CSE 572 Data Mining - Homework 1

Submitted by - Aman Bhala

Dataset Description

The dataset is designed for a document classification task, aiming to categorize documents into one of five predefined categories: sport, business, politics, entertainment, and tech. This classification challenge involves analyzing the content of news articles and assigning them to the most relevant category based on their content.

Training Dataset

- Content: The training dataset comprises 1000 news articles, each annotated with a category label that signifies the article's subject matter. This dataset is pivotal for training the models to understand and learn the distinctions between different types of news articles.
- Format: The dataset is stored in a CSV file, featuring three columns:
    - `ArticleId` : A unique identifier for each news article.
    - `Text` : The raw text of the news article, which serves as the input data for model training.
    - `Category` : The label indicating the article's category. This is the target variable for the classification task.

Test Dataset

- Content: The test dataset contains 735 news articles. Unlike the training set, these articles are not labeled, and the task is to predict their categories using the trained model.
- Format: This dataset also comes in a CSV format with two columns:
    - `ArticleId` : A unique identifier for each news article in the test set.
    - `Text` : The raw text of the news article. This data will be used to test the model's ability to classify unseen articles accurately.

The goal of this assignment is to employ tree-based models for document classification, evaluate their performance through 5-fold cross-validation, and use the best-performing model to predict the categories of the test dataset articles.

Preprocessing the Raw Training Data

The preprocessing of the training data is a crucial step in preparing the dataset for effective model training, especially in text classification tasks. The raw text data undergoes several preprocessing steps to transform it into a more analyzable form for machine learning models. Here's a breakdown of the preprocessing steps applied:

Text Preprocessing Steps

Normalization: The raw text of each document is converted to lowercase to ensure uniformity and reduce the complexity introduced by case differences.
Cleaning: Special characters and punctuation marks are removed from the text, leaving only alphanumeric characters and spaces. This step helps eliminate irrelevant features that could potentially confuse the model.
Tokenization: The cleaned text is then tokenized, splitting it into individual words or tokens. This step is essential for further processing like stop word removal and lemmatization.
Stop Word Removal: Common English stop words (e.g., "the", "is", "in") are removed from the tokens. Stop words are typically high-frequency words that add little to no value in understanding the context or sentiment of the text.
Lemmatization: The remaining tokens are lemmatized, converting them to their base or dictionary form. Lemmatization helps in reducing the morphological variation of words, thus consolidating similar forms of a word into a single item.

Feature Extraction

After preprocessing the text, feature extraction is performed using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization method. This technique converts the preprocessed text into a numerical format, creating a matrix where each row represents a document and each column represents a term's TF-IDF score across the corpus. The following settings were used:

- Max Features: The number of features (unique terms) is limited to 1000. This parameter is adjustable based on the model's needs and computational constraints.

The resulting matrix from the TF-IDF vectorization serves as the input features for training the machine learning models. This transformation is vital for analyzing text data, allowing models to understand the importance of each term in the context of the entire corpus.

Building and Evaluating Decision Tree Models

In this phase, the focus was on training and evaluating Decision Tree classifiers for document classification, utilizing the preprocessed and vectorized text data. The primary objective was to investigate the impact of different criterion parameters on model performance. Two criteria, "gini" and "entropy," were tested to determine their effectiveness in improving the model's accuracy in classifying documents into their respective categories.

Model Training and Validation Split

The dataset was divided into training and validation sets following an 80-20% split. This division ensured that the model could be trained on a substantial portion of the data while still being validated on a separate set to gauge its generalization capability.

Criterion Parameters Evaluation

Two Decision Tree models were trained, each using a different criterion for measuring the quality of splits:

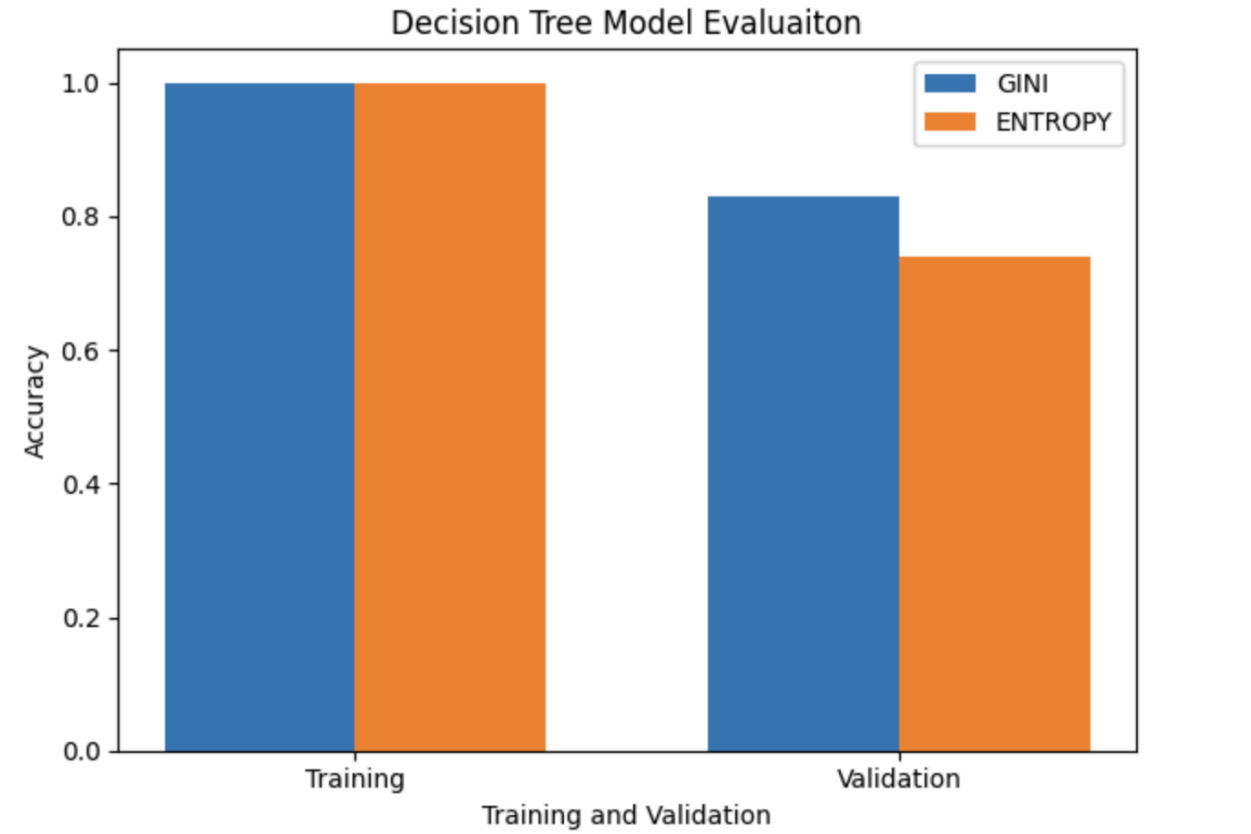
- **Gini Impurity ("gini")**: A measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.
- **Information Gain ("entropy"):** A measure of the reduction in entropy or uncertainty after the dataset is split on an attribute.

Each model was trained on the training set and then evaluated on both the training and validation sets to assess its accuracy.

Results Visualization

The performance of the models, in terms of training and validation accuracy, was visualized using a bar chart. This visualization facilitates a direct comparison between the "gini" and "entropy" criteria regarding their effectiveness in classifying the documents accurately.

- The training accuracy reflects how well each model learned to classify the documents it was trained on, indicating the model's ability to capture the relationship between the document features and their corresponding categories.
- The validation accuracy provides insight into how well the model can generalize its classification capability to unseen data, serving as a crucial indicator of its practical usefulness.



Evaluation of Decision Tree Classifier with Varying (`min_samples_leaf`)

This section of the analysis focused on assessing the impact of varying the `min_samples_leaf` parameter on the performance of a Decision Tree classifier. The evaluation was performed using 5-fold cross-validation, a method that ensures a robust and unbiased estimation of the model's performance by systematically partitioning the original dataset into a set of training and validation datasets for testing.

Parameter Range :

The `min_samples_leaf` parameter specifies the minimum number of samples required to be at a leaf node. This investigation varied `min_samples_leaf` from 1 to 201 in steps of 10, exploring its effect on model accuracy.

Cross-validation Results

The results of the 5-fold cross-validation are organized into a DataFrame, capturing the mean and standard deviation of both training and validation accuracies across different `min_samples_leaf` values. This quantitative analysis helps in understanding the balance between model complexity and generalization capability as influenced by the `min_samples_leaf` parameter.

