

# Document Classification Project Report: Resume vs. Non-Resume Documents.

By - [Divyanshu Yadav](#)

**Link to Google Colab Notebook:-** [https://colab.research.google.com/drive/1zFaAWomP5g5Rg4JXjXhM6STmGu\\_aNHqp?usp=sharing](https://colab.research.google.com/drive/1zFaAWomP5g5Rg4JXjXhM6STmGu_aNHqp?usp=sharing)

## 1. Introduction

- **Objective:** The aim of this project is to build a machine learning model that classifies text documents as resumes or non-resumes. This project focuses on preprocessing text data, feature extraction, and model training and evaluation.
- **Dataset Link:** <https://www.kaggle.com/datasets/gauravduttakiit/resume-dataset?resource=download>
- **Approach Overview:** The project followed these major steps:
  - Data Loading and Exploration
  - Text Preprocessing
  - Feature Extraction using TF-IDF
  - Model Building with Logistic Regression
  - Model Evaluation with metrics like accuracy, AUC, and confusion matrix
  - Feature Importance Analysis

## 2. Data Loading and Exploration

- The dataset was loaded from a CSV file into a Pandas DataFrame.
- Explored the first few rows of the dataset to understand its structure and ensured there were no missing values.

```
[3] print(data.head())
```

|   | Category     | Resume  |
|---|--------------|---|
| 0 | Data Science | Skills * Programming Languages: Python (pandas...                 |
| 1 | Data Science | Education Details \r\nMay 2013 to May 2017 B.E...                 |
| 2 | Data Science | Areas of Interest Deep Learning, Control Syste...                 |
| 3 | Data Science | Skills â&#226; R â&#226; Python â&#226; SAP HANA â&#226; Table... |
| 4 | Data Science | Education Details \r\n MCA YMAUST, Faridab...                     |

```
[4] # Check for missing values
print(data.isnull().sum())
```

```
Category    0
Resume      0
dtype: int64

No missing values found.
```

### 3. Text Preprocessing

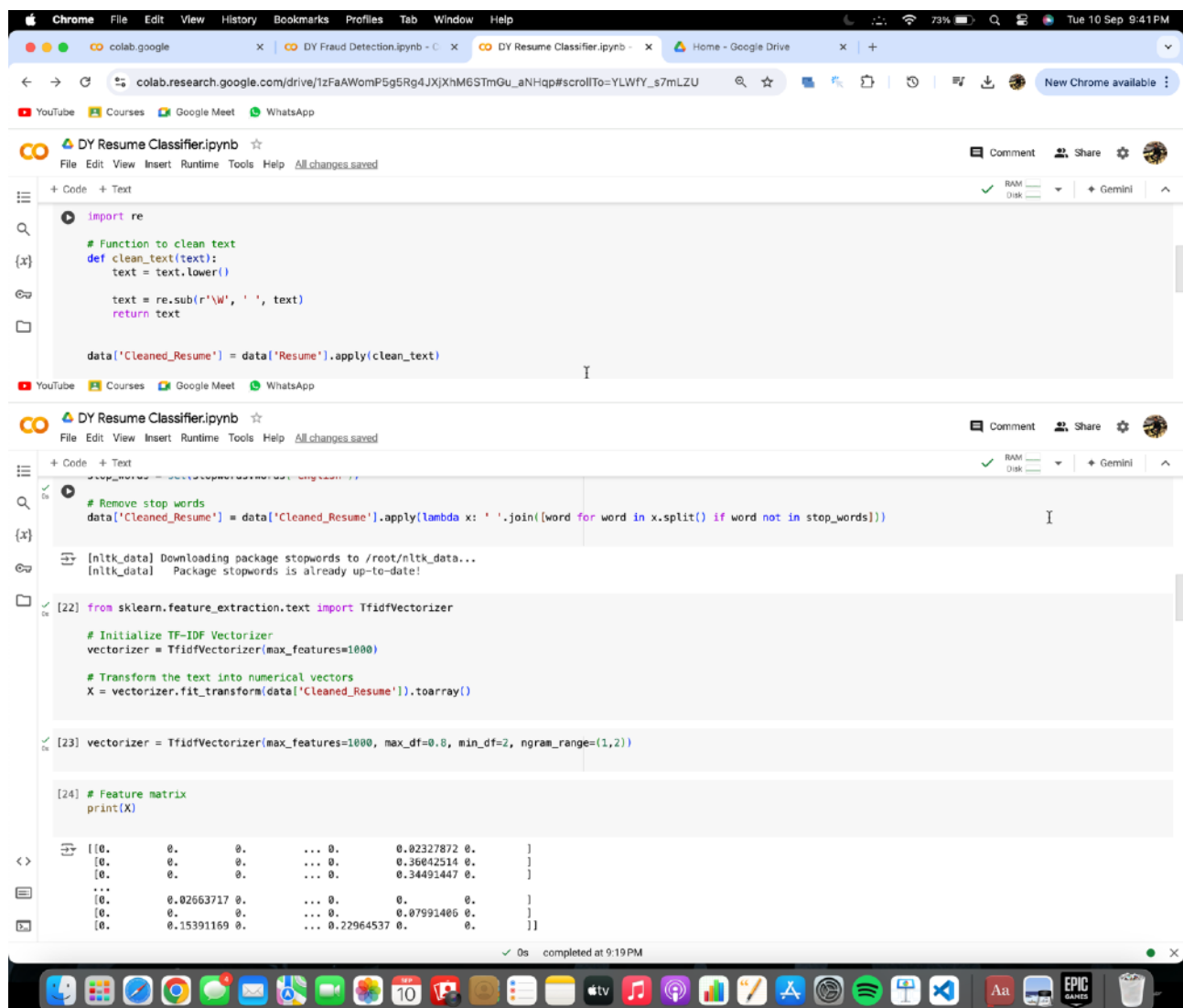
**Challenges:** Text preprocessing is crucial for preparing raw text data for machine learning models. The main challenges were handling inconsistencies in text formatting, stop words, and stemming/lemmatization.

**Steps:**

- **Cleaning:** Converted all text to lowercase and removed non-alphanumeric characters using regular expressions.
- **Tokenization:** Split the text into individual tokens using nltk.
- **Stop Words Removal:** Removed common words (like "the", "is", etc.) that do not provide significant value.
- **Lemmatization:** Reduced words to their base form to handle different word variations.

### 4. Feature Extraction using TF-IDF

**Approach:** To convert the text data into numerical features, I used the **TF-IDF (Term Frequency-Inverse Document Frequency)** method. I limited the maximum number of features to 1000 and also applied n-grams to capture patterns of two or more words.



```
import re

# Function to clean text
def clean_text(text):
    text = text.lower()
    text = re.sub(r'\W', ' ', text)
    return text

data['Cleaned_Resume'] = data['Resume'].apply(clean_text)

# Remove stop words
data['Cleaned_Resume'] = data['Cleaned_Resume'].apply(lambda x: ' '.join(word for word in x.split() if word not in stop_words))

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!

[22] from sklearn.feature_extraction.text import TfidfVectorizer

# Initialize TF-IDF Vectorizer
vectorizer = TfidfVectorizer(max_features=1000)

# Transform the text into numerical vectors
X = vectorizer.fit_transform(data['Cleaned_Resume']).toarray()

[23] vectorizer = TfidfVectorizer(max_features=1000, max_df=0.8, min_df=2, ngram_range=(1,2))

[24] # Feature matrix
print(X)

[[0. 0. 0. ... 0. 0.02327872 0. ]
 [0. 0. 0. ... 0. 0.36042514 0. ]
 [0. 0. 0. ... 0. 0.34491447 0. ]
 ...
 [0. 0.02663717 0. ... 0. 0. 0. ]
 [0. 0. 0. ... 0. 0.07991406 0. ]
 [0. 0.15391169 0. ... 0.22964537 0. 0. ]]
```

## 5. Model Building

**Algorithm Choice:** I used **Logistic Regression**, a robust and interpretable classification algorithm, as the primary model for this task.

- The data was split into training and testing sets (70%-30% split).
- Cross-validation was used to validate the model across different subsets of data.
- Hyperparameter tuning was performed using gridsearchcv to optimize the model.

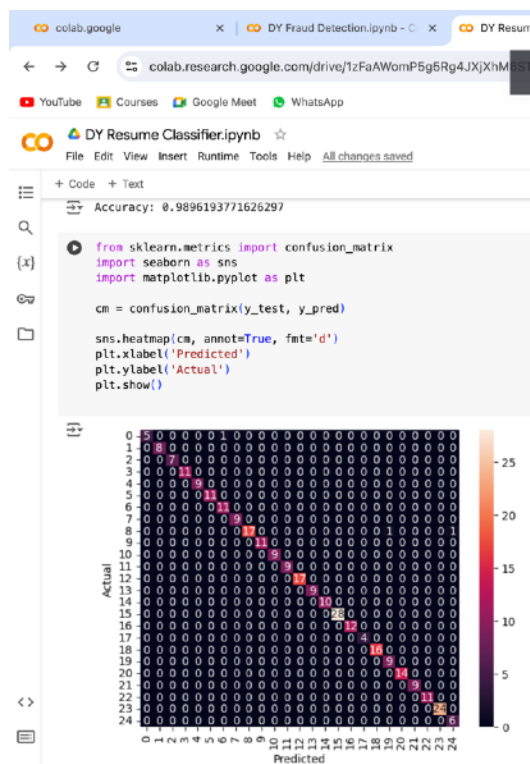
## 6. Model Evaluation

### Metrics Used:

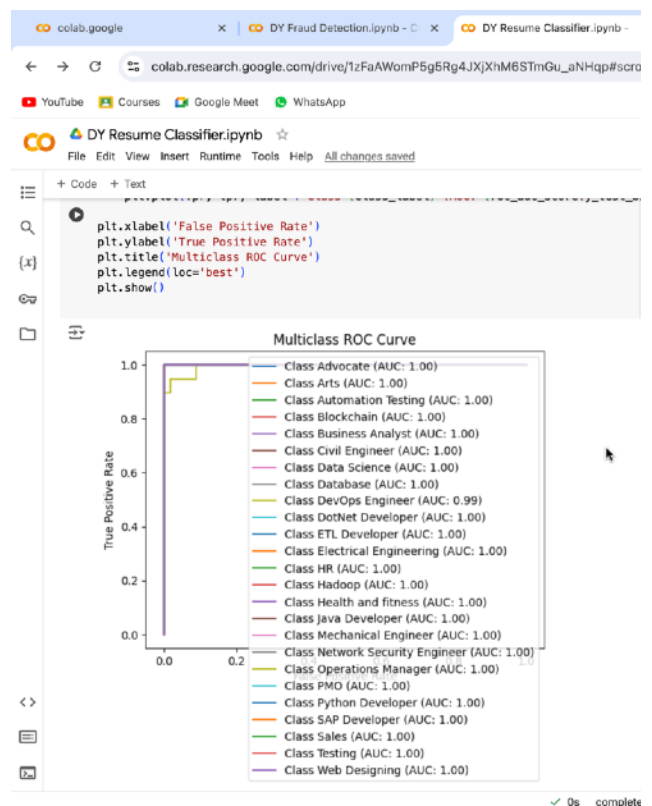
- **Accuracy:** The model achieved an accuracy of ~99%, indicating strong performance on the test set.
- **Confusion Matrix:** The confusion matrix was plotted to visualize the model's prediction performance.
- **ROC Curve and AUC:** ROC curves were plotted for each class, and the AUC scores were calculated to measure the model's classification performance.

### Visualization:

- Confusion Matrix:



- ROC Curve:



## 8. Challenges Faced

- **Text Preprocessing:** Handling inconsistencies in the data, such as formatting issues in text, required careful cleaning and tokenization.
- **Class Imbalance:** In some cases, certain categories had fewer samples, which could affect the model's ability to generalize. I addressed this using proper cross-validation.
- **Hyperparameter Tuning:** Fine-tuning the model's hyperparameters took several iterations to find the optimal values for performance

## 9. Conclusion

- The model successfully classified resumes with high accuracy and demonstrated strong performance across all metrics.
- Text preprocessing and feature extraction played a key role in improving model performance.
- Logistic regression proved to be effective for this task, though other models could be explored in future iterations.

**Accuracy:** The logistic regression model achieved an accuracy of **98.96%**.

**AUC:** The AUC score for multiclass classification using the 'ovr' strategy was **0.99977**.

**Most Important Features:** The words contributing the most to the classification were ['cases' 'com' 'maharashtra' 'details' 'board' 'accounts' 'january' 'university' 'mumbai' 'legal']

By:- Divyanshu Yadav  
divyanshuydv0002@gmail.com  
+91 8588889331