Rathinam College of Arts and Science

Resume ranking system for sustainability in Recruitment

Safvan C K

RCAS2021MDB044

Saravana Kumar

Sr. Faculty IT

Problem Statement

- Smart India Hackathon 2022
- All India Council for Technical Education (AICTE).
- Lack of sustainability in appointment of Faculty in Higher Educational Institutions and Colleges
- DR712

Problem Statement

- Faculty retirement and college faculty appointments are not standardized
- There are no plans in place to fill the vacancy that develops immediately.
- Pupils fall behind due to the this problem
- An application which will indicate the future vacancies and positions for planning of recruitment is created.
- Data of faculties leaving or retiring is updated for understanding the vacancies and faculties who are ready to join are found.

Literature Review

Year	Title	Objective	Existing	Future Enhancement
2016	Resume Ranking using NLP and Machine Learning.	Make the current resume ranking system more flexible for both the entity	**T**ake the bulk of input resume from the client company and that client company will also provide the requirement and the constraints according to which the resume should be ranked by system	The application can be extended further to other domains like Telecom, Healthcare, Ecommerce and public sector jobs.

Year	Title	Objective	Existing	Future Enhancement
2017	A hybrid approach to conceptual classification and ranking of resumes and their corresponding job posts	▪ Automatic matching between candidate resumes and their corresponding job offers • Efficiently route them to their appropriate occupati onal categories.	▪ Hybrid approach that employs conce ptul based classific ation of resumes. • Automatically ranks candidate resumesto t heir corresponding job offers	Utilize the extracted information from applicants' resumes to dynamically generate user profiles to be further used for recommending jobs to job seekers
2021	Design and Development of Machine Learning based Resume Ranking System	Identify the most qualified candidates for a certain vacancy	▪ Top applicants might be rated using content-based suggestion • Pick and rank Curriculum Vitaes (CV) based on job descriptions in huge quantities.	A hybrid recommendation system that combines both collaborative Filtering Recommendation Systems and Content-Based Filtering Systems

Year	Title	Objective	Existing	Future Enhancement
2021	Design and Development of Machine Learning based Resume Ranking System	Identify the most qualified candidates for a certain vacancy	▪ Top applicants might be rated using content-based suggestion • Pick and rank Curriculum Vitaes (CV) based on job descriptions in huge quantities.	A hybrid recommendation system that combines both collaborative Filtering Recommendation Systems and Content-Based Filtering Systems
2022	Resume Classification using various Machine Learning Algorithms.	Automate the hiring process in order to enhance efficiency and decrease manual labour that may be done electronically	Appropriate job profile may be retrieved from the categorized and pre-processed data and shown on the interviewer's screen.	Combination of other video interviewing features such as facial recognition, voice to text generation and voice analysis

Project Goals

Project Goals

- The system provides ranked information about the candidates.
- This will enable recruiters to screen resumes in less time and with less effort.
- The candidate must upload a Resume and create a profile. Then they can give preference for jobs based on their willingness such as pay range, employment location, and work kind.
- Recruiter must update its database with information on the Job.

Project Goals

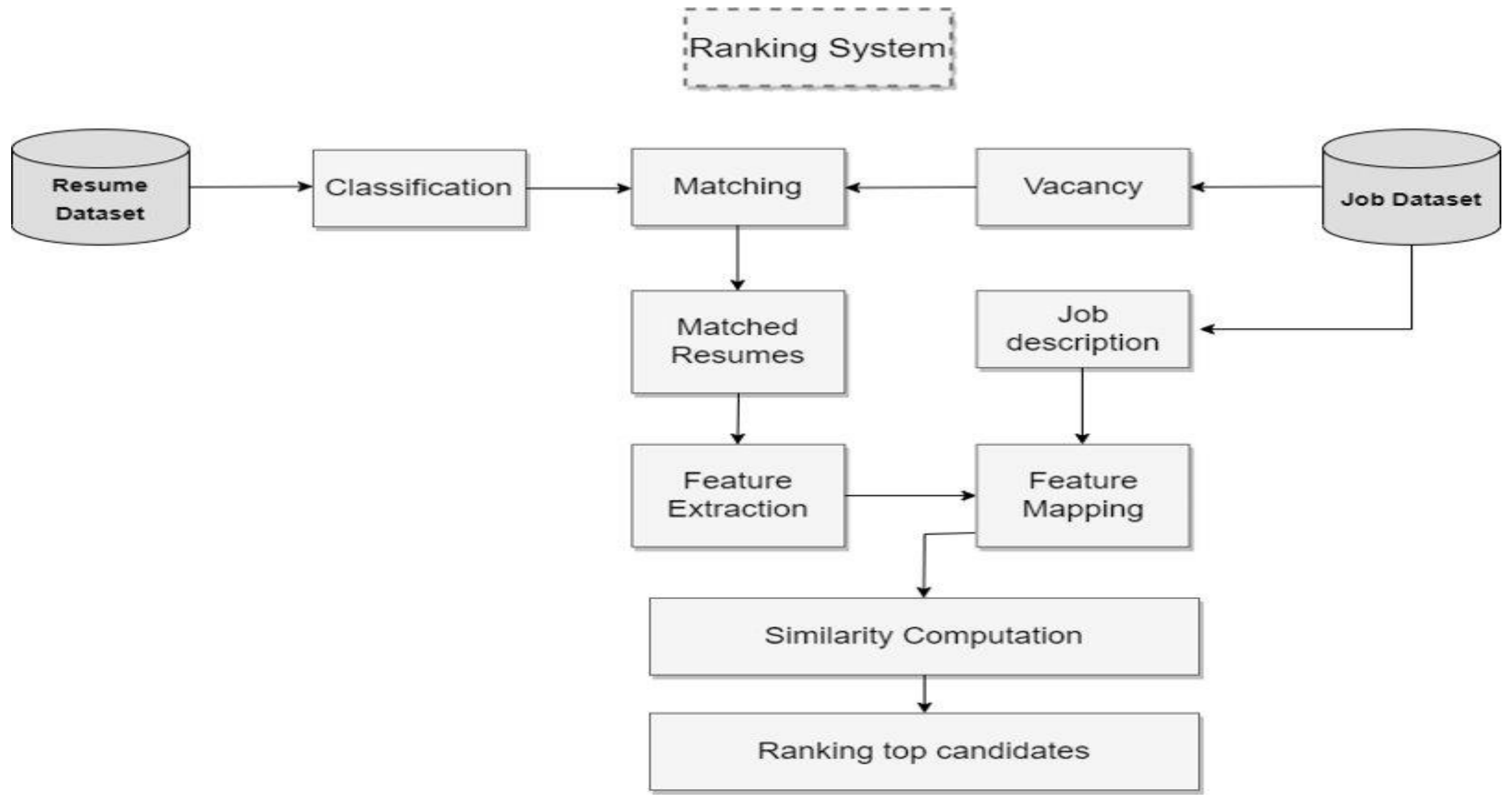
- Each resume is classified into each category.
- When a position opens up at that institution, our system immediately identifies the faculty members who are qualified and available to fill it.
- A system that will rank a candidate's resume in accordance with the job description. This system consists of resume classification and similarity computation according to the job description.

Project Goals

- Candidates receive an invitation to the interview via the application if their resume was selected for further consideration by the system.
- After that, they can choose to accept it or reject it; accepting it indicates that they are prepared to fill the position.
- The system provides ranked information about the candidates.
- This will enable recruiters to screen resumes in less time and with less effort.

Proposed System Architecture

Proposed System Architecture



Proposed System Architecture

- When a new resume is uploaded by a candidate, the KNN(K-nearest Neighbours)classifies by its category.
- Resumes are extracted from the dataset according to the identified vacancies by the system.
- Features are extracted from the resume and the job description with help of NLP (natural language processing).
- Each features are mapped and compute the similarity.
- Resumes are Ranked according to the score which given by the system.

Algorithms and highlight of the project features

Algorithms and highlight of the project features

- **KNN(K- nearest Neighbours):**
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K-NN algorithm.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Algorithms and highlight of the project features

- **Cosine similarity:**
- The similarity between two vectors in an inner product space is measured by cosine similarity.
- Determines whether two vectors are roughly pointing in the same direction by measuring the cosine of the angle between them.
- In text analysis, it is frequently used to gauge document similarity.
- Before calculating the cosine similarity between the job description and resumes, the model merges the cleaned resume data and job description into a single data set.

Summaries The Ultimate Findings of the Project

- **Resume Classification:**
- Using KNN algorithm the model can predict accurately.
- Training score : 99%
- Test Score : 98%
- **Resume Data Extraction:**
- Using NLP tools Spacy and nltk extracting the information from each resume which given as input.
- The skillExtraction class can extract skills from the resume.

