Northeastern University
College *of* Professional Studies

FUNDAMENTALS OF ARTIFICIAL INTELLIGENCE

EAI 6000, CRN 80527

PROFESSOR SERGIY SHEVCHENKO

FINAL PROJECT
RESUME CLASSIFICATION & PARSER

SUBMITTED BY:

Richa Umesh Rambhia

Sakshi Grover

Merlyn D’Souza

Abstract

A resume is a brief summary of your skills and experience. Companies' recruiters and HR teams have a tough time scanning thousands of qualified resumes. They either need many people to do this or miss out on qualified candidates. Spending too many labor hours segregating candidates' resume's manually is a waste of a company's time, money, and productivity. Recruiters, therefore, use resume parsers in order to streamline the resume and applicant screening process. Parsing technology allows recruiters to electronically gather, store, and organize large quantities of resumes. Once acquired, the resume data can be easily searched through and analyzed.

Resumes are an ideal example of unstructured data. Since there is no widely accepted resume layout, each resume may have its own style of formatting, different text blocks and different category titles. Building a resume parser is no easy task as there are so many kinds of layouts of resumes that you could imagine. For example, some people would put the date in front of the title of the resume, some people do not put the duration of the work experience, some people do not list down the company in their resumes.

In this project we dive into building a parser tool using Python and basic natural language processing techniques. We would be using Python's libraries to implement various NLP (natural language processing) techniques like tokenization, lemmatization, parts of speech tagging, etc., for building a resume parser in Python. The source of dataset is Kaggle for which initial exploratory data analysis is performed to get an understanding of the dataset and clean the data if required in order to avoid errors in the further modelling process. Data visualization gives insights about the data and helps in understanding the parameters of the dataset further helping to build a model to predict or classify the categories of the resume text and also build a parser for the same.

Table of Contents

Introduction

A resume parsing technology needs to be implemented in order to make it easy for the companies to process the huge number of resumes that are received by the organizations. This technology converts an unstructured form of resume data into a structured data format. The resumes received are in the form of documents from which the data needs to be extracted first such that the text can be classified or predicted based on the requirements. A resume parser analyzes resume data and extracts the information into the machine readable output. It helps automatically store, organize, and analyze the resume data to find out the best candidate for the particular job position and requirements. This thus helps the organizations eliminate the error-prone and time-consuming process of going through thousands of resumes manually and aids in improving the recruiters' efficiency.

Resume screening using machine learning is mostly used by each company in order to go through the resumes of the candidates as they do not have enough time to go through each resume manually and make a decision out of it. In order to perform this task and implement the resume parser system, the dataset needs to be first analyzed in order to get an overview of the data available for model building. The dataset obtained is from Kaggle which deals with resume category classification and for the resume parser system a sample resume is collected for which the model is implemented and works when any resume in a PDF format is passed to it. The basic data analysis process is performed such as data collection, data cleaning, exploratory data analysis, data visualization, and model building. The dataset consists of two columns, namely, Category and Resume, where category is the domain field of the industry and resume column consists of the text extracted from the resume document for each domain and industry.

The aim of this project is achieved by performing the various data analytical methods and using the machine learning models and natural language processing which will help in classifying the categories of the resume and building the resume parser. Thus, the objective of this project will help in developing a system using the ML models which would aid in solving the real-life problems which would help the companies in selecting the most favorable and least favorable resume of the candidate.

Statement Of Purpose

The project aims to analyze the resume data to gauge and classify the categories of the resumes of the candidates and also parse the resume to predict the most favorable and least favorable resume of the candidate which would help the companies select the best fit candidate for a particular position. In order to achieve this goal, the primary task is to analyze the data and build a classification model to classify the categories of the resume. Another part of this task is to predict the favorable resume of the candidate which would be developed by building a machine learning model and using the natural language processing to simplify the text classification process.

The business question that would be addressed here is that based on the text and skills extracted from the resume, the system will be able to categorize the resume into its respective categories of the field and also predict the resume which would be the most suitable for the position based on the category classified. It is important that the resume document received by each organization is taken into account which is not possible to be performed manually and thus this system would help in categorizing and predicting the best fit resume which saves most of the time of the company.

In order to deal with this task, the goal is to first categorize the resume data into its respective categories where the dataset would be split into training and testing sets and accuracy of the model would be gauged to analyze which model would be the best fit for this task. The resume screening or parser would be developed based on the resume document received from which the data and information needed would be extracted to further train the model and obtain the required results.

Exploratory Data Analysis

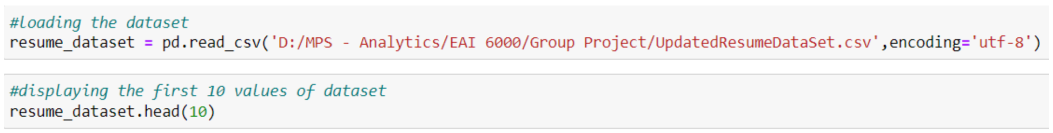
In order to analyze the resume dataset and get insights from it, descriptive analysis and exploratory data analysis methods are performed. The dataset consists of categorical data consisting of parameters, namely, Category and Resume which are described as the field name and skills extracted from the resume respectively.

The first step of the analysis performed in order to achieve the insights from the resume data as mentioned is descriptive analysis. Here the steps performed include loading the dataset in the data frame, describing the data frame values, and getting information about the statistical values of the dataset. Since the descriptive analysis of the given data needs to be performed in order to understand the dataset for further analysis, it is important to display the statistical values of the dataset, and rather performing individual steps to get the desired results, data profiling step is performed which gives a detailed report of the dataset.

Thus, for the given problem statement, along with the individual descriptive and statistical analysis, data profiling is also performed which gave a detailed report of the data including the statistical values, missing values, visualization of the dataset, correlation plot of the variables, etc.

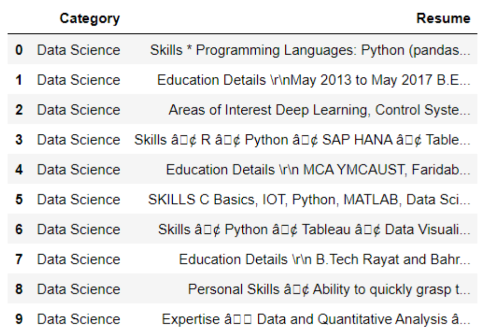
Descriptive Analysis:

a. <u>Loading the Dataset</u>



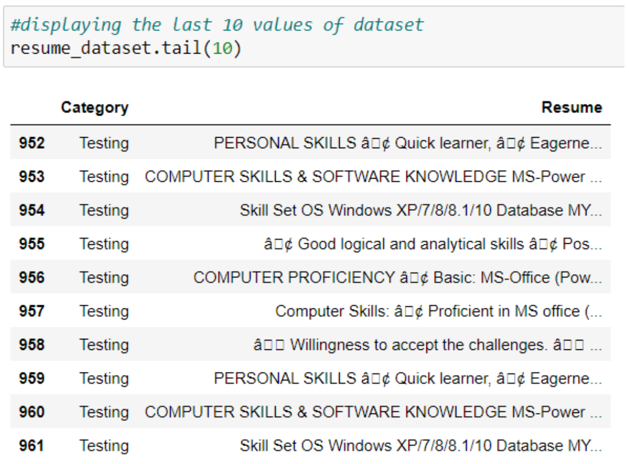
```
#loading the dataset
resume_dataset = pd.read_csv('D:/MPS - Analytics/EAI 6000/Group Project/UpdatedResumeDataSet.csv',encoding='utf-8')
```

```
#displaying the first 10 values of dataset
resume_dataset.head(10)
```



	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...

Here, the starting records of the dataset are displayed which are loaded in the data frame.

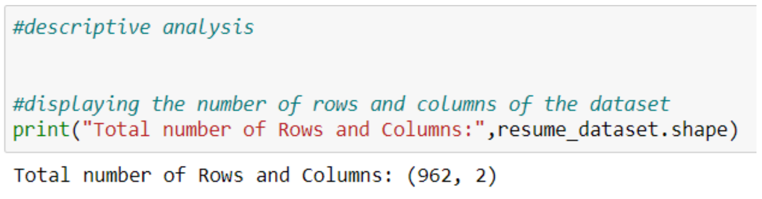


```
#displaying the last 10 values of dataset
resume_dataset.tail(10)
```

	Category	Resume
952	Testing	PERSONAL SKILLS â□¢ Quick learner, â□¢ Eagerne...
953	Testing	COMPUTER SKILLS & SOFTWARE KNOWLEDGE MS-Power ...
954	Testing	Skill Set OS Windows XP/7/8/8.1/10 Database MY...
955	Testing	â□¢ Good logical and analytical skills â□¢ Pos...
956	Testing	COMPUTER PROFICIENCY â□¢ Basic: MS-Office (Pow...
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...

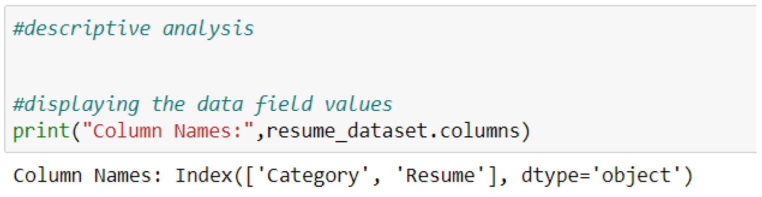
The end records of the dataset are displayed which are loaded in the data frame.

b. <u>Describing the Dataset</u>



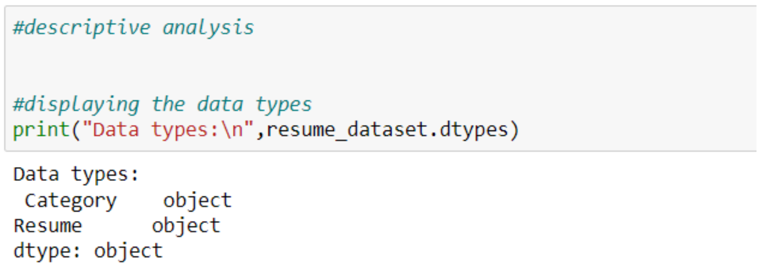
```
#descriptive analysis

#displaying the number of rows and columns of the dataset
print("Total number of Rows and Columns:",resume_dataset.shape)
```

```
Total number of Rows and Columns: (962, 2)
```



```
#descriptive analysis

#displaying the data field values
print("Column Names:",resume_dataset.columns)
```

```
Column Names: Index(['Category', 'Resume'], dtype='object')
```

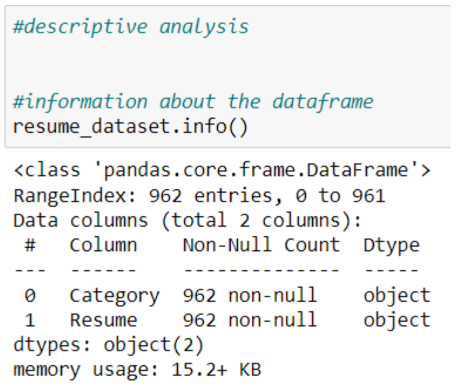


```
#descriptive analysis

#displaying the data types
print("Data types:\n",resume_dataset.dtypes)
```

```
Data types:
 Category    object
Resume      object
dtype: object
```

The descriptive analysis is performed on the dataset where the total number of rows and columns are displayed along with the column names and the type of data of each field value, as shown in the figure above.

