Resume Screening

Intro:

Resume screening is the process of evaluating job applications to find the most suitable candidate for a position. It can be a time-consuming and error-prone task, especially when there are thousands of resumes to review. Natural Language Processing (NLP) is a branch of artificial intelligence that deals with the analysis and generation of natural language. NLP can be used to automate resume screening by extracting relevant information from resumes, such as skills, education, and experience, and comparing them with the job requirements. NLP can also rank the resumes based on their similarity to the job description and provide feedback to the applicants.

Tools used and Methods:

1)Kaggle (Compiler):

Kaggle provides a platform for data science and machine learning projects, offering various datasets and tools for analysis.

2)Regular Expression (Data Preprocessing):

Regular expressions are used to efficiently search and manipulate text, aiding in the preprocessing of resume data.

3)Keras Framework (LSTM):

Keras is a high-level neural networks API, which can be utilized for implementing deep learning models such as Long Short-Term Memory (LSTM) networks, beneficial for sequence processing tasks like natural language understanding.

4)Pandas (Data Reading):

Pandas is a Python library commonly used for data manipulation and analysis, facilitating the reading and handling of resume datasets.

5)Matplotlib & Seaborn (Data Visualization):

These libraries enable the visualization of data insights, helping to understand patterns and relationships within the resume dataset.

Main Idea:

The goal of this project is to automate the process of job profile selection. The project involves training an NLP model on a dataset of resumes and job descriptions to identify the best candidates for a particular job. The model uses NLP techniques to extract relevant information from the resumes and job descriptions, such as skills, experience, and education. The model then uses this information to categorize the resumes based on their relevance to the job.

Motivation:

Improved candidate selection, time savings.

Aim:

Efficiency and accuracy in screening resumes and Identifying Best Candidates for Job Positions

Challenges:

Time-consuming and error-prone, especially with thousands of resumes.

How to run the project:

1- Jupyter notebook

-Import .ipynb file to the workspace

-Download the data from Kaggle

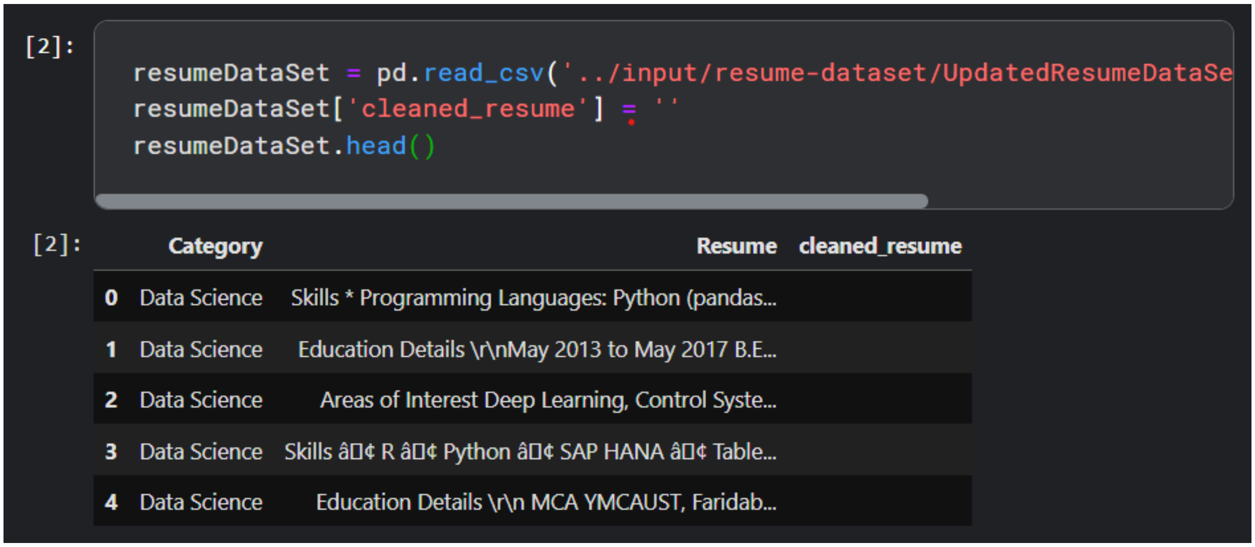
(https://www.kaggle.com/datasets/gauravduttakiit/resume-dataset)

