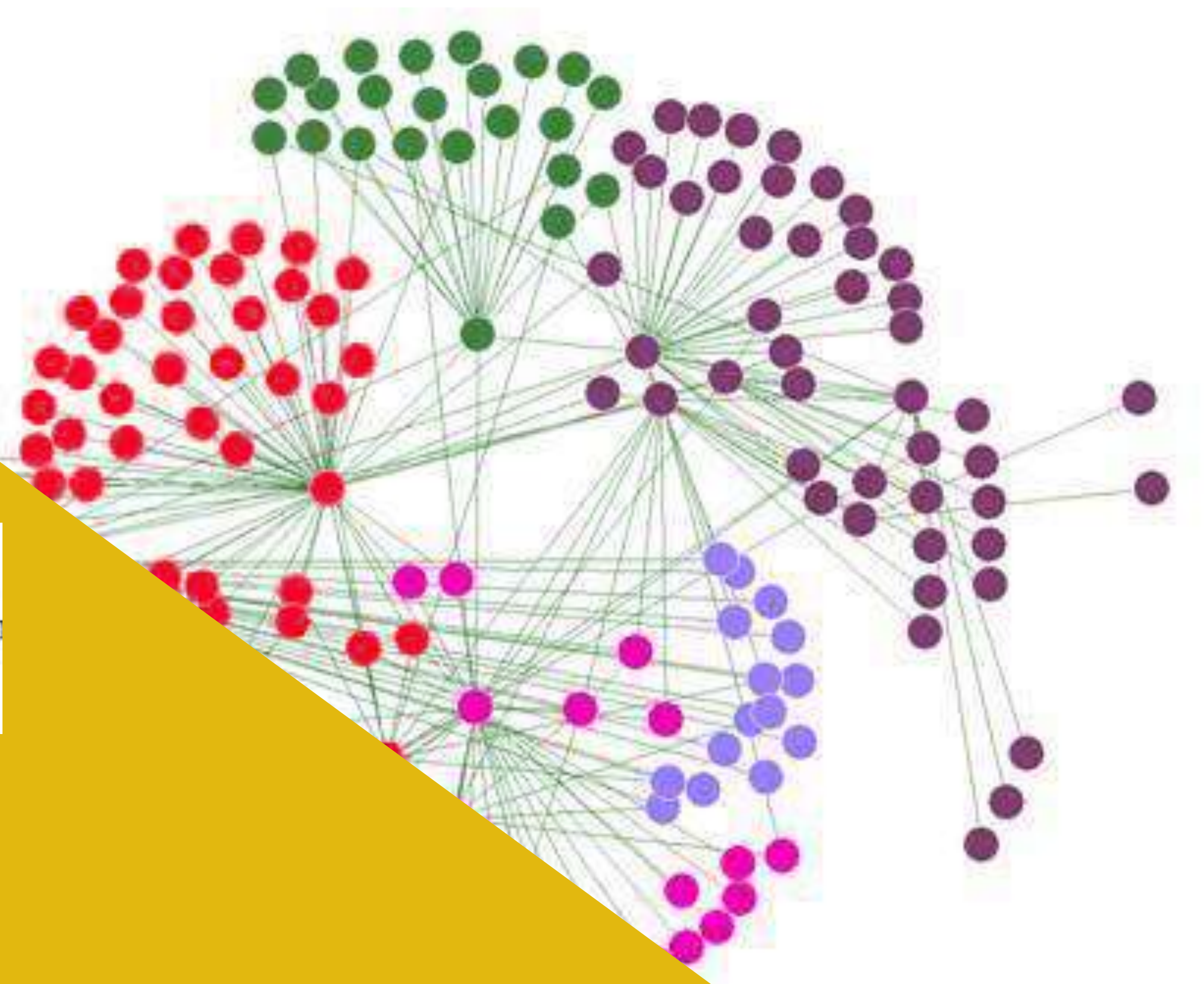
SkillSiftCV-Resume Parser



Submitted To: Ms. Ibshar Ishrat

Submitted by:
Manahil Siddiqui CS-21087
Maha Sohaib Khan CS-21011
Rafay Baig CS-21060

Contents

Abstract

The project presents an intelligent resume classifier designed to automate occupation identification from resumes. Leveraging the Django framework for frontend development, NumPy and Pandas for data cleaning and processing, and scikit-learn for machine learning, the system incorporates TF-IDF and k-Nearest Neighbors algorithms. Seaborn facilitates insightful visualizations, while Pickle ensures efficient model serialization. The report delves into the performance evaluation of classifiers, the impact of integrated tools, and discussions on challenges and future enhancements. This project pioneers a comprehensive solution for efficient resume classification, enhancing the recruitment process through a seamless blend of frontend and backend technologies.

