Final Presentation

AI vs Human Content Detection

Group 2 Ava Leon and Sofia Curi

Project Overview

Topic Goal

AI vs human content detection. The goal is to explore and analyze the differences between text written by a human and text generated by artificial intelligence.

df.head()

We loaded the dataset and used **df.head()** to show the first rows of the data, to give a quick look at what's inside.

Part 1 Dataset Loaded

We started bringing all the tools (libraries) we will need for the project

- pandas **for** working with data
- matplotlib and seaborn **for** making charts
- numpy **for** math operations

Part 2 Initial Exploration

Summary of the dataset

- Total number of **rows** (1367) and **columns** (17) in the data.
- The **Non-Null Count** shows how many of the rows have a value for that column.
- In columns like **sentiment_score** (1313), the number is *less than 1367* because it indicates there are **missing values** in these columns.
 * *key insight for the data cleaning step in Part 8*
- **Dtype** tells what kind of data is in each column (integers (numbers), floats, text (object)).
- Shows which columns have missing pieces of info. **flesch_reading_ease** has 79 missing values.

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1367 entries, 0 to 1366
Data columns (total 17 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   text_content          1367 non-null   object
 1   content_type          1367 non-null   object
 2   word_count            1367 non-null   int64
 3   character_count       1367 non-null   int64
 4   sentence_count        1367 non-null   int64
 5   lexical_diversity     1367 non-null   float64
 6   avg_sentence_length   1367 non-null   float64
 7   avg_word_length       1367 non-null   float64
 8   punctuation_ratio     1367 non-null   float64
 9   flesch_reading_ease   1288 non-null   float64
 10  gunning_fog_index     1332 non-null   float64
 11  grammar_errors        1367 non-null   int64
 12  passive_voice_ratio   1336 non-null   float64
 13  predictability_score  1367 non-null   float64
 14  burstiness            1367 non-null   float64
 15  sentiment_score       1313 non-null   float64
 16  label                 1367 non-null   int64
dtypes: float64(10), int64(5), object(2)
memory usage: 181.7+ KB
None
```

Part 2 Initial Exploration

Summary of the dataset

- **mean: *average value*** for the column (**word_count = 140**, meaning the average text in the dataset has like 140 words).
- **std**: is the ***standard deviation***, tells how spread out the data is. If it is large then it means the values are very spread out (std for **word_count = 97.4** is large, which shows the word counts *vary a lot* between the different texts).
- **min**: is the ***smallest value*** word_count = 3, so the shortest text has only three words.
- **max**: is the ***largest value*** in the column max.
- The distance between the **25% and 75%** values is the **(IQR)**, which tells how spread out the middle of the data is.

