# FP_Initial_Exploration (1)

September 18, 2025

# 1 Final Project – Stage 1: Dataset Selection and Initial Exploration

Welcome to the first stage of your final project! Each group has been assigned a unique dataset from Kaggle. Your task is to explore the dataset and begin identifying questions or problems you may want to investigate.

## 1.1 Group Assignments and Datasets

### 1.1.1 Group 1: Anais Serrano Fragoso & Valeria Mora Silva

**Dataset:** Student Stress Monitoring Datasets
Explore physiological and behavioral indicators related to student stress.

---

### 1.1.2 Group 2: Sofia Belinda Curi Pachas & Ava Sofia Leon Macias

**Dataset:** AI vs Human Content Detection – 1000+ Records in 2025
Analyze patterns in AI-generated vs human-written content.

---

### 1.1.3 Group 3: Quoc Anh Nguyen

**Dataset:** Food Preferences
Preferences in food choices across different demographics.

---

## 1.2 Instructions

1. **Visit the dataset link** assigned to your group.
2. **Read the dataset description** and download the CSV file.
3. **Load the dataset** in this notebook using `pandas`.
4. Perform initial exploration using:
   - `df.head()`
   - `df.info()`
   - `df.describe()`
5. **Create visualizations** such as histograms to understand data distributions.
6. **Write your observations** in markdown cells:

- Describe the dataset and its features.
- Identify potential questions or problems to explore.

Make sure to save your notebook and include the dataset file when submitting.

Good luck and enjoy exploring your data!

### 1.2.1 Step 1: Load the Dataset

Import all libraries and modules that will be used.

[13]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

### 1.2.2 Step 2: Load the Dataset

Use `pd.read_csv()` to load the dataset.

[14]:
```
df = pd.read_csv('ai_human_content_detection_dataset.csv')
```

### 1.2.3 Step 2: Initial Exploration

Use `.info()` and `.describe()` to understand the dataset.

[15]:
```
print("\nDataset Info:")
print(df.info())
```

```
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
```

[16]:
```
print("\nDataset Description:")
print(df.describe())
```

```
Dataset Description:
        word_count  character_count  sentence_count  lexical_diversity  \
count  1367.000000      1367.000000     1367.000000        1367.000000
mean    140.190929       940.329188       25.610095           0.967646
std      97.410218       654.335255       17.867480           0.026254
min       3.000000        14.000000        1.000000           0.875000
25%      61.500000       410.500000       11.000000           0.951550
50%     131.000000       882.000000       24.000000           0.969200
75%     193.000000      1294.500000       35.000000           0.989100
max     443.000000      2966.000000       83.000000           1.000000

       avg_sentence_length  avg_word_length  punctuation_ratio  \
count          1367.000000      1367.000000        1367.000000
mean              5.486423         5.717783           0.027440
std               0.447202         0.279636           0.002801
min               3.000000         4.000000           0.019400
25%               5.270000         5.590000           0.026100
50%               5.480000         5.710000           0.027200
75%               5.700000         5.830000           0.028400
max               8.000000         8.330000           0.071400

       flesch_reading_ease  gunning_fog_index  grammar_errors  \
count          1288.000000        1332.000000     1367.000000
mean             52.183377           7.556877        1.537674
std              10.466570           1.866676        1.912012
min             -50.010000           1.200000        0.000000
25%              47.712500           6.620000        0.000000
50%              52.190000           7.515000        1.000000
75%              57.322500           8.390000        3.000000
max              98.870000          27.870000       10.000000

       passive_voice_ratio  predictability_score  burstiness  \
count          1336.000000           1367.000000  1367.000000
mean              0.150198             62.779049     0.427041
std               0.056738             28.223550     0.199249
min               0.050000             20.030000     0.101100
25%               0.099675             39.015000     0.250000
50%               0.151350             56.820000     0.408500
```

```
75%               0.200150             86.645000     0.594300
max               0.250000            119.930000     0.799500

       sentiment_score        label
count      1313.000000  1367.000000
mean         -0.007997     0.499634
std           0.588354     0.500183
min          -0.999300     0.000000
25%          -0.525800     0.000000
50%          -0.006200     0.000000
75%           0.502800     1.000000
max           0.995900     1.000000
```

### 1.2.4 Step 3: Observations

Write your observations below: - Number of rows and columns - Types of data - Missing values - Initial impressions

Rows: 1367 Columns: 17 Types of data: Integers, Floats and Text (Objects) Missing Values: flesch_reading_ease (79 Missing) gunning_fog_index (35 Missing) passive_voice_ratio (31 Missing) sentiment_score (54 missing)

Initial Impressions: Most data is numerical, with some columns having missing values that need cleaning It may need classification It uses labels to differentiate between AI-generated and human content.

### 1.2.5 Step 4: Dataset Description

Provide a brief description of the dataset and its features. In a markdown cell, write a short paragraph: - What is the dataset about? - What are the key features (columns)? - What kind of data does it contain?

- The dataset is about classifying text AI vs. human-generated text content detection.
- Key features (columns) text_content: The actual text sample (string). content_type: Category/type of text content (e.g., article, essay, etc.). word_count: Total number of words character_count: Total number of characters sentence_count: Number of sentences lexical_diversity: Ratio of unique words to total words avg_sentence_length: Average words per sentence avg_word_length: Average length of words punctuation_ratio: Ratio of punctuation marks to words flesch_reading_ease: Readability score (higher = easier to read) gunning_fog_index: Another readability metric (higher = harder to read) grammar_errors: Number of grammatical mistakes passive_voice_ratio: Ratio of sentences in passive voice. predictability_score: How predictable the text is (possibly related to AI likelihood). burstiness: Variability in sentence length (human writing tends to have higher burstiness). sentiment_score: Sentiment polarity (-1 = negative, 0 = neutral, 1 = positive). label: Target variable (0 = human-written, 1 = AI-generated).
- Text data: text_content (string) Categorical data: content_type (type of text) Numerical data: Integer columns: word_count, character_count, sentence_count, grammar_errors, label. Float columns: lexical_diversity, avg_sentence_length, avg_word_length, punctuation_ratio, flesch_reading_ease, gunning_fog_index, passive_voice_ratio, predictability_score, burstiness, sentiment_score.

