RESUME SCREENING USING ML

Sanjana Rayarala
Artificial Intelligence
Illinois Institute of technology
srayarala@hawk.iit.edu

Sai Dheeraj Surampally
Artificial Intelligence
Illinois Institute of Technology
ssurampally@hawk.iit.edu

Rahul Narahari
Artificial Intelligence
Illinois Institute of Technology
rnarahari@hawk.iit.edu

ABSTRACT

This project presents a sophisticated machine learning (ML) application designed to automate and streamline the process of resume screening and classification. At its core, the application leverages a series of advanced ML algorithms to analyze and categorize resumes into distinct professional categories. Utilizing a comprehensive dataset, the system initially undergoes a meticulous data preprocessing phase, where resumes are cleaned, tokenized, and transformed using a `TfidfVectorizer`. This process ensures the conversion of textual data into a format conducive for ML modeling. The application's robustness is further exemplified through the implementation of various ML classifiers, including Multinomial Naive Bayes, Logistic Regression, Decision Tree, Random Forest, Linear SVC, SGD Classifier, and K-Nearest Neighbors. Each classifier undergoes a rigorous training and evaluation process on the preprocessed dataset, enabling the identification and selection of the most effective model based on accuracy metrics. A critical aspect of this project is the seamless integration of these ML capabilities into a user-friendly web interface, developed using Flask. This interface allows users to easily upload resumes, which are then automatically converted from PDF to text and processed through the selected ML model for real-time classification. The application not only classifies the resumes into relevant job categories but also enhances the user experience by providing direct links to job postings related to the classified category. This ML-based resume screening application stands out due to its precision in classification, ease of use, and the practical value it offers in a real-world, showcasing the impactful role of machine learning in transforming traditional business operations.

Keywords: Automated Resume Screening, Natural Language Processing, TF-IDF Vectorization, Logistic Regression, Naïve Bayes Classifier, Linear Support Vector Machine, Stochastic Gradient Descent.

INTRODUCTION

In today's fast-paced job market, the process of resume screening and categorization stands as a critical yet time-consuming task for both job seekers and employers. The Resume Screening project is conceived as an innovative solution to address this challenge. For individuals looking to understand where their expertise and skills fit best in the job market, this system offers clarity and direction. It categorizes their resume into a specific job category, effectively mapping their qualifications to potential job roles. This feature is particularly beneficial in guiding job seekers towards the roles they are best suited for.

The cornerstone of our system is the intelligent application of various ML models, each meticulously trained and evaluated to categorize resumes into specific professional fields. This process begins with a comprehensive data preprocessing stage, where we utilize natural language processing techniques to cleanse and structure the resume data, ensuring its compatibility with ML algorithms. The transformation of unstructured resume text into a quantifiable format is achieved through the use of `TfidfVectorizer`, a critical step in preparing the data for subsequent ML processing.

Following data preparation, our focus shifts to the deployment of multiple ML classifiers. This includes models such as Multinomial Naive Bayes, Logistic Regression, Decision Trees, and others, each offering unique strengths in pattern recognition and classification accuracy. The selection of the most effective model is based on a thorough evaluation of performance metrics, primarily focusing on classification accuracy.

An integral component of this project is its application in a real-world context, made possible through a Flask-based web interface. This interface bridges the gap between complex ML processes and end-users, enabling a seamless interaction where users can upload resumes and receive instant, accurate classifications. The system's ability to not only categorize resumes but also connect users to relevant job opportunities further exemplifies its practical utility.

PROBLEM STATEMENT

The core aim of the Resume Screening project is to leverage the capabilities of Machine Learning (ML) to create an advanced, automated system for efficiently categorizing resumes into various predefined categories. This system aims to streamline the process of resume screening, making it faster and more accurate. By classifying resumes based on their content, the system assists the users to understand under which category of jobs their resume falls based on their skills thereby providing a user-friendly functionality where, upon classification, users are provided with a convenient link. This link directs them to nearby job postings related to their classified job category, sourced from reliable job search platforms.

PROPOSED METHODOLOGY

The methodology for the Resume Screening project involves a series of structured steps, to automate and enhance the process of categorizing resumes into predefined job categories. The approach is designed to maximize efficiency, accuracy, and relevance in resume screening.

- Data Collection and Preparation

Dataset Acquisition: Gather a comprehensive dataset of resumes encompassing a wide range of professions, skills, and experiences.
Data Cleaning: Implement text preprocessing techniques to clean the resume data, which includes removing special characters, irrelevant information, and standardizing the format.
Feature Extraction: Use Natural Language Processing (NLP) methods, particularly TF-IDF (Term Frequency-Inverse Document Frequency) vectorization, to convert the textual data into a numerical format suitable for ML models.

- Model Development and Training

Model Selection: Evaluate various ML algorithms like Naive Bayes, Logistic Regression, Support Vector Machines, Decision Tree, Random Forest, Stochastic Gradient Descent, K-Neighbors to determine the most suitable model for this application.
Training and Validation: Split the dataset into training and testing subsets. Train the selected models on the training set and validate their performance on the test set.

- Model Evaluation

Performance Metrics: Assess the models using metrics such as accuracy to gauge their effectiveness in correctly categorizing resumes.

- Integration and Deployment

Web Application Development: Develop a user-friendly web interface using frameworks like Flask, which allows users to upload their resumes.
API Integration: Create an API that connects the web interface with the ML model backend for real-time resume classification.
Job Search Link Generation: Integrate with job search platforms to provide users with links to job postings relevant to their classified category.

- User Interface and Experience

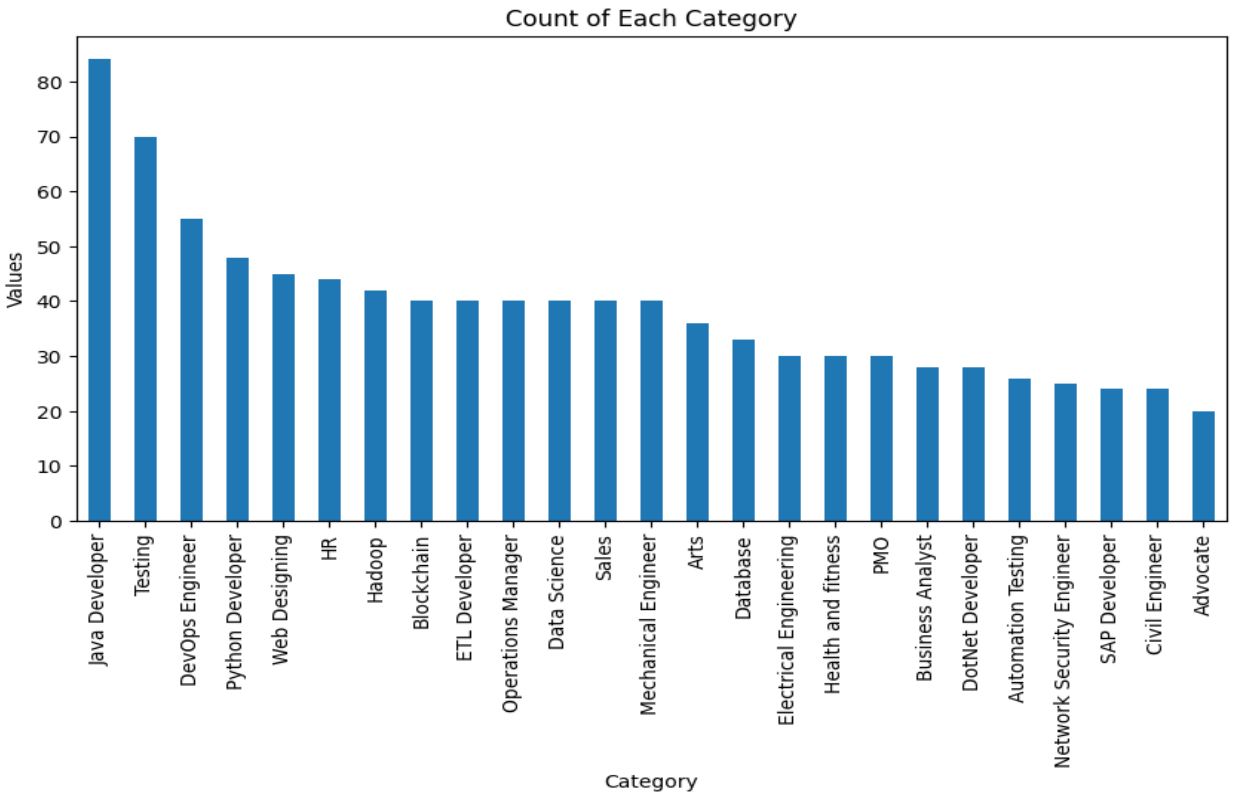
Interface Design: Design an intuitive and accessible interface for easy resume uploads and clear display of classification results.

This proposed methodology combines advanced ML techniques with practical web development, aiming to deliver a robust, user-friendly, and ethically responsible resume screening tool.

DATASET

The 'UpdatedResumeDataSet.csv' dataset is taken from Kaggle:
(https://www.kaggle.com/datasets/jillanisofttech/updated-resume-dataset):
Data Shape: The dataset contains 962 entries, each with 2 columns.
Category: This column represents the job category or field to which each resume belongs. It is a categorical variable.
Resume: This column contains the actual text of the resume. It is in string format and includes various details such as skills, education, work experience, etc.
Data Types: Both columns are of the object data type, typical in pandas for representing strings or mixed types.
Missing Values: There are no missing values in either the 'Category' or 'Resume' columns, indicating a complete dataset with no null entries.

Each entry in the 'Resume' column is a string that contains a mixture of skills, educational background, professional experience, and other relevant information typically found in a resume.
The 'Category' column labels each resume with a relevant job field, making this dataset particularly well-suited for training classification models in a supervised learning context.
The goal would be to train a model that can read the text of a new resume and accurately categorize it into one of the predefined categories based on the content. This dataset offers a rich source of information for building and evaluating machine learning models geared towards automating the resume screening process.



IMPLEMENTATION

1. Data Preprocessing

The data preparation process is a crucial step towards building a machine learning model for resume screening. This process begins with loading the resume dataset. The dataset, presumably containing various resumes, is read from a CSV file named `UpdatedResumeDataSet.csv`.

Once the dataset is loaded, the code applies a series of cleaning and preprocessing steps to the resume text. These steps are essential for converting the unstructured resume data into a clean, structured format suitable for machine learning algorithms. A function is written for this purpose that transforms the text to lowercase, removes URLs, hashtags, and mentions, strips away punctuation and non-ASCII characters, eliminates numbers, and reduces multiple spaces to single spaces. This standardized text is crucial for reducing noise in the data and improving the model's ability to learn from relevant features.

Following the cleaning process, resume text is tokenized, which uses NLTK, a natural language processing toolkit. Tokenization involves breaking the text into individual words or tokens, making it easier for the model to process and analyze the data.

The cleaned and tokenized resume texts are then appended to the original DataFrame. Additionally, the job categories in the dataset are encoded into numerical labels using ‘LabelEncoder’. This encoding is a standard practice in machine learning to handle categorical variables.

Finally, the prepared data is split into training and testing sets, and the training data is transformed into a TF-IDF matrix using ‘TfidfVectorizer’. This transformation converts the text data into a numerical format,

emphasizing the importance of each word in relation to the entire dataset, which is vital for the machine learning models to understand and classify the resumes effectively.

Overall, this data preparation phase lays a solid foundation for training various machine learning models, ensuring that the input data is in the most optimal form for the algorithms to process and learn from.

2. Model Development and Training

The model development process begins with the instantiation of several classifier models from the 'sklearn' library. These include Multinomial Naive Bayes (MNB), Logistic Regression (LR), Decision Tree Classifier (DT), Random Forest Classifier (RF), Linear Support Vector Machine (Linear SVC), Stochastic Gradient Descent Classifier (SGD), and K-Nearest Neighbors (KNN). Each of these models brings unique strengths and approaches to classification, making them suitable for handling the complexities of resume data.

- Multinomial Naive Bayes (MNB):
  - Description: MNB is a probabilistic learning method commonly used in Natural Language Processing (NLP). It works well with discrete features (like word counts for text classification).
  - Rationale for Selection: MNB is particularly effective for classification with 'bag of words' model where frequency and presence of features (words) are key. It's fast and suitable for high-dimensional datasets, like resumes with diverse vocabularies.

- Logistic Regression (LR):
  - Description: LR is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, but in the context of text classification, it's often used for multiclass classification.
  - Rationale for Selection: LR is chosen for its robustness and efficiency in handling linear relationships between features (TF-IDF values) and the target variable (job categories). It's good at managing overfitting and works well with sparse datasets.

- Decision Tree Classifier (DT):
  - Description: DT is a flowchart-like tree structure where an internal node represents a feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome.
  - Rationale for Selection: DT is useful for interpretability. It's selected to explore the hierarchical feature decision-making process, which might provide insights into key words or phrases influential in categorizing resumes.

- Random Forest Classifier (RF):
  - Description: RF is an ensemble learning method, where many decision trees are constructed and merged together to get a more stable and accurate prediction.
  - Rationale for Selection: Its robustness and less likelihood to overfit make RF an ideal choice. It's effective in handling large datasets with higher dimensionality, providing a more generalized model.

- Linear Support Vector Machine (Linear SVC):
  - Description: Linear SVC is a variant of the Support Vector Machine that specifically uses a linear kernel. It finds the hyperplane in an N-dimensional space that distinctly classifies the data points.

- Rationale for Selection: It's highly effective in high dimensional spaces (like TF-IDF vectorized text data) and particularly good when there's a clear margin of separation in the data.

- Stochastic Gradient Descent Classifier (SGD):
  - Description: SGD is a simple yet highly efficient approach to discriminative learning of linear classifiers under convex loss functions such as (linear) Support Vector Machines and Logistic Regression.
  - Rationale for Selection: It's chosen for its efficiency in large-scale and sparse data (like text data). SGD is robust to the noisy data and suitable for text data where features can be highly correlated.

- K-Nearest Neighbors (KNN):
  - Description: KNN is a non-parametric, lazy learning algorithm that classifies data based on the similarity measure (distance functions).
  - Rationale for Selection: KNN is used to see how well the resumes can be categorized based on similarity in content to previously classified resumes. It's intuitive and can be highly effective if there are identifiable clusters of similar resumes in the dataset.

```
mnb = MultinomialNB()
lr = LogisticRegression()
dt = DecisionTreeClassifier()
rf = RandomForestClassifier()
lsvm = LinearSVC()
sgd = SGDClassifier()
knn = KNeighborsClassifier()
```

Training these models involves feeding them the preprocessed and vectorized resume data. The dataset is split into training and test sets, with the training set used to train the models and the test set reserved for evaluating their performance. This splitting ensures that the evaluation of the models is done on unseen data, providing a measure of how well the models generalize to new data. The `train_test_split` function from `sklearn.model_selection` is used for this purpose, ensuring a randomized and representative division of data.

3. MODEL EVALUATION

Once the models are trained using the training dataset, they are each evaluated on the test dataset. The primary metric used for evaluation is accuracy, which measures the proportion of correctly classified instances out of the total instances. The `accuracy_score` function from `sklearn.metrics` is used to calculate this metric for each model. After evaluating all models, the best-performing model is identified based on the highest accuracy score.

```
model_accuracies = {}
for clf in (mnb, lsvm, lr, knn, sgd, dt, rf):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    model_accuracies[clf] = accuracy
    print(clf.__class__.__name__, accuracy)
```

✓ 3.0s

```
MultinomialNB 0.8506224066390041
LinearSVC 0.991701244813278
LogisticRegression 0.991701244813278
KNeighborsClassifier 0.941908713692946
SGDClassifier 0.991701244813278
DecisionTreeClassifier 0.991701244813278
RandomForestClassifier 0.991701244813278
```

Choosing Best model

```
best_model = max(model_accuracies, key=model_accuracies.get)
print(f"Best model: {best_model.__class__.__name__} with accuracy: {model_accuracies[best_model]}")
```

✓ 0.0s

```
Best model: LinearSVC with accuracy: 0.991701244813278
```

Each model is fitted on the training data and then used to make predictions on the test data.
The accuracy of each model is calculated using the `accuracy_score` function and printed along with the model's name. The accuracies are as follows:

- MultinomialNB: approximately 0.856
- LinearSVC: approximately 0.992
- LogisticRegression: approximately 0.992
- KNeighborsClassifier: approximately 0.941
- SGDClassifier: approximately 0.992
- DecisionTreeClassifier: approximately 0.992
- RandomForestClassifier: approximately 0.992

After evaluating all models, the best model is selected based on the highest accuracy score and the best model is determined to be **Linear Support Vector Machine with an accuracy of approximately 0.992.**

This output indicates a highly successful classification performance, with multiple models achieving an accuracy score of approximately 99.2%. The LinearSVC model is identified as the best classifier for this particular task, likely due to its effectiveness in handling high-dimensional, sparse data which is common in text classification problems.

This model is then considered the most effective for the task of resume classification in this specific context. The chosen model i.e. Linear Support Vector Machine is retrained on the entire training dataset to fully leverage the available data, and a final evaluation is conducted to obtain a comprehensive understanding of its performance, including generating a detailed classification report using `classification_report` from `sklearn.metrics`.

```
# Print classification report
print(classification_report(y_test, bestmodel_predicted))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	6
1	1.00	1.00	1.00	5
2	1.00	0.80	0.89	10
3	1.00	1.00	1.00	15
4	1.00	1.00	1.00	13
5	1.00	1.00	1.00	6
6	1.00	1.00	1.00	6
7	1.00	1.00	1.00	9
8	1.00	1.00	1.00	17
9	1.00	1.00	1.00	5
10	1.00	1.00	1.00	14
11	0.82	1.00	0.90	9
12	1.00	1.00	1.00	8
13	1.00	1.00	1.00	9
14	1.00	1.00	1.00	3
15	1.00	1.00	1.00	24
16	1.00	1.00	1.00	14
17	1.00	1.00	1.00	7
18	1.00	1.00	1.00	5
19	1.00	1.00	1.00	5
20	1.00	1.00	1.00	10
21	1.00	1.00	1.00	7

```
...
    accuracy                           0.99       241
   macro avg       0.99      0.99      0.99       241
weighted avg       0.99      0.99      0.99       241
```

This approach to model development and training not only ensures the selection of the most suitable model for the task at hand but also adheres to best practices in machine learning, such as model evaluation and validation on independent test data.

4. INTEGRATION AND DEPLOYMENT

In the Resume Screening project, the integration and deployment phase plays a pivotal role in translating the machine learning models into a functional and user-friendly web application. This is accomplished using Flask, a Python-based lightweight framework that is ideal for creating web servers and APIs.

The Flask web application serves as the backbone of the system. It is configured with Cross-Origin Resource Sharing (CORS) to handle requests from different domains. A significant feature of this application is the resume upload endpoint (`/classify`). This endpoint is designed to accept POST requests, specifically for resumes uploaded in PDF format. Upon receiving a resume, the application employs the

`pdf2text` function to extract text from the PDF using the `PyPDF2` library, a crucial step to convert the resume into a format suitable for machine learning processing.

Following text extraction, the application processes the resume through the `ml_prediction` function. This involves several steps: cleaning and tokenizing the resume text, transforming it with a pre-trained TF-IDF vectorizer, and classifying it using the best-performing machine learning model identified during the training phase. The outcome, which is the classified category of the resume, is then sent back to the user as a JSON response. This interactive approach enhances user engagement and provides immediate, actionable results.