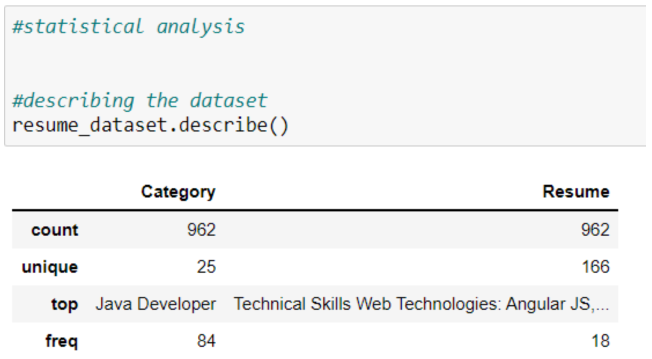
c. Information about the data frame



```
#descriptive analysis

#information about the dataframe
resume_dataset.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
```

d. Statistical Analysis of the Dataset



```
#statistical analysis

#describing the dataset
resume_dataset.describe()
```

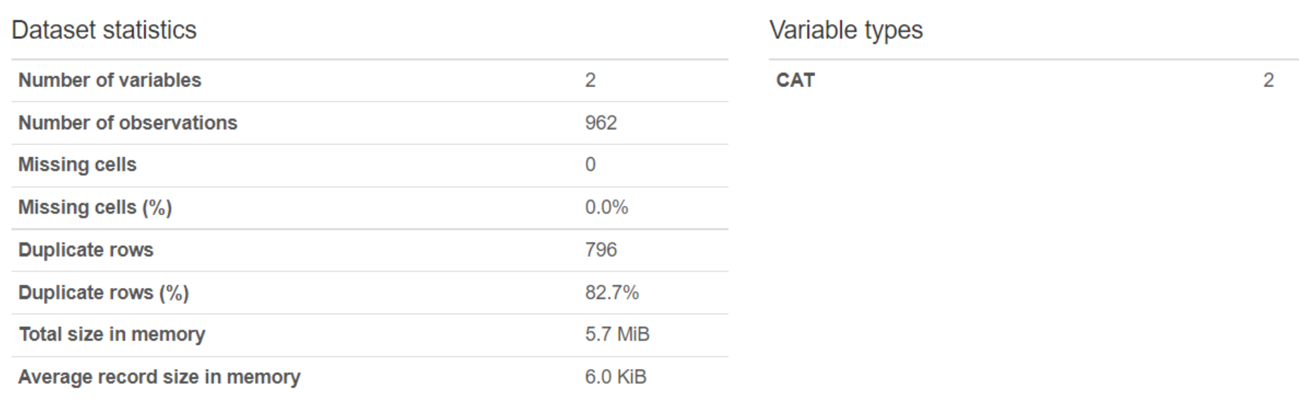
	Category	Resume
count	962	962
unique	25	166
top	Java Developer	Technical Skills Web Technologies: Angular JS,...
freq	84	18

Data Profiling:

The data profiling report gives an overview of the dataset and the data frame which includes reports of the statistics of each of the variables in the dataset, the dataset statistics, the missing data plot, and the correlation plot of the parameters. This is thus useful in deriving an overall analysis of the dataset and the variables of the dataset which can be further considered during the analysis and modeling process. The analysis from the data profiling report is as follows.

a. Dataset Statistics

The dataset statistics gives an overview of the data with respect to the number of field values, number of row values, missing values, duplicate entries, and the information related to the memory. The type of variables present in the dataset is also mentioned in the dataset statistics of the data profile report.



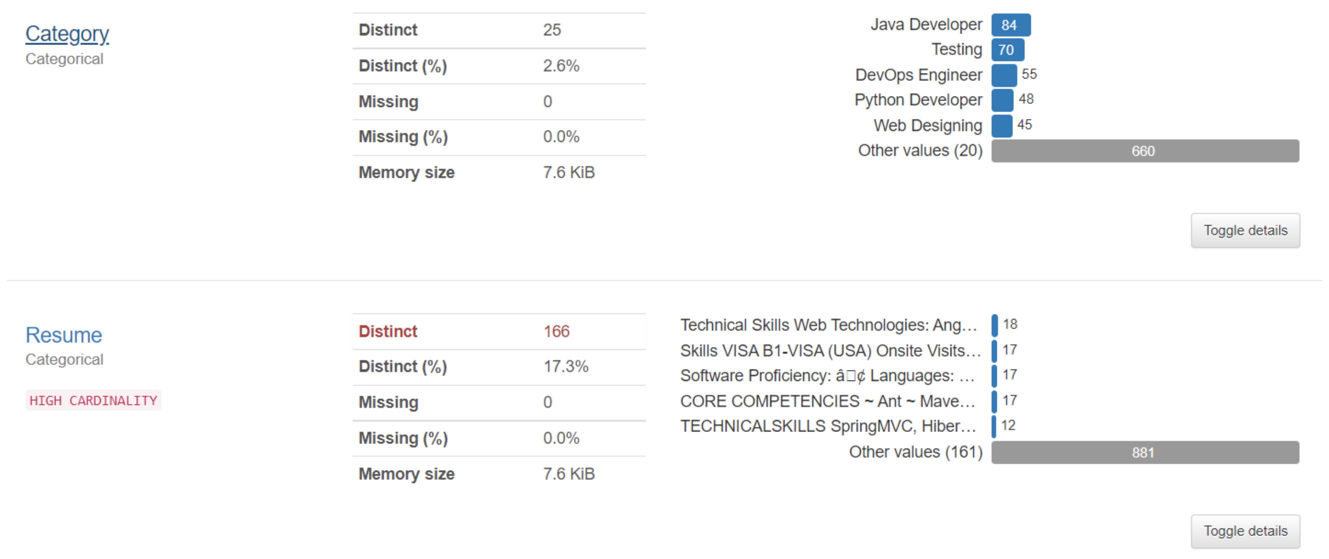
Dataset statistics

Number of variables	2
Number of observations	962
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	796
Duplicate rows (%)	82.7%
Total size in memory	5.7 MiB
Average record size in memory	6.0 KiB

Variable types

CAT	2

b. Variable Analysis



Category
Categorical

Distinct	25
Distinct (%)	2.6%
Missing	0
Missing (%)	0.0%
Memory size	7.6 KiB

Java Developer	84
Testing	70
DevOps Engineer	55
Python Developer	48
Web Designing	45
Other values (20)	660

Resume
Categorical
HIGH CARDINALITY

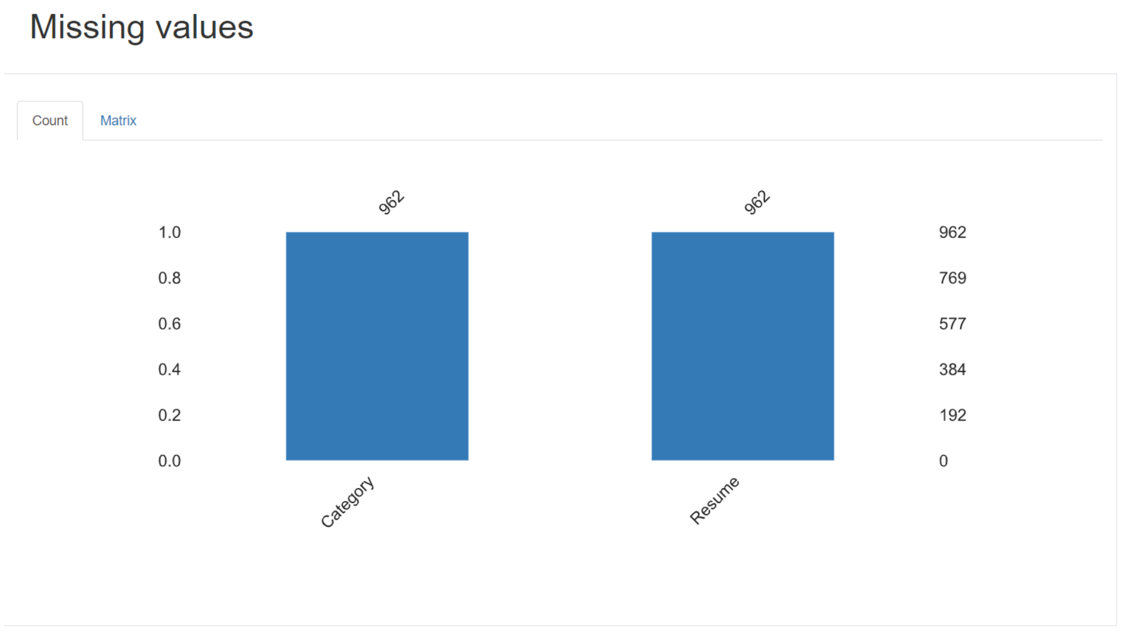
Distinct	166
Distinct (%)	17.3%
Missing	0
Missing (%)	0.0%
Memory size	7.6 KiB

Technical Skills Web Technologies: Ang…	18
Skills VISA B1-VISA (USA) Onsite Visits…	17
Software Proficiency: â□¢ Languages: …	17
CORE COMPETENCIES ~ Ant ~ Mave…	17
TECHNICALSKILLS SpringMVC, Hiber…	12
Other values (161)	881

The variable analysis gives an overview of each of the variable that is present in the dataset with respect to the frequency count, missing values, distinct values, etc., which is as displayed in the above figure.

Data Cleaning:

Data cleaning is one of the most important phases of the data analysis process which requires most of the time in the entire EDA and modeling process. The dataset needs to be cleaned before further analysis is performed, meaning that any form of missing values or unwanted and irrelevant data has to be checked and removed or cleaned in order to avoid errors in the further steps. The analysis obtained for missing and null values in the dataset from the data profiling report is as follows which will help in understanding what and how exactly does the data need to be cleaned.

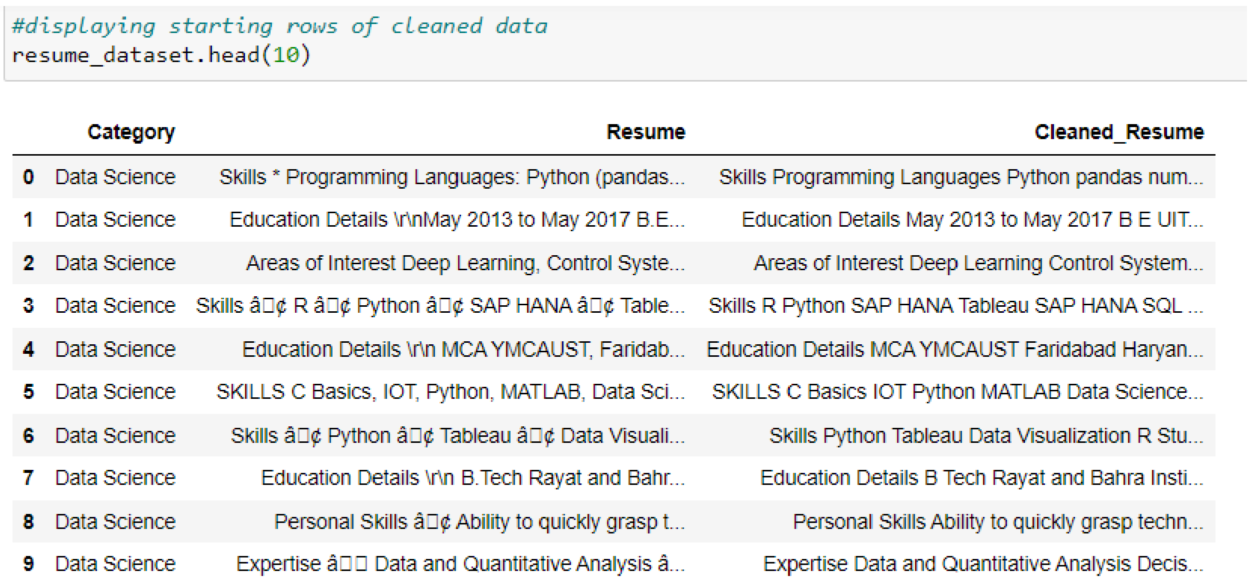


From the above plot of missing values, we get an overview of which parameters have the missing or null values that needs to be either removed or taken care of with respect to a values, as these blank values will cause an error in the further data analysis processes. Here, it is observed that there is no missing values or null values in either of the field values, thus the data is clean and further analysis can be performed.

The resume field column although has text and data which is not clean and does not contain pure form of text which would cause errors while classification and passing the data to the model.