### 1.2.6 Step 5: Potential Questions or Problems

List 2–3 questions or problems you want to explore using this dataset.

**Think critically about what can be explored. Examples:**

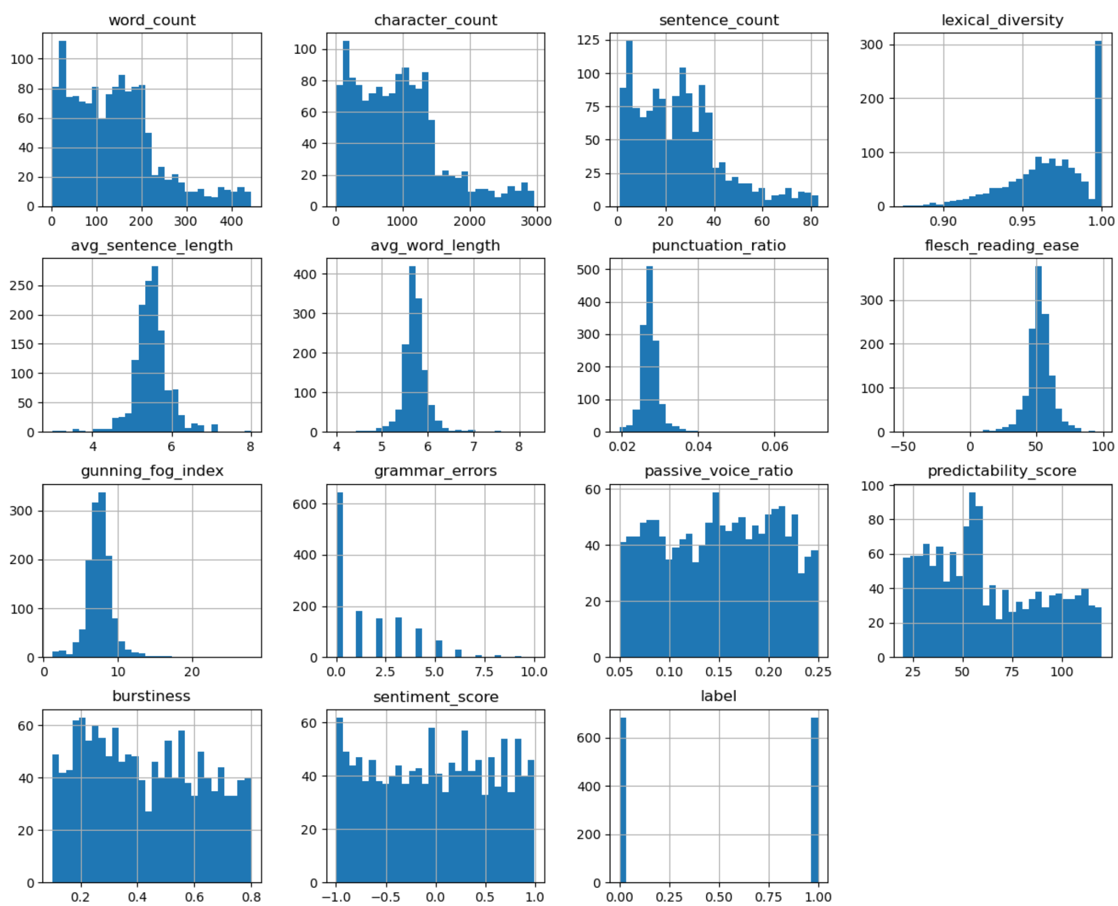
- Are there trends over time?
- Are there correlations between variables?
- Can we predict an outcome based on features?

Question 2: How is the writing style of a AI generated text different from human written? - Lexical_diversity and punctuation_ratio.

