Resume Screening using NLP+Different ML Algorithms

A summary of resume screening:

- **1. Definition**: Resume screening is the process of determining whether a candidate is qualified for a role based his or her education, experience, and other information captured on their resume.
- **2. How to screen resumes**: First, screen resumes based on the job's minimum qualifications. Second, screen resumes based on the job's preferred qualifications. Third, screen resumes based on the shortlist of candidates you want to move onto the interview phase.
- **3. The challenges recruiters face while screening resumes**: The high volume of resumes received – up to 88% of them are unqualified – greatly increases time to fill. Recruiters face increased pressure to show quality of hire but lack tools to link their resume screening to post-hire metrics.
- **4. Tech innovations in resume screening**: Intelligent resume screening by using AI to learn from historical hiring decisions to improve quality of hire and reduce employee turnover.

Source

In this project, machine learning models is developed for the Resume Screening task.

Notebook Content

Importing Basic Libraries and Loading Dataset

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

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

	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...

Understanding Dataset

```
df.shape
```

```
(962, 2)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 962 entries, 0 to 961
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Category  962 non-null    object
 1   Resume    962 non-null    object
```

```
dtypes: object(2)
memory usage: 15.2+ KB
```

```
df.isnull().sum()
```

```
Category    0
Resume      0
dtype: int64
```

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

```
array(['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'],
      dtype=object)
```

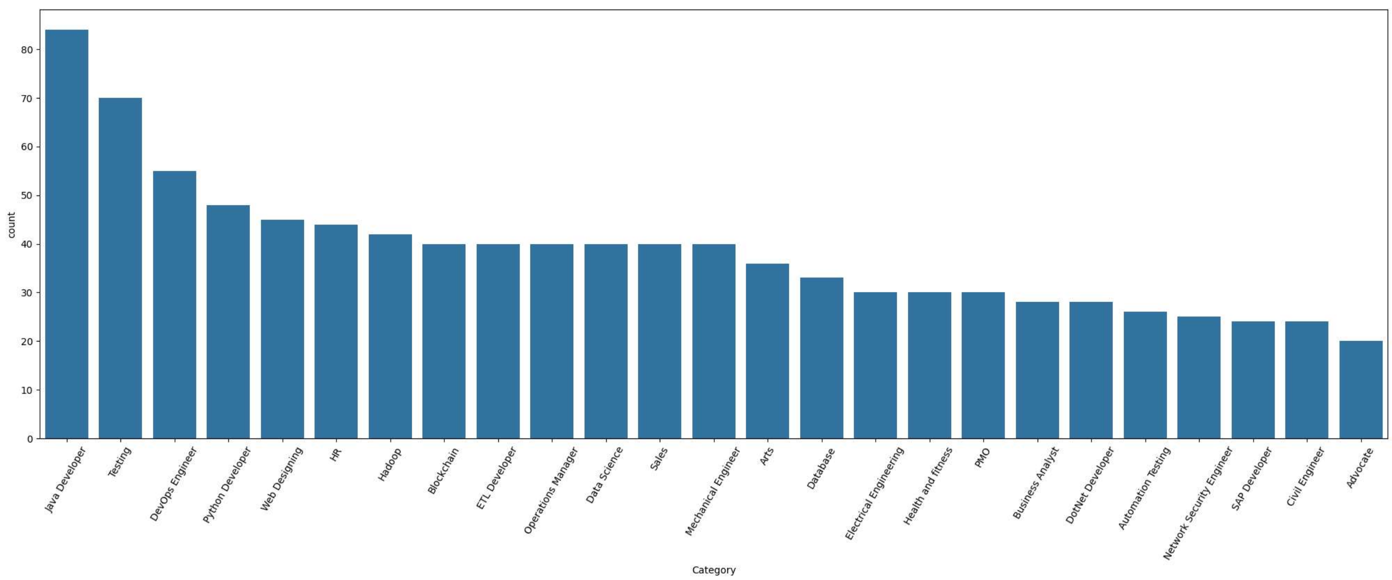
```
df['Category'].nunique()
```

```
25
```