-Add the local path of the data in the pandas function

2- Kaggle

-Just import the .ipynb file to the workspace and the data will automatically initialize and you can run the cells normally

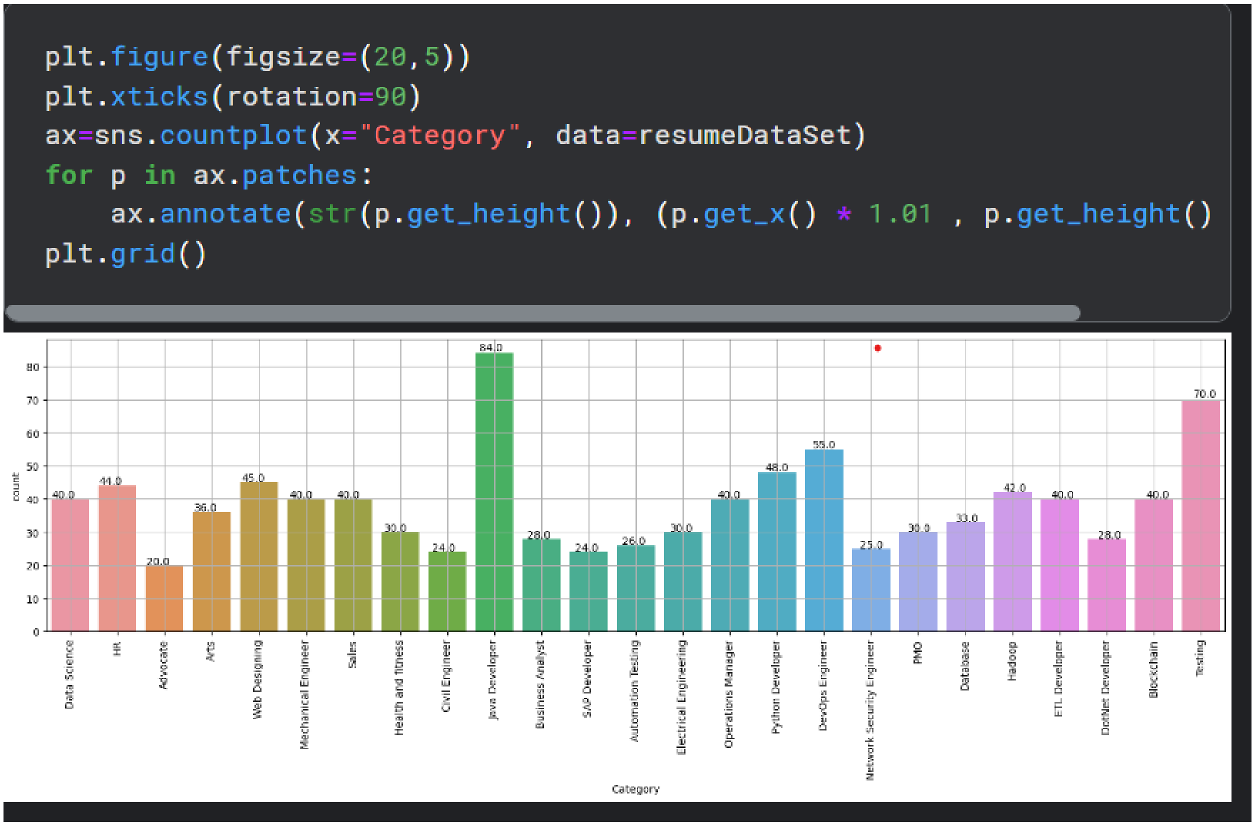
Code:



```
[2]:
resumeDataSet = pd.read_csv('../input/resume-dataset/UpdatedResumeDataSe
resumeDataSet['cleaned_resume'] = ''
resumeDataSet.head()
```