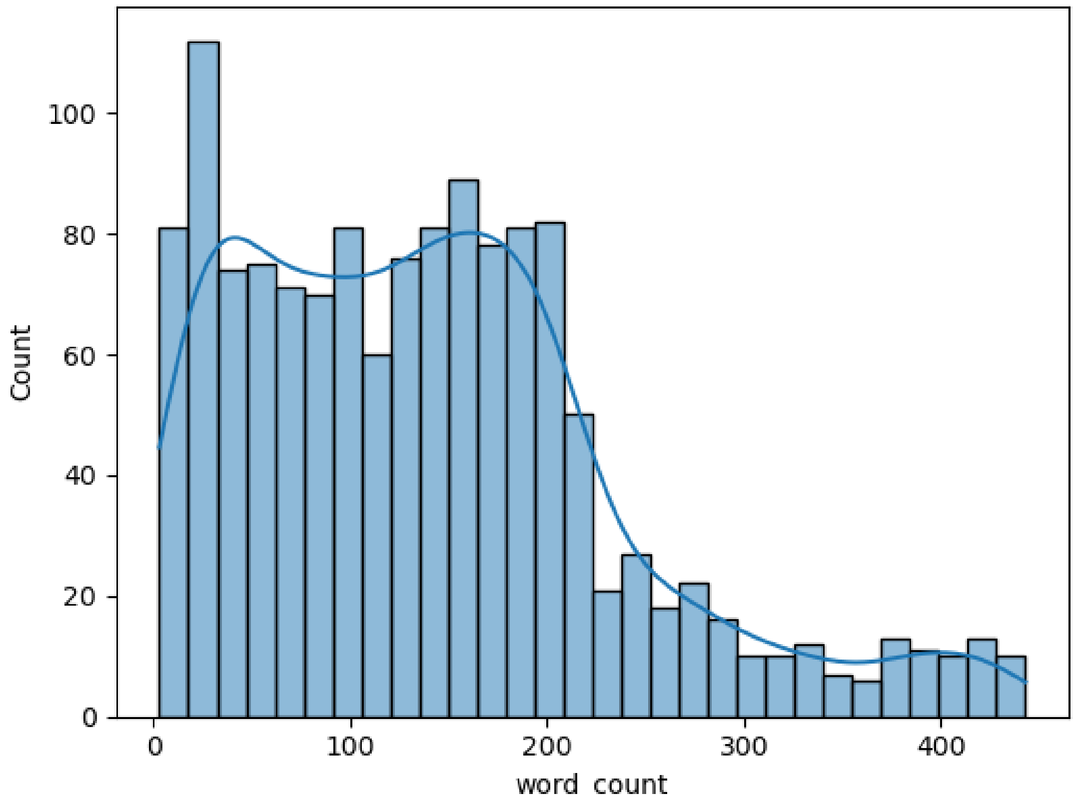
### 1.2.7 Step 6: Basic Visualizations

Create histograms to visualize the distribution of numerical columns.

[21]:
```
# Plot histograms for all numerical columns
df.hist(bins=30, figsize=(15, 12))
plt.show()
```



[22]:
```
# Using seaborn to plot a histogram for a specific column
sns.histplot(df['word_count'], bins=30, kde=True)
plt.show()
```




#### 1.2.8 Step 7: Interpretation of Visualizations

Explain what you observe from the histograms: - Are the distributions normal, skewed, or bimodal? - Are there any outliers? - What insights can you gain?

- Most of the histograms are right-skewed, with short low values

-There are outliers in the right-skewed distributions (In columns like word_count, character_count, and sentence_count). The long parts extended to the right show that a few texts are longer than the rest.

- Most texts are short (short sentences, words, and characters) There is AI vs Human distinction (The predictability_score and burstiness scores are useful for telling the difference between AI-written and human-written text). Ratios are mostly between 0.05 and 0.25. There is a balance between negative and positive.

### 1.3 Deliverables

**1. Jupyter Notebook with:**

- Dataset loaded and displayed
- .head(), .info(), .describe() outputs
- Comments or markdown cells explaining observations
- Brief dataset description
- List of potential questions/problems #### 2. Dataset file in .csv or .json format

[ ]:
```

```

# FP_Stage2_1

September 5, 2025

# 1 Final Project – Stage 2: Data Cleaning, Visualization & Feature Engineering

**Deadline:** Sep 05, 2025

## 1.1 Group Assignments and Datasets

### 1.1.1 Group 1: Anais Serrano Fragoso & Valeria Mora Silva

**Dataset:** Student Stress Monitoring Datasets
Explore physiological and behavioral indicators related to student stress.

For some ideas se the notebook by Denver Magtibay

---

### 1.1.2 Group 2: Sofia Belinda Curi Pachas & Ava Sofia Leon Macias

**Dataset:** AI vs Human Content Detection – 1000+ Records in 2025
Analyze patterns in AI-generated vs human-written content.

For some ideas se the notebook by Raayen

---

### 1.1.3 Group 3: Quoc Anh Nguyen

**Dataset:** Food Preferences
Preferences in food choices across different demographics.

For some ideas se the notebook by Rohith Mahadevan

---

### 1.1.4 Objectives:

- Clean the dataset (handle missing values, duplicates, outliers, etc.).
- Create meaningful visualizations (e.g., histograms, scatter plots, box plots).
- Perform feature engineering (create new variables or transform existing ones).
- Begin identifying patterns or trends in the data.

### 1.1.5 Deliverables:

- Updated Jupyter Notebook with:
  - Data cleaning steps and explanations.
  - At least 3 different types of visualizations.
  - Feature engineering examples.
  - Clear comments explaining your process and findings.

### 1.1.6 Tips for Completing Stage 2

- **Explore Examples**: Kaggle is a great resource not just for datasets, but also for learning from others. Check out public notebooks related to your dataset to see how others approach data cleaning, visualization, and feature engineering.
- **Be Curious**: Don't just clean the data—ask yourself *why* certain values are missing or *how* outliers might affect your analysis.
- **Visualize Often**: Use different types of plots to uncover patterns. Sometimes a scatter plot or box plot can reveal insights that summary statistics miss.
- **Document Your Work**: Use markdown cells to explain your decisions and findings. Clear documentation helps others understand your process—and helps you stay organized.
- **Ask Questions**: What trends are emerging? Are there relationships between variables? What features might be useful for prediction or classification?

Happy exploring!

[69]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load your dataset
df = pd.read_csv("ai_human_content_detection_dataset.csv")
```

## 1.2 Step 1: Data Cleaning

Handle missing values, duplicates, and outliers below.

[70]:
```
# Check for missing values
print("\nMissing values per column:")
print(df.isnull().sum())
```

```

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
```

[71]:
```
# Drop or fill missing values
df = df.dropna()
```

[72]:
```
# Remove duplicates
df = df.drop_duplicates()
```

[77]:
```
# Detect outliers using IQR
def check_outliers(data, column):
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR

    outliers = data[(data[column] < lower) | (data[column] > upper)]
    return outliers, lower, upper

for col in df.select_dtypes(include=["float64", "int64"]).columns:
    outliers, lower, upper = check_outliers(df, col)
    print("------")
    print("Column:", col)
    print("Lower limit:", lower)
    print("Upper limit:", upper)
    print("Number of outliers:", len(outliers))

    if len(outliers) > 0:
        print("Number of outliers:", len(outliers))
    else:
        print("No outliers are detected")
```

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