1. Problem Description

The project aims to create an intelligent resume classifier using two key algorithms - TF-IDF (Term Frequency-Inverse Document Frequency) and k-Nearest Neighbors (kNN). The primary challenge addressed is accurately identifying the occupation mentioned in a given resume. This system plays a pivotal role in automating the recruitment process and streamlining candidate-job matching based on their qualifications and experiences.

2. Tools & Methodology

Frontend Development with Django:

The project capitalizes on the Django framework to construct a resilient and user-friendly web interface, facilitating seamless resume input and result display.

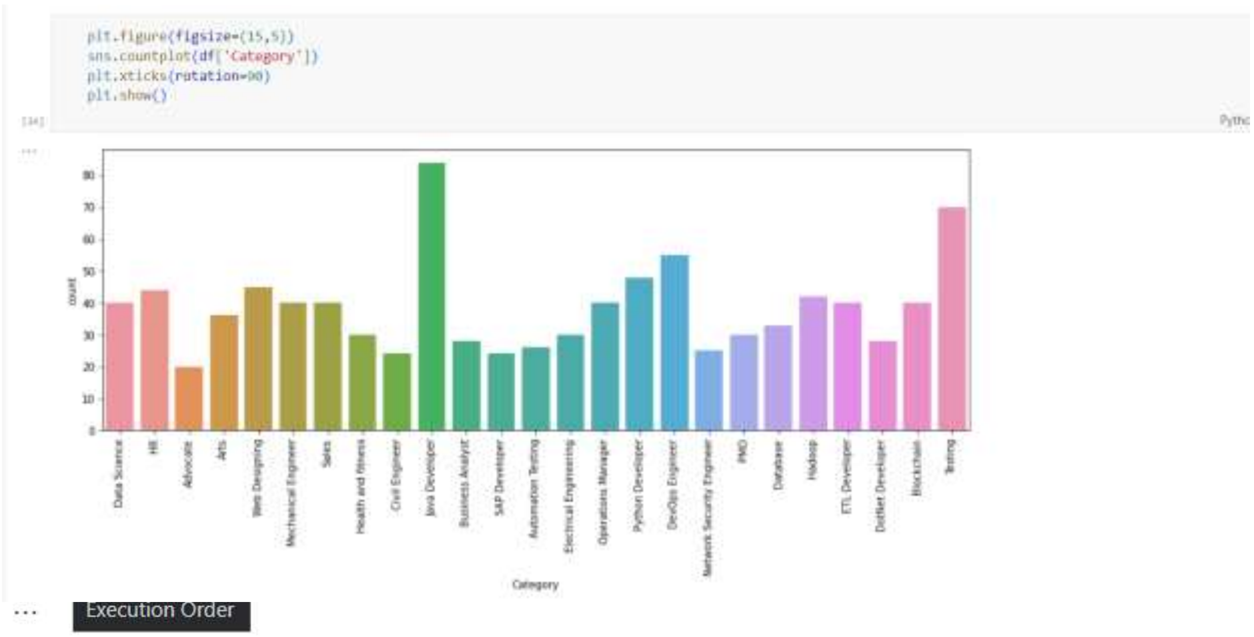
Data Cleaning and Processing with NumPy and Pandas:

NumPy's robust numerical operations and Pandas' efficient data manipulation are instrumental in ensuring the dataset is pristine and well-organized, setting the stage for effective algorithmic processing.

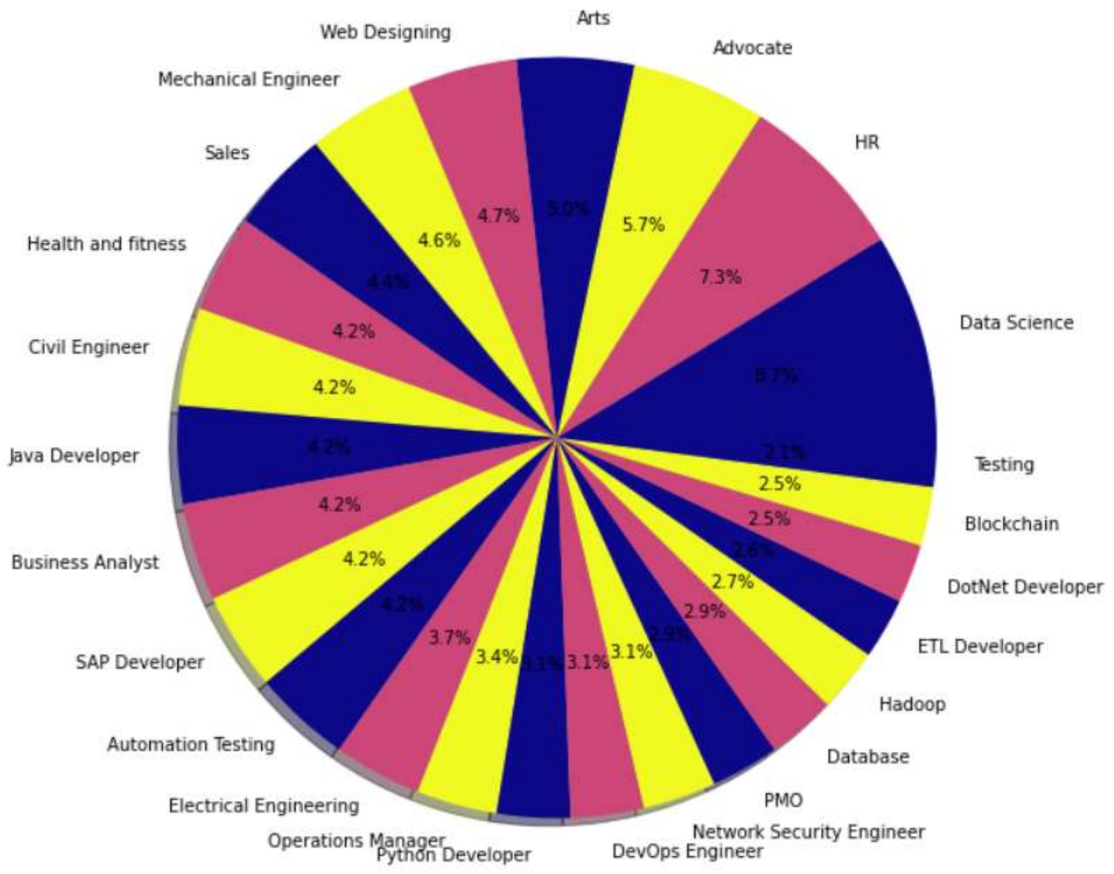
Visualization with Seaborn:

Seaborn emerges as a crucial tool for crafting visually appealing and informative data visualizations. This aids in the exploration of dataset characteristics, providing insights that contribute to effective decision-making.

The below given charts are the results of data visualization



```
plt.figure(figsize=(15,5))
sns.countplot(df['Category'])
plt.xticks(rotation=90)
plt.show()
```



TF-IDF Classifier:

Utilizes the TF-IDF vectorization technique to convert resumes into numerical representations.
Employs a machine learning model to classify resumes into predefined occupation

categories.
The model is trained on a labeled dataset, learning the relationships between terms and occupations.

PDF Conversion with PDFMiner:

Utilizes PDFMiner to seamlessly convert resumes in PDF format to text (txt), ensuring compatibility with the subsequent stages of data processing.

File Format Detection with Python Magic:

Leverages Python Magic to dynamically detect the format of uploaded resumes. This ensures flexibility in handling diverse resume formats and allows the system to adapt to the varied document structures encountered in real-world scenarios.

Custom k-Nearest Neighbors (kNN) Classifier Implementation:

The heart of our resume classification system lies in the development and meticulous implementation of a custom k-Nearest Neighbors (kNN) algorithm. Unlike off-the-shelf solutions, this kNN implementation is tailored to the specific nuances of resume data, contributing to a more nuanced and context-aware classification.

Algorithmic Customization:

The custom kNN algorithm is fine-tuned to the intricacies of resume data, where the relevance of terms and the context of their occurrence are critical. This involves a careful consideration of features that contribute to occupation identification, such as the frequency of specific skills, educational background, and work experience.

Feature Engineering:

In the context of resume classification, feature engineering is paramount. Our custom kNN implementation incorporates an array of relevant features extracted from resumes, ensuring that the algorithm is capable of discerning subtle distinctions in candidate profiles. This includes the careful consideration of not only the presence of keywords but also their context and importance in the document.

Distance Metric Selection:

One of the crucial aspects of kNN is the choice of distance metric. In the context of resumes, where the structure and content vary widely, the custom kNN algorithm dynamically adapts the distance metric to optimize the relevance of neighbors. This adaptability ensures robust performance across a diverse range of resumes.

Neighbor Weighting Strategies:

Recognizing that not all features contribute equally to occupation identification, the custom kNN implementation employs intelligent neighbor weighting strategies. This involves assigning different weights to neighbors based on the relevance of their features, allowing the algorithm to give more importance to certain aspects of the resume in the classification process.