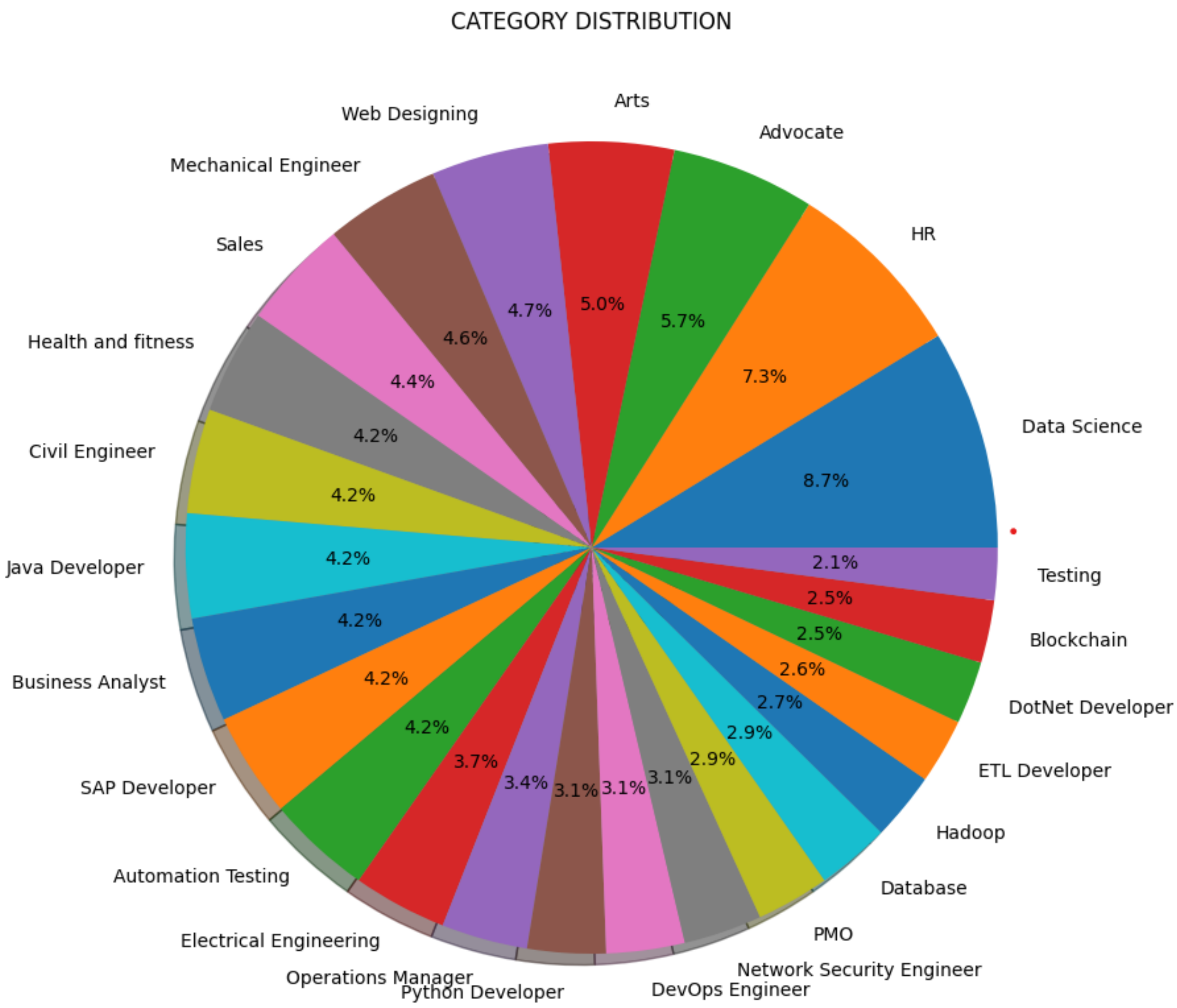
```
[2]:     Category                                           Resume  cleaned_resume
      0  Data Science    Skills * Programming Languages: Python (pandas...
      1  Data Science     Education Details \r\nMay 2013 to May 2017 B.E...
      2  Data Science        Areas of Interest Deep Learning, Control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

First we read data using pandas and creating a new columncalled "cleaned_resume" to add the preprocessed data in it

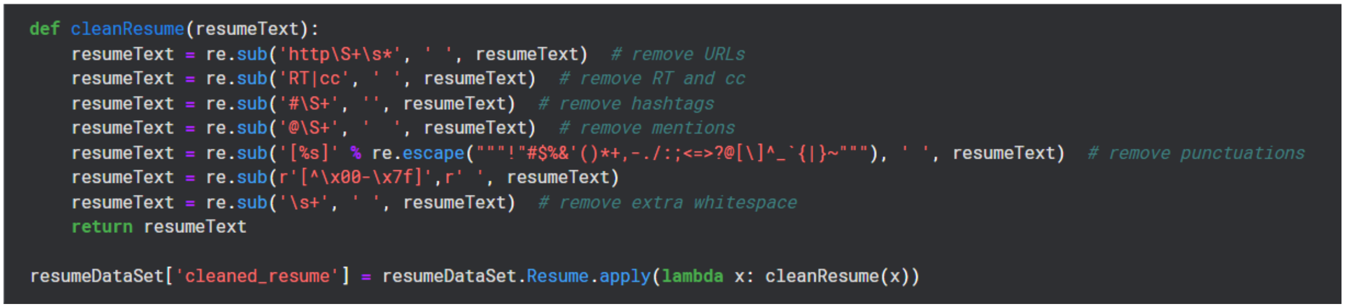