```
------
Column: character_count
Lower limit: -930.0
Upper limit: 2618.0
Number of outliers: 40
Number of outliers: 40
------
Column: sentence_count
Lower limit: -25.0
Upper limit: 71.0
Number of outliers: 38
Number of outliers: 38
------
Column: lexical_diversity
Lower limit: 0.895875
Upper limit: 1.044875
Number of outliers: 11
Number of outliers: 11
------
Column: avg_sentence_length
Lower limit: 4.675
Upper limit: 6.315
Number of outliers: 87
Number of outliers: 87
------
Column: avg_word_length
Lower limit: 5.2299999999999995
Upper limit: 6.19
Number of outliers: 87
Number of outliers: 87
------
Column: punctuation_ratio
Lower limit: 0.022650000000000003
Upper limit: 0.03185
Number of outliers: 74
Number of outliers: 74
------
Column: flesch_reading_ease
Lower limit: 33.2375
Upper limit: 71.6975
Number of outliers: 78
Number of outliers: 78
------
Column: gunning_fog_index
Lower limit: 3.957500000000001
Upper limit: 11.0575
Number of outliers: 62
Number of outliers: 62
```

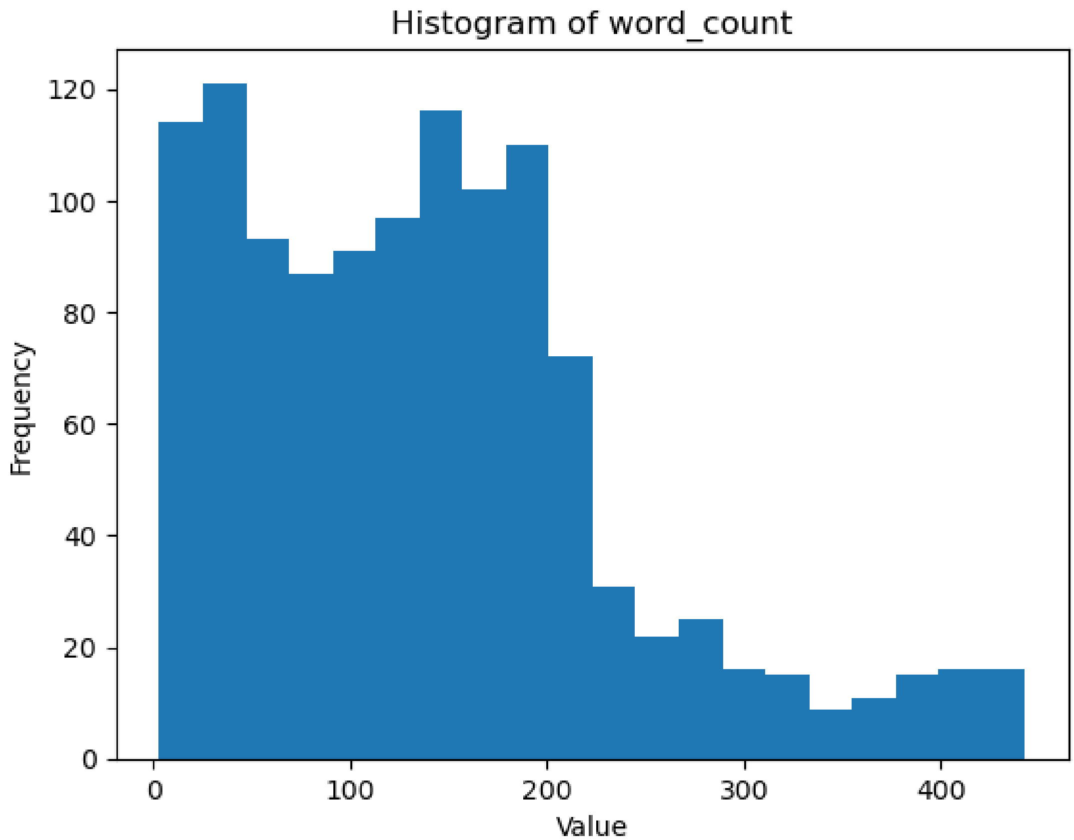
```
------
Column: grammar_errors
Lower limit: -4.5
Upper limit: 7.5
Number of outliers: 13
Number of outliers: 13
------
Column: passive_voice_ratio
Lower limit: -0.05242500000000003
Upper limit: 0.351775
Number of outliers: 0
No outliers are detected
------
Column: predictability_score
Lower limit: -32.815000000000005
Upper limit: 159.465
Number of outliers: 0
No outliers are detected
------
Column: burstiness
Lower limit: -0.2615000000000001
Upper limit: 1.1041000000000003
Number of outliers: 0
No outliers are detected
------
Column: sentiment_score
Lower limit: -2.090775
Upper limit: 2.0874249999999996
Number of outliers: 0
No outliers are detected
------
Column: label
Lower limit: -1.5
Upper limit: 2.5
Number of outliers: 0
No outliers are detected
```

### 1.3 Step 2: Visualizations

Create at least three types of visualizations to explore the data.

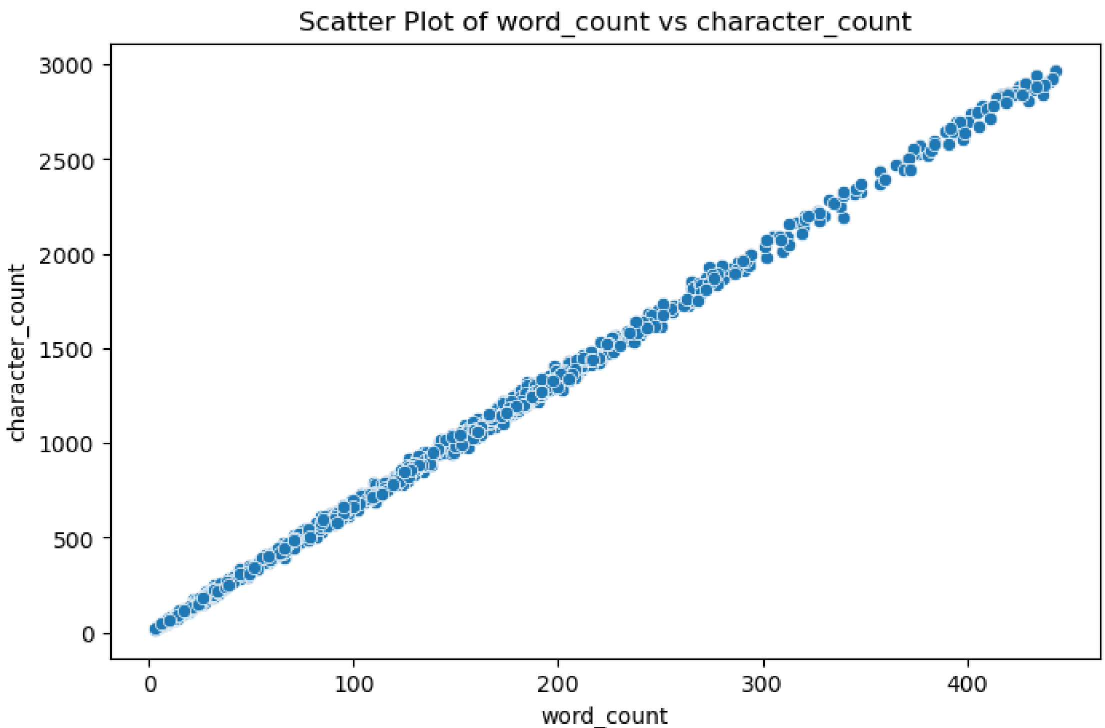
[74]:
```
# Histogram
if 'word_count' in df.columns:
    plt.hist(df['word_count'], bins=20)
    plt.title('Histogram of word_count')
    plt.xlabel('Value')
    plt.ylabel('Frequency')
    plt.show()
```

