Down Data Analyst Home Assignment

Irwan Mulyawan

Private & Confidential

Agenda

1. Data Analysis & Classification for News Article
2. Distribution and Characteristics of different news categories
3. Trend or Insight in the dataset
4. Bonus (Train a basic classification model to categorize the news articles into World, Sports, Business, or Sci/Tech)
5. Experiment Design for Dating App Badge

1. Data Analysis & Classification for news Article

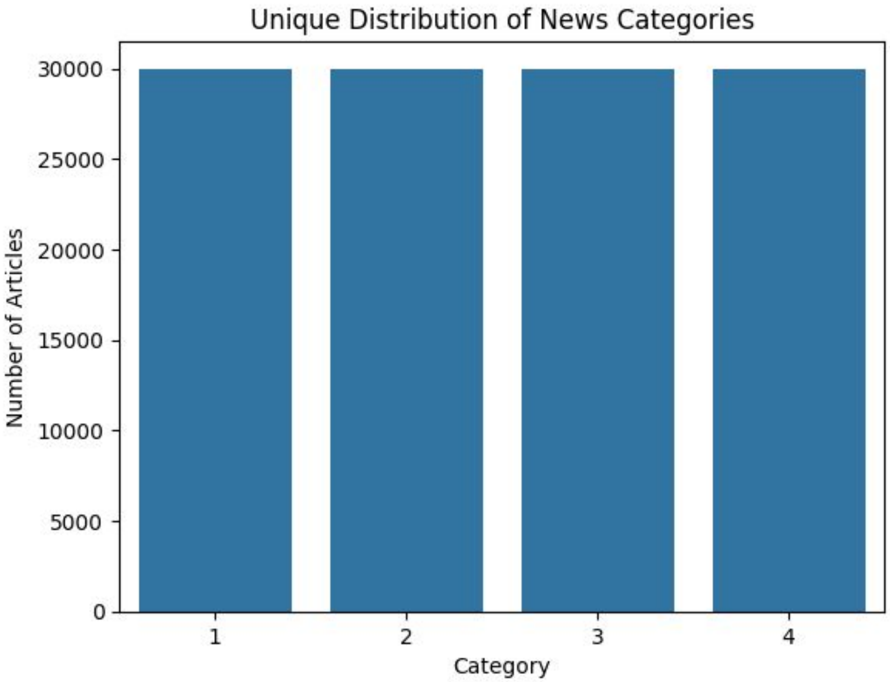
- Requirements:
  - Analysis
    - Perform exploratory data analysis to understand the distribution and characteristics of different news categories based on `train.csv`
    - Find any trend or insight in `train.csv` dataset.
      - Use statistical methods to validate findings.
      - Create visualizations to represent your insights.
      - You may use tools like Excel, SQL, or other data analysis tools, as well as Jupyter Notebook for some parts of the analysis.
  - Bonus (Optional but highly recommended)
    - Train a **basic** classification model to categorize the news articles into World, Sports, Business, or Sci/Tech.
      - How would you evaluate the model you train.
      - Use Jupyter Notebook for this part of the task.

