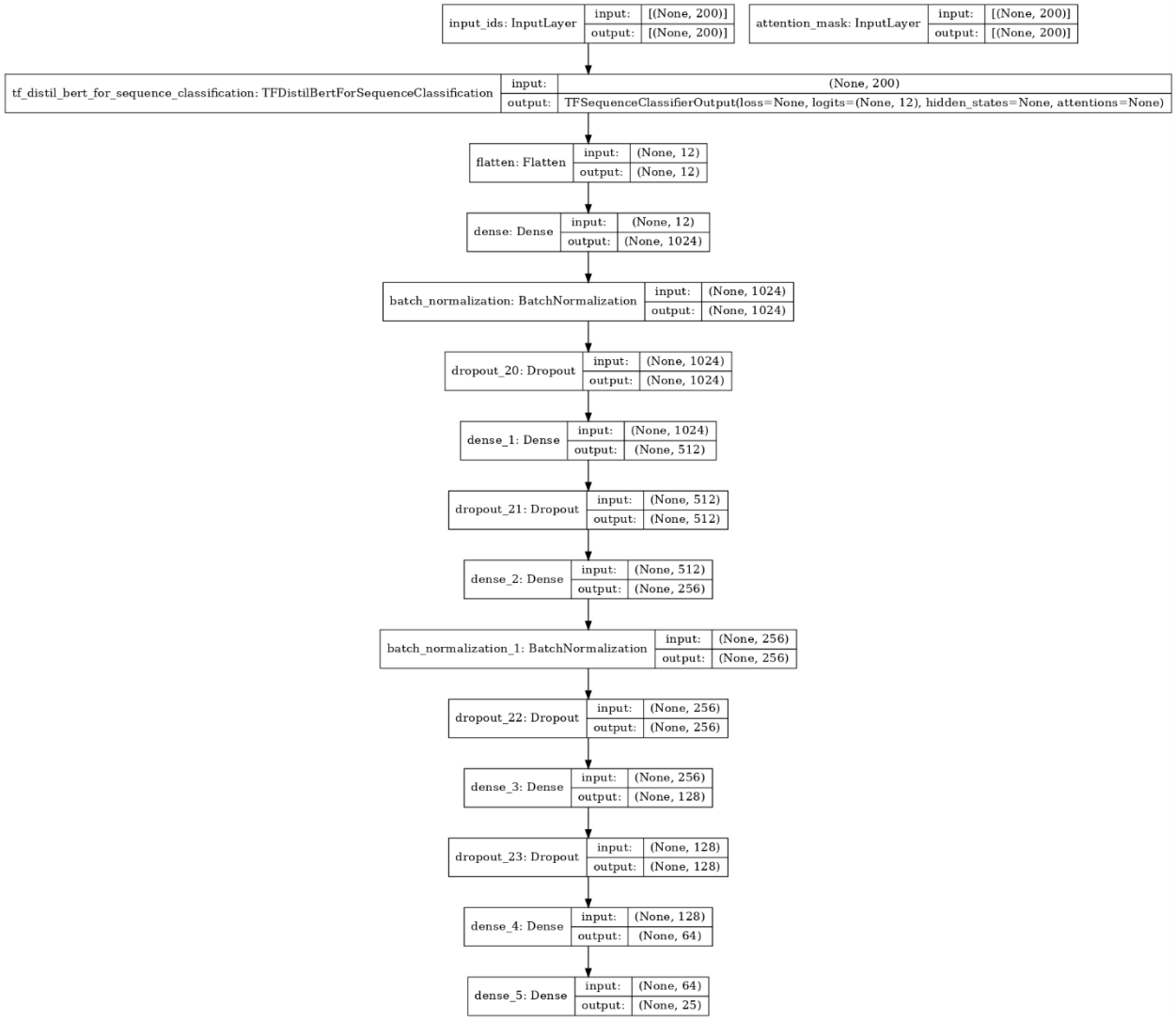
The chosen model and rationale behind the selection:

Model: DistilBERT



DistilBERT stands for "Distillable BERT." It is a variant of the BERT (Bidirectional Encoder Representations from Transformers) model that has been distilled or compressed to be more lightweight and efficient while retaining much of BERT's performance. The "Distillable" aspect refers to the process of distillation, where the knowledge and representations learned by a larger model (in this case, BERT) are transferred or distilled into a smaller model (DistilBERT) to achieve similar performance with reduced computational requirements.

The core idea of DistilBERT is to distill the knowledge of a large BERT model into a smaller, faster, and cheaper model that retains most of the language understanding capabilities of the original model. This is done using a technique called knowledge distillation, in which the small model is trained to predict the outputs of the large model.

The highlights of DistilBERT are:

- It is 40% smaller than BERT, with 66M parameters compared to 110M parameters.
- It is 60% faster than BERT, with a latency of 19ms per sentence on a single GPU compared to 34ms per sentence for BERT.
- It achieves comparable performance to BERT on a range of natural language processing tasks, with a slight drop in accuracy on some tasks.
- It is suitable for deployment on edge devices, such as smartphones and laptops.

DistilBERT is a useful tool for natural language processing tasks where speed and efficiency are important, such as question answering and natural language inference. It is also a good choice for deployment on edge devices, where space and power are limited.

- It is based on the BERT architecture, but with some modifications to reduce the number of parameters.
- The knowledge distillation process is done using a technique called temperature scaling.
- DistilBERT has been shown to be effective on a range of natural language processing tasks, including question answering, natural language inference, and sentiment analysis.

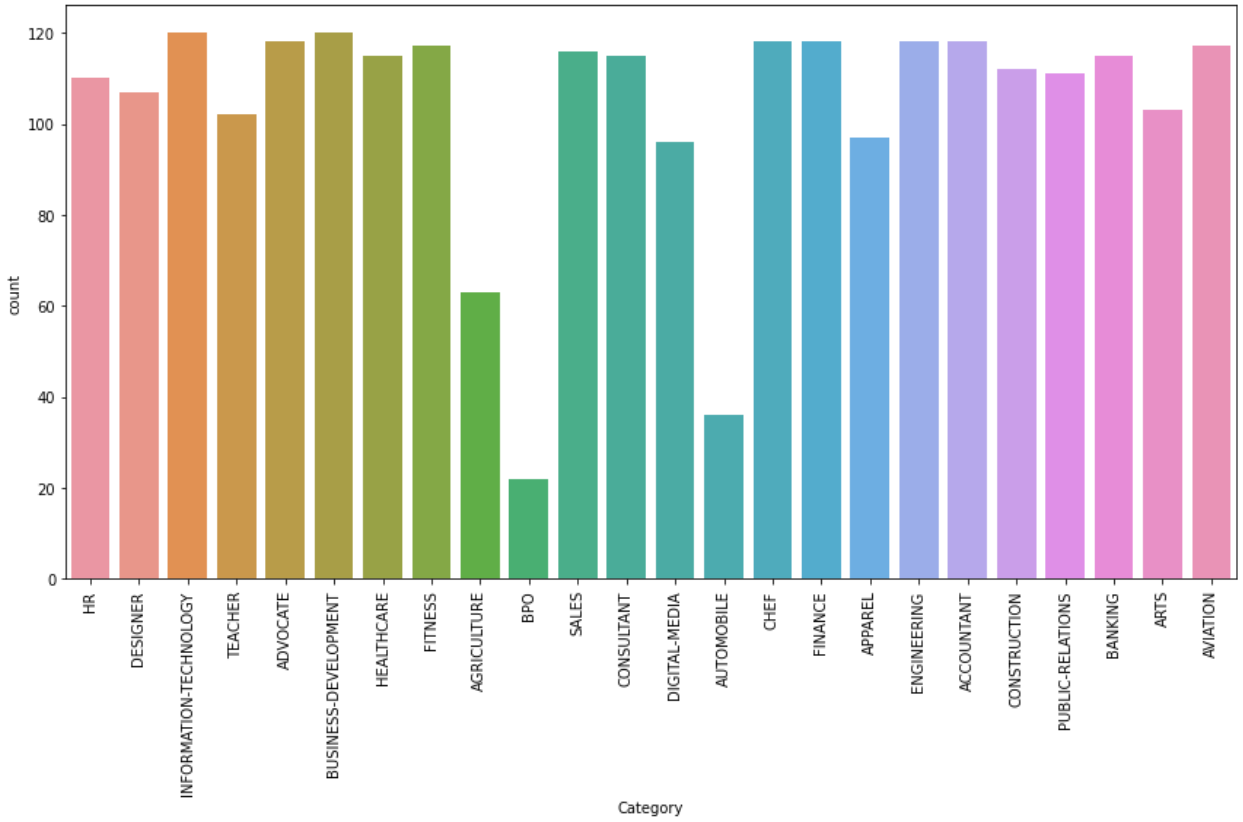
The chosen Preprocessor modules:

This function is used to be a text preprocessing pipeline applied to a specific column of the DataFrame. It follows a series of steps to clean and preprocess text data in that column. Here's a breakdown of each step:

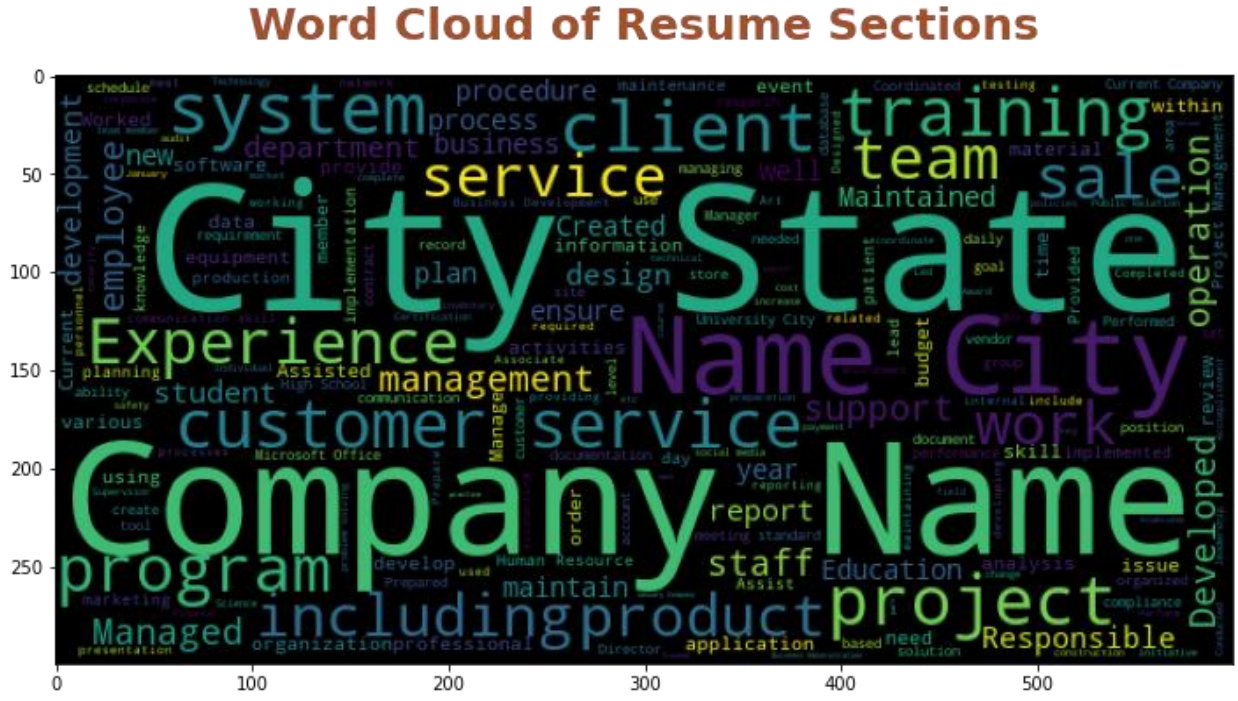
- `clean_html`: This function removes any HTML tags from the text, ensuring that the text is in plain text format without any HTML formatting.
- `remove_`: This function removes some specific characters or patterns from the text.
- `removeStopWords`: This function removes common stop words from the text. Stop words are commonly used words (e.g., "the," "and," "is") that are often removed to reduce noise in text data.
- `remove_digits`: This function removes any numerical digits from the text, potentially eliminating numbers from the text.
- `remove_links`: This function removes any URLs or hyperlinks from the text.
- `remove_special_characters`: This function removes any special characters from the text, such as punctuation marks or symbols.
- `punct`: This function applies punctuation processing to the text. It retains noun, adjectives and drop others.
- `non_ascii`: This function removes any non-ASCII characters from the text, ensuring that the text only contains ASCII characters.
- `email_address`: This function removes or masks email addresses from the text.
- `lower`: This function converts the text to lowercase, standardizing the text to a consistent case.

