```
Start coding or generate with AI.
```

## Task

Analyze the dataset and classify the test as AI or human generated

## Load data

```
import pandas as pd

df = pd.read_csv('/content/balanced_ai_human_prompts.csv')
display(df.head())
```

| | text | generated |
|---|---|---|
| 0 | Machine learning, a subset of artificial intel... | 1 |
| 1 | A decision tree, a prominent machine learning ... | 1 |
| 2 | Education, a cornerstone of societal progress,... | 1 |
| 3 | Computers, the backbone of modern technology, ... | 1 |
| 4 | Chess, a timeless game of strategy and intelle... | 1 |

```
import nltk
nltk.download('punkt_tab')
from nltk.tokenize import sent_tokenize
import pandas as pd

def split_text_into_sections(text):
    sentences = sent_tokenize(text)
    num_sentences = len(sentences)

    if num_sentences < 3:
        return text, "", ""
    elif num_sentences == 3:
        return sentences[0], sentences[1], sentences[2]

    section_size = num_sentences // 3
    early_sentences = sentences[:section_size]
    mid_sentences = sentences[section_size:2 * section_size]
    later_sentences = sentences[2 * section_size:]

    early_text = " ".join(early_sentences)
    mid_text = " ".join(mid_sentences)
    later_text = " ".join(later_sentences)

    return early_text, mid_text, later_text

df[['early_text', 'mid_text', 'later_text']] = df['text'].apply(lambda x: pd.Series(split_text_into_sections(x)))
display(df.head())
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
```

| | text | generated | early_text | mid_text | later_text |
|---|---|---|---|---|---|
| 0 | Machine learning, a subset of artificial intel... | 1 | Machine learning, a subset of artificial intel... | At its core, machine learning enables computer... | As machine learning continues to advance, it b... |
| 1 | A decision tree, a prominent machine learning ... | 1 | A decision tree, a prominent machine learning ... | The algorithm evaluates input features at each... | However, decision trees may suffer from overfi... |
| 2 | Education, a cornerstone of societal progress,... | 1 | Education, a cornerstone of societal progress,... | It encompasses formal and informal learning, e... | In the digital age, technology enhances educat... |
| 3 | Computers, the backbone of modern technology, ... | 1 | Computers, the backbone of modern technology, ... | From personal computing devices to massive dat... | Rapid advancements in processing power, memory... |
| 4 | Chess, a timeless game of strategy and intelle... | 1 | Chess, a timeless game of strategy and intelle... | Beyond its recreational appeal, chess fosters ... | As a metaphor for life's complexities, chess t... |

## Pos tagging

```
import nltk
from nltk import pos_tag
```

```
from nltk.tokenize import word_tokenize

nltk.download('averaged_perceptron_tagger')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger_eng')


def apply_pos_tagging(text):
    if isinstance(text, str):
        tokens = word_tokenize(text)
        return pos_tag(tokens)
    return []

df['early_pos_tags'] = df['early_text'].apply(apply_pos_tagging)
df['mid_pos_tags'] = df['mid_text'].apply(apply_pos_tagging)
df['later_pos_tags'] = df['later_text'].apply(apply_pos_tagging)

display(df.head())
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger_eng is already up-to-
[nltk_data]       date!
```

| | text | generated | early_text | mid_text | later_text | early_pos_tags | mid_pos_tags | later_pos_tags |
|---|---|---|---|---|---|---|---|---|
| **0** | Machine learning, a subset of artificial intel... | 1 | Machine learning, a subset of artificial intel... | At its core, machine learning enables computer... | As machine learning continues to advance, it b... | [(Machine, NN), (learning, NN), (,, ,), (a, DT... | [(At, IN), (its, PRP$), (core, NN), (,, ,), (m... | [(As, IN), (machine, NN), (learning, VBG), (co... |
| **1** | A decision tree, a prominent machine learning ... | 1 | A decision tree, a prominent machine learning ... | The algorithm evaluates input features at each... | However, decision trees may suffer from overfi... | [(A, DT), (decision, NN), (tree, NN), (,, ,), ... | [(The, DT), (algorithm, NN), (evaluates, VBZ),... | [(However, RB), (,, ,), (decision, NN), (trees... |
| **2** | Education, a cornerstone of societal progress,... | 1 | Education, a cornerstone of societal progress,... | It encompasses formal and informal learning, e... | In the digital age, technology enhances educat... | [(Education, NN), (,, ,), (a, DT), (cornerston... | [(It, PRP), (encompasses, VBZ), (formal, JJ), ... | [(In, IN), (the, DT), (digital, JJ), (age, NN)... |
| **3** | Computers, the backbone of modern technology, ... | 1 | Computers, the backbone of modern technology, ... | From personal computing devices to massive dat... | Rapid advancements in processing power, memory... | [(Computers, NNS), (,, ,), (the, DT), (backbon... | [(From, IN), (personal, JJ), (computing, VBG),... | [(Rapid, JJ), (advancements, NNS), (in, IN), (... |
| | Chess, a timeless | | Chess, a timeless game | Beyond its recreational | As a metaphor for life's | [(Chess, NNP), (,, | [(Beyond, IN), (its, PRP$) | [(As, IN), (a, DT), |

