```python
import pandas as pd
from google.colab import drive


drive.mount('/content/drive')


df_fake = pd.read_csv('/content/drive/MyDrive/Fake.csv')
df_real = pd.read_csv('/content/drive/MyDrive/True.csv')
df_fake['label'] = 0
df_real['label'] = 1


df = pd.concat([df_fake, df_real], ignore_index=True)


df = df[['title', 'text', 'label']]


df.dropna(subset=['text'], inplace=True)
df.reset_index(drop=True, inplace=True)
print("✅ Dataset loaded! Shape:", df.shape)
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer


nltk.download('stopwords')
nltk.download('wordnet')
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()


def clean_text(text):
    if isinstance(text, str):
        text = text.lower()
        text = re.sub(r'http\S+|www\S+|https\S+', '', text)
        text = re.sub(r'@\w+|#\w+', '', text)
        text = re.sub(r'[^a-zA-Z\s]', '', text)

        words = text.split()
        words = [lemmatizer.lemmatize(word) for word in words if word not in stop_words]
        return ' '.join(words)
    else:
        return ""


df['cleaned_text'] = df['text'].apply(clean_text)


print(df[['text', 'cleaned_text']].head(5))

df.to_csv('/content/drive/MyDrive/cleaned_fake_news_dataset.csv', index=False)

print("✅ Cleaned dataset saved successfully!")
import pandas as pd

label_counts = df['label'].value_counts()
print("Label Distribution:\n", label_counts)


imbalance_percentage = (label_counts[0] / label_counts.sum()) * 100
print(f"Imbalance Percentage: {imbalance_percentage:.2f}%")


df['label'].value_counts().plot(kind='bar', title='Distribution of Fake and Real News')

df['text_length'] = df['text'].apply(len)


print("Average text length for fake news:", df[df['label'] == 0]['text_length'].mean())
print("Average text length for real news:", df[df['label'] == 1]['text_length'].mean())


import matplotlib.pyplot as plt

df[df['label'] == 0]['text_length'].hist(alpha=0.5, label='Fake')
df[df['label'] == 1]['text_length'].hist(alpha=0.5, label='Real')
plt.legend()
plt.title('Distribution of News Story Length')
from collections import Counter

fake_text = ' '.join(df[df['label'] ==
0]['cleaned_text'].astype(str).tolist())
real_text = ' '.join(df[df['label'] ==
1]['cleaned_text'].astype(str).tolist())


fake_word_counts = Counter(fake_text.split())
real_word_counts = Counter(real_text.split())


top_fake_words = fake_word_counts.most_common(20)
top_real_words = real_word_counts.most_common(20)


print("Top 20 words in fake news:", top_fake_words)
print("Top 20 words in real news:", top_real_words)
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(max_features=5000)
vectorizer.fit(df['cleaned_text'])
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
from sklearn.feature_extraction.text import TfidfVectorizer


X = df_tfidf
y = df['label']

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

vectorizer = TfidfVectorizer(max_features=5000)
vectorizer.fit(df['cleaned_text'])


df_tfidf = vectorizer.transform(df['cleaned_text'])
```

```
X = df_tfidf
y = df['label']

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

logreg_model = LogisticRegression()
logreg_model.fit(X_train, y_train)

y_pred_logreg = logreg_model.predict(X_test)


accuracy_logreg = accuracy_score(y_test, y_pred_logreg)
report_logreg = classification_report(y_test, y_pred_logreg)

print("Logistic Regression Accuracy:", accuracy_logreg)
print("Logistic Regression Classification Report:\n", report_logreg)
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
!pip install pandas
import pandas as pd
from google.colab import drive


drive.mount('/content/drive')


df_fake = pd.read_csv('/content/drive/MyDrive/Fake.csv')
df_real = pd.read_csv('/content/drive/MyDrive/True.csv')


df_fake['label'] = 0
df_real['label'] = 1

df = pd.concat([df_fake, df_real], ignore_index=True)


df = df[['title', 'text', 'label']]

df.dropna(subset=['text'], inplace=True)
df.reset_index(drop=True, inplace=True)


label_counts = df['label'].value_counts()
print("Label Distribution:\n", label_counts)
import pandas as pd
from google.colab import drive
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from collections import Counter
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC


drive.mount('/content/drive')
import pandas as pd
from google.colab import drive
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from collections import Counter
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC



drive.mount('/content/drive')


df_fake = pd.read_csv('/content/drive/MyDrive/Fake.csv')
df.to_csv('/content/drive/MyDrive/cleaned_fake_news_dataset.csv',
index=False)

print("✅ Cleaned dataset saved successfully!")

import pandas as pd
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score


df =
pd.read_csv('/content/drive/MyDrive/cleaned_fake_news_dataset.csv')


X = df['cleaned_text']
y = df['label']


df.dropna(subset=['cleaned_text'], inplace=True)
df.reset_index(drop=True, inplace=True)
X = df['cleaned_text']
y = df['label']


vectorizer = TfidfVectorizer(max_features=5000)
X_vectorized = vectorizer.fit_transform(X)


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


model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)


y_pred = model.predict(X_test)


print("✅ Model Evaluation Results:")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test,
y_pred))
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS
from nltk.stem import PorterStemmer
```

```python
def preprocess(text):

    text = text.lower()
    text = ''.join([char for char in text if char.isalnum() or
char.isspace()])
    words = text.split()
    stemmer = PorterStemmer()
    return ' '.join([stemmer.stem(word) for word in words if word not
in ENGLISH_STOP_WORDS])

df['cleaned_text'] = df['cleaned_text'].apply(preprocess)
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, X_vectorized, y, cv=5)
print("Cross-validation scores:", scores)
print("Average cross-validation score:", scores.mean())
import joblib
joblib.dump(model, 'fake_news_model.pkl')

model = LogisticRegression(max_iter=1000, class_weight='balanced')
model.fit(X_train, y_train)


print(predict_fake_news("Example news text goes here."))
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix


conf_matrix = confusion_matrix(y_test, y_pred)


plt.figure(figsize=(6, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=["Fake", "Real"], yticklabels=["Fake", "Real"])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()

y_pred = model.predict(X_test)
```

```
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

print("✅ Model Testing Results:")
print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```