A. Analysis

1. Distribution and Characteristics of different news categories
2. Find any trend or insight in the dataset

1. a. Distribution Analysis

Distribution of Each Categories

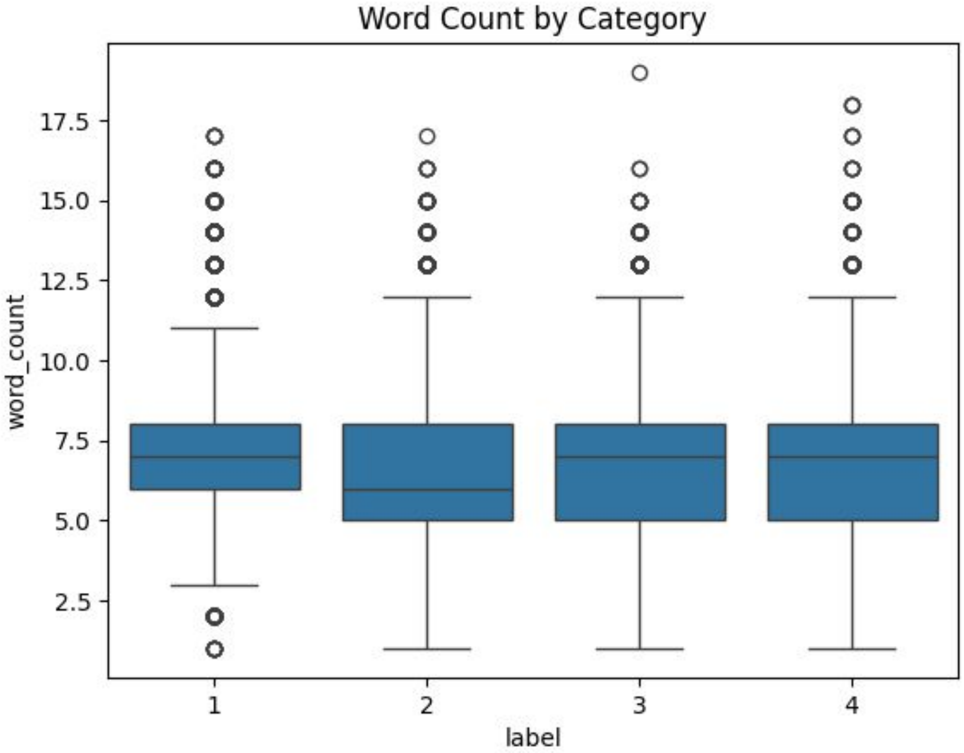


As you can see the number of Unique Articles distribution for each categories is the same, 30,000 articles

[Appendix] Python Code

```
print("Number of duplicate rows:", train.duplicated().sum())

# Remove duplicates
train_data_clean = train.drop_duplicates()

# Plot the count plot with the cleaned data
sns.countplot(x='label', data=train_data_clean)
plt.title('Unique Distribution of News Categories')
plt.xlabel('Category')
plt.ylabel('Number of Articles')
plt.show()
```

1. b. Characteristics Analysis



If we count the number of words per category and remove outliers from observations, then the first category has a symmetrical number of words between the minimum and maximum.

It could be said that the first category has similar characteristics in terms of the number of words compared to other three categories that more skew.

[Appendix] Python Code

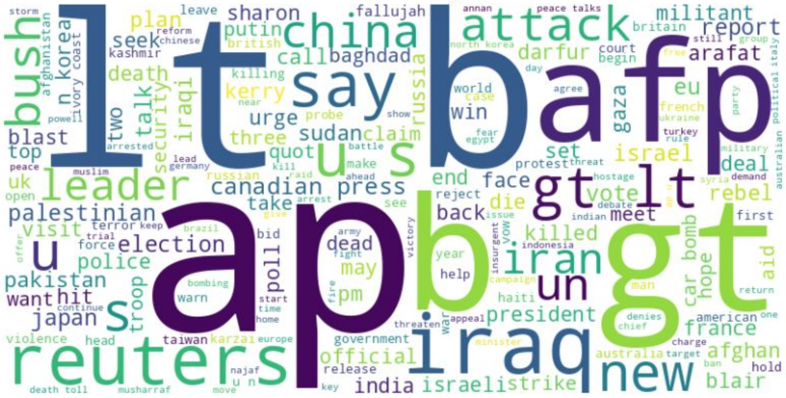
```
# Average word count per category
train['word_count'] = train['content'].apply(lambda x: len(x.split()))
sns.boxplot(x='label', y='word_count', data=train)
plt.title('Word Count by Category')
plt.show()
```

2. a. Word Cloud Analysis

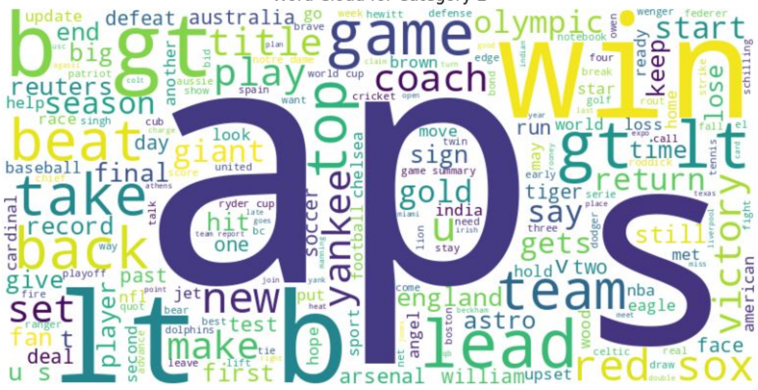
Trend Analysis

Investigate if there are common words or phrases within categories using a word cloud.