```
Dataset Description:
        word_count  c
count  1367.000000
mean    140.190929
std      97.410218
min       3.000000
25%      61.500000
50%     131.000000
75%     193.000000
max     443.000000
```

Part 4 & 5 Dataset Description & Questions

Brief

- The dataset is about ***classifying the detection*** between AI vs. human generated text content.

- word_count is the total number of words
- label tells if the content is human (0) or AI-generated (1).

QUESTION 1

Can we tell if a text is AI generated or a human written by looking at things like word choice, sentence length, and how the ideas are organized (features)?

QUESTION 2

How is the writing style of a AI generated text different from human written? - lexical_diversity and punctuation_ratio

QUESTION 3

Is there a relationship between sentiment (sentiment_score) and content type (content_type)?

Part 6 Basic Visualizations

- Show how often different values appear in a column *(ranges and frequency)*

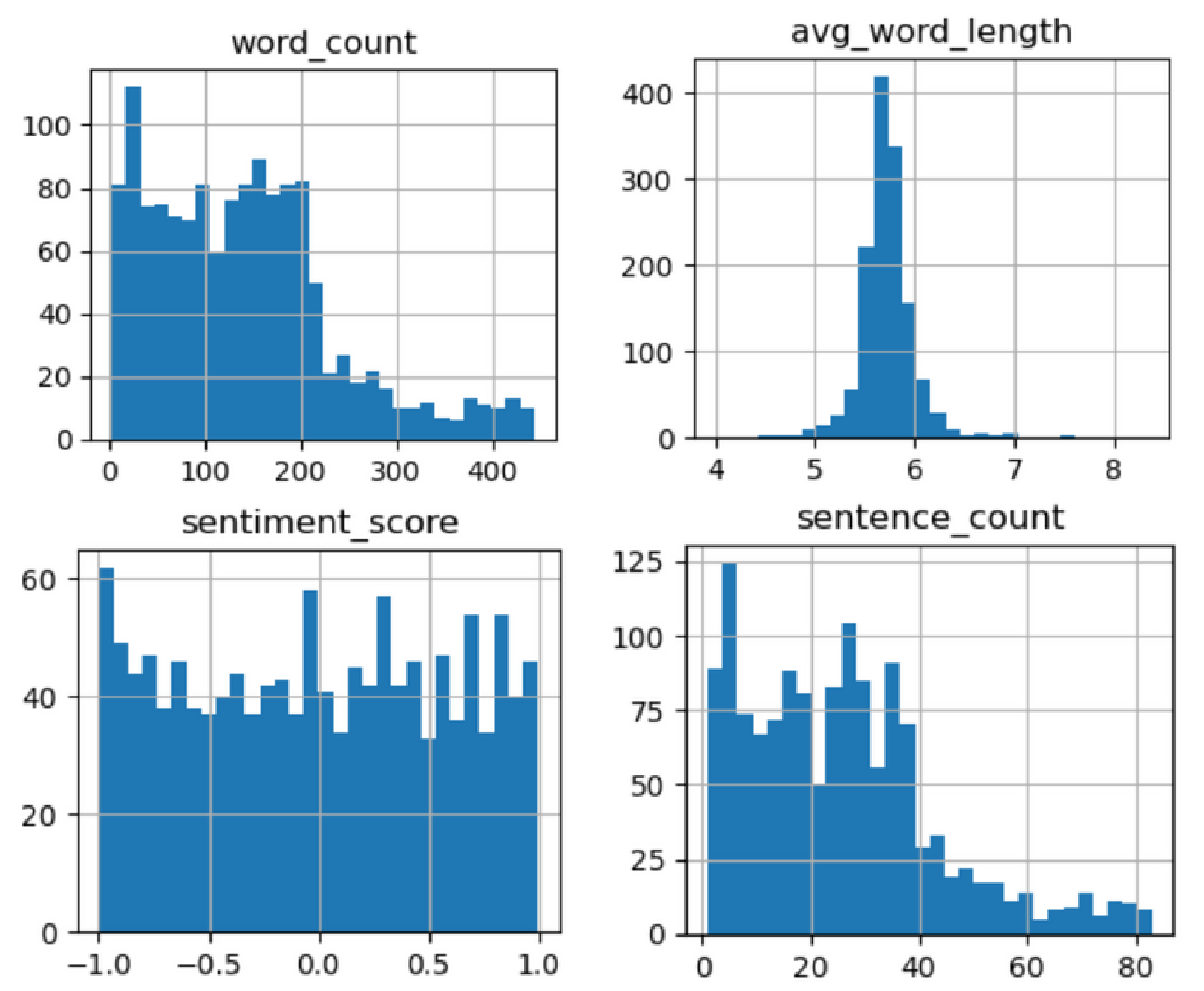
- The histogram **word_count** shows
1. the number of times a word count appears
2. how many texts are into a specific range of word counts.

- The histogram for **avg_word_lenght** shows
1. the distribution for all the texts, to understand their writing style.
2. If the bars are high, there is a consistent writing style.

** It helped us identify if the writers use many short words or a mix of shorter and longer words.*

- The histogram for **sentiment_score** shows a ranges from -1 (negative) to 1 (positive), to understand the "*emotional tone*" of the content.

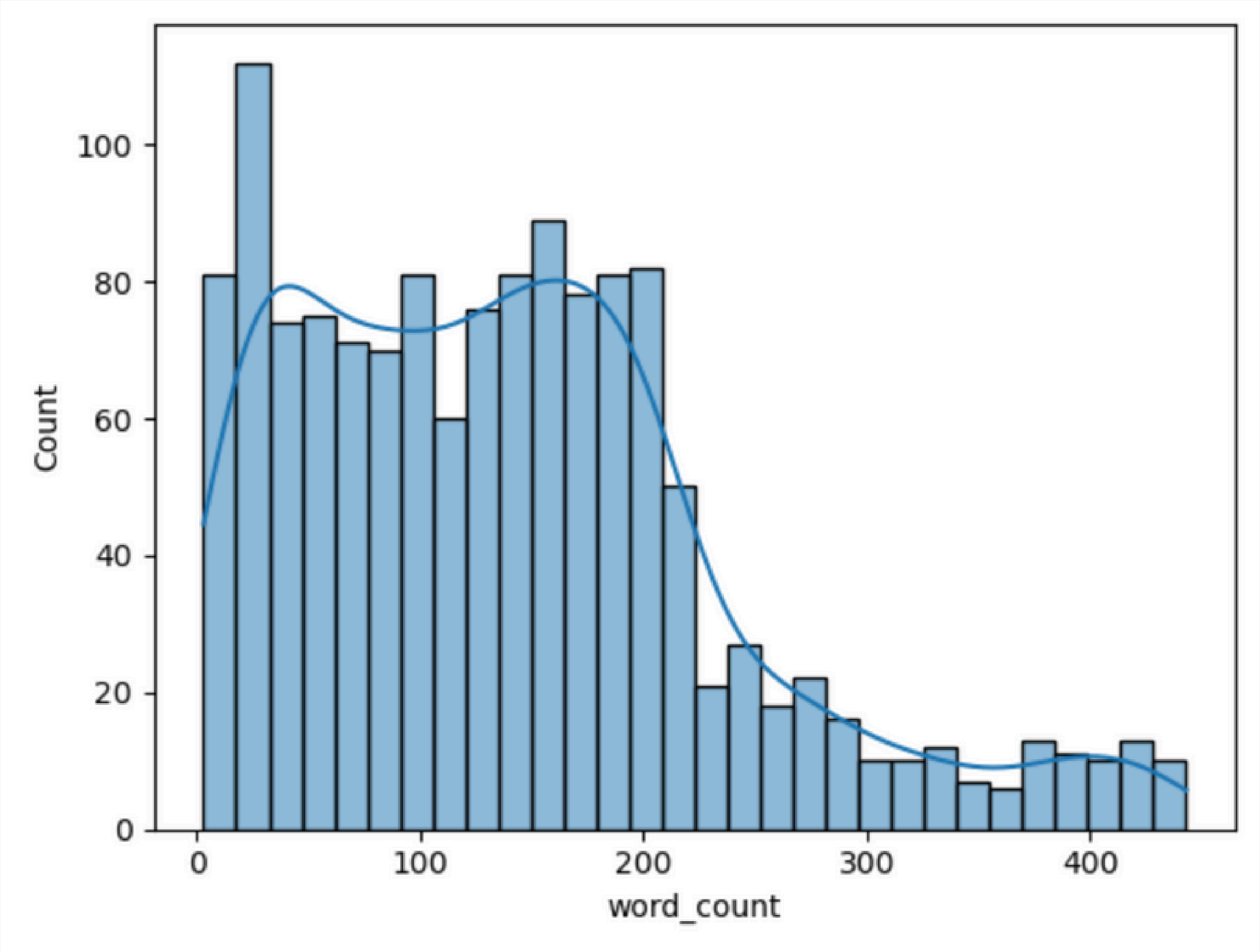
HISTOGRAM TO VISUALLY SHOW THE DATA



Part 6 Basic Visualizations

- Show how often different values appear in a column