Key Findings in Tabular Form

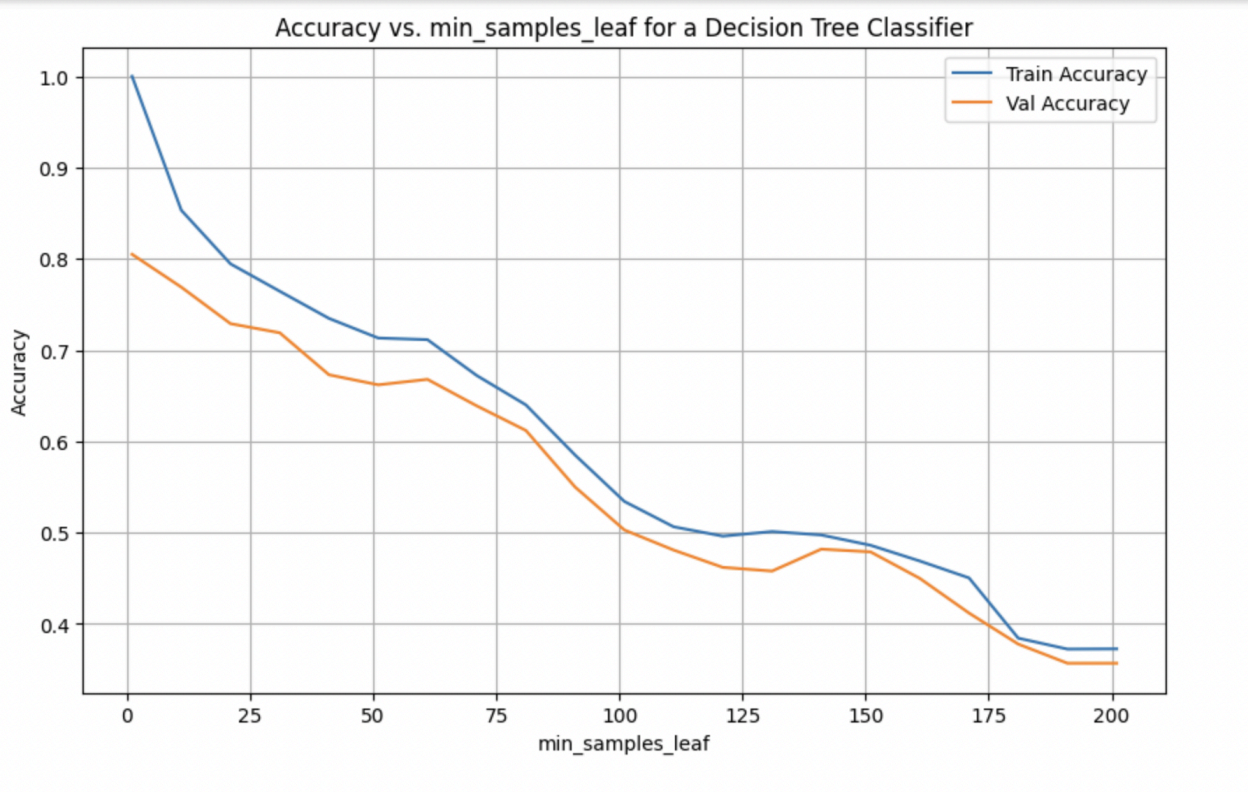
A table was created to systematically report the average training and validation accuracies along with their standard deviations for the tested range of `min_samples_leaf` values. This tabular representation facilitates an easy comparison across the parameter values, aiding in the identification of the optimal `min_samples_leaf` value that balances training performance with validation accuracy.

	min_samples_leaf	mean_train_accuracy	std_train_accuracy	mean_val_accuracy	std_val_accuracy
0	1	1.00000	0.000000	0.805	0.036194
1	11	0.85325	0.002806	0.769	0.037336
2	21	0.79450	0.014089	0.729	0.037603
3	31	0.76450	0.006828	0.719	0.026907
4	41	0.73475	0.012410	0.673	0.027677
5	51	0.71325	0.010112	0.662	0.032031
6	61	0.71150	0.011275	0.668	0.038288
7	71	0.67225	0.026094	0.639	0.043635
8	81	0.64000	0.023979	0.612	0.029933
9	91	0.58500	0.041322	0.550	0.046690
10	101	0.53425	0.028291	0.503	0.023152
11	111	0.50650	0.007842	0.481	0.047476
12	121	0.49625	0.007906	0.462	0.045233
13	131	0.50125	0.016432	0.458	0.036688
14	141	0.49750	0.015227	0.482	0.032031
15	151	0.48625	0.009454	0.479	0.044204
16	161	0.46900	0.012855	0.450	0.029665
17	171	0.45050	0.014504	0.412	0.032650
18	181	0.38450	0.026805	0.378	0.042024
19	191	0.37250	0.004108	0.357	0.009274

Graphical Representation

The analysis further included a line chart visualizing the relationship between the `min_samples_leaf` parameter and the corresponding training and validation accuracies. This visual depiction serves multiple purposes:

- **Training Accuracy Trend**: Demonstrates how the model's ability to learn from the training data varies with the complexity controlled by `min_samples_leaf`.
- **Validation Accuracy Trend**: Highlights how the model's generalization to unseen data is affected by the same parameter, providing insights into the potential for overfitting or underfitting at different parameter values.
- **Optimal Parameter Selection**: Helps in pinpointing the `min_samples_leaf` value that achieves a desirable trade-off between training accuracy and validation accuracy, guiding the selection of a model configuration that is likely to perform well on unseen data.



