COMSATS UNIVERSITY ISLAMABAD

Intro To Artificial Intelligence

SEMESTER PROJECT
Resume Screening with Python

Name: Zainab Rizwan
Reg No: SP22-BAI-054
Submitted To: Dr. Rasool Bukhsh

Table of Contents

Introduction

General Domain

Artificial intelligence (AI) refers to machines' capacity to imitate or augment human intelligence, such as thinking and learning from experience. Artificial intelligence has long been utilized in computer programs, but it is also being used in a variety of different products and services.

AI allows you to concentrate on the most important activities and make smarter decisions based on data collected for a specific instance. It is capable of doing difficult tasks such as forecasting maintenance needs, identifying credit card fraud, and calculating the optimal route for a delivery vehicle. In other words, AI can automate numerous commercial activities, making your job more convenient and efficient.

Domain

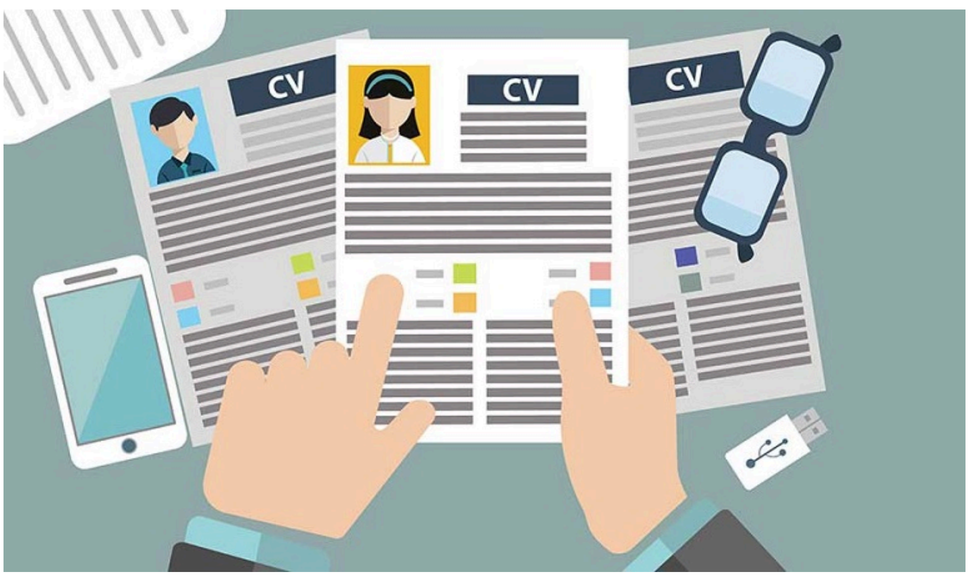
Artificial intelligence (AI) is the power that allows computers to accomplish processes that would need intellect if performed by humans. AI is divided into two key categories: Machine Learning (ML) and Neural Networks (NN). Both are subfields in Artificial Intelligence, and each has its own methodologies and algorithms for problem-solving.

Machine Learning

Machine Learning (ML) is the process through which computers learn from data and experience in order to enhance their efficiency in certain jobs or decision-making processes. For this objective, ML employs statistics and probability theory. Machine learning implements algorithms to process data, learn from it, and make decisions without any explicit programming. Machine learning algorithms are commonly categorized as either supervised or unsupervised. Supervised algorithms may apply previous learning to new data sets, whereas unsupervised algorithms can derive conclusions from datasets. Machine learning algorithms are programmed to seek out linear and non-linear correlations in a set of data. This task is accomplished by the application of statistical tools to train the algorithm to categorize or predict from a dataset.

Problem Statement

Companies receive heaps of resumes and CVs for various job positions. In such a fast paced era where everything has been optimized, recruiters face certain challenges while filtering candidate resumes. Going over a candidate's résumé takes a long time, especially when there are hundreds of jobs with several applicants for each. According to several polls, job advertisements attract an average of 250 applications, thus manually sifting resumes is a waste of time and resources of an organization. If resume screening is not done appropriately, it can be time-consuming and potentially biased. Resumes can also be long and wordy, making it difficult to decipher the true caliber of the applicant and the screening team can potentially overlook minute details that could be crucial.



Literature

Machine Learning

Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values. Machine learning algorithms construct a model from sample data, referred to as training data, in order to make predictions or decisions.