```
print(df['word_count'].head())
```

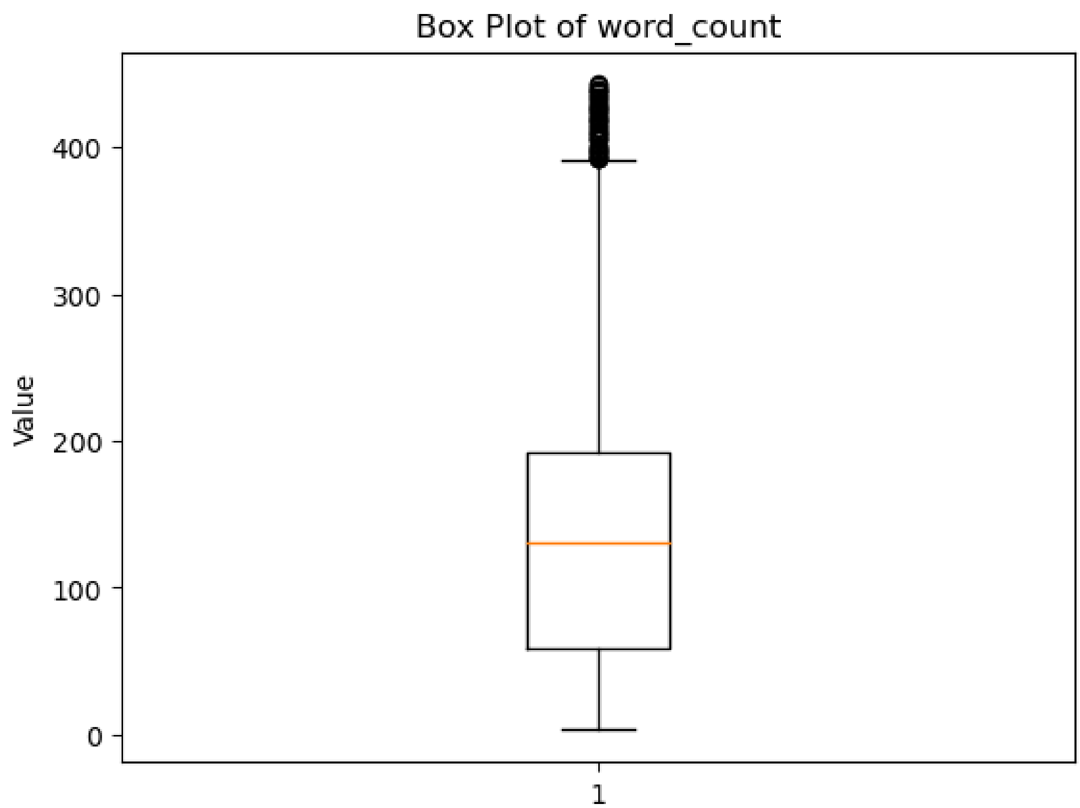



```
0    288
1    253
2    420
5    198
6     84
Name: word_count, dtype: int64
```

[75]:
```
# Scatter plot
if len(numeric_cols) >= 2:
    plt.figure(figsize=(8,5))
    sns.scatterplot(data=df, x=numeric_cols[0], y=numeric_cols[1])
    plt.title(f'Scatter Plot of {numeric_cols[0]} vs {numeric_cols[1]}')
    plt.xlabel(numeric_cols[0])
    plt.ylabel(numeric_cols[1])
    plt.show()
```




[76]:
```
# Box plot
plt.boxplot(df[numeric_cols[0]])
plt.title(f'Box Plot of {numeric_cols[0]}')
plt.ylabel('Value')
plt.show()
```




## 1.4 Step 3: Feature Engineering

Create new features or transform existing ones.

[80]:
```
# Example: Create a new feature
df["words_per_sentence"] = df["word_count"] / df["sentence_count"]
```

[81]:
```
# Example: Transform a feature
df["log_word_count"] = np.log1p(df["word_count"])

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

```
   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
```

### 1.5 Step 4: Observations and Findings

Use this section to explain your process and summarize your findings from cleaning, visualization, and feature engineering.

