Course: CSCE 5214 Section 002 - Software Development for Artificial Intelligence (Fall 2022)

Resume Classification Using NLP

Project Report

Instructor: *Dr. Russel Pears* (Russel.Pears@unt.edu)

Team Members

1. Bhashwitha Kolluri - 11526264 (bhashwithakolluri@my.unt.edu)
2. Harish Kashyap Vutukuru - 11518964 (harishkashyapvutukuru@my.unt.edu)
3. Manoj Kolluri - 11524958 (manojkolluri@my.unt.edu)
4. Santhoshi Kareddy - 11600006 (SanthoshiKareddyVIII@my.unt.edu)

GitHub Repository

https://github.com/bhashwitha/SDAI---PROJECT---1---Resume-Classification.git

TABLE OF CONTENT:

Abstract

In a normal online job advertisement, many applications are received in a short period of time. Manual resume screening is impractical because it requires too much time and money, both of which recruiting companies cannot afford. Because of the process of screening resumes, many competent persons may not receive the attention they deserve. This might result in the employment of unsuitable individuals or the rejection of good ones. In our project, we present a system that seeks to solve these problems by proposing the best applicants based on the job description.

As part of our project, we used natural language processing to extract critical data from resumes, such as skills, education, and experience, and then used that data to create a more streamlined version of each application. Recruiters will be able to properly assess each resume in less time if all superfluous content is removed, expediting the screening process. Following the completion of the text mining process, the technique employs a vectorization model and a cosine similarity model to match the resumes with the job description. The top candidates for the post may then be identified using the ranking scores generated depending on how well they suit the role.

Literature Review

1. **Domain Adaptation for Resume Classification Using Convolutional Neural Networks** [1]

This article suggests a method for sorting resumes into 27 different job categories using a convolutional neural network. Due to the sensitive nature of resume data, domain adaptation provides a cost-effective and dependable solution. Our method entails training a classifier on several publicly accessible job descriptions and then applying it to resume data. Despite a limited quantity of labelled resume data, we show that our technique has a respectable classification performance.

2. **Robotic Process Automation for Resume Processing System** [2]

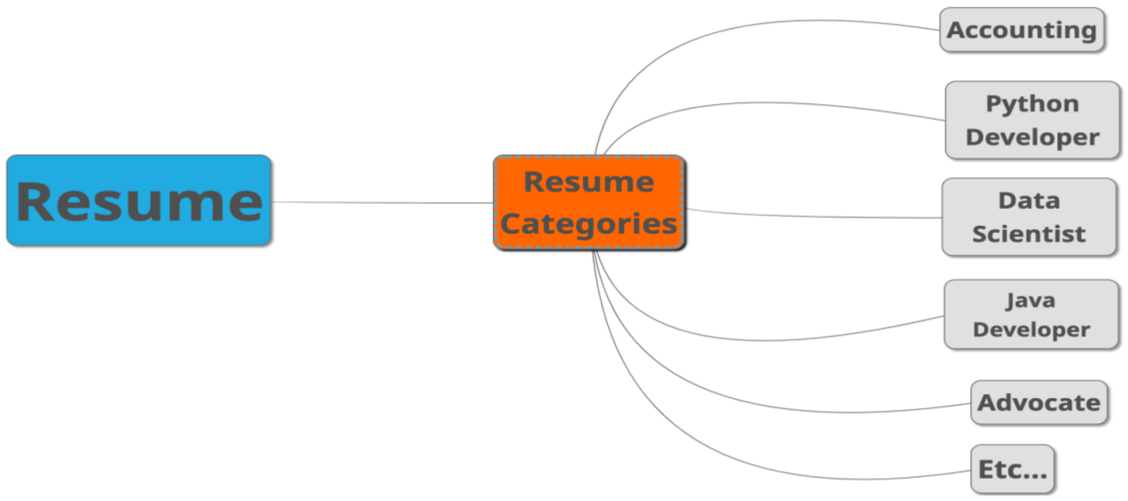
In this study, the author employs robotic process automation and machine learning approaches to aid with recruiting by automating such interview procedures. When a new email or file is received, the RPA bot downloads the attachments and then classifies them to see if they are resumes. The resumes are shortlisted based on their talents, experience, and educational background, and named entity recognition (NER) is used to extract the relevant facts from the attachments. The precision of the Naive Bayes Classifier is 92.08% for text classification, 91.36% for document classification on BERT, 86.2% for custom-trained NER, and 91.5% for web scanning.

3. **Resume classification system using natural language processing and machine learning techniques** [3]

The author provides a solution with improved accuracy and dependability in diverse contexts in the study, which makes use of machine learning algorithms and natural language processing approaches. To demonstrate the usefulness of NLP and machine learning approaches in RCS, nine ML classification models were used to analyze the retrieved characteristics. These models included the Support Vector Machine, Naive Bayes, KNN, and Logistic Regression. The created models were evaluated using the Confusion Matrix, F-Score, Accuracy, Precision, and Recall. On the research dataset of over 960 parsed resumes, the SVM class of Machine Learning classifiers performed better with an accuracy of 96% than the rest classifiers using the One-vs-Rest-Classification method. This study shows that NLP and Machine Learning may be utilized to create an efficient RCS with promising results.

Architecture Diagram

The architectural diagram for Resume Classification is depicted in the image below. Therefore we provide the input as documents or pdf, which is a dataset containing the profession description, experience, education, and a few phrases and sentences. After training the models on this dataset, we preprocess it using stop words, stemming, and lemmatization. The preprocessing is then separated into two methods, one for machine learning and the other for neural networks. We train the models using machine learning approaches such as decision tree models, Adaboost, and CNN. Then compute the accuracy rate and compare the accuracy of the models. A confusion matrix is used to improve accuracy. The result of these models is the classified resume.



Workflow

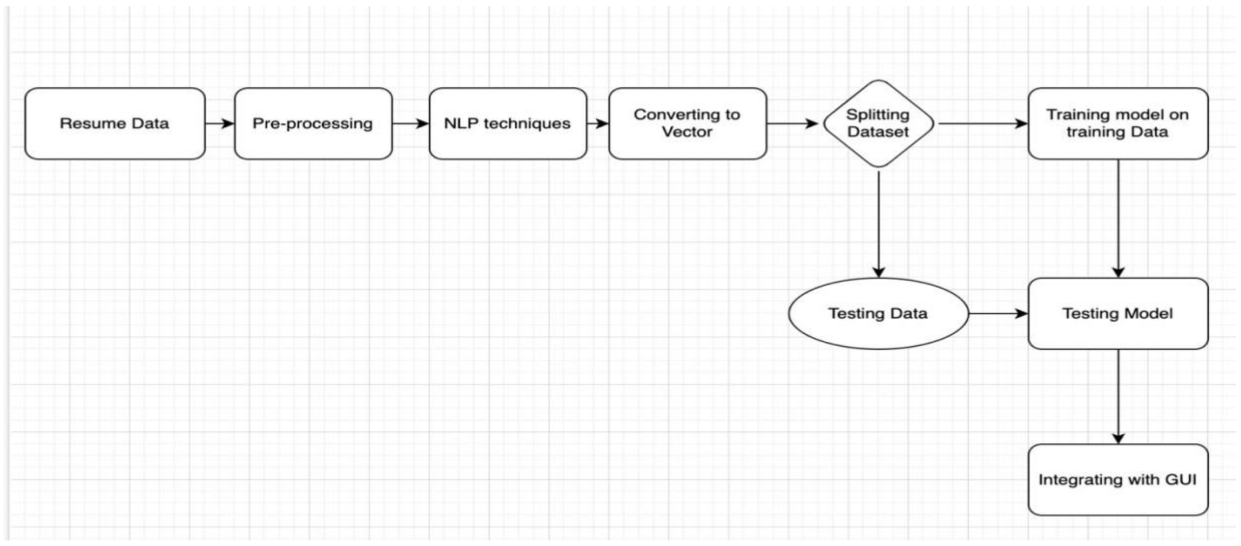
The preceding diagram depicts the Resume Detection process. In this stage, we offered the project with certain resume categorization methods. We will divide the workflow into four sections to make it easier to grasp. In the first section, we have two components: data gathering and data preparation, which we employed for both the unlabeled labeled and sections. When we use NLP techniques like stop words, stemming, and lemmatization to handle this unlabeled and labeled data. In phase two, the unlabeled data is automatically preprocessed and turned to labeled data using the active learning approach. This active learning is used to calculate the likelihood of the quality data set. The third section contains a Feature selection. It includes the TF-IDF, N-gram, and Word2Vector. This feature selection is separated into two stages: deep learning and machine learning. We chose the TF-IDF and N-gram models for machine learning. We employed the Adaboost, Nave Bayes, and Decision tree machine learning approaches. We assess the accuracy and confusion matrix for the machine learning model while also checking the accuracy for the LSTM and GRU. The findings are documented.

EDA, Pre-Processing, Model Training:

- In the first step, an exploratory data analysis is done on the dataset, and the dataset is also verified for missing data and imbalances in data distribution.
- If the dataset is unbalanced, procedures such as sampling are utilized in the first preparation step, and any null values that exist are deleted.
- Before converting data to vector form, NLP methods such as stop word removal and other techniques like as stemming and lemmatization are utilized in the following stage of preprocessing.
- To evaluate the models, a record of the usual performance measures, such as the confusion matrix, the ROC, the F1-score, precision, sensitivity, and accuracy, will be recorded.

UI Integration:

- When a user submits a resume, the backend does the necessary preprocessing and NLP algorithms.
- In addition, the content of the resume is run through a fully trained model, which forecasts the outcome.
- The resume result is then shown on the front end by the UI.



UI -Development:

1. For the development of our UI application, we used React JS.

2. Worked using Visual Studio Code as our Integrated Development Environment (IDE)

3. we have used standard HTML tables and CSS.

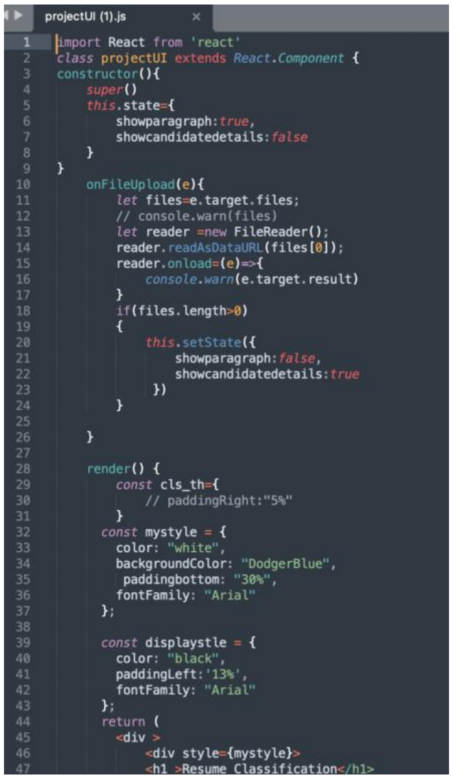
4. To upload files, we used file upload controllers

5. Initially, the user will see a button that enables them to search for and upload the required file.

6. To prevent the processing of incorrect files, such as audio, video, and image files, conditional rendering is used

7. After the process is complete, we can obtain information about the candidate in columns such as their serial number, first and last names, qualifications, areas of expertise, and a percentage that indicates how closely they fit the job description.

8. When the file is uploaded and the entire procedure is properly completed, the outcome seen above is what may be anticipated.



```
import React from 'react'
class projectUI extends React.Component {
constructor(){
    super()
    this.state={
        showparagraph:true,
        showcandidatedetails:false
    }
}
    onFileUpload(e){
        let files=e.target.files;
        // console.warn(files)
        let reader =new FileReader();
        reader.readAsDataURL(files[0]);
        reader.onload=(e)=>{
            console.warn(e.target.result)
        }
        if(files.length>0)
        {
            this.setState({
                showparagraph:false,
                showcandidatedetails:true
             })
        }

    }

    render() {
        const cls_th={
            // paddingRight:"5%"
        }
      const mystyle = {
        color: "white",
        backgroundColor: "DodgerBlue",
         paddingbottom: "30%",
        fontFamily: "Arial"
      };

      const displaystle = {
        color: "black",
        paddingLeft:'13%',
        fontFamily: "Arial"
      };
      return (
        <div >
            <div style={mystyle}>
            <h1 >Resume Classification</h1>
```

Design & Milestones

We employed Python programming language and Natural Language Processing techniques in a local environment using Anaconda Navigator and Jupiter Notebook to accomplish the project's objectives and milestones. During the first phase of the project, we worked on the data analysis and pre-processing sections. We first performed exploratory data analysis on the raw data and observed the most popular words preferred in each category of the resume. After that, we analyzed how evenly the dataset is distributed across all its categories. After performing the initial analysis, the next step was to perform data pre-processing; for this part of the project, we planned to employ methods like stop words removal, lemmatization, and stemming using the NLTK library. We accomplished this using Python's NumPy, pandas, matplotlib, and seaborn libraries. After the project's pre-processing phase is completed, we converted the cleaned data into vector form and then use TensorFlow to build a basic neural network and train it on the dataset. Finally, we assessed the decision tree model and CNN model performance metrics and incorporate it into a user-friendly interface designed with HTML and CSS and React.JS

Milestones:

1. On the dataset, run EDA.
2. As the dataset is not balanced, used preprocessing procedures to tidy it up.
3. Applied NLP methods to the data, such as lemmatization and stemming, using the NLTK library
4. Changed the data's format to vector.
5. Created a separate training and testing formats for the data.
6. Developed a fully linked neural network.
7. Tuning hyperparameters
8. Using assessment metrics, the decision tree model or CNN, is assessed after being trained, tested, and used on the dataset.
9. The model will be integrated into a GUI

Data specification

Data Collection:

The Kaggle resume classification dataset comprises 962 resumes that contain elements from several employment categories. These publications are classified into 25 categories such as Java Developer, Testing, DevOps Engineer, Python Developer, and Web Designing. The distribution of these materials is seen to be equal. The diagram below depicts resume categorization across the 25 job roles. There are often 40 to 50 resumes in each category for each resume that fits that category. We worked on the decision tree model and CNN model as it is supervised learning problem. The features of resume are Experience, Education, Career Objective, Skills, Tools, and Technologies, and so on.

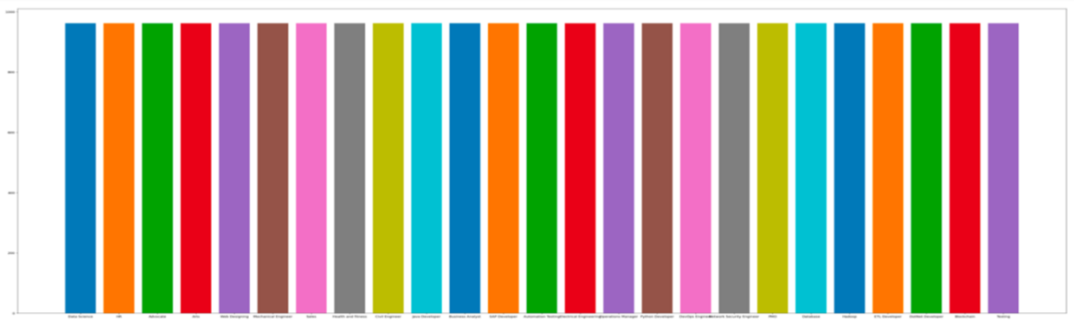
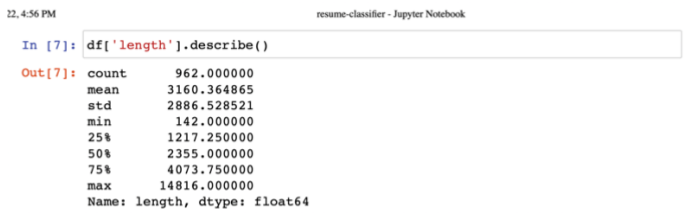


FIG-Resume Categorization across the 25 job roles

Design Features:

Figure, which depicts the descriptive statistical analysis of the length of each job category's resumes in the dataset, shows that the job category with the most resumes in the dataset comprises an aggregate of 50 resumes.



22, 4:56 PM resume-classifier - Jupyter Notebook

```
In [7]: df['length'].describe()

Out[7]: count      962.000000
        mean      3160.364865
        std       2886.528521
        min        142.000000
        25%       1217.250000
        50%       2355.000000
        75%       4073.750000
        max      14816.000000
        Name: length, dtype: float64
```

ANALYSIS OF DATA

Data Pre-Processing:

- In the preprocessing stage, we first check to see if there is any missing information in the data frames that contain all the resume and labels in the dataset.
- In the preprocessing stage, we first check to see if there is any missing information in the data frames that contain all the resume and labels in the dataset.
- We first convert all the alphabets in the email to lowercase letters from capitalized letters to normalize the text and reduce the variability since the computations may be case sensitive.
- We next remove any punctuation and special characters from the text to further reduce the variations.
- We also delete terms with less than three letters since they may not be relevant in sequence.
- Finally, we remove extra spaces, quotations, mild pronouns, and various characters like as \n and \t.
- We next use lemmatization to the text before removing stop-words to reduce undesired and unhelpful terms and reduce computation costs during training.
- Following per-handling, we encode the data and divide it into training and testing sets using a stratified split.

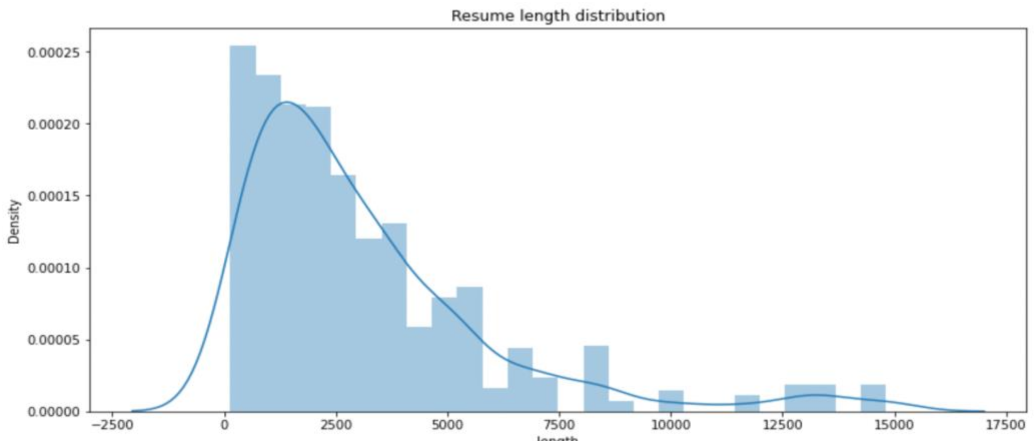
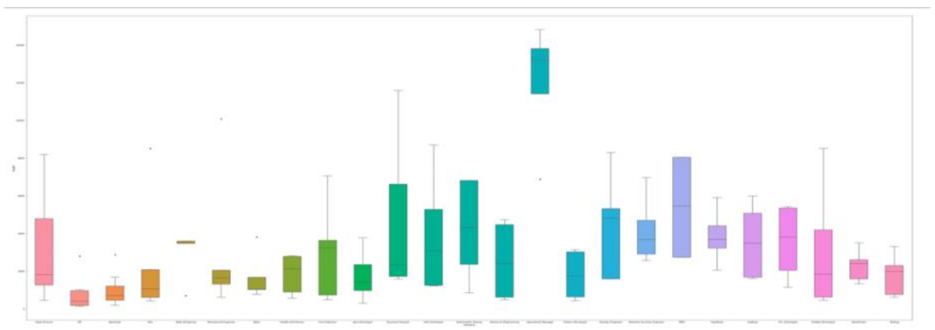
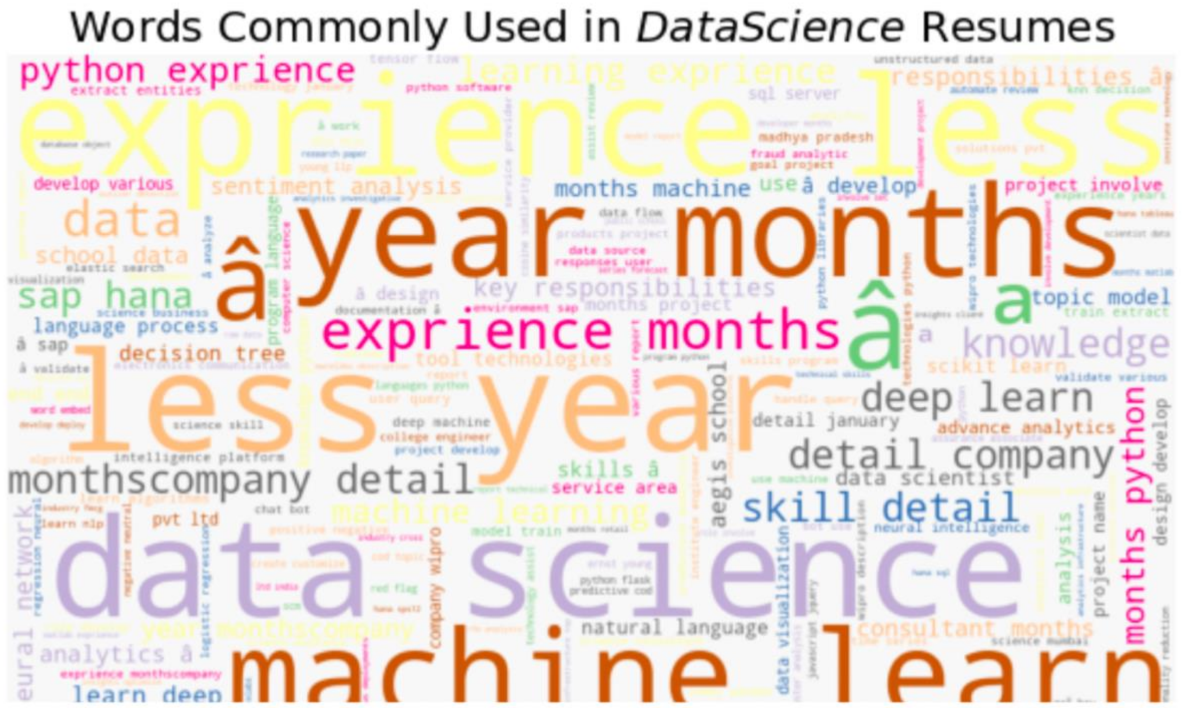


Figure Illustrates the boxplot of length distribution

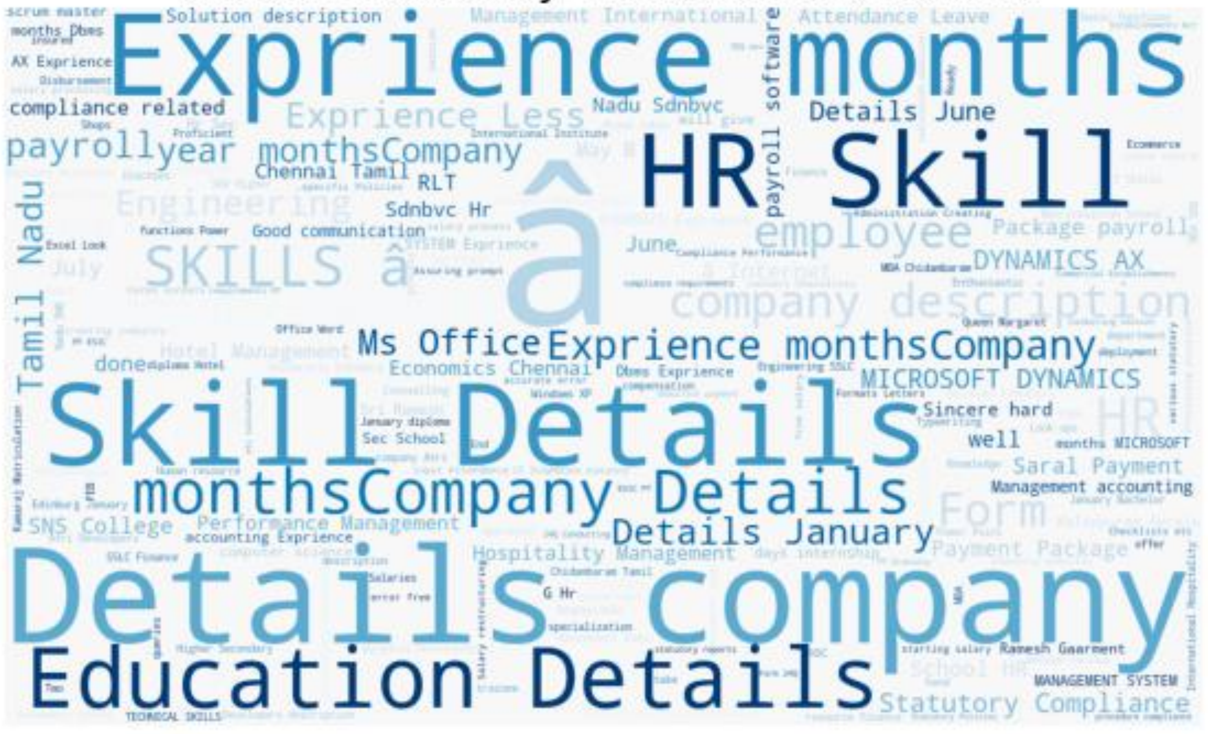


We then plotted the word count distribution of the items in the resumes having a place with every classification utilizing box plot to discover whether there is an overall distinction in the quantity of expressions of description having a place with the 25 classifications as the quantity of words could likewise be utilized as a variable by various calculations or might prompt a bias inclination when we attempt to group the messages in the future. We also attempted to imagine the most used terms in resumes fitting into the 25 categories.

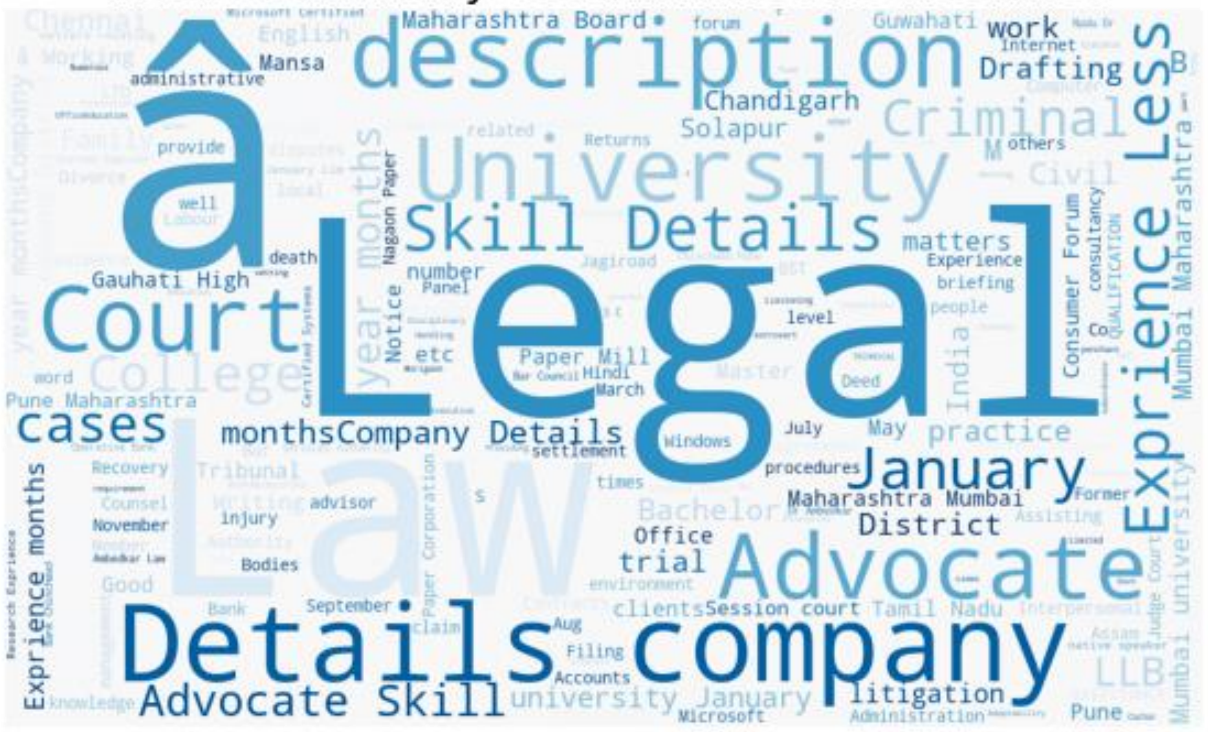
Figure illustrates the most used words in resumes that have a position in the resume classification.



Words Commonly Used in *HR* Resumes



Words Commonly Used in *Advocate* Resumes

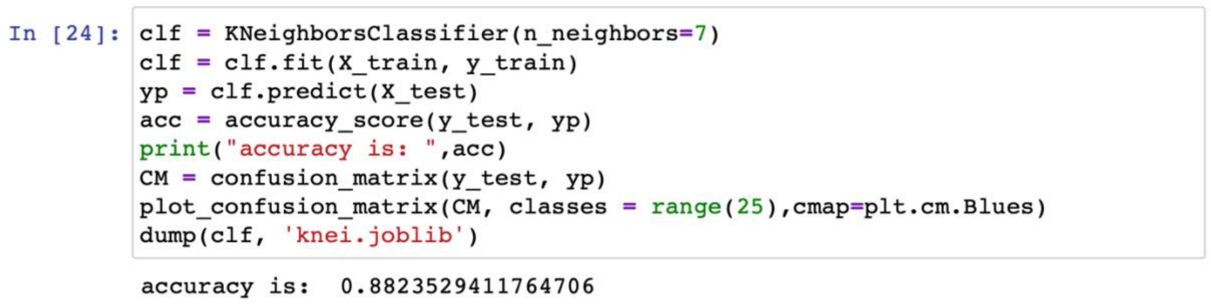


IMPLEMENTATION

We used three Machine Learning models and one Neural Network model to organize the datasets and examine their results. The Machine Learning models we used were the K-Neighbors classifier, Decision tree classifier, and AdaBoost classifier. CNN is the neural network model that we used.

K-Neighbors Model

This method also made use of the same data that was used to train the previous models. We used the sci-kit learn package to create a K-neighbors and trained it using the train set shown in the excerpt below. The above-mentioned model is then trained and tested based on this data.

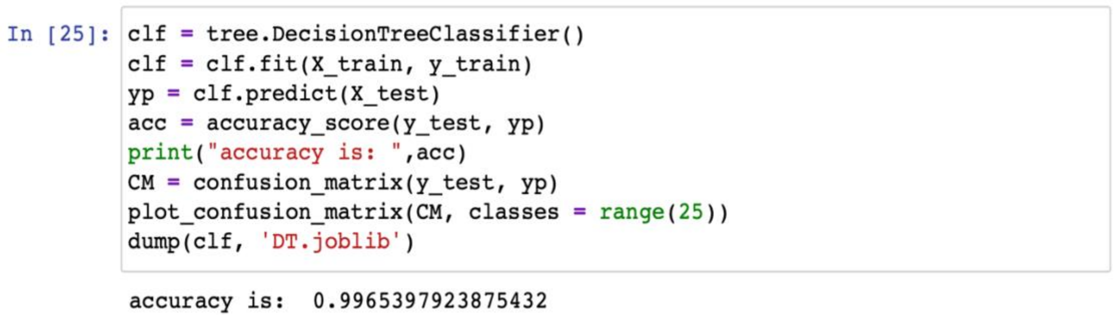


```
In [24]: clf = KNeighborsClassifier(n_neighbors=7)
         clf = clf.fit(X_train, y_train)
         yp = clf.predict(X_test)
         acc = accuracy_score(y_test, yp)
         print("accuracy is: ",acc)
         CM = confusion_matrix(y_test, yp)
         plot_confusion_matrix(CM, classes = range(25),cmap=plt.cm.Blues)
         dump(clf, 'knei.joblib')

         accuracy is:  0.8823529411764706
```

Decision tree Model:

The same data that was used to train the previous models was also used to train this algorithm. We chose the tree Model using the sci-kit learn library and trained it using the train set, as seen in the excerpt below. The above-mentioned model is then trained and tested based on this data.

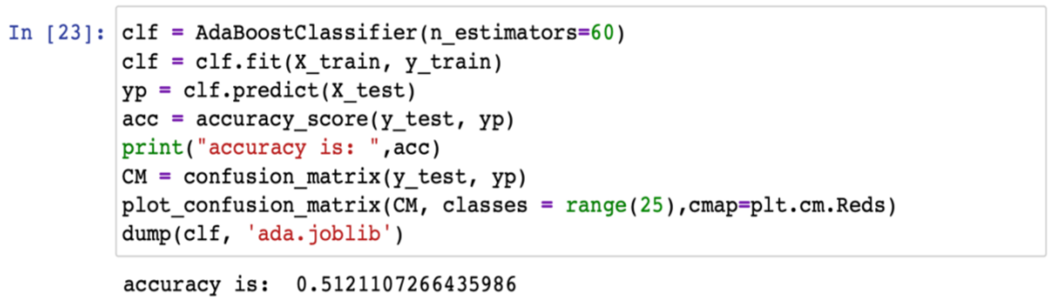


```
In [25]: clf = tree.DecisionTreeClassifier()
         clf = clf.fit(X_train, y_train)
         yp = clf.predict(X_test)
         acc = accuracy_score(y_test, yp)
         print("accuracy is: ",acc)
         CM = confusion_matrix(y_test, yp)
         plot_confusion_matrix(CM, classes = range(25))
         dump(clf, 'DT.joblib')

         accuracy is:  0.9965397923875432
```

AdaBoost Model

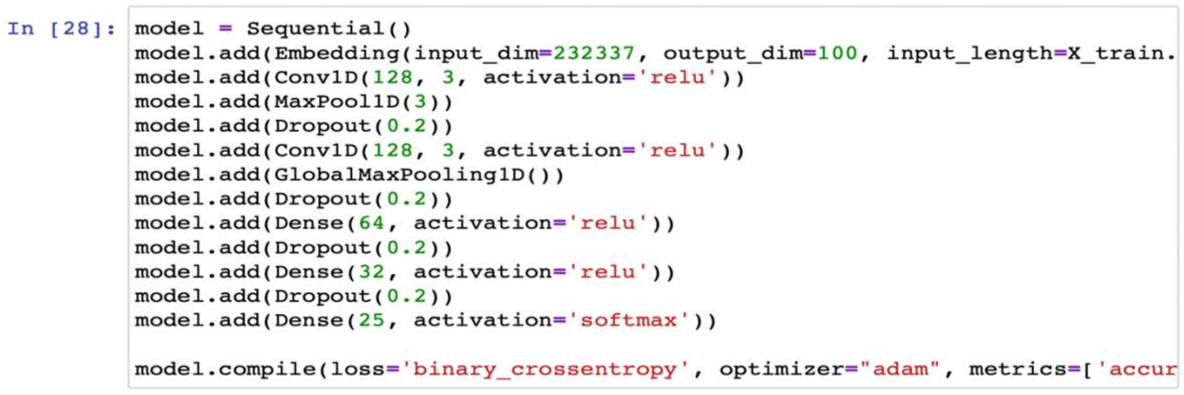
It is an assisting approach used as an ensemble technique in AI. It is also known as adaptive boosting since all the weights are reassigned to each occurrence. Each resultant learner is derived from previously generated learners. Every one of the weak learners has been entirely transformed into strong learners. To begin, it settles on several decision trees while data training is guided. As a result of the decision tree, the model's incorrectly described record is granted a need. Only incorrectly categorized records are provided as contributions to the next model. The interaction continues until the specified number of base learners is reached.This method made use of the same information that was used to train the previous models. We used the sci-kit learn package to create an Adaboost Model and trained it using the train set, as seen in the sample below.


```
In [23]: clf = AdaBoostClassifier(n_estimators=60)
         clf = clf.fit(X_train, y_train)
         yp = clf.predict(X_test)
         acc = accuracy_score(y_test, yp)
         print("accuracy is: ",acc)
         CM = confusion_matrix(y_test, yp)
         plot_confusion_matrix(CM, classes = range(25),cmap=plt.cm.Reds)
         dump(clf, 'ada.joblib')

         accuracy is:  0.5121107266435986
```

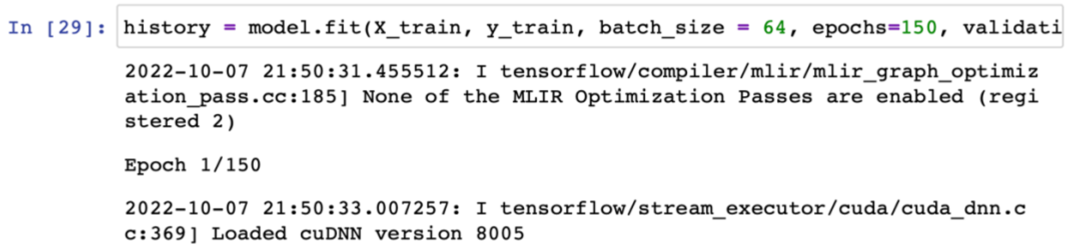
1-Dimenssional – CNN Model:

Convolutional neural network models were created for image classification issues, in which the model learns an internal representation of a two-dimensional input through a process known as feature learning. This approach may also be used to one-dimensional data sequences, such as acceleration and gyroscopic data for human activity identification. The model learns to extract characteristics from observation sequences as well as how to map internal features to distinct activity kinds. The advantage of utilizing CNNs for sequence classification is that they may learn directly from raw time series data, eliminating the need for domain expertise to create input characteristics.



```
In [28]: model = Sequential()
         model.add(Embedding(input_dim=232337, output_dim=100, input_length=X_train.
         model.add(Conv1D(128, 3, activation='relu'))
         model.add(MaxPool1D(3))
         model.add(Dropout(0.2))
         model.add(Conv1D(128, 3, activation='relu'))
         model.add(GlobalMaxPooling1D())
         model.add(Dropout(0.2))
         model.add(Dense(64, activation='relu'))
         model.add(Dropout(0.2))
         model.add(Dense(32, activation='relu'))
         model.add(Dropout(0.2))
         model.add(Dense(25, activation='softmax'))

         model.compile(loss='binary_crossentropy', optimizer="adam", metrics=['accur
```

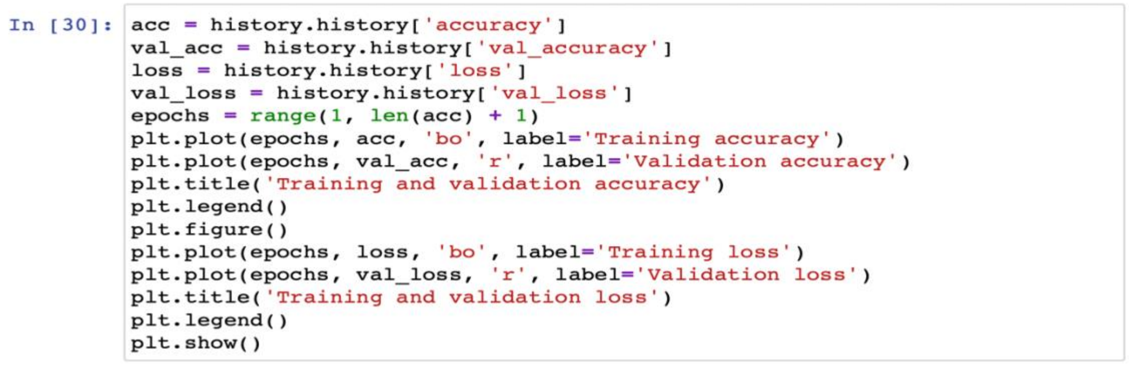


```
In [29]: history = model.fit(X_train, y_train, batch_size = 64, epochs=150, validati

         2022-10-07 21:50:31.455512: I tensorflow/compiler/mlir/mlir_graph_optimiz
         ation_pass.cc:185] None of the MLIR Optimization Passes are enabled (regi
         stered 2)

         Epoch 1/150

         2022-10-07 21:50:33.007257: I tensorflow/stream_executor/cuda/cuda_dnn.c
         c:369] Loaded cuDNN version 8005
```

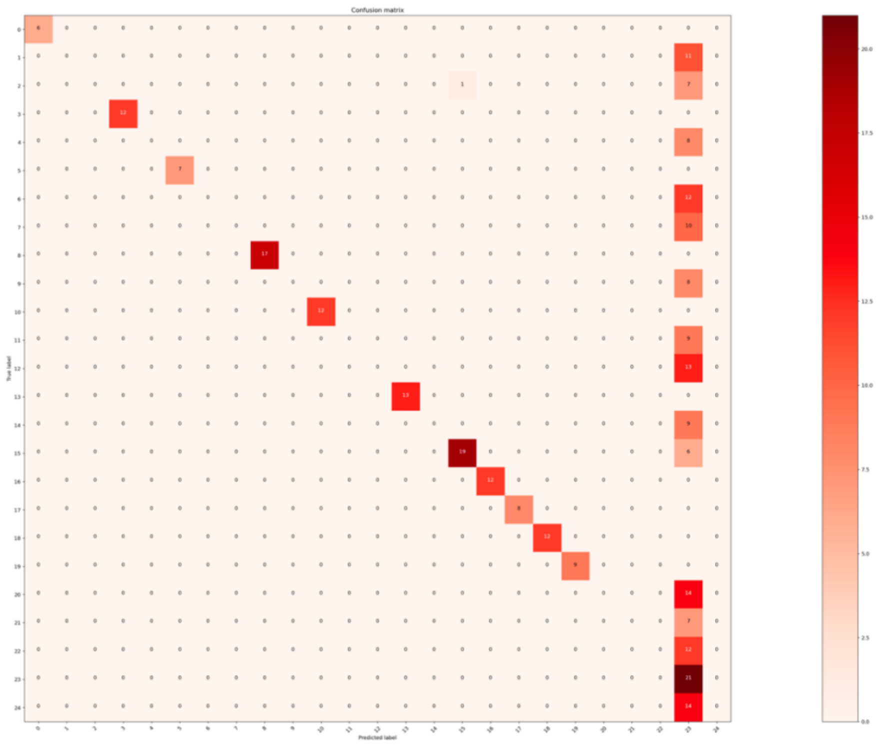


```
In [30]: acc = history.history['accuracy']
         val_acc = history.history['val_accuracy']
         loss = history.history['loss']
         val_loss = history.history['val_loss']
         epochs = range(1, len(acc) + 1)
         plt.plot(epochs, acc, 'bo', label='Training accuracy')
         plt.plot(epochs, val_acc, 'r', label='Validation accuracy')
         plt.title('Training and validation accuracy')
         plt.legend()
         plt.figure()
         plt.plot(epochs, loss, 'bo', label='Training loss')
         plt.plot(epochs, val_loss, 'r', label='Validation loss')
         plt.title('Training and validation loss')
         plt.legend()
         plt.show()
```

RESULT

AdaBoost Model:

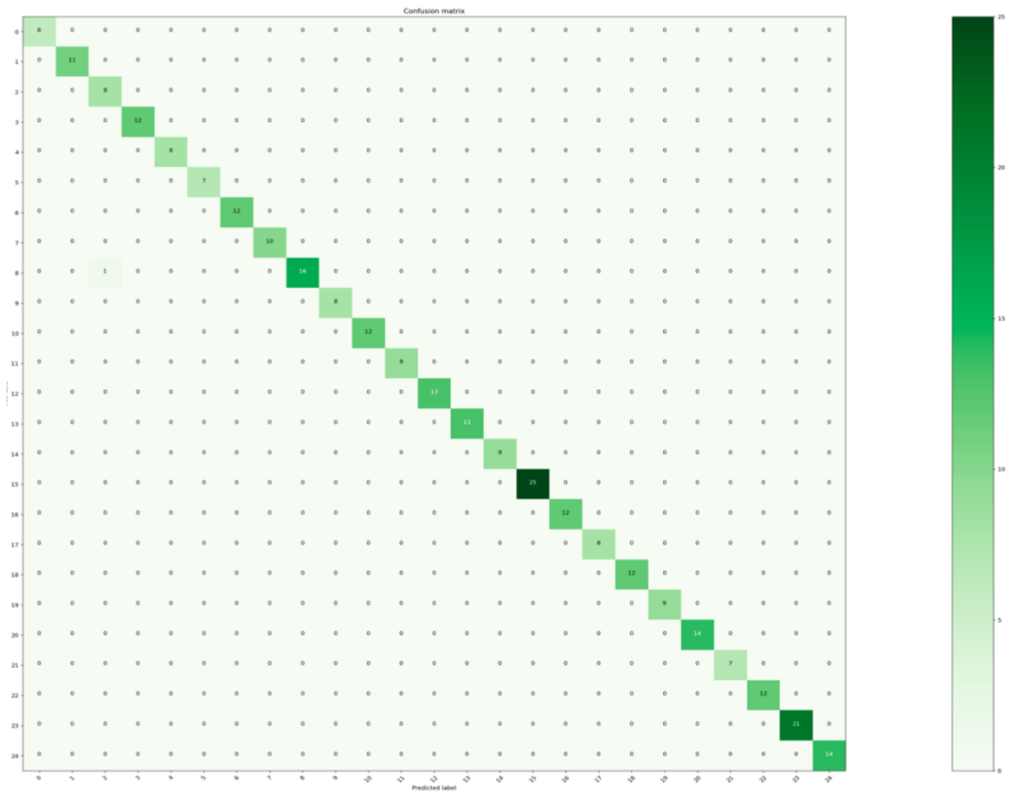
The Confusion Matrix is used as an assessment metric for the Model. The confusion matrix in Figure depicts the model's performance implications for the testing dataset.



Confusion Matrix after using AdaBoost

Decision Tree Model:

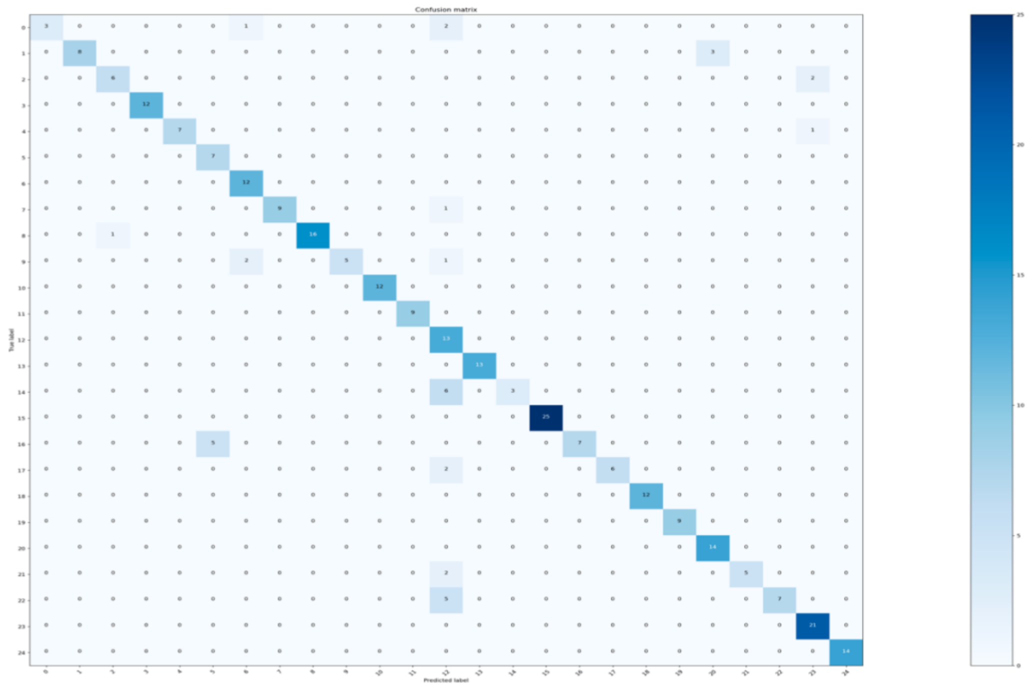
The Confusion Matrix is used as an assessment metric for the Model. The confusion matrix in Figure depicts the model's performance implications for the testing dataset.