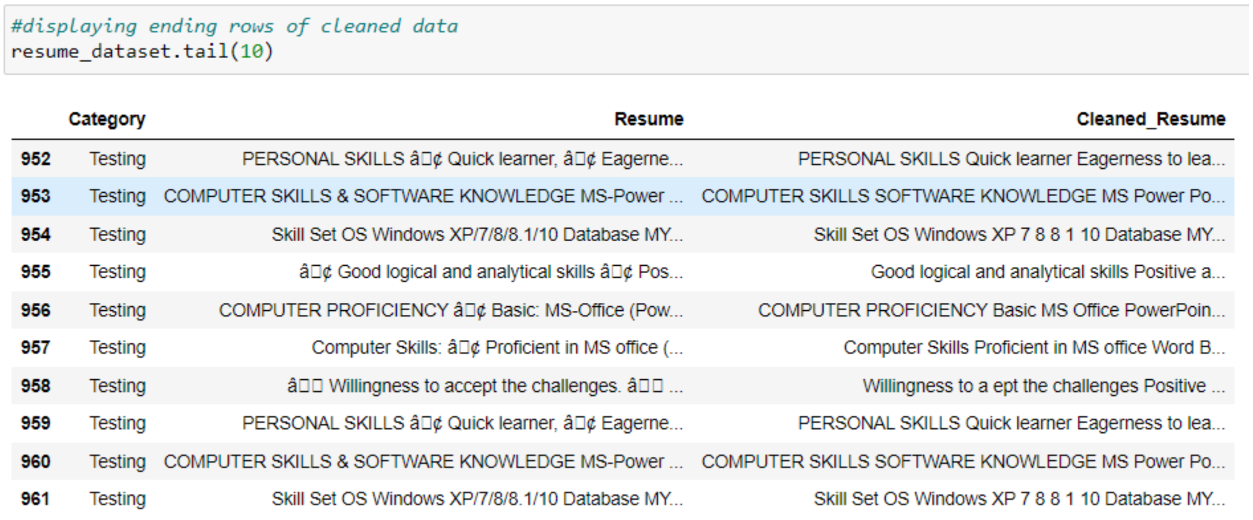
Thus, the data field needs to be taken into account and cleaned for which the regex function in Python is applied to clean the data and the cleaned set of field value is stored back into a new column of the data frame as shown in the output below.

Output: Cleaned data column



```
#displaying starting rows of cleaned data
resume_dataset.head(10)
```

	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...
5	Data Science	SKILLS C Basics, IOT, Python, MATLAB, Data Sci...	SKILLS C Basics IOT Python MATLAB Data Science...
6	Data Science	Skills â□¢ Python â□¢ Tableau â□¢ Data Visuali...	Skills Python Tableau Data Visualization R Stu...
7	Data Science	Education Details \r\n B.Tech Rayat and Bahr...	Education Details B Tech Rayat and Bahra Insti...
8	Data Science	Personal Skills â□¢ Ability to quickly grasp t...	Personal Skills Ability to quickly grasp techn...
9	Data Science	Expertise â□□ Data and Quantitative Analysis â...	Expertise Data and Quantitative Analysis Decis...



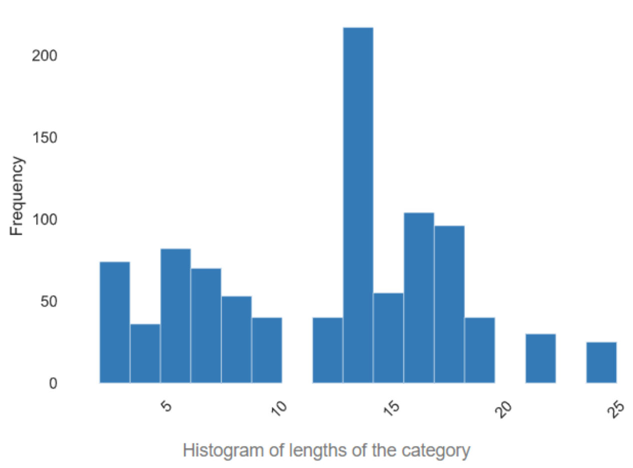
```
#displaying ending rows of cleaned data
resume_dataset.tail(10)
```

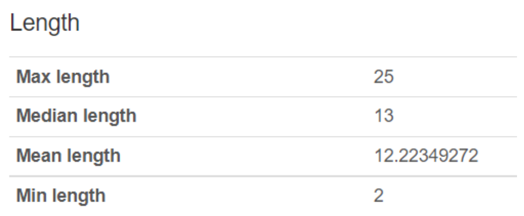
	Category	Resume	Cleaned_Resume
952	Testing	PERSONAL SKILLS â□¢ Quick learner, â□¢ Eagerne...	PERSONAL SKILLS Quick learner Eagerness to lea...
953	Testing	COMPUTER SKILLS & SOFTWARE KNOWLEDGE MS-Power ...	COMPUTER SKILLS SOFTWARE KNOWLEDGE MS Power Po...
954	Testing	Skill Set OS Windows XP/7/8/8.1/10 Database MY...	Skill Set OS Windows XP 7 8 8 1 10 Database MY...
955	Testing	â□¢ Good logical and analytical skills â□¢ Pos...	Good logical and analytical skills Positive a...
956	Testing	COMPUTER PROFICIENCY â□¢ Basic: MS-Office (Pow...	COMPUTER PROFICIENCY Basic MS Office PowerPoin...
957	Testing	Computer Skills: â□¢ Proficient in MS office (...	Computer Skills Proficient in MS office Word B...
958	Testing	â□□ Willingness to accept the challenges. â□□ ...	Willingness to a ept the challenges Positive ...
959	Testing	PERSONAL SKILLS â□¢ Quick learner, â□¢ Eagerne...	PERSONAL SKILLS Quick learner Eagerness to lea...
960	Testing	COMPUTER SKILLS & SOFTWARE KNOWLEDGE MS-Power ...	COMPUTER SKILLS SOFTWARE KNOWLEDGE MS Power Po...
961	Testing	Skill Set OS Windows XP/7/8/8.1/10 Database MY...	Skill Set OS Windows XP 7 8 8 1 10 Database MY...

Data Visualization

Data visualization helps in analyzing and presenting the data in an effective way through graphical representations. The dataset is analyzed using the data visualization method where the various parameters are plotted which gives an overview of the data. Visual representations such as correlation plot, bar graphs, analysis of category count, etc. are some of the graphs that are plotted to get an overview of the dataset.

Visual Analysis 1: Histogram of lengths of the Category



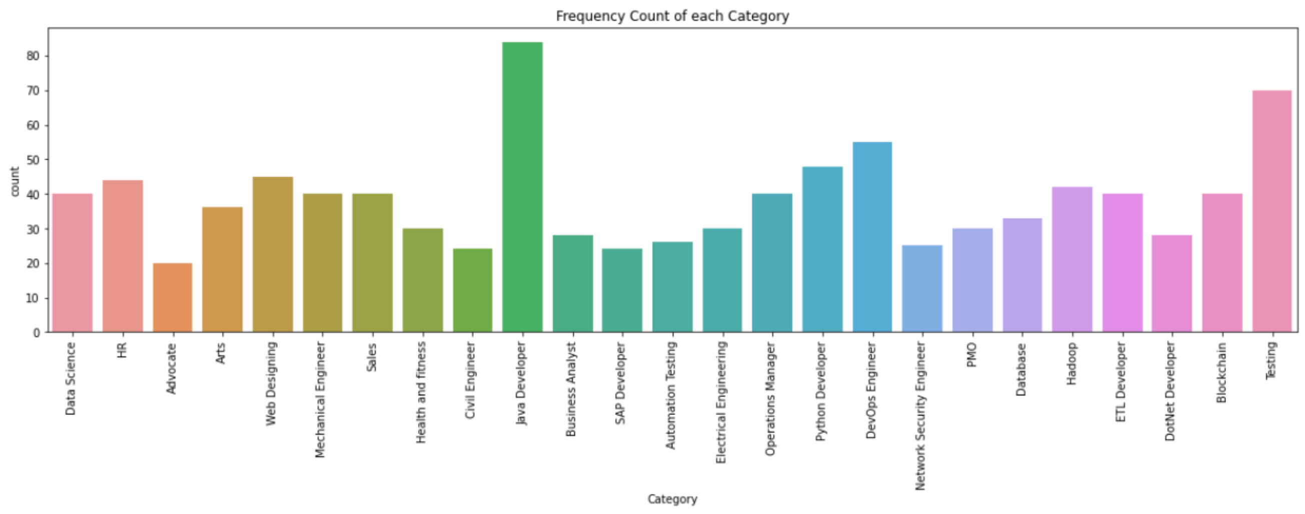


Length

Max length	25
Median length	13
Mean length	12.22349272
Min length	2

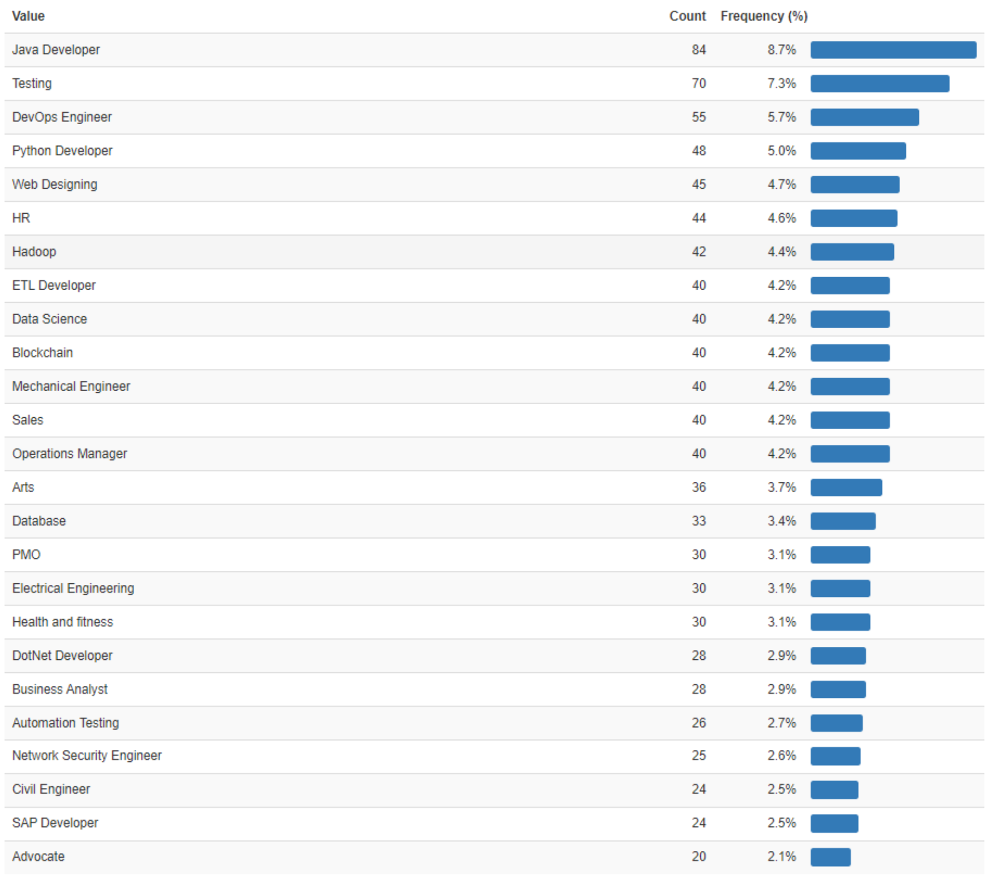
The above histogram represents the frequency length or count of the category in the dataset.

Visual Analysis 2: Frequency Count of each Category



The visual 2 gives an overview of the frequency count of each category present in the resume dataset. This helps in understanding the total count of categories that are present for each resume data that is present in the data.

Visual Analysis 3: Count and Frequency Percent of each Category



Value	Count	Frequency (%)
Java Developer	84	8.7%
Testing	70	7.3%
DevOps Engineer	55	5.7%
Python Developer	48	5.0%
Web Designing	45	4.7%
HR	44	4.6%
Hadoop	42	4.4%
ETL Developer	40	4.2%
Data Science	40	4.2%
Blockchain	40	4.2%
Mechanical Engineer	40	4.2%
Sales	40	4.2%
Operations Manager	40	4.2%
Arts	36	3.7%
Database	33	3.4%
PMO	30	3.1%
Electrical Engineering	30	3.1%
Health and fitness	30	3.1%
DotNet Developer	28	2.9%
Business Analyst	28	2.9%
Automation Testing	26	2.7%
Network Security Engineer	25	2.6%
Civil Engineer	24	2.5%
SAP Developer	24	2.5%
Advocate	20	2.1%

The count and frequency percent for each category is displayed which is generated in the data profiling report implemented on the dataset.