Data Cleaning: - We removed rows with missing values to keep only complete data. - We removed duplicated rows to avoid repetition and make the data better. - For the Outlier detection we used the IQR method to find extreme values in numerical columns, our data didn't have outliers so created a function that reads and explains it.

Data Visualization: - We created a histogram for the word_count column to understand the distribution, and helped us spot unusual patterns. - For the scatter plot we focused on seeing the relationship between two numerical columns, this helped to identify any correlations. - We created a box plot for the word_count column to show its range, median, and any potential outliers, this helped us see if there were any extreme values we needed to pay more attention to.

Feature Engineering: - We created a new feature, words_per_sentence, by dividing word_count by sentence_count. To gives us an average number of words per sentence, which would helps us analyze the text structure, and make a comparison on the topic. - We also transformed the word_count column by applying a log transformation to create the log_word_count feature. To helps us reduce the size of very large word counts and makes the data more manageable for analysis.

# WIM250_SU25_Stage3

September 18, 2025

# 1 Final Project – Stage 3: Modeling, Insights & Final Presentation

**Deadline: September 19, 2025**

Welcome to the final stage of your project! Follow the steps below to complete your analysis and prepare your final submission.

## 1.1 Group Assignments and Datasets

### 1.1.1 Group 1: Anais Serrano Fragoso & Valeria Mora Silva

**Dataset:** Student Stress Monitoring Datasets
Explore physiological and behavioral indicators related to student stress.

For some ideas se the notebook by ErimCENGIZ

---

### 1.1.2 Group 2: Sofia Belinda Curi Pachas & Ava Sofia Leon Macias

**Dataset:** AI vs Human Content Detection – 1000+ Records in 2025
Analyze patterns in AI-generated vs human-written content.

For some ideas se the notebook by Omar Essa

---

### 1.1.3 Group 3: Quoc Anh Nguyen

**Dataset:** Food Preferences
Preferences in food choices across different demographics.

For some ideas se the notebook by Gabriel Atkin

---

### 1.1.4 Objectives:

1. **Apply a Model**
   Use **at least one** machine learning algorithm or statistical model to analyze your dataset.
2. **Extract Insights**
   Summarize the **key findings** from your analysis. What patterns did you discover?

3. **Reflect on the Process**
   Briefly discuss:
   - Challenges you faced
   - What you learned
   - How your understanding evolved
4. **Cite Your Sources**
   Include proper citations for:
   - Kaggle notebooks or datasets
   - AI tools (e.g., Copilot, ChatGPT)
   - Any external code or references

#### 1.1.5 Deliverables:

Submit a **.zip folder** containing:

1. **Final Jupyter Notebook**

- Well-organized and commented
- Includes all code, outputs, and explanations

2. **Dataset File(s)**

- Include the original or cleaned dataset used in your analysis

3. **Bibliography or References Section**

- A markdown cell or separate file listing all sources

#### 1.1.6 Tips for Completing Stage 3

- **Choose the Right Model**
  Match your model to your data type and project question. For example:
  - Classification: Logistic Regression, Decision Trees
  - Regression: Linear Regression, Ridge/Lasso
  - Clustering: K-Means, DBSCAN
  - Statistical: t-tests, ANOVA
- **Explain Your Steps Clearly**
  Use markdown cells to describe:
  - Why you chose a model
  - How you prepared the data
  - What the results mean
- **Visualize Your Results**
  Include graphs or charts to support your insights (e.g., confusion matrix, scatter plots, bar charts).

- **Keep It Clean**
  Remove unused code, test cells, or irrelevant outputs. Make it easy to follow.
- **Reflect Honestly**
  Your reflection doesn't need to be perfect—just thoughtful. What surprised you? What would you do differently?
- **Check Your Citations**
  Use APA or MLA format, or just be consistent. Cite tools like this:
  > "Analysis supported by Copilot (Microsoft AI Assistant)."

## 1.2 Step 1: Load Your Dataset

Upload and load the dataset you will use for modeling.

We loaded the data set and used "df.head()" to look at the first few rows to understand a little bit of the data

[23]:
```python
import pandas as pd
df = pd.read_csv("ai_human_content_detection_dataset.csv")
df.head()
```

[23]:
```
                                        text_content      content_type  \
0  Score each cause. Quality throughout beautiful…    academic_paper
1  Board its rock. Job worker break tonight coupl…             essay
2  Way debate decision produce. Dream necessary c…    academic_paper
3  Story turn because such during open model. Tha…  creative_writing
4  Place specific as simply leader fall analysis…      news_article

   word_count  character_count  sentence_count  lexical_diversity  \
0         288             1927              54             0.9514
1         253             1719              45             0.9723
2         420             2849              75             0.9071
3         196             1310              34             0.9592
4         160             1115              28             0.9688

   avg_sentence_length  avg_word_length  punctuation_ratio  \
0                 5.33             5.69             0.0280
1                 5.62             5.80             0.0262
2                 5.60             5.79             0.0263
3                 5.76             5.69             0.0260
4                 5.71             5.97             0.0251

   flesch_reading_ease  gunning_fog_index  grammar_errors  \
0                53.08               7.41               1
1                50.32               8.10               6
2                46.86               7.86               5
3                53.80               7.00               2
4                44.53               8.29               0
```

```
   passive_voice_ratio  predictability_score  burstiness  sentiment_score  \
0               0.1041                105.86      0.5531           0.2034
1               0.2045                100.29      0.5643           0.4854
2               0.2308                 96.88      0.4979          -0.2369
3               0.1912                 88.79      0.6241              NaN
4               0.1318                 26.15      0.2894              NaN

   label
0      1
1      1
2      1
3      1
4      1
```

### 1.3 Step 2: Data Preprocessing

Clean and prepare your data for modeling.