```
plt.figure(figsize=(20,5))
plt.xticks(rotation=90)
ax=sns.countplot(x="Category", data=resumeDataSet)
for p in ax.patches:
    ax.annotate(str(p.get_height()), (p.get_x() * 1.01 , p.get_height()
plt.grid()
```

Then we make data visualization to see the number of documents that belong to each category in the data



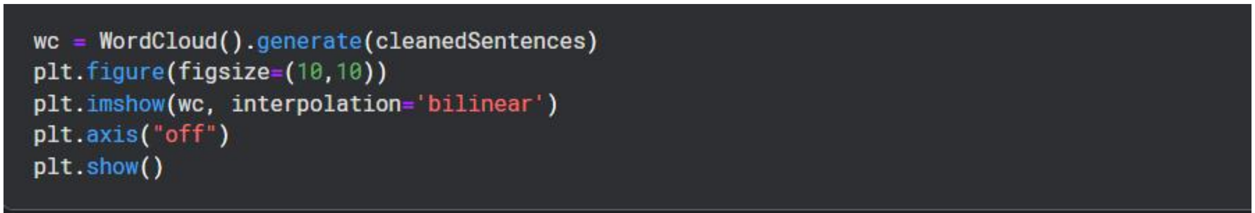
Here we conclude that although Java developer category had the most number of documents but the category “Data Science” was the most distributed one



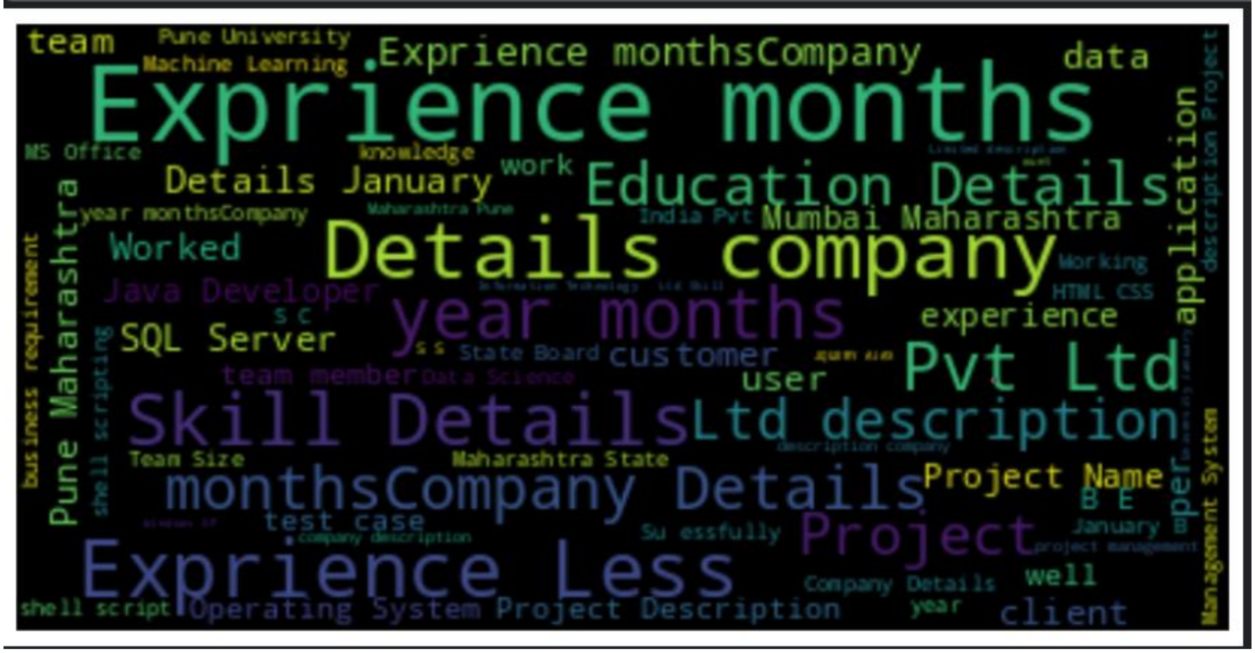
```
def cleanResume(resumeText):
    resumeText = re.sub('http\S+\s*', ' ', resumeText)  # remove URLs
    resumeText = re.sub('RT|cc', ' ', resumeText)  # remove RT and cc
    resumeText = re.sub('#\S+', '', resumeText)  # remove hashtags
    resumeText = re.sub('@\S+', '  ', resumeText)  # remove mentions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

