	193
macro avg	0.99	0.99	0.99	193
weighted avg	0.99	0.99	0.99	193

Step 15: Classifying the Data using SVM classifier, screening the test samples in the data set and printing the classification report

```
from sklearn import svm
sclf = OneVsRestClassifier(svm.SVC())
#Support vector machines (SVMs) are particular linear classifiers which are based on the margin maximization principle.
#They perform structural risk minimization, which improves the complexity of the classifier ...
# ...with the aim of achieving excellent generalization performance.
sclf.fit(X_train, y_train)
prediction = sclf.predict(X_test)
print('Accuracy of SVM Classifier on training set: {:.2f}'.format(sclf.score(X_train, y_train)))
print('Accuracy of SVM Classifier on test set: {:.2f}'.format(sclf.score(X_test, y_test)))
print("\n Classification report for classifier %s:\n%s\n" % (sclf, metrics.classification_report(y_test, prediction)))
```

The following is the output in Step 15

```
Accuracy of SVM Classifier on training set: 1.00
Accuracy of SVM Classifier on test set: 0.99

 Classification report for classifier OneVsRestClassifier(estimator=SVC(C=1.0, break_ties=False, cache_size=200,
                                    class_weight=None, coef0=0.0,
                                    decision_function_shape='ovr', degree=3,
                                    gamma='scale', kernel='rbf', max_iter=-1,
                                    probability=False, random_state=None,
                                    shrinking=True, tol=0.001, verbose=False),
                    n_jobs=None):
```

The following is the output in Step 15

….continued

	precision	recall	f1-score	support
0	1.00	1.00	1.00	3
1	1.00	1.00	1.00	3
2	1.00	1.00	1.00	5
3	1.00	1.00	1.00	9
4	1.00	1.00	1.00	6
5	1.00	1.00	1.00	5
6	1.00	1.00	1.00	9
7	1.00	1.00	1.00	7
8	1.00	0.91	0.95	11
9	1.00	1.00	1.00	9
10	1.00	1.00	1.00	8
11	1.00	1.00	1.00	9
12	1.00	1.00	1.00	5
13	1.00	1.00	1.00	9
14	1.00	1.00	1.00	7
15	0.95	1.00	0.97	19
16	1.00	1.00	1.00	3
17	1.00	1.00	1.00	4
18	1.00	1.00	1.00	5
19	1.00	1.00	1.00	6
20	1.00	1.00	1.00	11
21	1.00	1.00	1.00	4
22	1.00	1.00	1.00	13
23	1.00	1.00	1.00	15
24	1.00	1.00	1.00	8
accuracy			0.99	193
macro avg	1.00	1.00	1.00	193
weighted avg	1.00	0.99	0.99	193

Step 16: Printing the Index for user convenience

```
print("where")
index=list(range(0,1000))
for i in index:
    try:
        print(i,'  ',le.classes_[i],)
    except:
        break
```

Output:

```
where
0    Advocate
1    Arts
2    Automation Testing
3    Blockchain
4    Business Analyst
5    Civil Engineer
6    Data Science
7    Database
8    DevOps Engineer
9    DotNet Developer
10    ETL Developer
11    Electrical Engineering
12    HR
13    Hadoop
14    Health and fitness
15    Java Developer
16    Mechanical Engineer
17    Network Security Engineer
18    Operations Manager
19    PMO
20    Python Developer
21    SAP Developer
22    Sales
23    Testing
24    Web Designing
```

After step 13,14,15,16
This model can be used for
classifying resumes

Execution video

Resources

https://www.kaggle.com/gauravduttakiit/resume-dataset

https://www.sciencedirect.com/science/article/pii/S187705092030750X/pdf?md5=f108828b0945aff4e4df03811ce3e57e&pid=1-s2.0-S187705092030750X-main.pdf

https://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors-classification

https://www.nltk.org/

https://research.google.com/colaboratory/