resume-classification

January 23, 2024

[1]:
```
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
```

[2]:
```
resumeDataSet = pd.read_csv('/content/resume_dataset.csv' ,encoding='utf-8')
resumeDataSet['cleaned_resume'] = ''
resumeDataSet.head()
```

[2]:
```
        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…

  cleaned_resume
0
1
2
3
4
```

[3]:
```
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'
```

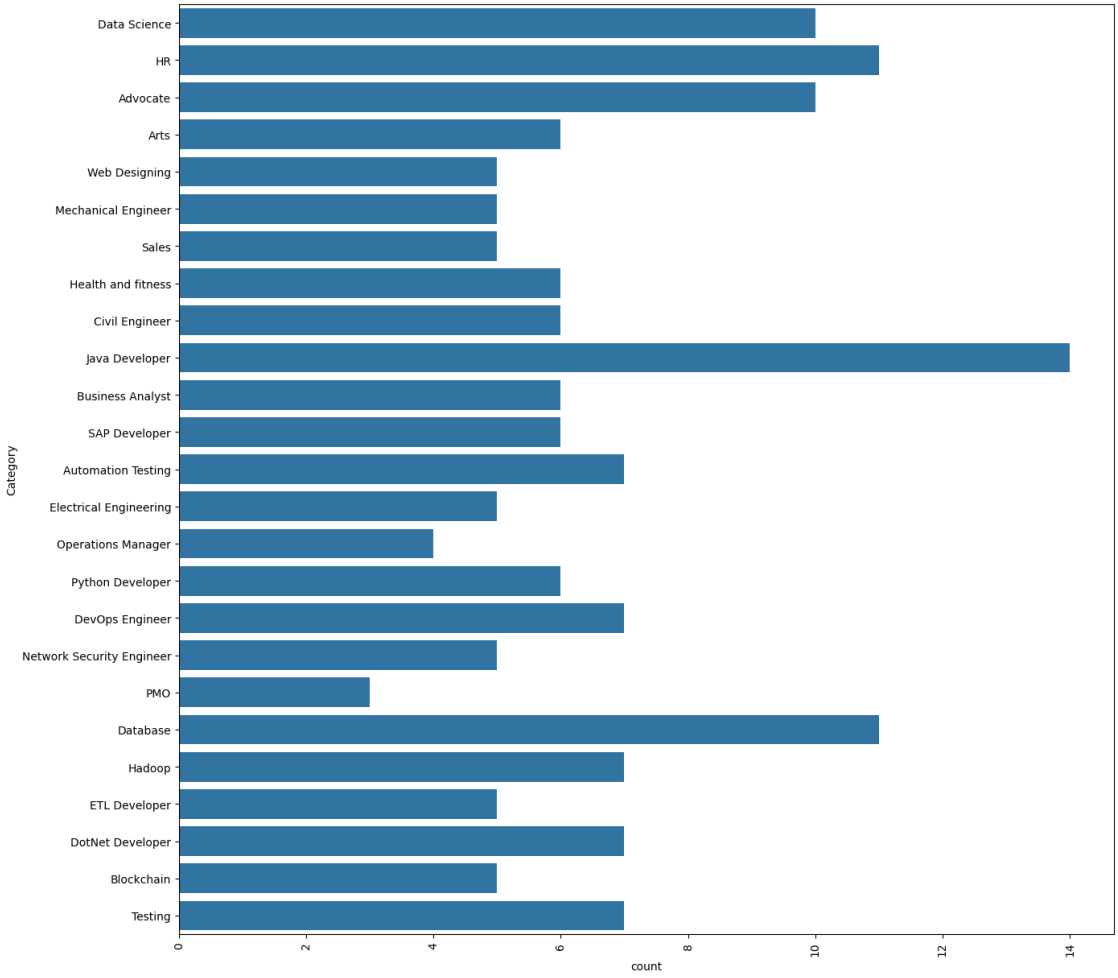
```
 'Electrical Engineering' 'Operations Manager' 'Python Developer'
 'DevOps Engineer' 'Network Security Engineer' 'PMO' 'Database' 'Hadoop'
 'ETL Developer' 'DotNet Developer' 'Blockchain' 'Testing']
```

