job description classification

In this notebook we are going to classify random job descriptions from linkedin into clusters. The goal is to identify which type of resume must go for the job application. We are going to use NLP techniques to first clean the job description and then we are going to pass it to our pickled model which will classify the cluster and then we will know what resume to use for the job application.

In [81]:
```
#getting the packages
import pandas as pd
import re
import nltk
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

import pickle
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [92]:
```
# Load the clustering model
with open('clustering_model.pkl', 'rb') as f:
    jobclassifier = pickle.load(f)
```

In [93]:
```
# Initialize the lemmatizer
lemmatizer = WordNetLemmatizer()
```

In [94]:
```
# Function to clean the text
def clean_text(text):
    cleaned_text = re.sub('[^a-zA-Z]', ' ', text)  # Remove special characters and numbers
    cleaned_text = cleaned_text.lower()  # Convert to lowercase
    return cleaned_text

# Function to tokenize the text
def tokenize_text(text):
    tokens = word_tokenize(text)  # Tokenize the text into words
    return tokens

# Function to lemmatize the tokens
def lemmatize_text(tokens):
    lemmatizer = WordNetLemmatizer()
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens]  # Lemmatize the tokens
    return lemmatized_tokens

# Preprocess the job description
def preprocess_job_description(job_description):
    cleaned_text = clean_text(job_description)
    tokens = tokenize_text(cleaned_text)
    lemmatized_tokens = lemmatize_text(tokens)
    cleaned_job_description = ' '.join(lemmatized_tokens)
    return cleaned_job_description
```

In [95]:
```
# Example usage
job_description="""About the job
Position Summary

To be a driven business analyst who can work on complex Analytical problems and help the customer in better bus

Job Responsibilities

Technical â€“ Able to comprehend complex tasks assigned and executes them with little to no supervision.

Logical Thinking â€“ Able to think analytically, use a systematic and logical approach to analyze data, problem

Task Management â€“ Basic level of project management knowledge and experience. Should be able to plan tasks, d

Communication â€“ Able to convey ideas and information clearly and accurately to self or others whether in writ

Education

Bachelor of Engineering in Statistics

Work Experience

Behavioural Competencies

Teamwork & Leadership

Motivation to Learn and Grow

Ownership

Cultural Fit

Communication

Technical Competencies

Python

R

SQL

EXCEL

MMx

Forecasting

Machine Learning

Pharma Commercial Know How

Patient Data Analytics Know How

Dataiku

KNIME

Others"
"
"""
cleaned_job_description = preprocess_job_description(job_description)
print("Cleaned Job Description:", cleaned_job_description)
```

```
Cleaned Job Description: about the job position summary to be a driven business analyst who can work on complex analytical problem and help the customer in better business decision making especially in the area of pharma do main job responsibility technical able to comprehend complex task assigned and executes them with little to no supervision logical thinking able to think analytically use a systematic and logical approach to analyze data p roblem and situation notice discrepancy and inconsistency in information and material task management basic lev el of project management knowledge and experience should be able to plan task discus and work on priority commu nication able to convey idea and information clearly and accurately to self or others whether in writing or ver bal education bachelor of engineering in statistic work experience behavioural competency teamwork leadership m otivation to learn and grow ownership cultural fit communication technical competency python r sql excel mmx fo recasting machine learning pharma commercial know how patient data analytics know how dataiku knime others
```

In [96]:
```
# Load the TF-IDF vectorizer
vectorizer = TfidfVectorizer()

vectorizer
```

Out[96]:
```
TfidfVectorizer()
```

In [97]:
```
# Fit the vectorizer on a corpus of documents
corpus = [cleaned_job_description]
vectorizer.fit(corpus)

# Transform the job description into a numerical representation
job_description_vector = vectorizer.transform([cleaned_job_description])
job_description_vector
```

Out[97]:
```
<1x105 sparse matrix of type '<class 'numpy.float64'>'
        with 105 stored elements in Compressed Sparse Row format>
```

In [98]:
```
#reshaping the vector
job_description_vector_reshaped = job_description_vector.reshape(-1, 1)
job_description_vector_reshaped
```

Out[98]:
```
<105x1 sparse matrix of type '<class 'numpy.float64'>'
        with 105 stored elements in COOrdinate format>
```

In [100…
```
# Predict the cluster label for the job description
predicted_cluster = jobclassifier.predict(job_description_vector_reshaped)[0]

# Assign the job description to a specific cluster
if cluster_label == 1:
    resume = "Resume Type 1"
elif cluster_label == 0:
    resume = "Resume Type 2"
else:
    resume = "Hybrid"

print("you should use the resume type: ", resume)
```

```
you should use the resume type:  Hybrid
```

In [ ]:

In [ ]: