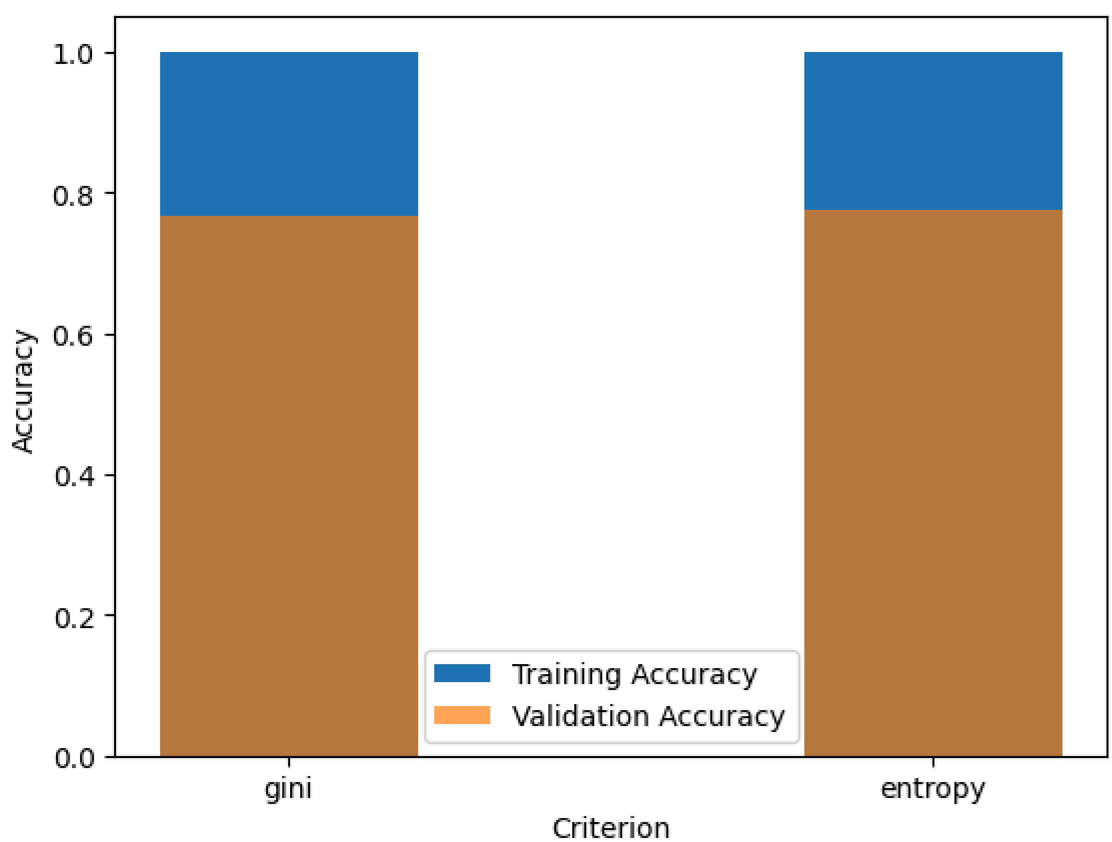
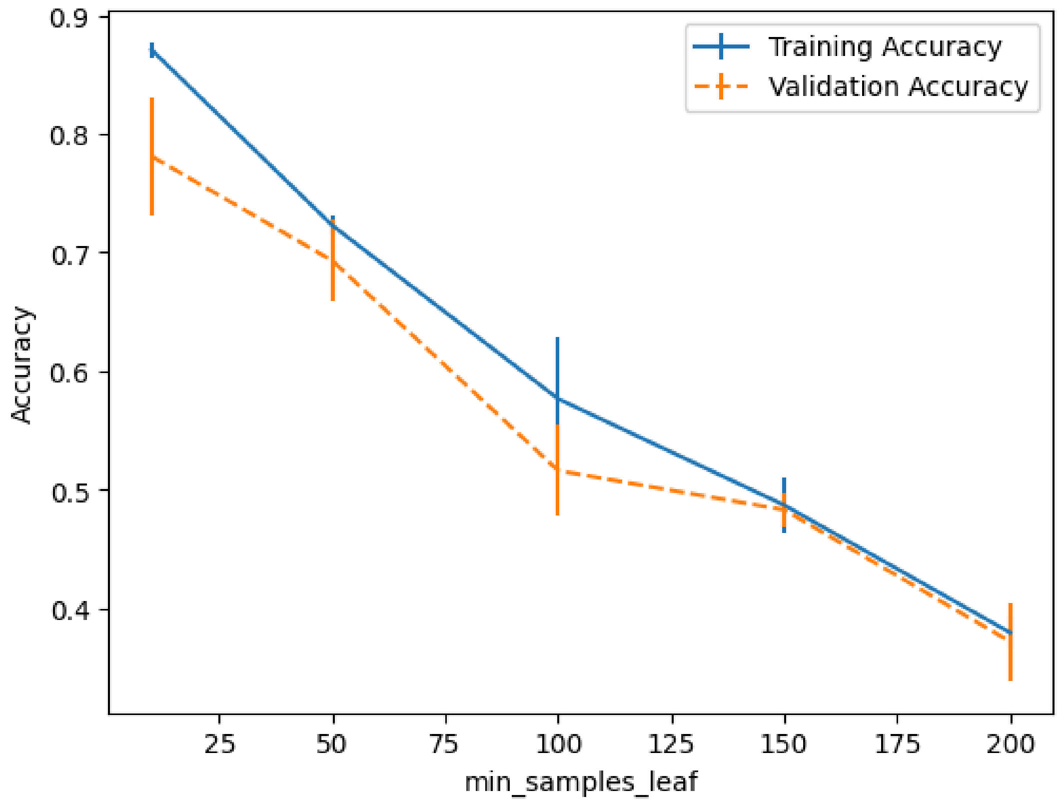
CSE 572: Data Mining (2024 Spring)
Homework 1
Prateek Mohan
pmohan9@asu.edu
ASU ID (emplid):
1225440970