- In the plot, we showed the relationship between a text's word count and its count
- In **word_count** the numbers range from about **0 to 450**, which matches the min and max word counts we saw in **df.describe()**.
- The thin line shows the “general trend”

PLOT



PART 7 INTERPRETATION OF VISUALIZATIONS

- Most histograms are right skewed, with short low values
- There are outliers in the right distributions (like in word_count)
- The long parts extended to the right show that a few texts that are longer than the rest.
- Most texts are short (short sentences, words, and characters)
- There is AI vs Human distinction (The predictability_score tell the difference between AI written and human written text).

Part 8 Data Cleaning

Cleaned what we found in Part 2 about some missing values, so that it can be used for modeling.

- To go through every value in the *DataFrame* and make a simple list that shows the total number of missing values for each column in the dataset.
- In the line **flesch_reading_ease** there are 79 missing values.
- We used this code to automatically ***find outliers*** in the numerical data using the **IQR**.

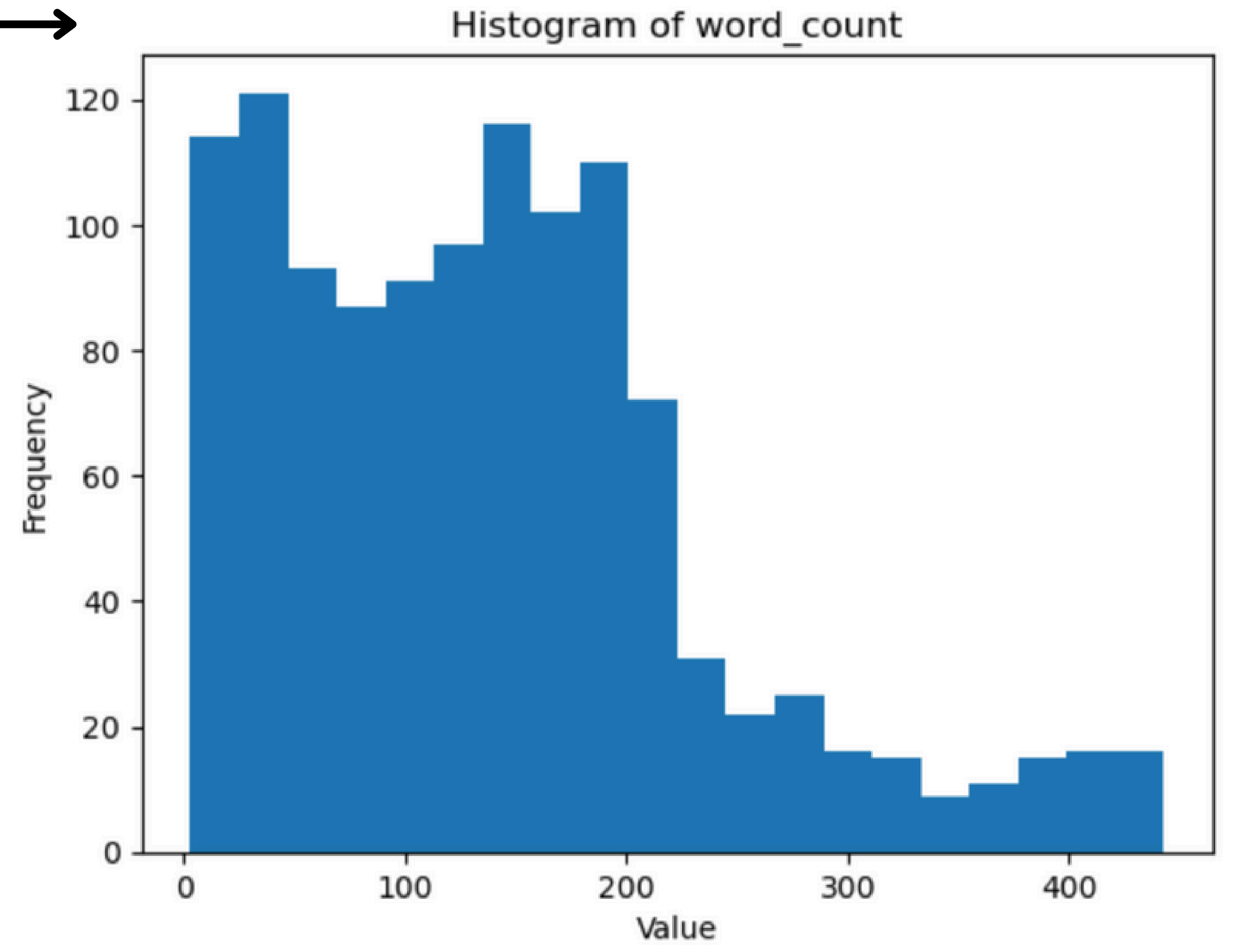
```
Missing values per column:
text_content              0
content_type              0
word_count                0
character_count           0
sentence_count            0
lexical_diversity         0
avg_sentence_length       0
avg_word_length           0
punctuation_ratio         0
flesch_reading_ease      79
gunning_fog_index        35
grammar_errors            0
passive_voice_ratio      31
predictability_score      0
burstiness                0
sentiment_score          54
label                     0
dtype: int64
```