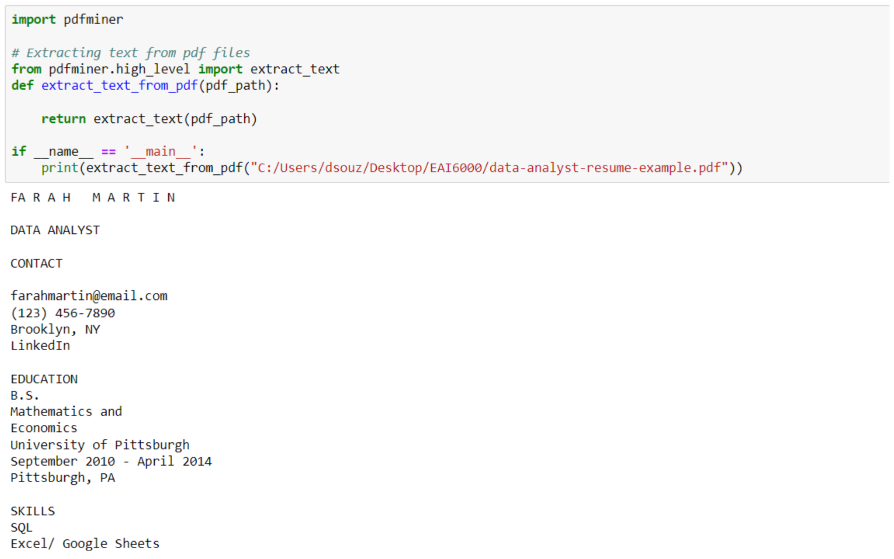
Model Building

The model building is the predictive analysis step in the data analysis process where various machine learning models are taken into consideration for building a model based on the requirements for either classification of the data or prediction of the data. The goal of this project is to classify the categories of the resume data for which the training dataset would be given to the model and thus a system would be developed which would classify the resume text and skills into its respective categories. The categories that are present in the dataset is categorical type of data and thus labeling for the category column needs to be done for which the model can easily classify the text. After performing the label encoding, word vectorization is performed using TFIDF Vectorizer in order to generate bag of words for the respective category which can be used for training the model. In order to gauge the accuracy of the model and understand which model would be best suited for this task, KNeighbors and Logistic Regression Models were built for which the accuracy was compared, and the respective confusion matrix was displayed.

Since the resume received by the companies is in the PDF format and passing the data to the model to build the resume parser would be a tough task to perform. Hence, it is required to extract the information and data from the resume and pass the text to the classification and prediction model. Here, we considered a sample resume which is in the PDF format and extracted the text and data from the document which can be used further for the analysis. The below steps explains the process of data extraction from the document in Python.

1. <u>Extracting text from a PDF file</u>

PDF files are very popular among resumes, and we have used the 'pdfminer' function in Python to extract the text from the document using the below mentioned code.

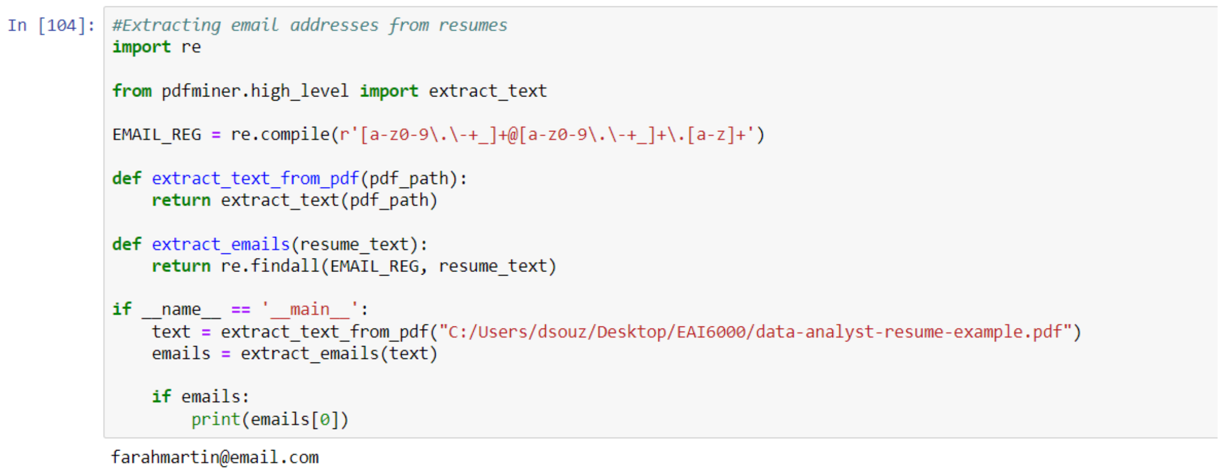


```python
import pdfminer

# Extracting text from pdf files
from pdfminer.high_level import extract_text
def extract_text_from_pdf(pdf_path):

    return extract_text(pdf_path)

if __name__ == '__main__':
    print(extract_text_from_pdf("C:/Users/dsouz/Desktop/EAI6000/data-analyst-resume-example.pdf"))
```

```
FA R A H   M A R T I N

DATA ANALYST

CONTACT

farahmartin@email.com
(123) 456-7890
Brooklyn, NY
LinkedIn

EDUCATION
B.S.
Mathematics and
Economics
University of Pittsburgh
September 2010 - April 2014
Pittsburgh, PA

SKILLS
SQL
Excel/ Google Sheets
```

2. Extracting email address from the resume

To extract emails from text, we can use regular expressions. In the example below we take the help of the regular expression package to define the pattern of an email ID and then write a function to retrieve the text that matches this function. When we run the program, we get the email ID as the output. The email that we receive as output is generally the applicant's actual email address since people tend to place their contact details in the header section of their resumes.

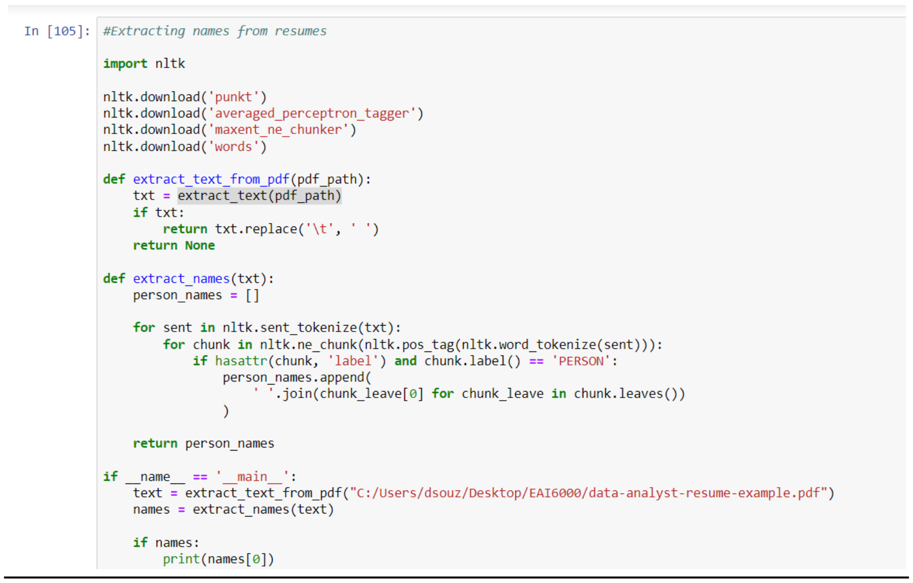


```python
In [104]: #Extracting email addresses from resumes
          import re

          from pdfminer.high_level import extract_text

          EMAIL_REG = re.compile(r'[a-z0-9\.\-+_]+@[a-z0-9\.\-+_]+\.[a-z]+')

          def extract_text_from_pdf(pdf_path):
              return extract_text(pdf_path)

          def extract_emails(resume_text):
              return re.findall(EMAIL_REG, resume_text)

          if __name__ == '__main__':
              text = extract_text_from_pdf("C:/Users/dsouz/Desktop/EAI6000/data-analyst-resume-example.pdf")
              emails = extract_emails(text)

              if emails:
                  print(emails[0])
```

```
          farahmartin@email.com
```

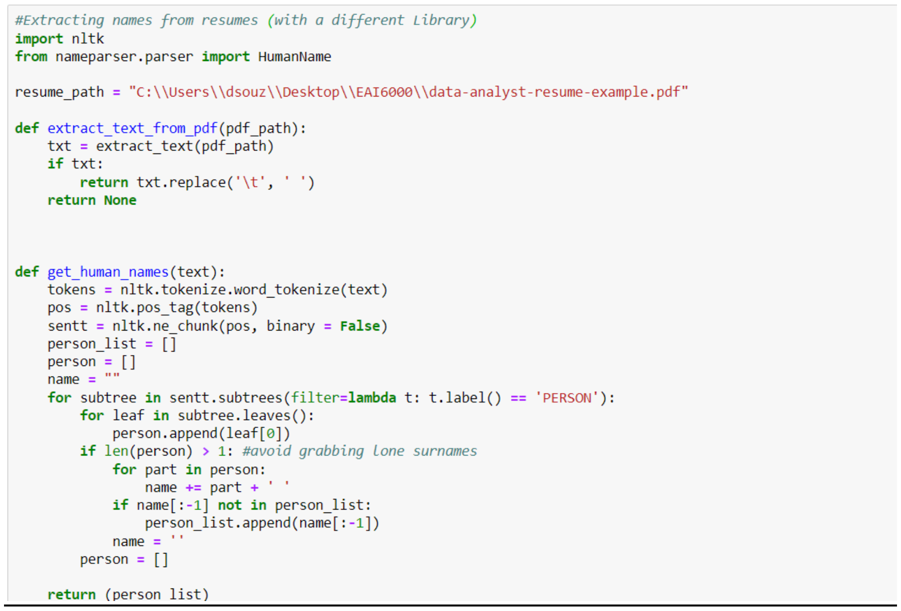
3. Extracting names from the resume

Building the resume parser is not easy. It is noticed that the nltk's person's name detection algorithm does not give correct results. We see the output as 'Pandas' which is the name of a python package instead of the name of an actual person in the resume which ideally should have been 'Farah Martin' (from the template resume provided in the path).