[4]:
```
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
Data Science                  10
Advocate                      10
DotNet Developer               7
Hadoop                         7
DevOps Engineer                7
Automation Testing             7
Testing                        7
Civil Engineer                 6
Business Analyst               6
SAP Developer                  6
Health and fitness             6
Python Developer               6
Arts                           6
Electrical Engineering         5
Sales                          5
Network Security Engineer      5
Mechanical Engineer            5
Web Designing                  5
ETL Developer                  5
Blockchain                     5
Operations Manager             4
PMO                            3
Name: Category, dtype: int64
```

[5]:
```
import seaborn as sns
plt.figure(figsize=(15,15))
plt.xticks(rotation=90)
sns.countplot(y="Category", data=resumeDataSet)
```

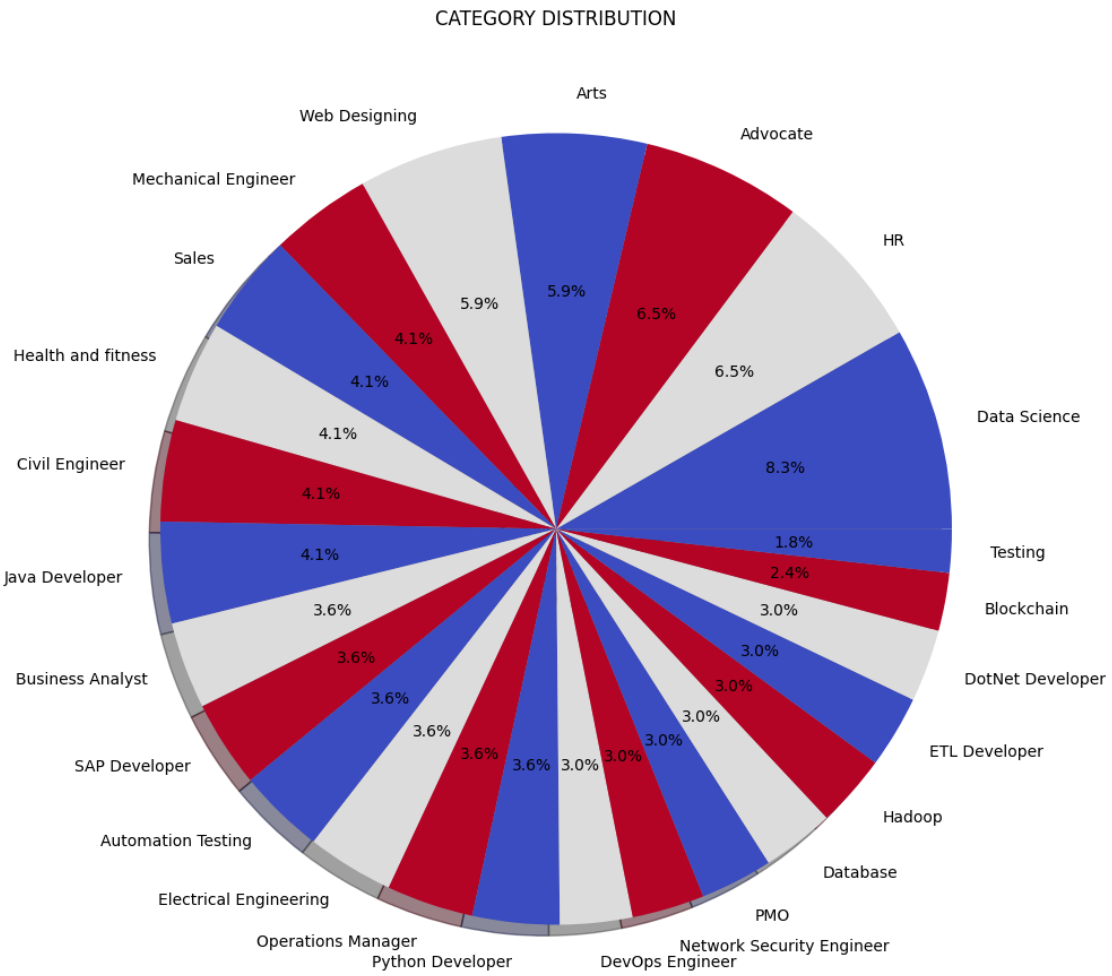
[5]:
```
<Axes: xlabel='count', ylabel='Category'>
```



[7]:
```
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, 3)]
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()
```



[8]:
```
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
```

[9]:
```
resumeDataSet['cleaned_resume'] = resumeDataSet.Resume.apply(lambda x:␣
 ↪cleanResume(x))
print (resumeDataSet['cleaned_resume'][31])
```

