```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv("UpdatedResumeDataSet.csv")
```

```
df
```

| | 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... |
| **...** | ... | ... |
| **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 × 2 columns

```
print('Category of Resume')
labels = df['Category'].unique()
labels
```

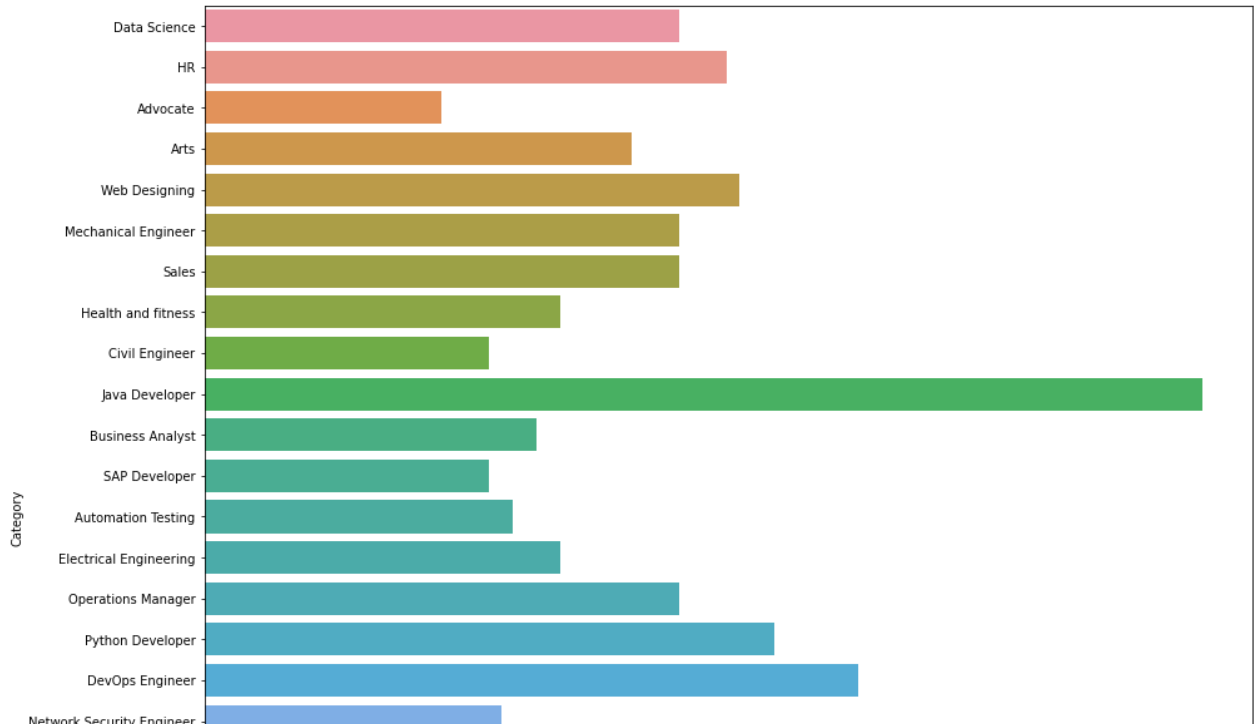
```
Category of Resume
array(['Data Science', 'HR', 'Advocate', 'Arts', 'Web Designing',
       'Mechanical Engineer', 'Sales', 'Health and fitness',
       'Civil Engineer', 'Java Developer', 'Business Analyst',
       'SAP Developer', 'Automation Testing', 'Electrical Engineering',
       'Operations Manager', 'Python Developer', 'DevOps Engineer',
       'Network Security Engineer', 'PMO', 'Database', 'Hadoop',
       'ETL Developer', 'DotNet Developer', 'Blockchain', 'Testing'],
      dtype=object)
```

```
counts = df['Category'].value_counts() #value counts of category
counts
```