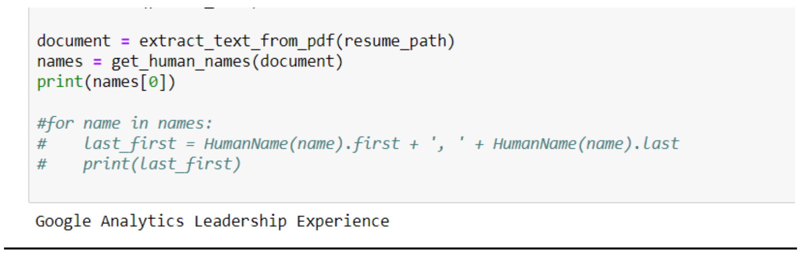


```
In [105]: #Extracting names from resumes

          import nltk

          nltk.download('punkt')
          nltk.download('averaged_perceptron_tagger')
          nltk.download('maxent_ne_chunker')
          nltk.download('words')

          def extract_text_from_pdf(pdf_path):
              txt = extract_text(pdf_path)
              if txt:
                  return txt.replace('\t', ' ')
              return None

          def extract_names(txt):
              person_names = []

              for sent in nltk.sent_tokenize(txt):
                  for chunk in nltk.ne_chunk(nltk.pos_tag(nltk.word_tokenize(sent))):
                      if hasattr(chunk, 'label') and chunk.label() == 'PERSON':
                          person_names.append(
                              ' '.join(chunk_leave[0] for chunk_leave in chunk.leaves())
                          )

              return person_names

          if __name__ == '__main__':
              text = extract_text_from_pdf("C:/Users/dsouz/Desktop/EAI6000/data-analyst-resume-example.pdf")
              names = extract_names(text)

              if names:
                  print(names[0])
```

a. <u>Extracting names from the resume using a different package</u>



```
#Extracting names from resumes (with a different Library)
import nltk
from nameparser.parser import HumanName

resume_path = "C:\\Users\\dsouz\\Desktop\\EAI6000\\data-analyst-resume-example.pdf"

def extract_text_from_pdf(pdf_path):
    txt = extract_text(pdf_path)
    if txt:
        return txt.replace('\t', ' ')
    return None



def get_human_names(text):
    tokens = nltk.tokenize.word_tokenize(text)
    pos = nltk.pos_tag(tokens)
    sentt = nltk.ne_chunk(pos, binary = False)
    person_list = []
    person = []
    name = ""
    for subtree in sentt.subtrees(filter=lambda t: t.label() == 'PERSON'):
        for leaf in subtree.leaves():
            person.append(leaf[0])
        if len(person) > 1: #avoid grabbing lone surnames
            for part in person:
                name += part + ' '
            if name[:-1] not in person_list:
                person_list.append(name[:-1])
            name = ''
        person = []

    return (person_list)
```

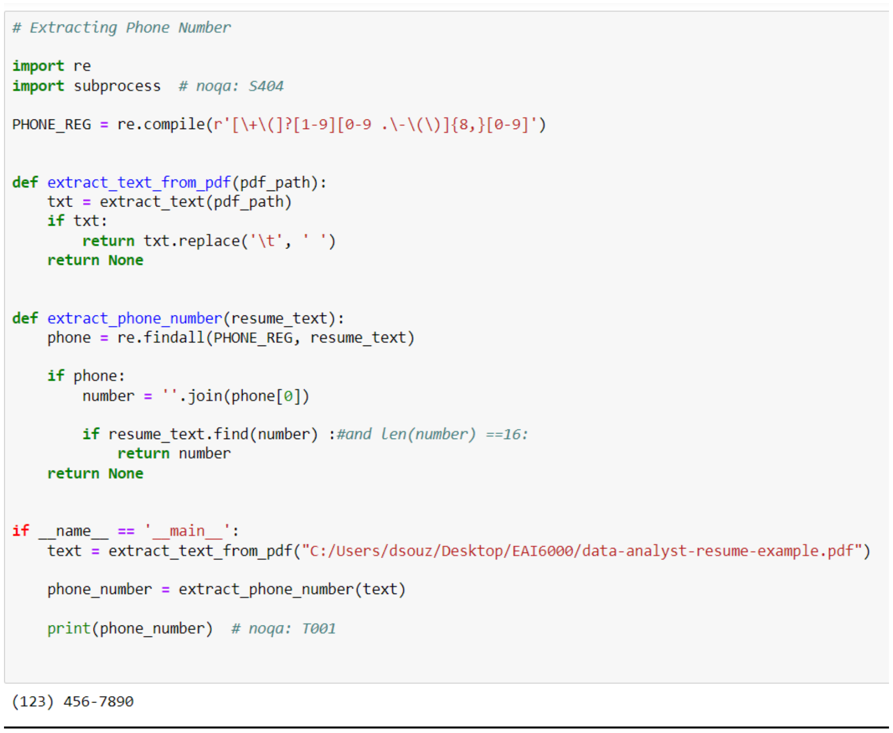


```
document = extract_text_from_pdf(resume_path)
names = get_human_names(document)
print(names[0])

#for name in names:
#    last_first = HumanName(name).first + ', ' + HumanName(name).last
#    print(last_first)
```

```
Google Analytics Leadership Experience
```

4. Extracting phone number from the resume

Unlike extracting human names from the resume, extracting the phone number is much easier to deal with. The following code return's the phone number of the resume owner.

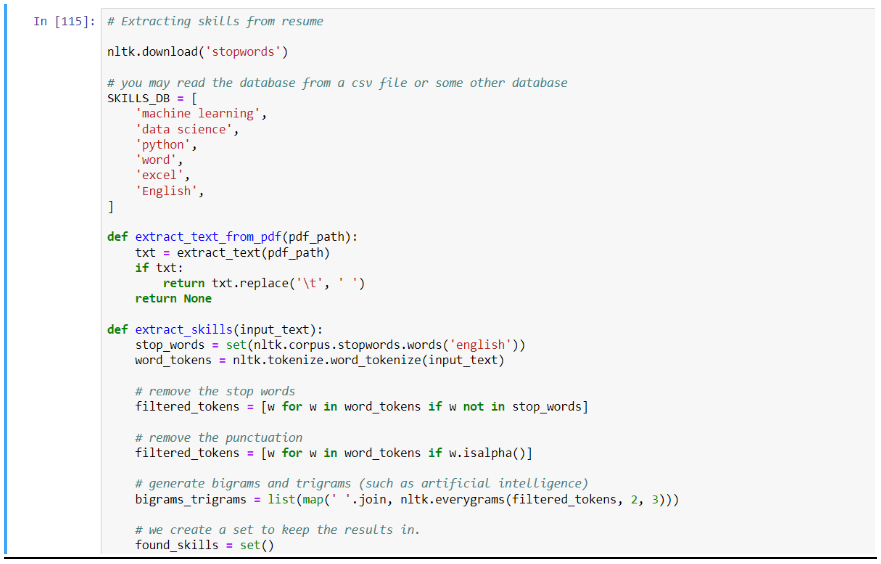


```
# Extracting Phone Number

import re
import subprocess  # noqa: S404

PHONE_REG = re.compile(r'[\+\(]?[1-9][0-9 .\-\(\)]{8,}[0-9]')


def extract_text_from_pdf(pdf_path):
    txt = extract_text(pdf_path)
    if txt:
        return txt.replace('\t', ' ')
    return None


def extract_phone_number(resume_text):
    phone = re.findall(PHONE_REG, resume_text)

    if phone:
        number = ''.join(phone[0])

        if resume_text.find(number) :#and len(number) ==16:
            return number
    return None


if __name__ == '__main__':
    text = extract_text_from_pdf("C:/Users/dsouz/Desktop/EAI6000/data-analyst-resume-example.pdf")

    phone_number = extract_phone_number(text)

    print(phone_number)  # noqa: T001
```

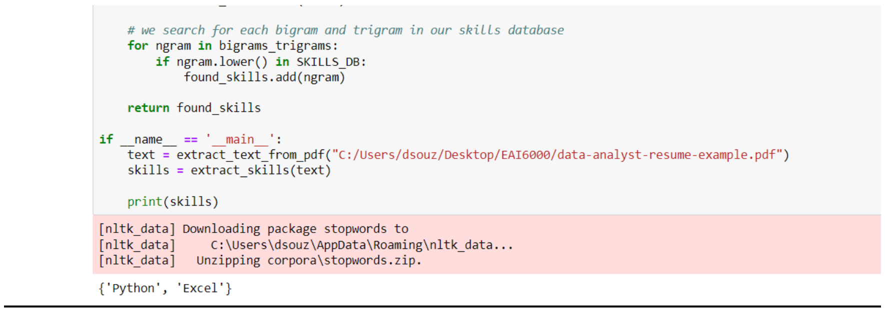
```
(123) 456-7890
```

5. Extracting skills from the resume

Extracting skills from a text is sometimes incredibly challenging and at times to increase accuracy, the API in use needs to be able to verify if a text is a skill or not. The following code uses the nltk library to filter out the stop words and generates tokens.



```
In [115]: # Extracting skills from resume

          nltk.download('stopwords')

          # you may read the database from a csv file or some other database
          SKILLS_DB = [
              'machine learning',
              'data science',
              'python',
              'word',
              'excel',
              'English',
          ]

          def extract_text_from_pdf(pdf_path):
              txt = extract_text(pdf_path)
              if txt:
                  return txt.replace('\t', ' ')
              return None

          def extract_skills(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()
```



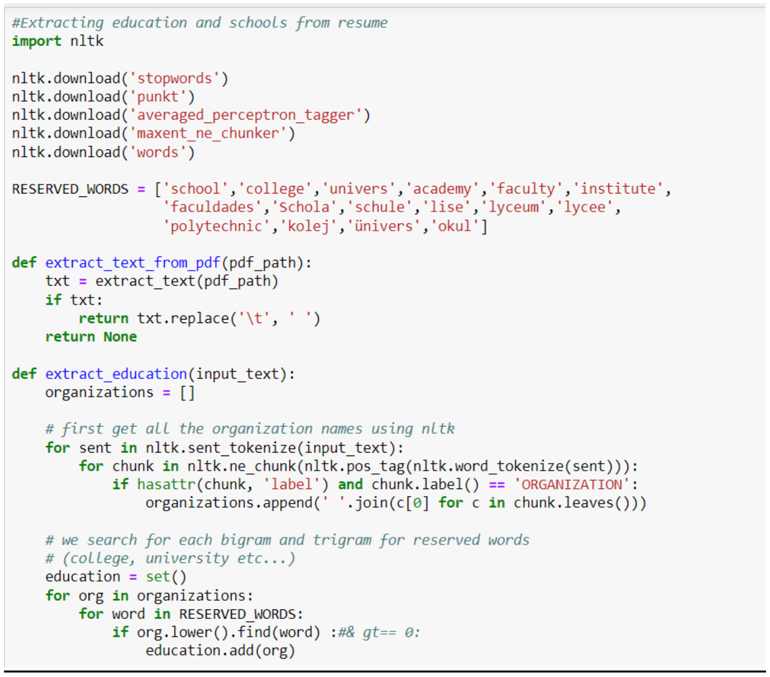
```
    # we search for each bigram and trigram in our skills database
    for ngram in bigrams_trigrams:
        if ngram.lower() in SKILLS_DB:
            found_skills.add(ngram)

    return found_skills

if __name__ == '__main__':
    text = extract_text_from_pdf("C:/Users/dsouz/Desktop/EAI6000/data-analyst-resume-example.pdf")
    skills = extract_skills(text)

    print(skills)
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\dsouz\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
{'Python', 'Excel'}
```

We see that ‘Python’ and ‘Excel’ have been returned as skills from the resume.

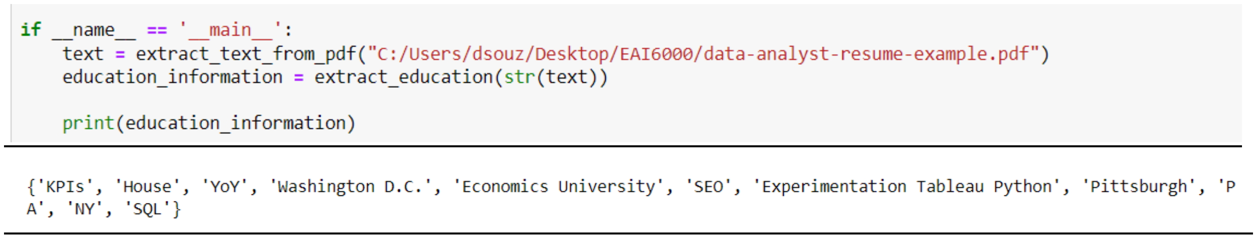
6. Extracting education and schools from the resume

Saying so, in the following code, we look for words such as “university, college, etc., in named entities labeled as organization types. It performed well in this case. The reserved_words list can be enriched in the code as per the organization’s requirement.