Conclusion

The evaluation underscores the importance of tuning the `min_samples_leaf` parameter in Decision Tree classifiers. By adjusting this parameter, one can significantly influence the model's performance, optimizing for higher accuracy and better generalization. The detailed tabular data, coupled with the graphical analysis, provides a comprehensive overview that supports informed decision-making regarding model configuration.

Evaluation of Decision Tree Classifier with Varying `max_features`

This analysis segment investigates the influence of the **`max_features`** parameter on the performance of Decision Tree classifiers. The **`max_features`** parameter determines the number of features to consider when looking for the best split, which can significantly impact the model's learning behavior and predictive accuracy.

Parameter Exploration

The **`max_features`** parameter was varied across several values, including:

- **`None`**: Considering all features at each split.

- `'sqrt'`: Considering a random subset of features whose size is the square root of the total number of features.
- `'log2'`: Considering a random subset of features whose size is the log (base 2) of the total number of features.
- A range of explicit numbers (and potentially fractions of the total features) from 10 up to 1000 in steps of 50, allowing for a comprehensive analysis across a spectrum of feature subset sizes.

Cross-validation Methodology

A 5-fold cross-validation strategy was employed for each value of `max_features` to ensure a thorough and unbiased evaluation of the model's performance. This approach partitions the dataset into five subsets, iteratively using one subset for validation and the remaining for training, thereby providing a robust estimate of the model's accuracy.

Results Compilation

The outcomes of the cross-validation process were meticulously compiled into a DataFrame, capturing the mean and standard deviation of both training and validation accuracies for each tested `max_features` value. This structured presentation of results facilitates an in-depth analysis of how different feature subset sizes influence model performance.

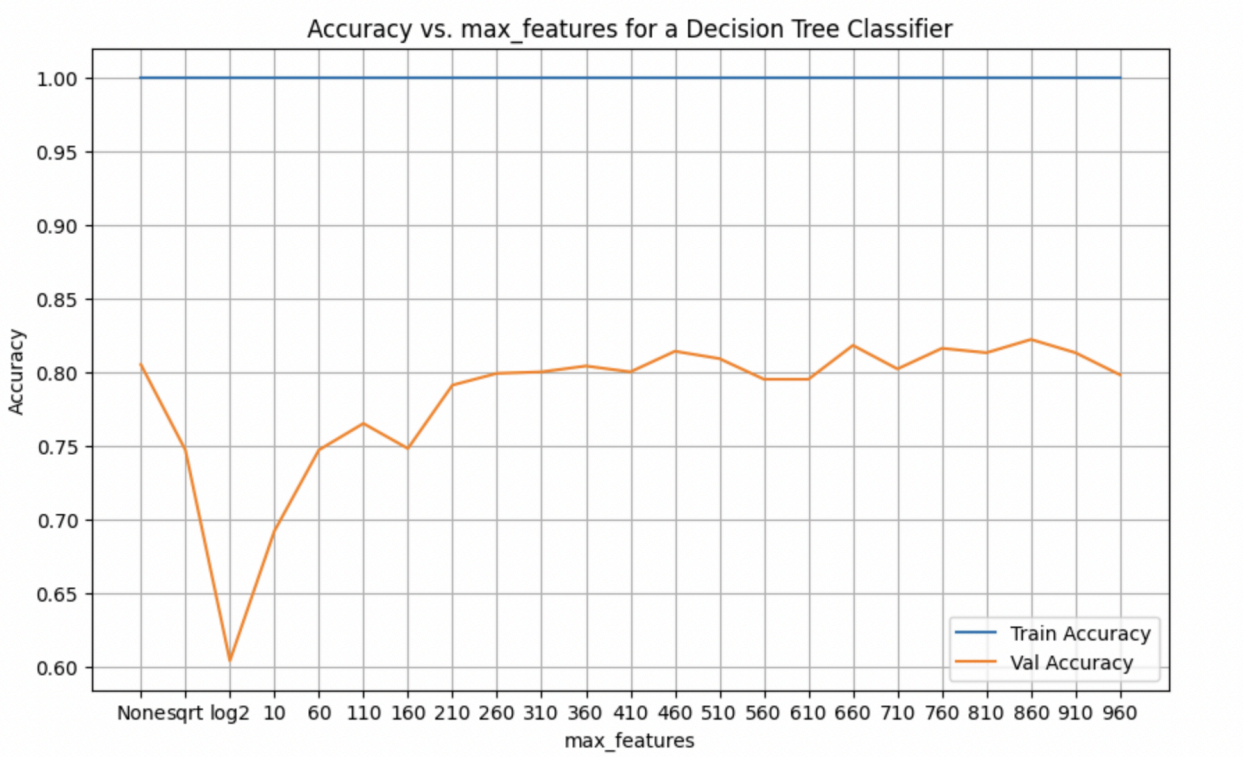
	max_features	mean_train_accuracy	std_train_accuracy	mean_val_accuracy	std_val_accuracy
0	None	1.0	0.0	0.805	0.036194
1	sqrt	1.0	0.0	0.747	0.030265
2	log2	1.0	0.0	0.604	0.025768
3	10	1.0	0.0	0.692	0.025020
4	60	1.0	0.0	0.747	0.053160
5	110	1.0	0.0	0.765	0.019235
6	160	1.0	0.0	0.748	0.019131
7	210	1.0	0.0	0.791	0.006633
8	260	1.0	0.0	0.799	0.013928
9	310	1.0	0.0	0.800	0.025495
10	360	1.0	0.0	0.804	0.026344
11	410	1.0	0.0	0.800	0.025298
12	460	1.0	0.0	0.814	0.021772
13	510	1.0	0.0	0.809	0.023749
14	560	1.0	0.0	0.795	0.024083
15	610	1.0	0.0	0.795	0.025884
16	660	1.0	0.0	0.818	0.011225
17	710	1.0	0.0	0.802	0.019900
18	760	1.0	0.0	0.816	0.011576
19	810	1.0	0.0	0.813	0.024413
20	860	1.0	0.0	0.822	0.025612
21	910	1.0	0.0	0.813	0.021354
22	960	1.0	0.0	0.798	0.031401