## Count pos tags

```
def count_pos_tags(pos_tagged_list):
    pos_counts = {}
    for word, tag in pos_tagged_list:
        pos_counts[tag] = pos_counts.get(tag, 0) + 1
    return pos_counts

df['early_pos_counts'] = df['early_pos_tags'].apply(count_pos_tags)
df['mid_pos_counts'] = df['mid_pos_tags'].apply(count_pos_tags)
df['later_pos_counts'] = df['later_pos_tags'].apply(count_pos_tags)

display(df.head())
```

| | text | generated | early_text | mid_text | later_text | early_pos_tags | mid_pos_tags | later_pos_tags | early_pos_counts |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Machine learning, a subset of artificial intel... | 1 | Machine learning, a subset of artificial intel... | At its core, machine learning enables computer... | As machine learning continues to advance, it b... | [(Machine, NN), (learning, NN), (,, ,), (a, DT... | [(At, IN), (its, PRP$), (core, NN), (,, ,), (m... | [(As, IN), (machine, NN), (learning, VBG), (co... | {'NN': 6, ',': 3, 'DT': 3, 'IN': 3, 'JJ': 2, '... |
| **1** | A decision tree, a prominent machine learning ... | 1 | A decision tree, a prominent machine learning ... | The algorithm evaluates input features at each... | However, decision trees may suffer from overfi... | [(A, DT), (decision, NN), (tree, NN), (,, ,), ... | [(The, DT), (algorithm, NN), (evaluates, VBZ),... | [(However, RB), (,, ,), (decision, NN), (trees... | {'DT': 3, 'NN': 7, ',': 4, 'JJ': 3, 'VBG': 1, ... |
| **2** | Education, a cornerstone of societal progress,... | 1 | Education, a cornerstone of societal progress,... | It encompasses formal and informal learning, e... | In the digital age, technology enhances educat... | [(Education, NN), (,, ,), (a, DT), (cornerston... | [(It, PRP), (encompasses, VBZ), (formal, JJ), ... | [(In, IN), (the, DT), (digital, JJ), (age, NN)... | {'NN': 5, ',': 3, 'DT': 3, 'IN': 3, 'JJ': 1, '... |
| **3** | Computers, the backbone of modern technology, ... | 1 | Computers, the backbone of modern technology, ... | From personal computing devices to massive dat... | Rapid advancements in processing power, memory... | [(Computers, NNS), (,, ,), (the, DT), (backbon... | [(From, IN), (personal, JJ), (computing, VBG),... | [(Rapid, JJ), (advancements, NNS), (in, IN), (... | {'NNS': 1, ',': 2, 'DT': 2, 'NN': 4, 'IN': 2, ... |
| **4** | Chess, a timeless game of strategy and intelle... | 1 | Chess, a timeless game of strategy and intelle... | Beyond its recreational appeal, chess fosters ... | As a metaphor for life's complexities, chess t... | [(Chess, NNP), (,, ,), (a, DT), (timeless, JJ)... | [(Beyond, IN), (its, PRP$), (recreational, JJ)... | [(As, IN), (a, DT), (metaphor, NN), (for, IN),... | {'NNP': 1, ',': 4, 'DT': 5, 'JJ': 7, 'NN': 9, ... |

```
Start coding or generate with AI.
```

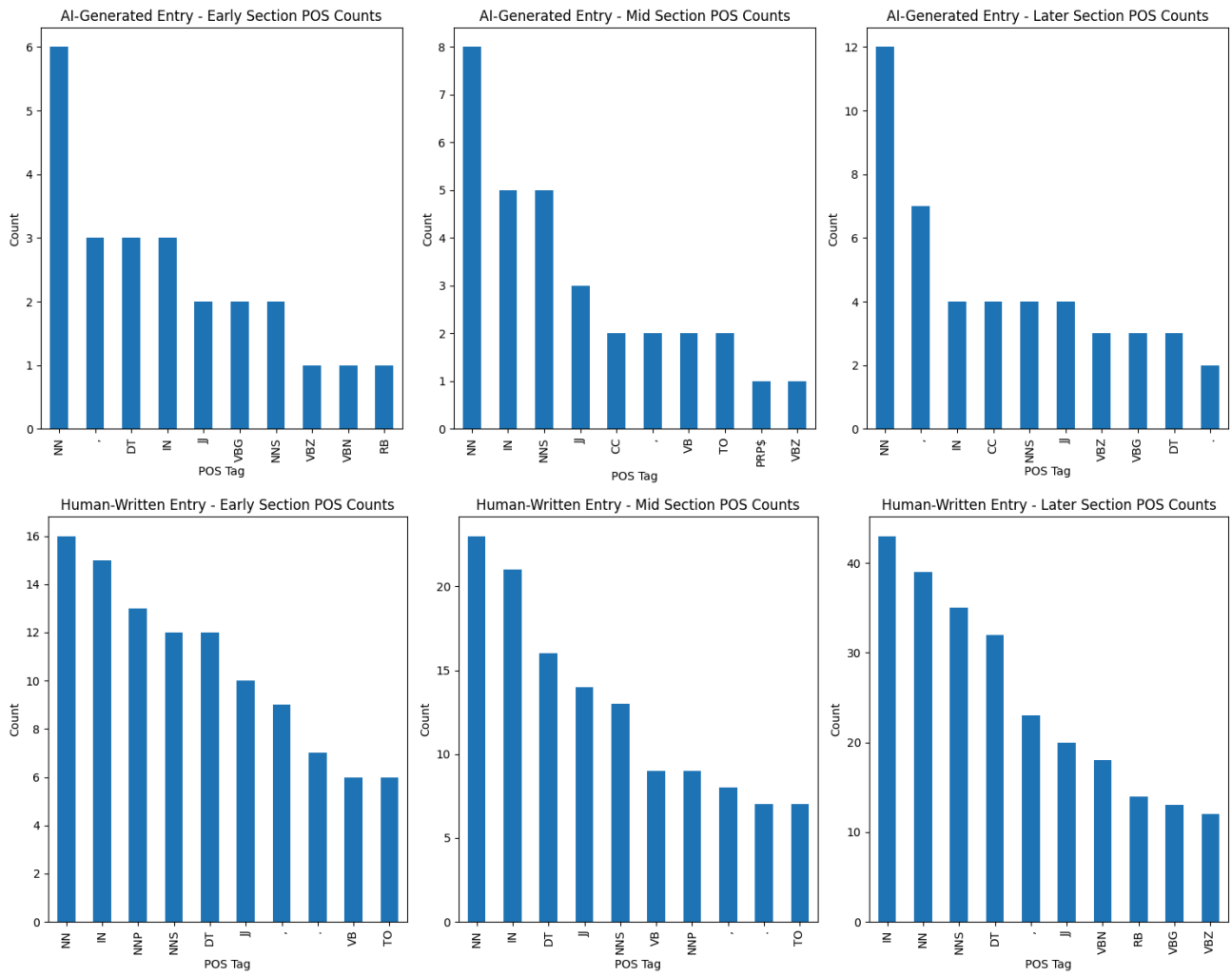
## Display POS counts for sample entries

```
print("Sample AI-Generated POS Counts:")
ai_examples = df[df['generated'] == 1].head(2)
for index, row in ai_examples.iterrows():
    print(f"\nEntry {index}:")
    print("Early section POS counts:", row['early_pos_counts'])
    print("Mid section POS counts:", row['mid_pos_counts'])
    print("Later section POS counts:", row['later_pos_counts'])

print("\nSample Human-Written POS Counts:")
human_examples = df[df['generated'] == 0].head(2)
for index, row in human_examples.iterrows():
    print(f"\nEntry {index}:")
    print("Early section POS counts:", row['early_pos_counts'])
    print("Mid section POS counts:", row['mid_pos_counts'])
    print("Later section POS counts:", row['later_pos_counts'])
```

```
ample AI-Generated POS Counts:

ntry 0:
arly section POS counts: {'NN': 6, ',': 3, 'DT': 3, 'IN': 3, 'JJ': 2, 'VBZ': 1, 'RB': 1, 'VBN': 1, 'VBG': 2, 'NNS': 2, 'CC': 1,
id section POS counts: {'IN': 5, 'PRP$': 1, 'NN': 8, ',': 2, 'VBZ': 1, 'NNS': 5, 'TO': 2, 'VB': 2, 'CC': 2, 'JJ': 3, 'VBG': 1,
ater section POS counts: {'IN': 4, 'NN': 12, 'VBG': 3, 'VBZ': 3, 'TO': 2, 'VB': 1, ',': 7, 'PRP': 1, 'CC': 4, 'NNS': 4, 'JJ': 4,

ntry 1:
arly section POS counts: {'DT': 3, 'NN': 7, ',': 4, 'JJ': 3, 'VBG': 1, 'VBZ': 2, 'IN': 1, '.': 2, 'NNP': 1, 'NNS': 5, 'VBP': 2,
id section POS counts: {'DT': 8, 'NN': 8, 'VBZ': 3, 'JJ': 8, 'NNS': 3, 'IN': 3, ',': 3, 'VBG': 3, 'TO': 1, 'VBN': 1, '.': 2, 'CC
ater section POS counts: {'RB': 1, ',': 4, 'NN': 8, 'NNS': 8, 'MD': 1, 'VB': 1, 'IN': 8, 'VBG': 3, 'VBD': 1, 'CC': 3, 'JJ': 1,

ample Human-Written POS Counts:

ntry 82:
arly section POS counts: {'NNS': 12, '.': 7, 'VBP': 6, 'VBN': 2, 'IN': 15, 'PRP': 2, 'VBD': 3, 'JJ': 10, 'DT': 12, 'CD': 1, ','
id section POS counts: {'NN': 23, 'NNS': 13, 'VBP': 4, 'JJ': 14, 'IN': 21, 'CD': 4, 'NNP': 9, ':': 1, 'CC': 2, 'RB': 3, 'TO': 7,
ater section POS counts: {'RB': 14, ',': 23, 'IN': 43, 'DT': 32, 'NN': 39, '``': 1, 'NNP': 10, 'VBZ': 12, 'VBG': 13, 'JJ': 20,

ntry 83:
arly section POS counts: {'NN': 23, 'VBZ': 4, 'DT': 12, 'JJ': 15, 'IN': 19, 'JJS': 1, 'NNS': 18, 'VBP': 8, '.': 9, ',': 10, 'CC
id section POS counts: {'DT': 14, 'NN': 30, 'TO': 2, 'VB': 5, 'NNS': 12, 'VBZ': 3, 'NNP': 11, 'POS': 2, 'IN': 23, '.': 9, 'VBD'
ater section POS counts: {'IN': 22, 'DT': 22, 'NN': 35, 'NNS': 14, 'CC': 10, 'VBG': 7, 'VBN': 6, ',': 6, 'VBP': 2, 'TO': 3, 'VB
```

## Visualize aggregated POS counts

```
import matplotlib.pyplot as plt
import pandas as pd

# Function to plot POS counts for early, mid, and later sections of a single entry
def plot_pos_counts_by_section(row, title_prefix):
    early_counts = pd.Series(row['early_pos_counts'])
    mid_counts = pd.Series(row['mid_pos_counts'])
    later_counts = pd.Series(row['later_pos_counts'])

    plt.figure(figsize=(15, 6))

    plt.subplot(1, 3, 1)
    early_counts.sort_values(ascending=False).head(10).plot(kind='bar')
    plt.title(f'{title_prefix} - Early Section POS Counts')
    plt.xlabel('POS Tag')
    plt.ylabel('Count')

    plt.subplot(1, 3, 2)
    mid_counts.sort_values(ascending=False).head(10).plot(kind='bar')
    plt.title(f'{title_prefix} - Mid Section POS Counts')
    plt.xlabel('POS Tag')
    plt.ylabel('Count')

    plt.subplot(1, 3, 3)
    later_counts.sort_values(ascending=False).head(10).plot(kind='bar')
    plt.title(f'{title_prefix} - Later Section POS Counts')
    plt.xlabel('POS Tag')
    plt.ylabel('Count')

    plt.tight_layout()
    plt.show()

# Display POS counts for a sample AI-generated example
if not ai_examples.empty:
    plot_pos_counts_by_section(ai_examples.iloc[0], "AI-Generated Entry")

# Display POS counts for a sample human-written example
if not human_examples.empty:
    plot_pos_counts_by_section(human_examples.iloc[0], "Human-Written Entry")
```