We see that the keywords for the name of the school and the University have been successfully extracted. However, this also includes keywords that may not be relevant to a school or education degree.



```
#Extracting education and schools from resume
import nltk

nltk.download('stopwords')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')

RESERVED_WORDS = ['school','college','univers','academy','faculty','institute',
                  'faculdades','Schola','schule','lise','lyceum','lycee',
                  'polytechnic','kolej','ünivers','okul']

def extract_text_from_pdf(pdf_path):
    txt = extract_text(pdf_path)
    if txt:
        return txt.replace('\t', ' ')
    return None

def extract_education(input_text):
    organizations = []

    # first get all the organization names using nltk
    for sent in nltk.sent_tokenize(input_text):
        for chunk in nltk.ne_chunk(nltk.pos_tag(nltk.word_tokenize(sent))):
            if hasattr(chunk, 'label') and chunk.label() == 'ORGANIZATION':
                organizations.append(' '.join(c[0] for c in chunk.leaves()))

    # we search for each bigram and trigram for reserved words
    # (college, university etc...)
    education = set()
    for org in organizations:
        for word in RESERVED_WORDS:
            if org.lower().find(word) :#& gt== 0:
                education.add(org)
```



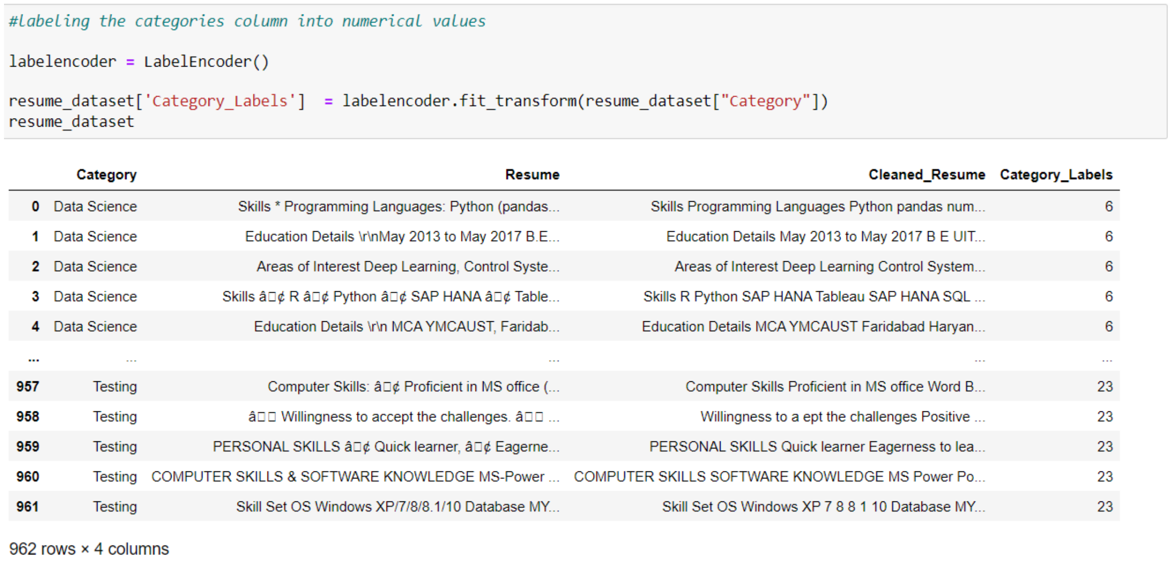
```
if __name__ == '__main__':
    text = extract_text_from_pdf("C:/Users/dsouz/Desktop/EAI6000/data-analyst-resume-example.pdf")
    education_information = extract_education(str(text))

    print(education_information)
```

```
{'KPIs', 'House', 'YoY', 'Washington D.C.', 'Economics University', 'SEO', 'Experimentation Tableau Python', 'Pittsburgh', 'P
A', 'NY', 'SQL'}
```

Thus, the data extracted from the resume document can be stored in a data frame or a excel file which can be used further while performing the model building phase. This data stored can now be easily used in text classification and categorizing them into their respective categories or fields which would help in determining the exact domain of each category and thus a particular resume can be further viewed. The functions of pdfminer helps in extracting the text and data from the document and it is helpful in such applications.

Below are the steps performed for the classification model building and the outputs obtained.

1. <u>Label Encoding</u>



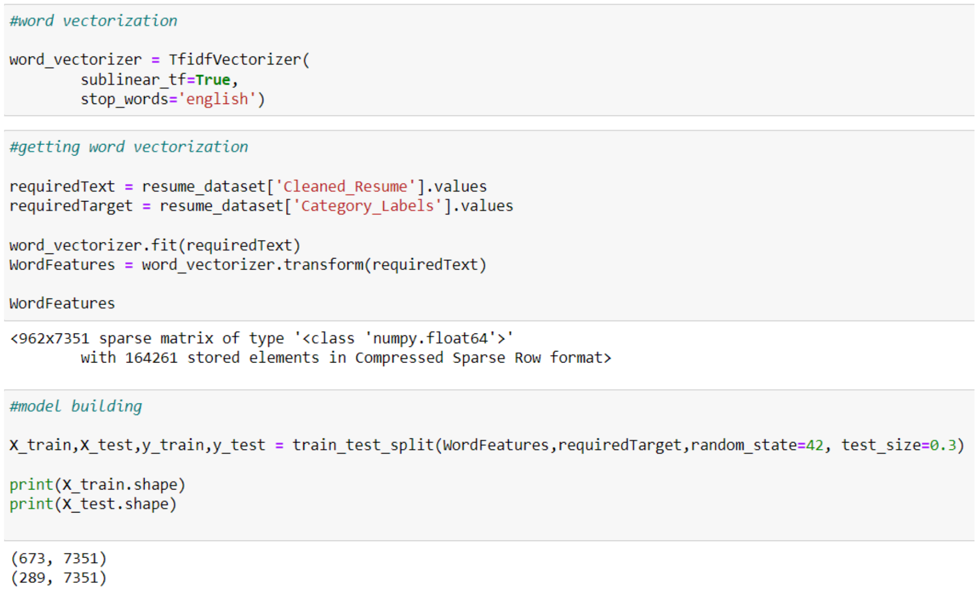
```
#labeling the categories column into numerical values

labelencoder = LabelEncoder()

resume_dataset['Category_Labels']  = labelencoder.fit_transform(resume_dataset["Category"])
resume_dataset
```

	Category	Resume	Cleaned_Resume	Category_Labels
0	Data Science	Skills * Programming Languages: Python (pandas...	Skills Programming Languages Python pandas num...	6
1	Data Science	Education Details \r\nMay 2013 to May 2017 B.E...	Education Details May 2013 to May 2017 B E UIT...	6
2	Data Science	Areas of Interest Deep Learning, Control Syste...	Areas of Interest Deep Learning Control System...	6
3	Data Science	Skills â□¢ R â□¢ Python â□¢ SAP HANA â□¢ Table...	Skills R Python SAP HANA Tableau SAP HANA SQL ...	6
4	Data Science	Education Details \r\n MCA YMCAUST, Faridab...	Education Details MCA YMCAUST Faridabad Haryan...	6
...	...	...	...	...
957	Testing	Computer Skills: â□¢ Proficient in MS office (...	Computer Skills Proficient in MS office Word B...	23
958	Testing	â□□ Willingness to accept the challenges. â□□ ...	Willingness to a ept the challenges Positive ...	23
959	Testing	PERSONAL SKILLS â□¢ Quick learner, â□¢ Eagerne...	PERSONAL SKILLS Quick learner Eagerness to lea...	23
960	Testing	COMPUTER SKILLS & SOFTWARE KNOWLEDGE MS-Power ...	COMPUTER SKILLS SOFTWARE KNOWLEDGE MS Power Po...	23
961	Testing	Skill Set OS Windows XP/7/8/8.1/10 Database MY...	Skill Set OS Windows XP 7 8 8 1 10 Database MY...	23

962 rows × 4 columns

The category is labelled into its respective labeling in numerical format which would be easily classified by the model.

2. <u>Word Vectorization</u>



```
#word vectorization

word_vectorizer = TfidfVectorizer(
        sublinear_tf=True,
        stop_words='english')
```

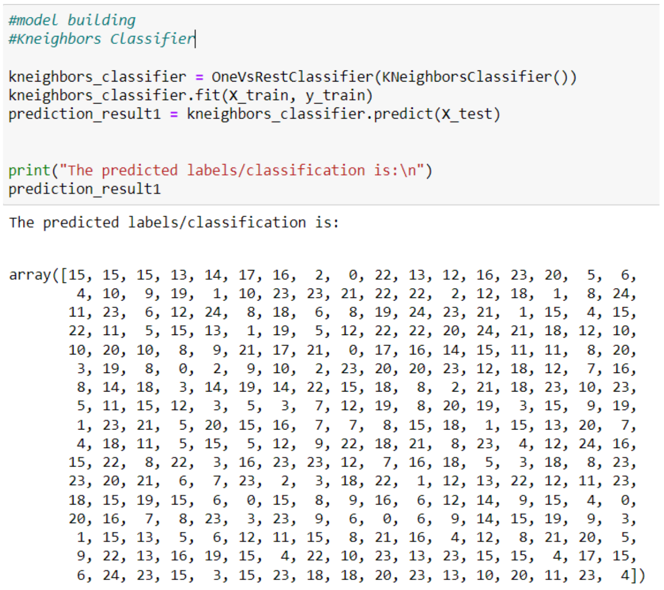
```
#getting word vectorization

requiredText = resume_dataset['Cleaned_Resume'].values
requiredTarget = resume_dataset['Category_Labels'].values

word_vectorizer.fit(requiredText)
WordFeatures = word_vectorizer.transform(requiredText)

WordFeatures
```

```
<962x7351 sparse matrix of type '<class 'numpy.float64'>'
        with 164261 stored elements in Compressed Sparse Row format>
```

```
#model building

X_train,X_test,y_train,y_test = train_test_split(WordFeatures,requiredTarget,random_state=42, test_size=0.3)

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

```
(673, 7351)
(289, 7351)
```

3. Model Building **KNeighbors Classifier**

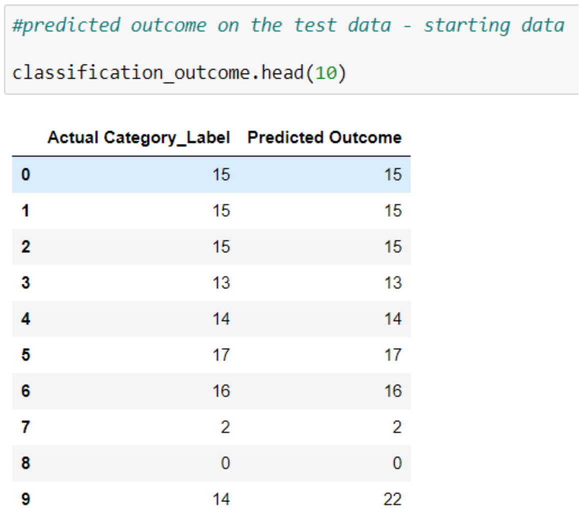


```
#model building
#Kneighbors Classifier

kneighbors_classifier = OneVsRestClassifier(KNeighborsClassifier())
kneighbors_classifier.fit(X_train, y_train)
prediction_result1 = kneighbors_classifier.predict(X_test)


print("The predicted labels/classification is:\n")
prediction_result1
```

```
The predicted labels/classification is:


array([15, 15, 15, 13, 14, 17, 16,  2,  0, 22, 13, 12, 16, 23, 20,  5,  6,
        4, 10,  9, 19,  1, 10, 23, 23, 21, 22, 22,  2, 12, 18,  1,  8, 24,
       11, 23,  6, 12, 24,  8, 18,  6,  8, 19, 24, 23, 21,  1, 15,  4, 15,
       22, 11,  5, 15, 13,  1, 19,  5, 12, 22, 22, 20, 24, 21, 18, 12, 10,
       10, 20, 10,  8,  9, 21, 17, 21,  0, 17, 16, 14, 15, 11, 11,  8, 20,
        3, 19,  8,  0,  2,  9, 10,  2, 23, 20, 20, 23, 12, 18, 12,  7, 16,
        8, 14, 18,  3, 14, 19, 14, 22, 15, 18,  8,  2, 21, 18, 23, 10, 23,
        5, 11, 15, 12,  3,  5,  3,  7, 12, 19,  8, 20, 19,  3, 15,  9, 19,
        1, 23, 21,  5, 20, 15, 16,  7,  7,  8, 15, 18,  1, 15, 13, 20,  7,
        4, 18, 11,  5, 15,  5, 12,  9, 22, 18, 21,  8, 23,  4, 12, 24, 16,
       15, 22,  8, 22,  3, 16, 23, 23, 12,  7, 16, 18,  5,  3, 18,  8, 23,
       23, 20, 21,  6,  7, 23,  2,  3, 18, 22,  1, 12, 13, 22, 12, 11, 23,
       18, 15, 19, 15,  6,  0, 15,  8,  9, 16,  6, 12, 14,  9, 15,  4,  0,
       20, 16,  7,  8, 23,  3, 23,  9,  6,  0,  6,  9, 14, 15, 19,  9,  3,
        1, 15, 13,  5,  6, 12, 11, 15,  8, 21, 16,  4, 12,  8, 21, 20,  5,
        9, 22, 13, 16, 19, 15,  4, 22, 10, 23, 13, 23, 15, 15,  4, 17, 15,
        6, 24, 23, 15,  3, 15, 23, 18, 18, 20, 23, 13, 10, 20, 11, 23,  4])
```

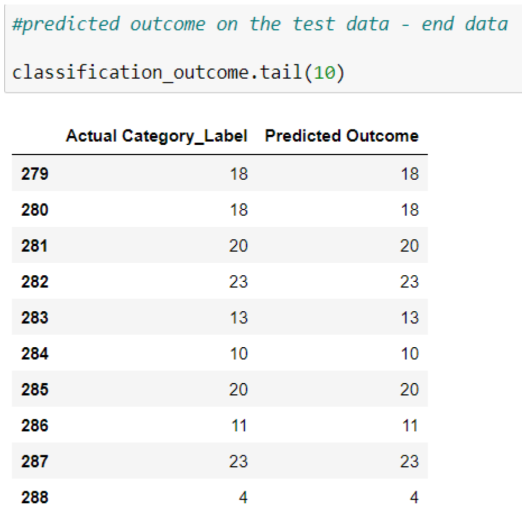
The KNeighbors Classifier algorithm is implemented in order to classify the categories of the resume dataset for which the data is split into training and testing dataset where 70% of the data is used for training the model and 30% data is used as the testing data. The predicted outcome array is displayed for the test dataset as shown in the above figure.

4. Predicted Outcome