Machine learning algorithms are utilized in a wide range of applications, including medical, email filtering, speech recognition, agriculture, and computer vision, when developing traditional algorithms to execute the required tasks would be difficult or impossible. The iterative aspect of machine learning is important because as models are exposed to new data, they are able to independently adapt. They learn from previous computations to produce reliable, repeatable decisions and results. It’s a science that’s not new – but one that has gained fresh momentum.

Natural Language Processing

Humans communicate with each other using words and text. The way that humans convey information to each other is called Natural Language. Every day humans share a large quality of information with each other in various languages such as speech or text. However, computers cannot interpret this data, which is in natural language, as they communicate in 1s and 0s. The data produced is precious and can offer valuable insights. Hence, you need computers to be able to understand, emulate and respond intelligently to human speech.
While natural language processing isn’t a new science, the technology is rapidly advancing thanks to an increased interest in human-to-machine communications, plus the availability of big data, powerful computing, and enhanced algorithms.

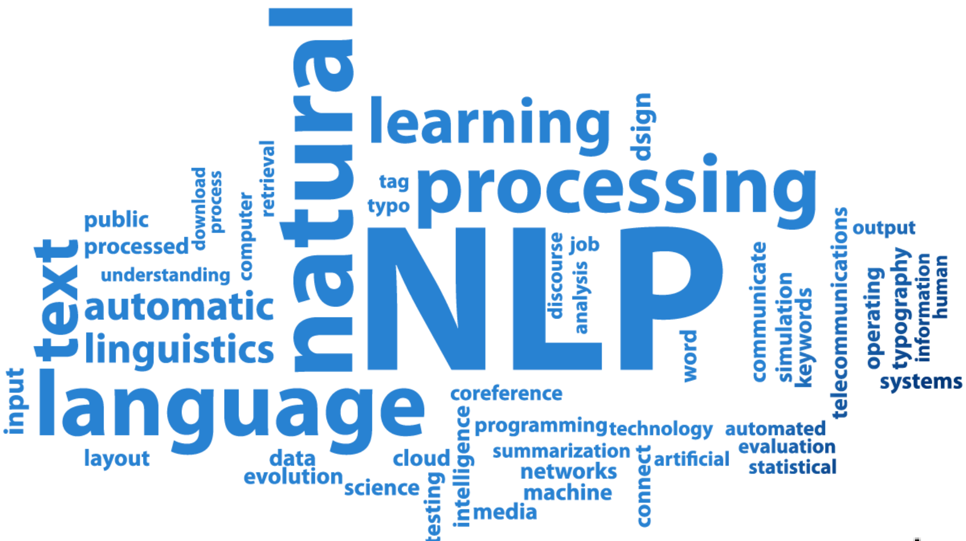
Natural Language Processing or NLP refers to the branch of Artificial Intelligence that gives machines the ability to read, understand and derive meaning from human languages. Natural language processing (NLP) helps computers understand, interpret and manipulate human language.

Machine learning (ML) for natural language processing (NLP) and text analytics involves utilizing machine learning algorithms and "narrow" artificial intelligence (AI) to interpret the meaning of text documents. These papers can be anything that contains text, including social media comments, online reviews, survey replies, and even financial, medical, legal, and regulatory records. In essence, the aim of machine learning and AI in natural language processing and text analytics is to improve, speed, and automate the underlying text analytics algorithms and NLP features that convert this unstructured text into usable data and insights.

Natural Language Processing (NLP) applies two techniques to help computers understand text: syntactic analysis and semantic analysis. Syntactic analysis – or parsing – analyzes text using basic grammar rules to identify sentence structure, how words are organized, and how words relate to each other. The semantic analysis focuses on capturing the meaning of the text. First, it studies the meaning of each individual word (lexical semantics). Then, it looks at the combination of words and what they mean in context.

Proposed Model

To overcome the mentioned issues in the resume short-listing process, this report presents an automated Machine Learning python-based model. After passing the raw data from NLP Pipeline, this model feeds the vectorized features to the Machine Learning Algorithm. This algorithm uses K-Nearest Neighbour Classification to highlight the similarity ratio between the requirements and the resume.



Discussion

We start off by importing all the relevant python libraries that will be used in this model, the description of which is given in the table below.