## Summarize pos counts

```
from collections import Counter

aggregated_early_pos_counts = Counter()
aggregated_mid_pos_counts = Counter()
aggregated_later_pos_counts = Counter()

for index, row in df.iterrows():
    aggregated_early_pos_counts.update(row['early_pos_counts'])
    aggregated_mid_pos_counts.update(row['mid_pos_counts'])
    aggregated_later_pos_counts.update(row['later_pos_counts'])

print("Aggregated Early POS Counts:")
print(aggregated_early_pos_counts)
print("\nAggregated Mid POS Counts:")
print(aggregated_mid_pos_counts)
print("\nAggregated Later POS Counts:")
print(aggregated_later_pos_counts)
```

```
Aggregated Early POS Counts:
Counter({'NN': 44028, 'IN': 35230, 'DT': 30912, 'JJ': 20374, 'NNS': 18391, '.': 13786, 'VB': 12965, 'NNP': 12161, 'RB': 11946,

Aggregated Mid POS Counts:
Counter({'NN': 38751, 'IN': 31543, 'DT': 28897, 'NNS': 17924, 'JJ': 17576, 'RB': 13434, 'VB': 12539, '.': 12424, 'NNP': 11421,

Aggregated Later POS Counts:
Counter({'NN': 43349, 'IN': 33901, 'DT': 31546, 'JJ': 20236, 'NNS': 20008, 'VB': 16390, 'RB': 15098, '.': 13866, ',': 12312, 'P
```

Display POS counts for sample AI-generated and human text.

```
print("Aggregated POS Counts for Early Section:")
print(aggregated_early_pos_counts)
print("\nAggregated POS Counts for Mid Section:")
print(aggregated_mid_pos_counts)
print("\nAggregated POS Counts for Later Section:")
print(aggregated_later_pos_counts)
```

```
Aggregated POS Counts for Early Section:
Counter({'NN': 44028, 'IN': 35230, 'DT': 30912, 'JJ': 20374, 'NNS': 18391, '.': 13786, 'VB': 12965, 'NNP': 12161, 'RB': 11946,

Aggregated POS Counts for Mid Section:
Counter({'NN': 38751, 'IN': 31543, 'DT': 28897, 'NNS': 17924, 'JJ': 17576, 'RB': 13434, 'VB': 12539, '.': 12424, 'NNP': 11421,

Aggregated POS Counts for Later Section:
Counter({'NN': 43349, 'IN': 33901, 'DT': 31546, 'JJ': 20236, 'NNS': 20008, 'VB': 16390, 'RB': 15098, '.': 13866, ',': 12312, 'P
```

## Count total pos tags

```
import nltk
from nltk.tokenize import word_tokenize
from nltk import pos_tag

def count_total_pos_tags(text):
    if isinstance(text, str):
        tokens = word_tokenize(text)
        pos_tagged_tokens = pos_tag(tokens)
        total_pos_counts = {}
        for word, tag in pos_tagged_tokens:
            total_pos_counts[tag] = total_pos_counts.get(tag, 0) + 1
        return total_pos_counts
    return {}

df['total_pos_counts'] = df['text'].apply(count_total_pos_tags)
display(df.head())
```