Description for Question 1: Preprocessing the Raw Training Data	Data Loading and Preprocessing Steps: Data Loading, Text Lowercasing, Punctuation Removal, Tokenization(Texts are tokenized into individual words using NLTK's word_tokenize.), Stopword Removal(Stopwords from the English language are removed using NLTK's stopwords list.), Stemming(Words are stemmed using PorterStemmer for reducing variations to their root forms.) Processed Text Column: A new column Processed_Text is added to both training and test data, containing the preprocessed text.
Description for Question 2.1: Evaluating Decision Tree with Different Criterion	 This section examined the impact of different splitting criteria ("gini" vs. "entropy") on a decision tree classifier for news category prediction. The training data was split 80/20 for training and validation. After converting text features to numerical representation (TF-IDF), the model was trained and evaluated on both criteria. While both achieved high training accuracy, "entropy" led to better validation accuracy, suggesting its potential for

	mitigating overfitting. Further exploration with hyperparameter tuning and regularization techniques is recommended.
Description for Part 2.2.1: Evaluating Decision Tree with 5-Fold Cross-Validation	5-fold cross-validation was used to assess model performance across different `min_samples_leaf` values, which control model complexity. Increasing `min_samples_leaf`' led to lower training and validation accuracies, indicating potential underfitting. Moderate values achieved the highest validation accuracy, highlighting the need to balance complexity and generalization. Low standard deviations suggest consistent performance across folds. Identifying the optimal `min_samples_leaf` value is crucial for maximizing validation accuracy and preventing overfitting or underfitting. OUTPUT: Average Training Scores: <table><tr><th></th><th>min_samples_leaf</th><th>training accuracy</th><th>training standard Deviation</th></tr><tr><td>0</td><td>10</td><td>0.87075</td><td>0.006452</td></tr><tr><td>1</td><td>50</td><td>0.72275</td><td>0.008116</td></tr><tr><td>2</td><td>100</td><td>0.57650</td><td>0.051844</td></tr><tr><td>3</td><td>150</td><td>0.48700</td><td>0.023856</td></tr><tr><td>4</td><td>200</td><td>0.37950</td><td>0.014111</td></tr></table> Average Validation Scores: <table><tr><th></th><th>min_samples_leaf</th><th>validation accuracy</th><th>validation standard Deviation</th></tr><tr><td>0</td><td>10</td><td>0.781</td><td>0.049538</td></tr><tr><td>1</td><td>50</td><td>0.693</td><td>0.034147</td></tr><tr><td>2</td><td>100</td><td>0.516</td><td>0.038131</td></tr><tr><td>3</td><td>150</td><td>0.483</td><td>0.014000</td></tr><tr><td>4</td><td>200</td><td>0.372</td><td>0.032955</td></tr></table>