Word Cloud for Category 1



Word Cloud for Category 2

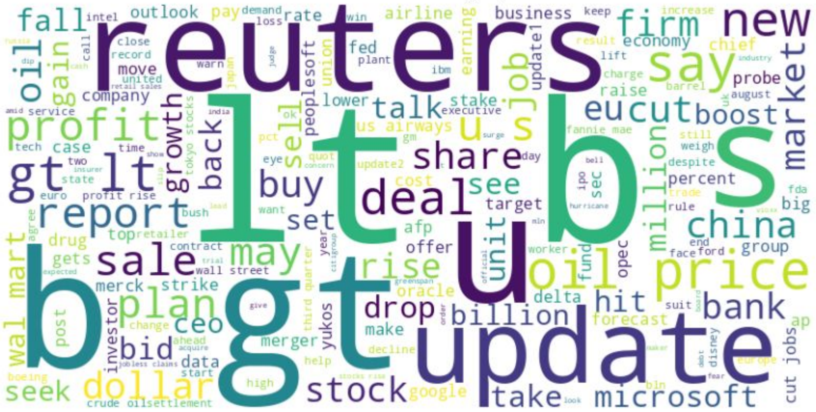


- If we look at category 1 and category 2, the news titles are dominated by AP media
- Category 1 is dominated by geopolitical news and category 2 is dominated by sports news

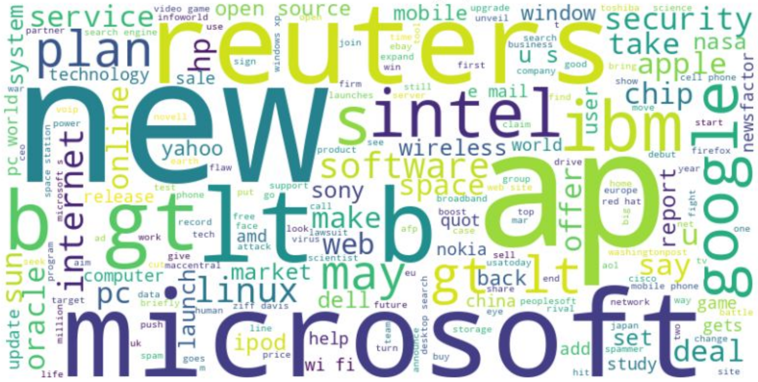
2. a. Word Cloud Analysis

Trend Analysis
Investigate if there are common words or phrases within categories using a word cloud.

Word Cloud for Category 3



Word Cloud for Category 4



- Category 3 is dominated by Reuters media, while Category 4 is dominated by Reuters, Microsoft and also AP media

2. a. Word Cloud Analysis

Trend Analysis
Media Conclusion

Category	Media
Category 1 (World)	AP
Category 2 (Sports)	AP
Category 3 (Business)	Reuters
Category 4 (Sci / Tech)	Reuters, Microsoft, AP

[Appendix] Python Code

```
from sklearn.feature_extraction.text import CountVectorizer
from nltk.corpus import stopwords
import nltk

# Ensure that nltk's resources are downloaded
nltk.download('stopwords')

data = pd.read_csv('train.csv', header=None, names=['label', 'content', 'detail'])

# Preprocessing steps
data['processed_content'] = data['content'].str.lower().str.replace(r'\W', ' ').str.replace(r'\s+', ' ')
stop_words = set(stopwords.words('english'))
data['processed_content'] = data['processed_content'].apply(lambda x: ' '.join([word for word in x.split() if word not in
stop_words]))

from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Define a function to plot word clouds
def plot_word_cloud(category):
    text = data[data['label'] == category]['processed_content'].str.cat(sep=' ')
    wordcloud = WordCloud(width=800, height=400, background_color ='white').generate(text)
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title(f'Word Cloud for Category {category}')
    plt.show()

# Plot word clouds for each category
for label in data['label'].unique():
    plot_word_cloud(label)
```