Summaries The Ultimate Findings of the Project

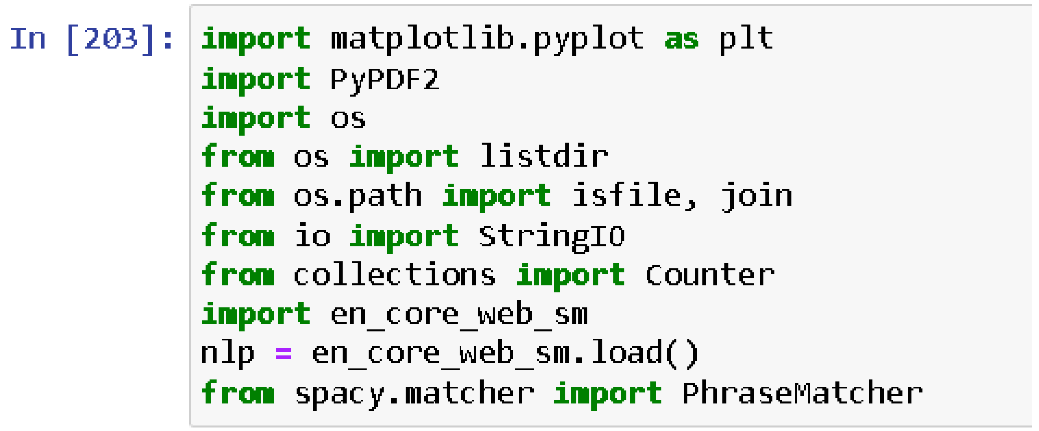
- **Matching Score:**
- Each resume compared with the main job description using cosine similarity.
- The algorithm give output as percentage value.
- **Skill Score:**
- Extracted skills from resume and job description
- Comparing these skills.
- The algorithm give output as percentage value.

Summaries The Ultimate Findings of the Project

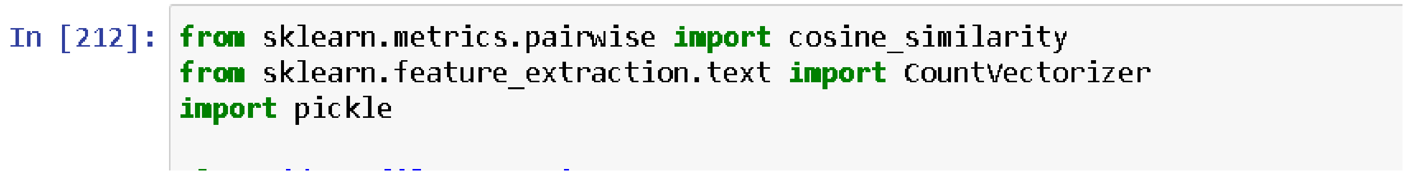
- **Data Base :**
- A table is created to save the output data with Name, Matching Score, Skill Score, Predicted Field as columns.
- **Ranking:**
- Ranking the resume according to the score gained by each algorithm

Explanation of Code

- **Necessary Packages**



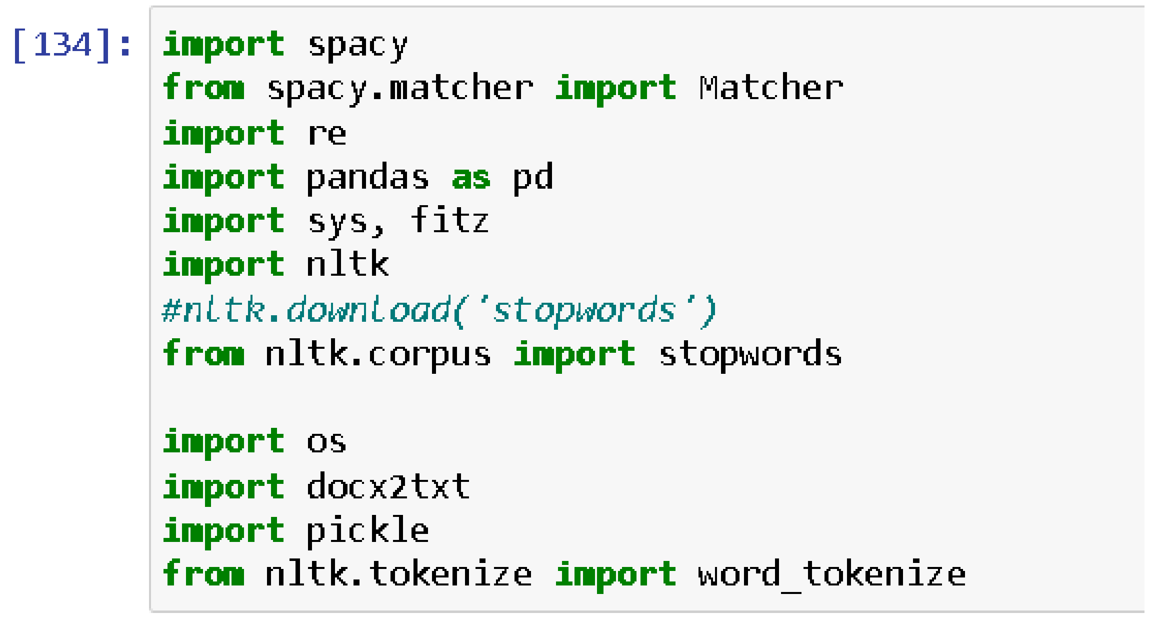
```
In [203]: import matplotlib.pyplot as plt
          import PyPDF2
          import os
          from os import listdir
          from os.path import isfile, join
          from io import StringIO
          from collections import Counter
          import en_core_web_sm
          nlp = en_core_web_sm.load()
          from spacy.matcher import PhraseMatcher
```



```
In [212]: from sklearn.metrics.pairwise import cosine_similarity
          from sklearn.feature_extraction.text import CountVectorizer
          import pickle
```

Explanation of Code

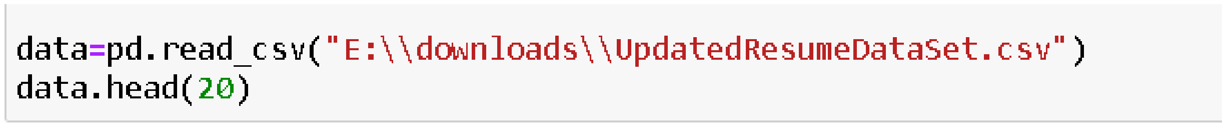
- **Necessary Packages**



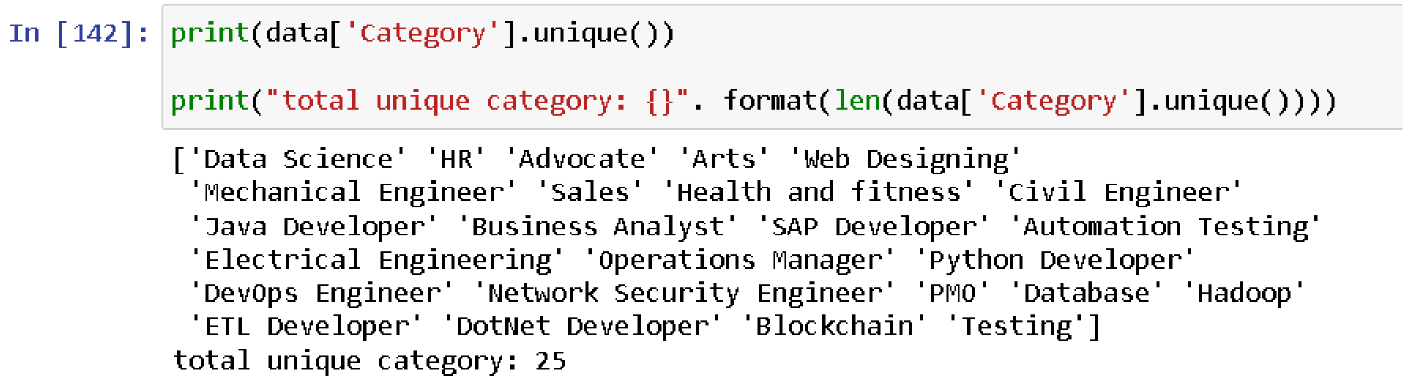
```
[134]: import spacy
       from spacy.matcher import Matcher
       import re
       import pandas as pd
       import sys, fitz
       import nltk
       #nltk.download('stopwords')
       from nltk.corpus import stopwords

       import os
       import docx2txt
       import pickle
       from nltk.tokenize import word_tokenize
```