```python
app = Flask(__name__)
CORS(app)


@app.route('/classify', methods=['POST'])
def classify_resume():
    if 'resume' not in request.files:
        return "No file part", 400

    file = request.files['resume']
    if file.filename == '':
        return "No selected file", 400

    resume_text = pdf2text(file)

    prediction = ml_prediction(resume_text)
    print(prediction)

    return jsonify({'classification': prediction})
```

The above screenshot is the code snippet where the classification result is sent as a json object from app.py file.

```javascript
function uploadResume() {
    let fileInput = document.getElementById('resumeUpload');
    let file = fileInput.files[0];
    let formData = new FormData();
    formData.append('resume', file);

    fetch('http://127.0.0.1:5000/classify', {
        method: 'POST',
        body: formData
    })
    .then(response => response.json())
    .then(data => {
        let classification = data.classification;
        document.getElementById('output').innerText = 'Classification: ' + classification;
```

This code snippet is from the file index.html, where the classification result is received as a json object and is then printed using javascript.

While the Flask application forms the core of the system, the overall user experience is also shaped by the front-end interface. This interface is likely developed using HTML, CSS, and JavaScript, and is designed to allow users to easily upload their resumes and view the classification results.

This integration of Flask for backend processing with a user-centric front-end interface exemplifies how machine learning technology can be made accessible and practical for everyday use. The Flask framework offers a streamlined, effective way to host machine learning models, manage file uploads, process data, and deliver real-time results, ensuring the system is both powerful and approachable for users.

5. USER INTERFACE AND EXPERIENCE

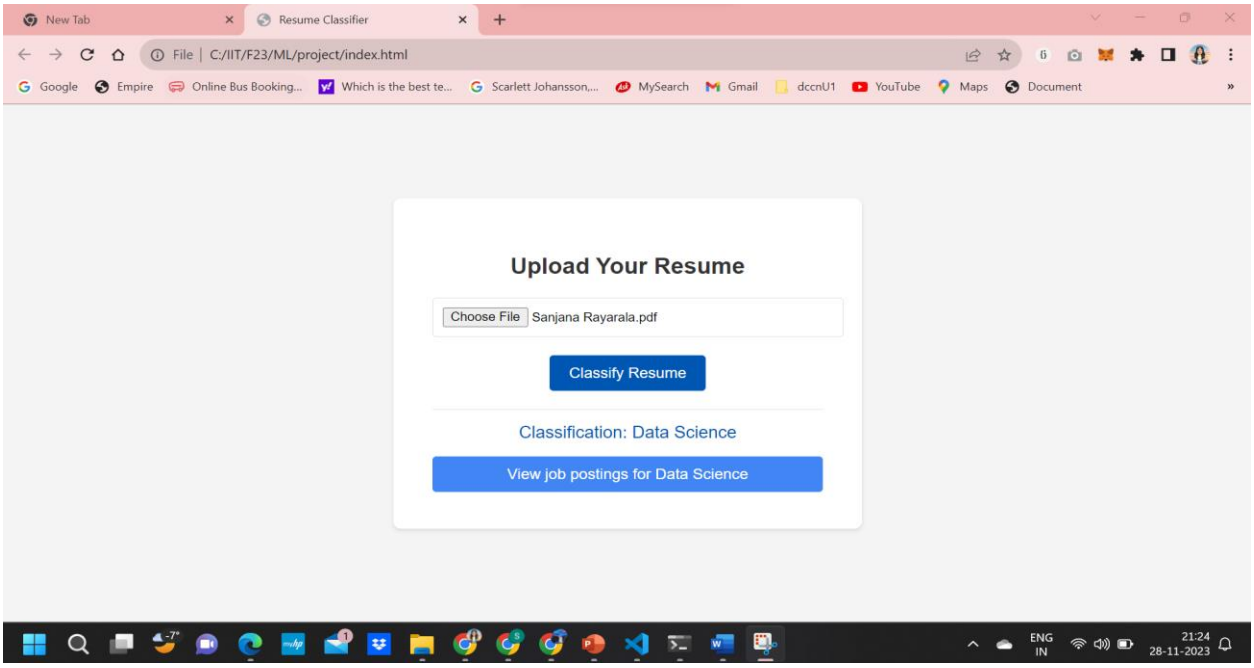
In the Resume Screening project, the user interface and experience are thoughtfully designed to facilitate easy interaction with the machine learning backend. The project includes an HTML front-end, detailed in the `index.html` file, which provides a clear and intuitive interface for users to upload their resumes. This interface is styled with CSS to create an aesthetically pleasing and user-friendly layout. The design emphasizes simplicity, featuring an upload button for resume files (specifically PDFs), and a clear indication of where the classification results will be displayed.