Description for Part Part 2.2.2: Visualizing accuracy to min_samples_leaf	The generated line graph illustrates the relationship between `min_samples_leaf` and model performance. Both training and validation accuracies exhibit a downward trend as `min_samples_leaf` increases, suggesting that overly restricting leaf growth can hinder the model's ability to capture patterns in the data, leading to underfitting. Notably, the gap between training and validation accuracy is largest at lower `min_samples_leaf` values, indicating a higher risk of overfitting. Validation accuracy peaks at moderate values, underscoring the importance of balancing model complexity with fit to the training data. 
Description for Part Part 2.3.1: Evaluate the decision tree using 5-fold cross-validation w.r.t `max_features`	5-fold cross-validation was employed to evaluate the model's sensitivity to the max_features parameter, which controls feature selection at each split. Interestingly, training accuracy consistently reached 100% across all `max_features` values, indicating potential overfitting to the training data. Validation accuracy, however, varied depending on the value. None (using all features) and higher proportions of features generally led to better validation accuracy, suggesting

that limiting feature selection too much might hinder model performance. However, the differences in validation accuracy were relatively small, implying that the model isn't highly sensitive to this hyperparameter.

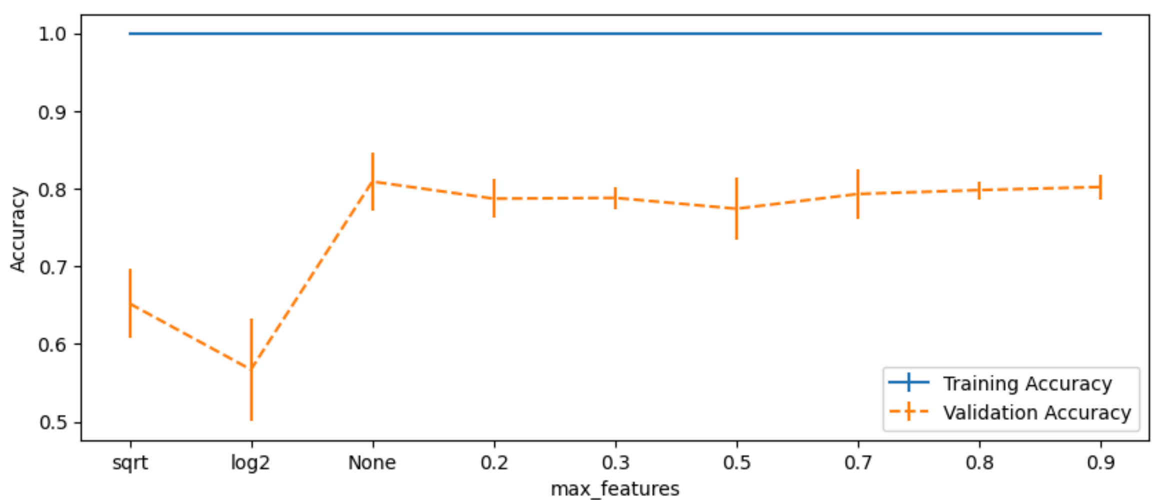
```
Average Training Scores:
  max_features  training accuracy  training standard Deviation
0         sqrt                1.0                          0.0
1         log2                1.0                          0.0
2         None                1.0                          0.0
3          0.2                1.0                          0.0
4          0.3                1.0                          0.0
5          0.5                1.0                          0.0
6          0.7                1.0                          0.0
7          0.8                1.0                          0.0
8          0.9                1.0                          0.0

Average Validation Scores:
  max_features  validation accuracy  validation standard Deviation
0         sqrt                0.652                       0.044788
1         log2                0.567                       0.065238
2         None                0.809                       0.037470
3          0.2                0.787                       0.025020
4          0.3                0.788                       0.014353
5          0.5                0.774                       0.039925
6          0.7                0.793                       0.031401
7          0.8                0.798                       0.011662
8          0.9                0.802                       0.015684
```