Explanation of Code

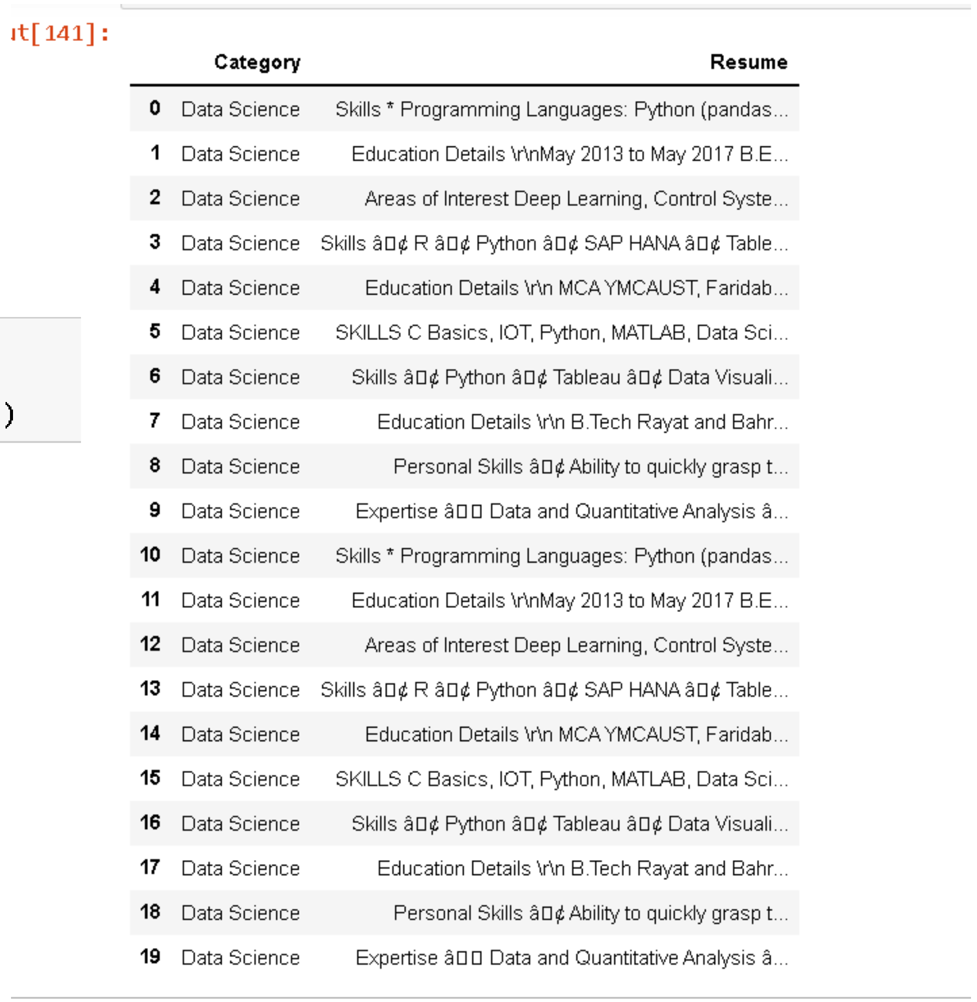
- **Data set**



```
data=pd.read_csv("E:\\downloads\\UpdatedResumeDataSet.csv")
data.head(20)
```



```
In [142]: print(data['Category'].unique())

          print("total unique category: {}". format(len(data['Category'].unique())))

          ['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']
          total unique category: 25
```

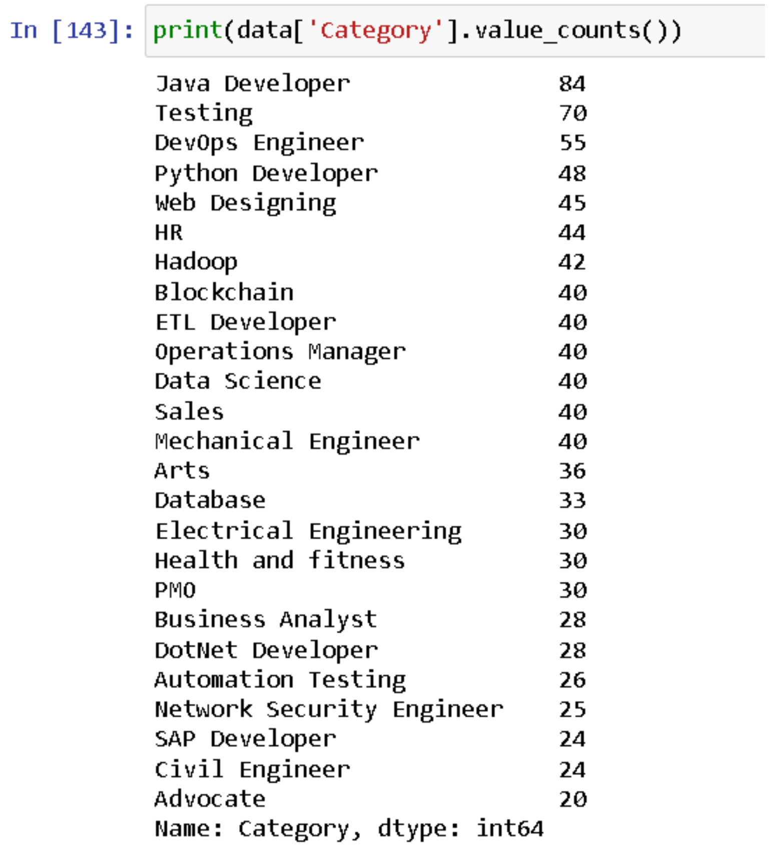


ıt[141]:

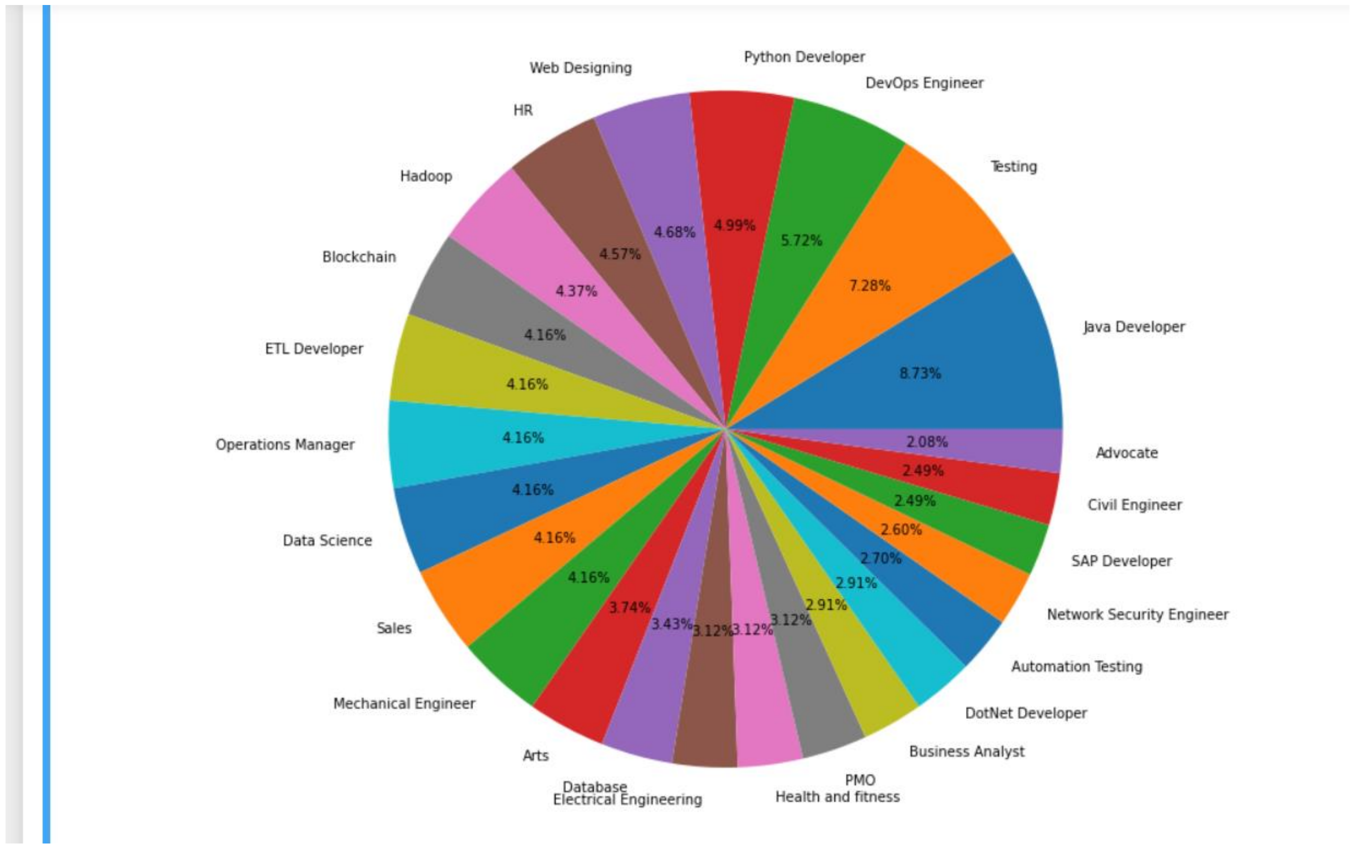
	Category	Resume
0	Data Science	Skills * Programming Languages: Python (pandas...
1	Data Science	Education Details \r\nMay 2013 to May 2017 B.E...
2	Data Science	Areas of Interest Deep Learning, Control Syste...
3	Data Science	Skills â□¢ R â□¢ Python â□¢ SAP HANA â□¢ Table...
4	Data Science	Education Details \r\n MCA YMCAUST, Faridab...
5	Data Science	SKILLS C Basics, IOT, Python, MATLAB, Data Sci...
6	Data Science	Skills â□¢ Python â□¢ Tableau â□¢ Data Visuali...
7	Data Science	Education Details \r\n B.Tech Rayat and Bahr...
8	Data Science	Personal Skills â□¢ Ability to quickly grasp t...
9	Data Science	Expertise â□□ Data and Quantitative Analysis â...
10	Data Science	Skills * Programming Languages: Python (pandas...
11	Data Science	Education Details \r\nMay 2013 to May 2017 B.E...
12	Data Science	Areas of Interest Deep Learning, Control Syste...
13	Data Science	Skills â□¢ R â□¢ Python â□¢ SAP HANA â□¢ Table...
14	Data Science	Education Details \r\n MCA YMCAUST, Faridab...
15	Data Science	SKILLS C Basics, IOT, Python, MATLAB, Data Sci...
16	Data Science	Skills â□¢ Python â□¢ Tableau â□¢ Data Visuali...
17	Data Science	Education Details \r\n B.Tech Rayat and Bahr...
18	Data Science	Personal Skills â□¢ Ability to quickly grasp t...
19	Data Science	Expertise â□□ Data and Quantitative Analysis â...