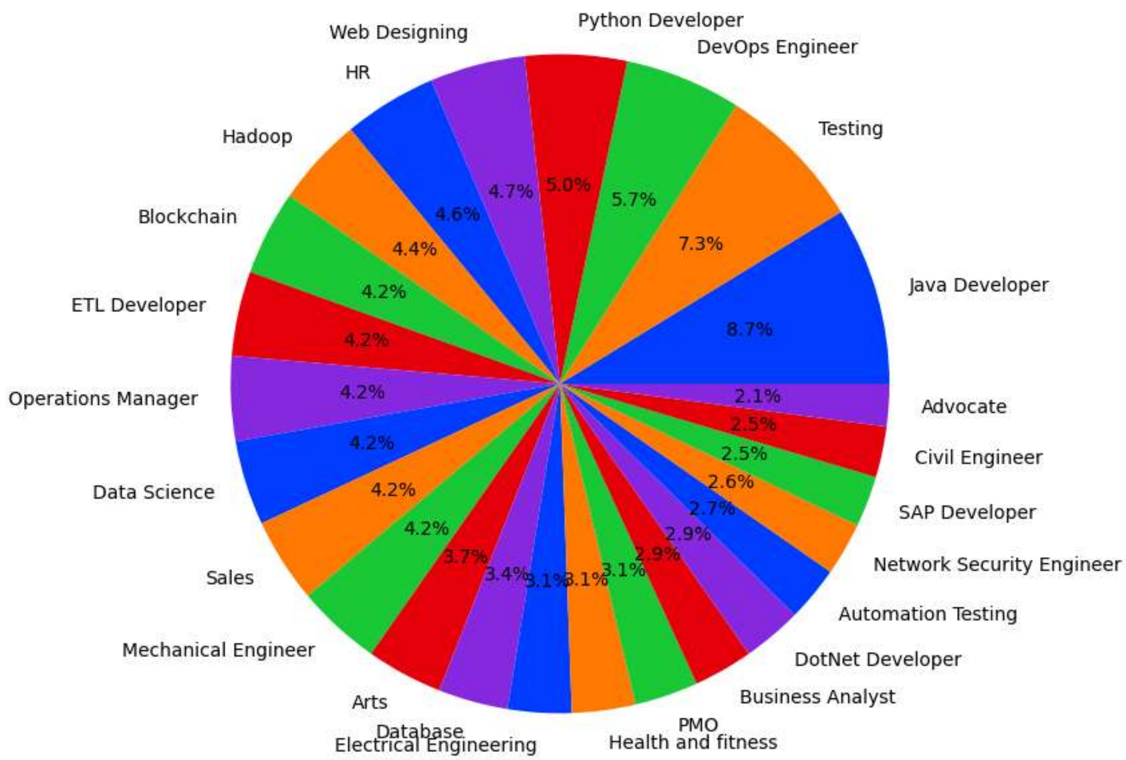
```
categories = df['Category'].value_counts().reset_index()
categories
```

	Category	count
0	Java Developer	84
1	Testing	70
2	DevOps Engineer	55
3	Python Developer	48
4	Web Designing	45
5	HR	44
6	Hadoop	42
7	Blockchain	40
8	ETL Developer	40
9	Operations Manager	40
10	Data Science	40
11	Sales	40
12	Mechanical Engineer	40
13	Arts	36
14	Database	33
15	Electrical Engineering	30
16	Health and fitness	30
17	PMO	30
18	Business Analyst	28
19	DotNet Developer	28
20	Automation Testing	26
21	Network Security Engineer	25
22	SAP Developer	24
23	Civil Engineer	24
24	Advocate	20

```
plt.figure(figsize=(25,8))
plt.xticks(rotation=60)
# count plot on single categorical variable
sns.countplot(x ='Category', data= df, order= df['Category'].value_counts().index)

# Show the plot
plt.show()
```




```
plt.figure(figsize=(25,8))

# Define Seaborn color palette to use
colors = sns.color_palette('bright')[0:5]

# Create pie chart
plt.pie(categories['count'], labels=categories['Category'], colors=colors, autopct='%.1f%%')
plt.show()
```



Preprocessing

Let's create a helper function to remove URLs, hashtags, mentions, special letters and punctuation

Firstly, Let's add a new column for this:

```
df1= df.copy()
df1['cleaned_resume']= ""
df1
```

	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...	
...	...	...	...
957	Testing	Computer Skills: â¢ Proficient in MS office (...	
958	Testing	â Willingness to accept the challenges. â ...	
959	Testing	PERSONAL SKILLS â¢ Quick learner, â¢ Eagerne...	
960	Testing	COMPUTER SKILLS & SOFTWARE KNOWLEDGE MS-Power ...	
961	Testing	Skill Set OS Windows XP/7/8/8.1/10 Database MY...	

962 rows × 3 columns

Function:

```
import re
def clean_function(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
```

Let's apply to columns:

```
df1['cleaned_resume'] = df1['Resume'].apply(lambda x: clean_function(x))
df1.head()
```

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

Let's encode the Category column:

```
from sklearn.preprocessing import LabelEncoder
df2= df1.copy()
df2['Category']= LabelEncoder().fit_transform(df2['Category'])
df2.head()
```

	Category	Resume	cleaned_resume
0	6	Skills * Programming Languages: Python (pandas...	Skills Programming Languages Python pandas num...
1	6	Education Details \r\nMay 2013 to May 2017 B.E...	Education Details May 2013 to May 2017 B E UIT...
2	6	Areas of Interest Deep Learning, Control Syste...	Areas of Interest Deep Learning Control System...
3	6	Skills â¢ R â¢ Python â¢ SAP HANA â¢ Table...	Skills R Python SAP HANA Tableau SAP HANA SQL ...
4	6	Education Details \r\n MCA YMCAUST, Faridab...	Education Details MCA YMCAUST Faridabad Haryan...

Let's create wordcloud:

```
import nltk
from nltk.corpus import stopwords
import string
from wordcloud import WordCloud
nltk.download('stopwords')
nltk.download('punkt')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
#Stop words are generally the most common words in a language.
#English stop words from nltk:
SetOfStopWords= set(stopwords.words('english')+['``',"''"])
```

```
totalWords= []
```

```
Sentences= df2['Resume'].values
```

```
cleanedSentences= ""
```

```
for records in Sentences:
    cleanedText= clean_function(records)
    cleanedSentences += cleanedText
    requiredWords = nltk.word_tokenize(cleanedText)
    for word in requiredWords:
        if word not in SetOfStopWords and word not in string.punctuation:
            totalWords.append(word)
```

```
wordfreqdist = nltk.FreqDist(totalWords)
```

```
wordfreqdist
```

```
FreqDist({'Exprience': 3829, 'months': 3233, 'company': 3130, 'Details': 2967, 'description': 2634, '1': 2134, 'Project': 1808,
'project': 1579, '6': 1499, 'data': 1438, ...})
```

```
mostcommon = wordfreqdist.most_common(30)
```

```
mostcommon
```

```
[('Exprience', 3829),
 ('months', 3233),
 ('company', 3130),
 ('Details', 2967),
 ('description', 2634),
 ('1', 2134),
 ('Project', 1808),
 ('project', 1579),
 ('6', 1499),
 ('data', 1438),
 ('team', 1424),
 ('Maharashtra', 1385),
 ('year', 1244),
 ('Less', 1137),
 ('January', 1086),
 ('using', 1041),
 ('Skill', 1018),
 ('Pune', 1016),
 ('Management', 1010),
 ('SQL', 990),
 ('Ltd', 934),
 ('management', 927),
 ('C', 896),
 ('Engineering', 855),
 ('Education', 833),
 ('Developer', 806),
 ('Java', 773),
 ('2', 754),
 ('development', 752),
 ('monthsCompany', 746)]
```

```
WordCloud= WordCloud().generate(cleanedSentences)
plt.figure(figsize=(10,10))
plt.imshow(WordCloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

Building Models

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from scipy.sparse import hstack
```

```
Text= df2['cleaned_resume'].values
Target= df2['Category'].values
```