Visual Analysis

A line plot was created to visually compare the training and validation accuracies across the various `max_features` values. This graphical representation is instrumental in:

- Illustrating the trend in accuracy as the number of features considered at each ~split changes.
- Identifying the `max_features` value that optimizes model performance, balancing overfitting and underfitting concerns.
- Understanding the model's learning dynamics and its ability to generalize from training to unseen data.

The x-axis of the plot denotes the different `max_features` values, while the y-axis represents the corresponding accuracies, providing a clear and immediate understanding of the parameter's impact.



Insights and Conclusion

This analysis underscores the critical role of the `max_features` parameter in tuning Decision Tree classifiers. By adjusting `max_features`, one can directly influence the complexity and behavior of the model, potentially enhancing its predictive performance and generalization capability. The collected data, alongside the visual insights from the plot, offers a comprehensive overview that assists in selecting an optimal model configuration for subsequent tasks.

Building and Evaluating Random Forests Models

This analysis segment explores the impact of varying the number of trees (`n_estimators`) in Random Forest classifiers on their performance. Random Forests are

an ensemble learning method that operate by constructing multiple decision trees during training time and outputting the class that is the mode of the classes (classification) of the individual trees.

Parameter Setting

The `n_estimators` parameter, which controls the number of trees in the forest, was systematically varied in this evaluation. The tested range spanned from 10 to 200 trees, incremented in steps of 10, allowing for a detailed examination of how the number of trees affects the model's accuracy.

Cross-validation Methodology

Utilizing a 5-fold cross-validation approach ensured that each model configuration was evaluated across different subsets of the data, providing a robust estimate of the model's performance. This method partitions the dataset into five equal parts, iteratively training the model on four parts and validating it on the remaining part.