```
Column: word_count
Lower limit: -140.5
Upper limit: 391.5
Number of outliers: 40
Number of outliers: 40
```

Part 9 Visualization

We created a histogram to visualize the distribution of word_count in the dataset.

- Typical length of texts in the dataset, the tallest bars in the histogram show the most common word count ranges.
- See if the texts are all similar in length or if there's a variety of short, medium, and long texts.
- Spot extreme values that might be outliers.
- Help us into the insight of the data before building a machine learning model to distinguish between AI and human content.




```
0    288
1    253
2    420
5    198
6     84
Name: word_count, dtype: int64
```

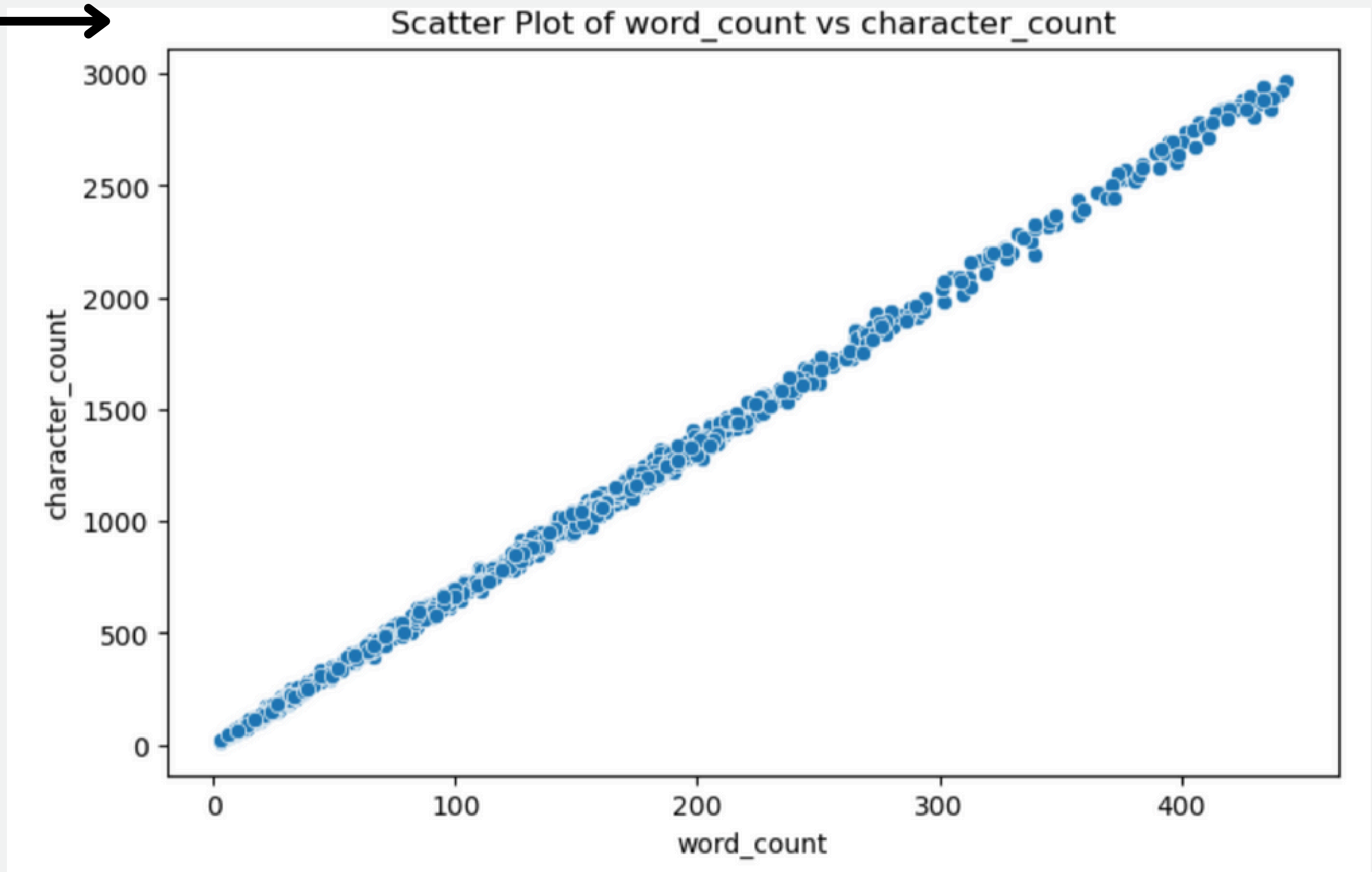
Part 9 Visualization

Scatter Plot

- The scatter plot of **word_count** vs **character_count** helps us visualize the relationship between them.

- Which we expected to be a strong one, because longer texts almost always have a higher character count. The *scatter plot* visually confirms this.

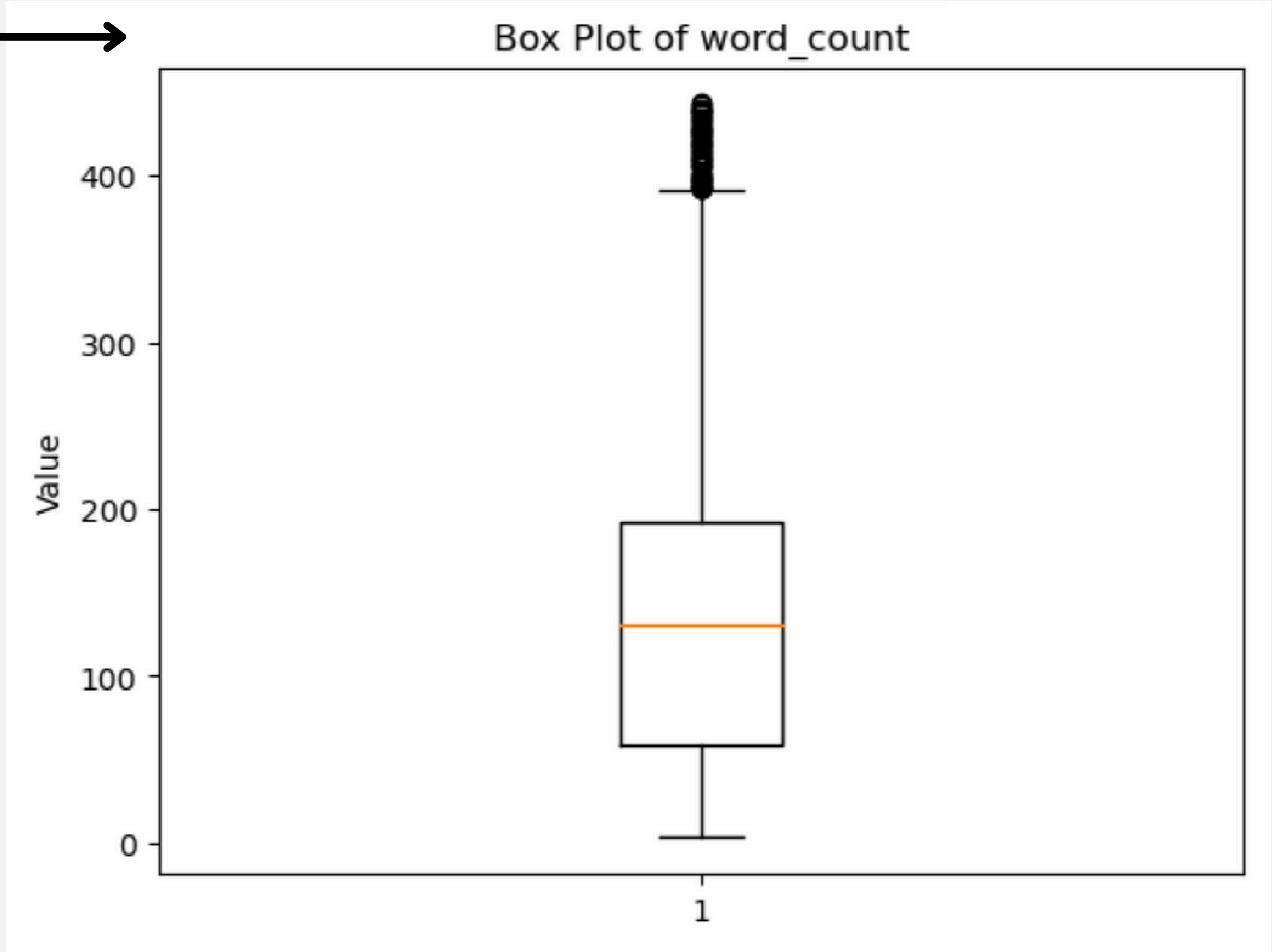
- It also helps us *spot inconsistencies*: If we saw points with a high word count but a very low character count, it could mean an error or an outlier that we missed.



Part 9 Visualization

Box Plot

- The box plot to understand the key features of word_count.
- The line in the middle of the box shows the **median**, which is the middle value of the data.
- This tells us the typical **word_count** in the dataset.
- The size of the box shows the **(IQR)**, representing the middle of the data.
- We did this to identify the central point, spread, and extreme values.



Part 9 Visualization

To create a New Feature

- Creates a new column called **words_per_sentence**.
- Calculates the average number of words per sentence for each text by dividing the word_count by the sentence_count.
- To distinguish between different writing styles.