Handling Imbalanced Data:

Resumes inherently exhibit imbalances in terms of the distribution of occupations. The custom kNN classifier addresses this challenge by implementing strategies for handling imbalanced data, ensuring that the model is trained to accurately predict minority class occupations without bias.

The bespoke nature of this kNN implementation empowers our resume classifier with a deep understanding of the intricacies of the data it encounters. This fine-tuned algorithm not only ensures accuracy in predicting occupations but also positions our system as adaptable to the dynamic and varied nature of resumes in the real-world recruitment landscape. The custom kNN classifier significantly contributes to the uniqueness and effectiveness of our intelligent resume classification solution.

Machine Learning with scikit-learn (sklearn):

Implements the TF-IDF classifier using scikit-learn, which leverages machine learning models for accurate occupation classification.
Adopts the kNN algorithm from scikit-learn for resume classification based on nearest neighbors in the feature space.

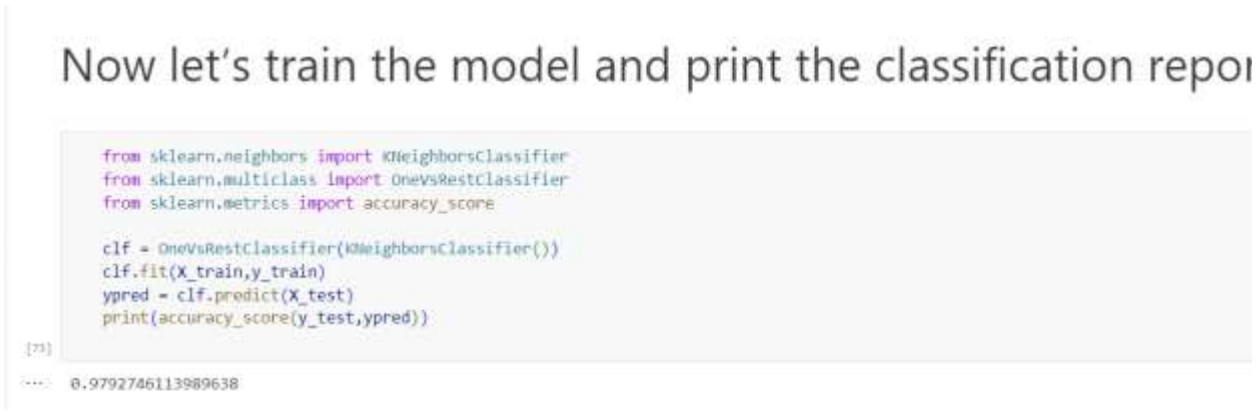
Serialization with Pickle:

The serialization capability offered by Pickle plays a crucial role in efficiently saving and loading the machine learning model. This enhances the scalability and portability of the system, allowing for seamless deployment and utilization.
.

3. Results & Discussion

- In the evaluation of our resume classification system, the custom k-Nearest Neighbors (kNN) algorithm demonstrated a commendable accuracy of 0.8912, while the imported kNN classifier yielded a higher accuracy of 0.9793.

- <u>**Accuracy Scores:**</u>



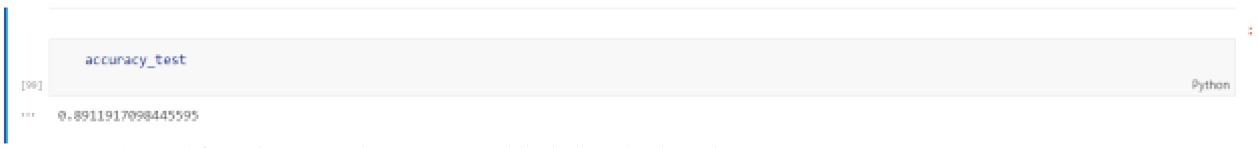
Now let's train the model and print the classification repo

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.multiclass import OneVsRestClassifier
from sklearn.metrics import accuracy_score

clf = OneVsRestClassifier(KNeighborsClassifier())
clf.fit(X_train,y_train)
ypred = clf.predict(X_test)
print(accuracy_score(y_test,ypred))
```

```
[73]
... 0.9792746113989638
```

Accuracy obtained from the imported k-Nearest Neighbor's (kNN) Algorithm