Results Compilation and Analysis

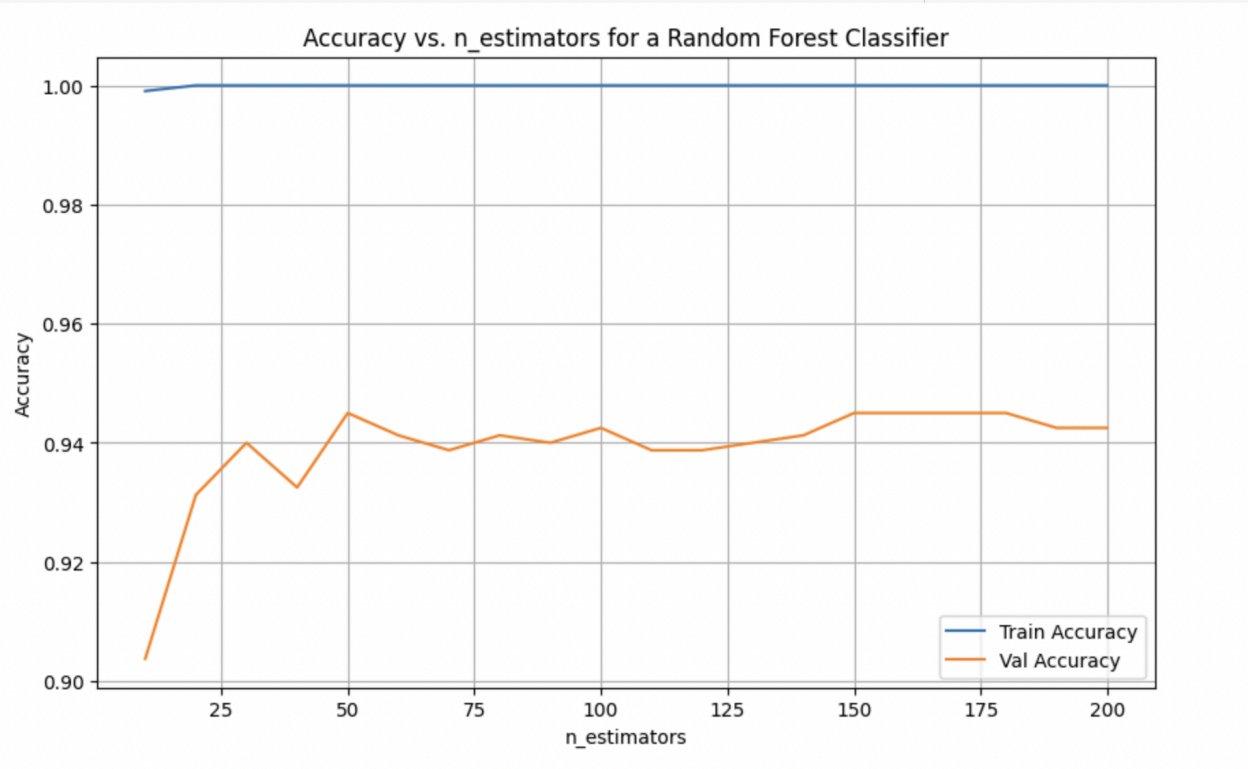
The results of the cross-validation were collated into a DataFrame, detailing the mean and standard deviation of both training and validation accuracies for each `n_estimators` value tested. This data organization allows for an easy comparison of performance metrics across different model configurations.

	n_estimators	mean_train_accuracy	std_train_accuracy	mean_val_accuracy	std_val_accuracy
0	10	0.999062	0.000765	0.90375	0.028940
1	20	1.000000	0.000000	0.93125	0.012500
2	30	1.000000	0.000000	0.94000	0.015104
3	40	1.000000	0.000000	0.93250	0.016489
4	50	1.000000	0.000000	0.94500	0.018286
5	60	1.000000	0.000000	0.94125	0.015104
6	70	1.000000	0.000000	0.93875	0.018286
7	80	1.000000	0.000000	0.94125	0.018792
8	90	1.000000	0.000000	0.94000	0.015612
9	100	1.000000	0.000000	0.94250	0.018708
10	110	1.000000	0.000000	0.93875	0.016489
11	120	1.000000	0.000000	0.93875	0.020691
12	130	1.000000	0.000000	0.94000	0.022220
13	140	1.000000	0.000000	0.94125	0.024559
14	150	1.000000	0.000000	0.94500	0.022150
15	160	1.000000	0.000000	0.94500	0.022150
16	170	1.000000	0.000000	0.94500	0.022150
17	180	1.000000	0.000000	0.94500	0.021794
18	190	1.000000	0.000000	0.94250	0.023519
19	200	1.000000	0.000000	0.94250	0.023519

Visual Representation

A line plot was employed to graphically present the relationship between the number of trees in the forest (`n_estimators`) and the corresponding accuracies (both training and validation). Key aspects of this visualization include:

- Training Accuracy Trend: Illustrates how the model's ability to learn from the data improves or stabilizes as more trees are added to the forest.
- Validation Accuracy Trend: Shows how the addition of trees affects the model's generalization to unseen data, highlighting any gains in predictive performance or potential signs of overfitting.
- Optimal Parameter Selection: Assists in identifying an `n_estimators` value that offers a favorable balance between learning capabilities and generalization, based on the convergence of training and validation accuracies.



Insights and Conclusion

The evaluation demonstrates that the number of trees (`n_estimators`) in a Random Forest significantly influences the model's performance. Increasing `n_estimators` typically enhances the model's accuracy and stability, up to a point where further additions yield diminishing returns or no significant improvement in validation accuracy. This analysis aids in selecting an appropriate `n_estimators` value that optimizes the trade-off between computational efficiency and predictive performance.