| | text | generated | early_text | mid_text | later_text | early_pos_tags | mid_pos_tags | later_pos_tags | early_pos_counts |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Machine learning, a subset of artificial intel... | 1 | Machine learning, a subset of artificial intel... | At its core, machine learning enables computer... | As machine learning continues to advance, it b... | [(Machine, NN), (learning, NN), (,, ,), (a, DT... | [(At, IN), (its, PRP$), (core, NN), (,, ,), (m... | [(As, IN), (machine, NN), (learning, VBG), (co... | {'NN': 6, ',': 3, 'DT': 3, 'IN': 3, 'JJ': 2, '... |
| **1** | A decision tree, a prominent machine learning ... | 1 | A decision tree, a prominent machine learning ... | The algorithm evaluates input features at each... | However, decision trees may suffer from overfi... | [(A, DT), (decision, NN), (tree, NN), (,, ,), ... | [(The, DT), (algorithm, NN), (evaluates, VBZ),... | [(However, RB), (,, ,), (decision, NN), (trees... | {'DT': 3, 'NN': 7, ',': 4, 'JJ': 3, 'VBG': 1, ... |
| **2** | Education, a cornerstone of societal progress,... | 1 | Education, a cornerstone of societal progress,... | It encompasses formal and informal learning, e... | In the digital age, technology enhances educat... | [(Education, NN), (,, ,), (a, DT), (cornerston... | [(It, PRP), (encompasses, VBZ), (formal, JJ), ... | [(In, IN), (the, DT), (digital, JJ), (age, NN)... | {'NN': 5, ',': 3, 'DT': 3, 'IN': 3, 'JJ': 1, '... |
| **3** | Computers, the backbone of modern technology, ... | 1 | Computers, the backbone of modern technology, ... | From personal computing devices to massive dat... | Rapid advancements in processing power, memory... | [(Computers, NNS), (,, ,), (the, DT), (backbon... | [(From, IN), (personal, JJ), (computing, VBG),... | [(Rapid, JJ), (advancements, NNS), (in, IN), (... | {'NNS': 1, ',': 2, 'DT': 2, 'NN': 4, 'IN': 2, ... |
| **4** | Chess, a timeless game of strategy and intelle... | 1 | Chess, a timeless game of strategy and intelle... | Beyond its recreational appeal, chess fosters ... | As a metaphor for life's complexities, chess t... | [(Chess, NNP), (,, ,), (a, DT), (timeless, JJ)... | [(Beyond, IN), (its, PRP$), (recreational, JJ)... | [(As, IN), (a, DT), (metaphor, NN), (for, IN),... | {'NNP': 1, ',': 4, 'DT': 5, 'JJ': 7, 'NN': 9, ... |

## Prepare data for modeling

```
pos_counts_df = pd.json_normalize(df['total_pos_counts'])
pos_counts_df = pos_counts_df.fillna(0)

X = pos_counts_df
y = df['generated']

display(X.head())
display(y.head())
```

| | NN | , | DT | IN | JJ | VBZ | RB | VBN | VBG | NNS | ... | : | NNPS | RP | EX | WP | UH | WP$ | FW | PDT | $ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 26 | 12.0 | 6 | 12 | 9 | 5.0 | 1.0 | 1.0 | 6.0 | 11.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **1** | 23 | 11.0 | 12 | 12 | 12 | 5.0 | 1.0 | 1.0 | 7.0 | 16.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **2** | 21 | 16.0 | 6 | 9 | 19 | 6.0 | 1.0 | 0.0 | 7.0 | 14.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **3** | 26 | 15.0 | 5 | 13 | 11 | 2.0 | 2.0 | 3.0 | 4.0 | 11.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **4** | 28 | 13.0 | 10 | 12 | 16 | 2.0 | 4.0 | 0.0 | 8.0 | 13.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

5 rows × 42 columns

| | generated |
|---|---|
| **0** | 1 |
| **1** | 1 |
| **2** | 1 |
| **3** | 1 |
| **4** | 1 |

**dtype:** int64

## Split data

```
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)

print("Shape of X_train:", X_train.shape)
print("Shape of X_test:", X_test.shape)
print("Shape of y_train:", y_train.shape)
print("Shape of y_test:", y_test.shape)
```

```
Shape of X_train: (2200, 42)
Shape of X_test: (550, 42)
Shape of y_train: (2200,)
Shape of y_test: (550,)
```

## Train and evaluate models

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

models = {
    "Random Forest": RandomForestClassifier(random_state=42),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Logistic Regression": LogisticRegression(random_state=42, max_iter=1000)
}

accuracy_scores = {}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_scores[name] = accuracy

print("Model Accuracy Scores:")
for name, accuracy in accuracy_scores.items():
    print(f"{name}: {accuracy:.4f}")
```

```
Model Accuracy Scores:
Random Forest: 0.9945
Decision Tree: 0.9927
Logistic Regression: 0.9945
```

## Compare accuracies

```
print("Model Accuracy Scores:")
for name, accuracy in accuracy_scores.items():
    print(f"{name}: {accuracy:.4f}")
```