Here we will preprocess and convert the ‘cleaned_resume’ column into vectors. We will be using the ‘Tf-Idf’ method to get the vectors:

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

We have ‘WordFeatures’ as vectors and ‘Target’ and target after this step.

```
WordFeatures.shape
```

```
(962, 7351)
```

Let’s split the data into training and test set:

```
X_train,X_test,y_train,y_test= train_test_split(WordFeatures, Target, random_state=42)
```

```
print(X_train.shape)
print(X_test.shape)
```

```
(721, 7351)
(241, 7351)
```

We have trained and tested the data and now let’s build the models:

```
from sklearn.multiclass import OneVsRestClassifier

from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
models = {
    'K-Nearest Neighbors' : KNeighborsClassifier(),
    'Logistic Regression' : LogisticRegression(),
    'Support Vector Machine' : SVC(),
    'Random Forest' : RandomForestClassifier()
}
model_list=[]
for model in models.values():
    model_list.append(OneVsRestClassifier(model))
model_list
for i in model_list:
    i.fit(X_train, y_train)
    print(f'{i} trained')

print("*"*60)
print("all models trained")
for count, value in enumerate(model_list):
    print(f"Accuracy of {value} on training set :", model_list[count].score(X_train, y_train))
    print(f"Accuracy of {value} on test set :", model_list[count].score(X_test, y_test))
    print("*"*100)

print("all scores calculated")
from sklearn.metrics import confusion_matrix as CM
from sklearn.metrics import classification_report

from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
for count, value in enumerate(model_list):
    print(f'{value} classification report')
    print("-"*80)
    print(classification_report(y_test, model_list[count].predict(X_test)))
    print("*"*100)
    print(" ")
```

```
OneVsRestClassifier(estimator=KNeighborsClassifier()) trained
OneVsRestClassifier(estimator=LogisticRegression()) trained
OneVsRestClassifier(estimator=SVC()) trained
OneVsRestClassifier(estimator=RandomForestClassifier()) trained
************************************************************
all models trained
Accuracy of OneVsRestClassifier(estimator=KNeighborsClassifier()) on training set : 0.9805825242718447
Accuracy of OneVsRestClassifier(estimator=KNeighborsClassifier()) on test set : 0.966804979253112
****************************************************************************************************
Accuracy of OneVsRestClassifier(estimator=LogisticRegression()) on training set : 1.0
Accuracy of OneVsRestClassifier(estimator=LogisticRegression()) on test set : 0.991701244813278
****************************************************************************************************
Accuracy of OneVsRestClassifier(estimator=SVC()) on training set : 1.0
Accuracy of OneVsRestClassifier(estimator=SVC()) on test set : 0.991701244813278
****************************************************************************************************
Accuracy of OneVsRestClassifier(estimator=RandomForestClassifier()) on training set : 1.0
Accuracy of OneVsRestClassifier(estimator=RandomForestClassifier()) on test set : 0.983402489626556
****************************************************************************************************
all scores calculated
OneVsRestClassifier(estimator=KNeighborsClassifier()) classification report
--------------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         6
           1       1.00      1.00      1.00         8
           2       1.00      1.00      1.00         6
           3       1.00      1.00      1.00        10
           4       1.00      1.00      1.00         5
           5       1.00      1.00      1.00         9
           6       0.88      0.78      0.82         9
           7       1.00      0.89      0.94         9
           8       1.00      0.88      0.94        17
           9       1.00      1.00      1.00        10
          10       1.00      1.00      1.00         7
          11       1.00      1.00      1.00         7
          12       1.00      1.00      1.00        15
          13       1.00      1.00      1.00         6
          14       1.00      0.70      0.82        10
```

```
              15       1.00      1.00      1.00        21
              16       1.00      1.00      1.00        10
              17       1.00      1.00      1.00         3
              18       1.00      1.00      1.00        14
              19       0.80      1.00      0.89         8
              20       1.00      1.00      1.00        11
              21       0.78      1.00      0.88         7
              22       0.75      1.00      0.86         9
              23       1.00      1.00      1.00        19
              24       1.00      1.00      1.00         5

        accuracy                           0.97       241
       macro avg       0.97      0.97      0.97       241
    weighted avg       0.97      0.97      0.97       241

    ****************************************************************************************************

    OneVsRestClassifier(estimator=LogisticRegression()) classification report
    --------------------------------------------------------------------------------
```