2. b. Top N-grams Analysis

Top 5 bi-grams Analysis

Investigate the top 5 phrases consisting of 2 words, to see what phrases are dominant in a certain category

Category 1

Phrases	Frequency
canadian press	451
ivory coast	171
car bomb	149
peace talks	144
death toll	134

Category 2

Phrases	Frequency
red sox	410
world cup	143
ryder cup	137
game summary	137
notre dame	113

2. b. Top N-grams Analysis

Top 5 bi-grams Analysis

Investigate the top 5 phrases consisting of 2 words, to see what phrases are dominant in a certain category

Category 3

Phrases	Frequency
oil prices	582
wal mart	230
us airways	225
wall street	155
profit rises	153

Category 4

Phrases	Frequency
open sources	223
wifi	202
pc world	191
washingtonpost com	189
desktop search	165

2. b. Top N-gram Analysis

Conclusion

Category 1 needs a further deep dive into the topics discussed by the Canadian press, while Category 2 mostly discusses about Red Sox, the basketball team from Boston.

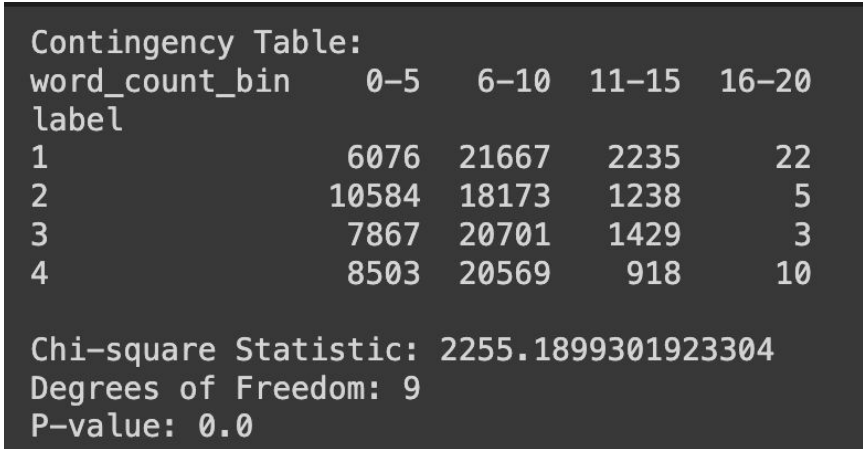
Category 3 majority discusses oil prices which is probably related to the increase in oil prices, while category 4 discusses open sources, software where users can develop themselves for free.

[Appendix] Python Code

```
# Function to extract top n-grams
def get_top_ngrams(corpus, n=None, ngrams=2):
    vec = CountVectorizer(ngram_range=(ngrams, ngrams)).fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq = sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]

# Example of extracting top bi-grams for each category
for label in data['label'].unique():
    common_phrases = get_top_ngrams(data[data['label'] == label]['processed_content'], n=10, ngrams=2)
    print(f"Top 10 bi-grams for Category {label}:")
    for phrase, freq in common_phrases:
        print(f"{phrase}: {freq}")
```

2. c. Statistical methods

Using statistical test (like chi-square test for categorical data) to validate if the differences observed in distributions are statistically significant.
Firstly, we use word count across category



```
Contingency Table:
word_count_bin    0-5   6-10  11-15  16-20
label
1                6076  21667   2235     22
2               10584  18173   1238      5
3                7867  20701   1429      3
4                8503  20569    918     10

Chi-square Statistic: 2255.1899301923304
Degrees of Freedom: 9
P-value: 0.0
```

We are using null hypothesis:
There is no association between the news category and the distribution of word counts.

Conclusion

The very small P-value effectively indicates that we can reject the null hypothesis with a high degree of confidence.