We removed some empty rows and separated features as(X = text) and labels as (y = AI or Human) We also turned the text into numbers using TF-IDF, because computers can't read the words directly, so we turned them into numbers.

- TF: Says how often a word appears in a document and IDF: Says how unique that word is in all documents.
- TF-IDF combines them, highlighting words that are frequent in one text, but not too common everywhere else.

[24]:
```
df = df.dropna(subset=["text_content", "label"])

# To define features (X) as the text we want to classify and target (y) as the
↪label (AI or Human)
X = df["text_content"]
y = df["label"]

# To split the data into 80% train, 20% test
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Convert text into numbers using TF-IDF, and used "max_features = 5000" to
↪keep only the top 5000 words
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(max_features=5000, stop_words="english")
X_train_vec = vectorizer.fit_transform(X_train)
```

```
X_test_vec = vectorizer.transform(X_test)
```

### 1.4 Step 3: Apply a Machine Learning or Statistical Model

Choose and apply a model that fits your data and project goals.

[25]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

# Create the model
model = LogisticRegression(max_iter=1000)
model.fit(X_train_vec, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test_vec)

print("Classification Report (Logistic Regression):\n")
print(classification_report(y_test, y_pred))

print("Confusion Matrix:\n")
print(confusion_matrix(y_test, y_pred))
```

```
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]]
```

Random Forest Model

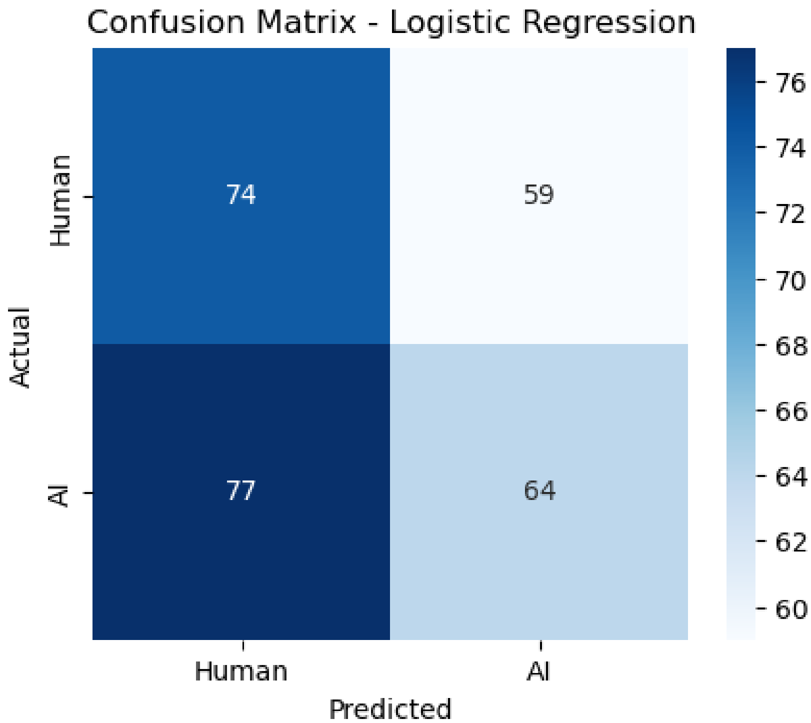
[27]:
```
from sklearn.ensemble import RandomForestClassifier

# Create Random Forest
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train_vec, y_train)

# Predictions
y_pred_rf = rf_model.predict(X_test_vec)
```

```python
print("Classification Report (Random Forest):\n")
print(classification_report(y_test, y_pred_rf))
```

```
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
```

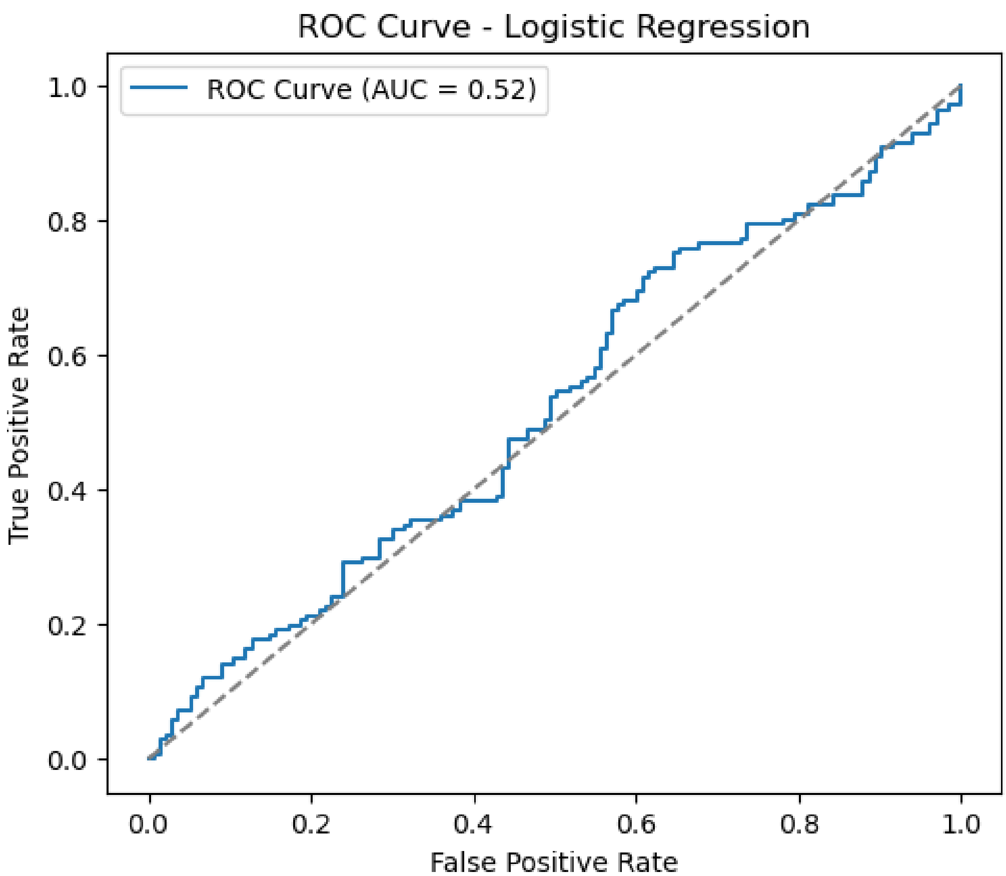
[ ]: Confusion Matrix (to show mistakes)

[28]:
```python
import matplotlib.pyplot as plt
import seaborn as sns

cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=["Human", "AI"],
            yticklabels=["Human", "AI"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Logistic Regression")
plt.show()
```



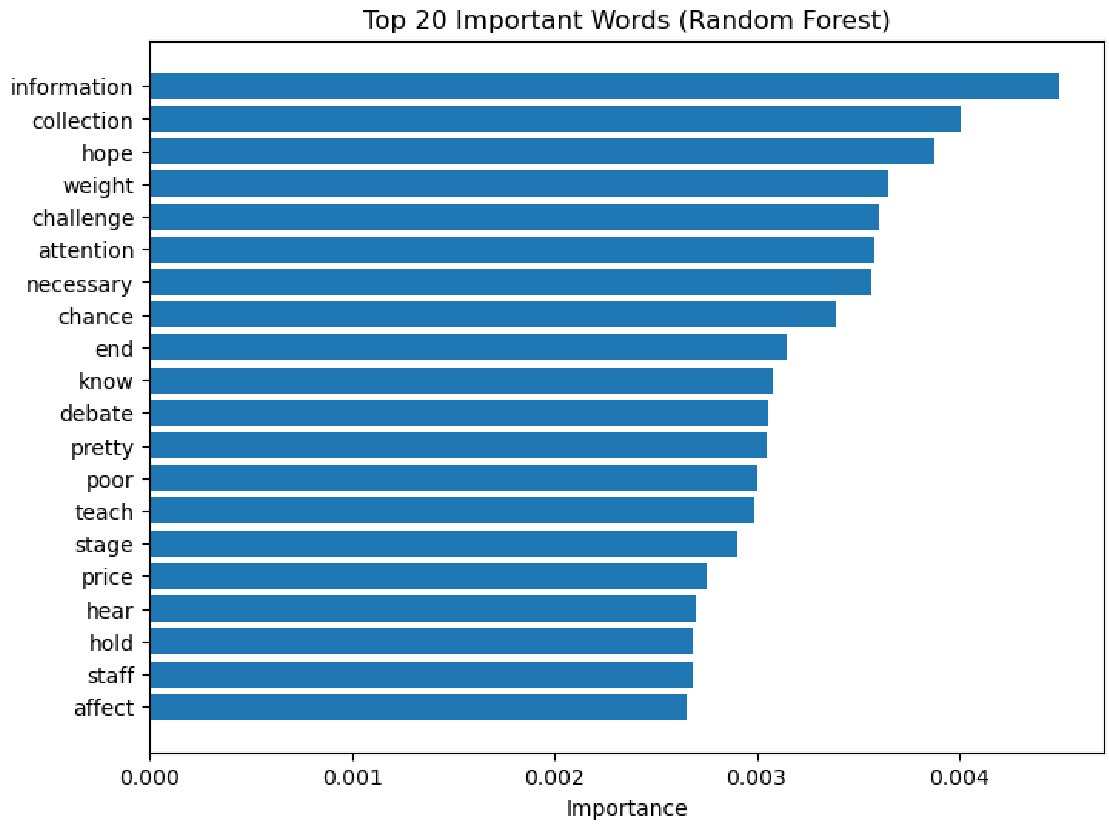

[ ]: ROC Curve (to measure how well the model separates AI vs Human)

[29]:
```
from sklearn.metrics import roc_curve, auc

y_prob = model.predict_proba(X_test_vec)[:,1]
fpr, tpr, _ = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, label=f"ROC Curve (AUC = {roc_auc:.2f})")
plt.plot([0,1], [0,1], linestyle="--", color="gray")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve - Logistic Regression")
plt.legend()
plt.show()
```




[ ]: Feature Importance (Random Forest)

[30]:
```
import numpy as np

importances = rf_model.feature_importances_
indices = np.argsort(importances)[-20:]

plt.figure(figsize=(8,6))
plt.barh(range(len(indices)), importances[indices], align="center")
plt.yticks(range(len(indices)), [vectorizer.get_feature_names_out()[i] for i in␣
 ↪indices])
plt.xlabel("Importance")
plt.title("Top 20 Important Words (Random Forest)")
plt.show()
```




### 1.5 Step 4: Summarize Key Insights

Describe the main findings from your analysis.

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.

### 1.6 Step 5: Reflect on Challenges and Learning

Write a short reflection on what you learned and any challenges you faced.

- 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 in the precision, recall, F1). 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.

## 1.7 Final Reflection

- What suprised is 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.

## 1.8 Step 6: Cite Your Sources

List all external sources used (e.g., Kaggle notebooks, AI tools).

Dataset from Kaggle: https://www.kaggle.com/datasets/pratyushpuri/ai-vs-human-content-detection-1000-record-in-2025?resource=download

Inspiration from Kaggle Notebook by Omar Essa: https://www.kaggle.com/code/jockeroika/ai-vs-human-2025

Analysis supported by OpenAI (ChatGPT)

## 1.9 Final Checklist

- [ ] Final Jupyter Notebook is clean and commented
- [ ] Dataset file(s) included
- [ ] Bibliography or references section completed
- [ ] All code runs without errors

Zip all files and submit before **September 19, 2025**.