Library	Description
pandas	Pandas is a software library written for the Python programming language for data manipulation and analysis.
numpy	NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.
matplotlib	Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.
sklearn	Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms.
NLTK	The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language.
wordcloud	Python wordcloud library is used to create tag clouds
seaborn	Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphs.

The dataset named “UpdatedResumeDataSet” is also imported and distinct categories present in the data set are displayed.

```python
# importing the necessary Python libraries and the dataset
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings

warnings.filterwarnings('ignore')
from sklearn.naive_bayes import MultinomialNB
from sklearn.multiclass import OneVsRestClassifier
from sklearn import metrics
from sklearn.metrics import accuracy_score
from pandas.plotting import scatter_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

resumeDataSet = pd.read_csv('UpdatedResumeDataSet.csv', encoding='utf-8')
resumeDataSet['cleaned_resume'] = ''
resumeDataSet.head()
```

```python
print("Displaying the distinct categories of resume -")
print(resumeDataSet['Category'].unique())
```

```
Run:   main
G:\PycharmProjects\AI\venv\Scripts\python.exe G:/PycharmProjects/AI/main.py
Displaying the distinct categories of resume -
['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']

Process finished with exit code 0
```

Version Control   Run   Python Packages   TODO   Python Console   Problems   Terminal   Services

We have a total of 25 unique categories present in the data set.

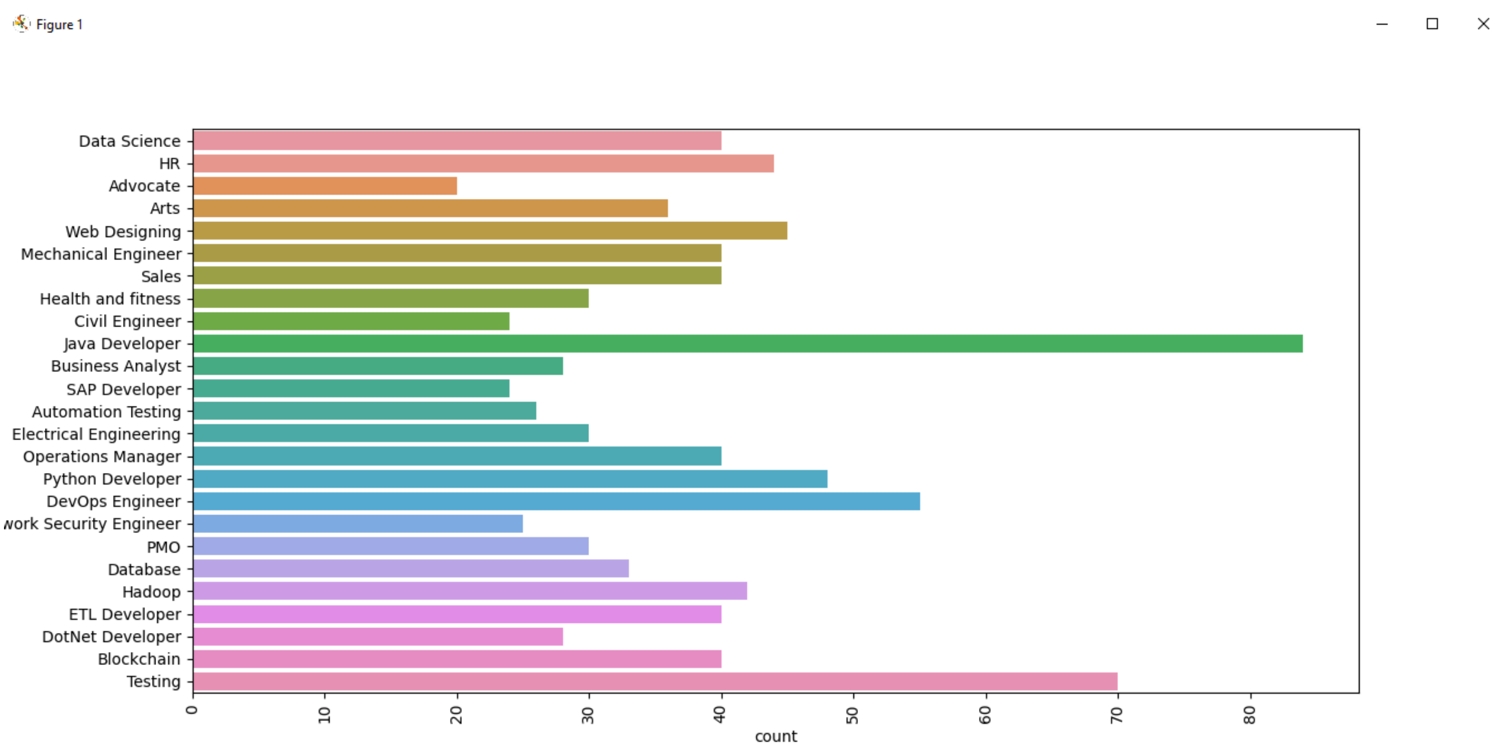
Afterwards we analyze the distinct categories of resume and the number of records belonging to each category.

```python
# the distinct categories of resume and the number of records belonging to each category
print("Displaying the distinct categories of resume and the number of records belonging to each category -")
print(resumeDataSet['Category'].value_counts())
```

```
Displaying the distinct categories of resume and the number of records belonging to each category -
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
```