Explanation of Code

- **Data set**

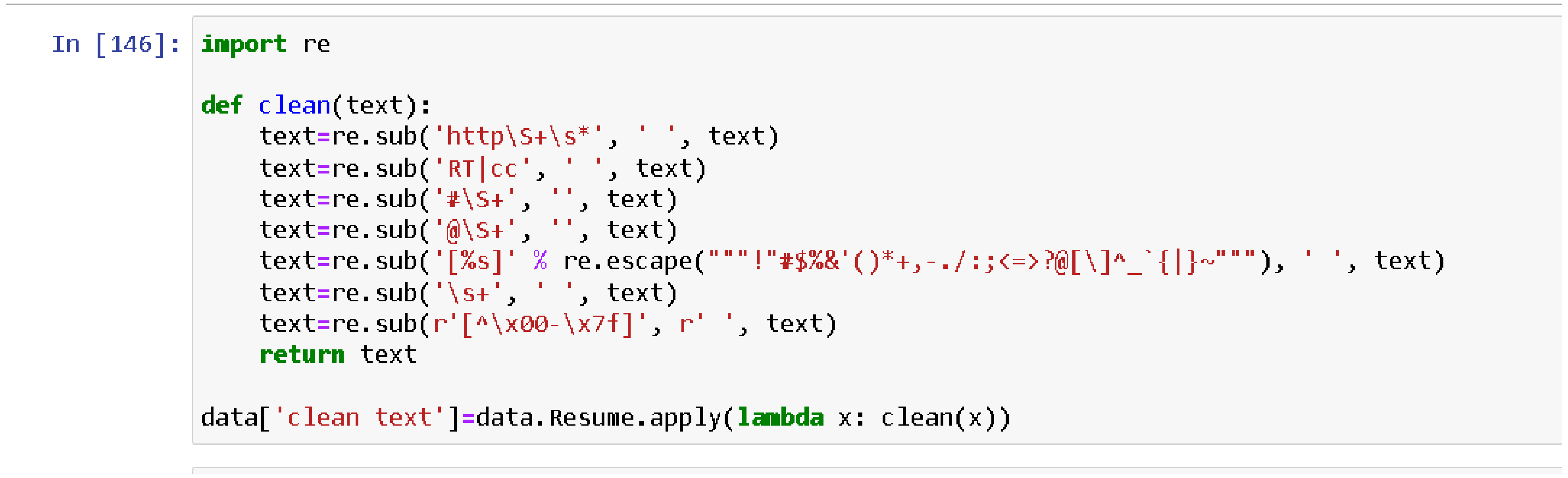


```
In [143]: print(data['Category'].value_counts())

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
```



Explanation of Code

- **Data Cleaning**

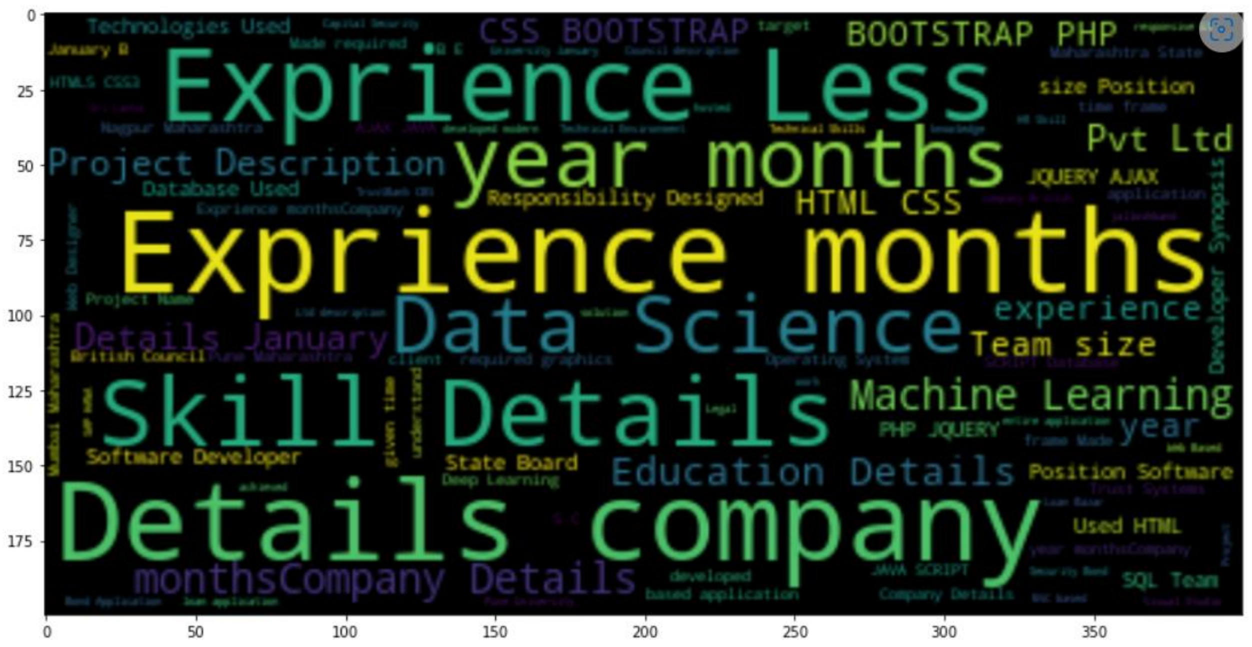


```
In [146]: import re

          def clean(text):
              text=re.sub('http\S+\s*', ' ', text)
              text=re.sub('RT|cc', ' ', text)
              text=re.sub('#\S+', '', text)
              text=re.sub('@\S+', '', text)
              text=re.sub('[%s]' % re.escape("""!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"""), ' ', text)
              text=re.sub('\s+', ' ', text)
              text=re.sub(r'[^\x00-\x7f]', r' ', text)
              return text

          data['clean text']=data.Resume.apply(lambda x: clean(x))
```

Explanation of Code

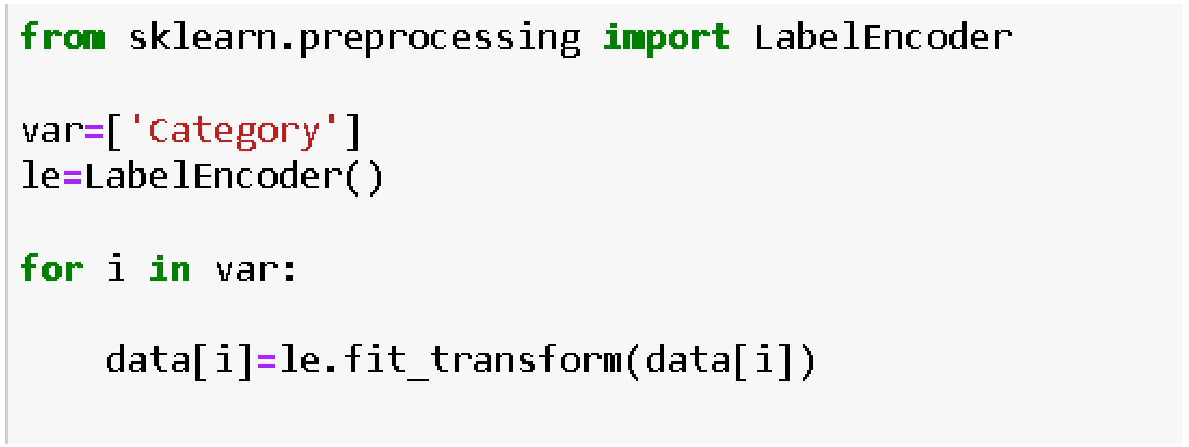
- **Word Cloud**

```
[148]: WC=WordCloud().generate(cleanSentences)
       plt.figure(figsize=(15,15))
       plt.imshow(WC, interpolation='bilinear')

t[148]: <matplotlib.image.AxesImage at 0x23afe80f4f0>
```