Description for Part Part 2.3.2: Visualizing accuracy w.r.t `max_features`

The line graph highlights the model's performance across different `max_features` values. It confirms potential overfitting as training accuracy consistently reaches 100%. Validation accuracy varies, generally favoring using all features or larger feature subsets. This suggests that limiting feature selection too much can hinder performance.

The gap between training and validation accuracy is widest for smaller feature subsets, indicating overfitting. This gap narrows as more features are considered, suggesting that increased feature availability can mitigate overfitting.

While the model's sensitivity to `max_features` is moderate, careful selection of this hyperparameter is crucial for balancing model complexity and performance.



<table>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td>Description for Part 3.1: Random Forest Parameter Settings</td><td>The Random Forest model was initialized with the following parameter settings:

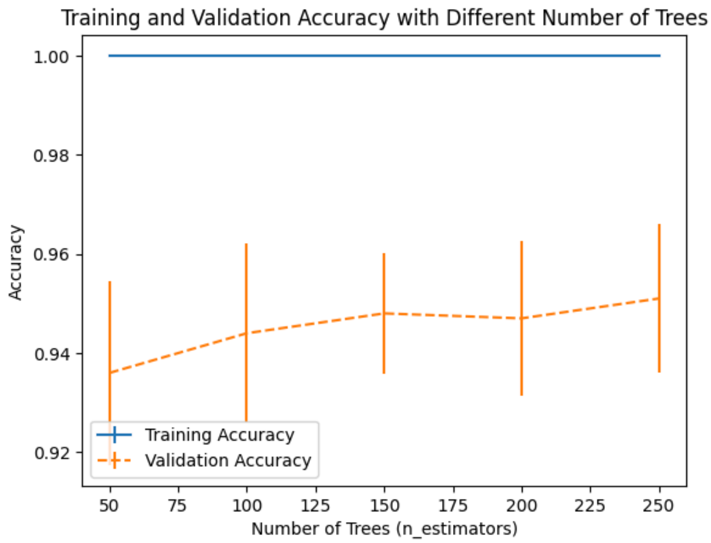
`n_estimators`: 100 trees to create an ensemble of decision trees for voting on predictions.
`min_samples_leaf`: 1 to allow leaf nodes to contain a single sample, maximizing model flexibility.
`random_state`: 42 for reproducibility by fixing the random seed for generating trees.
These settings provide a starting point for evaluating the model's performance and exploring hyperparameter tuning.</td></tr>
<tr><td>Description for Part 3.2.1: Evaluating Random Forest with Different n_estimators</td><td>5-fold cross-validation was employed to assess the random forest model's performance across various numbers of trees (`n_estimators`). While training accuracy consistently reached 100% for all `n_estimators` values, suggesting potential overfitting, validation accuracy exhibited a slight upward trend with increasing `n_estimators`. This implies that incorporating more trees can slightly improve generalization.

