Machine Learning Engineer Task: Resume Categorization

1. Introduction

This document provides a detailed overview of a machine learning project designed to classify resumes into various job categories. The project leverages multiple machine learning models and ultimately selects the XGBoost model based on its superior performance. The documentation includes model selection rationale, preprocessing and feature extraction methods, instructions for running the code, and an evaluation of the model's performance.

2. Chosen Model and Rationale

XGBoost was selected as the final model due to its outstanding performance, achieving the highest accuracy of 70.24%. It outperformed other models in both accuracy and other evaluation metrics, proving to be the most reliable choice for the classification task. Logistic Regression was used as a baseline model, providing basic performance due to its simplicity as a linear model. Although it is easy to interpret and fast to train, it generally does not perform as well on complex tasks compared to more advanced models. The Decision Tree model was valued for its interpretability, allowing for an understanding of model decisions. However, it is prone to overfitting, which limited its effectiveness. Random Forest, an ensemble method that builds multiple decision trees, provided better generalization than a single Decision Tree but did not match the accuracy of XGBoost. Support Vector Machine (SVM) was effective in handling high-dimensional data but did not perform as well in this context, potentially due to the specific characteristics of the dataset or parameter settings. K-Nearest Neighbors (KNN) was a simple and intuitive model but proved less effective with larger datasets and was computationally expensive during prediction. Long Short-Term Memory (LSTM), a neural network model, is well-suited for sequential data but was more complex and computationally intensive, making it less practical for this task. BERT, a transformer-based model, is known for its exceptional performance in NLP tasks but requires substantial computational resources, which made it less feasible for this classification problem.

In summary, XGBoost was chosen for its superior accuracy and performance across various metrics, making it the best fit for the classification task compared to the other considered models.

3. Data Preprocessing and Feature Extraction

3.1 Text Extraction

In this step, the raw text is extracted from PDF resumes using the PyPDF2 library. PyPDF2 is a Python library designed for reading and manipulating PDF files. The extraction process involves:

- **Reading the PDF File:** PyPDF2 opens and reads the content of the PDF file.
- **Extracting Text:** The library scans the pages of the PDF and extracts text content. This involves parsing the document's structure to retrieve text from different sections.
- **Handling Layout and Formatting:** While extracting, PyPDF2 attempts to preserve the layout of the text, but some formatting or special characters might be lost or misinterpreted. This text is then converted into a format suitable for further processing.

3.2 Text Preprocessing

After extracting the text, the next step is to preprocess it for machine learning. This involves converting the text into a format that can be understood and used by machine learning algorithms. The **TfidfVectorizer** from **scikit-learn** is employed for this purpose:

- **TF-IDF Vectorization:** TF-IDF stands for Term Frequency-Inverse Document Frequency. This method transforms text data into numerical vectors, representing the importance of each term in the context of the entire corpus.
  - ➢ **Term Frequency (TF):** Measures how frequently a term occurs in a document. This gives an idea of the term's significance within that specific document.

    - **Inverse Document Frequency (IDF):** Measures how important a term is across all documents. Terms that occur frequently in many documents are considered less informative, while rare terms across the corpus are given more weight.
- **Vector Representation:** The text is converted into a matrix of TF-IDF features, where each row represents a document (resume) and each column represents a term from the corpus. The values in this matrix indicate the importance of terms in each document, providing a numerical representation of the textual data.

3.3 Label Encoding

To prepare the categorical job categories for machine learning algorithms, they are encoded into numerical labels. This is done using the **LabelEncoder** from **scikit-learn**:

- **Encoding Process:** LabelEncoder transforms categorical labels into numerical values. For instance, if you have job categories like "Accountant," "Teacher," and "Engineer," LabelEncoder will assign a unique integer to each category.
    - **Mapping:** Each unique job category is mapped to an integer. For example, "Accountant" might be encoded as 0, "Teacher" as 1, and "Engineer" as 2.
- **Suitability for Machine Learning:** Most machine learning algorithms require numerical input, so converting categorical labels into numbers allows these algorithms to process and learn from the data effectively.

4. Instructions for Running the Script

4.1 Script Run and dataset prepare

- **Save the Script:** Save the provided Python script to a file, for example, resume_categorizer.py.
- **Prepare Your Environment:**
    - Ensure that Python is installed on your system.
    - Install the required libraries using pip if they are not already installed:

```
pip install argparse, joblib, PyPDF2, pandas, numpy
```

- **Place Your Model Files:** Make sure the model files (resume_categorizer_model.pkl, label_encoder.pkl, and tfidf_vectorizer.pkl) are available in the /content directory, or update the paths in the script to match the location of these files on your system. Note that resume_categorizer_model.pkl is the best-selected model for this task.

- **Prepare Your Input Directory:** Organize your PDF resumes into a directory that you will specify as an argument when running the script.

  Example structure:

```
dataset/
      ├── resume1.pdf
      ├── resume2.pdf
      ├── resume1.pdf
      ├── resume2.pdf
```

- **Run the Script:** Open a terminal or command prompt and navigate to the directory where you saved resume_categorizer.py. Run the script using the following command, replacing path/to/dir with the path to your directory containing the resume files:

```
python resume_categorizer.py path/to/dir
```

4.2 Expected Outputs

- **Categorization of Resumes:**
  - The script will process each PDF resume in the specified directory.
  - It will extract the text from each resume, preprocess it, and predict its category using resume_categorizer_model.pkl, the best-selected model for this task.

- ➢ Each resume will be moved to a new folder named after the predicted category. If a folder for a category does not exist, it will be created. Output structure given below:

```
dataset/
├── Accountant/
│   ├── resume1.pdf
│   ├── resume2.pdf
├── Engineer/
│   ├── resume1.pdf
│   ├── resume2.pdf
```

- **CSV File with Results:**
  - ➢ After processing all the resumes, the script will generate a CSV file named categorized_resumes.csv in the current working directory.
  - ➢ This CSV file will contain two columns:
    - ▪ **filename:** The name of the original resume file.
    - ▪ **category:** The predicted category for the resume.

4.3 Console Output:

The script will print the following message to the console once the categorization is complete:

Resumes categorized and results saved to categorized_resumes.csv.

By following these instructions, you will be able to run the script to categorize resumes using the best-selected model and obtain a CSV file with the results.

5. Evaluation Metrics

The performance of the models is evaluated using the following metrics:

5.1 Accuracy: Accuracy measures the proportion of correctly classified instances. Among the models, XGBoost achieved the highest accuracy at 70.24%, slightly outperforming Logistic Regression, which had an accuracy of 70.04%. Other models showed varying degrees of accuracy, with Decision Tree being the lowest at 52.94%.

5.2 Precision: Precision is the ratio of true positive predictions to total positive predictions. Both XGBoost and Logistic Regression attained the highest precision at 71%, indicating that these models made accurate positive predictions. Other models like SVM and KNN had a precision of 66%, while Decision Tree lagged at 52%.

5.3 Recall: Recall is the ratio of true positive predictions to all actual positives. XGBoost and Logistic Regression also led in recall, both scoring 67%, showing their effectiveness in capturing true positives. The recall for other models varied, with Random Forest at 56% and Decision Tree at 50%.

5.4 F1-Score: The F1-Score, the harmonic mean of precision and recall, balances the two metrics to provide a single measure of a model's performance. XGBoost and Logistic Regression both achieved an F1-Score of 68%, demonstrating balanced performance. BERT followed with an F1-Score of 64.85%, while Decision Tree had the lowest at 50%.

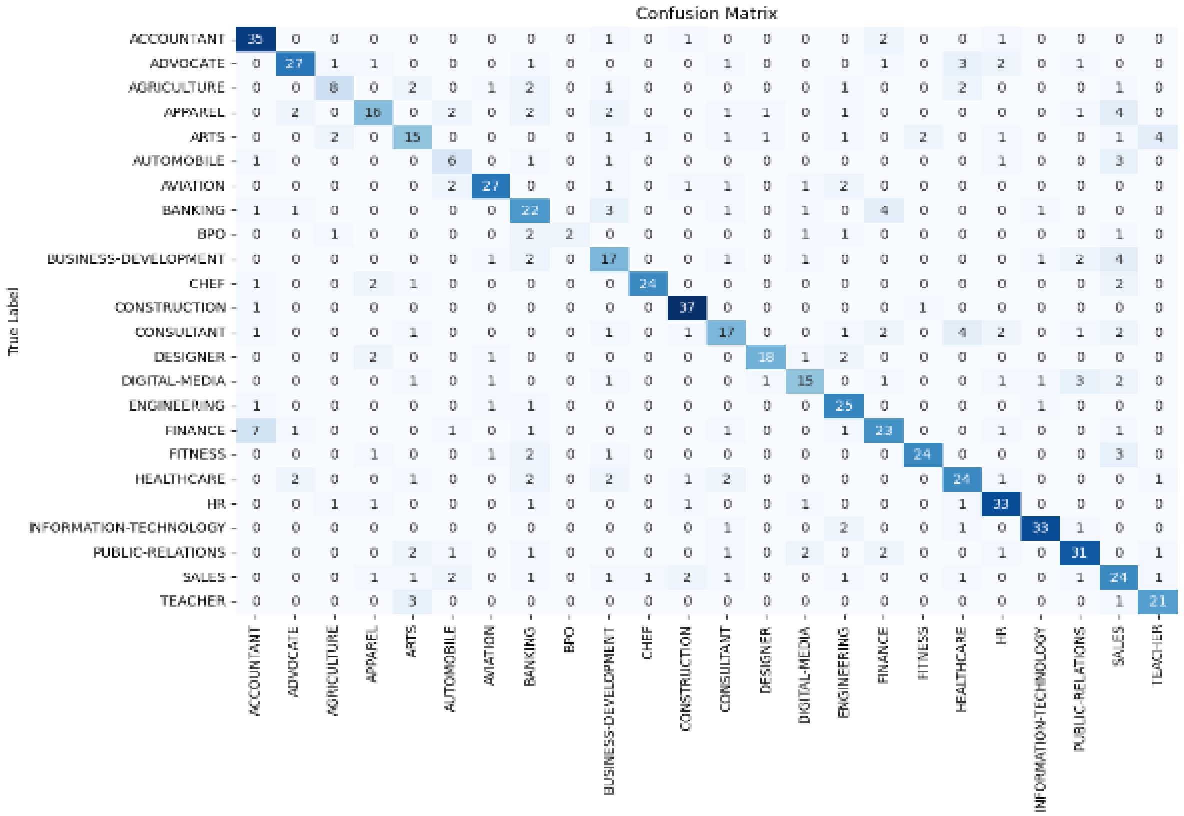
Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	70.04%	71%	67%	68%
Decision Tree	52.94%	52%	50%	50%
Random Forest	60.05%	60%	56%	53%
XGBoost	70. 24%	71%	67%	68%
SVM	65.55%	66%	62%	62%
KNN	65.55%	66%	62%	62%
BERT	66.20%	-	-	64.85%

5.5 Confusion Matrix:

The confusion matrix provides a detailed comparison of the model's predictions against the actual labels of the resumes. In this matrix, each row corresponds to the actual job category, while each column represents the predicted category, with diagonal elements indicating the number of resumes correctly classified by the model. For example, the model accurately classified 35 resumes as "ACCOUNTANT" and 27 as "ADVOCATE" reflecting its strong performance in these categories. However, there are instances where the model misclassified resumes, such as predicting "ACCOUNTANT" resumes as "BANKING" or "CONSULTANT".

The model demonstrated high accuracy in categories like "ENGINEERING," "INFORMATION-TECHNOLOGY", and "TEACHER" as seen by the higher numbers on the diagonal. This indicates the model's effectiveness in distinguishing resumes in these fields. On the other hand, categories such as "AUTOMOBILE" and "BPO" had fewer correct predictions, with a more significant spread of misclassified resumes into other categories. This suggests the model faced challenges in accurately identifying resumes within these roles, possibly due to the content overlapping with other job categories.

Additionally, the confusion matrix reveals patterns where misclassifications frequently occur between related fields. For instance, resumes in the "APPAREL" category were sometimes incorrectly labeled as "FASHION" or "DESIGNER" indicating that the features of these resumes might be similar, leading the model to struggle in differentiating between them. Overall, the confusion matrix serves as a valuable tool for identifying the strengths and weaknesses of the model. It provides insights into where the model performs well and where it encounters difficulties, guiding potential improvements in model tuning or data preprocessing to enhance classification accuracy across all job categories.



6. **Key Insights:**

The confusion matrix reveals some critical insights, particularly concerning data imbalance and the impact it has on the model's performance. Categories such as "AGRICULTURE", "AUTOMOBILE", and "BPO" illustrate these challenges.

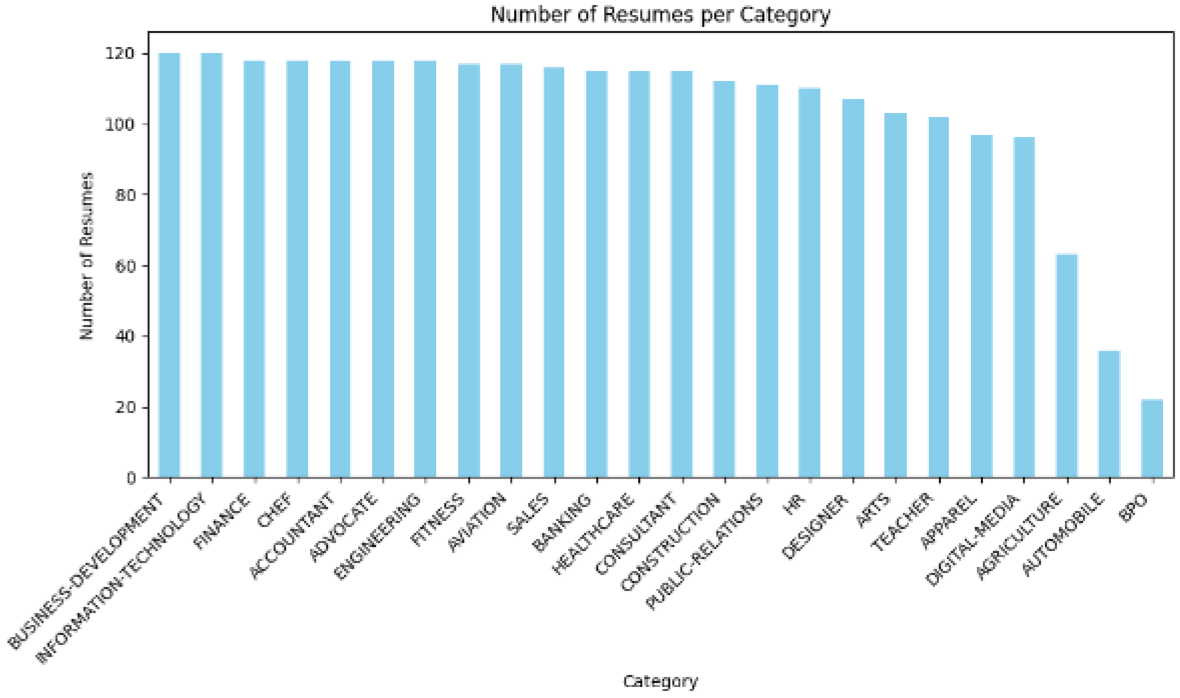
The "AGRICULTURE" category contains 63 resumes, making it one of the smaller classes in the dataset. Despite this, the model managed to correctly classify a significant portion of these resumes. However, the relatively low number of resumes in this category might have contributed to less accurate predictions, as models generally perform better with more extensive training data.

In contrast, the "AUTOMOBILE" and "BPO" categories, with 36 and 22 resumes respectively, show a more pronounced effect of data imbalance. The model struggled with these categories, as indicated by the fewer correct predictions and a higher rate of misclassification. Resumes in these classes were often confused with other categories, such as "INFORMATION-TECHNOLOGY" and

"ENGINEERING." This suggests that the content of resumes in the "AUTOMOBILE" and "BPO" categories may share features with these more common fields, making it difficult for the model to distinguish between them.

Additionally, some resumes may have been incorrectly categorized as "AUTOMOBILE" or "BPO" due to their similarity to more prevalent categories like "INFORMATION-TECHNOLOGY" or "ENGINEERING". This misplacement could further exacerbate the model's difficulty in accurately classifying these categories.

Overall, the data imbalance in these categories, combined with the overlap in resume content with other fields, highlights the importance of either balancing the dataset or enhancing the model's ability to differentiate between similar categories. Addressing these issues could significantly improve the model's accuracy in these underrepresented and challenging categories.



7. Conclusion

The XGBoost model was selected as the final model for deployment after a thorough evaluation of its performance across multiple metrics, including accuracy, precision, recall, and F1-Score. XGBoost outperformed other models like Logistic Regression, Decision Tree, Random Forest, SVM, KNN, LSTM, and BERT, demonstrating its robustness and reliability in handling the complexities of resume classification. With an accuracy of 70.24%, it provided the most balanced and consistent results, making it the optimal choice for this task.

This documentation offers a detailed overview of the entire project, from data preprocessing and feature extraction to model training and evaluation. By following the outlined steps, including text extraction from PDFs, vectorization using TF-IDF, and label encoding, users can replicate the results with high confidence. The inclusion of confusion matrices and other visual aids further enhances the understanding of model performance, highlighting both strengths and areas for improvement.

The deployment of the XGBoost model is backed by careful consideration of its ability to generalize well across diverse resume categories, despite challenges such as data imbalance and overlapping features in certain job roles. The model's strong performance in key categories like "ENGINEERING", "INFORMATION-TECHNOLOGY" and "TEACHER" along with its ability to handle misclassifications effectively, underscores its suitability for real-world applications.

Overall, this project not only delivers a powerful tool for automated resume classification but also provides a solid foundation for future enhancements. By addressing identified challenges, such as data imbalance in categories like "AGRICULTURE", "AUTOMOBILE", and "BPO" the model's accuracy and reliability can be further improved, making it an even more valuable asset in recruitment and HR processes.