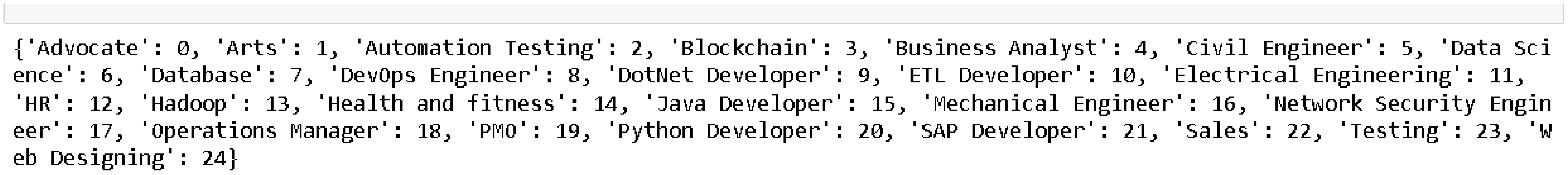


Explanation of Code

- **Label Encoding.**



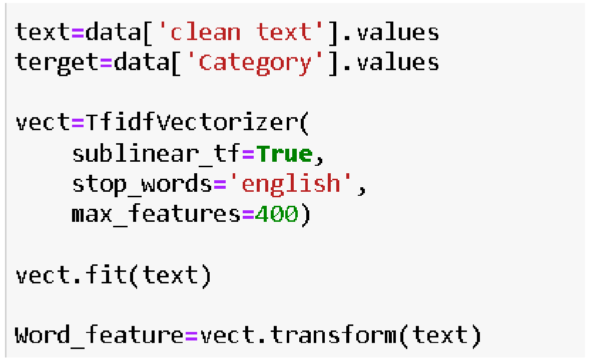
```
from sklearn.preprocessing import LabelEncoder

var=['Category']
le=LabelEncoder()

for i in var:

    data[i]=le.fit_transform(data[i])
```



```
{'Advocate': 0, 'Arts': 1, 'Automation Testing': 2, 'Blockchain': 3, 'Business Analyst': 4, 'Civil Engineer': 5, 'Data Sci
ence': 6, 'Database': 7, 'DevOps Engineer': 8, 'DotNet Developer': 9, 'ETL Developer': 10, 'Electrical Engineering': 11,
'HR': 12, 'Hadoop': 13, 'Health and fitness': 14, 'Java Developer': 15, 'Mechanical Engineer': 16, 'Network Security Engin
eer': 17, 'Operations Manager': 18, 'PMO': 19, 'Python Developer': 20, 'SAP Developer': 21, 'Sales': 22, 'Testing': 23, 'W
eb Designing': 24}
```

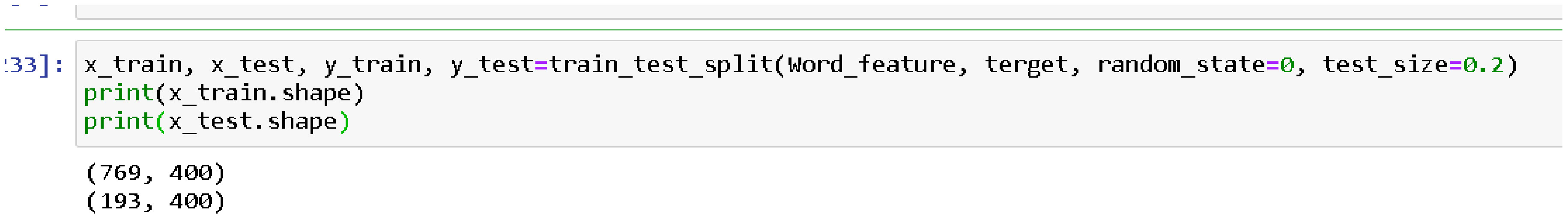
Explanation of Code

- **Vectorization.**



```
text=data['clean text'].values
terget=data['Category'].values

vect=TfidfVectorizer(
    sublinear_tf=True,
    stop_words='english',
    max_features=400)

vect.fit(text)

Word_feature=vect.transform(text)
```

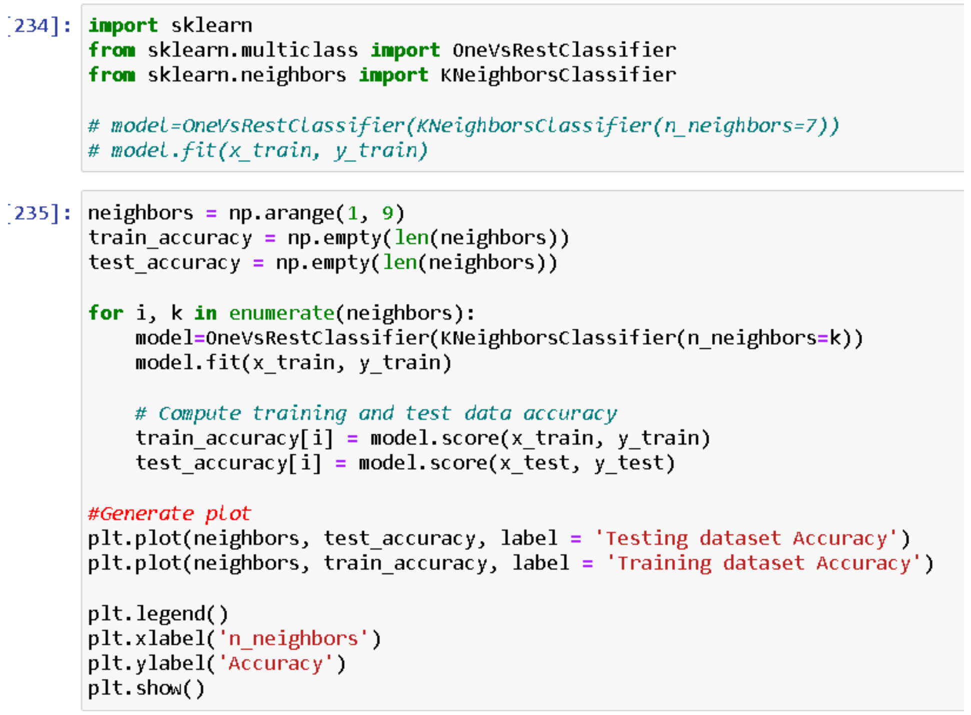
- Splitting.



```
:33]: x_train, x_test, y_train, y_test=train_test_split(Word_feature, terget, random_state=0, test_size=0.2)
      print(x_train.shape)
      print(x_test.shape)

      (769, 400)
      (193, 400)
```

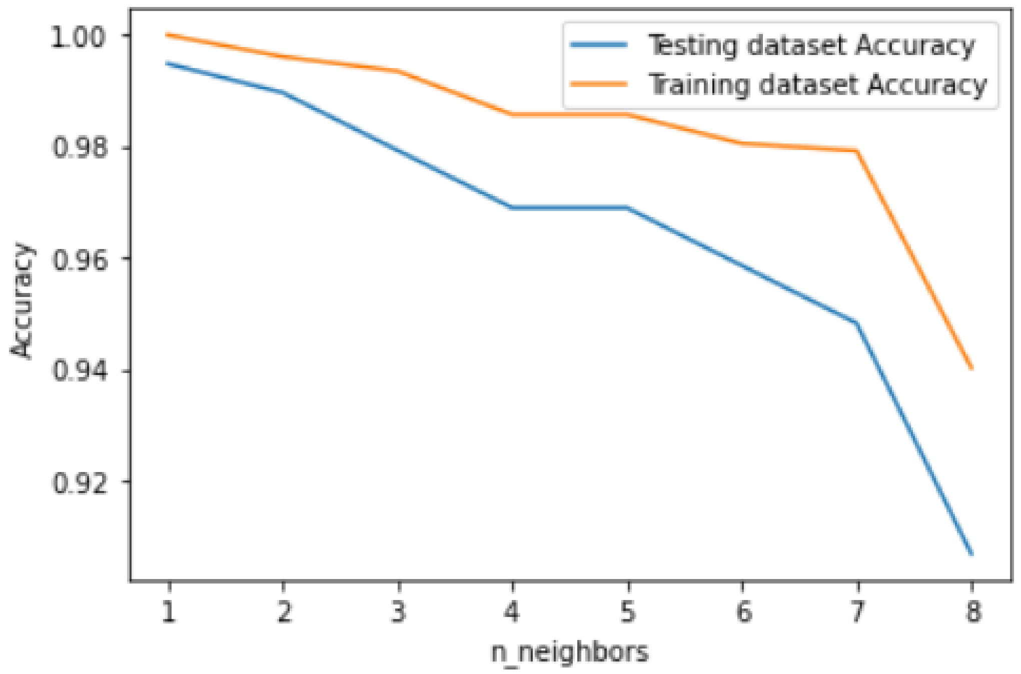
Explanation of Code

Model



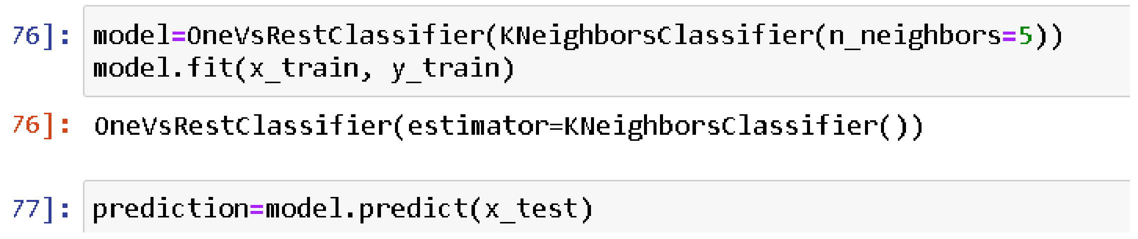
```
[234]: import sklearn
       from sklearn.multiclass import OneVsRestClassifier
       from sklearn.neighbors import KNeighborsClassifier

       # model=OneVsRestClassifier(KNeighborsClassifier(n_neighbors=7))
       # model.fit(x_train, y_train)
```

```
[235]: neighbors = np.arange(1, 9)
       train_accuracy = np.empty(len(neighbors))
       test_accuracy = np.empty(len(neighbors))

       for i, k in enumerate(neighbors):
           model=OneVsRestClassifier(KNeighborsClassifier(n_neighbors=k))
           model.fit(x_train, y_train)

           # Compute training and test data accuracy
           train_accuracy[i] = model.score(x_train, y_train)
           test_accuracy[i] = model.score(x_test, y_test)

       #Generate plot
       plt.plot(neighbors, test_accuracy, label = 'Testing dataset Accuracy')
       plt.plot(neighbors, train_accuracy, label = 'Training dataset Accuracy')

       plt.legend()
       plt.xlabel('n_neighbors')
       plt.ylabel('Accuracy')
       plt.show()
```