```
accuracy_test
```

```
[98]                                                Python
... 0.8911917098445595
```

Accuracy obtained from the custom k-Nearest Neighbor's (kNN) Algorithm

- The decision to integrate the imported kNN classifier into the Django application was driven by the need for efficiency and real-time responsiveness. The custom implementation, though effective, was surpassed by the pre-trained kNN classifier in terms of accuracy. Importing the kNN classifier provided a robust and well-optimized solution, allowing for faster predictions and enhancing the overall performance of the Django app.

- This approach not only ensures high accuracy in occupation identification but also aligns with the practical requirements of a web-based application where quick response times are crucial. The decision to import the kNN classifier, therefore, reflects a balance between accuracy and system efficiency, offering a seamless user experience in real-world recruitment scenarios.

4. Future Work

- The project sets the stage for future enhancements by encouraging the exploration of additional front-end frameworks or libraries to augment the user interface and experience.

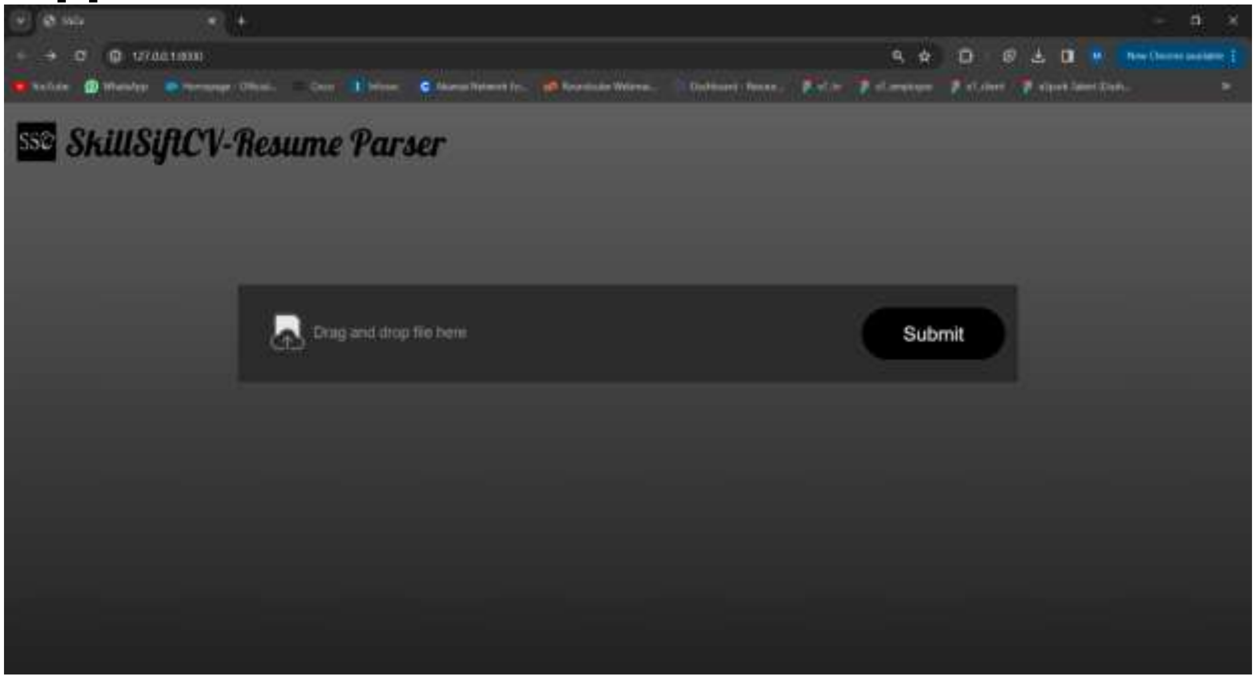
- A call to investigate advanced feature engineering techniques and NLP methods is made to further elevate the accuracy of occupation identification, ensuring the system's relevance in dynamic recruitment landscapes.