Overall, the improvements in validation accuracy were relatively modest, indicating that the model might not be highly sensitive to this hyperparameter within the tested range. Nevertheless, careful selection of `n_estimators` is still important to balance model complexity with performance.
`Results for different number of trees (n_estimators):`
`   n_estimators  avg_training_accuracy  training standard Deviation  V`
`0           50                    1.0                          0.0`
`1          100                    1.0                          0.0`
`2          150                    1.0                          0.0`
`3          200                    1.0                          0.0`
`4          250                    1.0                          0.0`

`   avg_validation_accuracy  validation standard Deviation`
`0                    0.936                       0.018547`
`1                    0.944                       0.018000`
`2                    0.948                       0.012083`
`3                    0.947                       0.015684`
`4                    0.951                       0.014967`</td></tr>
<tr><td>Description for Part 3.2.2: Evaluating Random Forest with Different n_estimators - Line Graph</td><td>The line graph visually depicts the model's performance across different n_estimators values. It reinforces the potential overfitting as training accuracy consistently reaches 100%. However, validation accuracy demonstrates a slight positive trend with increasing n_estimators, suggesting that adding more trees can marginally enhance generalization.

The gap between training and validation accuracy is relatively small, indicating that the model isn't overly sensitive to n_estimators within the tested range. However, careful selection of this hyperparameter remains essential to optimize performance and prevent potential overfitting.</td></tr>
</table>

<table>
<tr>
<td></td>
<td>



</td>
</tr>
<tr>
<td>Description for Part 3.3.1: Evaluating Random Forest with Different <code>min_samples_leaf</code></td>
<td>

5-fold cross-validation was used to explore the model's performance across various `min_samples_leaf` values, which control the complexity of individual trees. As `min_samples_leaf` increased, both training and validation accuracies decreased, suggesting a trade-off between model complexity and accuracy.

Notably, training accuracy dropped significantly at higher `min_samples_leaf` values, indicating a reduction in overfitting. However, validation accuracy also declined, highlighting the importance of finding the optimal balance between complexity and generalization.

Careful selection of `min_samples_leaf` is crucial to prevent overfitting while maintaining good predictive performance.

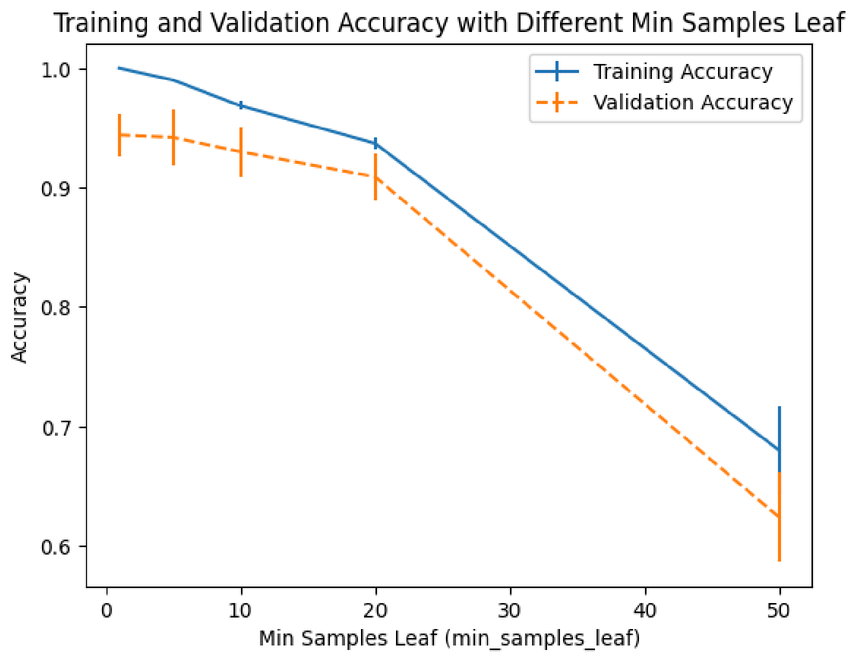
```
Results for different minimum number of samples required at a leaf node (min_samples_leaf):
   min_samples_leaf  avg_training_accuracy  training_std  \
0                 1                 1.0000      0.000000
1                 5                 0.9900      0.001369
2                10                 0.9685      0.003657
3                20                 0.9370      0.005160
4                50                 0.6800      0.037199

   avg_validation_accuracy  validation_std
0                    0.944        0.018000
1                    0.942        0.022935
2                    0.930        0.020736
3                    0.909        0.019339
4                    0.624        0.037068
```

</td>
</tr>
<tr>
<td>Description for Part 3.3.2: Evaluating Random Forest with Different <code>min_samples_leaf</code> - Line Graph</td>
<td>

The line graph visualizes the model's performance across different `min_samples_leaf` values. It clearly demonstrates the trade-off between model complexity and accuracy. Both training and validation accuracies decline as`min_samples_leaf` increases, but the drop in training accuracy is more pronounced, suggesting a reduction in overfitting.