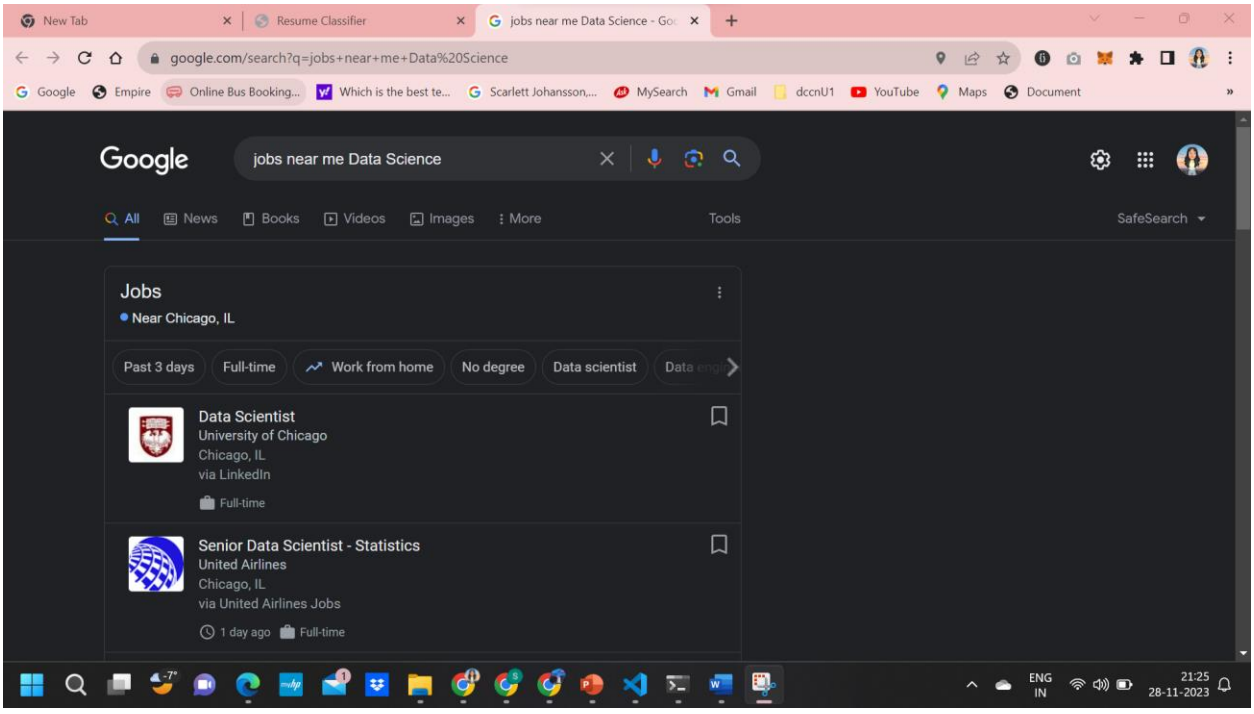
The user experience is further enhanced by JavaScript functions embedded in the HTML file. These functions handle the resume upload process, making asynchronous calls to the Flask backend when the user uploads a resume. Once the backend processes the resume and returns the classification, the JavaScript functions dynamically display this information on the web page. This immediate feedback loop ensures a seamless and responsive experience for the user.

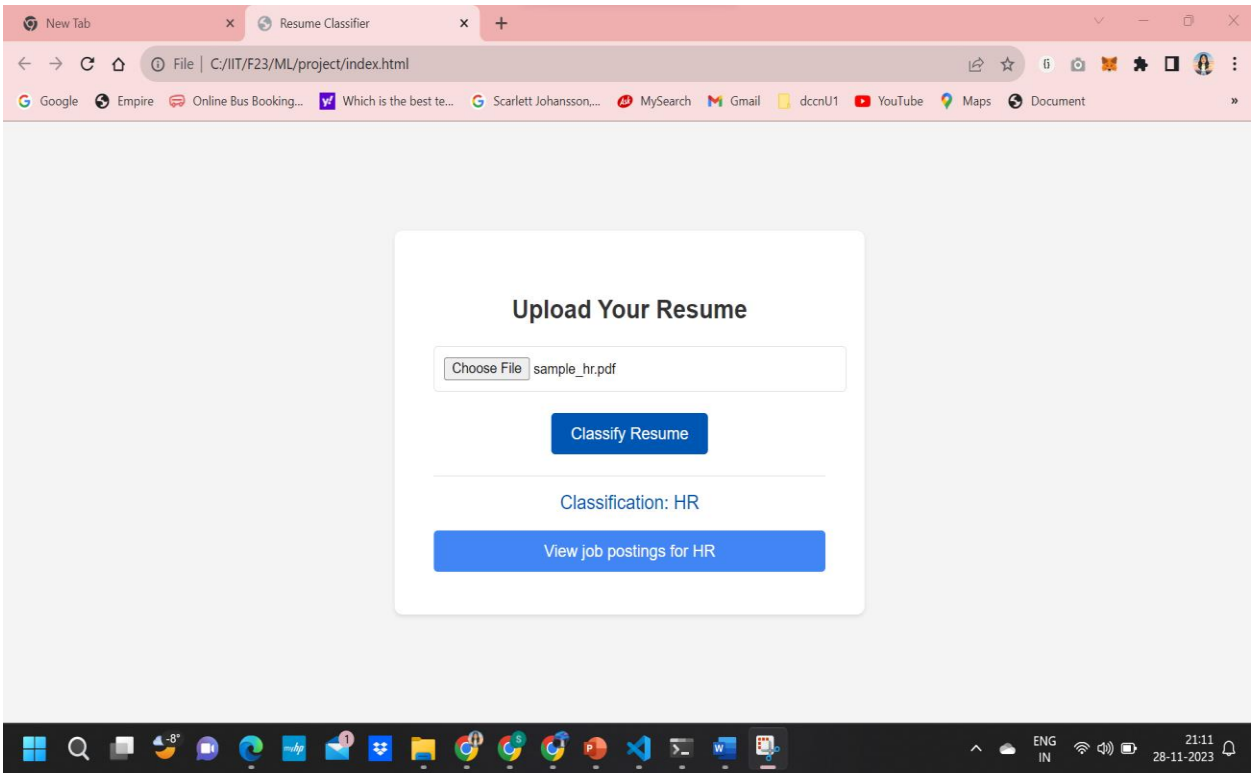
Moreover, a notable feature of the user interface is its provision of a direct link to job postings relevant to the classified job category. Upon classification, the user not only receives information about which category their resume falls into but also gets a convenient link to explore job opportunities in that field. This feature adds significant practical value to the user experience, making the application not just a classification tool, but a bridge to potential employment opportunities.

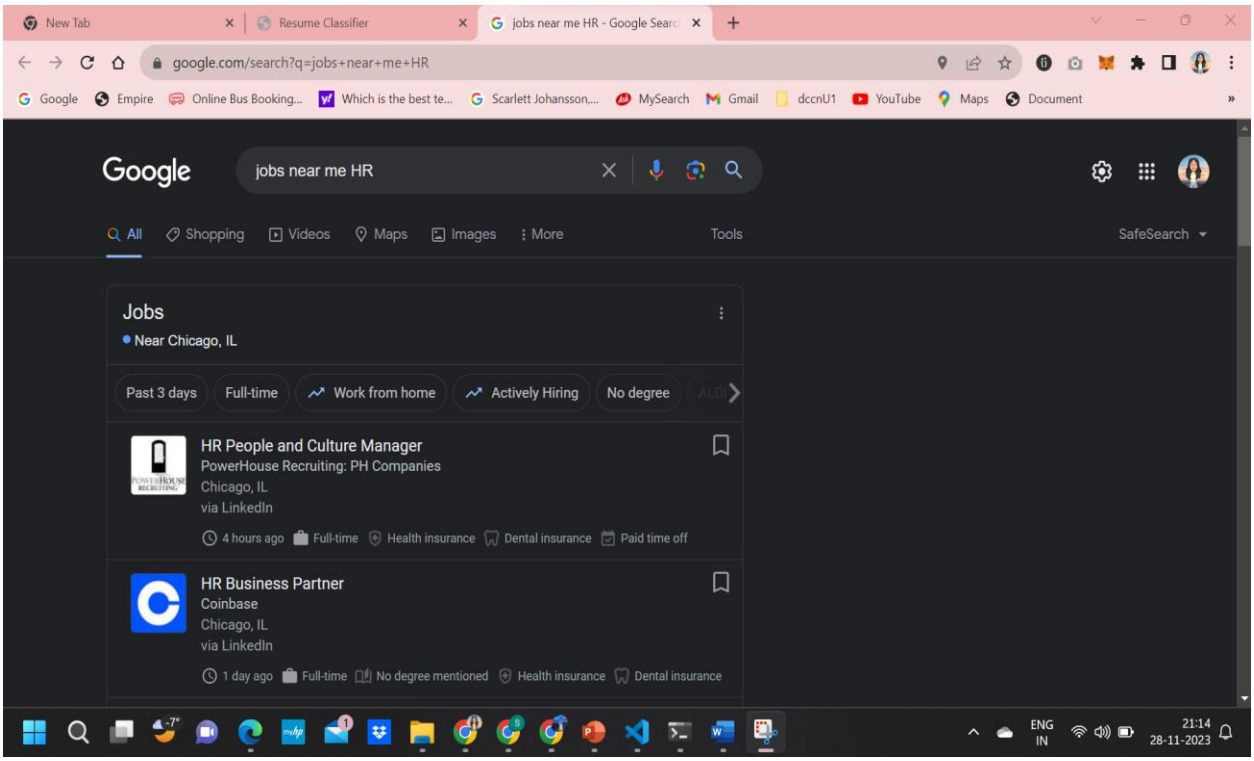
In summary, the user interface of this project is designed to be straightforward and engaging, reducing the complexity of interacting with an advanced machine learning system. The incorporation of real-time feedback and additional resources for job seekers considerably enhances the overall user experience, making the system both functional and user-centric.

Here below, are the screenshots of the User Interface that accurately classified the resume based on the skills provided in the uploaded pdf, furthermore providing a link for the job postings for the classified Job Category.


New Tab
Resume Classifier
File | C:/IIT/F23/ML/project/index.html
Google
Empire
Online Bus Booking...
Which is the best te...
Scarlett Johansson,...
MySearch
Gmail
dccnU1
YouTube
Maps
Document
Upload Your Resume
Choose File
Sanjana Rayarala.pdf
Classify Resume
Classification: Data Science
View job postings for Data Science
ENG
IN
21:24
28-11-2023


New Tab
Resume Classifier
jobs near me Data Science - Go