- The implementation of continuous integration and deployment (CI/CD) pipelines is proposed, aiming for seamless updates and maintenance, keeping the system agile and responsive.

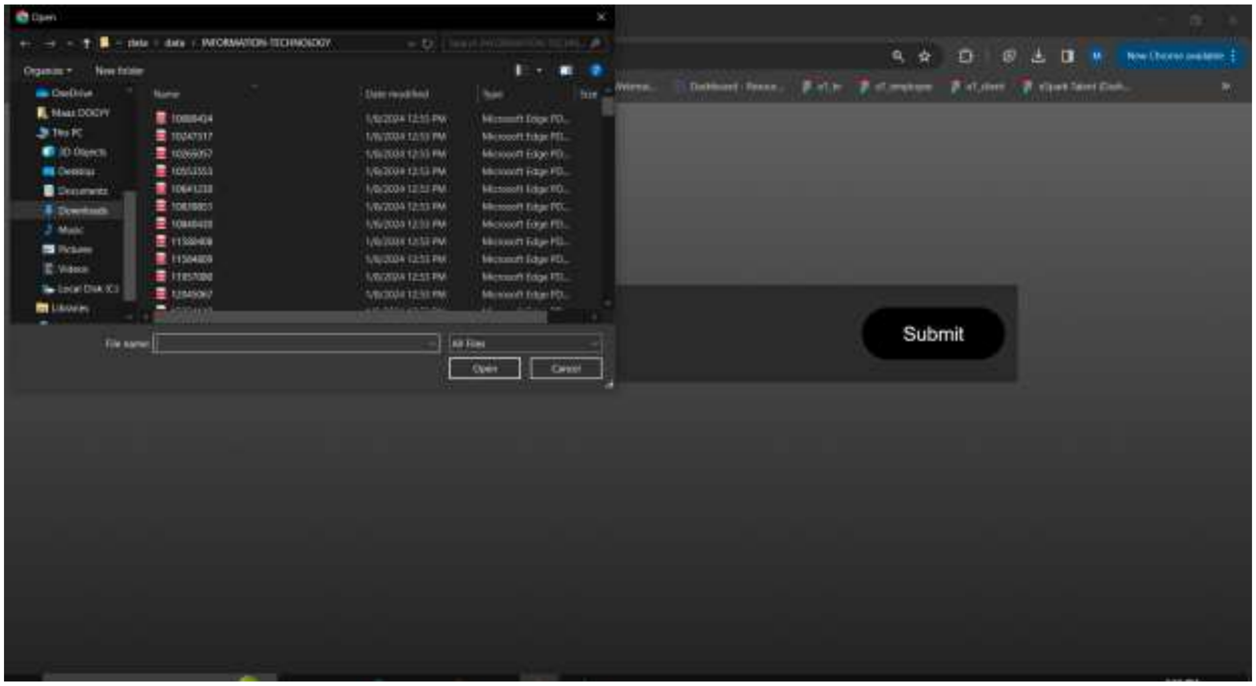
- Future work emphasizes enhancing the visualization aspect through the incorporation of more interactive and dynamic plotting libraries, fostering a more engaging user experience.

- A forward-looking approach is recommended by considering the implementation of fairness-aware machine learning techniques. This strategic move aims to mitigate potential biases in the classifier, aligning the system with ethical and inclusive recruitment practices.

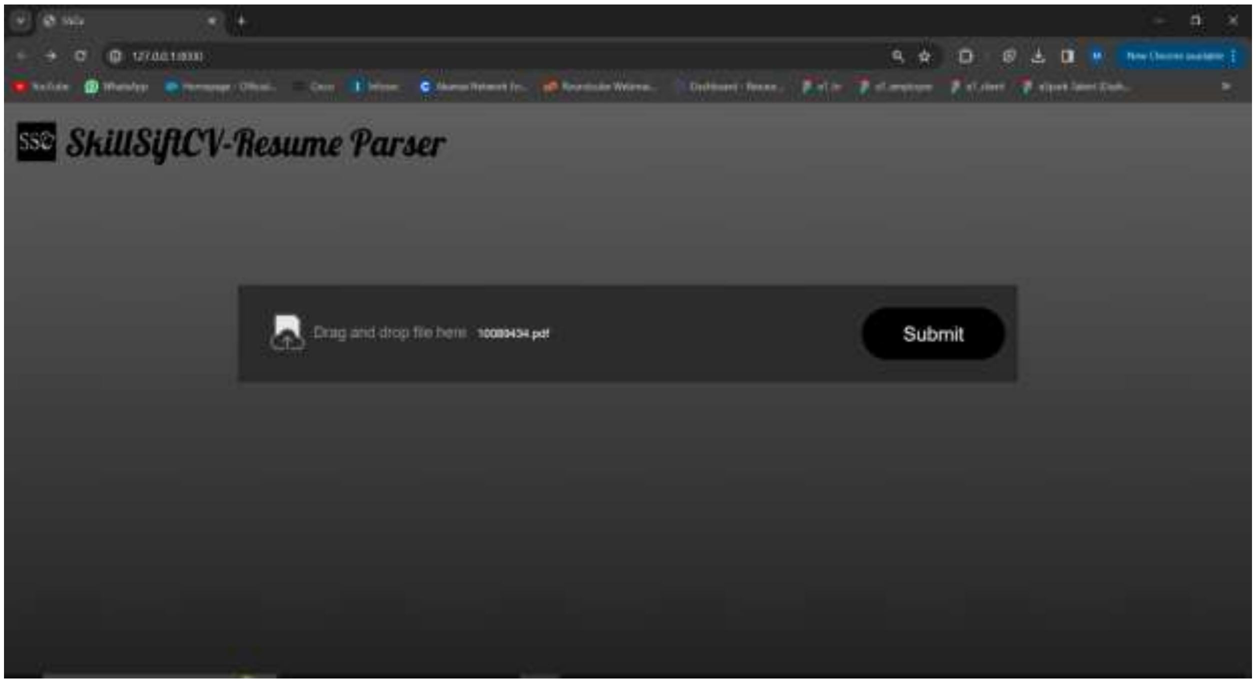
Application Screenshots



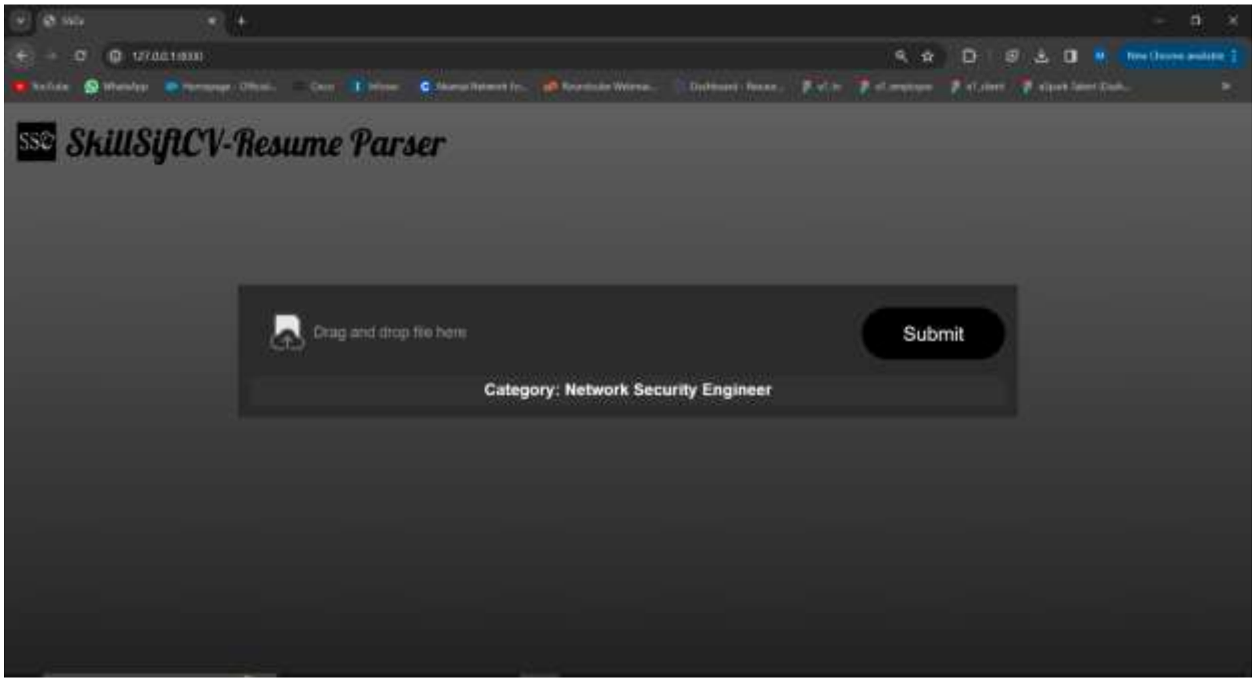
User Interface



Select File