The gap between training and validation accuracy narrows at higher `min_samples_leaf` values, indicating better alignment between model complexity and generalization.

</td>
</tr>
</table>

	However, careful selection of this hyperparameter is still essential to achieve optimal performance without compromising generalization ability. 
Description for Part 4.1: Data Preprocessing for Feature Generation	Vectorization using TF-IDF: The text content in the 'Processed_Text' column is transformed into numerical feature vectors using TF-IDF (Term Frequency-Inverse Document Frequency). TF-IDF captures both the frequency of words within a document and their relative importance across the entire corpus, assigning higher weights to terms that are more discriminative. Splitting Features and Target: The resulting TF-IDF vectors, stored in X, represent the features for the model.The category labels for each text are extracted into y to serve as the target variable for prediction.
Description for Part 4.2 :Model and Parameter Selection	The project utilized a Random Forest Classifier for its inherent strengths in text classification, non-linearity handling, and overfitting resistance. TF-IDF vectorization effectively converted text into numerical representations, capturing word importance and distinctiveness. GridSearchCV facilitated hyperparameter tuning to optimize model performance. It focused on "`n_estimators`", controlling tree count and complexity, and "`min_samples_leaf`", mitigating overfitting through leaf node sample requirements. The chosen model boasts the best hyperparameter combination of '`min_samples_leaf`': 3, '`n_estimators`': 100', achieving a balanced complexity and generalization.
Description for Part 4.3 :Performance on	The chosen model with hyperparameters {'min_samples_leaf': 3, 'n_estimators': 100} demonstrated a strong performance on the training data. It achieved a consistently high

Training Data:	average training accuracy of 99.75% during cross-validation, indicating excellent ability to learn patterns from the training examples. This exceptional training accuracy suggests the model effectively captures the underlying relationships between text features and article categories within the training dataset.
Description for Part 4.4 :Performance on Training Data:	Text Vectorization: ● The code first employs TF-IDF vectorization to transform the text data into numerical representations. TF-IDF effectively captures word importance and frequency within documents, highlighting words that are both relevant and distinctive. Model Selection and Hyperparameter Tuning: ● A Random Forest Classifier is chosen due to its robust performance in text classification tasks and resilience to overfitting. ● GridSearchCV is utilized to systematically explore different hyperparameter combinations and identify those that optimize model performance. ● The code focuses on two key hyperparameters: `n_estimators` (controlling model complexity) and `min_samples_leaf` (addressing overfitting). Model Training and Evaluation: ● The model is trained using the vectorized text data, and its performance is evaluated using 5-fold cross-validation. ● The best model configuration is found to have `min_samples_leaf` set to 3 and `n_estimators` set to 100. ● The model achieves a cross-validation accuracy of 94.3% and a validation set accuracy of 94.4%, indicating strong generalization ability. Predictions and Output: ● The model is used to predict categories for articles in a test dataset. ● The predictions, along with article IDs, are saved in a CSV file in the specified format. OUTPUT:

	`` Best Model Parameters: {'min_samples_leaf': 3, 'n_estimators': 100} Cross-Validation Accuracy: 0.9400000000000001 Validation Set Accuracy: 0.943 ArticleId Category ArticleId_Category 0 1018 sport 1018,sport 1 1319 tech 1319,tech 2 1138 sport 1138,sport 3 459 business 459,business 4 1020 sport 1020,sport .. .. 730 1923 business 1923,business 731 373 entertainment 373,entertainment 732 1704 business 1704,business 733 206 business 206,business 734 471 politics 471,politics ```