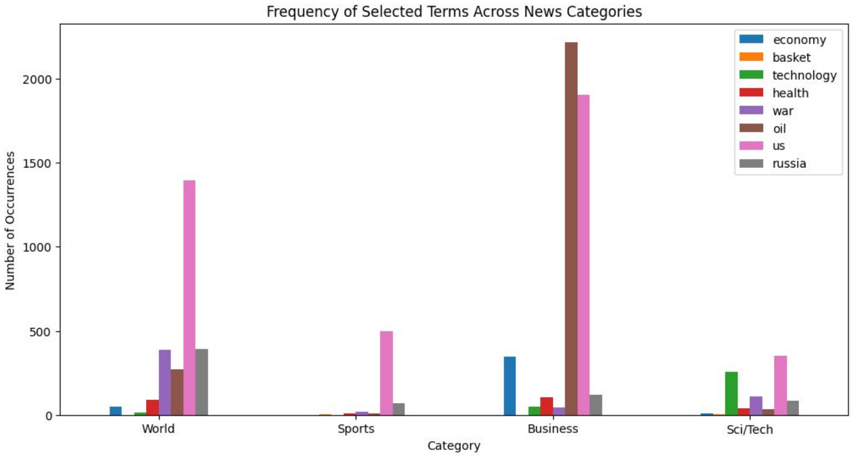
In summary, our results are strongly statistically significant and indicate a robust association between the news categories and the distribution of word counts or other features we've analyzed.

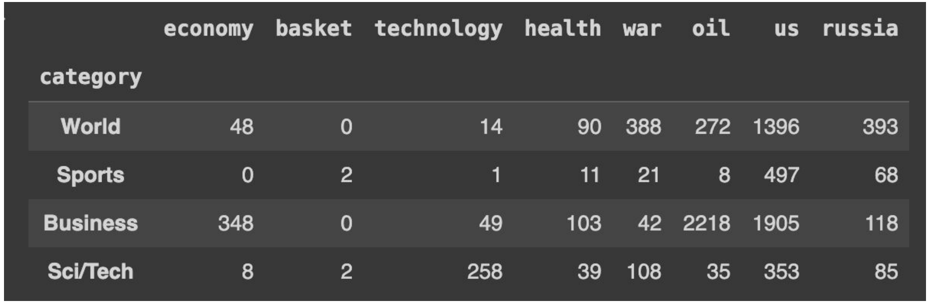
[Appendix] Python Code

```
import numpy as np

# Load the dataset
data = pd.read_csv('train.csv', header=None, names=['label', 'content', 'detail'])

# Process text to count words
data['word_count'] = data['content'].apply(lambda x: len(x.split()))

# Categorize word counts to make them definite
data['word_count_bin'] = pd.cut(data['word_count'], bins=[0, 100, 200, 300, 400, np.inf], labels=['0-100',
'101-200', '201-300', '301-400', '400+'])


from scipy.stats import chi2_contingency

# Create a contingency table
contingency_table = pd.crosstab(data['label'], data['word_count_bin'])

print("Contingency Table:")
print(contingency_table)

# Apply the Chi-square test
chi2, p, dof, expected = chi2_contingency(contingency_table)

print("\nChi-square Statistic:", chi2)
```

2. d. Visualization Findings (Additional)

We want to predict what common topics are discussed across all categories





	economy	basket	technology	health	war	oil	us	russia
category								
World	48	0	14	90	388	272	1396	393
Sports	0	2	1	11	21	8	497	68
Business	348	0	49	103	42	2218	1905	118
Sci/Tech	8	2	258	39	108	35	353	85

Based on analysis of the frequency of occurrence of words
As can be seen in the four categories, the words that appear most frequently are oil, us, and Russia

I predict this news appear when the Russian war occurs which will cause oil prices to increase

[Appendix] Python Code

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer

# Load your dataset
data = pd.read_csv('train.csv', header=None, names=['label', 'content','detail'])

# Terms you want to visualize
terms = ['economy', 'basket', 'technology', 'health', 'war', 'oil', 'us', 'russia']

# Create a document-term matrix
vectorizer = CountVectorizer(vocabulary=terms, binary=False)
dtm = vectorizer.fit_transform(data['content'].str.lower())  # Using str.lower to normalize the text

# Create a DataFrame for easier manipulation
dtm_df = pd.DataFrame(dtm.toarray(), columns=vectorizer.get_feature_names_out())
dtm_df['category'] = data['label']