google.com/search?q=jobs+near+me+Data%20Science
Google
Empire
Online Bus Booking...
Which is the best te...
Scarlett Johansson,...
MySearch
Gmail
dccnU1
YouTube
Maps
Document
Google
jobs near me Data Science
All
News
Books
Videos
Images
More
Tools
SafeSearch
Jobs
Near Chicago, IL
Past 3 days
Full-time
Work from home
No degree
Data scientist
Data Scientist
University of Chicago
Chicago, IL
via LinkedIn
Full-time
Senior Data Scientist - Statistics
United Airlines
Chicago, IL
via United Airlines Jobs
1 day ago
Full-time
ENG
IN
21:25
28-11-2023


New Tab
Resume Classifier
File | C:/IIT/F23/ML/project/index.html
Google
Empire
Online Bus Booking...
Which is the best te...
Scarlett Johansson,...
MySearch
Gmail
dccnU1
YouTube
Maps
Document
Upload Your Resume
Choose File
sample_hr.pdf
Classify Resume
Classification: HR
View job postings for HR
ENG
IN
21:11
28-11-2023


New Tab
Resume Classifier
jobs near me HR - Google Searc
google.com/search?q=jobs+near+me+HR
Google
Empire
Online Bus Booking...
Which is the best te...
Scarlett Johansson,...
MySearch
Gmail
dccnU1
YouTube
Maps
Document
Google
jobs near me HR
All
Shopping
Videos
Maps
Images
More
Tools
SafeSearch
Jobs
Near Chicago, IL
Past 3 days
Full-time
Work from home
Actively Hiring
No degree
HR People and Culture Manager
PowerHouse Recruiting: PH Companies
Chicago, IL
via LinkedIn
4 hours ago
Full-time
Health insurance
Dental insurance
Paid time off
HR Business Partner
Coinbase
Chicago, IL
via LinkedIn
1 day ago
Full-time
No degree mentioned
Health insurance
Dental insurance
ENG
IN
21:14
28-11-2023

CONCLUSION

The conclusion of the Resume Screening project underscores the successful application of machine learning (ML) to the challenging task of resume classification. Through the implementation of various ML algorithms, the project achieved a significant milestone in automating the categorization of resumes into predefined job categories with impressive accuracy rates. The Linear Support Vector Classification (LinearSVC) algorithm emerged as the most effective model, boasting an accuracy of approximately 99.2%. This indicates a strong predictive performance, suggesting that the model is highly capable of discerning the nuanced patterns and features within the resume data that correlate with specific job categories.