The Evaluation Matrices:

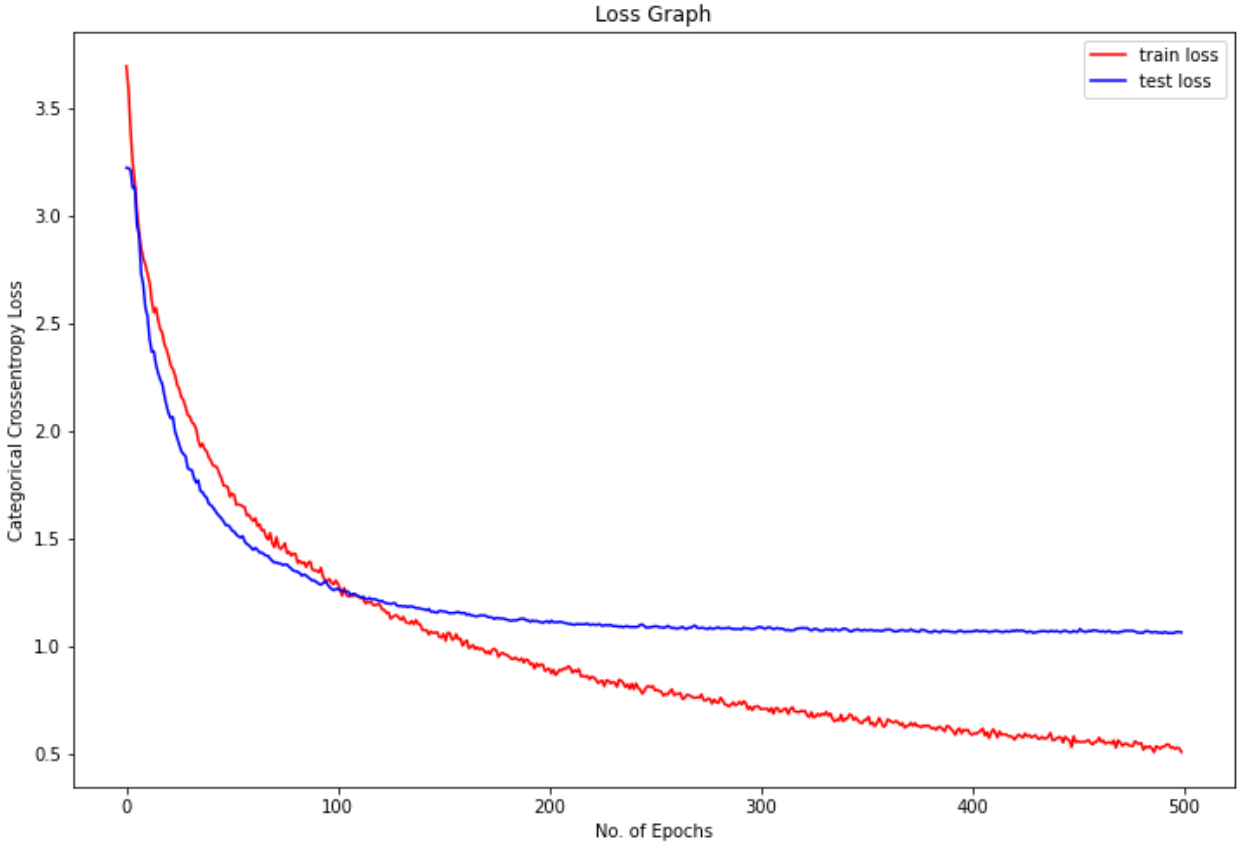
The provided dataset has 24 classes. The below figure showing the total number of resumes in each category.