The **cleanResume** function preprocesses a resume text by performing the following cleaning steps:

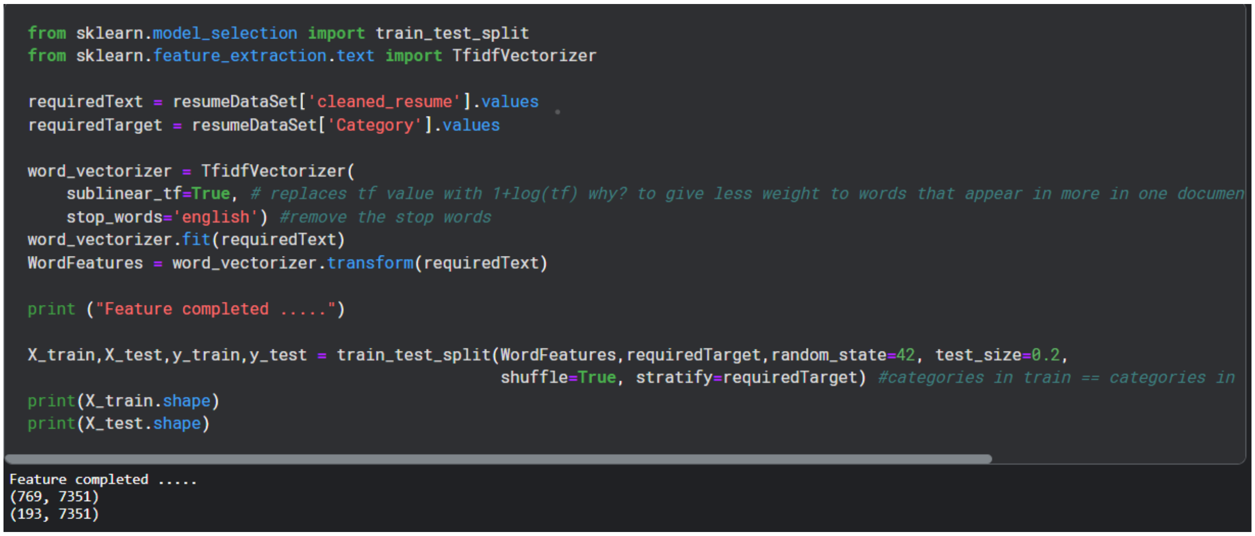
1. **Remove URLs**: Removes URLs from the text.
2. **Remove RT and cc**: Removes 'RT' and 'cc' abbreviations.
3. **Remove Hashtags**: Removes hashtags.
4. **Remove Mentions**: Removes mentions or usernames.
5. **Remove Punctuations**: Removes special characters and punctuations.
6. **Remove Non-ASCII Characters**: Removes non-ASCII characters.
7. **Remove Extra Whitespace**: Removes extra whitespace characters.



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



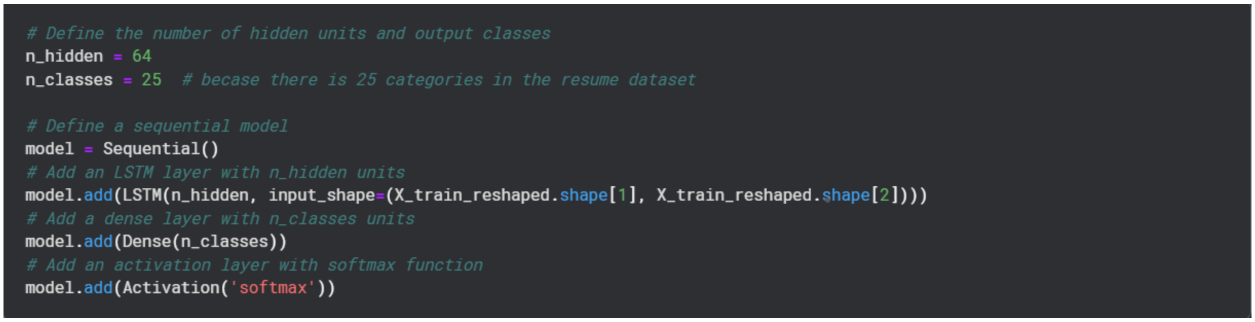
Another data visualization technique using Word Cloud which shows bigger words means more appearance of the word in the data



```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer

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

word_vectorizer = TfidfVectorizer(
    sublinear_tf=True, # replaces tf value with 1+log(tf) why? to give less weight to words that appear in more in one documen
    stop_words='english') #remove the stop words
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=42, test_size=0.2,
                                                 shuffle=True, stratify=requiredTarget) #categories in train == categories in
print(X_train.shape)
print(X_test.shape)
```

```
Feature completed .....
(769, 7351)
(193, 7351)
```