```
# Example: Create a new feature
df["words_per_sentence"] = df["word_count"] / df["sentence_count"]
```

```
# Example: Transform a feature
df["log_word_count"] = np.log1p(df["word_count"])

print(df[["word_count", "log_word_count"]].head())
```

Transforming an Existing Feature

- Creates another new column called **log_word_count.**
- Takes the **word_count** values and applies the logarithm math function.
- The table shows the original and the newly created column.
- For each row, you can see how the **word_count** is converted into the transformed **log_word_count**

	word_count	log_word_count
0	288	5.666427
1	253	5.537334
2	420	6.042633
5	198	5.293305
6	84	4.442651

Part 11 Observations and Findings

This part is where we explained the process and summarized the findings from the previous step.

DATA CLEANING

- Removed rows with missing values.
- Removed duplicate rows.
- Checked for outliers using the IQR method (none found).

DATA VISUALIZATION

- Histogram: showed word count distribution.
- Scatter plot: explored relationships between columns.
- Box plot: displayed range, median, and potential outliers.

FEATURE ENGINEERING

- Created a feature for average words per sentence.
- Applied log math function to transform the word count and have an easier analysis.

Part 12 Data Processing

Prepared the data for a machine learning model

Using **TF-IDF** (Term Frequency-Inverse Document Frequency) method.

- Counts how often a word appears in a document and also gives more importance to words that are unique in a document.
- max_features=5000 considers the 5,000 most common words, which helps keep the data manageable.
- stop_words="english" ignores common English words like "the," "a" and "is" because they are not very useful for distinguishing between different types of content.

This is where we predict whether a text is AI or human written.

Part 13 Applied to a Machine Learning Model

Classification Report

Gives a detailed breakdown of the model's performance for each class (0 for human and 1 for AI).

It includes:

- **Precision:** Out of all the times the model predicted a class ("AI"), how many times was it correct?
- **Recall:** Out of all the actual examples of a class ("human"), how many of them did the model correctly identify?
- **F1-Score:** Combines precision and recall, and gives an overall measure of performance.
- **Support:** The number of actual examples of that class in the test data.