```
#predicted outcome on the test data - starting data

classification_outcome.head(10)
```

	Actual Category_Label	Predicted Outcome
0	15	15
1	15	15
2	15	15
3	13	13
4	14	14
5	17	17
6	16	16
7	2	2
8	0	0
9	14	22



```
#predicted outcome on the test data - end data

classification_outcome.tail(10)
```

	Actual Category_Label	Predicted Outcome
279	18	18
280	18	18
281	20	20
282	23	23
283	13	13
284	10	10
285	20	20
286	11	11
287	23	23
288	4	4

5. Accuracy of the model

```
#accuracy of the model
#Kneighbors Classifier

print('Accuracy of KNeighbors Classifier on training set: {:.2f}'.format(kneighbors_classifier.score(X_train, y_train)))
print('Accuracy of KNeighbors Classifier on test set:     {:.2f}'.format(kneighbors_classifier.score(X_test, y_test)))

test_accuracy_KN = kneighbors_classifier.score(X_test, y_test)
test_accuracy_KN
```

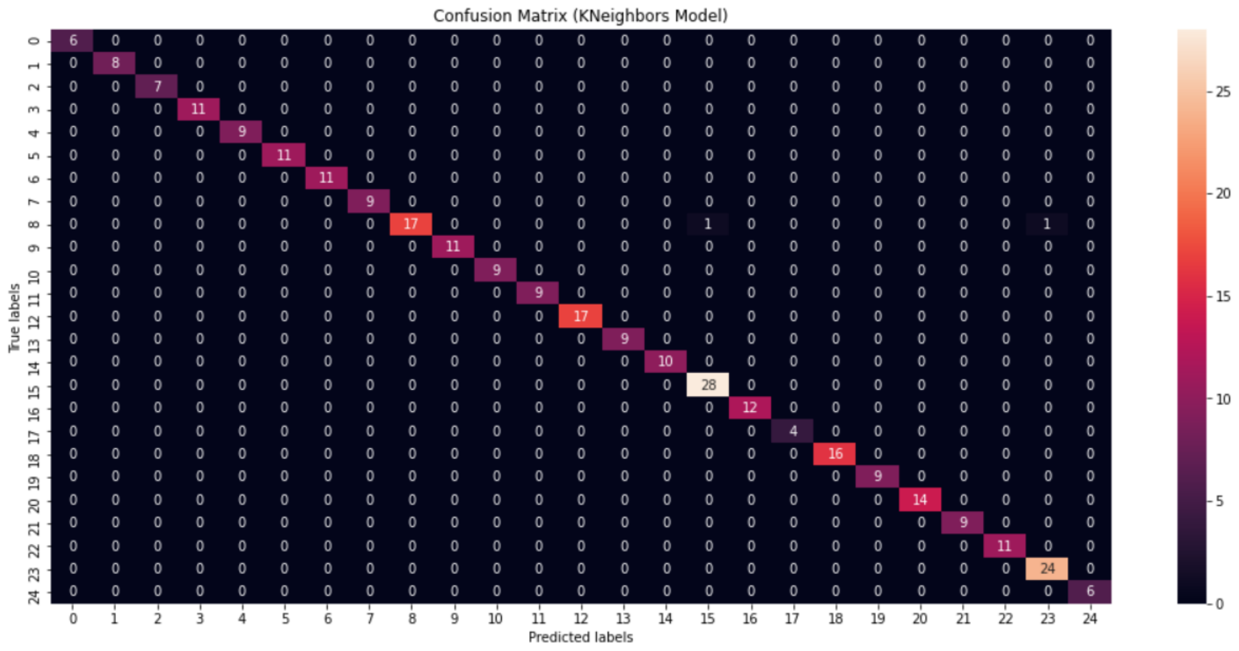
```
Accuracy of KNeighbors Classifier on training set: 0.98
Accuracy of KNeighbors Classifier on test set:     0.97

0.9688581314878892
```

Here, the accuracy of the model obtained for the test dataset is 97%.

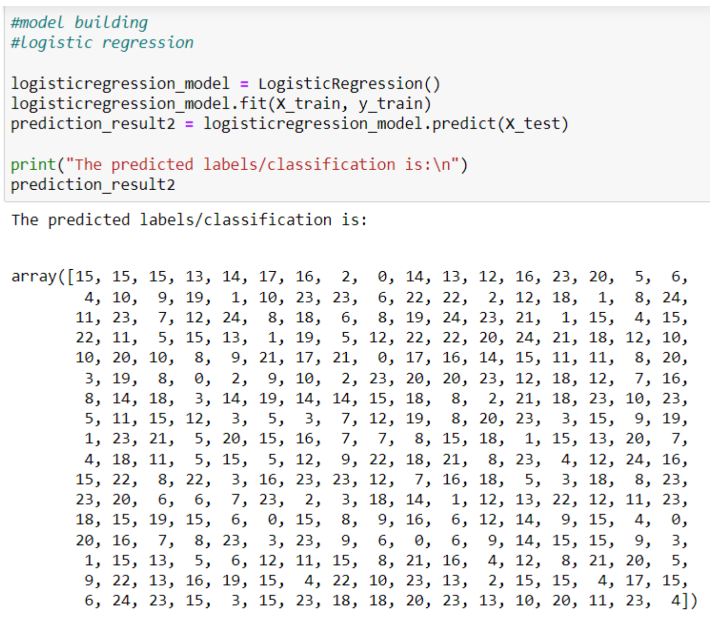
6. Classification Report

```
 Classification report for classifier OneVsRestClassifier(estimator=KNeighborsClassifier()):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         6
           1       1.00      1.00      1.00         8
           2       1.00      0.86      0.92         7
           3       1.00      1.00      1.00        11
           4       1.00      1.00      1.00         9
           5       1.00      1.00      1.00        11
           6       0.90      0.82      0.86        11
           7       1.00      0.89      0.94         9
           8       1.00      0.89      0.94        19
           9       1.00      1.00      1.00        11
          10       1.00      1.00      1.00         9
          11       1.00      1.00      1.00         9
          12       1.00      1.00      1.00        17
          13       1.00      1.00      1.00         9
          14       1.00      0.70      0.82        10
          15       1.00      1.00      1.00        28
          16       1.00      1.00      1.00        12
          17       1.00      1.00      1.00         4
          18       1.00      1.00      1.00        16
          19       0.82      1.00      0.90         9
          20       1.00      1.00      1.00        14
          21       0.82      1.00      0.90         9
          22       0.79      1.00      0.88        11
          23       0.96      1.00      0.98        24
          24       1.00      1.00      1.00         6

    accuracy                           0.97       289
   macro avg       0.97      0.97      0.97       289
weighted avg       0.97      0.97      0.97       289
```

7. Confusion Matrix



8. Model Building **Logistic Regression**

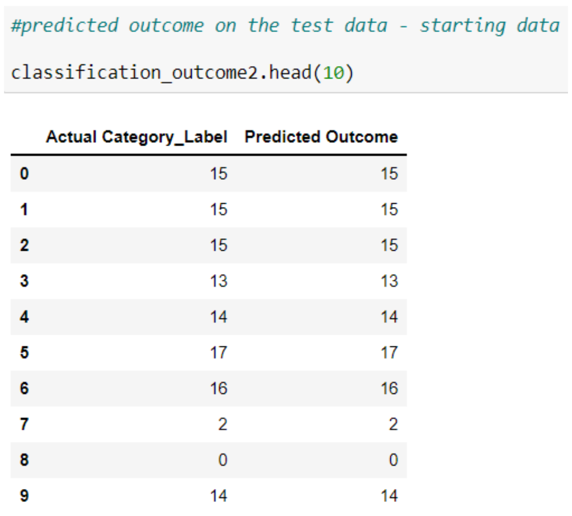


```
#model building
#logistic regression

logisticregression_model = LogisticRegression()
logisticregression_model.fit(X_train, y_train)
prediction_result2 = logisticregression_model.predict(X_test)

print("The predicted labels/classification is:\n")
prediction_result2
```

```
The predicted labels/classification is:


array([15, 15, 15, 13, 14, 17, 16,  2,  0, 14, 13, 12, 16, 23, 20,  5,  6,
        4, 10,  9, 19,  1, 10, 23, 23,  6, 22, 22,  2, 12, 18,  1,  8, 24,
       11, 23,  7, 12, 24,  8, 18,  6,  8, 19, 24, 23, 21,  1, 15,  4, 15,
       22, 11,  5, 15, 13,  1, 19,  5, 12, 22, 22, 20, 24, 21, 18, 12, 10,
       10, 20, 10,  8,  9, 21, 17, 21,  0, 17, 16, 14, 15, 11, 11,  8, 20,
        3, 19,  8,  0,  2,  9, 10,  2, 23, 20, 20, 23, 12, 18, 12,  7, 16,
        8, 14, 18,  3, 14, 19, 14, 14, 15, 18,  8,  2, 21, 18, 23, 10, 23,
        5, 11, 15, 12,  3,  5,  3,  7, 12, 19,  8, 20, 23,  3, 15,  9, 19,
        1, 23, 21,  5, 20, 15, 16,  7,  7,  8, 15, 18,  1, 15, 13, 20,  7,
        4, 18, 11,  5, 15,  5, 12,  9, 22, 18, 21,  8, 23,  4, 12, 24, 16,
       15, 22,  8, 22,  3, 16, 23, 23, 12,  7, 16, 18,  5,  3, 18,  8, 23,
       23, 20,  6,  6,  7, 23,  2,  3, 18, 14,  1, 12, 13, 22, 12, 11, 23,
       18, 15, 19, 15,  6,  0, 15,  8,  9, 16,  6, 12, 14,  9, 15,  4,  0,
       20, 16,  7,  8, 23,  3, 23,  9,  6,  0,  6,  9, 14, 15, 15,  9,  3,
        1, 15, 13,  5,  6, 12, 11, 15,  8, 21, 16,  4, 12,  8, 21, 20,  5,
        9, 22, 13, 16, 19, 15,  4, 22, 10, 23, 13,  2, 15, 15,  4, 17, 15,
        6, 24, 23, 15,  3, 15, 23, 18, 18, 20, 23, 13, 10, 20, 11, 23,  4])
```

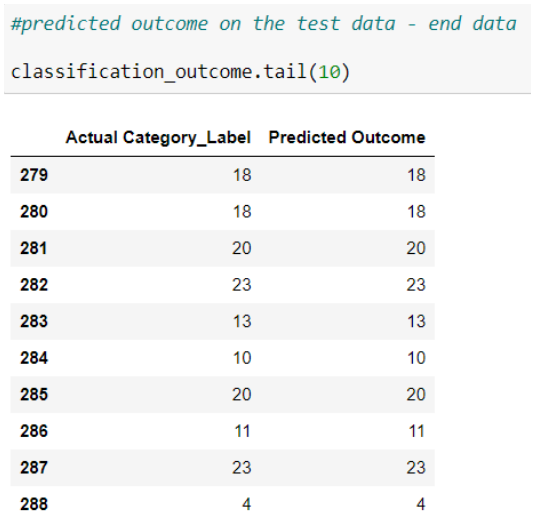
The Logistic Regression algorithm is implemented in order to classify the categories of the resume dataset for which the data is split into training and testing dataset where 70% of the data is used for training the model and 30% data is used as the testing data. The predicted outcome array is displayed for the test dataset as shown in the above figure.

9. Predicted Outcome