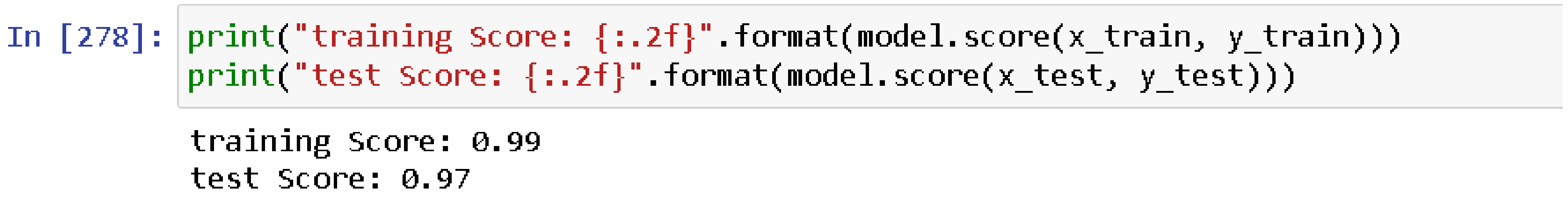


Explanation of Code

Model



```
76]: model=OneVsRestClassifier(KNeighborsClassifier(n_neighbors=5))
     model.fit(x_train, y_train)

76]: OneVsRestClassifier(estimator=KNeighborsClassifier())

77]: prediction=model.predict(x_test)
```

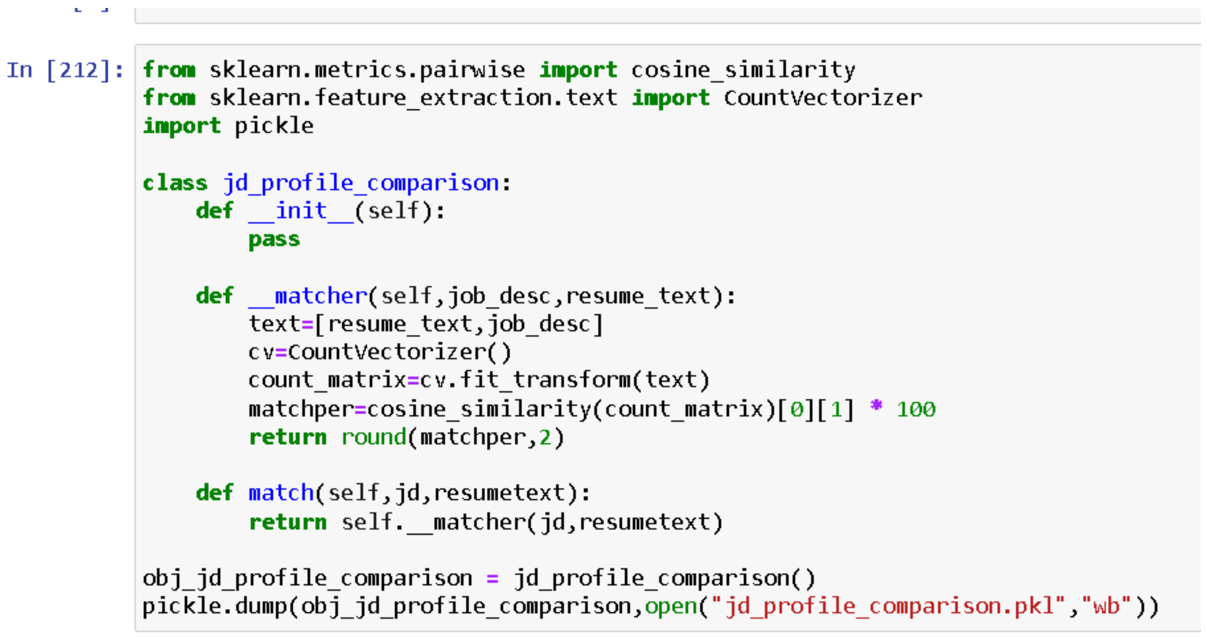


```
In [278]: print("training Score: {:.2f}".format(model.score(x_train, y_train)))
          print("test Score: {:.2f}".format(model.score(x_test, y_test)))

          training Score: 0.99
          test Score: 0.97
```

Explanation of Code

Matching

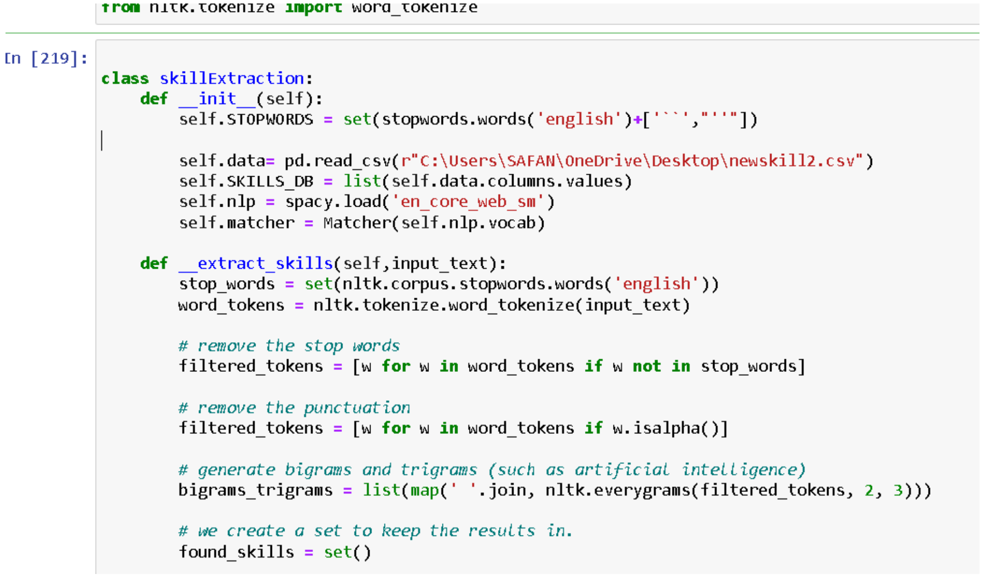
•



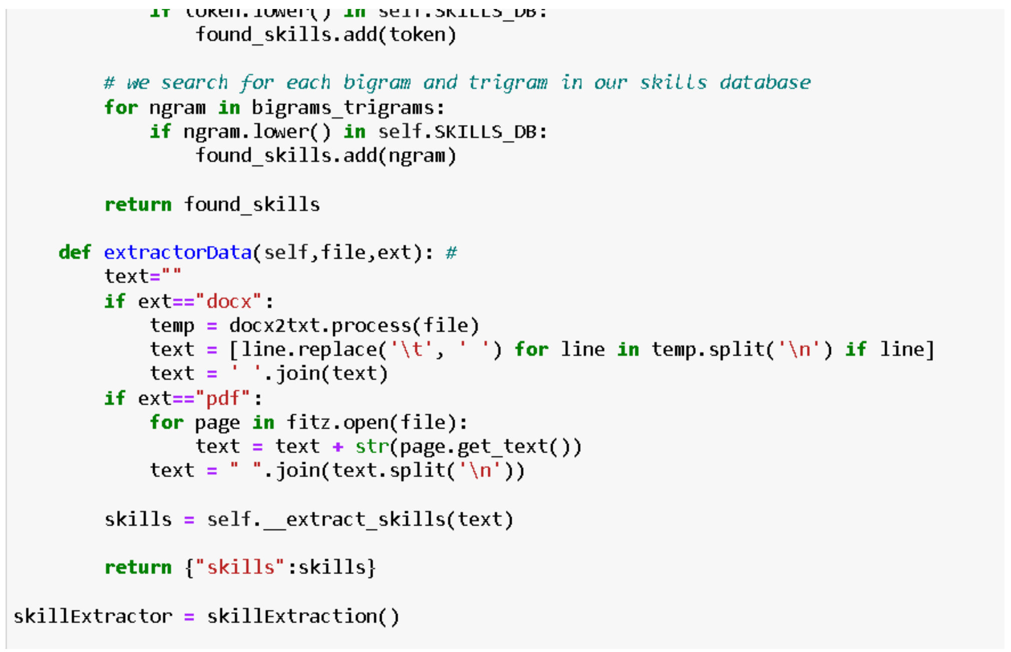
```
In [212]: from sklearn.metrics.pairwise import cosine_similarity
          from sklearn.feature_extraction.text import CountVectorizer
          import pickle

          class jd_profile_comparison:
              def __init__(self):
                  pass

              def __matcher(self,job_desc,resume_text):
                  text=[resume_text,job_desc]
                  cv=CountVectorizer()
                  count_matrix=cv.fit_transform(text)
                  matchper=cosine_similarity(count_matrix)[0][1] * 100
                  return round(matchper,2)

              def match(self,jd,resumetext):
                  return self.__matcher(jd,resumetext)

          obj_jd_profile_comparison = jd_profile_comparison()
          pickle.dump(obj_jd_profile_comparison,open("jd_profile_comparison.pkl","wb"))
```