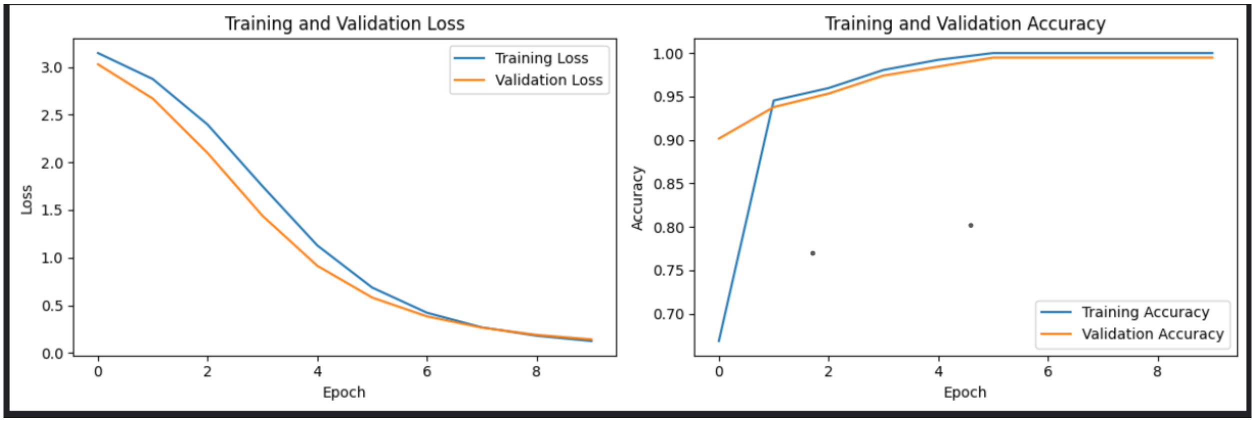
1. **Data Extraction:** Extracts the cleaned resume texts and target categories from the **resumeDataSet** DataFrame.
2. **Feature Extraction**: Uses **TfidfVectorizer** to convert the cleaned resume texts into a matrix of TF-IDF features (**WordFeatures**).
3. **Data Splitting:** Splits the TF-IDF features and target categories into training and testing sets using **train_test_split**.