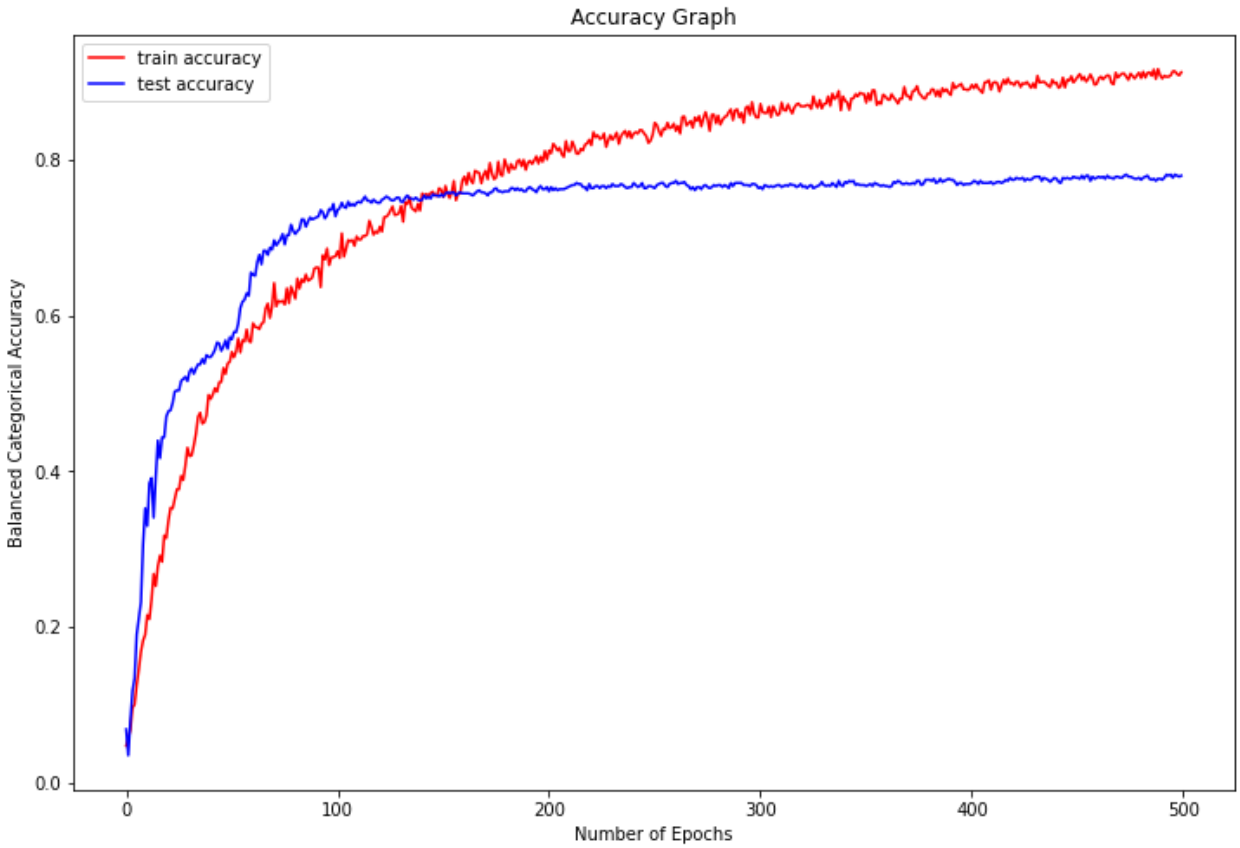
Most frequent words in the provided resumes can be seen in the below word cloud:



Loss Curve in training and validatioin:



Accuracy Curve in training and validation:



The both curve shows that the test loss is increasing and test accuracy is decreasing after around 120 epochs. It is a common phenomenon in machine learning known as **overfitting**. Overfitting occurs when a model becomes too complex and starts to learn noise in the training data, making it perform well on the training data but poorly on new, unseen data (test data). Data diversity should address the overfitting issue.

Accuracy of the model:

Test Sparse Categorical Crossentropy Loss: 1.065347671508789

Test Balanced Categorical Accuracy: 0.7801608443260193

Confusion Matrix:

[[30 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[0 47 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0]

[0 0 35 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 2 1 2]

[0 1 0 22 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0]

[1 0 0 0 27 0 3 2 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 2]

[0 0 0 0 0 37 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

[0 0 0 0 2 0 20 1 0 0 0 1 0 0 1 0 0 0 0 1 1 1 4 1]

[0 0 0 1 2 0 2 22 0 0 2 0 2 0 1 0 0 0 0 0 0 2 2 0]

[0 0 0 0 0 0 1 0 3 0 0 0 3 0 0 0 1 0 0 0 0 2 5 3]

[0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 2]

[0 0 0 0 0 0 0 2 0 0 34 0 0 0 0 0 1 0 0 1 1 0 0 0]

[0 0 0 0 0 0 0 0 0 0 0 37 0 0 0 0 0 0 0 0 0 0 0 1]

[0 0 0 1 0 1 2 0 0 0 2 0 15 0 0 0 0 0 0 0 2 0 1 2]

[0 0 0 0 1 0 2 0 0 0 0 0 0 1 0 0 0 0 1 0 0 2 1 1]

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 30 0 0 0 0 0 1 1 1 1]

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 31 0 0 0 0 0 0 0 0]

[0 3 0 0 1 1 1 5 0 0 1 0 2 0 0 0 10 0 1 0 3 1 2 1]

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 30 0 0 0 0 0 0]

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 30 0 0 0 0 0]

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 38 0 0 0 2]

[0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 1 0 0 0 20 2 1 0]

[0 0 0 0 0 1 1 1 0 0 1 0 0 0 0 1 0 0 1 0 0 28 1 2]

[0 0 0 0 0 0 2 0 0 0 1 1 4 0 0 0 1 0 0 0 0 2 12 4]

[0 0 0 0 2 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 3 3 23]]

Classification Report:

	precision	recall	f1-score	support

0	0.97	1.00	0.98	30
1	0.92	0.98	0.95	48
2	1.00	0.83	0.91	42
3	0.92	0.92	0.92	24
4	0.77	0.71	0.74	38
5	0.93	1.00	0.96	37
6	0.57	0.61	0.59	33
7	0.63	0.61	0.62	36
8	1.00	0.17	0.29	18
9	0.00	0.00	0.00	5
10	0.81	0.87	0.84	39
11	0.95	0.97	0.96	38
12	0.48	0.58	0.53	26
13	1.00	0.11	0.20	9
14	0.94	0.88	0.91	34
15	0.97	1.00	0.98	31
16	0.67	0.31	0.43	32
17	0.97	1.00	0.98	30
18	0.88	1.00	0.94	30
19	0.93	0.95	0.94	40
20	0.69	0.71	0.70	28
21	0.58	0.76	0.66	37
22	0.34	0.44	0.39	27
23	0.49	0.68	0.57	34

accuracy			0.78	746
macro avg	0.77	0.71	0.71	746
weighted avg	0.79	0.78	0.77	746