Good communication skill Quick learner Keen to find solutionsEducation Details MBA Marketing and International Business Management Pune Maharashtra Pune University B Tech Tech Nagpur Maharashtra M Nagpur UniversityG M Arts Commerce ScienceG M Arts Commerce ScienceSkill Details Company Details company Samarth College description of Engineering 30 7 210 5 College to campus VJ College of Pharmacy 10 days workshop 10 G M Arts Commerce Science 6 Soft Skills 6 days workshop 6 College Personality G M Institute of Agricultural 7 6 days workshop 6 Development Diploma 8 Soft Skills Samarth College of Polytechnic 20 days workshop 20 TOTAL 350 WORKING EXPERIENCE IN CORPORATE Sr No Topic Company No of days Total Hrs 1 Presentation skill Team Elringklinger Automotives Pvt 1 Day 8 building Workshop Ltd Ranjangaon Pune 2 Negotiation skill Kubler Automation Pvt Ltd 2 days 16 Communication skill Chakan Pune 3 Business Communication Finanza Home Loans Pimple 3 days 21 Stress management saudagar Pune 4 Team building Verbal Sharvari Products Pvt Ltd 2 days 16 communication Junner Pune 7 days 5 Entrepreneurship Agriculture Research Centre Workshop 168 Development Narayangaon Pune 8 batches TOTAL 229 ADJOINING SKILLS Working knowledge of Windows operating system and MS Office Communicate well in English Hindi Marathi Organized and participated in events like gathering teachers day fashion show and various science exhibitions at college

[11]:
```
import nltk
from nltk.corpus import stopwords
import string
from wordcloud import WordCloud
```

[13]:
```
from sklearn.preprocessing import LabelEncoder

var_mod = ['Category']
le = LabelEncoder()
for i in var_mod:
    resumeDataSet[i] = le.fit_transform(resumeDataSet[i])
print ("CONVERTED THE CATEGORICAL VARIABLES INTO NUMERICALS")
```

CONVERTED THE CATEGORICAL VARIABLES INTO NUMERICALS

[14]:
```
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,
    strip_accents='unicode',
    analyzer='word',
```

```
    token_pattern=r'\w{1,}',
    stop_words='english',
    ngram_range=(1, 1),
    max_features=2000)
word_vectorizer.fit(requiredText)
WordFeatures = word_vectorizer.transform(requiredText)

char_vectorizer = TfidfVectorizer(
    sublinear_tf=True,
    strip_accents='unicode',
    analyzer='char',
    stop_words='english',
    ngram_range=(2, 6),
    max_features=2000)
char_vectorizer.fit(requiredText)
CharFeatures = char_vectorizer.transform(requiredText)
totalFeatures = hstack([WordFeatures, CharFeatures])
'''
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)
```

```
Feature completed …
(135, 1500)
(34, 1500)
```

[15]:
```
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)))
#print("Confusion matrix:\n%s" % metrics.confusion_matrix(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
```

[16]:
```
clf = OneVsRestClassifier(MultinomialNB()).fit(X_train, y_train)
prediction = clf.predict(X_test)
print('Accuracy of MultinomialNB Classifier on training set: {:.2f}'.format(clf.
 ↪score(X_train, y_train)))
print('Accuracy of MultinomialNB 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 MultinomialNB Classifier on training set: 0.84
Accuracy of MultinomialNB Classifier on test set: 0.26

 Classification report for classifier
```

```
OneVsRestClassifier(estimator=MultinomialNB()):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       0.00      0.00      0.00         1
           2       0.00      0.00      0.00         2
           3       0.00      0.00      0.00         1
           5       1.00      1.00      1.00         1
           6       1.00      0.67      0.80         3
           7       0.25      1.00      0.40         1
           9       0.00      0.00      0.00         4
          11       0.00      0.00      0.00         3
          12       0.00      0.00      0.00         0
          13       1.00      0.50      0.67         2
          14       0.00      0.00      0.00         3
          15       0.13      1.00      0.24         2
          16       0.00      0.00      0.00         1
          17       0.00      0.00      0.00         2
          20       0.00      0.00      0.00         3
          21       0.00      0.00      0.00         1
          22       0.00      0.00      0.00         1
          23       0.00      0.00      0.00         1
          24       1.00      1.00      1.00         1

    accuracy                           0.26        34
   macro avg       0.27      0.31      0.26        34
weighted avg       0.25      0.26      0.22        34
```

[ ]: