Categorizing Resumes Using Data Mining Techniques

Prashanth Vangari
dept. Computer Science
University of North Texas
Denton, US
Prashanthvangari@my.unt.edu

Deekshith Bommireddy
dept. Computer Science
University of North Texas
Denton, US
Deekshithbommireddy@my.unt.edu

Mekala Maheshwar Reddy
dept. Computer Science
University of North Texas
Denton, US
MaheshwarReddymekala@my.unt.edu

Sanjay Thummanapally Bhargav
dept. Computer Science
University of North Texas
Denton, US
sanjaybhargavthummanapally@my.unt.edu

Koushik Kumar Reddy Devulapelly
dept. Computer Science
University of North Texas
Denton, US
koushikkumarreddydevulapelly@my.unt.edu

***Abstract*—The objective of this project is to improve the hiring process for organizations by tackling the difficulty of effectively categorizing resumes into particular job categories. Our goal is to automate the resume sorting process, which is currently labor-intensive and prone to human mistake, using a dataset of over 2400 resumes from a variety of career categories. We will clean and preprocess the resume data using data mining algorithms, using TF-IDF for text vectorization. In order to accurately allocate each resume to a specific job category—such as HR, IT, or education, among others—the process is handled as a classification problem(Support Vector Machines (SVMs), Naive Bayes, K-Nearest Neighbors (KNN),Logistic Regression,Random Forest Classifier). By using criteria like accuracy, f1-score, precision, and recall in our evaluation technique, we are able to compare our model's efficacy to baseline models.**

***Index Terms*—Resume, Data Mining, Categorization**

I. Introduction

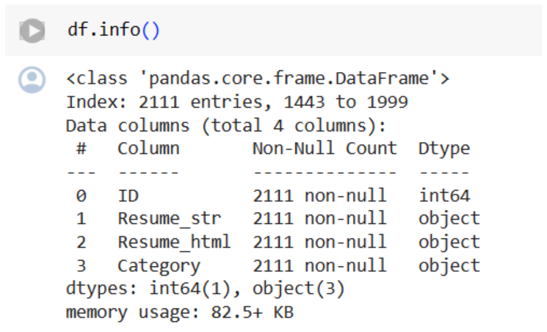
Manually sorting the resumes based on the job description is an hectic process for the organizations. There is a chance of manual errors in the process which may lead in failure in selection of a right candidate for the role. To solve this problem we are building a system which categorizes the resumes into their job roles. The current system identifies and categorizes the 24 job roles. In this project we have resumes of thousands of applicants which are accumulated in one csv file. We are using this CSV file to train our model which efficiently categorizes the resume into the job roles. First the text is cleaned using the regular expressions and vectorization is applied to convert them into the numbers and the dataset is split into train, test and validation and given to the machine learning models. Based on the accuracy we will decide the best model.

II. Background

This project falls under the Data mining domain because of the involvement of the text data This project problem has been taken from the kaggle.com. This dataset contains 2400 resumes which are scrapped from the livecareer website and stored it in the authors github. The dataset is readily available for the use and present in the compressed format.

III. Dataset

The dataset consists of 2400 resumes which are accumulated into a single file called Resume.csv. There are 2111 rows and 4 columns [1]. Here, Resume_str is the html file and



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2111 entries, 1443 to 1999
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   ID           2111 non-null   int64
 1   Resume_str   2111 non-null   object
 2   Resume_html  2111 non-null   object
 3   Category     2111 non-null   object
dtypes: int64(1), object(3)
memory usage: 82.5+ KB
```

Fig. 1. Data Information

scrapped from the livecareer website and added into one column. Category is the output column which is the category of the role.

IV. Experiment Methodology

A. *Experiment Setup*

The project is implemented in the google colab which is online ide to code the machine learning projects without concerning on installing the necessary libraries.

B. *Exploratory Data Analysis*

Exploratory Data Analysis is a important step because it is used to identify the patterns and outliers present in the data. By visualizing the data we can get deeper insights and understand the data better. In this section we performed the Bar chart representation and the Heat map analysis to understand the data better. The below image shows how the job categories are distributed [2]

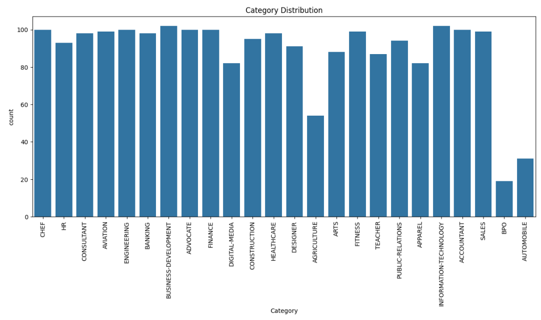


Fig. 2. Category Distribution

The below image shows the heat map analysis of the missing values in the dataset. [3]

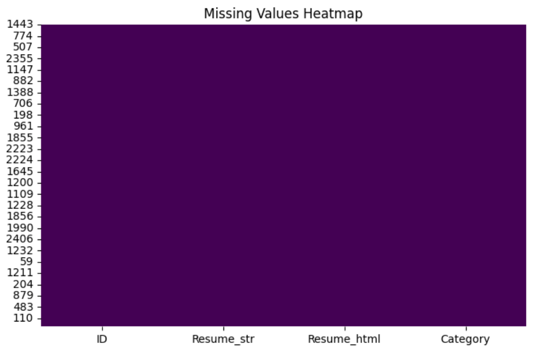


Fig. 3. Category Distribution

From the image [3] it is clear that we do not have any missing values.

C. Handling Missing Values and Text Cleaning

We do not have any missing values present in the dataset to handle. However, we have to clean the dataset because to apply the machine learning algorithms we have to convert the text data into the numbers so that we can train them on the machine learning model.

Before converting them to numbers we have a lot of unnecessary elements like comma, dots, redundant spaces, underscores present in our dataset.

After cleaning the data apply tokenization and stemming. In the tokenization process we convert the words into smaller words or sub words. Stemming is the process of converting the words to their root words.

D. Vectorization Using TF-IDF

In the Encoding section, we used Label Encoding to convert categorical values within the 'Category' column into numerical representations. This step streamlines the integration of categorical data into machine learning models.

During the text analysis phase, we employed vectorization methods like TF-IDF (Term Frequency-Inverse Document Frequency) and Word2Vec. These techniques transform textual data into numerical vectors, facilitating computational analysis and modeling on text-based datasets.

E. Splitting Dataset

We split the dataset into training dataset and testing dataset. The percentage of the training dataset is 85% and the rest 15% is the testing dataset. So we have total 1793 rows in the training dataset and we have 317 rows in the testing dataset.

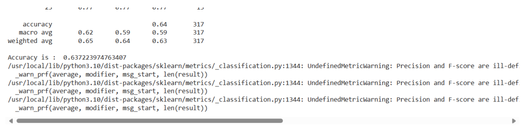
V. Model Training

A. Support Vector Machine

Support Vector Machine (SVM) is an imperative supervised learning algorithm applied for classification and regression tasks. It works by finding the hyperplane that separates better various classes in the feature space. SVM is quite productive in high-dimensional spaces, and it's especially used when one has to work with complicated datasets where more distinguishment in classes is relevant. It is even capable of working on non-linear data, making use of kernel functions, such as polynomial or radial basis function. In classification tasks, SVM targets to escalate the margin between classes, thus helping with the best generalization and robustness.

We are using the sklearn library and importing SVC svm_classifier = SVC() and passing the values to this model.

The accuracy of the model is shown in the below image [4]
.



```
          23       0.77      0.77      0.77        13

    accuracy                           0.64       317
   macro avg       0.62      0.59      0.59       317
weighted avg       0.65      0.64      0.63       317

Accuracy is :  0.637223974763407
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-def
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-def
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-def
  _warn_prf(average, modifier, msg_start, len(result))
```

Fig. 4. SVM Model Accuracy

B. Naive Bayes

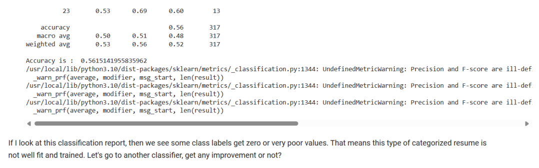
Naive Bayes is a probabilistic machine learning algorithm based on Bayes' theorem, which assumes independence among features. Despite its simplistic assumption, Naive Bayes often performs well in text classification and spam filtering tasks. It calculates the probability of each class given the input features and predicts the class with the highest probability. Naive Bayes is computationally efficient and requires minimal training data, making it suitable for large-scale datasets with many features.

We are importing MultinomialNB from sklearn library.

The accuracy of the model is shown in the below image [5]
.

C. K-Nearest Neighbors (KNN)

K-Nearest Neighbors is the simplest and most intuitive algorithm for classification as well as regression. It looks at counting the nearest k points in the feature space, and it classifies a new point according to the majority class of its



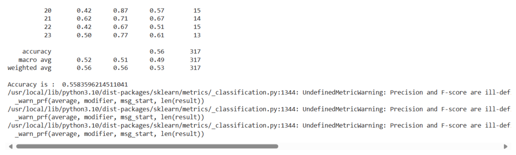
```
          23       0.53      0.69      0.60        13

    accuracy                           0.56       317
   macro avg       0.50      0.51      0.48       317
weighted avg       0.53      0.56      0.52       317

Accuracy is :  0.5615141955835962
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-def
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-def
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-def
  _warn_prf(average, modifier, msg_start, len(result))
```

If I look at this classification report, then we see some class labels get zero or very poor values. That means this type of categorized resume is not well fit and trained. Let's go to another classifier, get any improvement or not?

Fig. 5. Naive Bayes Model Accuracy

neighbors. KNN is a non-parametric, instance-based method-thus, it does not assume anything about the distribution of data. In fact, the value of k determines how smooth the decision boundary will be: k will be smaller and, thus, smoother for a smoother boundary.

The accuracy of the model is shown in the below image [6] .



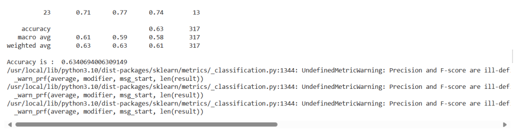
```
          20       0.42      0.87      0.57        15
          21       0.62      0.71      0.67        14
          22       0.42      0.67      0.51        15
          23       0.50      0.77      0.61        13

    accuracy                           0.56       317
   macro avg       0.52      0.51      0.49       317
weighted avg       0.56      0.56      0.53       317

Accuracy is :  0.5583596214511041
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-def
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-def
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-def
  _warn_prf(average, modifier, msg_start, len(result))
```

Fig. 6. KNN Model Accuracy

D. Logistic Regression

Logistic Regression is a popular statistical model used for binary classification tasks. Despite its name, logistic regression is a linear model that predicts the probability of a binary outcome based on one or more predictor variables. It uses the logistic function (also known as the sigmoid function) to transform the output of a linear regression into a probability between 0 and 1. Logistic regression is interpretable and can provide insights into the relationship between the input variables and the probability of the outcome.

The accuracy of the model is shown in the below image [7]



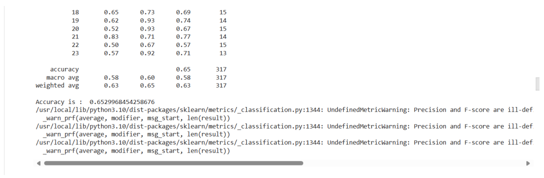
```
          23       0.71      0.77      0.74        13

    accuracy                           0.63       317
   macro avg       0.61      0.59      0.58       317
weighted avg       0.63      0.63      0.61       317

Accuracy is :  0.6340694006309149
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-def
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-def
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-def
  _warn_prf(average, modifier, msg_start, len(result))
```

Fig. 7. Logistic Regression Model Accuracy

E. Random Forest Classifier

. Random Forest Classifier is an ensemble learning method that combines multiple decision trees to make predictions. Each decision tree is trained on a random subset of the data and features, and the final prediction is made by aggregating the predictions of all individual trees. Random Forest is robust to overfitting and noisy data, and it can handle both classification and regression tasks. It is highly scalable and suitable for large datasets with high dimensionality. Random Forests also provide feature importance scores, allowing for interpretability of the model's predictions.

The accuracy of the model is shown in the below image [8] .

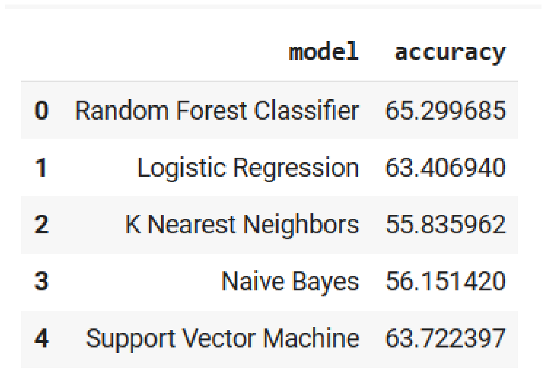