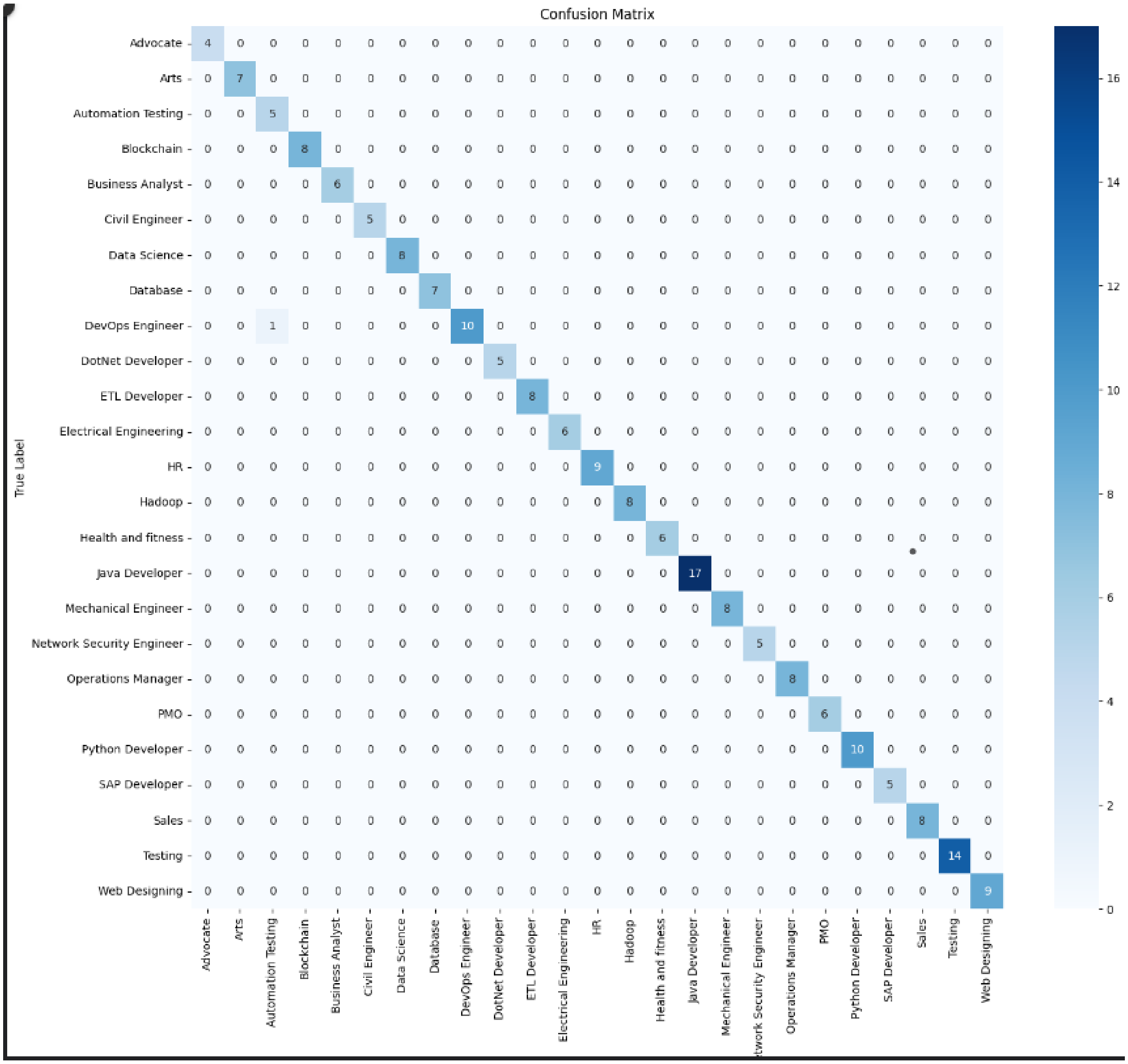
```
# Define the number of hidden units and output classes
n_hidden = 64
n_classes = 25  # becase there is 25 categories in the resume dataset

# Define a sequential model
model = Sequential()
# Add an LSTM layer with n_hidden units
model.add(LSTM(n_hidden, input_shape=(X_train_reshaped.shape[1], X_train_reshaped.shape[2])))
# Add a dense layer with n_classes units
model.add(Dense(n_classes))
# Add an activation layer with softmax function
model.add(Activation('softmax'))
```

-Here we used Long-Short term memory (LSTM) which is a NLP model which can capture long term dependencies allowing the model to remember large sequences of data to produce accurate results

Model Results :



Confusion Matrix



	Advocate	Arts	Automation Testing	Blockchain	Business Analyst	Civil Engineer	Data Science	Database	DevOps Engineer	DotNet Developer	ETL Developer	Electrical Engineering	HR	Hadoop	Health and fitness	Java Developer	Mechanical Engineer	twork Security Engineer	Operations Manager	PMO	Python Developer	SAP Developer	Sales	Testing	Web Designing
Advocate	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Arts	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
Automation Testing	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
Blockchain	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
Business Analyst	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
Civil Engineer	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
Data Science	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
Database	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
DevOps Engineer	0	0	1	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DotNet Developer	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
ETL Developer	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
Electrical Engineering	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
HR	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
Hadoop	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
Health and fitness	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
Java Developer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0
Mechanical Engineer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0
Network Security Engineer	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
Operations Manager	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0
PMO	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
Python Developer	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
SAP Developer	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
Sales	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0
Testing	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0
Web Designing	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

Team Rules:

Abdelrahman Mohamed Elsayed Shehata 20201375008

Idea + Decoder +Encoder +Test + Confusion Matrix + Report + presentation

Mostafa Marwan 20201477542

Model + TF-IDF tokenization

Abdelrahman Mohamed Ali 2103115

Data Cleaning

Hisham Gaber Fathy 20201040744

Data Visualization