Explanation of Code

Skill Extraction



```
from nltk.tokenize import word_tokenize

In [219]:

class skillExtraction:
    def __init__(self):
        self.STOPWORDS = set(stopwords.words('english')+['``',"''"])

        self.data= pd.read_csv(r"C:\Users\SAFAN\OneDrive\Desktop\newskill2.csv")
        self.SKILLS_DB = list(self.data.columns.values)
        self.nlp = spacy.load('en_core_web_sm')
        self.matcher = Matcher(self.nlp.vocab)

    def __extract_skills(self,input_text):
        stop_words = set(nltk.corpus.stopwords.words('english'))
        word_tokens = nltk.tokenize.word_tokenize(input_text)

        # remove the stop words
        filtered_tokens = [w for w in word_tokens if w not in stop_words]

        # remove the punctuation
        filtered_tokens = [w for w in word_tokens if w.isalpha()]

        # generate bigrams and trigrams (such as artificial intelligence)
        bigrams_trigrams = list(map(' '.join, nltk.everygrams(filtered_tokens, 2, 3)))

        # we create a set to keep the results in.
        found_skills = set()
```

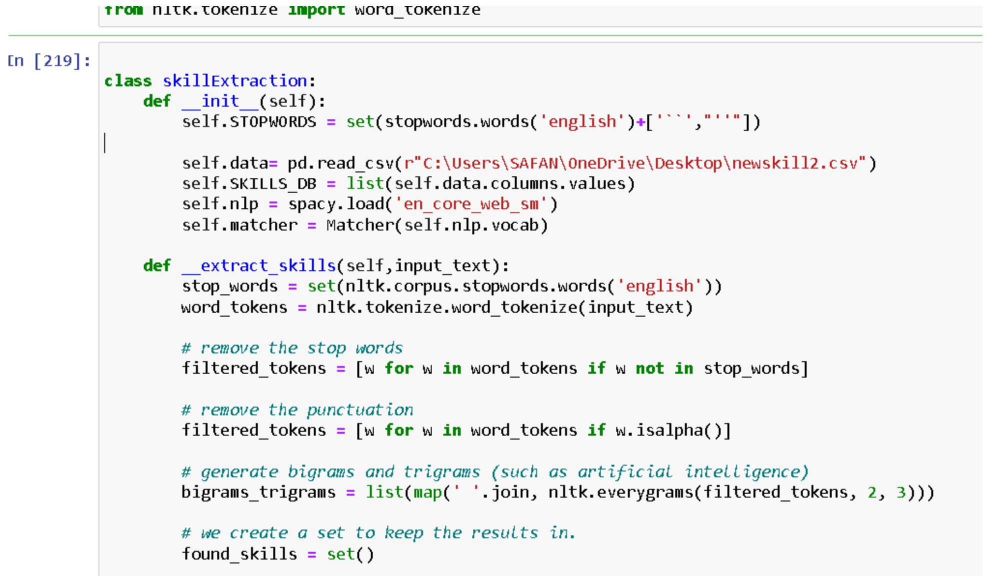


```
            if token.lower() in self.SKILLS_DB:
                found_skills.add(token)

        # we search for each bigram and trigram in our skills database
        for ngram in bigrams_trigrams:
            if ngram.lower() in self.SKILLS_DB:
                found_skills.add(ngram)

        return found_skills

    def extractorData(self,file,ext): #
        text=""
        if ext=="docx":
            temp = docx2txt.process(file)
            text = [line.replace('\t', ' ') for line in temp.split('\n') if line]
            text = ' '.join(text)
        if ext=="pdf":
            for page in fitz.open(file):
                text = text + str(page.get_text())
            text = " ".join(text.split('\n'))

        skills = self.__extract_skills(text)

        return {"skills":skills}

skillExtractor = skillExtraction()
```

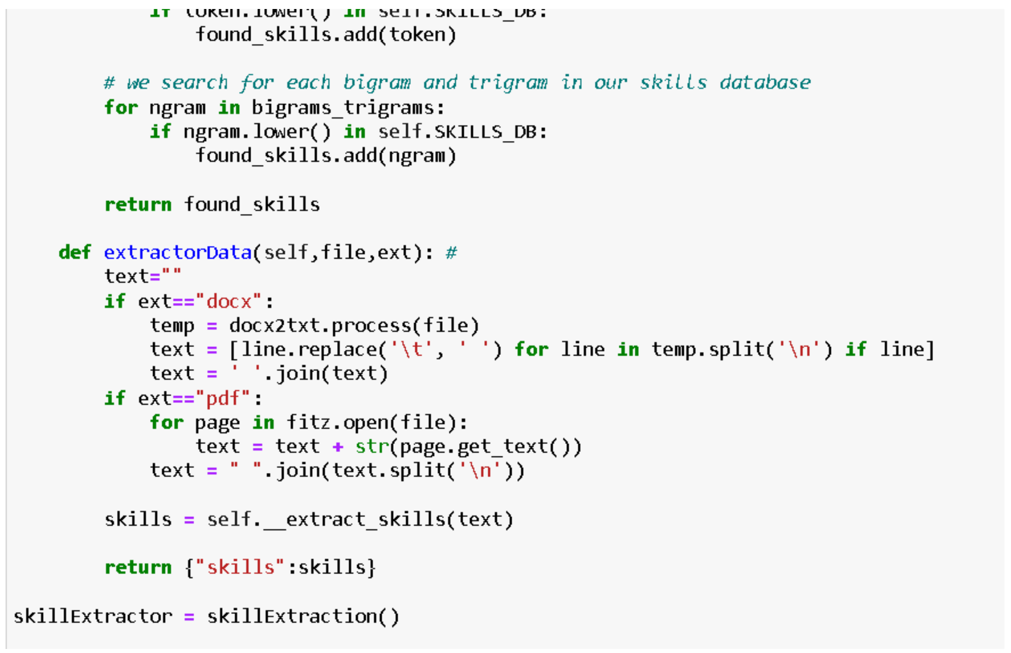
Explanation of Code

Skill Extraction

•



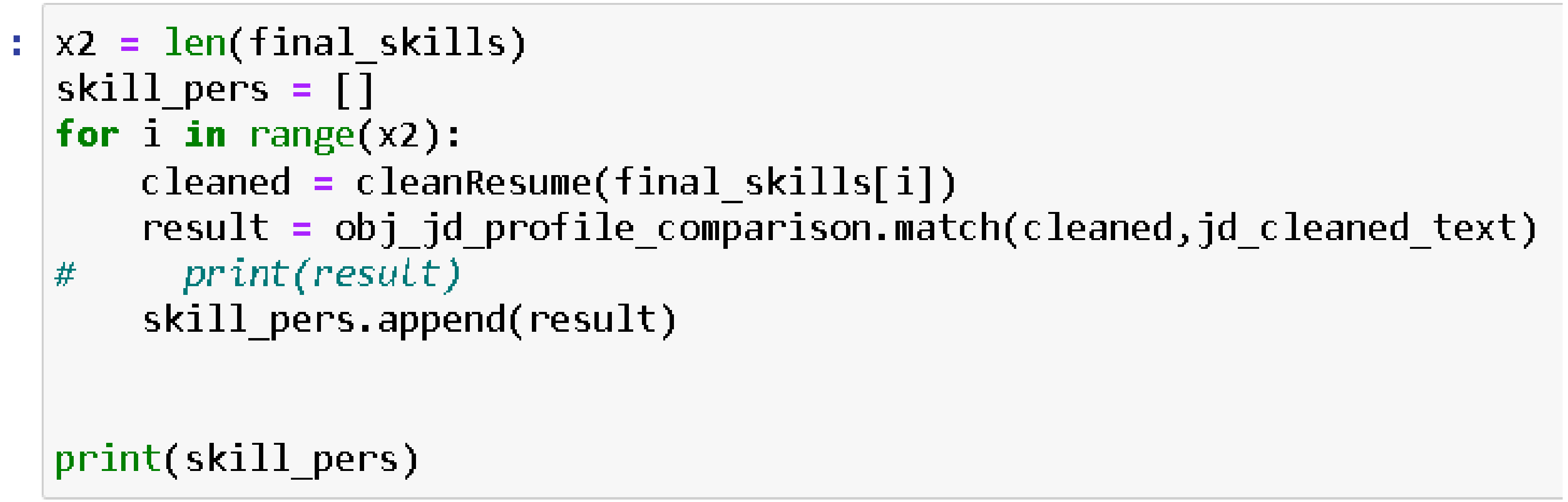
```
from nltk.tokenize import word_tokenize

In [219]:
class skillExtraction:
    def __init__(self):
        self.STOPWORDS = set(stopwords.words('english')+['``',"''"])

        self.data= pd.read_csv(r"C:\Users\SAFAN\OneDrive\Desktop\newskill2.csv")
        self.SKILLS_DB = list(self.data.columns.values)
        self.nlp = spacy.load('en_core_web_sm')
        self.matcher = Matcher(self.nlp.vocab)

    def __extract_skills(self,input_text):
        stop_words = set(nltk.corpus.stopwords.words('english'))
        word_tokens = nltk.tokenize.word_tokenize(input_text)

        # remove the stop words
        filtered_tokens = [w for w in word_tokens if w not in stop_words]

        # remove the punctuation
        filtered_tokens = [w for w in word_tokens if w.isalpha()]

        # generate bigrams and trigrams (such as artificial intelligence)
        bigrams_trigrams = list(map(' '.join, nltk.everygrams(filtered_tokens, 2, 3)))

        # we create a set to keep the results in.
        found_skills = set()
```



```
            if token.lower() in self.SKILLS_DB:
                found_skills.add(token)

        # we search for each bigram and trigram in our skills database
        for ngram in bigrams_trigrams:
            if ngram.lower() in self.SKILLS_DB:
                found_skills.add(ngram)

        return found_skills

    def extractorData(self,file,ext): #
        text=""
        if ext=="docx":
            temp = docx2txt.process(file)
            text = [line.replace('\t', ' ') for line in temp.split('\n') if line]
            text = ' '.join(text)
        if ext=="pdf":
            for page in fitz.open(file):
                text = text + str(page.get_text())
            text = " ".join(text.split('\n'))

        skills = self.__extract_skills(text)

        return {"skills":skills}

skillExtractor = skillExtraction()
```