```
#predicted outcome on the test data - starting data

classification_outcome2.head(10)
```

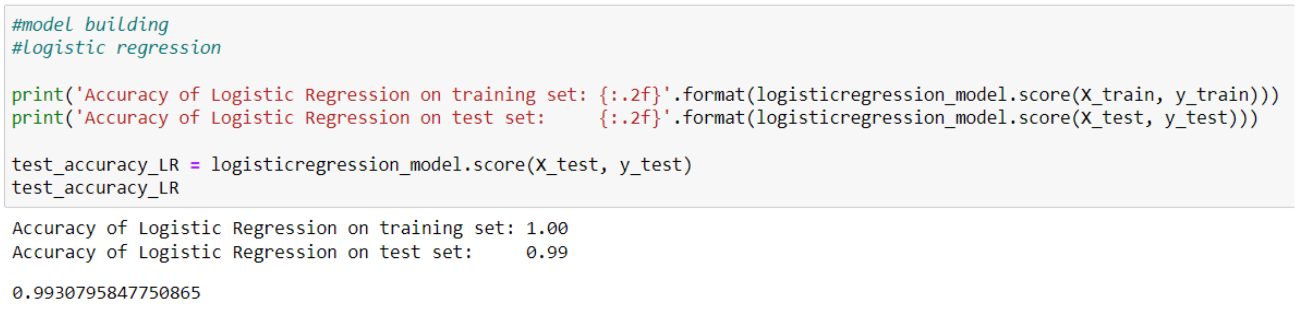
	Actual Category_Label	Predicted Outcome
0	15	15
1	15	15
2	15	15
3	13	13
4	14	14
5	17	17
6	16	16
7	2	2
8	0	0
9	14	14



```
#predicted outcome on the test data - end data

classification_outcome.tail(10)
```

	Actual Category_Label	Predicted Outcome
279	18	18
280	18	18
281	20	20
282	23	23
283	13	13
284	10	10
285	20	20
286	11	11
287	23	23
288	4	4

10. Accuracy of the model



```
#model building
#logistic regression

print('Accuracy of Logistic Regression on training set: {:.2f}'.format(logisticregression_model.score(X_train, y_train)))
print('Accuracy of Logistic Regression on test set:     {:.2f}'.format(logisticregression_model.score(X_test, y_test)))

test_accuracy_LR = logisticregression_model.score(X_test, y_test)
test_accuracy_LR
```

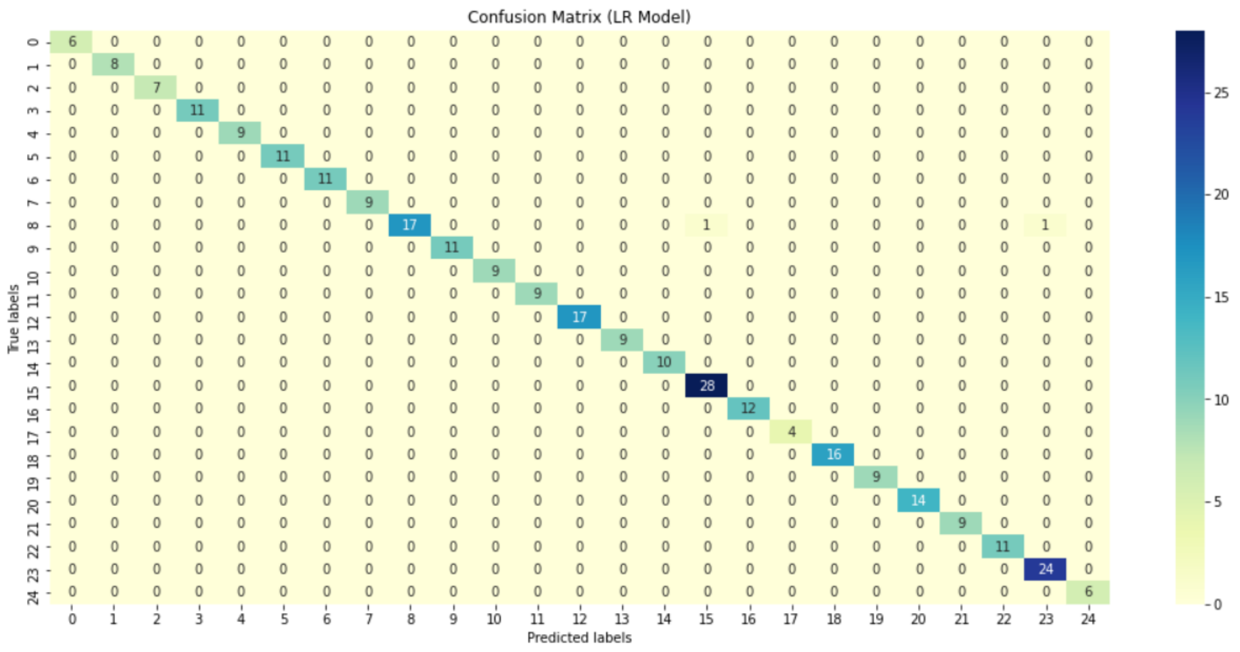
```
Accuracy of Logistic Regression on training set: 1.00
Accuracy of Logistic Regression on test set:     0.99

0.9930795847750865
```

Here, the accuracy of the Logistic Regression model obtained for the test dataset is 99%.

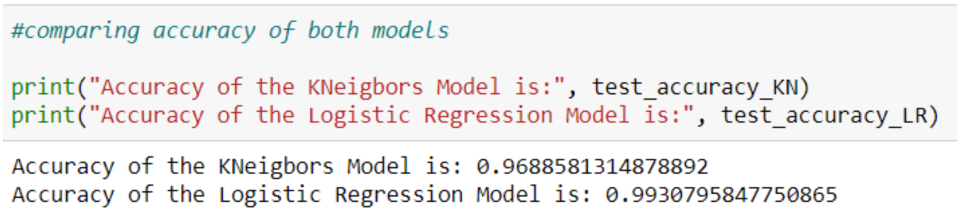
11. Classification Report

```
Classification report LogisticRegression():
              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         7
           3       1.00      1.00      1.00        11
           4       1.00      1.00      1.00         9
           5       1.00      1.00      1.00        11
           6       1.00      1.00      1.00        11
           7       1.00      1.00      1.00         9
           8       1.00      0.89      0.94        19
           9       1.00      1.00      1.00        11
          10       1.00      1.00      1.00         9
          11       1.00      1.00      1.00         9
          12       1.00      1.00      1.00        17
          13       1.00      1.00      1.00         9
          14       1.00      1.00      1.00        10
          15       0.97      1.00      0.98        28
          16       1.00      1.00      1.00        12
          17       1.00      1.00      1.00         4
          18       1.00      1.00      1.00        16
          19       1.00      1.00      1.00         9
          20       1.00      1.00      1.00        14
          21       1.00      1.00      1.00         9
          22       1.00      1.00      1.00        11
          23       0.96      1.00      0.98        24
          24       1.00      1.00      1.00         6

    accuracy                           0.99       289
   macro avg       1.00      1.00      1.00       289
weighted avg       0.99      0.99      0.99       289
```

12. Confusion Matrix



Thus, the two models are implemented for which the accuracy, classification report, and confusion matrix is generated. The two models built are able to classify and categorize the categories of the resume data with an accuracy of 97% and 99% for KNeighbors Classifier and Logistic Regression respectively. Hence, it can be observed that Logistic Regression best performed for this task and the model will be able to accurately classify the data into its categories. The confusion matrix for both the models clearly explains for which categories it correctly classified and which were wrongly classified by the model. The accuracy of the two models obtained are as follows.



```
#comparing accuracy of both models

print("Accuracy of the KNeigbors Model is:", test_accuracy_KN)
print("Accuracy of the Logistic Regression Model is:", test_accuracy_LR)
```

```
Accuracy of the KNeigbors Model is: 0.9688581314878892
Accuracy of the Logistic Regression Model is: 0.9930795847750865
```

Conclusion

The resume parser is developed in order to make it easy for the companies to scan the resume of each candidate saving them time and energy. The unstructured form of data is converted into a structured format which is used by the model to classify the categories and predict the most favorable and least favorable resume of the candidate. This parser built analyzes resume data and extracts the text and information for further classification and prediction. The system built is effective for all the businesses as this time-consuming process of manually going over the resumes of all candidates is eliminated, and the automated system built can perform the task for the same.

The model built for classification and prediction of the resume uses the dataset containing the categories and the resume information extracted from the document. This classifies the skills into their respective categories which makes it easy to differentiate between the skills and each of the categories. Apart from this, the parser built takes document as input passed through the model where the various methods of natural language processing are applied and the text and information from the resume document is extracted in order to send the information to the model for training. In order to achieve this goal of classification of categories and prediction of the resume, exploratory data analysis was performed to gain insights about the data, following which the machine learning models were implemented and the accuracy of the model was gauge to determine the best fit model for this application.

The tasks performed in this project are to extract information and text from the document and second to classify the categories of the resume data. A sample document was taken into consideration in order to extract the data from the resume document for which various extraction functions were performed and implemented and thus for any document these functions can be performed and used. The text and data extracted from the document is the skills of the candidate, name, contact details, school details, etc., which can be send to the model for training and the fields of a particular resume can be categorized.

The two machine learning models implemented were KNeighbors Classifier and Logistic Regression model for classification of the categories on the training set of data collected from Kaggle. The accuracy obtained for the KNeighbors model was 97% whereas the accuracy for the Logistic Regression model was 99%. This applies that the Logistic Regression model performed better as compared to the KNeighbors Classifier and thus the various machine learning models perform differently in different applications. The confusion matrix plotted gives the information about the category labels that are correctly classified by the model and the category labels which are wrongly classified by the model. Thus, it can be observed that, with respect to the Logistic Regression, the model will correctly classify the categories 99% of the time.

References

Resume Dataset. (2021, February 24). Kaggle. https://www.kaggle.com/datasets/gauravduttakiit/resume-dataset

Coder, U. (2022, May 16). How To Extract Text Using PDFMiner In Python. Unbiased Coder. https://unbiased-coder.com/extract-text-pdfminer-python/

Kumar, A. (2020, October 5). Python - Extract Text from PDF file using PDFMiner. Data Analytics. https://vitalflux.com/python-extract-text-pdf-file-using-pdfminer/

T. (n.d.). Scikit Learn - KNeighborsClassifier. Tutorialspoint. https://www.tutorialspoint.com/scikit_learn/scikit_learn_kneighbors_classifier.htm

Li, S. (2019, February 27). Building A Logistic Regression in Python, Step by Step. Medium. https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8

Goyal, C. (2021, June 22). Text Vectorization and Word Embedding | Guide to Master NLP (Part 5). Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/part-5-step-by-step-guide-to-master-nlp-text-vectorization-approaches/

Ganesan, K. (2020, August 6). How to Use Tfidftransformer & Tfidfvectorizer? Kavita Ganesan, PhD. https://kavita-ganesan.com/tfidftransformer-tfidfvectorizer-usage-differences/#.YoME4YfMJPY

Aruchamy, V. (2021, December 15). How To Plot Confusion Matrix In Python And Why You Need To? Stack Vidhya. https://www.stackvidhya.com/plot-confusion-matrix-in-python-and-why/