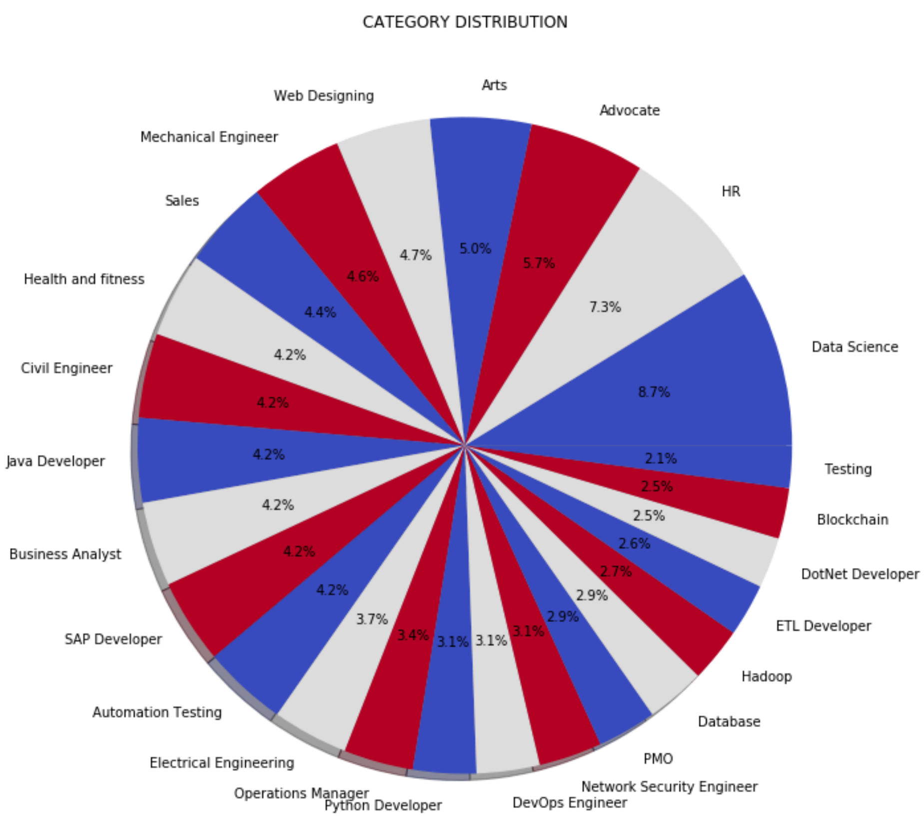
It would be much more convenient to read this data in a graphical format so the above information is further represented in the form of a bar chart by using the seaborn library's countplot function.

```
# visualize the number of categories in the dataset

plt.figure(figsize=(15, 15))
plt.xticks(rotation=90)
sns.countplot(y="Category", data=resumeDataSet)
plt.show()
```