Explanation of Code

Skill Comparison

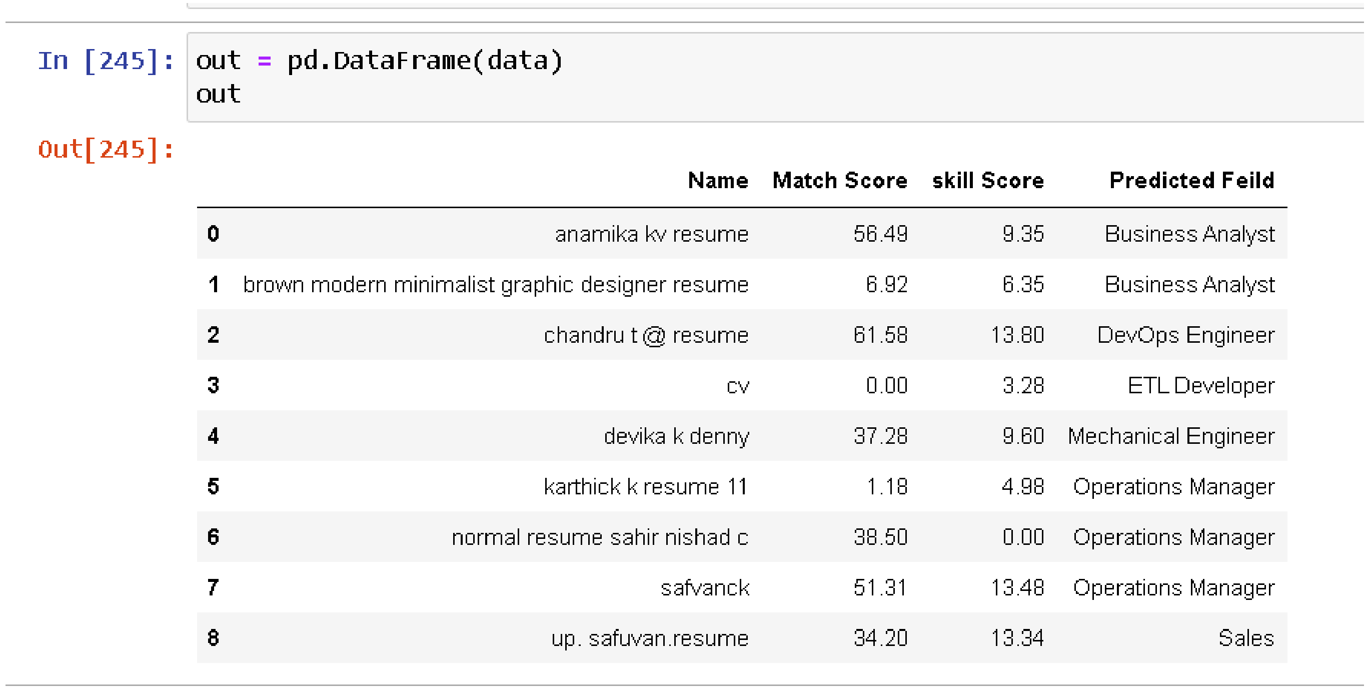


```
:  x2 = len(final_skills)
   skill_pers = []
   for i in range(x2):
       cleaned = cleanResume(final_skills[i])
       result = obj_jd_profile_comparison.match(cleaned,jd_cleaned_text)
   #     print(result)
       skill_pers.append(result)


   print(skill_pers)
```

•

Explanation of Code

Skill Comparison

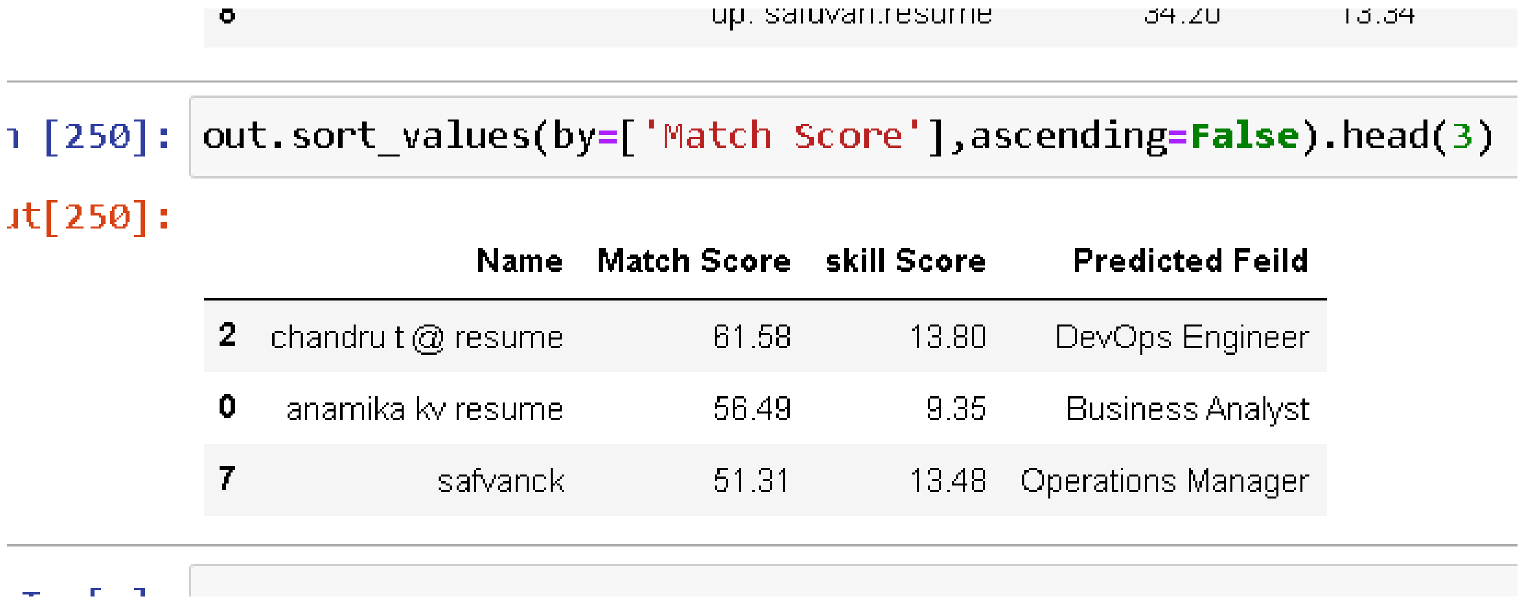


```
In [245]: out = pd.DataFrame(data)
          out
Out[245]:
```

	Name	Match Score	skill Score	Predicted Feild
0	anamika kv resume	56.49	9.35	Business Analyst
1	brown modern minimalist graphic designer resume	6.92	6.35	Business Analyst
2	chandru t @ resume	61.58	13.80	DevOps Engineer
3	cv	0.00	3.28	ETL Developer
4	devika k denny	37.28	9.60	Mechanical Engineer
5	karthick k resume 11	1.18	4.98	Operations Manager
6	normal resume sahir nishad c	38.50	0.00	Operations Manager
7	safvanck	51.31	13.48	Operations Manager
8	up. safuvan.resume	34.20	13.34	Sales

Explanation of Code

Top 3 candidates



```
n [250]: out.sort_values(by=['Match Score'],ascending=False).head(3)
ut[250]:
```

	Name	Match Score	skill Score	Predicted Feild
2	chandru t @ resume	61.58	13.80	DevOps Engineer
0	anamika kv resume	56.49	9.35	Business Analyst
7	safvanck	51.31	13.48	Operations Manager

Thank you