```
Classification Report (Logistic Regression):

              precision    recall  f1-score   support

           0       0.49      0.56      0.52       133
           1       0.52      0.45      0.48       141

    accuracy                           0.50       274
   macro avg       0.51      0.51      0.50       274
weighted avg       0.51      0.50      0.50       274

Confusion Matrix:

[[74 59]
 [77 64]]
```

Gives a summary of the predictions.

- Top left number is **True Positives**: How many times the model predicted correctly "AI".
- The bottom right number is **True Negatives**: How many times the model predicted correctly "human".
- The other two numbers show the **False Positives and Negatives** where the model was wrong.

Part 13 Applied to a Machine Learning Model

Random Forest Classifier

Predicts whether a text is AI or human-written.

- **precision:**
  - For class 0 (Human): 0.49. means that when the model predicted a text was human, it was only correct about 49% of the time.
- **recall:**
  - For class 1 (AI)
- **f1-score:** Combination of precision and recall.
- **support:** Number of examples in the test data for each class. There were 133 human written texts and 141 AI generated texts.
- **accuracy:** 0.50. means the model was correct 50% of the time.

Classification Report (Random Forest):

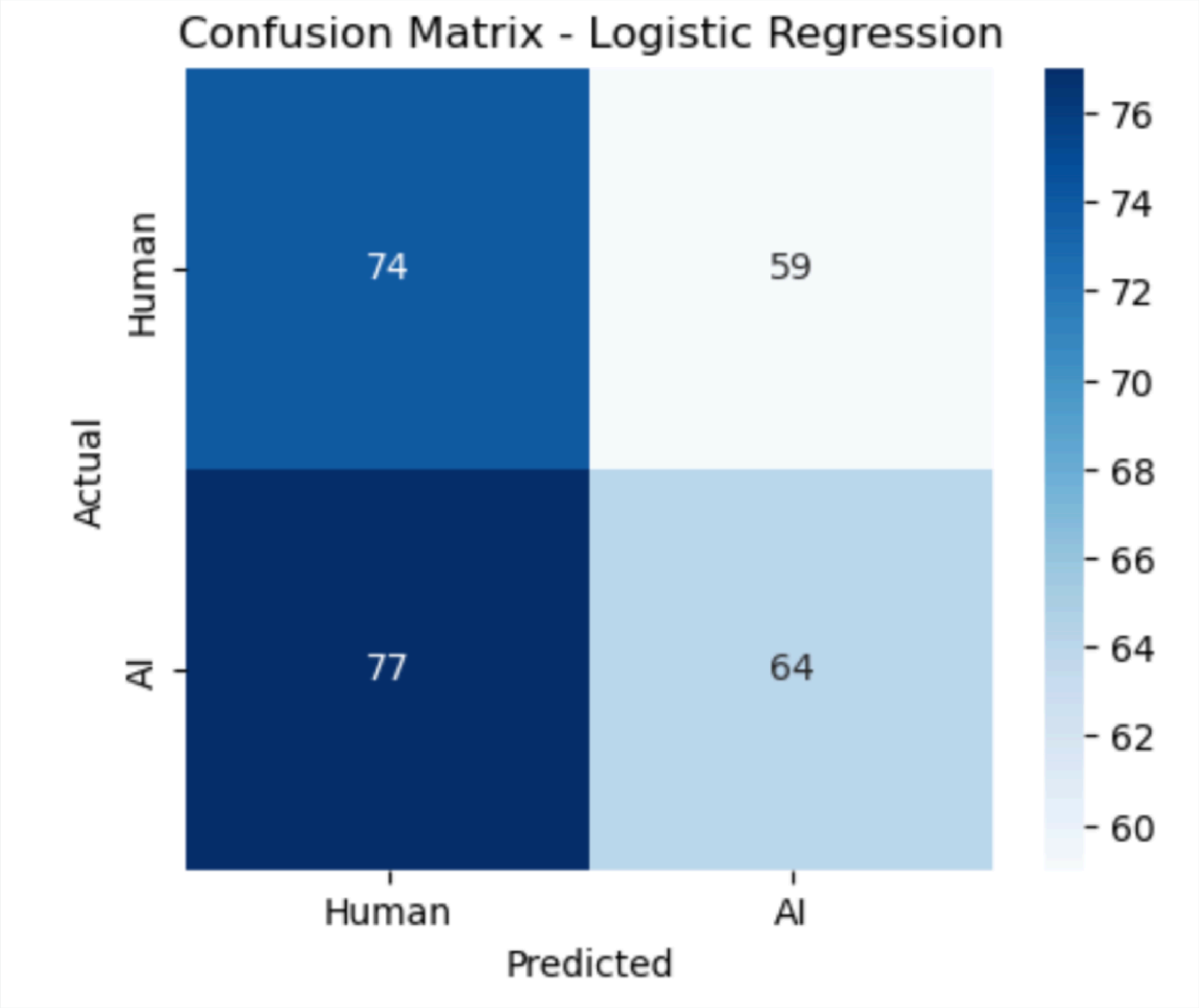
	precision	recall	f1-score	support
0	0.49	0.56	0.52	133
1	0.52	0.44	0.48	141
accuracy			0.50	274
macro avg	0.50	0.50	0.50	274
weighted avg	0.50	0.50	0.50	274

Part 13 Applied to a Machine Learning Model

Confusion Matrix

Visualizes it as a heatmap, a way to understand the
It shows the model's predictions into four categories:

Predicted: Human	**Predicted: AI**
Correct Predictions (True negative)	Incorrect Predictions (False negative)
Incorrect Predictions (False negative)	Correct Predictions (True positive)



We can see how many times it correctly identified a human written text and how many it labeled incorrectly an AI generated text as human.

Part 13 Applied to a Machine Learning Model

The ROC Curve (Receiver Operating Characteristic Curve)

Calculates the AUC (Area Under the Curve) to evaluate the performance of the model.

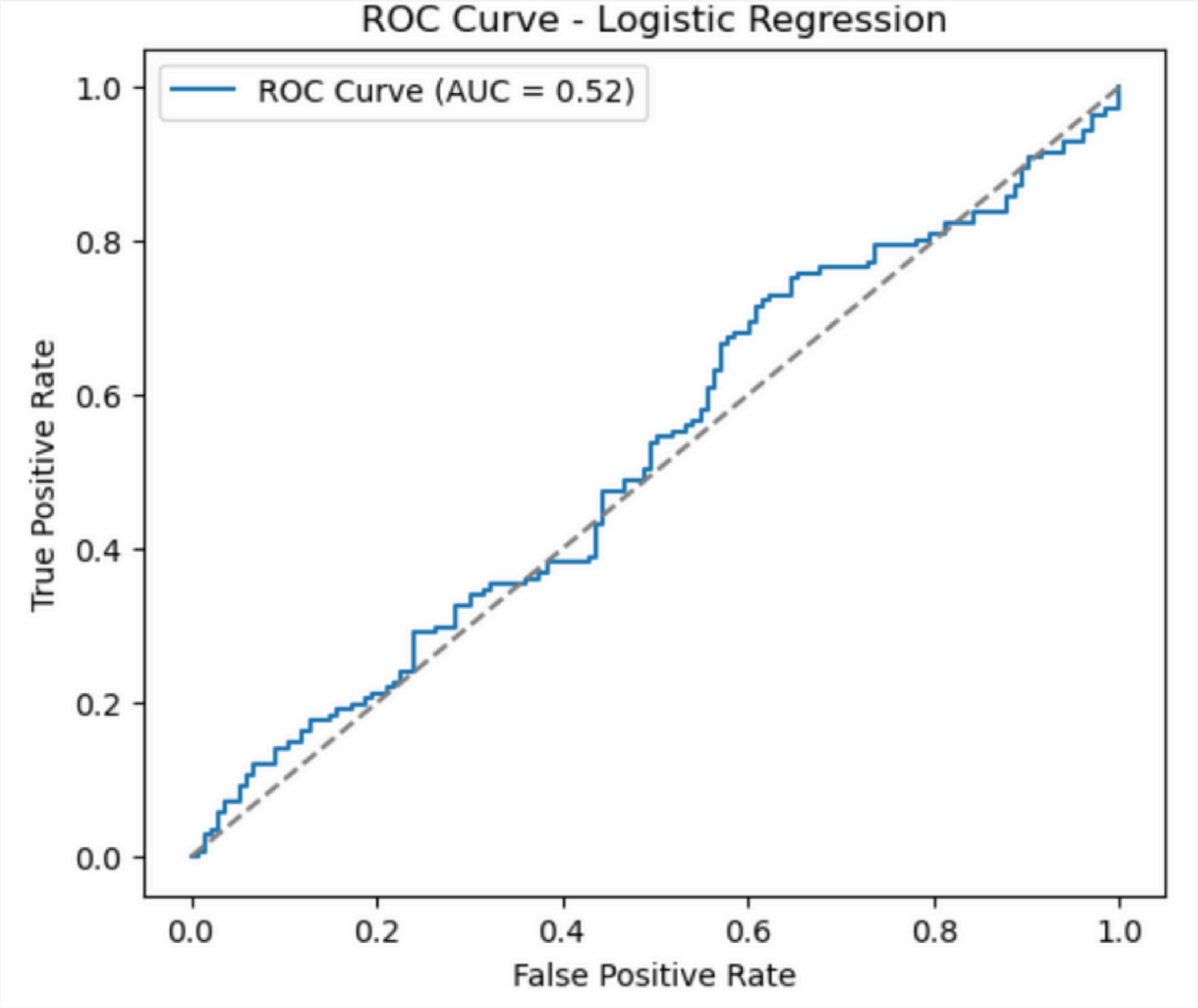
- Another way to see how well the model can distinguish between the two Human vs. AI.

The plot visualizes between the **True Positive** and the **False Positive** Rate.

- A visual way to see if the model is a good classifier. *The higher the blue curve is above the gray line, the better the model is.*

The **blue curve** shows the model's performance. The closer it is to the top left corner, the better the model is at distinguishing the two classes.

The **dotted grey** line represents a random guess.



Part 13 Applied to a Machine Learning Model

Code Block

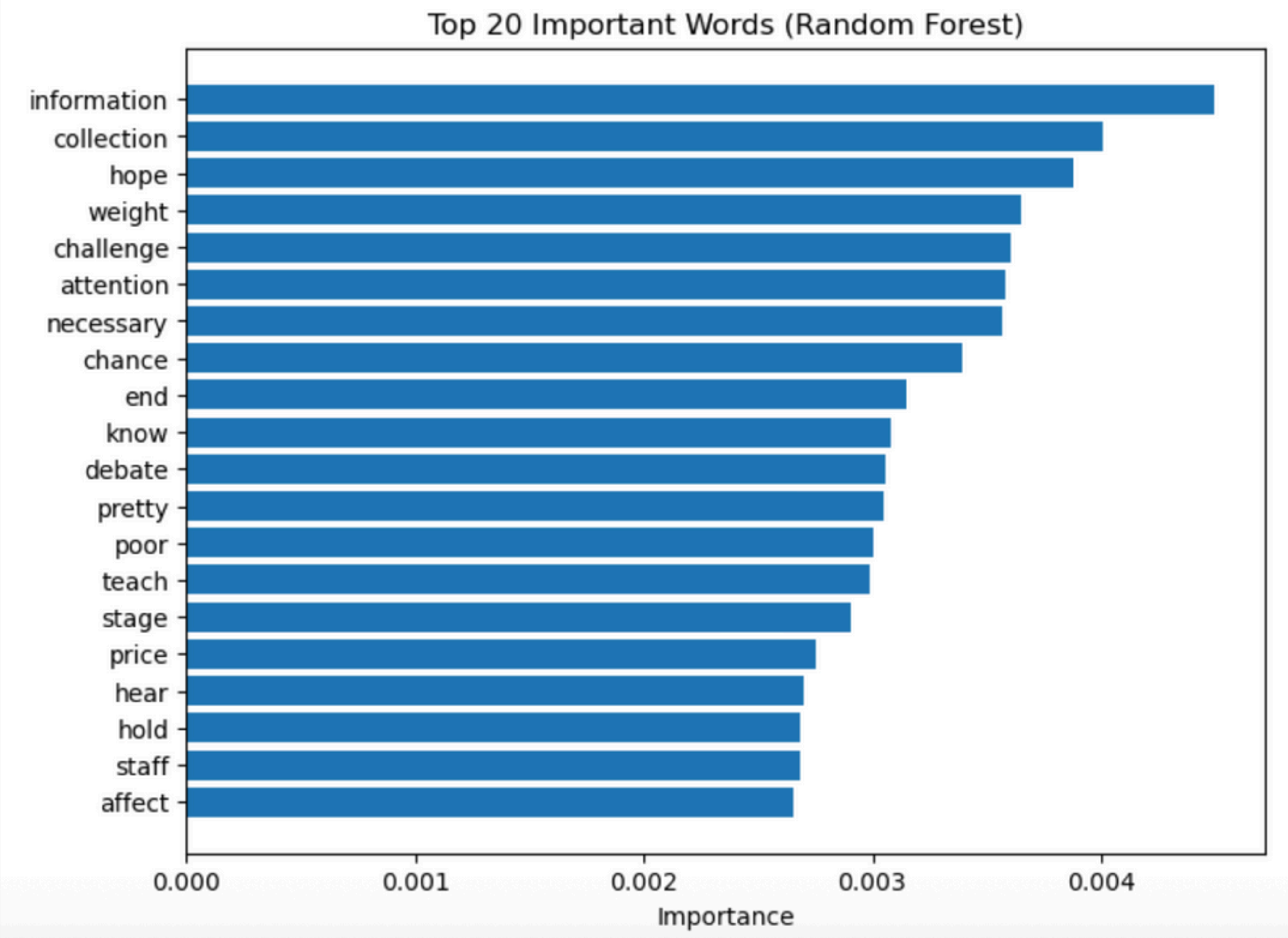
Identifies the most important words that the Random Forest model used to make its predictions.

- Tells which words were most “influential” in classifying a text.

This helps interpret the model's findings, see the writing styles and helps us understand why it made the predictions it did.

We can answer questions like:

- What are the key differences between AI and human content?
- Do AI-generated texts use different vocabulary than human-written ones?



Part 14 Summary Key Insights