Evaluation of Random Forest Models with Varying `min_samples_leaf`

This section of the analysis delves into the effects of adjusting the `min_samples_leaf` parameter in Random Forest classifiers. The `min_samples_leaf` parameter specifies the minimum number of samples required to be at a leaf node, impacting both the depth and generalization capability of the trees within the forest.

Parameter Exploration

The evaluation spanned a range of `min_samples_leaf` values from 1 to 20, incremented by 1. This range allowed for a detailed examination of how increasing the minimum

number of samples per leaf node influences the model's learning dynamics and predictive accuracy.

Cross-validation Methodology

Employing a 5-fold cross-validation strategy ensured a robust evaluation process. This technique involves partitioning the dataset into five subsets, using each in turn for validation while training on the remaining four. This approach offers a comprehensive estimate of the model's performance and its capability to generalize across different data subsets.

Results Compilation and Analysis

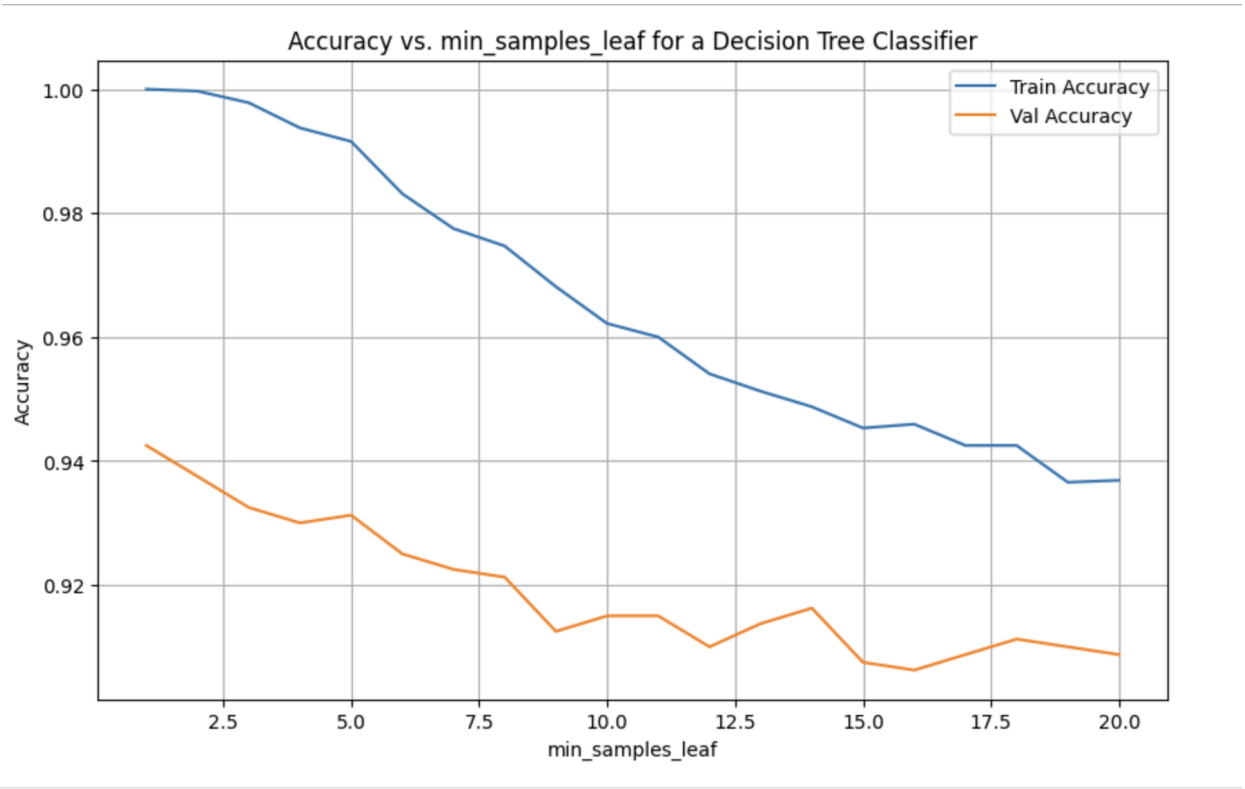
The outcomes from the cross-validation were organized into a DataFrame, presenting the mean and standard deviation for both training and validation accuracies across the tested `min_samples_leaf` values. This structured format allows for straightforward comparisons and insights into the model's performance under different configurations.