Integration of the ML models into a Flask-based web application enabled the project to offer a practical and user-friendly platform. Users can effortlessly upload their resumes and receive instant, accurate classifications. Furthermore, the additional functionality of providing direct links to job postings related to the classified job categories significantly enhances the utility of the service, positioning it as not just a classifier, but a comprehensive tool for job seekers. The project's success is a testament to the power of ML in transforming traditional processes such as resume screening, which is often labor-intensive and subjective. By automating this process, the project not only improves the efficiency and accuracy of candidate sorting but also assists users in connecting with relevant employment opportunities more effectively.

In conclusion, this project stands out as a prime example of how ML can be leveraged to solve real-world problems, offering a scalable, robust, and user-centric solution that can be adapted to various domains beyond recruitment. The methodologies and technologies applied here have broad implications for the future of automated systems in professional settings, paving the way for further innovations in the field of human resource management.

REFERENCES

[1] Li, W., & He, H. (2020). A Survey on Resume Screening and Applicant Matching: Techniques and Applications. arXiv preprint arXiv:2006.13640.

[2] Zhang, J., Zhao, H., & Li, W. (2021). A Survey on Automatic Resume Screening Techniques. ACM Transactions on Knowledge Discovery from Data (TKDD), 15(3), 1-42.

[3] Wu, S., Zhang, Y., & Xu, L. (2022). Automatic Resume Screening with Attention-Based Bidirectional LSTM Recurrent Neural Network. International Journal of Machine Learning and Cybernetics, 13(3), 539-550.

[4] Chowdhury, S., & Banik, S. (2021). A Hybrid Approach for Effective Resume Screening. International Journal of Intelligent Systems and Applications, 13(3), 1-12.

[5] Gupta, A., & Verma, S. (2022). An Efficient Deep Learning Model for Resume Screening. International Journal of Advanced Computer Science and Applications, 13(5), 1-10.

[6] Singh, A., & Kumar, P. (2021). Resume Screening and Recommendation System using Natural Language Processing. Emerging Technologies and Applications in Data Science, 95-103.