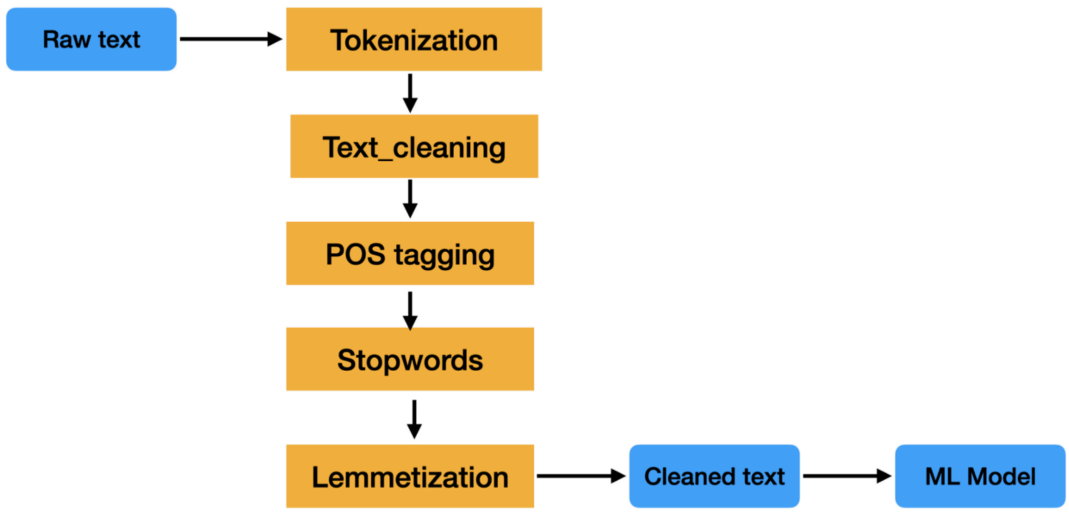
Now we are in a much better position to analyze the frequency of resumes submitted for each job category. Let's further enhance this by visualizing the distribution of categories in the form of a pie chart. The gridspec class of matplotlib library is utilized for this purpose.

```
# visualize the distribution of categories
from matplotlib.gridspec import GridSpec

targetCounts = resumeDataSet['Category'].value_counts()
targetLabels = resumeDataSet['Category'].unique()
# Make square figures and axes
plt.figure(1, figsize=(25, 25))
the_grid = GridSpec(2, 2)

cmap = plt.get_cmap('coolwarm')
colors = [cmap(i) for i in np.linspace(0, 1, 3)]
plt.subplot(the_grid[0, 1], aspect=1, title='CATEGORY DISTRIBUTION')

source_pie = plt.pie(targetCounts, labels=targetLabels, autopct='%1.1f%%', shadow=True, colors=colors)
```



NLP Pipeline

Gathering, sorting, and preparing data is the most important step in the data analysis process – bad data can have cumulative negative effects downstream if it is not corrected. Therefore, we now move forward with NLP pipelining. Starting from taking the raw data and cleaning it.



Text Cleaning

Clean text is human language rearranged into a format that machine models can understand. Text cleaning can be performed using simple Python code that eliminates stopwords, removes unicode words, punctuation, URLs etc. and simplifies complex words to their root form.

```python
# create a helper function to remove the URLs, hashtags, mentions, special letters, and punctuations
import re
def cleanResume(resumeText):
    resumeText = re.sub('http\S+\s*', ' ', resumeText)  # remove URLs
    resumeText = re.sub('RT|cc', ' ', resumeText)  # remove RT and cc
    resumeText = re.sub('#\S+', '', resumeText)  # remove hashtags
    resumeText = re.sub('@\S+', '  ', resumeText)  # remove mentions
    resumeText = re.sub('[%s]' % re.escape("""!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"""), ' ',
                        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))
```

Now that the data set has been cleared, we will have a look at the Wordcloud. A word cloud is a visual representation of information or data. It shows the popularity of words or phrases by making the most frequently used words appear larger or bolder compared with the other words around them. Along with that we will be performing tokenization which is the next step in NLP Pipelining.

Tokenization

A tokenizer breaks unstructured data and natural language text into chunks of information that can be considered as discrete elements. The token occurrences in a document can be used directly as a vector representing that document. For example the sentence 'I am a student' would be tokenized as [I, am, a, student].