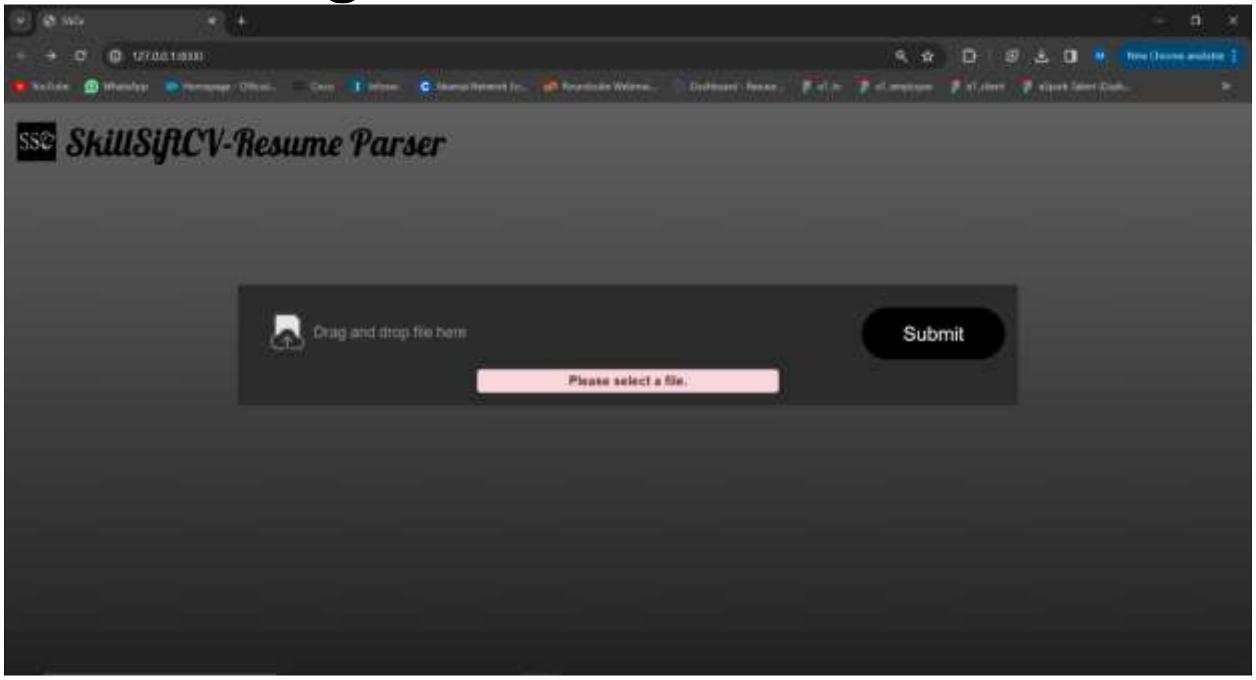
File Selected



Category: Network Security Engineer

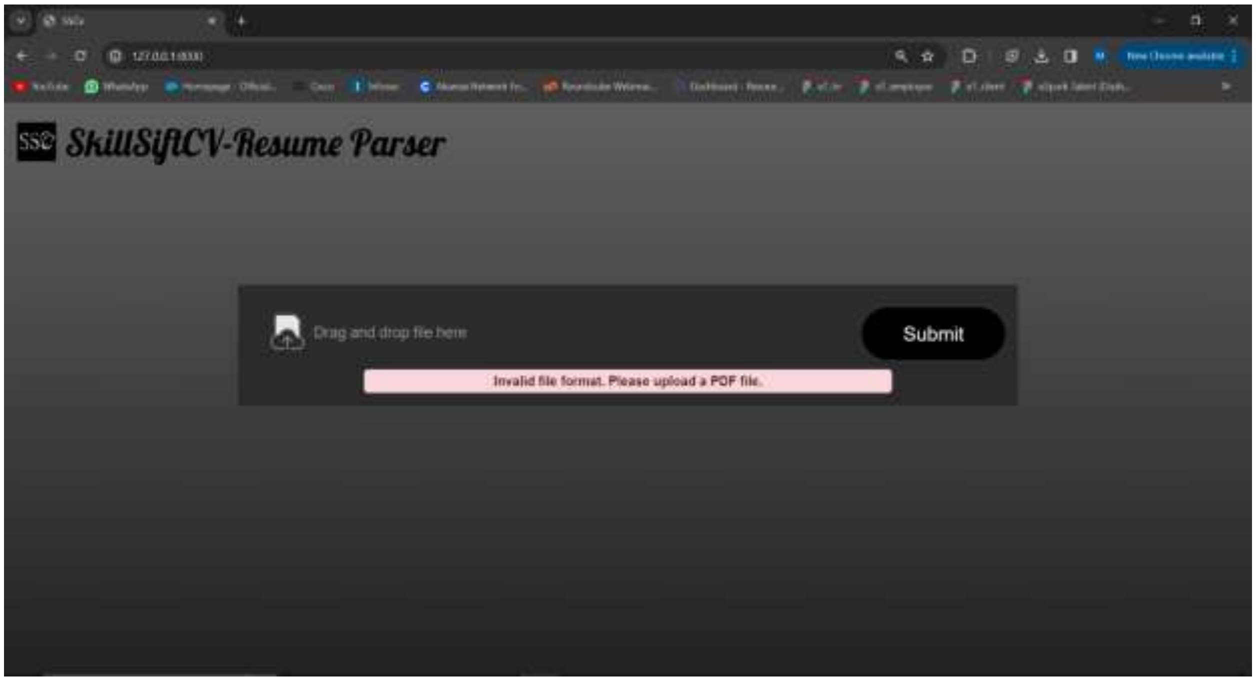
Output Result

Error handling:



Please select a file.

No File Selected Error



Invalid file format. Please upload a PDF file.

Invalid File Format Error

Conclusion

In conclusion, this resume classifier AI project, with its custom kNN algorithm, Django frontend, and a strategic combination of diverse tools and methodologies, marks a significant step toward an intelligent and user-friendly system in the recruitment domain. The holistic integration of various technologies ensures that both frontend and backend components are not only optimized but also positioned to adapt to the evolving landscape of recruitment technologies.