	min_samples_leaf	mean_train_accuracy	std_train_accuracy	mean_val_accuracy	std_val_accuracy
0	1	1.000000	0.000000	0.94250	0.018708
1	2	0.999687	0.000625	0.93750	0.019365
2	3	0.997812	0.001250	0.93250	0.026041
3	4	0.993750	0.001712	0.93000	0.023184
4	5	0.991562	0.001593	0.93125	0.018957
5	6	0.983125	0.003187	0.92500	0.016298
6	7	0.977500	0.004698	0.92250	0.019605
7	8	0.974687	0.003750	0.92125	0.024875
8	9	0.968125	0.005000	0.91250	0.020917
9	10	0.962187	0.007355	0.91500	0.021506
10	11	0.960000	0.002539	0.91500	0.023251
11	12	0.954063	0.002539	0.91000	0.021139
12	13	0.951250	0.004881	0.91375	0.020691
13	14	0.948750	0.005880	0.91625	0.022220
14	15	0.945312	0.008728	0.90750	0.021065
15	16	0.945937	0.005978	0.90625	0.027099
16	17	0.942500	0.004353	0.90875	0.023251
17	18	0.942500	0.007487	0.91125	0.023184
18	19	0.936562	0.008185	0.91000	0.023914
19	20	0.936875	0.007629	0.90875	0.021506

Visual Representation

A line plot was generated to visually depict the relationship between `min_samples_leaf` and model accuracies. This visualization is crucial for several reasons:

- Training Accuracy Trend: Showcases how the model's ability to fit the training data varies with the complexity controlled by `min_samples_leaf`.
- Validation Accuracy Trend: Highlights the impact of `min_samples_leaf` on the model's generalization to unseen data, indicating the potential for overfitting or underfitting at various parameter settings.
- Optimal Parameter Identification: Aids in identifying a `min_samples_leaf` value that strikes an optimal balance between learning from the training data and generalizing well to validation data.



Insights and Conclusion

The analysis demonstrates the significant role of the `min_samples_leaf` parameter in tuning the performance of Random Forest models. Adjusting this parameter can effectively control the model's complexity, enhancing its predictive accuracy and reducing the risk of overfitting. The detailed tabular data and graphical analysis provide

a clear perspective on selecting an optimal `min_samples_leaf` value for balanced and effective model performance.

Running on Test data

Preprocessing Steps:

- **Text Normalization**: The raw text data from both the training and testing datasets were normalized by converting all characters to lowercase to ensure consistency.
- **Cleaning and Tokenization**: Special characters and punctuation were removed, leaving only alphanumeric characters and spaces. The text was then tokenized into individual words.
- **Stop Word Removal:** Common English stop words were filtered out to reduce noise and focus on more meaningful words in the text.
- **Lemmatization**: Words were lemmatized to their base or dictionary form to reduce the morphological variation of words.

Feature Extraction:

- **TF-IDF Vectorization**: The preprocessed text was transformed into numerical features using TF-IDF (Term Frequency-Inverse Document Frequency) vectorization. This method was applied to both the training and testing data, with the number of max features set to 1000 to capture the most relevant terms while managing dimensionality and computational efficiency.

Model and Parameter Selection

Model Choice:

- The final model selected for making predictions was an AdaBoostClassifier with a RandomForestClassifier as its base estimator. This ensemble method leverages the strengths of both AdaBoost, known for its ability to boost the performance of decision trees on challenging classification problems, and Random Forests, recognized for their robustness and ability to handle overfitting.

Parameter Settings:

- RandomForestClassifier Parameters: `min_samples_leaf` was set to 11, and `n_estimators` to 10. These parameters were chosen based on previous

cross-validation results, aiming to balance the model's ability to generalize without overfitting.
- AdaBoostClassifier Parameters: The number of estimators (`n_estimators`) was set to 65. This setting was determined to optimize performance, enhancing the model's accuracy by adjusting the weight of incorrectly classified instances.

Model Performance

The AdaBoost model, utilizing RandomForest as the base estimator, demonstrated commendable performance on the training data. The accuracy score on the validation set was reported, reflecting the model's efficacy in classifying the documents into their respective categories accurately. The model achieved **95.5% accuracy** on the validation set.

Prediction and Output

- After training, the AdaBoost model was used to predict labels for the unseen test data, leveraging the features extracted via TF-IDF vectorization.
- The predictions were then saved to a CSV file, `labels.csv`, without an index or header, aligning with the expected format for submission or further evaluation.

Conclusion

The chosen approach effectively leverages advanced ensemble techniques, combining AdaBoost and RandomForest, to address the document classification challenge. By meticulously preprocessing the text data and judiciously selecting model parameters based on cross-validation, the methodology not only ensures robust feature representation but also optimizes the classification model's performance. The successful application of this strategy is evidenced by the accuracy achieved on the validation set and the generation of predictions for the test dataset.