# Group by category and sum occurrences
grouped = dtm_df.groupby('category').sum()
grouped = grouped.rename(index={1: 'World', 2: 'Sports', 3: 'Business', 4: 'Sci/Tech'})
grouped

import matplotlib.pyplot as plt

grouped.plot(kind='bar', figsize=(12, 6))
plt.title('Frequency of Selected Terms Across News Categories')
plt.xlabel('Category')
plt.ylabel('Number of Occurrences')
```

2. Conclusion for this question:
Find any trend or insight in the dataset

Word Cloud Analysis:

Category	Media
Category 1 (World)	AP
Category 2 (Sports)	AP
Category 3 (Business)	Reuters
Category 4 (Sci / Tech)	Reuters, Microsoft, AP

N-grams Analysis:

Conclusion

Category 1 needs a further deep dive into the topics discussed by the Canadian press, while Category 2 mostly discusses about Red Sox, the basketball team from Boston.

Category 3 majority discusses oil prices which is probably related to the increase in oil prices, while category 4 discusses open sources, software where users can develop themselves for free.

2. Conclusion for this question:
Find any trend or insight in the dataset

Statistical Method:

We are using null hypothesis:
There is no association between the news category and the distribution of word counts.

Conclusion

The very small P-value effectively indicates that we can reject the null hypothesis with a high degree of confidence.
In summary, our results are strongly statistically significant and indicate a robust association between the news categories and the distribution of word counts or other features we've analyzed.

Visualization Findings:

Based on analysis of the frequency of occurrence of words
As can be seen in the four categories, the words that appear most frequently are oil, us, and Russia

I predict this news appear when the Russian war occurs which will cause oil prices to increase

B. Bonus

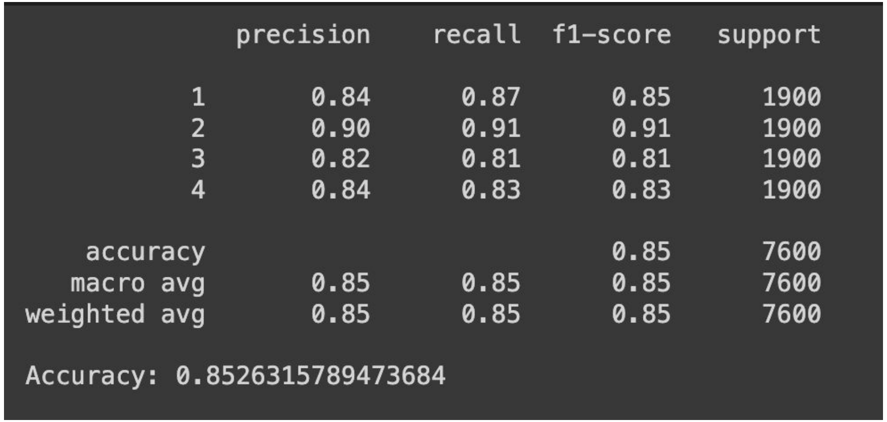
1 Train a basic classification model to categorize the news articles into World, Sports, Business, or Sci/Tech.

1.1 How would you evaluate the model you train

B. 1. Training Model

Training & Testing Result
Using Multinomial Naive Bayes, we can test our testing model with high accuracy.

Classifier is based on Bayes’ theorem, with “naive” assumption of conditional independence between every pair of features given the value of the class variable.
It assumes that the presence or absence of a particular feature is unrelated to the presence or absence of any other feature



```
              precision    recall  f1-score   support

           1       0.84      0.87      0.85      1900
           2       0.90      0.91      0.91      1900
           3       0.82      0.81      0.81      1900
           4       0.84      0.83      0.83      1900

    accuracy                           0.85      7600
   macro avg       0.85      0.85      0.85      7600
weighted avg       0.85      0.85      0.85      7600

Accuracy:  0.8526315789473684
```

Precision over 0.8 indicates that more than 80% of the articles our model labeled as belonging to a certain category were correctly classified. This is a strong indicator of our model's accuracy in making predictions.

Recall over 0.8 suggests that our model successfully identified more than 80% of all relevant articles within each category. This shows that the model is capable of detecting the majority of articles for each category.

F-score (or F1-score) also above 0.8 confirms a good balance between precision and recall, which is particularly important in scenarios where a trade-off between these metrics exists.

Accuracy:
An overall accuracy of 0.85 means that 85% of all predictions made across all categories were correct. This is a very solid performance, especially for initial model training phases.

[Appendix] Python Code