Cross Validation for Models

```
from sklearn.model_selection import cross_val_score, KFold


results = {}

kf = KFold(n_splits= 10)

for count, value in enumerate(model_list):
    result = cross_val_score(model_list[count], X_train, y_train, scoring= 'accuracy', cv= kf)
    results[value] = result


print("r2 scores")
print("*********************************")
for name, result in results.items():

    print(f'{name} : {round(np.mean(result),3)}')
    print("----------------")
```

```
r2 scores
*********************************
OneVsRestClassifier(estimator=KNeighborsClassifier()) : 0.958
----------------
OneVsRestClassifier(estimator=LogisticRegression()) : 0.99
----------------
OneVsRestClassifier(estimator=SVC()) : 0.997
----------------
OneVsRestClassifier(estimator=RandomForestClassifier()) : 0.994
----------------
```

```
import matplotlib.pyplot as plt

# Bar plot for accuracy scores
train_accuracy = [model_list[i].score(X_train, y_train) for i in range(len(model_list))]
test_accuracy = [model_list[i].score(X_test, y_test) for i in range(len(model_list))]

plt.figure(figsize=(10, 6))
plt.barh(list(models.keys()), train_accuracy, color='skyblue', label='Training Accuracy')
plt.barh(list(models.keys()), test_accuracy, color='orange', label='Test Accuracy')
plt.xlabel('Accuracy')
plt.title('Accuracy of Different Models on Training and Test Sets')
plt.legend()
plt.show()

# Line plot for cross-validation scores
cv_scores = [np.mean(results[model]) for model in model_list]

plt.figure(figsize=(10, 6))
plt.plot(list(models.keys()), cv_scores, marker='o', color='green')
plt.xlabel('Models')
plt.ylabel('Cross-Validation Score')
plt.title('Cross-Validation Scores of Different Models')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

```python
from sklearn.metrics import confusion_matrix

for count, value in enumerate(model_list):
    print(f'{value} confusion matrix')
    print("-" * 40)
    print(confusion_matrix(y_test, model_list[count].predict(X_test)))
    print("*" * 100)
    print(" ")
```

```
  [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
    5]]
****************************************************************************************************

OneVsRestClassifier(estimator=RandomForestClassifier()) confusion matrix
----------------------------------------
[[ 4  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0
   0]
 [ 0  8  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0]
 [ 0  0  6  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0]
 [ 0  0  0 10  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0]
 [ 0  0  0  0  5  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0]
 [ 0  0  0  0  0  9  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0]
 [ 0  0  0  0  0  0  9  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0]
 [ 0  0  0  0  0  0  0  9  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0]
 [ 0  0  0  0  0  0  0  0 15  0  0  0  0  0  0  1  0  0  0  1  0  0  0  0
   0]
 [ 0  0  0  0  0  0  0  0  0 10  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0]
 [ 0  0  0  0  0  0  0  0  0  0  7  0  0  0  0  0  0  0  0  0  0  0  0  0
   0]
 [ 0  0  0  0  0  0  0  0  0  0  0  7  0  0  0  0  0  0  0  0  0  0  0  0
   0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 15  0  0  0  0  0  0  0  0  0  0  0
   0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  6  0  0  0  0  0  0  0  0  0  0
   0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 10  0  0  0  0  0  0  0  0  0
   0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 21  0  0  0  0  0  0  0  0
   0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 10  0  0  0  0  0  0  0
   0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0
```