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

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

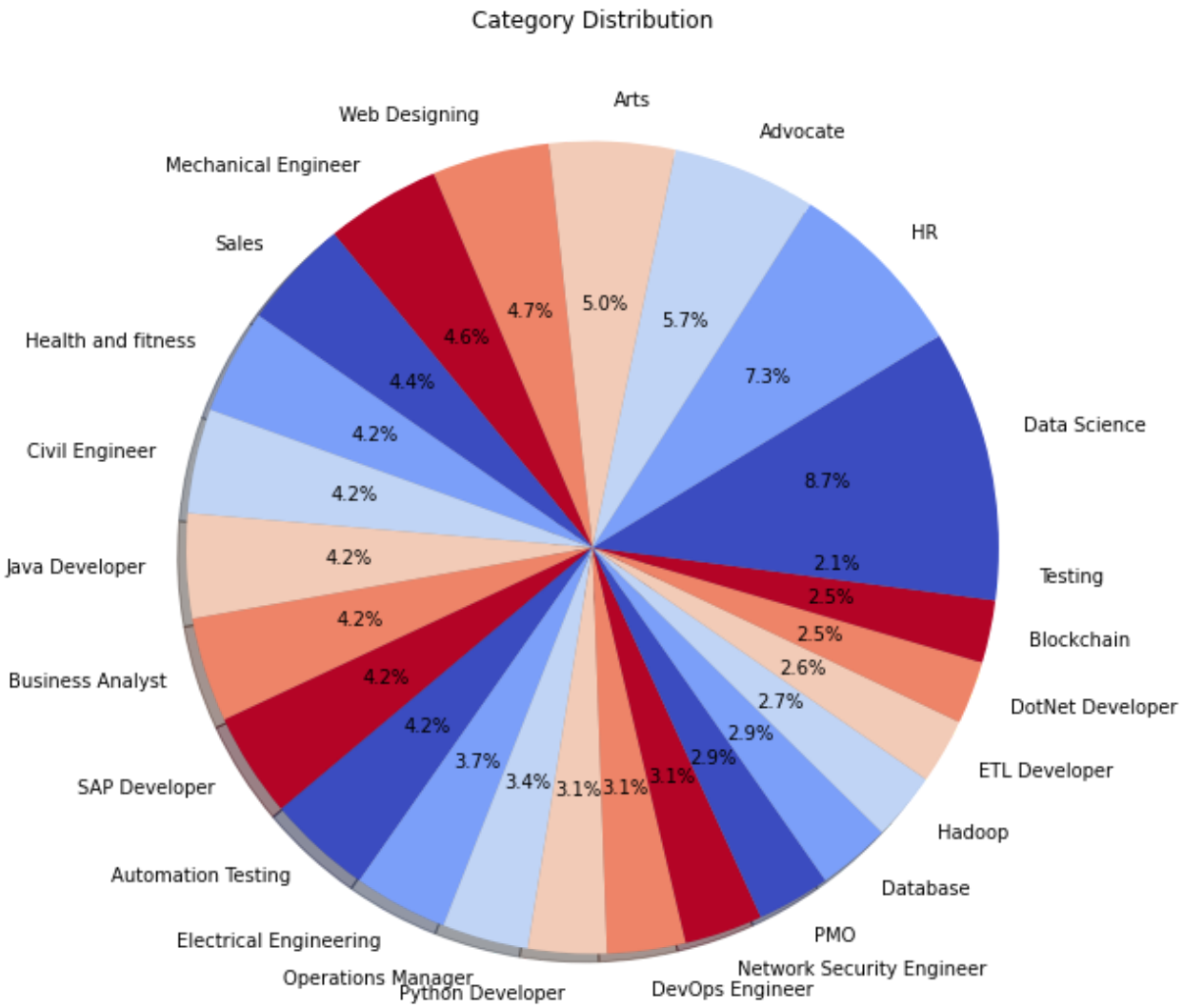
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8a3e304c40>
```




By Looking from the graph we can know that There are high java Developer resumes occur in the dataset than others.


```
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("""!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"""), ' ', r
    resumeText = re.sub(r'[^\x00-\x7f]',r' ', resumeText)
    resumeText = re.sub('\s+', ' ', resumeText)  # remove extra whitespace
    return resumeText

df['cleaned_resume'] = df.Resume.apply(lambda x: cleanResume(x))
```

```
from matplotlib.gridspec import GridSpec
counts = df['Category'].value_counts()
labels = df['Category'].unique()
plt.figure(1, figsize=(22,22))
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(counts, labels=labels, autopct='%1.1f%%', shadow=True, colors=colors)
plt.show()
```




By looking from the piechart that we can know that the percentage of data science is high than others.

```
import nltk
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
```

```
from nltk.corpus import stopwords
import string
from wordcloud import WordCloud


oneSetOfStopWords = set(stopwords.words('english')+['``',"''"])
totalWords =[]
Sentences = df['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()
```

```
[('Details', 484), ('Exprience', 446), ('months', 376), ('company', 330), ('descri
```


```
from sklearn.preprocessing import LabelEncoder

var_mod = ['Category']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i])
```

To convert the objects into the number we have perform label encoding.

---

Spliting the data

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

```
x = df['cleaned_resume'].values
y = df['Category'].values


vec_word = TfidfVectorizer()



vec_word.fit(x)
WordFeatures = vec_word.transform(x)



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

X_train,X_test,y_train,y_test = train_test_split(WordFeatures,y,random_state=1, test_size=
print(X_train.shape)
print(X_test.shape)
```

```
Feature completed .....
(769, 7566)
(193, 7566)
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn import metrics
from sklearn.multiclass import OneVsRestClassifier
```

```
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_trair
print('Accuracy of KNeighbors Classifier on test set: {:.2f}'.format(clf.score(X_test, y_t

print("\n Classification report for classifier %s:\n%s\n" % (clf, metrics.classification_r
```

```
Accuracy of KNeighbors Classifier on training set: 0.98
Accuracy of KNeighbors Classifier on test set: 0.98

 Classification report for classifier OneVsRestClassifier(estimator=KNeighborsClassi
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         4
           1       1.00      1.00      1.00         6
           2       1.00      1.00      1.00         5
           3       1.00      1.00      1.00        13
           4       0.86      1.00      0.92         6
           5       1.00      1.00      1.00         7
           6       1.00      0.78      0.88         9
           7       1.00      1.00      1.00         5
           8       1.00      0.90      0.95        10
           9       1.00      1.00      1.00         4
          10       1.00      1.00      1.00         9
          11       1.00      1.00      1.00         7
          12       1.00      1.00      1.00        10
          13       1.00      1.00      1.00         3
          14       1.00      1.00      1.00         4
          15       1.00      1.00      1.00        14
          16       1.00      1.00      1.00         9
          17       1.00      1.00      1.00         8
          18       1.00      1.00      1.00         8
          19       1.00      1.00      1.00         5
          20       1.00      1.00      1.00        11
          21       0.75      1.00      0.86         6
          22       1.00      1.00      1.00        11
          23       1.00      1.00      1.00        12
          24       1.00      1.00      1.00         7

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

Result:

Accuracy of KNeighbors Classifier on training set: 0.98

Accuracy of KNeighbors Classifier on test set: 0.98

0s completed at 11:34 PM