```
          18       0.65      0.73      0.69        15
          19       0.62      0.93      0.74        14
          20       0.52      0.93      0.67        15
          21       0.83      0.71      0.77        14
          22       0.50      0.67      0.57        15
          23       0.57      0.92      0.71        13

    accuracy                           0.65       317
   macro avg       0.58      0.60      0.58       317
weighted avg       0.63      0.65      0.63       317

Accuracy is :  0.6529968454258676
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-def
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-def
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-def
  _warn_prf(average, modifier, msg_start, len(result))
```

Fig. 8. Random Forest Model Accuracy

VI. Performance Evaluation

We evaluated the performance based on the metrics such as Accuracy, F1-score, Recall, Precision. We used classification_report, accuracy_score from sklearn.metrics. Accuracy measures the proportion of correctly classified instances out of the total instances in a dataset, providing an overall assessment of model performance. Precision quantifies the ratio of true positive predictions to the total predicted positives, emphasizing the correctness of positive predictions. [9]



	model	accuracy
0	Random Forest Classifier	65.299685
1	Logistic Regression	63.406940
2	K Nearest Neighbors	55.835962
3	Naive Bayes	56.151420
4	Support Vector Machine	63.722397

Fig. 9. All Models Accuracy

VII. Results

We can clearly see that Random forest outperformed all the models with the accuracy of 65% where as the other models only ranged from the 55%-63% only [10] .

From the figure we can see that all the models performed almost equally but Random forest with slightly higher than the rest of the models.

The same is also represented in the below as a table format:

TABLE I
Model Accuracy

Model	Accuracy (%)
Random Forest Classifier	64.67
Logistic Regression	63.41
K Nearest Neighbors	55.84
Naive Bayes	56.15
Support Vector Machine	63.72

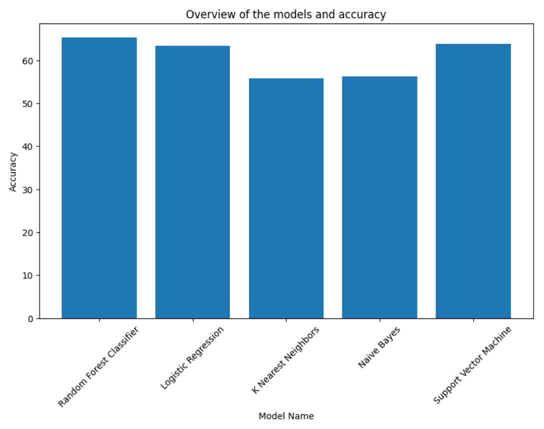


Fig. 10. All Models Accuracy

VIII. Related Work

We found this dataset in the kaggle which has been provided with both the dataset in one single file and also in the individual folders. This dataset contains 2400 resumes. This data has been scrapped from the webiste www.livecareer.com website and stored in the authors github repository.

IX. Limitations

The dataset's 24 class labels present challenges due to data scarcity and imbalance, resulting in diminished model accuracy, particularly for underrepresented categories. Insufficient data within certain classes leads to zero precision, recall, and F1-score values, highlighting the need for strategies to address imbalance and enhance model performance.

X. Future work

Efforts should focus on implementing techniques like oversampling, undersampling, or synthetic data generation to balance the dataset and improve model accuracy across all categories. Additionally, exploring advanced machine learning algorithms and ensemble methods capable of handling imbalanced data effectively could further enhance classification performance. Conducting in-depth analysis to identify factors contributing to data scarcity and devising strategies to collect or generate more representative samples will be crucial for ensuring robust model training and evaluation.

XI. Conclusion

In summary, we only got 65% of the results which may not be deployed in the real world applications. A significant challenge emerged when analyzing the dataset, revealing 24 distinct class labels. This diversity posed difficulties, particularly when some class labels had insufficient data during model training, leading to inaccuracies in classification. Consequently, certain categories exhibited poor model fit, evident in precision, recall, and f1-score metrics where some classes registered zero values. This outcome underscores the importance of data adequacy in achieving accurate model performance. Despite robust comparative analysis across various models, limitations in class label data quantity significantly impacted overall model accuracy.

References

[1] https://www.kaggle.com/datasets/snehaanbhawal/resume-dataset/data