```
import pandas as pd
from sklearn.model_selection import train_test_split

# Load data
train_data = pd.read_csv('train.csv', header=None,
names=['label', 'content', 'detail'])
test_data = pd.read_csv('test.csv', header=None,
names=['label', 'content', 'detail'])

# Preview the data
print(train_data.head())

from sklearn.feature_extraction.text import TfidfVectorizer

# Text preprocessing and vectorization
vectorizer = TfidfVectorizer(stop_words='english',
max_features=10000)

X_train = vectorizer.fit_transform(train_data['content'])
y_train = train_data['label']
X_test = vectorizer.transform(test_data['content'])
y_test = test_data['label']
```

```
from sklearn.naive_bayes import MultinomialNB

# Initialize the classifier
model = MultinomialNB()

# Train the model
model.fit(X_train, y_train)

from sklearn.metrics import classification_report,
accuracy_score

# Predict on the test set
y_pred = model.predict(X_test)

# Print the classification report
print(classification_report(y_test, y_pred))
print("Accuracy:", accuracy_score(y_test, y_pred))
```

2. Experiment Design for Dating App Badge

1 Design an experiment for the following hypothesis for a gender-asymmetric feature for a generic dating app

1.1 If we let male users display a badge on their profiles, then they will get more matches with female users

Outline

1. Hypothesis
2. Cohorts
3. Duration and Sample Size
4. Events
5. Key Metrics
6. Reporting

2. 1. a. Hypothesis

Null Hypothesis (H0)

Displaying a badge on male users' profiles does not lead to a statistically significant increase in the number of matches with female users.

Alternative Hypothesis (H1)

Displaying a badge on male users' profiles leads to a statistically significant increase in the number of matches with female users.

2. 1. b. Cohorts

Treatment Groups

Treatment 1: Male users who are given the option to add a badge on their profiles
Treatment 2: Female users who can see the badge on male profiles

Control Groups

Control 1: Male users who are not given the option to add a badge
Control 2: Female users who cannot see any badge on male profiles.

Targeting for Enrollment:
Randomly select a subset of male and female users. Ensure that they are representative of the broader user base in terms of demographic and activity levels.
Use stratified sampling if necessary to maintain balance across key demographic such as age and location

2. 1. c. Duration and Sample Size

Sample Size Calculation

Use power analysis to determine the required sample size, aiming for a power of 0.80 and an alpha of 0.05, considering the effect size estimated from previous data or pilot projects
Use statistical software of online calculators specifying the expected difference in match rates, standard deviation, and the desired power level. such as: https://www.socscistatistics.com/tests/ or

Duration

Duration should be determined based on the average daily user activity and the calculated sample size. Run the experiment until the needed sample size is reached, which could be estimated through pilot data or historical app activity.

If we are using Google Optimize, that tools can directly give the information if experiment is good enough or need more running time

2. 1. d. Events

Data Collection

Profile View Event:

Log each time a user views another user’s profile, including timestamp, viewer’s gender, and viewed profile’s gender

Badge Display Event:

Record when a badge is displayed on a profile

Match Event:

Log each time two users like each other resulting in a match, including the presence or absence of the badge

2. 1. e. Key Metrics

Primary Metric:

Match rate for male users (number of matches per number of profile views)

Secondary Metrics:

Like rate (number of likes per profile view) for males with and without badges.

Engagement metrics such as time spent on the app and number of profiles chatted

Safety Metric:

User feedback scores and report of negative interactions to monitor any potential adverse effect of the badge feature

2. 1. f. Reporting

Analysis Plan:

- Conduct inferential statistical tests (e.g. Chi-squared test for match rates, t-test for continuous engagement metrics) to compare control and treatment groups
- Use regression analysis to adjust for covariates like user activity level and demographic

Communication:

- Prepare a detailed report including methodology, analysis results, and interpretations.
- Include visualizations such as bar charts and line graphs to show key metric comparisons between groups.
- Present findings to stakeholders in a formal presentation, outlining potential business impacts and recommendations based on the results.