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

oneSetOfStopWords = set(stopwords.words('english') + ['``', "''"])
totalWords = []
Sentences = resumeDataSet['Resume'].values
cleanedSentences = ""
for i in range(0, 160):
    cleanedText = cleanResume(Sentences[i])
    cleanedSentences += cleanedText
    requiredWords = nltk.word_tokenize(cleanedText)
    for word in requiredWords:
        if word not in oneSetOfStopWords and word not in string.punctuation:
            totalWords.append(word)

wordfreqdist = nltk.FreqDist(totalWords)
mostcommon = wordfreqdist.most_common(50)
print(mostcommon)

wc = WordCloud().generate(cleanedSentences)

plt.figure(figsize=(15, 15))
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
```

This is the word cloud generated. We can clearly conclude that the frequently used words are bolder and bigger like Experience, details, company, skill, months, Data Science and so on.

Now these words will be converted into categorical values. A categorical variable is a value that assumes a limited and fixed number of possible values, allowing a data unit to be assigned to a broad category for classification. Assigning each individual datapoint under observation to a labeled category is the first step in supervised deep learning. However, these values are distinct from each other unlike nominal values.
NLTK Library will be used for tokenization of our data set.

```
# convert these words into categorical values
from sklearn.preprocessing import LabelEncoder

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

Vectorization

In programming, a vector is a data structure that is similar to a list or an array. For the purpose of input representation, it is simply a succession of values, with the number of values representing the vector's "dimensionality." Vector representations contain information about the qualities of an input object. They offer a uniform format that computers can easily process. So basically we take human understandable language and turn it into machine understandable language in vectorization.

Hence we use TfidfVectorizer to convert our tokenized data into vectorized features to make it machine understandable.

Training Machine Learning Model

Next we will be Training Machine Learning Model for Resume Screening. But before that the data is split in training and test sets using the train_test_split function.

The algorithm implemented here is one vs the rest classifier; KNeighborsClassifier

K-Nearest Neighbor Algorithm

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

In order to determine which data points are closest to a given query point, the distance between the query point and the other data points need to be calculated. There are two common metrics available for this: Euclidean distance and Manhattan distance. The k value in the k-NN algorithm defines how many neighbors will be checked to determine the classification.

```
# train a model for the task of Resume Screening
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from scipy.sparse import hstack

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

word_vectorizer = TfidfVectorizer(
    sublinear_tf=True,
    stop_words='english',
    max_features=1500)
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=0, test_size=0.2)
print(X_train.shape)
print(X_test.shape)
```

Finally we train the model and print the classification report.

```python
# train the model and print the classification report
clf = OneVsRestClassifier(KNeighborsClassifier())
clf.fit(X_train, y_train)
prediction = clf.predict(X_test)
print('Accuracy of KNeighbors Classifier on training set: {:.2f}'.format(clf.score(X_train, y_train)))
print('Accuracy of KNeighbors Classifier on test set: {:.2f}'.format(clf.score(X_test, y_test)))

print("\n Classification report for classifier %s:\n%s\n" % (clf, metrics.classification_report(y_test, prediction)))
```

Implementation

This model can now be implemented by recruiters to help optimize their recruitment drives. The ML based model will filter out the resumes based on job requirements and will screen out the ones that are not relevant to the position. It will keep the resumes of people with high expertise that best match the job description.

The Model can be used by different organizations and companies without a need to hire a screening team.

Results

The results show a 99% accuracy on the test data set which shows this model is successful.

```
Accuracy of KNeighbors Classifier on training set: 0.99
Accuracy of KNeighbors Classifier on test set: 0.99

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

           0       1.00      1.00      1.00         3
           1       1.00      1.00      1.00         3
           2       1.00      0.80      0.89         5
           3       1.00      1.00      1.00         9
           4       1.00      1.00      1.00         6
           5       0.83      1.00      0.91         5
           6       1.00      1.00      1.00         9
           7       1.00      1.00      1.00         7
           8       1.00      0.91      0.95        11
           9       1.00      1.00      1.00         9
          10       1.00      1.00      1.00         8
          11       0.90      1.00      0.95         9
          12       1.00      1.00      1.00         5
```

Conclusion

In the current times when AI is the most relevant thing, many corporate and commercial tasks can be made more efficient by great folds with the help of Machine Learning and other interesting domains of Artificial Intelligence. The Resume Screening Model for instance is a program of a mere 100 words but it can help recruiters with hundreds of Resumes in a single moment. Therefore, the most logical solutions of the real-world problems lie within such intelligent models.