The Logistic Regression worked very well for telling the difference in AI vs Human writing.

The Random Forest also did a good job and showed which words mattered the most.

- Some words and writing styles made it easier to guess if the text was written by AI or a human.
- The TF-IDF method helped highlight rare or unique words that are strong signals.
- The ROC curve showed that the Logistic Regression model was good at separating AI text from Human text.
- The visuals showed that most predictions were correct, with a few mistakes.
- The Random Forest chart showed the top words that helped the model make decisions.

Part 15 Reflection on Challenges and Learning

- Some challenges we had was figuring out how to clean and prepare the text so the computer could read it, and also understanding the different scores and outputs n the precision.
- Also choosing which models to test at first made us a bit confusing.
- Finding out how the TF-IDF works to turn text into numbers, and understanding how to train simple models and check how well they perform, and how to understand results.
- At first, we thought it was just about counting words, but now we see how weighting the words makes models smarter. We also realized accuracy isn't enough, other scores give a better result.
- We think we became more confident in explaining the results, not just running the code.

Final Reflection

- What surprised us that is a very simple model and it was very effective. In the future, maybe we would try more advanced models.
- This project helped me understand datasets and learn the machine learning basics, specially how to prepare data, train models, and explain my results.

Thank You

Group 2 Ava Leon and Sofia Curi