Confusion Matrix after using Decision Tree

K-Neighbors Model:

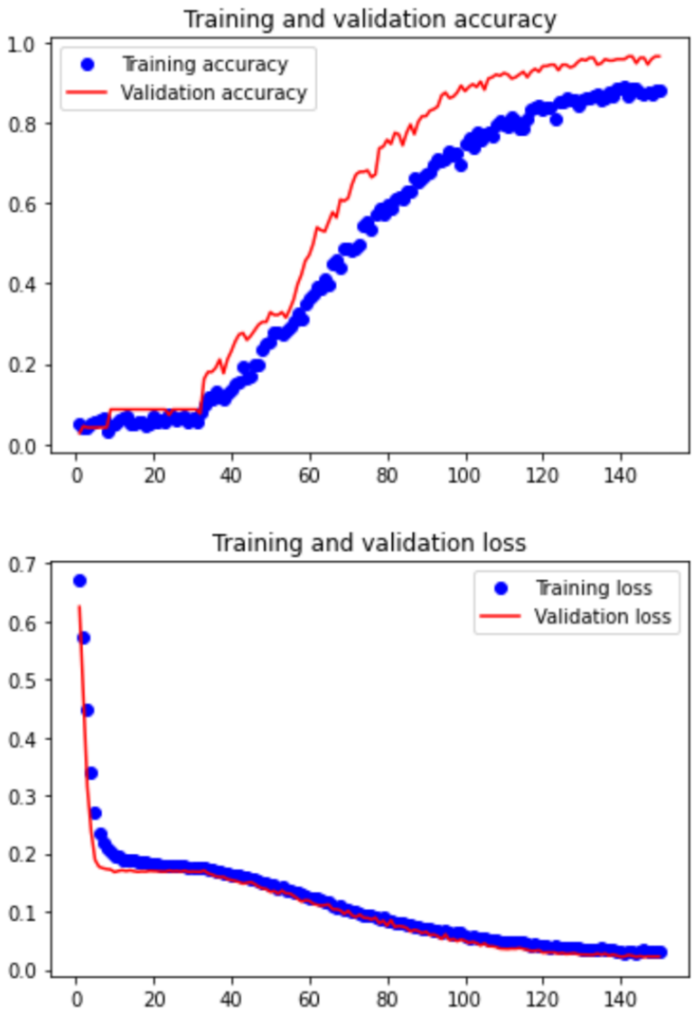
The Confusion Matrix is used as an assessment metric for the Model. The confusion matrix in Figure depicts the model's performance implications for the testing dataset.



Confusion Matrix after using KNeighbors Classifier

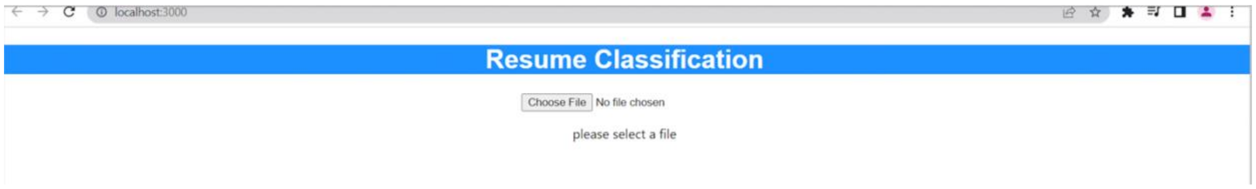
1-Dimenssional - CNN:

Figure shows the charts generated by the CNN Neural Network model for Loss and Accuracy measurements on the training set and assessed on the testing set.

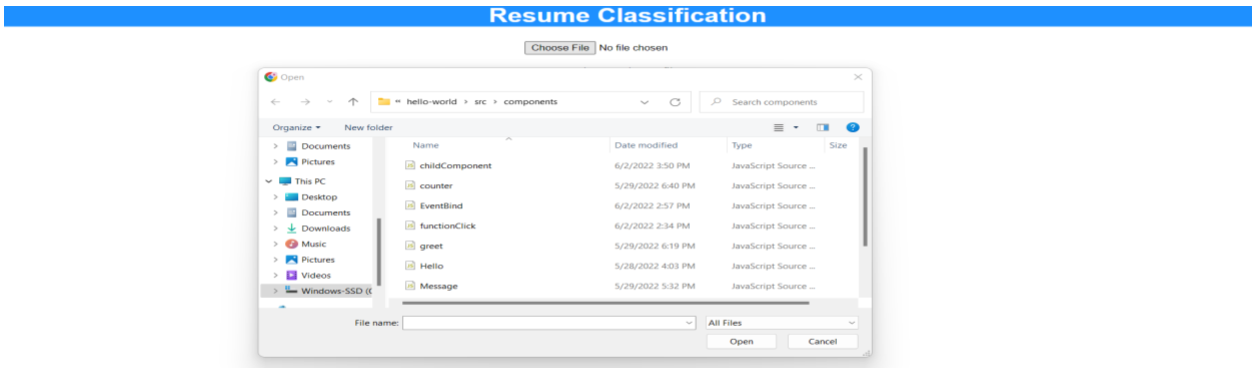


Accuracy and Loss result using CNN.

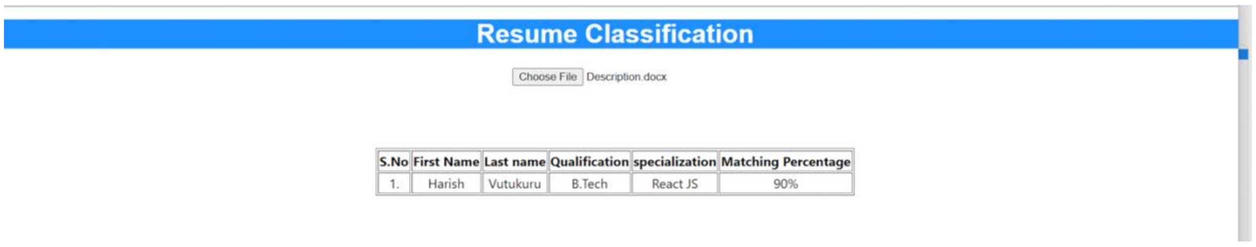
User Interface:



please select a file



Name	Date modified	Type	Size
childComponent	6/2/2022 3:50 PM	JavaScript Source ...	
counter	5/29/2022 6:40 PM	JavaScript Source ...	
EventBind	6/2/2022 2:57 PM	JavaScript Source ...	
functionClick	6/2/2022 2:34 PM	JavaScript Source ...	
greet	5/29/2022 6:19 PM	JavaScript Source ...	
Hello	5/28/2022 4:03 PM	JavaScript Source ...	
Message	5/29/2022 5:32 PM	JavaScript Source ...	



S.No	First Name	Last name	Qualification	specialization	Matching Percentage
1.	Harish	Vutukuru	B.Tech	React JS	90%

References

1. Sayfullina, L., Malmi, E., Liao, Y., & Jung, A. (2017). Domain adaptation for resume classification using convolutional neural networks. Retrieved from https://arxiv.org/abs/1707.05576

2. Roopesh, N., & Babu, C. N. (Aug 27, 2021). Robotic process automation for the resume processing system. Paper presented at the 180-184. doi:10.1109/RTEICT52294.2021.9573595 Retrieved from https://ieeexplore.ieee.org/document/9573595

3. Ali, I., Mughal, N., Khan, Z. H., Ahmed, J., & Mujtaba, G. (2022). Resume classification system using natural language processing and machine learning techniques. Mehran University Research Journal of Engineering and Technology, 41(1), 65-79. doi:10.22581/muet1982.2201.07

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

5. https://github.com/bhashwitha/SDAI---PROJECT---1---Resume-Classification.git